

# Package ‘deGradInfer’

December 5, 2017

**Title** Parameter Inference for Systems of Differential Equation

**Version** 1.0.0

**Description** Efficient Bayesian parameter inference for systems of ordinary differential equations. The inference is based on adaptive gradient matching (AGM, Dondelinger et al. 2013 <<http://proceedings.mlr.press/v31/dondelinger13a.pdf>>, Macdonald 2017 <<http://theses.gla.ac.uk/7987/1/2017macdonaldphd.pdf>>), which offers orders-of-magnitude improvements in computational efficiency over standard methods that require solving the differential equation system. Features of the package include flexible specification of custom ODE systems as R functions, support for missing variables, Bayesian inference via population MCMC.

**Depends** R (>= 3.3.1)

**License** GPL-3

**Encoding** UTF-8

**LazyData** true

**Imports** deSolve, gdata, gptk, graphics, stats

**RoxygenNote** 6.0.1

**Suggests** testthat, knitr, rmarkdown, ggplot2

**VignetteBuilder** knitr

**NeedsCompilation** no

**Author** Benn Macdonald [aut],  
Frank Dondelinger [aut, cre]

**Maintainer** Frank Dondelinger <[fdondelinger.work@gmail.com](mailto:fdondelinger.work@gmail.com)>

**Repository** CRAN

**Date/Publication** 2017-12-05 18:14:24 UTC

## R topics documented:

agm . . . . .	2
doMCMC . . . . .	4
getODEGradient . . . . .	5

LV_example_dataset . . . . .	6
proposeParamsMCMC . . . . .	6
solveODE . . . . .	7

<b>Index</b>	<b>8</b>
--------------	----------

---

agm	<i>Main function for adaptive gradient matching</i>
-----	---

---

## Description

Function `agm` uses adaptive gradient matching to infer the parameters of a user-defined ODE system from data. For details on AGM, see e.g. Dondelinger et al. (2013), MacDonald (2017).

## Usage

```
agm(data, time, ode.system, numberOfParameters, noise.sd = 0.001,
     observedVariables = 1:ncol(data), temperMismatchParameter = FALSE,
     initialisedParameters = NULL, chainNum = 20, gpCovType = "rbf",
     saveFile = NULL, defaultTemperingScheme = NULL, maxIterations = 3e+05,
     showPlot = FALSE, showProgress = FALSE, mismatchParameterValues = NULL,
     originalSignalOnlyPositive = FALSE, logPrior = "Uniform",
     explicit = FALSE, explicitNoiseInfer = TRUE)
```

## Arguments

<code>data</code>	A matrix of observations of the ODE system over time. The number of rows is equal to the number time points and the number of columns is equal to the number of variables in the system.
<code>time</code>	A vector containing the time points at which the observations were made.
<code>ode.system</code>	A function describing the ODE system. See Details for more information.
<code>numberOfParameters</code>	A scalar specifying the number of parameters in the ODE system. If explicitly solving the ODE system, the number of parameters will (usually) be equal to the number of ODE parameters plus the number of initial conditions of the system.
<code>noise.sd</code>	A scalar specifying the value at which to fix the standard deviation of the observational noise. Default <code>noise.sd=1e-3</code> .
<code>observedVariables</code>	A vector specifying which variables are observed in the system. Default is <code>observedVariables=1:ncol(data)</code> (fully observed system).
<code>temperMismatchParameter</code>	Logical: whether tempering of the gradient mismatch parameter be carried out? Default is <code>temperMismatchParameter=FALSE</code> .
<code>initialisedParameters</code>	A vector containing ODE parameters at which to initialise the MCMC. Can be set as <code>NULL</code> to initialise with a random draw from the prior distribution. Default is <code>initialisedParameters=NULL</code> .

chainNum	A scalar specifying the number of parallel temperature chains. Default is chainNum=20.
gpCovType	A string specifying the choice of kernel for the Gaussian process. Currently, there are two: gpCovType="rbf" and gpCovType="sigmoidVar".
saveFile	A string specifying the path and name of the file containing the result output. Can be set as NULL to save as "AGM Results.R" to the current working directory. Default is saveFile=NULL.
defaultTemperingScheme	A string indicating which of the two default gradient mismatch parameter value ladders to use. Choices are defaultTemperingScheme="LB2" or defaultTemperingScheme="LB10". Should only be used when temperMismatchParameter=TRUE. Default is defaultTemperingScheme=NULL.
maxIterations	A scalar specifying the number of total MCMC iterations. Default is maxIterations=300000.
showPlot	Logical: whether plots of the MCMC progress should be displayed. Default is showPlot=FALSE.
showProgress	Logical: whether % completion and various parameter values should be printed to the workspace. Default is showProgress=FALSE.
mismatchParameterValues	A matrix containing user specified values for the gradient mismatch parameter. The number of rows should be equal to chainNum and the number of columns should be equal to the number of variables in the system. A typical ladder should have the largest value in the first row and the smallest value in the last row. Should only be used when defaultTemperingScheme=NULL. Default is mismatchParameterValues=NULL.
originalSignalOnlyPositive	Logical: whether all signals observed should be non-negative. When originalSignalOnlyPositive=TRUE any negative values of the sampled interpolant will be set to zero. Default is originalSignalOnlyPositive=FALSE.
logPrior	A string specifying whether one of the default log priors for the ODE parameters should be used, or a user-specified function. Current choices for the default prior are "Uniform", "Gamma" (shape=4, rate=2) and "Mixed" (3 ODE parameters; N(mean=0, sd=0.4), N(mean=0, sd=0.4)). Alternatively the user may specify a function for calculating the prior, see Details below. Default is logPrior='Uniform'.
explicit	Logical: whether the ODE system should be explicitly solved, rather than doing gradient matching. This means that the Gaussian process model is ignored, and the ODE system is directly fitted to the observed data. Default is explicit=FALSE.
explicitNoiseInfer	Logical: whether the standard deviation of the observational noise should be inferred when using the method that explicitly solves the ODEs. Only considered when explicit=TRUE. Default is explicitNoiseInfer=TRUE.

## Details

The parameters `ode.system` should be a function of the form  $f(t, X, \text{params})$  where  $t$  is the time point vector for which the derivatives should be calculated,  $X$  is a  $T$  by  $p$  matrix containing the values of the variables in the system at time  $t$ , and  $\text{params}$  is a vector with the current estimated

parameter values. The function should return a matrix with the derivatives of  $x$  with respect to time (in the same order as in  $x$ ). Note that in order to be consistent with the ode in package `deSolve`, we require that the function also works for input at a single time point.

For specifying a custom prior on the parameters, the user should write their function to take as input a vector of parameters, and return a vector of log densities for a given parameter set. For example, `logPrior = function(params) c(dgamma(params, 1, 1, log=TRUE))` defines a Gamma parameter prior with shape and scale 1.

## Value

Function returns a list with elements `posterior.mean`, the mean of the parameter samples from the posterior (after burning of 1/4 of the number of samples taken), `posterior.sd`, the standard deviation, `posterior.samples`, the parameter samples, `ll` the log likelihood, `x.samples`, the samples of the latent variables, `gp.samples`, the samples of the GP hyperparameters, `noise.samples` the samples of sigma, `swappedChains` the number of times the chains have been swapped, `ll.all.chain`, the log likelihood for all chains and `tuning`, the inferred tuning parameters for acceptance of the MCMC moves.

## Examples

```
dataTest <- LV_example_dataset$data
timeTest <- LV_example_dataset$time
noiseTest <- LV_example_dataset$noise

LV_func = function(t, X, params) {
  dxdt = cbind(
    X[,1]*(params[1] - params[2]*X[,2]),
    - X[,2]*(params[3] - params[4]*X[,1])
  )
  return(dxdt)
}

# Example run only; to achieve convergence the number of iterations and
# chains must be increased.
param.result = agm(data=dataTest,time=timeTest,noise.sd=0.31,ode.system=LV_func,
  numberOfParameters=4,temperMismatchParameter=TRUE,
  chainNum=4, maxIterations=150,originalSignalOnlyPositive=TRUE,
  logPrior="Gamma",defaultTemperingScheme="LB10")

print(param.result$posterior.mean)
```

**Description**

Main MCMC function Runs the MCMC for the specified number of iterations and returns the sampled parameter values

**Usage**

```
doMCMC(timePoints, data, auxVars, options)
```

**Arguments**

timePoints	Measured time points for the ODE system.
data	Observed data.
auxVars	Auxiliary variables.
options	Options for MCMC run.

**Value**

List with samples parameters and information about the sampling run.

---

getODEGradient	<i>Calculate gradients from ODE system</i>
----------------	--

---

**Description**

Calculate gradients from ODE system

**Usage**

```
getODEGradient(X, timePoints, params, auxVars, species = 1:dim(X)[2])
```

**Arguments**

X	Latent values for the species
timePoints	Times at which to calculate the ODE gradients
params	Current parameter estimates
auxVars	Auxiliary variables (including function for ODE gradients)
species	Which species to return (default=all)

**Value**

A T by length(species) matrix with the gradients calculated at each time point for the specified species.

---

LV_example_dataset	<i>Data from a Lotka-Volterra ODE system with 2 species and 4 parameters. Species in order are: 1. Sheep (Prey) 2. Wolves (Predators)</i>
--------------------	---

---

**Description**

Data from a Lotka-Volterra ODE system with 2 species and 4 parameters. Species in order are: 1. Sheep (Prey) 2. Wolves (Predators)

**Usage**

```
LV_example_dataset
```

**Format**

A list with 5 components

**data** Observed species with observation noise.

**data.true** Observed species without observation noise.

**time** Observed time points.

**params** Observed parameters.

**noise** Variance for the Gaussian observation noise.

**Source**

Generated by function simulateLotkaVolterraModel.

---

proposeParamsMCMC	<i>Sample from proposal distribution for MCMC</i>
-------------------	---

---

**Description**

Sample from proposal distribution for MCMC

**Usage**

```
proposeParamsMCMC(oldParams, inferredParams, width)
```

**Arguments**

oldParams Previous parameter values

inferredParams Proposed parameter values

width Width of random walk proposal

**Value**

List with proposed parameters, indicator variable of which parameters have changed, old and new proposal probabilities (if

---

solveODE	<i>Solve ODE system explicitly.</i>
----------	-------------------------------------

---

**Description**

Solve ODE system explicitly.

**Usage**

```
solveODE(num.species, timePoints, ode.system, params)
```

**Arguments**

num.species	Number of variables (species) in the system.
timePoints	Time points at which to evaluate the ODE system.
ode.system	Function for calculating the derivatives of the ODE system.
params	Current values for the ODE parameter estimates.

**Value**

A list with two elements: x contains the results of integrating the ODE at the given time points, and error flags if there has been an error while invoking deSolve.

# Index

## \*Topic **datasets**

LV\_example\_dataset, 6

agm, 2

doMCMC, 4

getODEGradient, 5

LV\_example\_dataset, 6

proposeParamsMCMC, 6

solveODE, 7