Package 'GPoM'

April 4, 2017

Type Package

Title Generalized Polynomial Modelling

Version 1.0

Date 2017-03-31

Author Sylvain Mangiarotti [aut], Flavie Le Jean [ctb], Malika Chassan [ctb], Laurent Drapeau [ctb], Mireille Huc [cre]

Maintainer Mireille Huc <mireille.huc@cesbio.cnes.fr>

Description Platform dedicated to the Global Modelling technique.

License CeCILL-2

LazyData TRUE

RoxygenNote 5.0.1

Depends R (>= 2.10), deSolve, rgl

Suggests testthat, knitr, rmarkdown

VignetteBuilder knitr

NeedsCompilation no

Repository CRAN

Date/Publication 2017-04-04 15:38:35 UTC

R topics documented:

`allToTest` *This is data to be included in my package*

Description

List of 6 models to be tested by autoGPoMoTest Each model $mToTest1, $mToTest2, etc. is provided as a matrix of coefficients of dimension 10 * 3 that corresponds to a formulation of three equations (nVar = 3) of maximal polynomial degree dMax = 2. Each column column corresponds to one equation. The order of the terms is given by `poLabs(nVar = 3, dMax = 2)`.

Usage

```
allToTest
```

Format

An object of class `list` of length 6.

Author(s)

Sylvain Mangiarotti, Mireille Huc

Examples

```
###########
# example #
###########
data("allToTest")
# 6 models are available in this list:
names(allToTest)
# The parameter of their formulation (nVar and dMax)
# can be retrieved:
nVar <- dim(allToTest$mToTest6)[2]
dMax <- p2dMax(nVar = 3, pMaxKnown = dim(allToTest$mToTest6)[1])
# Their equation can be edited as follows:
visuEq(nVar, dMax, allToTest$mToTest6, approx = 2)
```

`autoGPoMoSearch` *autoGPoMoSearch : automatic search of polynomial Equations*

Description

The algorithm aims to get an ensemble of equations to be tested from a given template of allowed terms. The maximum size of the equation depends on the model dimension nVar, and on the maximum polynomial degree dMax. The algorithm remove polynomial terms one by one, considering that a term is poorly useful when removing it leads to the smaller fitting degradation.

Usage

```
autoGPoMoSearch(data, dt, nVar = nVar, dMax = dMax, weight = NULL,
  show = 0, underSamp = NULL, filterReg = NULL)
```

Arguments

`data`	Input Time series: Each column corresponds to one input variable.
`dt`	Time sampling of the input series.
`nVar`	The model dimension expected. This parameter will be deduced from the input data (`series`) if `series` is a matrix. If `series` is a vector, the expected dimension nVar should be provided.
`dMax`	Maximum degree of the polynomial functions allowed in the model (see `poLabs`).
`weight`	Weighting function of input data series. By default uniform weight is applied. This weighting function can also be used to apply the analysis piecewise using zeros and ones.
`show`	Indicates (2) or not (0-1) the algorithm progress
`underSamp`	Points used for undersampling the data. For undersamp = 1 the complete time series is used. For undersamp = 2, only one data out of two is kept, etc.
`filterReg`	A vector that specifies the a priori structure of the the function of the output model in canonical form. The organisation of the filter follows the convention defined in `poLabs` and can be obtained using `poLabs(nVar,dMax)`. Value is 1 if the regressor is available, 0 if it is not.

Value

A list of two matrices:

`$filtMemo` describes the selected terms (1 if the term is used, 0 if not)

`$KMemo` provides the correspoing coefficients

Author(s)

Sylvain Mangiarotti, Flavie Le Jean

Examples

```
data('RosYco')
filt = autoGPoMoSearch(RosYco[,2], nVar = 3, dMax = 2,
                       dt = 1/125, show = 1)
# As an example, the equations of the fourth line has the following terms:
poLabs(nVar = 3, dMax = 2)[filt$filtMemo[5,] != 0]
# which coefficients correspond to
cbind(filt$KMemo[5,], poLabs(nVar = 3, dMax = 2))[filt$filtMemo[5,] != 0,]
```

`autoGPoMoTest` *autoGPoMoTest: Test models numerical integrability & classify the attractor reached at the convergence (from chosen initial conditions)*

Description

Test the numerical integrability of models (of polynomial structure) possibly obtained with function autoGPoMoSearch, and classify these models as divergent, Fixed Points, Periodic or not classificable (potentially chaotic).

Usage

```
autoGPoMoTest(data, tin = NULL, dt = NULL, nVar = nVar, dMax = dMax,
  show = 1, verbose = 1, allKL = allKL, IstepMin = 10,
  IstepMax = 10000, tooFarThreshold = 4, LimCyclThreshold = 0,
  fixedPtThreshold = 1e-08, method = "lsoda")
```

Arguments

`data`	Input Time series: Each column corresponds to one input variable.
`tin`	Input date vector which length should correspond to the variables of the input data (same number of lines).
`dt`	The time sampling of the input series.
`nVar`	The model dimension expected. This parameter will be deduced from the input data (`series`) if `series` is a matrix. If `series` is a vector, the expected dimension nVar should be provided.

dMax	Maximum degree of the polynomial functions allowed in the model (see poLabs).
show	Indicates (2) or not (0-1) the algorithm progress
verbose	Gives information (if set to 1) about the algorithm progress and keeps silent if set to 0
allKL	A list of the all the models $mToTest1, $mToTest2, etc. to be tested. Each model is provided as a matrix.
IstepMin	Minimum step of integration at the beginning of the analysis (by default IstepMin=10).
IstepMax	Maximum step of integration before stopping the analysis, with the interface this value can be changed during the analysis
tooFarThreshold	Divergence threshold, maximum times number that the model can be greater than the data standard deviation, without being removed from the analysis
LimCyclThreshold	Limit cycle threshold. Minimum neighbors distance between two integration steps, without being removed from the analysis
fixedPtThreshold	Limit cycle threshold. Minimum neighbors distance between two integration steps, without being removed from the analysis
method	The integration technique used for the numerical integration. Default is 'lsoda'. Others such as 'rk4' or 'ode45' may also be used. See package deSolve for details.

Value

A list containing:

$okMod A vector classifying the models: diverging models (0), periodic models of period-1 (-1), unclassified models (1).

$coeff A matrix with the coefficients of one selected model

$models A list of all the models to be tested $mToTest1, $mToTest2, etc. and all selected models $model1, $model2, etc.

$tout The time vector of the output time series (vector length corresponding to the longest numerical integration duration)

$stockoutreg A list of matrices with the integrated trajectories (variable X1 in column 1, X2 in 2, etc.) of all the models $model1, $model2, etc.

Author(s)

Sylvain Mangiarotti, Flavie Le Jean

Examples

```
#Examples
data('RosYco')
# Structure choice
```

```
data('allToTest')
outGPT <- autoGPoMoTest(RosYco, nVar= 3, dMax = 2, dt = 1/125, show=1,
                        allKL = allToTest, IstepMax = 60)
```

---

bDrvFilt *bDrvFilt : Build Derivative Filter*

Description

Build the Savitzky-Golay derivative filter (Savitzky-Golay, 1964).

Usage

```
bDrvFilt(nDrv, tstep, winL = 9)
```

Arguments

nDrv	The number of derivatives to be computed.
tstep	Time sampling.
winL	The local window length to be used for computing the derivatives [1].

Author(s)

Sylvain Mangiarotti

References

[1] Savitzky, A.; Golay, M.J.E., Smoothing and Differentiation of Data by Simplified Least Squares Procedures. Analytical Chemistry 36 (8), 1627-1639, 1964.

---

cano2M *canoToM : Converts canonical formulation into matricial formulation*

Description

Converts the vectorial formulation of canonical models into a matricial formulation. For both input, the list of terms follows the convention given in 'poLabs'.

Usage

```
cano2M(nVar, dMax, poly)
```

Arguments

nVar	The model dimension expected. This parameter will be deduced from the input data (series) if series is a matrix. If series is a vector, the expected dimension nVar should be provided.
dMax	Maximum degree of the polynomial functions allowed in the model (see poLabs).
poly	A vector of coefficients corresponding to the regressor of the canonical function

Author(s)

Sylvain Mangiarotti, Mireille Huc

Examples

```
polyTerms <- c(0.2,0,-1,0.5,0,0,0,0,0,0)
K <- cano2M(3,2,polyTerms)
visuEq(3,2,K)
```

---

compDeriv *compDeriv : Computes the successive derivatives of a time series vector*

Description

Compute the successive derivatives from one single time series, with the Savitzky-Golay approach (1964).

Usage

```
compDeriv(TS, nDrv, tstep, winL = 9)
```

Arguments

TS	A single time series provided as a single vector.
nDrv	The number of derivatives to be computed from the input series. The resulting number of time series obtained as an output will thus be nVar = nDrv + 1.
tstep	Time sampling of the input series.
winL	The local window length to be used for computing the derivatives [1-2].

Value

A matrix containing the original variable (as smoothed by the filtering process) and its nDrv first derivatives (note that winL values of the original time series will be lost both at the begining and the end of the time series due to boundary effect).

Author(s)

Sylvain Mangiarotti

References

[1] Savitzky, A.; Golay, M.J.E., Smoothing and Differentiation of Data by Simplified Least Squares Procedures. Analytical Chemistry 36 (8), 1627-1639, 1964.
[2] Steinier J., Termonia Y., Deltour, J. Comments on smoothing and differentiation of data by simplified least square procedure. Analytical Chemistry 44 (11): 1906-1909, 1972.

See Also

gloMoId, gPoMo, poLabs

---

d2pMax *d2pMax : provides pMax given dMax and nVar*

Description

Gives the maximum polynomial degree pMax given the number of polynomial terms dMax and the system dimension nVar

Usage

```
d2pMax(nVar, dMaxKnown)
```

Arguments

nVar	The model dimension expected. This parameter will be deduced from the input data (series) if series is a matrix. If series is a vector, the expected dimension nVar should be provided.
dMaxKnown	The number of polynoms terms to retrieve

Value

The maximum polynomial degree dMax used to code the polynomial

Author(s)

Sylvain Mangiarotti

See Also

gloMoId, gPoMo, poLabs

Examples

```
#############
# Example 1 #
#############
# Maximum polynomial degree ?
# number of variables:
nVar <- 3
# polynomial degree:
dMax <- 3
# The maximal polynomial degree used for coding the polynomial is:
d2pMax(nVar,dMax)
```

`derivODE2` *deriveODE2 : A Subfonction for the numerical integration of polynomial equations in the generic form defined by function* `poLabs`

Description

This function provides the one step integration of polynomial Ordinary Differential Equations (ODE). This function requires the function `ode` ("deSolve" package).

Usage

```
derivODE2(t, x, K, regS = NULL)
```

Arguments

`t`	All the dates for which the result of the numerical integration of the model will have to be provided
`x`	Current state vector (input from which the next state will be estimated)
`K`	is the model: each column corresponds to one equation which organisation is following the convention given by function `poLabs` which requires the definition of the model dimension `nVar` (i.e. the number of variables) and the maximum polynomial degree `dMax` allowed.
`regS`	Current states of each polynomial terms used in `poLabs`. These states can be deduced from the current state vector `x` (using function `regSeries`). When available, it can be provided as an input to avoid unecessary computation.

Author(s)

Sylvain Mangiarotti

#@export

detectP1limCycl *detectP1limCycl : Detect limit cycles of period-1*

Description

Detects the limit cycle from trajectories in a bidimensional projection

Usage

```
detectP1limCycl(data, LimCyclThreshold = 0.01, show = 2)
```

Arguments

data	A matrix of the trajectory in a 2D space (only the two first column are considered)
LimCyclThreshold	The detection threshold
show	Indicates the deepness of the feedback (from 0 to 2)

Value

indicates if a limit cycle is detected (1) or not (0)

Author(s)

Sylvain Mangiarotti

drvSucc *drvSucc : Computes the successive derivatives of a time series*

Description

Computes the successive derivatives from one single time series, using the Savitzky-Golay algorithm (1964).

Usage

```
drvSucc(tin = NULL, serie, nDeriv, weight = NULL, tstep = NULL,
  winL = 9)
```

Arguments

`tin`	Input date vector which length should correspond to the variables of the input data (same number of lines).
`serie`	A single time series provided as a single vector.
`nDeriv`	The number of derivatives to be computed from the input `series`. The resulting number of time series obtained as an output will thus be nVar = nDeriv + 1.
`weight`	Weighting function of input data series. By default uniform weight is applied. This weighting function can also be used to apply the analysis piecewise using zeros and ones.
`tstep`	Time sampling of the input `series`. Used only if time vector `tin` is not provided.
`winL`	Number (odd) of points of the local window used for computing the derivatives along the input time series (`series`). The Savitzky-Golay filter is used for this purpose [1,2].

Value

A list containing: $serie The original time serie $tin The time vector corresponding to the original time series $tstep The time step (assumed to be invariant) $tout The time vector of the output series ^seriesDeriv A matrix containing the original variable (as smoothed by the filtering process) and its `nDeriv` + 1 first derivatives (note that winL values of the original time series will be lost both (winL - 1)/2 at the begining and at the end of the time series due to boundary effect).

Author(s)

Sylvain Mangiarotti, Mireille Huc

References

[1] Savitzky, A.; Golay, M.J.E., Smoothing and Differentiation of Data by Simplified Least Squares Procedures. Analytical Chemistry 36 (8), 1627-1639, 1964.
[2] Steinier J., Termonia Y., Deltour, J. Comments on smoothing and differentiation of data by simplified least square procedure. Analytical Chemistry 44 (11): 1906-1909, 1972.

See Also

`gloMoId`, `gPoMo`, `poLabs`

Examples

```
#############
# Example 1 #
#############
tin <- seq(0, 5, by = 0.01)
data <- 2 * sin(5*tin)
dev.new()
par(mfrow = c(3, 1))
```

```
# numerical derivation
drv <- drvSucc(tin = tin, nDeriv = 2, serie = data, winL = 5)
#
# plot original and filtered series
plot(tin, data, type='l', col = 'black', xlab = 't', ylab = 'x(t)')
lines(drv$tout, drv$seriesDeriv[,1], lty = 3, lwd = 3, col = 'green')
#
# analytic 1st derivative
firstD <- 10 * cos(5 * tin)
# plot both
plot(tin, firstD, type = 'l', col = 'black', xlab = 't', ylab = 'dx/dt')
lines(drv$tout, drv$seriesDeriv[,2], lty = 3, lwd = 3, col = 'green')
#
# analytic 2nd derivative
scdD <- -50 * sin(5 * tin)
# plot both
plot(tin, scdD, type = 'l', col = 'black', xlab = 't', ylab = 'd2x/dt2')
lines(drv$tout, drv$seriesDeriv[,3], lty=3, lwd = 3, col = 'green')

#############
# Example 2 #
#############
# load data
data("Ross76")
tin <- Ross76[,1]
data <- Ross76[,2]

# Apply derivatives
drvOut <- drvSucc(tin, data, nDeriv=4)
dev.new()
par(mfrow = c(3, 1))
# original and smoothed variable:
plot(drvOut$tin, drvOut$serie,
     type='p', cex = 1, xlab = 'time', ylab = 'x(t)')
lines(drvOut$tout, drvOut$seriesDeriv[,1], type='p', col='red')
lines(drvOut$tout, drvOut$seriesDeriv[,1], type='l', col='red')
# 1st derivative:
plot(drvOut$tout, drvOut$seriesDeriv[,2],
     type='p', col='red', xlab = 'time', ylab = 'dx(t)/dt')
lines(drvOut$tout, drvOut$seriesDeriv[,2], type='l', col='red')
# 2nd derivative:
plot(drvOut$tout, drvOut$seriesDeriv[,3],
     type='p', col='red', xlab = 'time', ylab = 'd2x(t)/dt2')
lines(drvOut$tout, drvOut$seriesDeriv[,3], type='l', col='red')
```

`findAllSets` *findAllSets : find all potential systems (a set of equations) from a given ensemble of potential equations*

Description

For each equation to be retrieved, an ensemble of potential formulation is given. For instance, three formulations are provided for equation (1), one for equation (2) and two for equation (3). In this case, six possible sets can be obtained from it. The aim of this program is to formulate all the potential systems from the provided formulation.

Usage

```
findAllSets(allFilt, nS = c(3), nPmin = 1, nPmax = 14)
```

Arguments

allFilt	A list with: - a matrix allFilt$Xi of possible formulations for each variable Xi - a vector allFilt$Npi providing the number of parameters of each formulation Xi
nS	A vector providing the number of dimensions that will be used for each input variables (see Examples 1 and 2). The dimension of the resulting model will be nVar = sum(nS).
nPmin	Corresponds to the minimum number of parameters (and thus of regressor) of the model
nPmax	Corresponds to the maximum number of parameters (and thus of regressor) of the model

Author(s)

Sylvain Mangiarotti

Examples

```
#############
# Example 1 #
#############
allFilt <- list()
allFilt$Np1 <- 1         # only one formulation with one single parameter
allFilt$Np2 <- c(3, 4)   # two potential formulations, one with respectively
# three and four parameters
allFilt$Np3 <- c(2, 4)   # two potential formulations, one with respectively
# two and four parameters
# formulations for variables Xi:
allFilt$X1 <- t(as.matrix(c(0,0,0,1,0,0,0,0,0,0)))
allFilt$X2 <- t(matrix(c(0,-0.85,0,-0.27,0,0,0,0.46,0,0,
                         0,-0.64,0,0,0,0,0,0.43,0,0),
                       ncol=2, nrow=10))
allFilt$X3 <- t(matrix(c(0, 0.52,  0, -1.22e-05,  0, 0, 0.99, 5.38e-05, 0, 0,
                         0, 0.52, 0, 0, 0, 0, 0.99, 0, 0, 0),
                       ncol=2, nrow=10))
findAllSets(allFilt, nS=c(3), nPmin=1, nPmax=14)
```

gloMoId *gloMoId : global Model Identification*

Description

Algorithm for global modelling in a Polynomial and canonical formulation of Ordinary Differential Equations. Univariate Global modeling aims to obtain multidimensional models from single time series (Gouesbet & Letellier 1994, Mangiarotti et al. 2012). An example of such application can be found in Mangiarotti et al. (2014) For a multivariate application, see GPoMo (Mangiarotti 2015, Mangiarotti et al. 2016).

Example:
For a model dimension nVar=3, the global model will read:
$dX1/dt = X2$
$dX2/dt = X3$
$dX3/dt = P(X1, X2, X3)$.

Usage

```
gloMoId(series, tin = NULL, dt = NULL, nVar = NULL, dMax = 1,
  weight = NULL, show = 1, filterReg = NULL, winL = 9)
```

Arguments

series	The original data set: either a single vector corresponding to the original variable; Or a matrix containing the original variable in the first column and its successive derivatives in the next columns. In the latter case, for the construction of n-dimensional model, series should have $nVar + 1$ columns since one more derivative will be necessary to identify the model parameters. Variable nVar will be set equal to n. In the former case, that is when only a single vector is provided, the derivatives will be automatically recomputed. Therefore, the dimension nVar expected for the model has to be provided.
tin	Input date vector which length should correspond to the variables of the input data (same number of lines).
dt	The time sampling of the input series.
nVar	The model dimension expected. This parameter will be deduced from the input data (series) if series is a matrix. If series is a vector, the expected dimension nVar should be provided.
dMax	Maximum degree of the polynomial functions allowed in the model (see poLabs).
weight	Weighting function of input data series. By default uniform weight is applied. This weighting function can also be used to apply the analysis piecewise using zeros and ones.
show	Indicates (2) or not (0-1) the algorithm progress

`filterReg` A vector that specifies the a priori structure of the the function of the output model in canonical form. The organisation of the filter follows the convention defined in `poLabs` and can be obtained using `poLabs(nVar,dMax)`. Value is 1 if the regressor is available, 0 if it is not.

`winL` Total number of points used for computing the derivatives of the input time series (data). This parameter will be used as an input in function `drvSucc`.

Value

`A` list of five elements :

`init` Initial time series and the successive derivatives used in the modeling.

`filterReg` Structure of the output model. Value is 1 if the regressor is available, 0 if it is not. The corresponding terms can be obtained with poLabs(dMax,dMax).

`codereconstr` Simulated trajectory: the initial variable and its derivatives obtained by integrating the equations of the identified model.

`codeK` Values of the identified coefficients corresponding to the regressors defined in filterReg.

`coderesTot` Residual signal of the model.

`coderesSsMod` Residual signal of the closer submodels.

Author(s)

Sylvain Mangiarotti, Laurent Drapeau, Mireille Huc

References

[1] Gouesbet G., Letellier C., Global vector-field reconstruction by using a multivariate polynomial L2 approximation on nets, Physical Review E, 49 (6), 4955-4972, 1994.
[2] Mangiarotti S., Coudret R., Drapeau L., & Jarlan L., Polynomial search and global modeling : Two algorithms for modeling chaos, Physical Review E, 86, 046205, 2012.
[3] Mangiarotti S., Drapeau L. & Letellier C., Two chaotic models for cereal crops observed from satellite in northern Morocco. Chaos, 24(2), 023130, 2014.
[4] Mangiarotti S., Low dimensional chaotic models for the plague epidemic in Bombay (18961911), Chaos, Solitons & Fractals, 81(A), 184-196, 2015.
[5] Mangiarotti S., Peyre M. & Huc M., A chaotic model for the epidemic of Ebola Virus Disease in West Africa (2013-2016). Chaos, 26, 113112, 2016.

See Also

`gPoMo`, `autoGPoMoSearch`, `autoGPoMoTest`, `poLabs`

Examples

```
#############
# Example 1 #
#############
# load data
#Example 2
data("Ross76")
tin <- Ross76[,1]
data <- Ross76[,2:3]

# Application with a simplified structure
reg <- gloMoId(data[0:500,2], dt=1/100, nVar=2, dMax=2, show=0)

#############
# Example 2 #
#############
# load data
data(NDVI)

# Definition of the Model structure
terms <- c(1, 0, 0, 0, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1)
poLabs(3,3,terms==1)
reg <- gloMoId(NDVI [,1:1], dt=1/125, nVar=3, dMax=3,
               show=0, filterReg=terms==1)


## Not run:
#############
# Example 3 #
#############
# load data
data("Ross76")
# time vector
tin <- Ross76[1:500,1]
# single time series
series <- Ross76[1:500,3]
# some noise is added
series[1:100] <- series[1:100] + 0.01 * runif(1:100, min = -1, max = 1)
series[301:320] <- series[301:320] + 0.05 * runif(1:20, min = -1, max = 1)
# weighting function
W <- tin * 0 + 1
W[1:100] <- 0  # the first fourty values will not be considered
W[301:320] <- 0  # twenty other values will not be considered either
reg <- gloMoId(series, dt=1/100, weight = W, nVar=3, dMax=2, show=1)
visuEq(3, 2, reg$K, approx = 4)
# first weight which value not equal to zero:
i1 = which(reg$finalWeight == 1)[1]
v0 <-  reg$init[i1,1:3]

reconstr <- numicano(nVar=3, dMax=2, Istep=5000, onestep=1/250, PolyTerms=reg$K,
                     v0=v0, method="ode45")
```

```
plot(reconstr$reconstr[,2], reconstr$reconstr[,3], type='l', lwd = 3,
                        main='phase portrait', xlab='time t', ylab = 'x(t)', col='orange')
# original data:
lines(reg$init[,1], reg$init[,2], type='l',
      main='phase portrait', xlab='x', ylab = 'dx/dt', col='black')
# initial condition
lines(v0[1], v0[2], type = 'p', col = 'red')


## End(Not run)
```

GPoM *GPoM package: Generalized Polynomial Modelling*

Description

GPoM is a platform dedicated to the Global Modelling technique wjich aim is to obtain deterministic models of Ordinary Differential Equations from observational time series. It can be used to detect couplings between observed variables and for the inference of causal networks. It can also be applied to single time series. The present package focuses on models in Ordinary Differential Equations of polynomial form. Although designed for modelling chaos, it can be also applied to any linear or nonlinear dynamical system for which a deterministic behavior is expected.

Note

FOR USERS
This package was developped in Centre d'Etudes Spatiales de la Biosphere http://www.cesbio.ups-tlse.fr/. If you want to apply this package to single time series, please quote [5]. If you want to apply it to multivariate time series, please quote [9]. If you want to apply it to infer causal relations among time series, please quote [7].

HISTORICAL BACKGROUND
The global modelling technique was initiated during the early 1990s [1-3]. It takes its background from the Theory of Nonlinear Dynamical Systems. The approach became applicable to the analysis of real world environmental behaviours by the end of the 2000s [4-6]. Recent works have shown that the approach could be applied to numerous other dynamical behaviors [7-9]. Global modelling aims to obtain deterministic models directly from observed time series.

Author(s)

Sylvain Mangiarotti, Flavie Le Jean, Malika Chassan, Laurent Drapeau, Mireille Huc.

Maintainer: M. Huc <mireille.huc@cesbio.cnes.fr>

References

[1] J. P. Crutchfield and B. S. McNamara, 1987. Equations of motion from a data series, Complex Syst. 1, 417-452.
[2] Gouesbet G., Letellier C., 1994. Global vector-field reconstruction by using a multivariate polynomial L2 approximation on nets, Physical Review E, 49 (6), 4955-4972.

[3] C. Letellier, L. Le Sceller, E. Marechal, P. Dutertre, B. Maheu, G. Gouesbet, Z. Fei, and J. L. Hudson, 1995. Global vector field reconstruction from a chaotic experimental signal in copper electrodissolution, Phys. Rev. E 51, 4262-4266.
[4] J. Maquet, C. Letellier, and L. A. Aguirre, 2007. Global models from the Canadian Lynx cycles as a first evidence for chaos in real ecosystems, J. Math. Biol. 55(1), 21-39.
[5] Mangiarotti S., Coudret R., Drapeau L., & Jarlan L., 2012. Polynomial search and global modeling : Two algorithms for modeling chaos, Physical Review E, 86, 046205.
[6] Mangiarotti S., Drapeau L. & Letellier C., 2014. Two chaotic models for cereal crops observed from satellite in northern Morocco. Chaos, 24(2), 023130.
[7] Mangiarotti S., 2015. Low dimensional chaotic models for the plague epidemic in Bombay (1896-1911). Chaos, Solitons and Fractals, 81A, 184-186.
[8] Mangiarotti S., Peyre M. & Huc M., A chaotic model for the epidemic of Ebola Virus Disease in West Africa (2013-2016). Chaos, 26, 113112, 2016.
[9] Mangiarotti S., 2014. Modelisation globale et Caracterisation Topologique de dynamiques environnementales - de l'analyse des enveloppes fluides et du couvert de surface de la Terre a la caracterisation topolodynamique du chaos. Habilitation to Direct Research, University of Toulouse 3, France.

---

gPoMo *gPoMo: Generalized Polynomial Modeling*

Description

Algorithm for a Generalized Polynomial formulation of multivariate Global Modeling. Global modeling aims to obtain multidimensional models from single time series [1-2]. In the generalized (polynomial) formulation provided in this function, it can also be applied to multivariate time series [3-4].

Example:
Note that nS provides the number of dimensions used from each variable

case I
For nS=c(2,3) means that 2 dimensions are reconstructed from variable 1: the original variable X1 and its first derivative X2), and 3 dimensions are reconstructed from variable 2: the original variable X3 and its first and second derivatives X4 and X5. The generalized model will thus be such as:
$dX1/dt = X2$
$dX2/dt = P1(X1, X2, X3, X4, X5)$
$dX3/dt = X4$
$dX4/dt = X5$
$dX5/dt = P2(X1, X2, X3, X4, X5)$.

case II
For nS=c(1,1,1,1) means that only the original variables X1, X2, X3 and X4 will be used. The generalized model will thus be such as:
$dX1/dt = P1(X1, X2, X3, X4)$
$dX2/dt = P2(X1, X2, X3, X4)$
$dX3/dt = P3(X1, X2, X3, X4)$

$dX4/dt = P4(X1, X2, X3, X4).$

Usage

```
gPoMo(data, tin = NULL, dtFixe = NULL, dMax = 2, nS = c(3), winL = 9,
  weight = NULL, show = 1, verbose = 1, underSamp = NULL, EqS = NULL,
  IstepMin = 2, IstepMax = 2000, nPmin = 1, nPmax = 14,
  method = "lsoda")
```

Arguments

data	Input Time series: Each column corresponds to one input variable.
tin	Input date vector which length should correspond to the variables of the input data (same number of lines).
dtFixe	Time step used for the analysis. In principle, it should correspond to the time step of the input data. Modification of this time step may be used to stabilize the numerical computation. Such modification should be performed in full consciouness that it will lead to change the time scale accordingly.
dMax	Maximum degree of the polynomial functions allowed in the model (see poLabs).
nS	A vector providing the number of dimensions that will be used for each input variables (see Examples 1 and 2). The dimension of the resulting model will be nVar = sum(nS).
winL	Total number of points used for computing the derivatives of the input time series (data). This parameter will be used as an input in function drvSucc.
weight	Weighting function of input data series. By default uniform weight is applied. This weighting function can also be used to apply the analysis piecewise using zeros and ones.
show	Indicates (2) or not (0-1) the algorithm progress
verbose	Gives information (if set to 1) about the algorithm progress and keeps silent if set to 0
underSamp	Points used for undersampling the data. For undersamp = 1 the complete time series is used. For undersamp = 2, only one data out of two is kept, etc.
EqS	Model template including all allowed regressors. Each column corresponds to one equation. Each line corresponds to one regressor following the convention given by poLabs(nVar,dMax).
IstepMin	Minimum step of integration at the beginning of the analysis (by default IstepMin=10).
IstepMax	Maximum step of integration before stopping the analysis, with the interface this value can be changed during the analysis
nPmin	Corresponds to the minimum number of parameters (and thus of regressor) of the model
nPmax	Corresponds to the maximum number of parameters (and thus of regressor) of the model

`method` The integration technique used for the numerical integration. Default is 'lsoda'. Others such as 'rk4' or 'ode45' may also be used. See package deSolve for details.

Value

A list containing:

`$tin` The time vector of the input time series

`$inputdata` The input time series

`$tfiltdata` The time vector of the filtered time series (boudary removed)

`$filtdata` A matrix of the filtered time series with its derivatives

`$okMod` A vector classifying the models: diverging models (0), periodic models of period-1 (-1), unclassified models (1).

`$coeff` A matrix with the coefficients of one selected model

`$models` A list of all the models to be tested `$mToTest1`, `$mToTest2`, etc. and all selected models `$model1`, `$model2`, etc.

`$tout` The time vector of the output time series (vector length corresponding to the longest numerical integration duration)

`$stockoutreg` A list of matrices with the integrated trajectories (variable `X1` in column 1, `X2` in 2, etc.) of all the models `$model1`, `$model2`, etc.

Author(s)

Sylvain Mangiarotti, Flavie Le Jean, Mireille Huc

References

[1] Gouesbet G. & Letellier C., 1994. Global vector-field reconstruction by using a multivariate polynomial L2 approximation on nets, Physical Review E, 49 (6), 4955-4972.
[2] Mangiarotti S., Coudret R., Drapeau L. & Jarlan L., Polynomial search and Global modelling: two algorithms for modeling chaos. Physical Review E, 86(4), 046205.
[3] Mangiarotti S., Le Jean F., Huc M. & Letellier C., Global Modeling of aggregated and associated chaotic dynamics. Chaos, Solitons and Fractals, 83, 82-96.
[4] S. Mangiarotti, M. Peyre & M. Huc, 2016. A chaotic model for the epidemic of Ebola virus disease in West Africa (2013-2016). Chaos, 26, 113112.

See Also

`gloMoId`, `autoGPoMoSearch`, `autoGPoMoTest`

Examples

```
#Example 1
data("Ross76")
tin <- Ross76[,1]
data <- Ross76[,3]
```

```
dev.new()
out1 <- gPoMo(data, tin=tin, dMax = 2, nS=c(3), show = 1,
              IstepMax = 1000, nPmin = 9, nPmax = 11)
visuEq(3, 2, out1$models$model1, approx = 4)

## Not run:
#Example 2
data("Ross76")
tin <- Ross76[,1]
data <- Ross76[,2:4]
dev.new()
out2 <- gPoMo(data, tin=tin, dMax = 2, nS=c(1,1,1), show = 1,
              IstepMin = 10, IstepMax = 3000, nPmin = 7, nPmax = 8)
# the simplest model able to reproduce the observed dynamics is model #5
visuEq(3, 2, out2$models$model5, approx = 4) # the original Rossler system is thus retrieved

## End(Not run)

## Not run:
#Example 3
data("Ross76")
tin <- Ross76[,1]
data <- Ross76[,2:3]
# model template:
EqS <- matrix(1, ncol = 3, nrow = 10)
EqS[,1] <- c(0,0,0,1,0,0,0,0,0,0)
EqS[,2] <- c(1,1,0,1,0,1,1,1,1,1)
EqS[,3] <- c(0,1,0,0,0,0,1,1,0,0)
visuEq(3, 2, EqS, substit = c('X','Y','Z'))
dev.new()
out3 <- gPoMo(data, tin=tin, dMax = 2, nS=c(2,1), show = 1,
      EqS = EqS, IstepMin = 10, IstepMax = 2000,
      nPmin = 9, nPmax = 11)

## End(Not run)

## Not run:
#Example 4
data(sprottK)
data(rossler)
data <- cbind(rossler,sprottK)[1:400,]
dev.new()
out4 <- gPoMo(data, dt=1/20, dMax = 2, nS=c(1,1,1,1,1,1),
             show = 1, method = 'rk4',
             IstepMin = 2, IstepMax = 3, nPmin = 13, nPmax = 13)
# the original Rossler (variables x, y and z) and Sprott (variables X, Y and Z)
# systems are retrieved:
visuEq(6, 2, out4$models$model347, approx = 4,
           substit = c('x', 'y', 'z', 'X', 'Y', 'Z'))
   # to check the robustness of the model, the integration duration
   # should be chosen longer (at least IstepMax = 4000)

## End(Not run)
```

GSproc *GSproc : Gram-Schmidt procedure*

Description

Computes regressors coefficients using the Gram-Schmidt procedure.

Usage

```
GSproc(polyK, ivec, weight = NULL)
```

Arguments

polyK	One list including $Y and $phy with: $Y a matrix for which the ith column will be used to add one orthogonal vector to the (i-1)th vectors of the current orthogonal base; and $phy such as the current orthogonal base is given by the (i-1)th first columns of matrix polyK$phy.
ivec	Defines i, the current vector of polyK$Y and the current orthogonal base of pParam$phy.
weight	The weighing vector.

Value

uNew The model parameterization, that is: The residual orthogonal vector that can be included into the current orthogonal base. If the current base is empty, uNew is equal to the input vector of $Y; if the base is complete, uNew equals 0.

Author(s)

Sylvain Mangiarotti

#@export

lorenz-1963 time series

Data included in package GPoM

Description

The Lorenz system is the 3-dimensional chaotic system
$dx/dt = \sigma y - \sigma z$
$dy/dt = -xz + rx - y$
$dz/dt = xy - bz$,
discovered by Edouard N. Lorenz in 1963 [1]. The following parameters and initial conditions were used in the present data set:
$\sigma = 10, b = 8/3, r = 28$
and $(x0, y0, z0) = (0, 1, 1.05)$.
The following four columns are provided:
(1) time t, (2) x(t), (3) y(t) and (4) z(t).
For this parameterization, the Lorenz system produces a chaotic behavior.

Usage

```
lorenz
```

Format

An object of class `data.frame` with 10000 rows and 3 columns.

Author(s)

Sylvain Mangiarotti, Flavie Le Jean, Malika Chassan, Mireille Huc.

References

[1] Lorenz, Edward Norton (1963). "Deterministic nonperiodic flow". Journal of the Atmospheric Sciences. 20 (2): 130-141.

NDVI

This is data to be included in my package

Description

A time series of 28 years of Normalized Difference Vegetation Index measured from space based on the Advanced Very High Resolution Radiometer (AVHRR) sensor from 1982 to 2008 (see reference [1] for details).

Usage

```
NDVI
```

Format

An object of class `data.frame` with 9618 rows and 4 columns.

Author(s)

Sylvain Mangiarotti, Flavie Le Jean

References

[1] Mangiarotti S., Drapeau L. & Letellier C., 2014. Two chaotic models for cereal crops observed from satellite in northern Morocco.

`numicano` *Numerical Integration of models in ODE of polynomial form*

Description

Function for the numerical integration of canonical Ordinary Differential Equations in polynomial form.

Usage

```
numicano(nVar, dMax, Istep = 1000, onestep = 1/125, KL = NULL,
  PolyTerms = NULL, v0 = NULL, method = "ode45")
```

Arguments

`nVar`	The model dimension expected. This parameter will be deduced from the input data (`series`) if `series` is a matrix. If `series` is a vector, the expected dimension nVar should be provided.
`dMax`	Maximum degree of the polynomial functions allowed in the model (see `poLabs`).
`Istep`	Number of integration time steps
`onestep`	Time step length
`KL`	Matricial formulation of the model to integrate numerically
`PolyTerms`	Vectorial formulation of the model (only for models of canonical form)
`v0`	Initial conditions (a vector which length should correspond to the model dimension `nVar`)
`method`	Integration method (See deSolve), by default `method = 'ode45'`.

Value

A list of two variables:

$KL the model in its matrix formulation

$reconstr the integrated trajectory (first column is the time, next columns are the model variables)

Author(s)

Sylvain Mangiarotti

Examples

```
##############
# Example 1 #
##############
# For a model of general form (here the rossler model)
# model dimension:
nVar = 3
# maximal polynomial degree
dMax = 2
# Number of parameter number (by default)
pMax <- d2pMax(nVar, dMax)
# convention used for the model formulation
poLabs(nVar, dMax)
# Definition of the Model Function
a = 0.520
b = 2
c = 4
Eq1 <- c(0,-1, 0,-1, 0, 0, 0, 0, 0, 0)
Eq2 <- c(0, 0, 0, a, 0, 0, 1, 0, 0, 0)
Eq3 <- c(b,-c, 0, 0, 0, 0, 0, 1, 0, 0)
K <- cbind(Eq1, Eq2, Eq3)
# Edition of the equations
visuEq(nVar, dMax, K)
# initial conditions
v0 <- c(-0.6, 0.6, 0.4)
# model integration
reconstr <- numicano(nVar, dMax, Istep=1000, onestep=1/50, KL=K,
                     v0=v0, method="ode45")
# Plot of the simulated time series obtained
dev.new()
plot(reconstr$reconstr[,2], reconstr$reconstr[,3], type='l',
      main='phase portrait', xlab='x(t)', ylab = 'y(t)')

## Not run:
##############
# Example 2 #
##############
# For a model of canonical form
# model dimension:
```

```
nVar = 4
# maximal polynomial degree
dMax = 3
# Number of parameter number (by default)
pMax <- d2pMax(nVar, dMax)
# Definition of the Model Function
PolyTerms <- c(281000, 0, 0, 0, -2275, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
               861, 0, 0, 0, -878300, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)
# terms used in the model
poLabs(nVar, dMax, PolyTerms!=0)
# initial conditions
v0 <- c(0.54, 3.76, -90, -5200)
# model integration
reconstr <- numicano(nVar, dMax, Istep=500, onestep=1/250, PolyTerms=PolyTerms,
                     v0=v0, method="ode45")
# Plot of the simulated time series obtained
plot(reconstr$reconstr[,2], reconstr$reconstr[,3], type='l',
     main='phase portrait', xlab='x', ylab = 'dx/dt')
# Edition of the equations
visuEq(nVar, dMax, reconstr$KL)

## End(Not run)
```

`numinoisy` *numinoisy : Generates time series of deterministic-behaviour stochatically-perturbed*

Description

Generates time series from Ordinary Differential Equations perturbed by multiplicative and/or additive noises

Usage

```
numinoisy(x0, t, K, varData = NULL, txVarBruitA = NULL,
  txVarBruitM = NULL, varBruitA = NULL, varBruitM = NULL, taux = NULL,
  freq = NULL, variables = NULL, method = NULL)
```

Arguments

`x0`	Initial conditions. Should be a vector which size must be equal to the model dimension `dim(K)[2]` (the number of variables of the model defined by matrix K).
`t`	A vector providing all the dates for which the output are expected.
`K`	The Ordinary Differential Equations used to model the dynamics. The number of column should correspond to the number of variables, the number of lines should correspond to the number of parameters ordered following the convention defined by `poLabs(nVar,dMax)`, with nVar the number of variables and dMax the maximum polynomial degree allowed.

varData	A vector of size nVar providing the caracteristic variances of each variable of the dynamical systems in ODE defined by matrix K. If not provided, this variance is automatically estimated.
txVarBruitA	A vector defining the ratio of ADDITIVE noise for each variable of the dynamical system in ODE. The additive noise is added at the end of the numerical integration process. The ratio is defined relatively to the signal variance of each variable.
txVarBruitM	A vector defining the ratio of MULTIPLICATIVE noise for each variable of the dynamical system in ODE. The multiplicative noise is a perturbation added at each numerical integration step. The ratio is defined relatively to the signal variance of each variable.
varBruitA	A vector defining the variance of ADDITIVE noise for each variable of the dynamical system in ODE. The additive noise is added at the end of the numerical integration process.
varBruitM	A vector defining the ratio of MULTIPLICATIVE noise for each variable of the dynamical system in ODE. The multiplicative noise is a perturbation added at each numerical integration step.
taux	Generates random gaps in time series. Parameter taux defines the ratio of data to be kept (e.g. for $taux = 0.75$, 75 percents of the data are kept).
freq	Undesamples the time series. Parameter freq defines the periodicity of data kept (e.g. for $freq = 3$, 1 data out of 3 is kept).
variables	Defines which variables must be generated.
method	Defines the numerical integration method to be used. See function ode from package deSolve for details.

Value

A list of two variables:

$donnees the integrated trajectory (first column is the time, next columns are the model variables)

$idices

$signal_init the integrated trajectory (first column is the time, next columns are the model variables)

$bruitM The level of multiplicative noise

$bruitA The level of additive noise

$vectBruitM The multiplicative noise used to produce the time series

$vectBruitA The additive noise used to produce the time series

$ecart_type The level standard deviation

Author(s)

Sylvain Mangiarotti, Malika Chassan

Examples

```
#############
# Example 1 #
#############
# Rossler Model formulation
# The model dimension
nVar = 3
 # maximal polynomial degree
dMax = 2
a = 0.520
b = 2
c = 4
Eq1 <- c(0,-1, 0,-1, 0, 0, 0, 0, 0, 0)
Eq2 <- c(0, 0, 0, a, 0, 0, 1, 0, 0, 0)
Eq3 <- c(b,-c, 0, 0, 0, 0, 0, 1, 0, 0)
K <- cbind(Eq1, Eq2, Eq3)
# Edit the equations
visuEq(nVar, dMax, K)
# initial conditions
v0 <- c(-0.6, 0.6, 0.4)
# output time required
timeOut = (0:1000)/50
# variance of additive noise
varBruitA = c(0,0,0)^2
# variance of multiplitive noise
varBruitM = c(2E-2, 0, 2E-2)^2
# numerical integration with noise
intgr <- numinoisy(v0, timeOut, K, varBruitA = varBruitA, varBruitM = varBruitM, freq = 1)
# Plot of the simulated time series obtained
dev.new()
plot(intgr$donnees[,2], intgr$donnees[,3], type='l',
      main='phase portrait', xlab='x(t)', ylab = 'y(t)')
dev.new()
par(mfrow = c(3, 1))
plot(intgr$donnees[,1], intgr$donnees[,2], type='l',
      main='phase portrait', xlab='x(t)', ylab = 'y(t)')
lines(intgr$donnees[,1], intgr$vectBruitM[,2]*10, type='l',
      main='phase portrait', xlab='x(t)', ylab = 'e(t)*10', col='red')
plot(intgr$donnees[,1], intgr$donnees[,3], type='l',
      main='phase portrait', xlab='x(t)', ylab = 'y(t)')
lines(intgr$donnees[,1], intgr$vectBruitM[,3]*10, type='l',
      main='phase portrait', xlab='x(t)', ylab = 'e(t)*10', col='red')
plot(intgr$donnees[,1], intgr$donnees[,4], type='l',
      main='phase portrait', xlab='x(t)', ylab = 'y(t)')
lines(intgr$donnees[,1], intgr$vectBruitM[,4]*10, type='l',
```

```
main='phase portrait', xlab='x(t)', ylab = 'e(t)*10', col='red')
```

---

odeBruitMult2 *odeBruitMult2 : For the numerical integration of ordinary differential equations with multiplicative noise.*

Description

A function for the numerical integration of Ordinary Differential Equations provided in a generic polynomial form. Model formulation follows the convention defined by function poLabs.

Usage

```
odeBruitMult2(x0, t, K, varData = NULL, txVarBruitM = NULL,
  varBruitM = NULL, method = NULL)
```

Arguments

x0	Initial conditions
t	All the dates for which the result of the numerical integration of the model will have to be provided
K	is the model: each column corresponds to one equation which organisation is following the convention given by function poLabs which requires the definition of the model dimension nVar (i.e. the number of variables) and the maximum polynomial degree dMax allowed.
varData	A vector of size nVar providing the caracteristic variances of each variable of the dynamical systems in ODE defined by matrix K. If not provided, this variance is automatically estimated.
txVarBruitM	A vector defining the ratio of MULTIPLICATIVE noise for each variable of the dynamical system in ODE. The multiplicative noise is a perturbation added at each numerical integration step. The ratio is defined relatively to the signal variance of each variable.
varBruitM	A vector defining the ratio of MULTIPLICATIVE noise for each variable of the dynamical system in ODE. The multiplicative noise is a perturbation added at each numerical integration step.
method	Numerical method used in the integration process. (see ode function in deSolve package for details).

Author(s)

Sylvain Mangiarotti, Malika Chassan

#@export

p2dMax *p2dMax : provides dMax given pMax and nVar*

Description

Search the maximum polynomial degree dMax given the number of polynomial terms pMax and the system dimension nVar

Usage

```
p2dMax(nVar, pMaxKnown)
```

Arguments

nVar	The model dimension expected. This parameter will be deduced from the input data (series) if series is a matrix. If series is a vector, the expected dimension nVar should be provided.
pMaxKnown	The number of polynoms terms to retrieve

Value

The maximum polynomial degree dMax used to code the polynomial

Author(s)

Sylvain Mangiarotti, Laurent Drapeau

See Also

gloMoId, gPoMo, poLabs

Examples

```
#############
# Example 1 #
#############
# Maximum polynomial degree ?
# number of variables:
nVar <- 3
# size of the polynomial vector:
pMax <- 10
# The maximal polynomial degree used for coding the polynomial is:
p2dMax(nVar,pMax)

#############
# Example 2 #
#############
# for pMax = 462 and nVar = 6, then dMax is:
p2dMax(6,462)
```

```
# indeed:
length(poLabs(nVar=6, dMax=5))
```

---

paramId *paramId : parameter Identification*

Description

Estimate polynomial coefficients.

Usage

```
paramId(allForK, drv, weight)
```

Arguments

allForK	The list of input parameters
drv	the derivative (on the equation left hand)
weight	The weighting series

Value

allForK The initial list completed with the model parameters.

Author(s)

Sylvain Mangiarotti

---

poLabs *poLabs : polynomial Labels*

Description

Edits the polynomial labels for chosen dimension nVar and maximum polynomial degree dMax.

Usage

```
poLabs(nVar, dMax, findIt = NULL, Xnote = "X")
```

Arguments

nVar	The number of variables
dMax	The maximum degree allowed for the polynomial
findIt	A vector of selected terms.
Xnote	Notation used for the variable, by default Xnote = 'X'.

Value

`lbls` A vector of characters. Each element is the expression of a regressor such as $X_1^2 X_3 X_4$

Author(s)

Sylvain Mangiarotti

See Also

visuEq

Examples

```
#Regressor order for three variables \eqn{(X1,X2,X3)} (nVar = 3) for a maximum
#polynomial degree equal to 2 (dMax = 2): poLabs(3,2)
#and for two variables only : poLabs(2,2)

# For a quadratic equation of two variables,
# the polynomial \deqn{P(X1,X2) = 0.5 + 0.3 X1 -0.25 X1 X2}
# could thus be written as a vector Pvec such as:
Pvec = c(0.5, 0, 0, 0.3, -0.25, 0)
# considering the convention corresponding to
poLabs(2,2)
# Indeed:
poLabs(2, 2, findIt = Pvec!=0)
# An alternative notation can be used with parameter Xnote
poLabs(2, 2, findIt = Pvec!=0, Xnote = 'w')
# or also
poLabs(2, 2, findIt = Pvec!=0, Xnote = c('x','y'))
```

---

predictab *predictab : estimate the models predictability of gPoMo output*

Description

The algorithm aims to estimate automatically the predictability of the models obtained with ‘gPoMo‘.

Usage

```
predictab(ogp, fullt = NULL, fulldata = NULL, hp = NULL, Nech = 50,
  show = 1, selecmod = NULL, id = 1)
```

Arguments

ogp	The output list obtained from gPoMo.
fullt	Time vector of the data set for which predictability will be tested
fulldata	Data set for which predictability will be tested
hp	Time vector of the horizon of prediction
Nech	Number of simulations
show	Indicates (2) or not (0-1) the algorithm progress
selecmod	a vector of the model selected.
id	the type of model to identify. id = 1 correspond to the unidentified models, that is, potentialy chaotic models).

Value

ErrmodAll A list of matrix $Errmod1, $Errmod2, etc. providing the forecasting error of each model 1, 2, etc. Each column corresponds to one simulation starting from specific initial condition. Each line corresponds to the the horizon of prediction. Vectors corresponding to the initial condition time tE and the horizon of prediction hpE are also provided in $tE and $hpE, respectively.

Author(s)

Sylvain Mangiarotti

Examples

```
# load data
data("Ross76")
# time vector
tin <- Ross76[seq(1, 3000, by = 8), 1]
# single time series
data <- Ross76[seq(1, 3000, by = 8), 3]
# dev.new()
# plot(tin, data, xlab = 'time', ylab = 'y(t)')

# global modelling
# results are put in list outputGPoM
outputGPoM <- gPoMo(data[1:300], tin = tin[1:300], dMax = 2, nS=c(3),
                    show = 0, method = 'rk4',
                    nPmin = 10, nPmax = 12,
                    IstepMin = 150, IstepMax = 151)
#
visuOutGP(outputGPoM)

############################
# and test predictability #
############################
outpred <- predictab(outputGPoM, hp = 15, Nech = 30)

# manual visualisation of the outputs (e.g. for model 1):
dev.new()
```

```
image(outpred$tE, outpred$hpE, t(outpred$Errmod1),
xlab = 't', ylab = 'hp', main = 'Errmod1')
```

`regOrd` *regOrd : Defines the conventional order for regressors in the polynomial formulation*

Description

Defines the order convention for regressors in polynomial. It is formulated as a matrix of explonents. Each column of the matrix (a,b,c, ...) corresponds to one regressor defined as the product of X1^a * X2^b * X3^c, etc.

Usage

```
regOrd(nVar, dMax)
```

Arguments

`nVar`	The number of variables
`dMax`	The maximum degree allowed for the polynomial

Value

A matrix of exponent. Each column correspond to one polynomial term. Each line correspond to one variable. For example, a column of three exponents (0,2,1) correspond to regressor X1^0 * X2^2 * X3^1, that is X2^2 * X3.

Author(s)

Sylvain Mangiarotti

See Also

`poLabs`

regSeries *regSeries : Estimate regressors time series*

Description

Creates series by multiplying given time series among them.

Usage

```
regSeries(nVar, dMax, series, pReg = NULL)
```

Arguments

nVar	The model dimension expected. This parameter will be deduced from the input data (series) if series is a matrix. If series is a vector, the expected dimension nVar should be provided.
dMax	Maximum degree of the polynomial functions allowed in the model (see poLabs).
series	A matrix containing the original time series from which the regressors ones are built. One time series by column
pReg	A matrix filled, for each column, with powers of time series used to create

Value

rpFull A matrix of time series. Each column correspond to one regressor such as $X_1^2 X_3 X_4$

Author(s)

Sylvain Mangiarotti

Examples

```
data(sprottK)
sprottK <- as.matrix(sprottK)
dMax <- 2
nVar <- dim(sprottK)[2]

#Example 1
polySeries2 <- regSeries(nVar, dMax, sprottK)

#Example 2
p <- c(1,3,1)
polySeries2 <- regSeries(nVar, dMax, sprottK, pReg=p)
```

`Rossler-1976 data set` *Data included in package GPoM*

Description

The Rossler system is the 3-dimensional chaotic system
$dx/dt = -y - z$
$dy/dt = x + ay$
$dz/dt = b + z(x - c)$,
discovered by Otto Rossler in 1976 [1]. The following parameters and initial conditions were used in the present data set:
a = 0.520, b = 2, c = 4
and (x0, y0, z0) = (-0.04298734, 1.025536, 0.09057987).
The following four columns are provided:
(1) time t, (2) x(t), (3) y(t) and (4) z(t).
For this parameterization, the Rossler system produces a chaotic behavior non-coherent in phase.

The Rossler system is the 3-dimensional chaotic system
$dx/dt = -y - z$
$dy/dt = x + ay$
$dz/dt = b + z(x - c)$,
discovered by Otto Rossler in 1976 [1]. The following parameters an initial conditions were used in the present data set:
a = 0.520, b = 2, c = 4
and (x0, y0, z0) = (-0.04298734, 1.025536, 0.09057987).
The following four columns are provided:
(1) time t, (2) x(t), (3) y(t) and (4) z(t).
For this parameterization, the Rossler system produces a chaotic behavior non-coherent in phase.

Usage

```
Ross76

rossler
```

Format

An object of class `deSolve` (inherits from `matrix`) with 4000 rows and 4 columns.

Author(s)

Sylvain Mangiarotti, Flavie Le Jean, Malika Chassan, Laurent Drapeau, Mireille Huc.

Sylvain Mangiarotti, Flavie Le Jean, Malika Chassan, Laurent Drapeau, Mireille Huc.

References

[1] P. Rossler, 1976. An Equation for Continuous Chaos, Physics Letters, 71A, 2-3, 155-157.

[1] P. Rossler, 1976. An Equation for Continuous Chaos, Physics Letters, 71A, 2-3, 155-157.

`RosYco` *Twelve Rossler-y time series*

Description

Twelve independant Rossler-1976 time series (variable y). The parameters used to generate the time series correspond to a phase coherent behavior. Details can be found in [1]

Usage

```
RosYco
```

Format

An object of class `matrix` with 3000 rows and 12 columns.

Details

Data of the R-package GPoM

Author(s)

Sylvain Mangiarotti, Flavie Le Jean.

References

[1] Mangiarotti S., Le Jean F., Huc M. & Letellier C., Global Modeling of aggregated and associated chaotic dynamics. Chaos, Solitons and Fractals, 83, 82-96.

sprottK time series *Data included in package GPoM*

Description

The Sprott-K system is the 3-dimensional chaotic system
$dx/dt = xz - ay$
$dy/dt = x + by$
$dz/dt = x - z$,
discovered by Julien C. Sprott in 1994 [1]. The following parameters and initial conditions were used in the present data set:
a = 0.5, b = 0.2
and (x0, y0, z0) = (-0.01170987, 1.010009, -4.522642e-05).
The following four columns are provided:
(1) time t, (2) x(t), (3) y(t) and (4) z(t).
For this parameterization, the Sprott-K system produces a periodic behavior of period 4.

Usage

```
sprottK
```

Format

An object of class `data.frame` with 1000 rows and 3 columns.

Author(s)

Sylvain Mangiarotti, Flavie Le Jean.

References

[1] J.C. Sprott, Some simple flows, 1994. Physical Review E, 50(2), 647-650.

visuEq *visuEq : Displays the model Equations*

Description

Displays the model equations for a polynomial model which description is provided as a matrix K, each column corresponding to an equation, each equation being corresponding to a list of terms following a convention given in ‘poLabs‘.

Usage

```
visuEq(nVar, dMax, K, substit = 0, approx = FALSE)
```

Arguments

nVar	The number of variables
dMax	The maximum degree allowed for the polynomial
K	is the model: each column corresponds to one equation which organisation is following the convention given by function poLabs which requires the definition of the model dimension nVar (i.e. the number of variables) and the maximum polynomial degree dMax allowed.
substit	applies subtitutions: for substit = 0 (default value), variables are chosen as X1, X2, ... for substit = 1, variables X1, X2, ... are replaces by x,y,z, ... for substit = 2, the codes provides a LaTex-like formulation of the model
approx	number of digits to be used: for approx = FALSE (default value) digits are edited with double precision for approx = TRUE, only the minimum number of digits is edited (in order to have all the terms different from 0) for approx = 1, 2, etc. then respectively 1, 2, etc. digits are added to the minimum number of digits corresponding to approx = TRUE.

Author(s)

Sylvain Mangiarotti

Examples

```
#EQUATIONS VISUALISATION
# number of variables:
nVar <- 3
# maximum polynomial degree:
dMax <- 2
# polynomial organization:
poLabs(nVar,dMax)
# model construction
KL = matrix(0, ncol = 3, nrow = 10)
KL[1,1] <- KL[2,2] <- 1
KL[4,1] <- -1
KL[5,3] <- -0.123456789
# Equations visualisation:
# (a) by default, variables names X1, X2, X3 are used
visuEq(nVar, dMax, KL)
# (b) for susbstit=1, variables names x, y, y are used instead
visuEq(nVar, dMax, KL, approx = TRUE, substit=1)
# (c) the name of the variables can also be chosen manualy
visuEq(nVar, dMax, KL, approx = 3, substit=c('U', 'V', 'W'))

# A canonical model can be provided as a single vector
polyTerms <- c(0.2,0,-1,0.5,0,0,0,0,0,0)
visuEq(3,2,KL)
```

visuOutGP *visuOutGP : get a quick information of gPoMo output*

Description

The algorithm aims to get a quick information about the outputs obtained with gPoMo.

Usage

```
visuOutGP(ogp, selecmod = NULL, id = 1, prioMinMax = "data",
  opt3D = "TRUE")
```

Arguments

ogp	The output list obtained from gPoMo.
selecmod	a vector of the model selected.
id	the type of model to identify. id = 1 correspond to the unidentified models, that is, potentialy chaotic models).
prioMinMax	gives the priority for the plots among: "data", "model", "dataonly" and "modelonly".
opt3D	provides a 3D plot (x,y,z) when 'TRUE' (rgl library required).

Value

A Matrix describing the terms composing each model by row. The first row corresponds to the model detection (1 unclarified, 2 diverging, 0 is fixed point, -n with n an integer, is period-n cycle')

Author(s)

Sylvain Mangiarotti

Examples

```
# load data
data("Ross76")
# # time vector
tin <- Ross76[seq(1, 3000, by = 8), 1]
# single time series
data <- Ross76[seq(1, 3000, by = 8), 3]
dev.new()
plot(tin, data, type = 'l')
# global modelling
# results are put in list outputGPoM
outputGPoM <- gPoMo(data, tin=tin, dMax = 2, nS=c(3), show = 0,
                    nPmin = 9, nPmax = 12, method = 'rk4',
                    IstepMin = 200, IstepMax = 201)
visuOutGP(outputGPoM)
```

wInProd *wInProd : weighted inner product*

Description

Computes weighted inner product.

Usage

```
wInProd(A1, A2, weight = NULL)
```

Arguments

A1	The input matrix 1.
A2	The input matrix 2.
weight	: The weighting vector.

Value

inP The weighted inner product.

Examples

```
############
#Example 1 #
############
A1 = c(0,1,2,0,1,3)
A2 = c(1,2,0,0,4,1)
wInProd(A1, A2)

############
#Example 2 #
############
A1 = c(0,1,2,0,1,3)
A2 = c(1,2,0,0,4,1)
w = c(0,0,0,1,1,1)
wInProd(A1, A2, weight = w)
```

Index