Package ‘sysid’

October 14, 2022

Type Package

Title System Identification in R

Version 1.0.4

Date 2017-01-06

Author Suraj Yerramilli, Arun Tangirala

Maintainer Suraj Yerramilli <surajyerramilli@gmail.com>

Description Provides functions for constructing mathematical models of dynamical systems from measured input-output data.

Depends R (>= 3.1)

Imports signal,tframe, ggplot2 (>= 2.1.0), reshape2, polynom, bitops, zoo

RoxygenNote 5.0.1

NeedsCompilation no

Repository CRAN

Date/Publication 2017-01-07 20:00:56

R topics documented:

`armax` *Estimate ARMAX Models*

Description

Fit an ARMAX model of the specified order given the input-output data

Usage

```
armax(x, order = c(0, 1, 1, 0), init_sys = NULL, intNoise = FALSE,
  options = optimOptions())
```

Arguments

`x`	an object of class `idframe`
`order`	Specification of the orders: the four integer components (na,nb,nc,nk) are the order of polynolnomial A, order of polynomial B + 1, order of the polynomial C,and the input-output delay respectively
`init_sys`	Linear polynomial model that configures the initial parameterization. Must be an ARMAX model. Overrules the `order` argument
`intNoise`	Logical variable indicating whether to add integrators in the noise channel (Default=`FALSE`)
`options`	Estimation Options, setup using `optimOptions`

Details

SISO ARMAX models are of the form

$$y[k]+a_1y[k-1]+\ldots+a_{na}y[k-na] = b_{nk}u[k-nk]+\ldots+b_{nk+nb}u[k-nk-nb]+c_1e[k-1]+\ldots c_{nc}e[k-nc]+e[k]$$

The function estimates the coefficients using non-linear least squares (Levenberg-Marquardt Algorithm)
The data is expected to have no offsets or trends. They can be removed using the `detrend` function.

Value

An object of class `estpoly` containing the following elements:

`sys`	an `idpoly` object containing the fitted ARMAX coefficients
`fitted.values`	the predicted response
`residuals`	the residuals
`input`	the input data used
`call`	the matched call
`stats`	A list containing the following fields: `vcov` - the covariance matrix of the fitted coefficients `sigma` - the standard deviation of the innovations
`options`	Option set used for estimation. If no custom options were configured, this is a set of default options
`termination`	Termination conditions for the iterative search used for prediction error minimization: `WhyStop` - Reason for termination `iter` - Number of Iterations `iter` - Number of Function Evaluations

References

Arun K. Tangirala (2015), *Principles of System Identification: Theory and Practice*, CRC Press, Boca Raton. Sections 14.4.1, 21.6.2

Examples

```
data(armaxsim)
z <- dataSlice(armaxsim,end=1533) # training set
mod_armax <- armax(z,c(1,2,1,2))
mod_armax
```

`armaxsim` *Data simulated from an ARMAX model*

Description

This dataset contains 2555 samples simulated from the following ARMAX model:

$$y[k] = \frac{0.6q^{-2} - 0.2q^{-3}}{1 - 0.5q^{-1}} u[k] + \frac{1 - 0.3q^{-1}}{1 - 0.5q^{-1}} e[k]$$

Usage

```
armaxsim
```

Format

an `idframe` object with 2555 samples, one input and one output

Details

The model is simulated with a 2555 samples long full-band PRBS input. The noise variance is set to 0.1

`arx` *Estimate ARX Models*

Description

Fit an ARX model of the specified order given the input-output data

Usage

```
arx(x, order = c(1, 1, 1), lambda = 0.1, intNoise = FALSE, fixed = NULL)
```

Arguments

`x`	an object of class `idframe`
`order`	Specification of the orders: the three integer components (na,nb,nk) are the order of polynolnomial A, (order of polynomial B + 1) and the input-output delay
`lambda`	Regularization parameter(Default=`0.1`)
`intNoise`	Logical variable indicating whether to add integrators in the noise channel (Default=`FALSE`)
`fixed`	list containing fixed parameters. If supplied, only `NA` entries will be varied. Specified as a list of two vectors, each containing the parameters of polynomials A and B respectively.

Details

SISO ARX models are of the form

$$y[k] + a_1 y[k-1] + \ldots + a_{na} y[k-na] = b_{nk} u[k-nk] + \ldots + b_{nk+nb} u[k-nk-nb] + e[k]$$

The function estimates the coefficients using linear least squares (with regularization).
The data is expected to have no offsets or trends. They can be removed using the `detrend` function.
To estimate finite impulse response(`FIR`) models, specify the first order to be zero.

Value

An object of class `estpoly` containing the following elements:

`sys`	an `idpoly` object containing the fitted ARX coefficients
`fitted.values`	the predicted response
`residuals`	the residuals
`input`	the input data used
`call`	the matched call
`stats`	A list containing the following fields: `vcov` - the covariance matrix of the fitted coefficients `sigma` - the standard deviation of the innovations `df` - the residual degrees of freedom

References

Arun K. Tangirala (2015), *Principles of System Identification: Theory and Practice*, CRC Press, Boca Raton. Section 21.6.1

Lennart Ljung (1999), *System Identification: Theory for the User*, 2nd Edition, Prentice Hall, New York. Section 10.1

Examples

```
data(arxsim)
mod_arx <- arx(arxsim,c(1,2,2))
mod_arx
plot(mod_arx) # plot the predicted and actual responses
```

`arxsim` *Data simulated from an ARX model*

Description

This dataset contains 2555 samples simulated from the following ARX model:

$$y[k] = \frac{0.6q^{-2} - 0.2q^{-3}}{1 - 0.5q^{-1}} u[k] + \frac{1}{1 - 0.5q^{-1}} e[k]$$

Usage

```
arxsim
```

Format

an `idframe` object with 2555 samples, one input and one output

Details

The model is simulated with a 2555 samples long full-band PRBS input. The noise variance is set to 0.1

`bj` *Estimate Box-Jenkins Models*

Description

Fit a box-jenkins model of the specified order from input-output data

Usage

```
bj(z, order = c(1, 1, 1, 1, 0), init_sys = NULL, options = optimOptions())
```

Arguments

`z`	an `idframe` object containing the data
`order`	Specification of the orders: the five integer components (nb,nc,nd,nf,nk) are order of polynomial B + 1, order of the polynomial C, order of the polynomial D, order of the polynomial F, and the input-output delay respectively
`init_sys`	Linear polynomial model that configures the initial parameterization. Must be a BJ model. Overrules the `order` argument
`options`	Estimation Options, setup using `optimOptions`

Details

SISO BJ models are of the form

$$y[k] = \frac{B(q^{-1})}{F(q^{-1})} u[k-nk] + \frac{C(q^{-1})}{D(q^{-1})} e[k]$$

The orders of Box-Jenkins model are defined as follows:

$$B(q^{-1}) = b_1 + b_2 q^{-1} + \ldots + b_{nb} q^{-nb+1}$$

$$C(q^{-1}) = 1 + c_1 q^{-1} + \ldots + c_{nc} q^{-nc}$$

$$D(q^{-1}) = 1 + d_1 q^{-1} + \ldots + d_{nd} q^{-nd}$$

$$F(q^{-1}) = 1 + f_1 q^{-1} + \ldots + f_{nf} q^{-nf}$$

The function estimates the coefficients using non-linear least squares (Levenberg-Marquardt Algorithm)
The data is expected to have no offsets or trends. They can be removed using the `detrend` function.

Value

An object of class `estpoly` containing the following elements:

`sys`	an `idpoly` object containing the fitted BJ coefficients
`fitted.values`	the predicted response
`residuals`	the residuals
`input`	the input data used
`call`	the matched call
`stats`	A list containing the following fields: `vcov` - the covariance matrix of the fitted coefficients `sigma` - the standard deviation of the innovations
`options`	Option set used for estimation. If no custom options were configured, this is a set of default options
`termination`	Termination conditions for the iterative search used for prediction error minimization: `WhyStop` - Reason for termination `iter` - Number of Iterations `iter` - Number of Function Evaluations

References

Arun K. Tangirala (2015), *Principles of System Identification: Theory and Practice*, CRC Press, Boca Raton. Sections 14.4.1, 17.5.2, 21.6.3

Examples

```
data(bjsim)
z <- dataSlice(bjsim,end=1500) # training set
mod_bj <- bj(z,c(2,1,1,1,2))
mod_bj
residplot(mod_bj) # residual plots
```

`bjsim` *Data simulated from an BJ model*

Description

This dataset contains 2046 samples simulated from the following BJ model:

$$y[k] = \frac{0.6q^{-2} - 0.2q^{-3}}{1 - 0.5q^{-1}}u[k] + \frac{1 + 0.2q^{-1}}{1 - 0.3q^{-1}}e[k]$$

Usage

```
bjsim
```

Format

an `idframe` object with 2046 samples, one input and one output

Details

The model is simulated with a 2046 samples long full-band PRBS input. The noise variance is set to 0.1

`compare` *Compare the measured output and the predicted output(s)*

Description

Plots the output predictions of model(s) superimposed over validation data, data, for comparison.

Usage

```
compare(data, nahead = 1, ...)
```

Arguments

`data`	validation data in the form of an `idframe` object
`nahead`	number of steps ahead at which to predict (Default:1). For infinite- step ahead predictions, supply `Inf`.
`...`	models whose predictions are to be compared

See Also

predict.estpoly for obtaining model predictions

Examples

```
data(arxsim)
mod1 <- arx(arxsim,c(1,2,2))
mod2 <- oe(arxsim,c(2,1,1))
compare(arxsim,nahead=Inf,mod1,mod2)
```

---

cstr *Continuous stirred tank reactor data (idframe)*

Description

The Process is a model of a Continuous Stirring Tank Reactor, where the reaction is exothermic and the concentration is controlled by regulating the coolant flow.

Usage

```
cstr
```

Format

an `idframe` object with 7500 samples, one input and two outputs

Details

Inputs: q, Coolant Flow l/min Outputs:

Ca Concentration mol/l

T Temperature Kelvin

cstrData *Continuous stirred tank reactor data (data.frame)*

Description

The Process is a model of a Continuous Stirring Tank Reactor, where the reaction is exothermic and the concentration is controlled by regulating the coolant flow.

Usage

```
cstrData
```

Format

an `data.frame` object with 7500 rows and three columns: q, Ca and T

Details

Inputs: q, Coolant Flow l/min Outputs:

Ca Concentration mol/l

T Temperature Kelvin

Source

ftp://ftp.esat.kuleuven.be/pub/SISTA/data/process_industry/cstr.dat.gz

cstr_mis *Continuous stirred tank reactor data with missing values*

Description

This dataset is derived from the `cstr` dataset with few samples containing missing values, in one or all variables. It is used to demonstrate the capabilities of the `misdata` routine.

Usage

```
cstr_mis
```

Format

an `idframe` object with 7500 samples, one input and two outputs

See Also

`cstr`, `misdata`

dataSlice *Subset or Resample idframe data*

Description

dataSlice is a subsetting method for objects of class idframe. It extracts the subset of the object data observed between indices start and end. If a frequency is specified, the series is then resampled at the new frequency.

Usage

```
dataSlice(data, start = NULL, end = NULL, freq = NULL)
```

Arguments

data	an object of class idframe
start	the start index
end	the end index
freq	fraction of the original frequency at which the series to be sampled.

Details

The dataSlice function extends the window function for idframe objects

Value

an idframe object

See Also

window

Examples

```
data(cstr)
cstrsub <- dataSlice(cstr,start=200,end=400) # extract between indices 200 and 400
cstrTrain <- dataSlice(cstr,end=4500) # extract upto index 4500
cstrTest <- dataSlice(cstr,start=6501) # extract from index 6501 till the end
cstr_new <- dataSlice(cstr,freq=0.5) # resample data at half the original frequency
```

detrend — *Remove offsets and linear trends*

Description

Removes offsets or trends from data

Usage

```
detrend(x, type = 0)
```

Arguments

x	an object of class `idframe`
type	argument indicating the type of trend to be removed (Default=`0`) • type=`0`: Subtracts mean value from each signal • type=`1`: Subtracts a linear trend (least-squres fit) • type=`trInfo` object: Subtracts a trend specified by the object

Details

`R` by default doesn't allow return of multiple objects. The `%=%` operator and `g` function in this package facillitate this behaviour. See the examples section for more information.

Value

A list containing two objects: the detrended data and the trend information

See Also

`lm`

Examples

```
data(cstr)
datatrain <- dataSlice(cstr,end=4500)
datatest <- dataSlice(cstr,4501)
g(Ztrain,tr) %=% detrend(datatrain) # Remove means
g(Ztest) %=% detrend(datatest,tr)
```

`estpoly` *Estimated polynomial object*

Description

Estimated discrete-time polynomial model returned from an estimation routine.

Usage

```
estpoly(sys, fitted.values, residuals, options = NULL, call, stats,
  termination = NULL, input)
```

Arguments

`sys`	an `idpoly` object containing the estimated polynomial coefficients
`fitted.values`	1-step ahead predictions on the training dataset
`residuals`	1-step ahead prediction errors
`options`	optimization specification ser used (applicable for non-linear least squares)
`call`	the matched call
`stats`	a list containing estimation statistics
`termination`	termination criteria for optimization
`input`	input signal of the training data-set

Details

Do not use `estpoly` for directly specifing an input-output polynomial model. `idpoly` is to be used instead

`etfe` *Estimate empirical transfer function*

Description

Estimates the emperical transfer function from the data by taking the ratio of the fourier transforms of the output and the input variables

Usage

```
etfe(data, n = 128)
```

Arguments

`data`	an object of class `idframe`
`n`	frequency spacing (Default: `128`)

Value

an `idfrd` object containing the estimated frequency response

References

Arun K. Tangirala (2015), *Principles of System Identification: Theory and Practice*, CRC Press, Boca Raton. Sections 5.3 and 20.4.2

See Also

`fft`

Examples

```
data(arxsim)
frf <- etfe(arxsim)
```

`fitch` *Fit Characteristics*

Description

Returns quantitative assessment of the estimated model as a list

Usage

```
fitch(x)
```

Arguments

`x`	the estimated model

Value

A list containing the following elements

`MSE`	Mean Square Error measure of how well the response of the model fits the estimation data
`FPE`	Final Prediction Error
`FitPer`	Normalized root mean squared error (NRMSE) measure of how well the response of the model fits the estimation data, expressed as a percentage.
`AIC`	Raw Akaike Information Citeria (AIC) measure of model quality
`AICc`	Small sample-size corrected AIC
`nAIC`	Normalized AIC
`BIC`	Bayesian Information Criteria (BIC)

`frd` *Frequency response data*

Description

This dataset contains frequency response data of an unknown SISO system.

Usage

```
frd
```

Format

an `idfrd` object with response at 128 frequency points

`getcov` *Parameter covariance of the identified model*

Description

Obtain the parameter covariance matrix of the linear, identified parametric model

Usage

```
getcov(sys)
```

Arguments

`sys`	a linear, identified parametric model

`idframe` *S3 class for storing input-output data.*

Description

`idframe` is an S3 class for storing and manipulating input-ouput data. It supports discrete time and frequency domain data.

Usage

```
idframe(output, input = NULL, Ts = 1, start = 0, end = NULL,
  unit = c("seconds", "minutes", "hours", "days")[1])
```

Arguments

`output`	dataframe/matrix/vector containing the outputs
`input`	dataframe/matrix/vector containing the inputs
`Ts`	sampling interval (Default: 1)
`start`	Time of the first observation
`end`	Time of the last observation Optional Argument
`unit`	Time unit (Default: "seconds")

Value

an idframe object

See Also

`plot.idframe`, the plot method for idframe objects

Examples

```
dataMatrix <- matrix(rnorm(1000),ncol=5)
data <- idframe(output=dataMatrix[,3:5],input=dataMatrix[,1:2],Ts=1)
```

---

idfrd — *S3 class constructor for storing frequency response data*

Description

S3 class constructor for storing frequency response data

Usage

```
idfrd(respData, freq, Ts, spec = NULL, covData = NULL, noiseCov = NULL)
```

Arguments

`respData`	frequency response data. For SISO systems, supply a vector of frequency response values. For MIMO systems with Ny outputs and Nu inputs, supply an array of size c(Ny,Nu,Nw).
`freq`	frequency points of the response
`Ts`	sampling time of data
`spec`	power spectra and cross spectra of the system output disturbances (noise). Supply an array of size (Ny,Ny,Nw)
`covData`	response data covariance matrices. Supply an array of size (Ny,Nu,Nw,2,2). covData[ky,ku,kw,,] is the covariance matrix of respData[ky,ku,kw]
`noiseCov`	power spectra variance. Supply an array of size (Ny,Ny,Nw)

Value

an idfrd object

See Also

plot.idfrd for generating bode plots, spa and etfe for estimating the frequency response given input/output data

---

idinput *function to generate input singals (rgs/rbs/prbs/sine)*

Description

idinput is a function for generating input signals (rgs/rbs/prbs/sine) for identification purposes

Usage

```
idinput(n, type = "rgs", band = c(0, 1), levels = c(-1, 1))
```

Arguments

n	integer length of the input singal to be generated
type	the type of input signal to be generated. 'rgs' - generates random gaussian signal 'rbs' - generates random binary signal 'prbs' - generates pseudorandom binary signal 'sine' - generates a signal that is a sum of sinusoids Default value is type='rgs'
band	determines the frequency content of the signal. For type='rbs'/'sine'/, band = [wlow,whigh] which specifies the lower and the upper bound of the passband frequencies(expressed as fractions of Nyquist frequency). Default is c(0,1) For type='prbs', band=[0,B] where B is such that the singal is constant over 1/B (clock period). Default is c(0,1)
levels	row vector defining the input level. It is of the form levels=c(minu, maxu) For 'rbs','prbs', 'sine', the generated signal always between minu and maxu. For 'rgs', minu=mean value of signal minus one standard deviation and maxu=mean value of signal plus one standard deviation Default value is levels=c(-1,1)

`idpoly` *Polynomial model with identifiable parameters*

Description

Creates a polynomial model with identifiable coefficients

Usage

```
idpoly(A = 1, B = 1, C = 1, D = 1, F1 = 1, ioDelay = 0, Ts = 1,
  noiseVar = 1, intNoise = F, unit = c("seconds", "minutes", "hours",
  "days")[1])
```

Arguments

`A`	autoregressive coefficients
`B, F1`	coefficients of the numerator and denominator respectively of the deterministic model between the input and output
`C, D`	coefficients of the numerator and denominator respectively of the stochastic model
`ioDelay`	the delay in the input-output channel
`Ts`	sampling interval
`noiseVar`	variance of the white noise source (Default=1)
`intNoise`	Logical variable indicating presence or absence of integrator in the noise channel (Default=`FALSE`)
`unit`	time unit (Default=`"seconds"`)

Details

Discrete-time polynomials are of the form

$$A(q^{-1})y[k] = \frac{B(q^{-1})}{F1(q^{-1})}u[k] + \frac{C(q^{-1})}{D(q^{-1})}e[k]$$

Examples

```
# define output-error model
mod_oe <- idpoly(B=c(0.6,-0.2),F1=c(1,-0.5),ioDelay = 2,Ts=0.1,
noiseVar = 0.1)

# define box-jenkins model with unit variance
B <- c(0.6,-0.2)
C <- c(1,-0.3)
D <- c(1,1.5,0.7)
F1 <- c(1,-0.5)
mod_bj <- idpoly(1,B,C,D,F1,ioDelay=1)
```

`impulseest` *Estimate Impulse Response Coefficients*

Description

`impulseest` is used to estimate impulse response coefficients from the data

Usage

```
impulseest(x, M = 30, K = NULL, regul = F, lambda = 1)
```

Arguments

`x`	an object of class `idframe`
`M`	Order of the FIR Model (Default:`30`)
`K`	Transport delay in the estimated impulse response (Default:NULL)
`regul`	Parameter indicating whether regularization should be used. (Default:`FALSE`)
`lambda`	The value of the regularization parameter. Valid only if `regul=TRUE`. (Default:`1`)

Details

The IR Coefficients are estimated using linear least squares. Future Versions will provide support for multivariate data.

References

Arun K. Tangirala (2015), *Principles of System Identification: Theory and Practice*, CRC Press, Boca Raton. Sections 17.4.11 and 20.2

See Also

`step`

Examples

```
uk <- rnorm(1000,1)
yk <- filter (uk,c(0.9,-0.4),method="recursive") + rnorm(1000,1)
data <- idframe(output=data.frame(yk),input=data.frame(uk))
fit <- impulseest(data)
impulseplot(fit)
```

impulseplot *Impulse Response Plots*

Description

Plots the estimated IR coefficients along with the significance limits at each lag.

Usage

```
impulseplot(model, sd = 2)
```

Arguments

`model`	an object of class `impulseest`
`sd`	standard deviation of the confidence region (Default: `2`)

See Also

`impulseest`,`step`

inputData *Output or Input-data*

Description

Extract output-data or input-data in idframe objects

Usage

```
inputData(x, series)
```

Arguments

`x`	`idframe` object
`series`	the indices to extract

`inputNames<-` *Extract or set series' names*

Description

Extract or set names of series in input or output

Usage

```
inputNames(x) <- value
```

Arguments

`x`	`idframe` object
`value`	vector of strings

`iv` *ARX model estimation using instrumental variable method*

Description

Estimates an ARX model of the specified order from input-output data using the instrument variable method. If arbitrary instruments are not supplied by the user, the instruments are generated using the arx routine

Usage

```
iv(z, order = c(0, 1, 0), x = NULL)
```

Arguments

`z`	an idframe object containing the data
`order`	Specification of the orders: the three integer components (na,nb,nk) are the order of polynolnomial A, (order of polynomial B + 1) and the input-output delay
`x`	instrument variable matrix. x must be of the same size as the output data. (Default: `NULL`)

Value

An object of class `estpoly` containing the following elements:

`sys`	an `idpoly` object containing the fitted ARX coefficients
`fitted.values`	the predicted response
`residuals`	the residuals
`input`	the input data used
`call`	the matched call
`stats`	A list containing the following fields: `vcov` - the covariance matrix of the fitted coefficients `sigma` - the standard deviation of the innovations `df` - the residual degrees of freedom

References

Arun K. Tangirala (2015), *Principles of System Identification: Theory and Practice*, CRC Press, Boca Raton. Sections 21.7.1, 21.7.2

Lennart Ljung (1999), *System Identification: Theory for the User*, 2nd Edition, Prentice Hall, New York. Section 7.6

See Also

`arx`, `iv4`

Examples

```
data(arxsim)
mod_iv <- iv(arxsim,c(2,1,1))
```

`iv4` *ARX model estimation using four-stage instrumental variable method*

Description

Estimates an ARX model of the specified order from input-output data using the instrument variable method. The estimation algorithm is insensitive to the color of the noise term.

Usage

```
iv4(z, order = c(0, 1, 0))
```

Arguments

`z`	an idframe object containing the data
`order`	Specification of the orders: the three integer components (na,nb,nk) are the order of polynolnomial A, (order of polynomial B + 1) and the input-output delay

Details

Estimation is performed in 4 stages. The first stage uses the arx function. The resulting model generates the instruments for a second-stage IV estimate. The residuals obtained from this model are modeled using a sufficently high-order AR model. At the fourth stage, the input-output data is filtered through this AR model and then subjected to the IV function with the same instrument filters as in the second stage.

Value

An object of class `estpoly` containing the following elements:

`sys`	an `idpoly` object containing the fitted ARX coefficients
`fitted.values`	the predicted response
`residuals`	the residuals
`input`	the input data used
`call`	the matched call
`stats`	A list containing the following fields: `vcov` - the covariance matrix of the fitted coefficients `sigma` - the standard deviation of the innovations `df` - the residual degrees of freedom

References

Lennart Ljung (1999), *System Identification: Theory for the User*, 2nd Edition, Prentice Hall, New York. Section 15.3

See Also

`arx`, `iv4`

Examples

```
mod_dgp <- idpoly(A=c(1,-0.5),B=c(0.6,-.2),C=c(1,0.6),ioDelay = 2,noiseVar = 0.1)
u <- idinput(400,"prbs")
y <- sim(mod_dgp,u,addNoise=TRUE)
z <- idframe(y,u)
mod_iv4 <- iv4(z,c(1,2,2))
```

misdata — *Replace Missing Data by Interpolation*

Description

Function for replacing missing values with interpolated ones. This is an extension of the `na.approx` function from the `zoo` package. The missing data is indicated using the value *NA*.

Usage

```
misdata(data)
```

Arguments

`data`	an object of class `idframe`

Value

data (an idframe object) with missing data replaced.

See Also

`na.approx`

Examples

```
data(cstr_mis)
summary(cstr_mis) # finding out the number of NAs
cstr <- misdata(cstr_mis)
```

nInputSeries — *Number of series in input or output*

Description

Number of series in input or output in a idframe object

Usage

```
nInputSeries(data)
```

Arguments

`data`	`idframe` object

oe *Estimate Output-Error Models*

Description

Fit an output-error model of the specified order given the input-output data

Usage

```
oe(x, order = c(1, 1, 0), init_sys = NULL, options = optimOptions())
```

Arguments

x	an object of class `idframe`
order	Specification of the orders: the four integer components (nb,nf,nk) are order of polynomial B + 1, order of the polynomial F, and the input-output delay respectively
init_sys	Linear polynomial model that configures the initial parameterization. Must be an OE model. Overrules the `order` argument
options	Estimation Options, setup using `optimOptions`

Details

SISO OE models are of the form

$$y[k]+f_1y[k-1]+\ldots+f_{nf}y[k-nf] = b_{nk}u[k-nk]+\ldots+b_{nk+nb}u[k-nk-nb]+f_1e[k-1]+\ldots f_{nf}e[k-nf]+e[k]$$

The function estimates the coefficients using non-linear least squares (Levenberg-Marquardt Algorithm)
The data is expected to have no offsets or trends. They can be removed using the `detrend` function.

Value

An object of class `estpoly` containing the following elements:

sys	an `idpoly` object containing the fitted OE coefficients
fitted.values	the predicted response
residuals	the residuals
input	the input data used
call	the matched call
stats	A list containing the following fields: `vcov` - the covariance matrix of the fitted coefficients `sigma` - the standard deviation of the innovations
options	Option set used for estimation. If no custom options were configured, this is a set of default options

termination — Termination conditions for the iterative search used for prediction error minimization: `WhyStop` - Reason for termination
`iter` - Number of Iterations
`iter` - Number of Function Evaluations

References

Arun K. Tangirala (2015), *Principles of System Identification: Theory and Practice*, CRC Press, Boca Raton. Sections 14.4.1, 17.5.2, 21.6.3

Examples

```
data(oesim)
z <- dataSlice(oesim,end=1533) # training set
mod_oe <- oe(z,c(2,1,2))
mod_oe
plot(mod_oe) # plot the predicted and actual responses
```

`oesim` *Data simulated from an OE model*

Description

This dataset contains 2555 samples simulated from the following OE model:

$$y[k] = \frac{0.6q^{-2} - 0.2q^{-3}}{1 - 0.5q^{-1}} u[k] + e[k]$$

Usage

```
oesim
```

Format

an `idframe` object with 2555 samples, one input and one output

Details

The model is simulated with a 2555 samples long full-band PRBS input. The noise variance is set to 0.1

optimOptions — *Create optimization options*

Description

Specify optimization options that are to be passed to the numerical estimation routines

Usage

```
optimOptions(tol = 0.01, maxIter = 20, LMinit = 0.01, LMstep = 2,
  display = c("off", "on")[1])
```

Arguments

`tol`	Minimum 2-norm of the gradient (Default: `1e-2`)
`maxIter`	Maximum number of iterations to be performed
`LMinit`	Starting value of search-direction length in the Levenberg-Marquardt method (Default: `0.01`)
`LMstep`	Size of the Levenberg-Marquardt step (Default: `2`)
`display`	Argument whether to display iteration details or not (Default: `"off"`)

plot.idframe — *Plotting idframe objects*

Description

Plotting method for objects inherting from class `idframe`

Usage

```
## S3 method for class 'idframe'
plot(x, col = "steelblue", lwd = 1, main = NULL,
  size = 12, ...)
```

Arguments

`x`	an `idframe` object
`col`	line color, to be passed to plot.(Default=`"steelblue"`)
`lwd`	line width, in millimeters(Default=1)
`main`	the plot title. (Default = `NULL`)
`size`	text size (Default = `12`)
`...`	additional arguments

Examples

```
data(cstr)
plot(cstr,col="blue")
```

---

plot.idfrd *Plotting idfrd objects*

Description

Generates the bode plot of the given frequency response data. It uses the ggplot2 plotting engine

Usage

```
## S3 method for class 'idfrd'
plot(x, col = "steelblue", lwd = 1, ...)
```

Arguments

`x`	An object of class `idframe`
`col`	a specification for the line colour (Default : `" steelblue"`)
`lwd`	the line width, a positive number, defaulting to 1
`...`	additional arguments

See Also

`ggplot`

Examples

```
data(frd)
plot(frd)
```

---

predict.estpoly *Predictions of identified model*

Description

Predicts the output of an identified model (`estpoly`) object K steps ahead.

Usage

```
## S3 method for class 'estpoly'
predict(object, newdata = NULL, nahead = 1, ...)
```

Arguments

`object`	`estpoly` object containing the identified model
`newdata`	optional dataset to be used for predictions. If not supplied, predictions are made on the training set.
`nahead`	number of steps ahead at which to predict (Default:1). For infinite- step ahead predictions or pure simulation, supply `Inf`.
`...`	other arguments

Value

Time-series containing the predictions

References

Arun K. Tangirala (2015), *Principles of System Identification: Theory and Practice*, CRC Press, Boca Raton. Chapter 18

Examples

```
data(arxsim)
mod1 <- oe(arxsim,c(2,1,1))
Yhat <- predict(mod1,arxsim) #  1-step ahead predictions
Yhat_2 <- predict(mod1,arxsim,nahead=2) # 2-step ahead predictions
Yhat_inf <- predict(mod1,arxsim,nahead=Inf) # Infinite-step ahead predictions
```

`rarx` *Estimate parameters of ARX recursively*

Description

Estimates the parameters of a single-output ARX model of the specified order from data using the recursive weighted least-squares algorithm.

Usage

```
rarx(x, order = c(1, 1, 1), lambda = 0.95)
```

Arguments

`x`	an object of class `idframe`
`order`	Specification of the orders: the three integer components (na,nb,nk) are the order of polynolnomial A, (order of polynomial B + 1) and the input-output delay
`lambda`	Forgetting factor(Default=`0.95`)

Value

A list containing the following objects

theta Estimated parameters of the model. The k^{th} row contains the parameters associated with the k^{th} sample. Each row in `theta` has the following format:
theta[i,:]=[a1,a2,...,ana,b1,...bnb]

yhat Predicted value of the output, according to the current model - parameters based on all past data

References

Arun K. Tangirala (2015), *Principles of System Identification: Theory and Practice*, CRC Press, Boca Raton. Section 25.1.3

Lennart Ljung (1999), *System Identification: Theory for the User*, 2nd Edition, Prentice Hall, New York. Section 11.2

Examples

```
Gp1 <- idpoly(c(1,-0.9,0.2),2,ioDelay=2,noiseVar = 0.1)
Gp2 <- idpoly(c(1,-1.2,0.35),2.5,ioDelay=2,noiseVar = 0.1)
uk = idinput(2044,'prbs',c(0,1/4)); N = length(uk);
N1 = round(0.35*N); N2 = round(0.4*N); N3 = N-N1-N2;
yk1 <- sim(Gp1,uk[1:N1],addNoise = TRUE)
yk2 <- sim(Gp2,uk[N1+1:N2],addNoise = TRUE)
yk3 <- sim(Gp1,uk[N1+N2+1:N3],addNoise = TRUE)
yk <- c(yk1,yk2,yk3)
z <- idframe(yk,uk,1)
g(theta,yhat) %=% rarx(z,c(2,1,2))
```

`read.idframe` *Data input into a idframe object*

Description

Read the contents of a data.frame/matrix into a `idframe` object.

Usage

```
read.idframe(data, ninputs = NULL, Ts = 1, unit = c("seconds", "minutes",
  "hours", "days")[1])
```

Arguments

`data`	a `data.frame` object
`ninputs`	the number of input columns. (Default: 0)
`Ts`	sampling interval (Default: 1)
`unit`	Time Unit (Default: "seconds")

Value

an idframe object

Examples

```
data(cstrData)
data <- read.idframe(cstrData,ninputs=1,Ts= 1,unit="minutes")
```

---

read.table.idframe   *Read the contents of a table-formatted file*

Description

Read the contents of an file in table format into a `idframe` object.

Usage

```
read.table.idframe(file, header = TRUE, sep = ",", ninputs = 0, Ts = 1,
  unit = c("seconds", "minutes", "hours", "days")[1], ...)
```

Arguments

`file`	the path to the file to read
`header`	a logical value indicating whether the first row corresponding to the first element of the rowIndex vector contains the names of the variables. (Default: `TRUE`)
`sep`	the field separator character. Values on each line of the file are separated by this character. (Default: `","`)
`ninputs`	the number of input columns. (Default: 0)
`Ts`	sampling interval (Default: 1)
`unit`	Time Unit (Default: "seconds")
`...`	additional arguments to be passed to the `read.table` function

Details

The `read.table.idframe` function uses the `read.table` function, provided by the **utils** package, to read data from a table-formatted file and then calls the `read.idframe` function to read the data into a idframe object

Value

an idframe object

See Also

`read.table`

Examples

```
dataMatrix <- data.frame(matrix(rnorm(1000),ncol=5))
colnames(dataMatrix) <- c("u1","u2","y1","y2","y3")
write.csv(dataMatrix,file="test.csv",row.names=FALSE)

data <- read.table.idframe("test.csv",ninputs=2,unit="minutes")
```

---

residplot *Plot residual characteristics*

Description

Computes the 1-step ahead prediction errors (residuals) for an estimated polynomial model, and plots auto-correlation of the residuals and the cross-correlation of the residuals with the input signals.

Usage

```
residplot(model, newdata = NULL)
```

Arguments

model	estimated polynomial model
newdata	an optional dataset on which predictions are to be computed. If not supplied, predictions are computed on the training dataset.

---

sim *Simulate response of dynamic system*

Description

Simulate the response of a system to a given input

Usage

```
sim(model, input, addNoise = F, innov = NULL, seed = NULL)
```

Arguments

`model`	the linear system to simulate
`input`	a vector/matrix containing the input
`addNoise`	logical variable indicating whether to add noise to the response model. (Default: `FALSE`)
`innov`	an optional times series of innovations. If not supplied (specified as `NULL`), gaussian white noise is generated, with the variance specified in the model (Property: `noiseVar`)
`seed`	integer indicating the seed value of the random number generator. Useful for reproducibility purposes.

Details

The routine is currently built only for SISO systems. Future versions will include support for MIMO systems.

Value

a vector containing the simulated output

Examples

```
# ARX Model
u <- idinput(300,"rgs")
model <- idpoly(A=c(1,-1.5,0.7),B=c(0.8,-0.25),ioDelay=1,
noiseVar=0.1)
y <- sim(model,u,addNoise=TRUE)
```

`spa` — *Estimate frequency response*

Description

Estimates frequency response and noise spectrum from data with fixed resolution using spectral analysis

Usage

```
spa(x, winsize = NULL, freq = NULL)
```

Arguments

`x`	an `idframe` object
`winsize`	lag size of the Hanning window (Default: `min (length(x)/10,30)`)
`freq`	frequency points at which the response is evaluated (Default: `seq(1,128)/128*pi/Ts`)

Value

an `idfrd` object containing the estimated frequency response and the noise spectrum

References

Arun K. Tangirala (2015), *Principles of System Identification: Theory and Practice*, CRC Press, Boca Raton. Sections 16.5 and 20.4

Examples

```
data(arxsim)
frf <- spa(arxsim)
```

---

step — *Step Response Plots*

Description

Plots the step response of a system, given the IR model

Usage

```
step(model)
```

Arguments

`model`	an object of class `impulseest`

See Also

`impulseest`

Examples

```
uk <- rnorm(1000,1)
yk <- filter (uk,c(0.9,-0.4),method="recursive") + rnorm(1000,1)
data <- idframe(output=data.frame(yk),input=data.frame(uk))
fit <- impulseest(data)
step(fit)
```

`time` *Sampling times of IO data* `time` *creates the vector of times at which data was sampled.* `frequency` *returns the number of damples per unit time and* `deltat` *the time-interval between observations*

Description

Sampling times of IO data

`time` creates the vector of times at which data was sampled. `frequency` returns the number of damples per unit time and `deltat` the time-interval between observations

Usage

```
time(x)
```

Arguments

`x`	a idframe object, or a univariate or multivariate time-series, or a vector or matrix

---

`%=%` *Multiple assignment operator*

Description

Assign multiple variables from a list or function return object

Usage

```
l %=% r
```

Arguments

`l`	the variables to be assigned
`r`	the list or function-return object

Index