Package ‘learningmodule’

November 16, 2017

Title A collection of four data assimilation routines.

Version 1.0

Author Riddhi Singh

Description This package implements four commonly used methods of data assimilation - the Ensemble Kalman filter, the Partile filter, Precalibration, and MCMC. Two models are built into the package with associated code for implementing them.

License GPL 2.0

LazyData true

Suggests testthat

Imports MASS,
mvtnorm,
numDeriv,
mcmc

RoxygenNote 5.0.1

R topics documented:

`addprocessnoise` *addprocessnoise*

Description

function to adds process noise to the deterministic model input

Usage

```
addprocessnoise(function_handle, Qmod, qtype, seedval, x, theta,
  timestep = 0.01, t = 0)
```

Arguments

`function_handle`	model function that takes a vector (x) as input and returns an output vector
`Qmod`	A scalar/ vector/ array with process (co)variances. If scalar, the value is multiplied by an identity matrix with dimensions equal to the length of the vector x. If vector, the values form the diagonal of the covariance matrix with off-diagonal elements set to zero. The array is directly used without any further processing.
`seedval`	A scalar value to be used as a random seed for generating process noise
`x`	a vector at which the function_handle is being evaluated

Value

Model output vector with process noise added to it

Author(s)

Riddhi Singh

Examples

```
addprocessnoise(f_model, c(1e5, 6000, 2500), 10, 100)
```

`basecasesetup` *Problem set up for the base case model*

Description

This function generates the synthetic truth and observations for the base case model

Usage

```
basecasesetup(input)
```

Arguments

`input`	A list of user defined or default values for model

Value

A list with the synthetic truth and observations

Author(s)

Riddhi Singh

See Also

`setnltsdefault`

Examples

```
#using the demostration example set up
input             <- setnltsdefault()
setup             <- nltssetup(input)
```

`dafilter` *Define the generic dafilter function*

Usage

```
dafilter(model_handle, ...)
```

Arguments

`model_handle`	The function that executes the model, must take in a numvarsx1 input and return a numvarsx1 output, where numvars is the number of inputs to the model
`measurement_handle`	The function that converts the model output state to measurement
`observations`	The set of observations that the filter will use to update its estimates.
`input`	A list with: the first guess of forecast error covariance matrix (Pguess), first guess values of forecast vector (Xguess), model process error covariance matrix (Qmod), measurement covaraince (R), and, time resolution of model (T)
`option`	Either: "KALMAN", "EXTENDED", "ENSEMBLE", "PARTICLE"

Author(s)

Riddhi Singh

See Also

`kalmanfilter` and `extendedkalmanfilter` and `ensemblekalmanfilter` and `particlefilter`

`dafilter.default` *Implement the default filter which is the Kalman filter*

Description

This function implements the Kalman filter for a given set of model inputs and observations. The Kalman filter is the optimal estimator if the model is linear and the measurement/ process noises are normally distributed.

Usage

```
## Default S3 method:
dafilter(model_handle, measurement_handle, observations,
  input, option = "KALMAN")
```

Arguments

`model_handle`	The function that executes the model, must take in a numvarsx1 input and return a numvarsx1 output, where numvars is the number of inputs to the model
`measurement_handle`	The function that converts the model output state to measurement
`observations`	The set of observations that the filter will use to update its estimates.
`input`	A list with: the first guess of forecast error covariance matrix (Pguess), first guess values of forecast vector (xguess), model process error covariance matrix (Qmod), measurement covaraince (R), and, time resolution of model (T)
`option`	Either: "KALMAN", "EXTENDED", "ENSEMBLE", "PARTICLE"

Value

A filter class which is a list comprising of the mean of the forecasted variables and their associated covariance matrix

Author(s)

Riddhi Singh

See Also

`kalmanfilter` `extendedkalmanfilter` `ensemblekalmanfilter` `particlefilter`

Examples

```
Example 1: running the demonstration example
model_handle       <- f_model
measurement_handle <- h_meas
input              <- setdefault()
setup              <- problemsetup(input)
kalmanout        <- dafilter(model_handle, measurement_handle, setup$z, input, option="KALMAN")
Example 2: running the lake model
model_handle       <- f_lake
measurement_handle <- h_lake
input              <- setlakedefault()
setup              <- lakesetup(input)
ensembleout      <- dafilter(model_handle, measurement_handle, setup$z, input, option="ENSEMBLE")
```

`ensemblekalmanfilter` *Implement The Ensemble Kalman Filter*

Description

This function implements the ensemble Kalman filter for a given set of model inputs and observations. The filter approximates a Gaussian distribution using a set of randomly generated, and is a good choice for models with significant non linearity.

Usage

```
ensemblekalmanfilter(input, observations)
```

Arguments

`input`	A list with the problem setup including the first guess of forecast error covariance matrix (Pguess), first guess values of forecast vector (xguess), model process error covariance matrix (Qmod), measurement covaraince (R), and, time step of model (timestep)
`observations`	The set of observations that the filter will use to update its estimates.

Value

A filter class which is a list comprising of the mean of the forecasted variables and their associated covariance matrix

Author(s)

Riddhi Singh

See Also

`kalmanfilter`, `extendedkalmanfilter`, `particlefilter`

Examples

```
input          <- setdefault()
setup          <- problemsetup(input)
ensembleout    <- ensemblekalmanfilter(input,setup$z)
```

`extendedkalmanfilter` *Implement the extended Kalman filter*

Description

This function implements the extended Kalman filter for a given set of model inputs and observations.

Usage

```
extendedkalmanfilter(input, observations)
```

Arguments

`input`	A list with the problem setup including the first guess of forecast error covariance matrix (Pguess), first guess values of forecast vector (xguess), model process error covariance matrix (Qmod), measurement covaraince (R), and, time step of model (timestep)
`observations`	The set of observations that the filter will use to update its estimates.

Value

A filter class which is a list comprising of the mean of the forecasted variables and their associated covariance matrix

Author(s)

Riddhi Singh

See Also

`kalmanfilter`, `ensemblekalmanfilter`, `particlefilter`

Examples

```
Example: running the lake model
input             <- setlakedefault(1) # 1 for the eutrophication case
setup             <- lakesetup(input)
exkalout          <- extendedkalmanfilter(input,setup$z)
```

`filterdemo` *Function to run the demonstration example*

Description

This function can be called to run the demostration example that implements the filters. The function allows the user to run the model without any inputs (by setting all values at pre-defined default). Alternatively, the user can enter the choice of filters, random seeds, etc.

Usage

```
filterdemo()
```

Arguments

`x`	a vector of inputs to the model

Author(s)

Riddhi Singh

References

Analytic Sciences Corporation (1974), Applied optimal estimation, Gelb, A. (ed.), M.I.T Press, Cambridge, Mass, ISBN: 0262200279

See Also

`setdefault`, `readinput`, `problemsetup`, `f_model`, `h_meas`

Examples

```
filterdemo()
```

`filterperformance` *Assess the performance of various filters for any test problem*

Description

This function assesses the ability of a filter to estimate the state variables and/or unknown parameters of a filter. Two criteria are assessed: 1) time taken to identify the state variables within ten percent of their values and 2) ability to track a key state variable of interest

Usage

```
filterperformance(object, truth, mcmcout = NULL, varindex = NULL)
```

Arguments

`object`	An output of filter class
`truth`	The true values of the variables (in case of a synthetic experiment or the demostration example)
`varindex`	The index of the key variable to be used in assessing performance using criteria 2

Value

A list with values of both criteria

Author(s)

Riddhi Singh

See Also

`kalmanfitler` `extendedkalmanfitler` `ensemblekalmanfitler` `particlefitler`

Examples

```
model_handle       <- f_model
measurement_handle <- h_meas
input              <- setdefault()
setup              <- problemsetup(input)
kalmanout          <- dafilter(model_handle, measurement_handle, setup, input)
kalperf            <- filterperformance(kalmanout, setup$xtrue, mcmcout)
exkalperf          <- filterperformance(exkalmanout, setup$xtrue, mcmcout)
enkalperf          <- filterperformance(enkalmanout, setup$xtrue, mcmcout)
partperf           <- filterperformance(partout, setup$xtrue, mcmcout)
precalibperf       <- filterperformance(precalib, setup$xtrue, mcmcout)
```

`fmat_lakepars` *Fmat function for the parameters of the lake model*

Description

Fmat function for the parameters of the lake model (for use in the Kalman and extended Kalman filters)

Usage

```
fmat_lakepars(x, timestep = 1, t = 0)
```

Arguments

`x`	input to the model: b and q at previous time step

Value

ouput phosphorus values at the next time step

Author(s)

Riddhi Singh

See Also

`lakesetup` `setlakedefault`

`fmat_lakestate` *Fmat function for the state variable of the lake model*

Description

Fmat function for the state variable of the lake model (for use in the Kalman and extended Kalman filters)

Usage

```
fmat_lakestate(x, theta, timestep = 1, t = 0)
```

Arguments

`x`	input to the model: phosphorus at previous time step
`theta`	b and q parameter values

Value

ouput phosphorus values at the next time step

Author(s)

Riddhi Singh

See Also

`lakesetup` `setlakedefault`

`f_basecasepars` *the linear base case model used to benchmark filter perforamce*

Description

Insert description

Usage

```
f_basecasepars(x, timestep = 1, t = 0)
```

Arguments

`x`	a vector of inputs to the model: this is phosphorus and parameter b which are the unknowns for the problem
`invar`	a vector of inputs that are not a part of the updating process

Value

A vector of output y values based on the linear model

Author(s)

Riddhi Singh

`f_basecasestate` *the linear base case model used to benchmark filter perforamce*

Description

Insert description

Usage

```
f_basecasestate(x, theta, timestep = 1, t = 0)
```

Arguments

`x`	a vector of inputs to the model: this is phosphorus and parameter b which are the unknowns for the problem
`invar`	a vector of inputs that are not a part of the updating process

Value

A vector of output y values based on the linear model

Author(s)

Riddhi Singh

`f_lakepars` *Updates parameters of the lake model*

Description

Function to update the parameters of the lake model for the next time step

Usage

```
f_lakepars(x, timestep = 1, t = 0)
```

Arguments

`x`	input to the model: b and q values

Value

ouput phosphorus values at the next time step

Author(s)

Riddhi Singh

See Also

`lakesetup` `setlakedefault`

Examples

```
x  <- c(0.4,2)
y  <- f_lakestate(x)
```

`f_lakestate` *the lake model*

Description

Model of the lake to update the phosphorus fluxes at every time step from Carpenter et al. 1999. The model is set up to update the lake's state variable (Phosphorus in the lake).

Usage

```
f_lakestate(x, theta, timestep = 1, t = 0)
```

Arguments

`x`	input to the model: phosphorus at previous time step
`theta`	b and q parameter values at which to estimate the next state

Value

ouput phosphorus values at the next time step

Author(s)

Riddhi Singh

See Also

`lakesetup` `setlakedefault`

Examples

```
x  <- c(0.5, c(0.4,2))
y  <- f_lakestate(x)
```

`getQ` *Calculate the model/process covariance matrix*

Description

This function can be called to run the demostration example that implements the filters. The function allows the user to run the model without any inputs (by setting all values at pre-defined default). Alternatively, the user can enter the choice of filters, random seeds, etc.

Usage

```
getQ(Qmod, numvars)
```

Arguments

`Qmod`	A scalar/ vector/ array with process (co)variances. If scalar, the value is multiplied by an identity matrix with dimensions equal to the length of the vector x. If vector, the values form the diagonal of the covariance matrix with off-diagonal elements set to zero. The array is directly used without any further processing.
`numvars`	The number of variables being forecasted

Value

The covariance matrix Q

Author(s)

Riddhi Singh

Examples

```
#initializing using a scalar
Qmod  <- 1
#initializing using a vector
getQ(Qmod,3)
Qmod  <- c(1,1,10)
#initializing using a matrix
Qmod <- array(1, dim=c(3,3))
getQ(Qmod,3)
```

`hmat_basecasepars` *Measurement function for the linear model parameter*

Description

The measurement function for the linear model parameter, to be used in the lake model, to be used in Kalman and extended Kalman filters

Usage

```
hmat_basecasepars(x, state)
```

Arguments

`x`	State variable vector for the model
`theta`	Parameter vector for the model

Value

Returns the hmat values

Author(s)

Riddhi Singh

`hmat_lakepars` *Measurement function for the lake parameters*

Description

The measurement function for the lake parameters used in the lake model, to be used in Kalman and extended Kalman filters

Usage

```
hmat_lakepars(x, state)
```

Arguments

`x`	State variable vector for the lake model
`theta`	Parameter vector for the lake model

Value

Returns the hmat values

Author(s)

Riddhi Singh

See Also

`f_lake`

`h_basecase` *The linear measurement function used in the basecase model*

Usage

```
h_basecase(x)
```

Arguments

`x`	A vector of variables that are output from the model

Value

Returns the measurement values

Author(s)

Riddhi Singh

`h_lakestate` *The linear measurement function used in the lake model*

Usage

```
h_lakestate(x)
```

Arguments

`x`	A vector of variables that are output from the model f_lake

Value

Returns the measurement values

Author(s)

Riddhi Singh

See Also

`f_lakestate`

Examples

```
x <- c(0.5)
# cast it into an array format
h_lakestate(x)
```

`kalmanfilter` *Implement the Kalman filter*

Description

This function implements the Kalman filter for a given set of model inputs and observations. The Kalman filter is the optimal estimator if the model is linear and the measurement/ process noises are normally distributed.

Usage

```
kalmanfilter(input, observations)
```

Arguments

`input`	A list with the problem setup including the first guess of forecast error covariance matrix (Pguess), first guess values of forecast vector (xguess), model process error covariance matrix (Qmod), measurement covaraince (R), and, time step of model (timestep)
`observations`	The set of observations that the filter will use to update its estimates.

Value

A filter class which is a list comprising of the mean of the forecasted variables and their associated covariance matrix

Author(s)

Riddhi Singh

See Also

`extendedkalmanfilter`, `ensemblekalmanfilter`, `particlefilter`

Examples

```
Example: running the lake model
input              <- setlakedefault(1) # 1 for the eutrophication case
setup              <- lakesetup(input)
kalmanout          <- kalmanfilter(input,setup$z)
```

`lakeparscheck` *Checks if given matrix satisfies ranges of parameters for the lake model*

Description

This function checks for unrealistic values of the lake model parameters and returns realistic sets.

Usage

```
lakeparscheck(x_enspars)
```

Arguments

`x_enspars`	A matrix of parameters of size numpars x N, where N is the number of sets and numpars are number of lake model parameters

Value

indices The locations of points to be removed

Author(s)

Riddhi Singh

See Also

`ensemblekalmanfilter`, `particlefilter`

Examples

```
indices           <- lakeparscheck(x_enspars)
```

`lakesetup` *Problem set up for the lake problem*

Description

This function generates the synthetic truth and observations for the lake problem

Usage

```
lakesetup(input)
```

Arguments

`input`	A list of user defined or default values for lake parameters, input polution, total time for the simulation, etc. See setlakedefault() for a full list of input values.

Value

A list with the synthetic truth and observations

Author(s)

Riddhi Singh

See Also

`setlakedefault`

Examples

```
#using the demostration example set up
input   <- setlakedefault()
setup   <- lakesetup(input)
```

`lakestatecheck` *Checks if given matrix satisfies ranges of state for the lake model*

Description

This function checks for unrealistic values of the lake model states and returns realistic states.

Usage

```
lakestatecheck(x_ensstate)
```

Arguments

`x_ensstate`	A matrix of states of size numstates x N, where N is the number of sets and numstates are number of lake model states
`limits`	A matrix of lower and upper limits of each state, size 2xnumstates

Value

Updated x_ensstate

Author(s)

Riddhi Singh

See Also

`ensemblekalmanfilter`, `particlefilter`

Examples

```
x_ensstate          <- lakestatescheck(x_ensstate)
```

`learningmodule` *learningmodule: a collection of four data assimilation schemes: Ensemble Kalman Filter, Particle Filter, Precalibration, and MCMC*

Description

This package implements four commonly used methods of data assimilation.

learningmodule functions

dafilter.default dafilter.ensemblekalmanfilter dafilter.particlefilter dafilter.precalibration dafilter.runmcmc

---

`likelihoodscans` *Likelihood as a function of lake parameters*

Description

Plots the likelihood function as a function of b and q parameters - this helps in diagnosing performance of MCMC and the particle filter

Usage

```
likelihoodscan()
```

Author(s)

Riddhi Singh

`likelihood_update` *Update particle weights based on the likelihood calculated using the simulations with observations*

Description

This measurement function assumes that the radar is located

Usage

```
likelihood_update(xfor, observations, R)
```

Arguments

`observations`	A scalar with the observation at the time of forecast
`R`	Measurement variance
`xforPF`	An array of dimension n_ens x numfor with the forecasted variable values for each particle
`par_w`	A vector of particle weights

Value

Updated particle weights

Author(s)

Riddhi Singh

See Also

dafilter.particle

Examples

```
%%  ~~any examples~~
```

`logposteriorlake` *The log of the posterior for the lake model*

Description

This function estimates the full posterior for the lake model, to be used for MCMC analysis

Usage

```
logposteriorlake(beta, input, yobs, xstate = NULL, begintime = 1, endtime,
  option = 0, priornext = NULL)
```

Arguments

`beta`	The list of parameters that MCMC needs to identify
`input`	Any additional inputs for the lake model, same as the output from setlakedefault()
`yobs`	The vector of observations

Value

log of the posetrior (unnormalized)

Author(s)

Riddhi Singh

See Also

`basecasesetup` `setbasecasedefault`

Examples

```
%%  ~~any examples~~
```

`logpriorbasecase` *The log of the prior for the linear model*

Description

This function estimates the prior, to be used for MCMC analysis

Usage

```
logpriorbasecase(beta, priornext, priorchoice)
```

Arguments

`beta`	The list of parameters that MCMC needs to identify
`input`	Any additional inputs for the lake model, same as the output from setbasecasedefault()
`yobs`	The vector of observations

Value

log of the posetrior (unnormalized)

Author(s)

Riddhi Singh

`logpriorlake` *The log of the prior for the lake model*

Description

This function estimates the prior, to be used for MCMC analysis

Usage

```
logpriorlake(beta, priornext, priorchoice)
```

Arguments

`beta`	The list of parameters that MCMC needs to identify
`input`	Any additional inputs for the lake model, same as the output from setlakedefault()
`yobs`	The vector of observations

Value

log of the posetrior (unnormalized)

Author(s)

Riddhi Singh

See Also

`basecasesetup` `setbasecasedefault`

Examples

```
%%  ~~any examples~~
```

`multirandnorm` *Generate a random sample from a multi-normal distribution*

Description

Generate a random sample from a multi-normal distribution with specified mean values and covariance matrix

Usage

```
multirandnorm(meanvec, covmat, num_pts, seedval)
```

Arguments

`meanvec`	A vector with mean values of the multi-normal distribution
`covmat`	The covaraince matrix
`num_pts`	The number of points to be generated
`seedval`	The seed for random number generation

Value

An array of dimension num_pts x length(meanvec)

Author(s)

Riddhi Singh

Examples

```
covmat  <- diag(c(1,1,1))
randmat <- multirandnorm(c(0,0,0), covmat, 5, 10 )
```

`paperplot1` *Runs the code for the generating Figure 1 in the publication.*

Description

This function runs the code for generating Figure 1 in the study.

Usage

```
paperplot1()
```

Author(s)

Riddhi Singh

`paperplot2` *Runs the code for the generating Figure 2 in the publication.*

Description

This function runs the code for generating Figure 2. The figure shows the trajectories of state variables with time for each method.

Plots the likelihood function as a function of b and q parameters.

Usage

```
paperplot2(option = 0, filterout = NULL)

paperplot3(option = 0, endyear = 100, alldata = NULL)
```

Author(s)

Riddhi Singh

Riddhi Singh

`paperplot2_Sup` *Runs the code for the generating Figure 2(supplementary version) in the publication*

Description

This function runs the code for generating Figure 2's supplement with different input policies. The figure shows the trajectories of state variables with time for each method.

Usage

```
paperplot2_Sup(option = 2, filterout = NULL)
```

Author(s)

Riddhi Singh

`paperplot4par` *This function runs the code for generating Figure 4.*

Description

Runs the code for the convergence plot for the ensemble Kalman filter, particle filter, pre-calibration, and MCMC.

Usage

```
paperplot4par(option = 0)
```

Author(s)

Riddhi Singh

`paperplot4par_Sup` *This function runs the code for generating Figure 4's supplementary versions.*

Description

Runs the code for the convergence plot for the ensemble Kalman filter, particle filter, pre-calibration, and MCMC.

Usage

```
paperplot4par_Sup(option = 2)
```

Author(s)

Riddhi Singh

`paperplot5` *Runs the code for the plots in the publication*

Description

Plots the PDFs of state variables at the point of no return for the flipping case.

Usage

```
paperplot5(option = 1, evalyear = NULL)
```

Author(s)

Riddhi Singh

`paperplot6par` *Runs the code for Figure 6 - parallel version for execution*

Description

Plotting the predicting probability of eutrophication for each method as a function of learning time.

Usage

```
paperplot6par(option = 1)
```

Author(s)

Riddhi Singh

`paperplot6par_Sup` *Runs the code for Figure 6's supplementary version - parallel version for execution*

Description

Plotting the predicting probability of eutrophication for each method as a function of learning time.

Usage

```
paperplot6par_Sup(option = 2)
```

Author(s)

Riddhi Singh

`paperplot_Sup1` *Runs the code for the supplementary figures in the publication.*

Description

Runs the code for plotting the mean and 90

Usage

```
paperplot_Sup1(parplot = 1)
```

Author(s)

Riddhi Singh

`paperplot_Sup2` *Runs the code for the supplementary figures in the publication.*

Description

This function runs the code for generating the random realizations used in the analysis for Figures 4 and 6.

Usage

```
paperplot_Sup2(option = 1)
```

Author(s)

Riddhi Singh

`paperplot_Sup3` *Runs the code for the supplementary figures in the publication.*

Description

This function runs the code for plotting number of usable ensembles/particles/samplse as a function of time.

Usage

```
paperplot_Sup3()
```

Author(s)

Riddhi Singh

`paperplot_Sup4` *Runs the code for the supplementary figures in the publication.*

Description

This function runs the code for generating the MCMC performance plot.

Usage

```
paperplot_Sup4(option = 1)
```

Author(s)

Riddhi Singh

`paperplot_Sup5` *Runs the code for the generating supplementary figures in the publication.*

Description

This function runs the code for plotting alternative pollution strategies tested.

Usage

```
paperplot_Sup5()
```

Author(s)

Riddhi Singh

`particlefilter` *Implement the particle filter*

Description

This function implements the particle filter for a given set of model inputs and observations. The filter approximates a Gaussian distribution using a set of randomly generated, and is a good choice for models with significant non linearity

Usage

```
particlefilter(input, observations)
```

Arguments

`input`	A list with the problem setup including the first guess of forecast error covariance matrix (Pguess), first guess values of forecast vector (xguess), model process error covariance matrix (Qmod), measurement covaraince (R), and, time step of model (timestep)
`observations`	The set of observations that the filter will use to update its estimates.

Value

A filter class which is a list comprising of the mean of the forecasted variables and their associated covariance matrix

Author(s)

Riddhi Singh

See Also

`kalmanfilter`, `extendedkalmanfilter`, `ensemblekalmanfilter`

Examples

```
input           <- setdefault()
setup           <- problemsetup(input)
particleout     <- particlefilter(input,setup$z)
```

`pcritestimator` *Estimate the critical phosphorus threshold value*

Description

This function returns the critical phosphorus threshold value for a given set of b and q parameters

Usage

```
pcritestimator(parvec)
```

Author(s)

Riddhi Singh

`plot.filter` *Plot function for the filter class*

Description

Plot the variable output from the filter, one sub plot is created for each variable

Usage

```
## S3 method for class 'filter'
plot(object, truth = NULL, filtername = "Filter ",
  label = NULL)
```

Arguments

`object`	An output of filter class
`truth`	The true values of the variables (in case of a synthetic experiment or the demostration example)
`filtername`	The name of the filter for the title of the plot
`label`	The names of the variables being plotted

Author(s)

Riddhi Singh

See Also

`print.filter`, `summary.filter`

Examples

```
#using the demostration example set up
model_handle       <- f_model
measurement_handle <- h_meas
input              <- setdefault()
setup              <- problemsetup(input)
kalmanout          <- dafilter(model_handle, measurement_handle, setup, input)
plot(kalmanout)
```

`precalibration` *Perform precalibration on selected model*

Description

This function performs precalibration on a selected model. First, model parameters are sampled using assumed priors. Then output from each instance is assessed. If it is within 2 s.d. of the observations, the parameters are accepted, else rejected.

Usage

```
precalibration(input, setup, obsendtime, predendtime = input$tend,
  x_pars = NULL)
```

Arguments

`input`	The problem's input including the values for measurement nosie, process noise, etc.
`setup`	The problem's setup, including synthetic observations
`preendtime`	The time until which to predict

Value

Mean prediction along with shortliste parameter sets

Author(s)

Riddhi Singh

See Also

`basecasesetup` `setbasecasedefault`

Examples

```
%%  ~~any examples~~
```

`print.filter` *Print the output filter class*

Description

Prints the output from the filter class

Usage

```
## S3 method for class 'filter'
print(object)
```

Arguments

`object`	An output of filter class

Author(s)

Riddhi Singh

See Also

`plot.filter`, `summary.filter`

Examples

```
#using the demostration example set up
model_handle       <- f_model
measurement_handle <- h_meas
input              <- setdefault()
setup              <- problemsetup(input)
kalmanout        <- dafilter(model_handle, measurement_handle, setup, input, option="KALMAN")
print(kalmanout)
```

`print.summary.filter` *Print the sumarry of the output filter summary class*

Description

Prints the summary of the output from the filter summary class

Usage

```
## S3 method for class 'summary.filter'
print(filtersum)
```

Arguments

`object`	An output of filter summary class

Author(s)

Riddhi Singh

See Also

`summary.filter`

Examples

```
#using the demostration example set up
model_handle       <- f_model
measurement_handle <- h_meas
input              <- setdefault()
setup              <- problemsetup(input)
kalmanout          <- dafilter(model_handle, measurement_handle, setup, input)
kalmansummary      <- summary(kalmanout)
print(kalmansummary)
```

`probeutro` *Assess the odds ratio of tip vs. no tip for a given projection*

Description

This function assesses the ability of a filter to detect tipping points

Usage

```
probeutro(object, input, setup, tipthres, numsim)
```

Arguments

`object`	A matrix or vector with initial condition(s)
`truth`	The true values of the variables (in case of a synthetic experiment or the demostration example)

Value

A list with values of both criteria

Author(s)

Riddhi Singh

See Also

`tippingpredict` `paperplot4`

`randnormmat` *Random matrix with all values sampled from a normal distribution*

Description

Generate a random matrix with all values sampled from a normal distribution with specified mean and standard deviation

Usage

```
randnormmat(meanval, sdval, num_pts, rows, cols, seedval)
```

Arguments

`meanval`	The mean of the normal distribution to be sampled from
`sdval`	The standard deviation of the normal distribution to be sampled from
`num_pts`	The number of points to be generated
`rows`	The number of rows in the matrix
`cols`	The number of columns in the matrix
`seedval`	The seed for random number generation

Value

An array of dimension rows x cols

Author(s)

Riddhi Singh

Examples

```
randmat <- randnormmat(0, 1, 10, 2, 5, 100)
```

`resample` *Implement the resampling scheme for the particle filter*

Description

This function implements three resampling scheme for the particle filter: stratified, systematic, and Ripley’s

Usage

```
resample(x, w, resamplechoice)
```

Arguments

`x`	An array of particles
`w`	A vector of particle weights
`resamplechoice`	Choice of the resampling scheme, 1 for stratified, 2 for systematic, and 3 for Ripley's

Value

A list with updated particles and associated weights

Author(s)

Riddhi Singh

See Also

dafilter.particle

Examples

```
 %%  ~~any examples~~
```

`runmcmc` *Run MCMC on any of the test cases*

Description

This function runs MCMC on the test case

Usage

```
runmcmc(logposterior, input, setup, calendtime = input$tend,
  predendtime = input$tend, option = 0)
```

Arguments

`logposterior`	The posterior density calculation function
`input`	The problem's input including the values for measurement nosie, process noise, etc.

Value

Mean values of the variable vector from the last 50

Author(s)

Riddhi Singh

See Also

`basecasesetup` `setbasecasedefault`

Examples

```
%%  ~~any examples~~
```

`setbasecasedefault` *Set the default values of the base case model parameters*

Description

This function sets the default values of the base case model parameters such as a, b, and initial values of x

Usage

```
setbasecasedefault(seed = NULL)
```

Value

A list with the default values of the model

Author(s)

Riddhi Singh

See Also

`nltssetup`, `f_nlts`, `h_nlts`

Examples

```
#using the demostration example set up
input <- setnltsdefault()
```

`setlakedefault` *Set the default values of the lake data assimilation problem*

Description

This function sets the default values of the lake input parameters such as parameter q, input poolution vector, the number of filters to run, etc.

Usage

```
setlakedefault(seed = NULL, option = 1)
```

Value

A list with the default values of the demostration example

Author(s)

Riddhi Singh

See Also

`lakesetup`, `f_lakestate`, `f_lakepars`, `h_lake`

Examples

```
#using the demostration example set up
input <- setlakedefault()
```

`summary.filter` *Estimate he summary statistics of the output filter class*

Description

This function estimates the first and last values of the mean of the forecasted variables, and the standard deviation from the covaraince matrices of the output filter class

Usage

```
## S3 method for class 'filter'
summary(object)
```

Arguments

`object`	An output of filter class

Value

An object of the filter summary class

Author(s)

Riddhi Singh

See Also

dafilter.default

print.summary.filter

Examples

```
#using the demostration example set up
model_handle       <- f_model
measurement_handle <- h_meas
input              <- setdefault()
setup              <- problemsetup(input)
kalmanout          <- dafilter(model_handle, measurement_handle, setup, input)
kalmansummary      <- summary(kalmanout)
print.summary.filter(kalmansummary)
```

`threspredict` *Assess the threshold crossing time*

Description

This function the threshold crossing time for a given set of initial parameter set(s)

Usage

```
threspredict(object, input, setup, predyr, thresval)
```

Arguments

`object`	A matrix or vector with initial condition(s)
`truth`	The true values of the variables (in case of a synthetic experiment or the demostration example)

Value

A list with values of both criteria

Author(s)

Riddhi Singh

See Also

`kalmanfitler` `extendedkalmanfitler` `ensemblekalmanfitler` `particlefitler`

Examples

```
model_handle       <- f_model
measurement_handle <- h_meas
input              <- setdefault()
setup              <- problemsetup(input)
kalmanout          <- dafilter(model_handle, measurement_handle, setup, input)
kalperf            <- filterperformance(kalmanout, setup$xtrue, mcmcout)
exkalperf          <- filterperformance(exkalmanout, setup$xtrue, mcmcout)
enkalperf          <- filterperformance(enkalmanout, setup$xtrue, mcmcout)
partperf           <- filterperformance(partout, setup$xtrue, mcmcout)
precalibperf       <- filterperformance(precalib, setup$xtrue, mcmcout)
```

`tippingdetect` *Detect the tipping point for the true parameters.*

Description

The tipping point is the farthest year at which all the emissions can be reduced to zero without tipping the lake over

Usage

```
tippingdetect(input, setup, tipthres)
```

Arguments

`input`	The input object containing the initialized values of the filter's true parameters

Value

The year of tipping

Author(s)

Riddhi Singh

See Also

`lakesetup` `setlakedefault`

Examples

```
input           <- setlakedefault()
setup           <- lakesetup(input)
tippingyr       <- tippingdetect(input, setup)
```

`tippingpredict` *Project the future based on the output of a learning method until a given time*

Description

This function runs the lake model using output from the either of the six methods

Usage

```
tippingpredict(object, input, setup, calyr)
```

Arguments

`object`	A matrix or vector with initial condition(s)
`truth`	The true values of the variables (in case of a synthetic experiment or the demostration example)

Value

A list with values of both criteria

Author(s)

Riddhi Singh

See Also

`kalmanfitler` `extendedkalmanfitler` `ensemblekalmanfitler` `particlefitler`

Index