Package ‘magi’

June 22, 2024

Type Package
Title MAnifold-Constrained Gaussian Process Inference
Version 1.2.4
Date 2024-06-21
Encoding UTF-8

Description

URL https://doi.org/10.18637/jss.v109.i04
License MIT + file LICENSE

VignetteBuilder knitr

Imports Rcpp (>= 1.0.6), gridExtra, gridBase, grid, methods, deSolve

LinkingTo Rcpp, RcppArmadillo, BH, roptim

RoxygenNote 7.3.1
Suggests testthat, mvtnorm, covr, knitr, MASS, rmarkdown, markdown

Depends R (>= 3.6.0)

NeedsCompilation yes

Author Shihao Yang [aut, cre] (<https://orcid.org/0000-0003-3910-4969>), Samuel W.K. Wong [aut] (<https://orcid.org/0000-0002-7325-7267>), S.C. Kou [ctb, cph] (Contributor of MAGI method development)

Maintainer Shihao Yang <shihao.yang@isye.gatech.edu>

Repository CRAN

Date/Publication 2024-06-22 17:30:02 UTC
Contents

calCov .................................................. 2
FNdat ..................................................... 4
fnmodelODE ............................................. 5
gpcov .................................................... 6
gpmean .................................................... 7
gpsmoothing ............................................. 8
gpsmoothllik ........................................... 9
hes1modelODE .......................................... 10
is.magioutput .......................................... 11
magi ..................................................... 12
MagiPosterior ......................................... 13
MagiSolver .............................................. 14
plot.magioutput ........................................ 17
ptransmodelODE ....................................... 19
setDiscretization ..................................... 21
summary.magioutput ................................... 22
testDynamicalModel ................................... 23

Index .................................................... 25

calCov  Calculate stationary Gaussian process kernel

Description

Covariance calculations for Gaussian process kernels. Currently supports matern, rbf, compact1, periodicMatern, generalMatern, and rationalQuadratic kernels. Can also return m_phi and other additional quantities useful for ODE inference.

Usage

calCov(
  phi,
  rInput,
  signrInput,
  bandsize = NULL,
  complexity = 3,
  kerneltype = "matern",
  df,
  noiseInjection = 1e-07
)
Arguments

phi  the kernel hyper-parameters. See details for hyper-parameter specification for each kernel type.

rInput  the distance matrix between all time points s and t, i.e., |s - t|

signrInput  the sign matrix of the time differences, i.e., sign(s - t)

bandsize  size for band matrix approximation. See details.

complexity  integer value for the complexity of the kernel calculations desired:
  • 0 includes C only
  • 1 additionally includes Cprime, Cdoubleprime, dCdphi
  • 2 or above additionally includes Ceigen1over, CeigenVec, Cinv, mphi, Kphi, Keigen1over, KeigenVec, Kinv, mphiLeftHalf, dCdphiCube

See details for their definitions.

kerneltype  must be one of matern, rbf, compact1, periodicMatern, generalMatern, rationalQuadratic. See details for the kernel formulae.

df  degrees of freedom, for generalMatern and rationalQuadratic kernels only. Default is df=2.01 for generalMatern and df=0.01 for rationalQuadratic.

noiseInjection  a small value added to the diagonal elements of C and Kphi for numerical stability

Details

The covariance formulae and the hyper-parameters phi for the supported kernels are as follows. Stationary kernels have $C(s, t) = C(r)$ where $r = |s - t|$ is the distance between the two time points. Generally, the hyper-parameter phi[1] controls the overall variance level while phi[2] controls the bandwidth.

matern  This is the simplified Matern covariance with $df = 5/2$:

$$C(r) = \phi[1]*(1 + \sqrt{5}r/\phi[2] + 5r^2/(3\phi[2]^2)) * \exp(-\sqrt{5}r/\phi[2])$$

rbf

$$C(r) = \phi[1] * \exp(-r^2/(2\phi[2]^2))$$

compact1

$$C(r) = \phi[1] * \max(1 - r/\phi[2], 0)^4 * (4r/\phi[2] + 1)$$

periodicMatern  Define $r' = |\sin(r\pi/\phi[3])| * 2$. Then the covariance is given by $C(r')$ using the Matern formula.

generalMatern

$$C(r) = \phi[1] * 2^{(1-df)} / \Gamma(df) * (\sqrt{2(0+df)r/\phi[2]})^df * \text{besselK}(\sqrt{2(0+df)r/\phi[2]}, df)$$

where besselK is the modified Bessel function of the second kind.

rationalQuadratic

$$C(r) = \phi[1] * (1 + r^2/(2df\phi[2]^2))^d - df$$

The kernel calculations available and their definitions are as follows:
C The covariance matrix corresponding to the distance matrix rInput.

Cprime The cross-covariance matrix $dC(s, t)/ds$.

Cdoubleprime The cross-covariance matrix $d^2C(s, t)/dsdt$.

dCdphi A list with the matrices $dC/d\phi$ for each element of $\phi$.

Ceigen1over The reciprocals of the eigenvalues of $C$.

CeigenVec Matrix of eigenvectors of $C$.

Cinv The inverse of $C$.

mphi The matrix $Cprime \times Cinv$.

Kphi The matrix $Cdoubleprime - Cprime \times Kinv \times t(Cprime)$.

Keigen1over The reciprocals of the eigenvalues of $Kphi$.

Kinv The inverse of $Kphi$.

mphiLeftHalf The matrix $Cprime \times CeigenVec$.

dCdphiCube $dC/d\phi$ as a 3-D array, with the third dimension corresponding to the elements of $\phi$.

If bandsize is a positive integer, additionally CinvBand, mphiBand, and KinvBand are provided in the return list, which are band matrix approximations to Cinv, mphi, and Kinv with the specified bandsize.

Value

A list containing the kernel calculations included by the value of complexity.

Examples

foo <- outer(0:40, t(0:40), '-')[, 1, ]
r <- abs(foo)
signr <- -sign(foo)
calCov(c(0.2, 2), r, signr, bandsize = 20, kerneltype = "generalMatern", df = 2.01)

| FNdat | Dataset of noisy observations from the FitzHugh-Nagumo (FN) equations |

Description

The classic FN equations model the spike potentials of neurons, where system components $V$ and $R$ are the voltage and recovery variables, respectively.

$V$ and $R$ are governed by the following differential equations:

$$\frac{dV}{dt} = c(V - \frac{V^3}{3} + R)$$
\[
\frac{dR}{dt} = -\frac{1}{c}(V - a + bR)
\]

where \(\theta = (a, b, c)\) are system parameters. This dataset was generated by first numerically solving these ODEs from \(t = 0\) to \(t = 20\), with initial conditions \(V(0) = -1\) and \(R(0) = 1\) and parameters \(\theta = (0.2, 0.2, 3)\). The system components were taken to be measured at 28 observation time points (as indicated in time column) with additive Gaussian noise (standard deviation 0.2).

**Usage**

```r
data(FNdat)
```

**Format**

A data frame with 28 rows and 3 columns (time, \(V\), \(R\)).

**References**


---

**fnmodelODE**

*The FitzHugh-Nagumo (FN) equations*

**Description**

The classic FN equations model the spike potentials of neurons, where system components \(X = (V, R)\) represent the voltage and recovery variables, respectively.

\(V\) and \(R\) are governed by the following differential equations:

\[
\frac{dV}{dt} = c(V - \frac{V^3}{3} + R)
\]

\[
\frac{dR}{dt} = -\frac{1}{c}(V - a + bR)
\]

where \(\theta = (a, b, c)\) are system parameters.

**Usage**

```r
fnmodelODE(theta, x, tvec)
```

```r
fnmodelDx(theta, x, tvec)
```

```r
fnmodelDtheta(theta, x, tvec)
```
Arguments

- \( \theta \) vector of parameters.
- \( x \) matrix of system states (one per column) at the time points in \( tvec \).
- \( tvec \) vector of time points

Value

- \( \text{fnmodelODE} \) returns an array with the values of the derivatives \( \dot{X} \).
- \( \text{fnmodelDx} \) returns a 3-D array with the values of the gradients with respect to \( X \).
- \( \text{fnmodelDtheta} \) returns a 3-D array with the values of the gradients with respect to \( \theta \).

References


Examples

```r
theta <- c(0.2, 0.2, 3)
x <- matrix(1:10, nrow = 5, ncol = 2)
tvec <- 1:5

fnmodelODE(theta, x, tvec)
```

---

### gpcov

**Conditional covariance of Gaussian process given observations**

**Description**

Compute the conditional covariance of a Gaussian process, given a vector of observations, hyper-parameters \( \phi \), and noise standard deviation \( \sigma \).

**Usage**

```r
gpcov(yobs, tvec, tnew, phi, sigma, kerneltype = "generalMatern")
```

**Arguments**

- \( yobs \) vector of observations
- \( tvec \) vector of time points corresponding to observations
- \( tnew \) vector of time points at which the conditional covariance should be computed
- \( \phi \) vector of hyper-parameters for the covariance kernel (kerneltype)
- \( \sigma \) the noise level (if known). By default, both \( \phi \) and \( \sigma \) are estimated. If a value for \( \sigma \) is supplied, then \( \sigma \) is held fixed at the supplied value and only \( \phi \) is estimated.
- \( \text{kerneltype} \) the covariance kernel, types matern, rbf, compact1, periodicMatern, generalMatern are supported. See \( \text{calCov} \) for their definitions.
Value

The conditional covariance matrix for the GP evaluated at the time points in tnew.

Examples

```r
# Load Fitzhugh-Nagumo dataset
data(FNdat)

tnew <- seq(15, 20, by = 0.5)

# GP covariance of V component at time points in tnew given observations
gpcov(FNdat$V, FNdat$time, tnew, c(2.3, 1.2), 0.2)
```

Description

Compute the conditional mean of a Gaussian process (and optionally, its derivative), given a vector of observations, hyper-parameters phi, and noise standard deviation sigma.

Usage

```r
gpmean(
  yobs, tvec, tnew, phi, sigma, kerneltype = "generalMatern", deriv = FALSE
)
```

Arguments

- `yobs`: vector of observations
- `tvec`: vector of time points corresponding to observations
- `tnew`: vector of time points at which the conditional mean should be computed
- `phi`: vector of hyper-parameters for the covariance kernel (kerneltype)
- `sigma`: the noise level (if known). By default, both phi and sigma are estimated. If a value for sigma is supplied, then sigma is held fixed at the supplied value and only phi is estimated.
- `kerneltype`: the covariance kernel, types `matern`, `rbf`, `compact1`, `periodicMatern`, `generalMatern` are supported. See `calCov` for their definitions.
- `deriv`: logical; if true, the conditional mean of the GP’s derivative is also computed
**Value**

A vector with the values of the conditional mean function evaluated at the time points in `tnew`. If `deriv = TRUE`, returned with an additional attribute `deriv` that contains the values of the conditional mean of the GP derivative evaluated at the time points in `tnew`.

**Examples**

```r
# Load Fitzhugh-Nagumo dataset
data(FNdat)
tnew <- seq(0, 20, by = 0.5)

# GP mean of V component at time points in tnew given observations
gpmean(FNdat$V, FNdat$time, tnew, c(2.3, 1.2), 0.2)
```

---

**gpsmoothing**

_Gaussian process smoothing_

**Description**

Estimate hyper-parameters phi and noise standard deviation sigma for a vector of observations using Gaussian process smoothing.

**Usage**

```r
gpsmoothing(yobs, tvec, kerneltype = "generalMatern", sigma = NULL)
```

**Arguments**

- `yobs`: vector of observations
- `tvec`: vector of time points corresponding to observations
- `kerneltype`: the covariance kernel, types `matern`, `rbf`, `compact1`, `periodicMatern`, `generalMatern` are supported. See `calCov` for their definitions.
- `sigma`: the noise level (if known). By default, both phi and sigma are estimated. If a value for `sigma` is supplied, then `sigma` is held fixed at the supplied value and only phi is estimated.

**Value**

A list containing the elements phi and sigma with their estimated values.
Examples

# Sample data and observation times
tvec <- seq(0, 20, by = 0.5)
y <- c(-1.16, -0.18, 1.57, 1.99, 1.95, 1.85, 1.49, 1.58, 1.47, 0.96,
0.75, 0.22, -1.34, -2.11, -1.56, -1.51, -1.29, -1.22,
-0.36, 1.78, 2.36, 1.78, 1.8, 1.76, 1.4, 1.02, 1.28, 1.21, 0.04,
-1.35, -2.1, -1.9, -1.49, -1.55, -1.35, -0.98, -0.34, 1.9, 1.99, 1.84)
gpsmoothing(y, tvec)

description

Marginal log-likelihood and gradient as a function of GP hyper-parameters phi and observation noise standard deviation sigma. For use in Gaussian process smoothing where values of phi and sigma may be optimized.

Usage

gpsmoothllik(phisig, yobs, rInput, kerneltype = "generalMatern")

Arguments

- **phisig** vector containing GP hyper-parameters phi and observation noise SD sigma. See `calCov` for the definitions of the hyper-parameters.
- **yobs** vector of observations
- **rInput** distance matrix between all time points of **yobs**
- **kerneltype** the covariance kernel, types matern, rbf, compact1, periodicMatern, generalMatern are supported. See `calCov` for their definitions.

Value

A list with elements value and grad, which are the log-likelihood value and gradient with respect to **phisig**, respectively.

Examples

# Suppose phi[1] = 0.5, phi[2] = 3, sigma = 0.1
gpsmoothllik(c(0.5, 3, 0.1), rnorm(10), abs(outer(0:9, t(0:9), "-"))[1, 1]))
Hes1 equations: oscillation of mRNA and protein levels

Description

The Hes1 equations model the oscillatory cycles of protein and messenger ribonucleic acid (mRNA) levels in cultured cells. The system components \( X = (P, M, H) \) represent the concentrations of protein, mRNA, and the Hes1-interacting factor that provides a negative feedback loop. 

\( P, M, \) and \( H \) are governed by the following differential equations:

\[
\frac{dP}{dt} = -aPH + bM - cP
\]

\[
\frac{dM}{dt} = -d_M M + \frac{e}{1 + P^2}
\]

\[
\frac{dH}{dt} = -aPH + \frac{f}{1 + P^2} - gH
\]

where \( \theta = (a, b, c, d_M, e, f, g) \) are system parameters.

Usage

\texttt{hes1modelODE(theta, x, tvec)}

\texttt{hes1modelDx(theta, x, tvec)}

\texttt{hes1modelDtheta(theta, x, tvec)}

\texttt{hes1logmodelODE(theta, x, tvec)}

\texttt{hes1logmodelDx(theta, x, tvec)}

\texttt{hes1logmodelDtheta(theta, x, tvec)}

Arguments

\texttt{theta} vector of parameters.

\texttt{x} matrix of system states (one per column) at the time points in \texttt{tvec}.

\texttt{tvec} vector of time points

Value

\texttt{hes1modelODE} returns an array with the values of the derivatives \( \dot{X} \).

\texttt{hes1modelDx} returns a 3-D array with the values of the gradients with respect to \( X \).

\texttt{hes1modelDtheta} returns a 3-D array with the values of the gradients with respect to \( \theta \).

\texttt{hes1logmodelODE}, \texttt{hes1logmodelDx}, and \texttt{hes1logmodelDtheta} are the log-transformed versions of \texttt{hes1modelODE}, \texttt{hes1modelDx}, and \texttt{hes1modelDtheta}, respectively.
References


Examples

\[
\theta <- c(0.022, 0.3, 0.031, 0.028, 0.5, 20, 0.3) \\
x <- matrix(1:15, nrow = 5, ncol = 3) \\
tvec <- 1:5 \\

hes1modelODE(theta, x, tvec)
\]

is.magioutput Description

Check for and create a magioutput object

Usage

is.magioutput(object)

magioutput(...)

Arguments

object an R object

... arguments required to create a magioutput object. See details.

Details

Using the core MagiSolver function returns a magioutput object as output, which is a list that contains the following elements:

theta matrix of MCMC samples for the system parameters \( \theta \), after burn-in.
xsampled array of MCMC samples for the system trajectories at each discretization time point, after burn-in.

sigma matrix of MCMC samples for the observation noise SDs \( \sigma \), after burn-in.

phi matrix of estimated GP hyper-parameters, one column for each system component.

lp vector of log-posterior values at each MCMC iteration, after burn-in.

y, tvec, odeModel from the inputs to MagiSolver.

Printing a magioutput object displays a brief summary of the settings used for the MagiSolver run. The summary method for a magioutput object prints a table of parameter estimates, see summary.magioutput for more details. Plotting a magioutput object by default shows the inferred trajectories for each component, see plot.magioutput for more details.
Value

logical. Is the input a magioutput object?

Examples

# Set up odeModel list for the Fitzhugh-Nagumo equations
fnmodel <- list(
  fOde = fnmodelODE,
  fOdeDx = fnmodelDx,
  fOdeDtheta = fnmodelDtheta,
  thetaLowerBound = c(0, 0, 0),
  thetaUpperBound = c(Inf, Inf, Inf)
)

# Example FN data
data(FNdat)

# Create magioutput from a short MagiSolver run (demo only, more iterations needed for convergence)
result <- MagiSolver(FNdat, fnmodel, control = list(nstepsHmc = 5, niterHmc = 50))
is.magioutput(result)

\---

**magi**

**magi**: MAnifold-Constrained Gaussian Process Inference

Description

magi is a package that provides fast and accurate inference for the parameter estimation problem in Ordinary Differential Equations, including the case when there are unobserved system components. In the references below, please see our software paper Wong, Yang, and Kou (2024) for a detailed user guide and Yang, Wong, and Kou (2021) for details of the MAGI method (MAnifold-constrained Gaussian process Inference).

References


**MagiPosterior**

**MAGI posterior density**

**Description**

Computes the MAGI log-posterior value and gradient for an ODE model with the given inputs: the observations \( Y \), the latent system trajectories \( X \), the parameters \( \theta \), the noise standard deviations \( \sigma \), and covariance kernels.

**Usage**

```r
MagiPosterior(
  y, xlatent, theta, sigma,
  covAllDimInput,
  odeModel,
  priorTemperatureInput = 1,
  useBand = FALSE
)
```

**Arguments**

- **y**: data matrix of observations
- **xlatent**: matrix of system trajectory values
- **theta**: vector of parameter values \( \theta \)
- **sigma**: vector of observation noise for each system component
- **covAllDimInput**: list of covariance kernel objects for each system component. Covariance calculations may be carried out with `calCov`.
- **odeModel**: list of ODE functions and inputs. See details.
- **priorTemperatureInput**: vector of tempering factors for the GP prior, derivatives, and observations, in that order. Controls the influence of the GP prior relative to the likelihood. Recommended values: the total number of observations divided by the total number of discretization points for the GP prior and derivatives, and 1 for the observations.
- **useBand**: logical: should the band matrix approximation be used? If `TRUE`, `covAllDimInput` must include `CinvBand`, `mphiBand`, and `KinvBand` as computed by `calCov`.

**Value**

A list with elements `value` for the value of the log-posterior density and `grad` for its gradient.
Examples

# Trajectories from the Fitzhugh-Nagumo equations
tvec <- seq(0, 20, 2)
Vtrue <- c(-1, 1.91, 1.38, -1.32, -1.5, 1.73, 1.66, 0.89, -1.82, -0.93, 1.89)
Rtrue <- c(1, 0.33, -0.62, -0.82, 0.5, 0.94, -0.22, -0.9, -0.08, 0.95, 0.3)

# Noisy observations
Vobs <- Vtrue + rnorm(length(tvec), sd = 0.05)
Rob <- Rtrue + rnorm(length(tvec), sd = 0.1)

# Prepare distance matrix for covariance kernel calculation
foo <- outer(tvec, t(tvec), '-')[, 1]
r <- abs(foo)
r2 <- r^2
signr <- -sign(foo)

# Choose some hyperparameter values to illustrate
rphi <- c(0.95, 3.27)
vphi <- c(1.98, 1.12)
phiTest <- cbind(vphi, rphi)

# Covariance computations
curCovV <- calCov(phiTest[,1], r, signr, kerneltype = "generalMatern")
curCovR <- calCov(phiTest[,2], r, signr, kerneltype = "generalMatern")

# Y and X inputs to MagiPosterior
yInput <- data.matrix(cbind(Vobs, Robs))
xlatentTest <- data.matrix(cbind(Vtrue, Rtrue))

# Create odeModel list for FN equations
fnmodel <- list(
  fOde = fnmodelODE,
  fOdeDx = fnmodelDx,
  fOdeDtheta = fnmodelDtheta,
  thetaLowerBound = c(0, 0, 0),
  thetaUpperBound = c(Inf, Inf, Inf)
)

MagiPosterior(yInput, xlatentTest, theta = c(0.2, 0.2, 3), sigma = c(0.05, 0.1),
list(curCovV, curCovR), fnmodel)

---

MagiSolver

MAnifold-constrained Gaussian process Inference (MAGI)

Description

Core function of the MAGI method for inferring the parameters and trajectories of dynamic systems governed by ordinary differential equations.
Usage

MagiSolver(y, odeModel, tvec, control = list())

Arguments

y         data matrix of observations
odeModel  list of ODE functions and inputs. See details.
tvec      vector of discretization time points corresponding to rows of y. If missing, MagiSolver will use the column named ‘time’ in y.

Details

The data matrix y has a column for each system component, and optionally a column ‘time’ with the discretization time points. If the column ‘time’ is not provided in y, a vector of time points must be provided via the tvec argument. The rows of y correspond to the discretization set J at which the GP is constrained to the derivatives of the ODE system. To set the desired discretization level for inference, use setDiscretization to prepare the data matrix for input into MagiSolver. Missing observations are indicated with NA or NaN.

The list odeModel is used for specification of the ODE system and its parameters. It must include five elements:

fOde  function that computes the ODEs, specified with the form f(theta, x, tvec). fOde should return a matrix where columns correspond to the system components of x, see examples.
fOdeDx function that computes the gradients of the ODEs with respect to the system components.
fOdeDx should return a 3-D array, where the slice [, i, j] is the partial derivative of the ODE for the j-th system component with respect to the i-th system component, see examples.
fOdeDtheta function that computes the gradients of the ODEs with respect to the parameters theta.
fOdeDtheta should return a 3-D array, where the slice [, i, j] is the partial derivative of the ODE for the j-th system component with respect to the i-th parameter in theta, see examples.

Additional control variables can be supplied to MagiSolver via the optional list control, which may include the following:

sigma a vector of noise levels (observation noise standard deviations) σ for each component, at which to initialize MCMC sampling. By default, MagiSolver computes starting values for sigma via Gaussian process (GP) smoothing. If the noise levels are known, specify sigma together with useFixedSigma = TRUE.

phi  a matrix of GP hyper-parameters for each component, with rows for the kernel hyper-parameters and columns for the system components. By default, MagiSolver estimates phi via an optimization routine.
theta a vector of starting values for the parameters $\theta$, at which to initialize MCMC sampling. By default, MagiSolver uses an optimization routine to obtain starting values.

xInit a matrix of values for the system trajectories of the same dimension as $y$, at which to initialize MCMC sampling. Default is linear interpolation between the observed (non-missing) values of $y$ and an optimization routine for entirely unobserved components of $y$.

mu a matrix of values for the mean function of the GP prior, of the same dimension as $y$. Default is a zero mean function.

dotmu a matrix of values for the derivatives of the GP prior mean function, of the same dimension as $y$. Default is zero.

priorTemperature the tempering factor by which to divide the contribution of the GP prior, to control the influence of the GP prior relative to the likelihood. Default is the total number of observations divided by the total number of discretization points.

niterHmc MCMC sampling from the posterior is carried out via the Hamiltonian Monte Carlo (HMC) algorithm. niterHmc specifies the number of HMC iterations to run. Default is 20000 HMC iterations.

nstepsHmc the number of leapfrog steps per HMC iteration. Default is 200.

burninRatio the proportion of HMC iterations to be discarded as burn-in. Default is 0.5, which discards the first half of the MCMC samples.

stepSizeFactor initial leapfrog step size factor for HMC. Can be a specified as a scalar (applied to all posterior dimensions) or a vector (with length corresponding to the dimension of the posterior). Default is 0.01, and the leapfrog step size is automatically tuned during burn-in to achieve an acceptance rate between 60-90%.

bandSize a band matrix approximation is used to speed up matrix operations, with default band size 20. Can be increased if MagiSolver returns an error indicating numerical instability.

useFixedSigma logical, set to TRUE if sigma is known. If useFixedSigma = TRUE, the known values of $\sigma$ must be supplied via the sigma control variable. Default is FALSE.

kerneltype the GP covariance kernel, generalMatern is the default and recommended choice. Other available choices are matern, rbf, compact1, periodicMatern. See calCov for their definitions.

skipMissingComponentOptimization logical, set to TRUE to skip automatic optimization for missing components. If skipMissingComponentOptimization = TRUE, values for xInit and phi must be supplied for all system components. Default is FALSE.

positiveSystem logical, set to TRUE if the system cannot be negative. Default is FALSE.

verbose logical, set to TRUE to output diagnostic and progress messages to the console. Default is FALSE.

Value

MagiSolver returns an object of class magioutput which contains the following elements:

theta matrix of MCMC samples for the system parameters $\theta$, after burn-in.

xsampled array of MCMC samples for the system trajectories at each discretization time point, after burn-in.

sigma matrix of MCMC samples for the observation noise SDs $\sigma$, after burn-in.
phi  matrix of estimated GP hyper-parameters, one column for each system component.
lp  vector of log-posterior values at each MCMC iteration, after burn-in.
y, tvec, odeModel  from the inputs to MagiSolver.

References


Examples

```r
# Set up odeModel list for the Fitzhugh-Nagumo equations
fnmodel <- list(
  fOde = fnmodelODE,
  fOdeDx = fnmodelDx,
  fOdeDtheta = fnmodelDtheta,
  thetaLowerBound = c(0, 0, 0),
  thetaUpperBound = c(Inf, Inf, Inf)
)

# Example noisy data observed from the FN system
data(FNdat)

# Set discretization for a total of 81 equally-spaced time points from 0 to 20
yinput <- setDiscretization(FNdat, by = 0.25)

# Run MagiSolver
# Short sampler run for demo only, more iterations needed for convergence
MagiSolver(yinput, fnmodel, control = list(nstepsHmc = 5, niterHmc = 101))

# Use 3000 HMC iterations with 100 leapfrog steps per iteration
FNres <- MagiSolver(yinput, fnmodel, control = list(nstepsHmc = 100, niterHmc = 3000))

# Summary of parameter estimates
summary(FNres)

# Plot of inferred trajectories
plot(FNres, comp.names = c("V", "R"), xlab = "Time", ylab = "Level")
```

plot.magiotput  Generate plots from magiotput object

Description

Plots inferred system trajectories or diagnostic traceplots from the output of MagiSolver
Usage

## S3 method for class 'magioutput'

plot(
  x,
  type = "traj",
  obs = TRUE,
  ci = TRUE,
  ci.col = "skyblue",
  comp.names,
  par.names,
  est = "mean",
  lower = 0.025,
  upper = 0.975,
  sigma = FALSE,
  lp = TRUE,
  nplotcol = 3,
  ...
)

Arguments

x:
  a magioutput object.

type:
  string; the default type = "traj" plots inferred trajectories, while setting type = "trace" generates diagnostic traceplots for the MCMC samples of the parameters and log-posterior values.

obs:
  logical; if true, points will be added on the plots for the observations when type = "traj".

 ci:
  logical; if true, credible bands/intervals will be added to the plots.

 ci.col:
  string; color to use for credible bands.

 comp.names:
  vector of system component names, when type = "traj". If provided, should be the same length as the number of system components in X.

 par.names:
  vector of parameter names, when type = "trace". If provided, should be the same length as the number of parameters in \( \theta \), or the combined length of \( \theta \) and \( \sigma \) when sigma = TRUE.

 est:
  string specifying the posterior quantity to plot as the estimate. Can be "mean", "median", "mode", or "none". Default is "mean", which plots the posterior mean of the MCMC samples.

 lower:
  the lower quantile of the credible band/interval, default is 0.025. Only used if ci = TRUE.

 upper:
  the upper quantile of the credible band/interval, default is 0.975. Only used if ci = TRUE.

 sigma:
  logical; if true, the noise levels \( \sigma \) will be included in the traceplots when type = "trace".

 lp:
  logical; if true, the values of the log-posterior will be included in the traceplots when type = "trace".
nplotcol  the number of subplots per row.
... additional arguments to plot.

Details

Plots the inferred system trajectories (when type = "traj") or diagnostic traceplots of the parameters and log-posterior (when type = "trace") from the MCMC samples. By default, the posterior mean is treated as the estimate of the trajectories and parameters (est = "mean"). Alternatives are the posterior median (est = "median", taken component-wise) and the posterior mode (est = "mode", approximated by the MCMC sample with the highest log-posterior value).

The default type = "traj" produces plots of the inferred trajectories and credible bands from the MCMC samples, one subplot for each system component. By default, lower = 0.025 and upper = 0.975 produces a central 95% credible band when ci = TRUE. Adding the observed data points (obs = TRUE) can provide a visual assessment of the inferred trajectories.

Setting type = "trace" generates diagnostic traceplots for the MCMC samples of the system parameters and the values of the log-posterior, which is a useful tool for informally assessing convergence. In this case, the est and ci options add horizontal lines to the plots that indicate the estimate (in red) and credible interval (in green) for each parameter.

Examples

```r
# Set up odeModel list for the Fitzhugh-Nagumo equations
fnmodel <- list(
  fOde = fnmodelODE,
  fOdeDx = fnmodelDx,
  fOdeDtheta = fnmodelDtheta,
  thetaLowerBound = c(0, 0, 0),
  thetaUpperBound = c(Inf, Inf, Inf)
)

# Example FN data
data(FNdat)
y <- setDiscretization(FNdat, by = 0.25)

# Create magioutput from a short MagiSolver run (demo only, more iterations needed for convergence)
result <- MagiSolver(y, fnmodel, control = list(nstepsHmc = 20, niterHmc = 500))

# Inferred trajectories
plot(result, comp.names = c("V", "R"), xlab = "Time", ylab = "Level")

# Parameter trace plots
plot(result, type = "trace", par.names = c("a", "b", "c", "sigmaV", "sigmaR"), sigma = TRUE)
```

---

**pntrmodelODE**  **Protein transduction model**
Description

The protein transduction equations model a biochemical reaction involving a signaling protein that degrades over time. The system components $X = (S, S_d, R, S_R, R_{pp})$ represent the levels of signaling protein, its degraded form, inactive state of $R$, $S - R$ complex, and activated state of $R$. $S, S_d, R, S_R$ and $R_{pp}$ are governed by the following differential equations:

\[
\frac{dS}{dt} = -k_1 \cdot S - k_2 \cdot S \cdot R + k_3 \cdot S_R
\]

\[
\frac{dS_d}{dt} = k_1 \cdot S
\]

\[
\frac{dR}{dt} = -k_2 \cdot S \cdot R + k_3 \cdot S_R + \frac{V \cdot R_{pp}}{K_m + R_{pp}}
\]

\[
\frac{dS_R}{dt} = k_2 \cdot S \cdot R - k_3 \cdot S_R - k_4 \cdot S_R
\]

\[
\frac{dR_{pp}}{dt} = k_4 \cdot S_R - \frac{V \cdot R_{pp}}{K_m + R_{pp}}
\]

where $\theta = (k_1, k_2, k_3, k_4, V, K_m)$ are system parameters.

Usage

\begin{verbatim}
ptransmodelODE(theta, x, tvec)
ptransmodelDx(theta, x, tvec)
ptransmodelDtheta(theta, x, tvec)
\end{verbatim}

Arguments

theta vector of parameters.

x matrix of system states (one per column) at the time points in tvec.

tvec vector of time points

Value

ptransmodelODE returns an array with the values of the derivatives $\dot{X}$.

ptransmodelDx returns a 3-D array with the values of the gradients with respect to $X$.

ptransmodelDtheta returns a 3-D array with the values of the gradients with respect to $\theta$.

References

setDiscretization

Examples

\[
\begin{align*}
\theta & \leftarrow c(0.07, 0.6, 0.05, 0.3, 0.017, 0.3) \\
x & \leftarrow \text{matrix}(1:25, \text{nrow} = 5, \text{ncol} = 5) \\
tvec & \leftarrow 1:5
\end{align*}
\]

\[
p\text{transmodelODE}(\theta, x, tvec)
\]

Description

Set the discretization level of a data matrix for input to \texttt{MagiSolver}, by inserting time points where the GP is constrained to the derivatives of the ODE system.

Usage

\[
\text{setDiscretization}(\text{dat}, \text{level}, \text{by})
\]

Arguments

- \textit{dat} data matrix. Must include a column with name ‘time’.
- \textit{level} discretization level (a positive integer). \(2^{\text{level}} - 1\) equally-spaced time points will be inserted between each row of \textit{dat}.
- \textit{by} discretization interval. As an alternative to \textit{level}, time points will be inserted (as needed) to form an equally-spaced discretization set from the first to last observations of \textit{dat}, with interval by between successive discretization points. This can be useful when the time points in \textit{dat} are unevenly spaced.

Details

Specify the desired discretization using \textit{level} or \textit{by}.

Value

Returns a data matrix with the same columns as \textit{dat}, with rows added for the inserted discretization time points.

Examples

\[
\begin{align*}
\text{dat} & \leftarrow \text{data.frame}(\text{time} = 0:10, x = \text{rnorm}(11)) \\
\text{setDiscretization}(\text{dat}, \text{level} = 2) \\
\text{setDiscretization}(\text{dat}, \text{by} = 0.2)
\end{align*}
\]
Summary of parameter estimates from `magioutput` object

**Description**
Computes a summary table of parameter estimates from the output of MagiSolver.

**Usage**
```r
## S3 method for class 'magioutput'
summary(
  object,
  sigma = FALSE,
  par.names,
  est = "mean",
  lower = 0.025,
  upper = 0.975,
  digits = 3,
  ...
)
```

**Arguments**
- `object`: a `magioutput` object.
- `sigma`: logical; if true, the noise levels $\sigma$ will be included in the summary.
- `par.names`: vector of parameter names for the summary table. If provided, should be the same length as the number of parameters in $\theta$, or the combined length of $\theta$ and $\sigma$ when `sigma = TRUE`.
- `est`: string specifying the posterior quantity to treat as the estimate. Default is `est = "mean"`, which treats the posterior mean as the estimate. Alternatives are the posterior median (`est = "median"`, taken component-wise) and the posterior mode (`est = "mode"`, approximated by the MCMC sample with the highest log-posterior value).
- `lower`: the lower quantile of the credible interval, default is 0.025.
- `upper`: the upper quantile of the credible interval, default is 0.975.
- `digits`: integer; the number of significant digits to print.
- `...`: additional arguments affecting the summary produced.

**Details**
Computes parameter estimates and credible intervals from the MCMC samples. By default, the posterior mean is treated as the parameter estimate, and `lower = 0.025` and `upper = 0.975` produces a central 95% credible interval.
Value

Returns a matrix where rows display the estimate, lower credible limit, and upper credible limit of each parameter.

Examples

# Set up odeModel list for the Fitzhugh-Nagumo equations
fnmodel <- list(
  fOde = fnmodelODE,  # function that computes the ODEs, specified with the form f(theta, x, tvec). See examples.
  fOdeDx = fnmodelDx,  # function that computes the gradients of the ODEs with respect to the system components. See examples.
  fOdeDtheta = fnmodelDtheta,  # function that computes the gradients of the ODEs with respect to the parameters \theta. See examples.
  thetaLowerBound = c(0, 0, 0),
  thetaUpperBound = c(Inf, Inf, Inf)
)

# Example FN data
data(FNdat)

# Create magioutput from a short MagiSolver run (demo only, more iterations needed for convergence)
result <- MagiSolver(FNdat, fnmodel, control = list(nstepsHmc = 5, niterHmc = 100))

summary(result, sigma = TRUE, par.names = c("a", "b", "c", "sigmaV", "sigmaR"))
Details

Calls test\_that to test equality of the analytic and numeric gradients.

Value

A list with elements testDx and testDtheta, each with value TRUE if the corresponding gradient check passed and FALSE if not.

Examples

# ODE system and gradients for Fitzhugh-Nagumo equations: fnmodelODE, fnmodelDx, fnmodelDtheta

# Example of incorrect gradient with respect to parameters theta
fnmodelDthetaWrong <- function(theta, x, tvec) {
  resultDtheta <- array(0, c(nrow(x), length(theta), ncol(x)))
  V = x[, 1]
  R = x[, 2]
  resultDtheta[, 3, 1] = V - V^3 / 3.0 - R
  resultDtheta[, 1, 2] = 1.0 / theta[3]
  resultDtheta[, 2, 2] = -R / theta[3]
  resultDtheta
}

# Sample data for testing gradient correctness
data(FNdat)

# Correct gradients
testDynamicalModel(fnmodelODE, fnmodelDx, fnmodelDtheta,
  "FN equations", FNdat[, c("V", "R")], c(.5, .6, 2), FNdat$time)

# Incorrect theta gradient (test fails)
testDynamicalModel(fnmodelODE, fnmodelDx, fnmodelDthetaWrong,
  "FN equations", FNdat[, c("V", "R")], c(.5, .6, 2), FNdat$time)
Index

* datasets
  FNdat, 4
  calcov, 2, 6–9, 13, 16
  FNdat, 4
  fnmodelDtheta (fnmodelODE), 5
  fnmodelDx (fnmodelODE), 5
  fnmodelODE, 5
  gpcov, 6
  gpmean, 7
  gpsmoothing, 8
  gpsmoothlik, 9
  hes1logmodelDtheta (hes1modelODE), 10
  hes1logmodelDx (hes1modelODE), 10
  hes1logmodelODE (hes1modelODE), 10
  hes1modelDtheta (hes1modelODE), 10
  hes1modelDx (hes1modelODE), 10
  hes1modelODE, 10
  is.magioutput, 11
  magi, 12
  magioutput, 16
  magioutput (is.magioutput), 11
  MagiPosterior, 13
  MagiSolver, 11, 14, 21
  plot.magioutput, 11, 17
  ptransmodelDtheta (ptransmodelODE), 19
  ptransmodelDx (ptransmodelODE), 19
  ptransmodelODE, 19
  setDiscretization, 15, 21
  summary.magioutput, 11, 22
  test_that, 24
  testDynamicalModel, 23