

Functions by category

Design base by category

Cryptonyms by category

Functions by name: a b c d e f g i k l m n p r s t u

a

algen

aloptim

arx2arx

arx2ful

arxc2can

arxc2fac

arxc2mix

b

betaln

bmtbinit

c

can2arxc

can2fac

can2marg

canarxls

cellcat

chnconst

chnget

chnset

com2can

com2com

com2pro

comarx

comarxls

comdel

commerge

complot

comunpk

contents

credit

credits

d

datagrid

dataplot

datascan

defaults

dial1

dial2

dial3

dydrs

e

equal
estfrg
f
fac2arxc
fac2ls
fac2marg
facarx
facarxls
facchng
facdpred
facflat
facfrg
facgmean
facmark
facmerge
facpred
facsort
facstr
facupdt
facupdtp
facupred
facvll
filoutm
fixerr
fullscreen

g
gauss1
gaussn
genmixe
genmixel
genmxi
genstr
gentab
getdvect
getflds
getth

i
inisyn
isdimeq
isstatic
iterplot

k
kldcom
kldiscom
kldisdir
kldist
kldistc

l
ld2ld
ld2ls

ld2ud
ld2v
ldchng
ldinv
ldperm
ldupdt
ldupdtp
linplot
loglik
ls2ld
ltdl
m
mix2arxc
mix2mix
mix2mix0
mix2mixm
mix2pro
mixbrow
mixconst
mixcopy
mixcut
mixdfms
mixdump
mixest
mixestbb
mixestbq
mixestem
mixestfe
mixestim
mixestimp
mixestims
mixestmt
mixestpb
mixestqb
mixestqbs
mixflat
mixflatv
mixfrg
mixgmean
mixgrid
mixinit
mixmaxtd
mixmesh
mixplot
mixplotc
mixpro
mixreorg
mixscan
mixshow
mixsimul

mixstats
mixstrid
mixupdt
mixupdtp
mversion
n
noise
p
perm
preaux
preaux1
preinit
preproc
prestep
prior
pristr
pristrd
pro2pre
pro2str
prodini
profix
profixn
profixn1
protest
prt
prtstr
psi
r
randnm
randun
relep
relepn
resizefig
restore
ricexp
ricpen
ricpenu
ricshift
s
scalepri
setaxis
setdbg
setfig
sigscan
sim2pdf
simeval
soptim
statgrid
statmesh
statplot

statsim
stopstac
straux1
streq
strmax
student
synmxi
t
target
tukinit
u
ud2ld
udform
udinv
udupdt
ufcgen
utdu
utinv

User's Functions

Constructors	
<code>Fac = facarx(ychn, str)</code>	build ARX factor
<code>Fac = facarxls(ychn, str)</code>	build ARX LS factor
<code>Com = comarxls(ychns, str)</code>	build matrix ARX LS component
<code>Com = comarx(ychns, str)</code>	build matrix ARX component
<code>Mix = mixconst(Facs, coms, dfcs)</code>	build ARX or ARX LS mixture
<code>Mix = mixconst(Coms, dfcs)</code>	build mixture of any type

Initialization of estimation	
<code>Mix = ...</code>	initialization of mixture estimation ^a
<code>mixinit(Mix0, frg, ndat, niter, opt, belief)</code>	
<code>Mix = comdel(Mix, com)</code>	cancel specified component
<code>Mix = commerge(Mix, Mix0, com)</code>	merge mixture components
<code>Mix = mixcut(Mix)</code>	cancel components that explain low amount of data
<code>Mix0 = genmixe(ncom, ychns, str, ndat)</code>	generate initial mixture
^a estimation options: 'p', 'q', 'b', 'f', 'm' 'n'+ number of iteration steps; belief expresses user's belief into the regressor specified	

Estimation operations	
<code>Mix = ...</code>	
<code>mixest(Mix0, frg, niter, opt)</code>	iterative mixture estimation ^b
<code>Mix = mixestim(Mix0, frg, ndat)</code>	quasi-Bayes mixture estimation
<code>Mix = mixestim(Mix0, frg)</code>	recursive quasi-Bayes mixture estimation
<code>Mix = mixestimp(Mix0, frg, ndat)</code>	projection based quasi-Bayes estimation
<code>Mix = mixestimp(Mix0, frg)</code>	projection based recursive quasi-Bayes estimation
<code>Mix0 = mixflat(Mix)</code>	mixture flattening
<code>Mix = mixstats(Mix, ndat)</code>	compute estimation statistics
<code>Mix = mixstats(Mix)</code>	compute statistics recursively
<code>Mix0 = genmixe(ncom, ychns, str)</code>	generate initial mixture for estimation
^b opt - options: 'q', 'b', 'f', 'm' for quasi-Bayes, batch Bayes, forgetting branching and estimation with fixed covariances	

Prediction operations

<code>pMix = mix2mixm(Mix, pchns)</code>	build marginal predictor
<code>pMix = mix2pro(Mix, pchns, cchns)</code>	build/re-build mixture projector
<code>pMix = profix(pMix, psi0, pre)</code>	build mixture prediction from projector
<code>pMix = ...</code>	
<code>mixpro(Mix0, pchns, cchns, psi0, pre)</code>	build mixture projection ^a
<code>[pMix, weights] = ...</code>	
<code>profixn(pMix, psi0, pre, nstep)</code>	prediction n-steps ahead ^b
<code>se = relep(Mix, ndat)</code>	prediction errors norm
<code>se = relepn(Mix, nstep)</code>	multistep prediction error norm

^adefaults: pchns - [1,2], cchns - no, psi0 - substituted from DATA, no data scaling
^bthe weights are data dependent even for static mixtures

Visualization ^a	
^a defaults: see Prediction operations. The functions allows definition of grid densities and ranges	
<code>mixplot (Mix, pchns, cchns, psi0, pre)</code>	mixture plot (shaded)
<code>mixplotc(Mix, pchns, cchns, psi0, pre)</code>	mixture plot (contours, components)
<code>[x,y,z] = ...</code>	
<code>mixgrid(Mix, pchns, cchns, psi0, pre)</code>	coordinates for mixture plot
<code>[x,y,z] = datagrid(Mix)</code>	coordinates for data plot
<code>datascan(chns)</code>	scan data for 2 dim clusters
<code>mixmesh(Mix, pchns, cchns, psi0, pre)</code>	mixture mesh plot
<code>mixscan(Mix, chns, pre)</code>	scan mixture for 2 dim. clusters
<code>setaxis(list, ax)</code>	set global axis in subplots ^a
<code>sigscan(chns)</code>	scan signal
<code>fullscreen</code>	set fullscreen for current plot
<code>resizefig()</code>	set plot position

^alist is list of subplots, ax a scaling see axis function

Interactive visualization	
<code>mixshow(Mix)</code>	interactive plot of mixture
<code>mixbrow(Mix)</code>	interactive display of mixture attributes
<code>setdbg('function')</code>	interactive setting of "dbstop"

Data preprocessing	
<code>pre = preproc(pre)</code>	preprocess data
<code>pre = preinit(pre)</code>	initialize preprocessing
<code>pre = prestep(pre)</code>	preprocessing step

Structure estimation	
<code>Mix = ...</code>	
<code>mixstr(Mix, Mix0, belief, nruns)</code>	estimate mixture structure
<code>MAPstr = ...</code>	estimate structure of a factor
<code>facstrid(Fac, Fac0, belief, nbest, nruns)</code>	

Mixture simulation	
--------------------	--

<code>mixsimul(Sim, ndat)</code>	batch mixture simulation
<code>mixsimul(Sim)</code>	recursive mixture simulation
<code>Sim = statsim((ndat, ncom, cove)</code>	create static mixture with components on unit circle

Basic conversion functions	
<code>LD = ltdl(V)</code>	decompose positive definite matrix to L'DL
<code>Mix = mix2mix(Mix, form)</code>	convert mixture to a specified form ^a
<code>Com = com2com(Com, form)</code>	convert component into a specified form
<code>X = arx2arx(X)</code>	convert between ARX and ARX LS representations
^a "form" is a coding summarized in "Codes"	

Design of advisory system	
<code>[aMix, aMixu] = ...</code>	
<code>inisyn(Mix,Mixu,pochn,uchn)</code>	initialize advisory design for normal mixture
<code>[aMix, aMixu] = ...</code>	
<code>inisyn(Mix,Mixu,Chns)</code>	call with channel descriptions
<code>aMix = ...</code>	
<code>aloptim(aMix,aMixu,ufc,nstep,chis)</code>	make academic advisory design for normal mixture
<code>ufc = ufcgen(Mixc, Mixc0)</code>	vector of unstable components
<code>aMix = ...</code>	
<code>soptim(aMix,aMixu,ufc,nstep,chis)</code>	perform simultaneous advisory design
<code>aMix = algen(aMix,aMixu,ufc)</code>	compute of probabilistic weights for advisory design
<code>[Mixu, ychns] = target(Chns)</code>	create user's target mixture
<code>Mix = mixcopy(Mix1, Mix2)</code>	copy of ARX or ARX LS statistics

Channel descriptions	
<code>Chns = chnconst(chns)</code>	build channel descriptions
<code>Chns = chnset(Chns,chns,fld, val)</code>	set channel descriptions field
<code>val = chnget(Chns,chns,fld)</code>	get values of channel descriptions fields

General purpose functions	
<code>prodini</code>	standard Mixtools session beginning
<code>prt(X)</code>	debugging prints
<code>is = equal(X1,X2, eps)</code>	test of equivalence up to a small difference
<code>str = genstr(order, nchn, td)</code>	generation of model structure of given order
<code>is = streq(str1, str2)</code>	compare two structures
<code>is = isstatic(Mix)</code>	test whether mixture is static
<code>is = isdimeq(X1,X2)</code>	test of equality of dimensions
<code>is = streq(str1,str2)</code>	test of equality of dimensions
<code>mversion</code>	display current Mixtools version

Dialog functions	
<code>dial1, dial2, dial3</code>	dialogue units for case studies

Design Base

Estimation related operations	
<code>[Mix, faclls] = mixupdt(Mix, flag, weight)</code>	one step of mixture update
<code>Mix = mixupdt(Mix, flag)</code>	mixture update by projection
<code>Mix = mixestpb(Mix, frg, ndat, niter)</code>	iterative estimation by projection
<code>Mix = mixestqb(Mix, frg, ndat, niter)</code>	iterative quasi-Bayes mixture estimation
<code>Mix = mixestbq(Mix, frg, ndat, niter)</code>	iterative batch quasi-Bayes mixture estimation
<code>Mix = mixestbb(Mix, frg, ndat, niter, nstep)</code>	iterative estimation by forgetting branching
<code>Mix = mixestmt(Mix0, frg, ndat, niter)</code>	quasi-Bayes iterative estimation with fixed variances
<code>Mix = mixestem(Mix0, ndat, niter)</code>	estimation by EM algorithm
<code>Mix = mixfrg(Mix, frg)</code>	mixture forgetting
<code>[Mix0, handle] = ...</code>	
<code>mixflatv(Mix, niter, ndat, frg)</code>	mixture flattening with variable rate
<code>[Mix0, handle] = mixflatv(Mix, handle) ^a</code>	
^a the first call in initialization, the second one in iterations	

Auxiliary estimation operations	
<code>Mix = mixgmean(weights, Mix1, ...)</code>	geometric means of mixtures
<code>dvec = getdvect(Fac)</code>	get data or regression vector
<code>Mix = facupdt(Mix, facs, weights)</code>	factor update
<code>lls = facdpred(Mix)</code>	compute trial factor predictions
<code>[s, s0] = mixdfms(Mix)</code>	sum degrees of freedom of the mixture
<code>Mix0 = mix2mix0(Mix)</code>	create initial mixture mimic to a mixture
<code>lls = loglik(LD, dfm, LD0, dfm0)</code>	increment of loglikelihood
<code>Sim = sim2pdf(Sim, ndat)</code>	convert simulator to estimator

Prediction related operations	
<code>Facs = fac2marg(Facs, pchns)</code>) convert factor into data-marginal factor
<code>Com = com2pro(Facs, pchns, cchns)</code>	convert ARX LS component to predictor
<code>[typ, ychns, ...] = comunpk(...)</code>	get information about component

Preprocessing	
<code>Pre = preaux1(method, time, Pre)</code>	auxiliary function for data pre-processing

Design of advisory system	
<code>Com = arx2ful(Com, str)</code>	weights needed for advisory system design
<code>Com = canarxls(ychns, str)</code>	build matrix ARX LS component
<code>pMix = facchng(pMix, com, Fac)</code>	auxiliary changes of mixture factors
<code>Mix = pro2str(Mix, str)</code>	additional pointers to external structure
<code>... = ricexp(...)</code>	auxiliary function for computing of expectation
<code>... = ricpen(...)</code>	auxiliary function for computing of penalisation
<code>... = ricpenu(...)</code>	computing of penalisation in simualt design
<code>... = ricshift(...)</code>	shift of matrices and vectors
<code>aMix = synmixi(Mix, uchn, strc)</code>	convert mixture to control form aMix

Kullback-Leibler distance	
---------------------------	--

<code>dist = kldist(fac, Mix, Mix0)</code>	distance of a factor in parameter space
<code>dist = kldist(0, Mix, Mix0)</code>	distance of all factors
<code>dist = kldist(Mix, Mix0)</code>	distance of all components
<code>[d1,d2,d3,d4] = kldist(Mix1, Mix2)</code>	distance of mixtures ^a
<code>dist = kldiscom(Mix, ndat)</code>	distance of components in data space
<code>dist = kldcom(Mix, Mix0)</code>	KL distance of components from initial ones
<code>kld = kldisdir(s, s0)</code>	Kulback-Leibler distance of Dirichlet pdfs
<code>kld = kldistc(Mix)</code>	KL distance of components in normal mixture

^a distances: d1 - overall distance, d2 - distances of factors, d3 - distance of components, d4 - distance of component weights

Conversion functions	
<i>Conversion of an array of ARX components to the mixture and back</i>	
<code>Sim = arxc2mix(Coms, dfcs)</code>	convert ARX components to simulator
<code>Coms = mix2arxc(Mix)</code>	convert normal mixture into array of ARX components
<code>Facs = arxc2fac(Com)</code>	convert ARX component to ARX LS factors
<code>Com = fac2arxc(Facs)</code>	convert ARX LS factors to ARX component
<i>Conversions of L'DL decompositions</i>	
<code>V = ld2v(LD)</code>	convert L'DL decomposition to original matrix
<code>LD = ld2ld(L, D)</code>	replace diagonal unit of L by D
<code>[L,D]= ld2ld(LD)</code>	extract D from diagonal LD and replace it by D
<code>LD1 = ldchng(LD, str, LD1, str1)</code>	change part of L'DL decomposition
<i>Conversion of L'DL to LS representations and back</i>	
<code>[Eth,Cth,cove,dfm] = fac2ls(Fac)</code>	convert ARX factor to least square representation
<code>LD = ls2ld(Eth,Cth,cove,dfm)</code>	convert Eth, Cth, cove, dfm into LD
<code>[Eth,Cth,cove]=ld2ls(LD,dfm,nychn)</code>	convert L'DL into Eth, Cth, cove
<i>Subselection from an L'DL decomposition</i>	
<code>LD = ld2ld(LD, str1, str2)</code>	marginal L'DL decomposition ^a
<code>LD = ldperm(LD, i)</code>	permute L'DL decomposition: i-th row to 1st row

^astr1 is source and str2 target LD structure, str2 has to be contained in str1

Operations over triangular matrices

<code>UD = ld2ud(LD)</code>	convert decomposition L'DL to U'DU ^a
<code>UD = utdu(X)</code>	upper triangular U'DU sym. matrix decomposition
<code>UD = ld2ud(LD)</code>	convert decomposition L'DL to U'DU
<code>LD = ud2ld(UD)</code>	convert decomposition U'DU to L'DL
<code>LDi = ldinv(LD)</code>	invert L'DL decomposition
<code>ut = udiv(ut)</code>	invert upper triangular matrix
<code>LD = ldupdt(LD , dvect , weight)</code>	update L'DL decomposition by weighted data vector
<code>UD = udupdt(UD , dvect , weight)</code>	update U'DU decomposition by weighted data vector
<code>[Eth, cove] = udform(Eth, cove)</code>	restore matrix factorized ARX component

^athe decomposition U'DU, U is upper triangular with unit diagonal, V = U'DU. "UD" is the upper triangular matrix with "D" on diagonal

Factorized matrix ARX and matrix LS components

<code>Can = arxc2can(Com)</code>	convert ARX LS to matrix factorized ARX component
<code>Com = can2arxc(Can, n)</code>	^a convert "Can" into matrix ARX LS component
<code>Can = can2marg(Can)</code>	permute matrix factorized ARX component

^a"n" is number of marginal channels

Visualisation

<code>statmesh(Mix)</code>	interactive mesh plot of static mixture or data
<code>statplot(Mix)</code>	plot components of static mixture components
<code>[x,y,z] = statgrid(Mix)</code>	coordinates grid for 3-D display
<code>complot(Mix, com)</code>	plot of component of a mixture
<code>iterplot(Mix0, Mix, iter)</code>	plot initial and resulting mixture of an iteration step
<code>setfig(number)</code>	set figures windows
<code>fixerr(Mix)</code>	interactive set TIME for plots

Dump/restore of a MATLAB array

<code>mixdump(Mix, filename,...)</code>	dump MATLAB object to disk
<code>Mix = restore(filename,...)</code>	restore dumped MATLAB object

General purpose functions

<code>val = defaults('item')</code>	get values from database of defaults
<code>e = noise(etyp)</code>	generate a random number with a "etyp" distribution ^a
<code>val = gauss1(dvect,Eth,cove)</code>	value of one-dimensional Gaussian pdf
<code>val = gaussn(dvect,Eth,cove)</code>	value of Gaussian pdf
<code>setfig(n)</code>	set figures windows
<code>val = getflds(cell_vect, 'field')</code>	get fields from a cell-vector of structures
<code>val = betaln(p,q)</code>	logarithm of Euler's beta function
<code>fac = facsort(Facs)</code>	sort factors of a component

^agenerators have mean 0 and covariance 1 (with exception of Cauchy); the etyp is: 1 - Gaussian, 2 - uniform, 3 - lognormal, 4 - Cauchy

[algen](#)

ALGEN compute probabilistic weights for academic advisory design,

aMix = algen(aMix, ufc)

aMix : advised mixture of the type ARX LS enriched on the following control states:

 strc : common control structure

 ufc : normalised vector qualifying components:

 dangerous component (0), not dangerous (positive number)

 kc : lift of quadratic forms

 UDc : cell vector of u'du decompositions of KLD kernels

 udca : u'du decomposition of average KLD kernel in UDc

 kca : average lift of quadratic forms kc

 dfcs : degrees of freedom of components (weights)

aMixu : desired mixture of the type ARX LS with control states

ufc : normalised vector qualifying components

Design : J. Bohm

Updated : June, 2002

Project : ProDaCTools, IST-1999-12058

See also : [udupdt](#), [getdvect](#)

[aloptim](#)

ALOPTIM perform academic advisory design for normal mixture

```
[aMix] = aloptim(aMix, aMixu, ufc, nstep, chis)
[aMix] = aloptim(aMix, aMixu, ufc, nstep) chis = 1
[aMix] = aloptim(aMix, aMixu, ufc)            nstep = [200, 1]
```

aMix : advised mixture of the type ARX LS enriched on following control states:
 strc : common control structure
 ufc : normalised vector qualifying components:
 dangerous component (0), not dangerous (positive number)
 kc : lift of quadratic forms
 UDc : cell vector of u'du decompositions of KLD kernels
 udca : u'du decomposition of average KLD kernel in UDc
 kca : average lift of quadratic forms kc

aMixu : desired mixture (user's target) of the type ARX LS with control states

ufc : vector qualifying components: 0 - dangerous component, (1) - not

nstep : parameters [ns1,per] determining design horizon, i.e. horizon = ns1*per;

 ns1 : number of block repetition

 per : horizon of a block

 if nstep is defined by parameter ns1 only then per is set to 1

chis : indicates strategy chosen: chis=1 for receding horizon (default) and chis=-1 for

Design : J. Bohm

Updated : June, 2002

Project : ProDaCTools, IST-1999-12058

See also : [udupdt](#), [getdvect](#)

arx2arx

conversion between ARX and ARX LS representations

$X = \text{arx2arx}(X)$

X: factor, component, mixture: recognised by the associated type
(types are described in the Guide)

Design : P. Nedoma

Updated: June 2001

Project: ProDaCTools

arx2ful

re-build the matrix ARX LS component
`Com1 = arx2ful(Com, str)`

`Com1` : rebuilt component
`Com` : component to be rebuilt
`str` : structure of the rebuilt component

Design : P. Nedoma
Updated: April 2002
Project: ProDaCTools

arxc2can

conversion of matrix ARX LS component to the factorized LS form
Can = arxc2can(Com)

Can : factorized ARX LS component
Com : matrix ARX LS component

Design : P. Nedoma
Updated: March 2001
Project: ProDaCTools

arxc2fac

conversion of matrix ARX LS component to ARX LS component
Facs = arxc2fac(Com)

Facs : ARX LS component, i.e array of ARX LS factors
Com : matrix ARX LS component

Design : P. Nedoma
Updated: September 2001
Project: ProDaCTools

arxc2mix

convert array of ARX components into mixture

```
Sim = arxc2mix(Coms, dfcs)
```

Sim : resulting mixture

Coms : components that create mixture

dfcs : degrees of freedom determining component weights

Design : P. Nedoma

Updated: October 2000

Project: ProDaCTools

[betaln](#)

logarithm of Euler's beta function

```
value=betaln(p,q)
```

value : value of the beta function

p,q : positive argumens of the beta function, order does not matter

Designed: MK

Updated : July 02

Project : ProDaCTool

See also: [gammaln](#)

bmtbinit

initialization of mixture estimation by BMTB algorithm
Mix1 = bmtbinit(Com, ndat, delta)

Mix1 : estimated mixture
Com : initial cell-vector of static components
(default: 10 components)
ndat : size of data sample (default: length of global DATA)
delta : ratio of window size with respect to +- std of data
(default: 0.8)

Design : I. Nagy
Updated: May 2002
Project: ProDaCTools

can2arxc

convert matrix factorized ARX LS component into matrix ARX LS component
Com = can2arxc(Can)

Com : matrix factorized ARX LS component
Can : matrix ARX LS component

Design : P. Nedoma
Updated: November 2002
Project: ProDaCTools

can2fac

convert matrix factorized ARX LS component to ARX LS component

```
Facs = can2ls(Can,eps)
```

```
Facs = can2ls(Can)      eps = 1e-16
```

Can : matrix factorized ARX LS component

Facs : ARX LS component

eps : thrshold determining the nonzero regression coefficient

Design : P. Nedoma

Updated: December 2002

Project: ProDaCTools

can2marg

re-organize matrix factorized component for prediction
Can = can2marg(Can, pchns)

Can : re-organized factorized matrix ARX LS component
Can : source, factorized matrix ARX LS component
pchns : marginal (predicted) channels

Design : P. Nedoma
Updated: March 2001
Project: ProDaCTools

canarxls

```
build matrix ARX LS component  
Com = comarxls(ychns, str)
```

```
Com      : built ARX LS component  
ychns    : modelled channels  
str      : model structure
```

```
Autor   : P. Nedoma  
Updated: May 2000  
Project: ProDaCTools
```


cellcat

concatenate complex cell lists

```
F = cellcat(F0)
```

F : cell vector of structures

F0 : cell vector containing cell vectors and structures

Design : P. Nedoma

Updated: February 2002

Project: ProDaCTools

[chnconst](#)

```
build channels description  
Chns = chnbldl(chns)
```

```
Chns : channel description: cell vector of individual  
      channel descriptions, for their structure see chnbldl  
chn   : channels (vector of positive integers)
```

```
Design  : P. Nedoma, M. Novak  
Updated : March 2003, MK July 2004  
Project : post-ProDaCTool
```

```
Problem: is this still valid?  
See also: chnbldl, chnget, chnset
```

[chnget](#)

get values of channel descriptions fields
values = chnget(Chns, chns, field)

values: returned values in the form required for chnset

Chns : channels description: cell vector of structures

	1	2	3	4	5	6	7	8	9	10
	chn	name	oitem	raction	prty	type	drange	prange	irange	scale

chns : list of channels inspected, when empty all channels are considered
field : field of the channel description

Design : P. Nedoma

Updated : July 2002, MK July 2004

Project : post-ProDaCTools

See also: [chnconst](#), [chnset](#)

[chnset](#)

set channel descriptions field

Chns = setchns(Chns, chns, field, val)

Chns : updated channels description, cell vector of structures

	1	2	3	4	5	6	7	8	9	10	11	12
	chn	name	oitem	raction	prty	type	drange	prange	irange	scale	prior	reference

chns : list of channels, if empty all channels
or defined from connection ????

field: field of the channel description

val : values to be set

Agregate setting for prior and scale: ????

Chns = setchns(Chns, field, val) ????

Design : P. Nedoma, M.Novak

Updated : March 2003, MK July 2004

Project : post-ProDaCTools

See also: [chnconst](#), [chnget](#)

com2can

convert ARX LS component into matrix factorized ARX LS component
[Can, ok] = com2can(Facs)

Can : matrix factorized component
ok : 1: the factors were already in the form required
 : 0: a real transformation was made
Facs : factorized component of the type 12 or 112

Design : P. Nedoma
Updated: November 2002
Project: ProDaCTools

com2com

convert component to a specified form
Com1 = com2com(Com, form)

Com1 : the resulting component having the type form

Com : component to be converted

form : coded component form:

11 - ARX

12 - ARX LS

13 - matrix ARX

14 - matrix ARX LS

+ 100 for predictor component

Design : P. Nedoma

Updated: June 2001

Project: ProDaCTools

com2pro

```
convert ARX LS component to predictor (112)
[Facs, comaux] = com2pro(Facs, pchns, cchns)
```

```
Facs  : ARX LS component (array of ARX LS factors)
pchns : predicted channels
cchns : channels in condition
Facs  : the resulting predictor
comaux: { matstr, pcove, ih, ic, pchns, cchns, itd0 }
        build at the end of the function
        matstr - matrix component structure
        pcove  - matrix component cove
        ih     - pointers to history in matstr
        ic     - pointers to all zero-delayed entries in matstr
        pchns  - memory of pchns
        cchns  - memory of cchns
        itd0   - pointer to not modelled zero-delayed entry in matstr
```

Design : P. Nedoma

Updated: May 2003

Project: ProDaCTools remake

[comarx](#)

build matrix ARX component
Com = comarx(ychns, str)

Com : matrix ARX component, type = 13
ychns : modeled channels
str : common regressor structure

Design : P. Nedoma
Updated : June 2001
Project : ProDaCTools
See also: [comarxls](#), [facarx](#)

[comarxls](#)

build matrix ARX LS component

```
Com = comarxls(ychns, str)
```

Com : matrix ARX LS component, type = 14

ychns : modeled channels

str : model structure

Design : P. Nedoma

Updated : May 2000

Project : ProDaCTools

See also: [comarx](#), [facarx](#)

comdel

cancel the specified component

```
Mix = comdel(Mix, com)
```

Mix : mixture without the specified component

com : component to be deleted

Mix : source mixture

[commerge](#)

optimum merging of mixture components or possibly with enforced range

```
[Mix, Mix0, changed] = commerge(Mix, Mix0, nlow, nhigh)
```

```
[Mix, Mix0, changed] = commerge(Mix, Mix0, nlow) nhigh = nlow
```

```
[Mix, Mix0, changed] = commerge(Mix, Mix0)           range unspecified
```

Mix : merged estimated mixture

Mix0 : merged initial mixture

changed : 0 - mixture not changed, otherwise 1

Mix : estimated mixture

Mix0 : initial mixture

nlow : lower bound on number of components to be preserved

nhigh : upper bound on number of components to be preserved

Note : merging is made when there is a chance for increasing of
v-likelihood; it acts as counterpart of mixsplit

Design : M. Kary, P. Nedoma

Updated : September 2002

Project : ProDaCTools

See also: [mixinit](#), [mixsplit](#), [mixcut](#)

complot

plot contour of level*std probability of a two-dimensional ARX or ARX LS static component
complot(Mix,com, level)

Mix = inspected mixture
com = considered component
level = probability level

Designed : P. Nedoma
Updated : May 2000, July 2004
Project : Prodactools

comunpk

get information about a component

```
[typ, ychns, notmodelled, zerodelayed, str] = comunpk(Com)
[typ, ychns]                                = comunpk(Com)
```

```
typ          : component type
ychns        : modelled channels
notmodelled  : not modelled channels
zerodelayed  : zero-delayed not modelled channels
str          : component structure (union of factors structures)
Com          : component of any form
```

Design : P. Nedomá

Updated: June 2001

Project: ProDaCTools

Calls : unique, setdiff

Updated : August 2001, MK added comments and problems checked

Problems: solved by PN Sempember 2001

credit

evaluation of an credibility interval

```
[Cl,Cu,Chat] = credit(C,beta)
```

Cl = lower credibility bound

Cu = upper credibility bound

Chat = center of the credibility interval

C = vector of independent realizations

beta = credibility level in (0,1)

designed : PN

updated : 15.07.04

reference:

credits

evaluation of an credibility interval with stopping

```
[Cl,Cu,Chat,flag,st] = credits(C,beta,epsi)
```

Cl = lower credibility bound

Cu = upper credibility bound

Chat = center of the credibility interval

flag = stopping flag [1/0] = stop/do_not_stop data acquisition

Q = value of the stopping test statistic

C = no-vector of independent realizations

beta = credibility level in (0.5,1)

epsi = upper bound on relative error

datagrid

compute histogram coordinates for DATA having two channels

```
[x,y,z] = datagrid(chns, nx, ny, rx, ry)
```

```
[x,y,z] = datagrid(chns, nx, ny)
```

```
[x,y,z] = datagrid(chns)
```

```
[x,y,z] = datagrid
```

x, y, z : computed coordinates

chns : considered channels

nx,ny : number of poits of grids at x and y directions

rx,ry : grid ranges (lower and upper bound)

Design : P. Nedoma

Updated: June 2001

Project: ProDaCTools

dataplot

plot data of 1 or 2 channels

```
dataplot(chns, nlev)
```

```
dataplot(chns)      nlev = 20
```

```
dataplot           chns = [1,2]
```

chns : channels displayed

nlev : the number of histogram boxes

designed : PN

updated : 15.07.04

`datascan`

```
scan data for 2 dimensional clusters  
datascan(chns)  
datascan chns = all channels
```

```
chns : list of channels
```

Design : P. Nedoma

Updated: July 2002

Project: ProDaCTools

defaults

database of processing defaults

use : varargout = defaults(default)

vargarout: supplied defaults: meaning and number depends on the input argument

default : char string, 1st letter decisive:

- 'a' : ARX LS factor
- 'A' : alternative ARX LS (used in prior)
- 'b' : ARX LS flat ("basr") mixture
- 'B' : ARX LS the most flat possible (used in prior)
- 'm' : Markov-chain factor
- 'x' : mixture
- 's' : structure estimation
- 'f' : forgetting
- 'i' : mixture initialization (MIXINIT)
- 'e' : mixture estimation
- 'E' : estimation with stopping rules
- 'm' : dfm for markov model

Design : P. Nedoma

Updated : 10.2.00

Project : ProDaCTools

See also: [comarx](#), [mixinit](#)

[dial1](#)

No help comments found in dial1.m.

[dial2](#)

No help comments found in dial2.m.

[dial3](#)

No help comments found in dial3.m.

diff-tg

DIFF_tg sets difference of vectors A,B; the result is NOT sorted
[C] = diff_tg(A,B)

C : vector containing those values of A which are not in B
A,B : compared vectors

Design : T.V.Guy
Updated : September 2002
Project : ProDaCTools

dydrs

DYDRS dyadic reduction, performs transformation of sum of 2 dyads

```
[rout, fout, Drout, Dfout, kr] = dydrs(r,f,Dr,Df,R,jl,jh);  
[rout, fout, Drout, Dfout] = dydrs(r,f,Dr,Df,R);
```

Description: dyadic reduction, performs transformation of sum of 2 dyads $r*Dr*r' + f*Df*f'$ so that the element of r pointed by R is zeroed

r : column vector of reduced dyad
f : column vector of reducing dyad
Dr : scalar with weight of reduced dyad
Df : scalar with weight of reducing dyad
R : scalar number giving 1 based index to the element of r,
which is to be reduced to
zero; the corresponding element of f is assumed to be 1.
jl : lower index of the range within which the dyads are
modified (can be omitted, then everything is updated)
jh : upper index of the range within which the dyads are
modified (can be omitted then everything is updated)
rout,fout,Drout,dfout : resulting two dyads
kr : coefficient used in the transformation of r
rnew = r + kr*f

Description: dyadic reduction, performs transformation of sum of 2 dyads $r*Dr*r' + f*Df*f'$ so that the element of r indexed by R is zeroed

Remark1: Constant mzero means machine zero and should be modified according to the precision of particular machine

Remark2: jl and jh are, in fact, obsolete. It takes longer time to compute them compared to plain version. The reason is that we are doing vector operations in m-file. Other reason is that we need to copy whole vector anyway. It can save half of time for c-file, if you use it correctly. (please do tests)

Remark3: indexes jl, jh are 1'based as opposed to c-version, where they are zero-based

Original Fortran design: V. Peterka 17-7-89

Modified for c-language: probably R. Kulhavy

Modified for m-language: L. Tesar 2/2003

Updated: Feb 2003

Jan 2004 comments updated

Project: post-ProDaCTool

Reference: none

equal

test entry-wise equality of matrices (up to a small difference) or structures
res = equal(a,b,eps)
res = equal(a,b) eps = MATLAB eps

res : res = 0 if matrices differ (then a message is printed) otherwise res = 1
a,b : matrices or structures to be compared
eps : maximum difference

designed : J. Andrysek, P. Nedoma
updated : December 2001
project : ProDaCTools

estfrg

Select the best forgetting rate for the mixture

```
[Mix, frg, mixlls] = estfrg(Mix0,frgs,ndat,niter,method,Mixa)
                                niter = 1; method = 'q'; enforced
[Mix, frg, mixlls] = estfrg(Mix0,frgs,ndat,niter,method)
                                exponential forgetting used
[Mix, frg, mixlls] = estfrg(Mix0,frgs,ndat,niter) method = quasi-Bayes ('q')
[Mix, frg, mixlls] = estfrg(Mix0,frgs,ndat)        niter = 1
[Mix, frg, mixlls] = estfrg(Mix0,frgs)             ndat = size(DATA,2)
[Mix, frg, mixlls] = estfrg(Mix0)                  frgs = (1+1e-3)-logspace(-3,-1,25)
```

Mix : mixture estimated with the best forgetting
frg : forgetting rate selected
mixlls : array of posterior data log-likelihood
frgs : inspected vector of forgetting rates

Mix0 : initial mixture
frgs : vector of forgetting rates (if missing or empty, frgs = (1+1e-3)-logspace(-3,-1,25))
ndat : number of data items (if missing or empty, ndat = size(DATA,2))
niter : number of iterations (if missing or empty, niter = 1)
method: the used estimation method (if missing or empty, method = 'q' = quasi-Bayes)
Mixa : mixture for alternative forgetting
(if missing, exponential forgetting is used)

Design : L. Pavelkova
Updated : May 2002, July 2004
Project : Designer, Baddy
Reference:

[fac2arxc](#)

convert ARX or ARX LS component into matrix ARX LS component
Com = fac2arxc(Facs)

Com : matrix ARX LS component
Facs: array of ARX or ARX LS factors

Design : P. Nedoma
Updated: December 2003
Project: ProDaCTools

[fac2ls](#)

conversion of factor to LS representation
`[Eth, Cth, cove, dfm] = fac2ls(Fac)`

Eth : LS estimate of regression coefficients
Cth : LS covariance matrix of Eth (LDL' decomposition)
cove : estimate of noise variance (LDL' decomposition)
dfm : degrees of freedom
Fac : the ARX factor to be converted

Designed : P. Nedoma
Updated : 16.2.00, August 2001, 17.07.04
Project : ProDaCTools, Baddy
See also : [ld2ls](#)

[fac2marg](#)

build marginal factors

```
Facs = fac2marg(Facs0, mchns)
```

Facs : array of marginal factors

Facs0 : component in the form of factor array

mchns : marginal channels to be evaluated

Designed : P. Nedoma

Updated : November 2000, July 2004

Project : ProDaCTools, Baddy

Calls : fac2cf, udform

[facarx](#)

build ARX-factor FOR PRIOR PROCESSING

```
Fac = facarx(ychn, str, default);
```

```
Fac = facarx(ychn, str)          default = 'A'
```

Fac : ARX factor, type = 1

ychn : modeled channel

str : factor structure

default : string determining type of default values to be used, see defaults.m

Design : P. Nedoma

Updated : 20.3.00, June 2002, July 2004, MK

Project : ProDaCTools

Note : constructor modified for selection of defaults

Reference:

See also : [comarx](#), [comarxls](#)

[facarxls](#)

```
build ARX LS factor
Fac = facarxls(ychn, str,default)
Fac = facarxls(ychn, str) default = 'A'
```

```
Fac      : ARX LS factor      type = 2
ychn     : output channel
str      : factor structure
default  : string determining type of default values to be used, see defaults.m
```

```
Design   : P. Nedoma
Updated  : 9.3.00
Project  : ProDaCTools
```

Reference:

```
Updated  : June 02, MK
```

```
See also: facarx, comarx, comarxls, defaults
```


facchng

replace predictor factor in a given component
`pMix = facchng(pMix, com, Fac)`

`pMix` : mixture predictor
`com` : component (number)
`Fac` : ARX LS factor

Design : P. Nedoma
Updated: March 2002, July 2004
Project: ProDaCTools

facdpred

FACDPRED computes logarithm of factor predictions

```
faclls = facdpred(Mix)
```

Mix : mixture containing the inspected factors

faclls : logarithms of factor predictions

```
log( f( dt | fac) )=log(pfd(current data|past,factor))
```

Design : L. Tesar, Jan 2004

Note : equals to logarithm of normalizing integral of virtually updated estimate
- logarithm of the current normalizing integral

Note1 : based on facdpred.c by and facdpred1() from mexlib (both by P. Nedoma)

Calls: facupred

Calls inline: facpred1

[facflat](#)

factor flattening

```
Fac0 = facflat(Fac, rate, FacA)
```

```
Fac0 = facflat(Fac, rate)      completely flat alternative factor is used
```

Fac0 : flattened factor

Fac : estimated factor

rate : factor flattening rate

FacA : alternative factor used in flattening

Design : M. Karny, P. Nedoma

Updated : January 2003, July 2004

Project : ProDaCTools, Baddy

Note : function of the construction base, the default rate is
computed in the calling function

See also: [mixflat](#), [mixflatv](#)

facfrg

factor forgetting

Fac = facfrg(Fac, rate, FacA)

Fac = facfrg(Fac, rate) exponential forgetting is used

Fac : forgotten factor

rate : forgetting rate

FacA : alternative factor used in the stabilized forgetting

Design : P. Nedoma

Updated: April 2003, July 2004

Project: ProDaCTools remake

facgmean

geometric mean of factors: $\text{Fac} = \text{weight}(1) * \text{Fac1} + \text{weight}(2) * \text{Fac2}$

`Fac = facgmean(Fac1, Fac2, [weight, weight1])`

`Fac = facgmean(Fac1, Fac2, weight) weight1=1-weight`

`Fac = facmerge(Fac1, Fac2) weight1=weight=1`

`Fac` : the resulting factor

`Fac1, Fac2` : factors of the same structure (not checked!)

`weight` : weight in [0,1]

`weight1` : weight in [0,1]

`Design` : P. Nedoma

`Updated` : April 2003, July 2004

`Project` : ProDaCTool remake

facmark

build Markov factor

```
Fac = facmark(ychn, str, levels);
```

Fac : Markov factor

ychn : factor output

str : factor structure

levels : maximum number of values for all channels

Designed : P. Nedoma

Updated : 10.2.00, August 2001, July 2004 MK

Project : ProDaCTools and its continuations

facmerge

```
merge factors Fac = weight*Fac1+weight1*Fac2
Fac = facmerge(Fac1, Fac2, weight, weight1)
Fac = facmerge(Fac1, Fac2, weight) weight1=weight
Fac = facmerge(Fac1, Fac2)          weight1=weight=1
```

Fac : the resulting factor
Fac1, Fac2 : factors of the same structure (not checked!)
weight : weight in [0,1]
weight1 : weight in [0,1]

Design : P. Nedoma

Updated : MK, January 2001, April 2002, July 2004

Project : Prodactools and its continuation

Reference:

Problems : MK check of the common structure should be added

See also: [mixinit](#), [commerge](#), [scalpri](#)

facpred

FACPRED computes logarithm of $\int f(d_t|d(t-1), \theta)^w f(\theta|d(t-1)) d\theta$
[LH,FSC] = facpred(Fac,w,Psi)
[LH,FSC] = facpred(Fac,w) Psi=getdvect(Fac)
[LH,FSC] = facpred(Fac) w=1

LH : logarithm of $\int f(d_t|d(t-1), \theta)^w f(\theta|d(t-1)) d\theta$
logarithm factor prediction $f(d_t | fac)$ for $w=1$
FSC : factor specific characteristics - depends on factor type

Fac : factor, for which the value is evaluated
w : exponent in evaluated expression $w=1$
Psi : data vector of the factor Psi=getdvect(Fac);

Design : J. Andrysek, Sep 2004
Project: BadDyr

Note : works for normal factors and MT-normal factors
Note2 : MT-normal factor ignores the input w

facsort

sort factors of a component (in order how they can appear in pdf)
fac = facsort(Facs)

fac: numbers of sorted factors
Facs: component factors

Designed : P. Nedoma
Updated : April 2000, MK August 2001, July 2004
Project : ProDaCTools and its continuation

Calls : sorttree

facstr

FACSTR factor-structure estimation

```
[optstr, lhs] = facstr(Fac, Fac0, belief, nbest, max_nrep, uchns, lambda, order_k)
[optstr, lhs] = facstr(Fac, Fac0, belief, nbest, max_nrep, uchns, lambda) order_k=2
[optstr, lhs] = facstr(Fac, Fac0, belief, nbest, max_nrep) uchns) lambda = 0.9
[optstr, lhs] = facstr(Fac, Fac0, belief, nbest, max_nrep) uchns = []
[optstr, lhs] = facstr(Fac, Fac0, belief, nbest) max_nrep = 100 !!!!! change this to 500
[optstr, lhs] = facstr(Fac, Fac0, belief) nbest = 1
[optstr, lhs] = facstr(Fac, Fac0) belief = 2
```

defaults generated by function defaults with the option 's'

```
optstr: maximum a posteriori probability estimate of factor structure
%% lhs : cell vector of the best regressors -- not true now
%% {1} values
%% {2} indicators
```

```
Fac      : factor          type = 1
Fac0     : initial factor  type = 1
belief   : user's belief on maximum structure items
           (1 items must be present, 2 items are probably present
           4 items must not be present, 3 items are probably not present)
nbest    : how many "best" regressors are maintained
max_nrep : maximal number of random starts in search for the best
           structure
uchns    : list of input channels - if specified, the resulting structure
           contains at least one of inputs (suboptimal solution)
           Note: channel description can be used instead of "uchns"
lambda   : stooping rule threshold
order_k  : order of k
```

```
Design  : L. Tesar. Interface Based on P. Nedoma's previous version of facstrid.
Updated : 14.4.2003 - 10.9.2003, MK July 2004
Project : post-ProDaCTool
Reference: straux1
```

facupdt

```
update factor statistics  
[Fac, ep] = facupdt(Fac, weight)
```

Design : P. Nedoma
Updated : April 2003
Project : ProDaCTools cnt.

facupdtp

```
[Fac ,er]=myfacupdt(Fac,w,ep,zeta);
```

facupred

FACUPRED computes logarithm of one factor prediction (trial (virtual) factor prediction)
LH = facupred(Fac)

LH : logarithm of factor prediction
 $\log(f(dt | fac)) = \log(pfd(\text{current data} | \text{past}, \text{factor}))$
Fac : factor, for which logarithm of factor prediction is calculated

Design : L. Tesar, Jan 2004

Updated: MK July 2004

Project: post-PorDaCTool

Note : LH = logarithm of normalizing integral of virtually updated estimate
 - logarithm of the current normalizing integral

Note1 : based upon facupred() from mexlib.c by P. Nedoma

Calls inline: redultx

IDENTICAL WITH SOME OTHER FUNCTION???

facvll

```
compute data log-likelihood  
vll = facvll(Fac, Fac0)  
vll = facvll(Fac)  the initial vll is set to zero
```

```
vll : log-likelihood  
Fac : estimated factor  
Fac0: initial factor - if not specified, a relative vll is computed
```

Design : P. Nedoma
Updated: June 2003, July 2004
Project: post-ProDaCTools

Problem: $0.5 \cdot \log(2 \cdot \pi)$ factor should be checked everywhere
here even $0.5 \cdot (\text{dfm} - \text{dfm0}) \cdot \log(2 \cdot \pi)$ seems to be missing

floutm

Outlier filtration by mixture estimation
n_out=filout(ord)

DATA : this GLOBAL matrix is modified by filout
n_out : the number of filtered outliers
ord : the order of the data component

Filtration is based on modelling of corrupted data by a mixture with two components. One of them, with a priori small noise variance, models normal data, the second is initialized with big noise variance models outliers. Prediction by the normal components replaces a detected outlier.

Design : P. Nemcova
Updated: March 2003, MK July 2004
Project: ProDaCTools
Problem: FIXED values (alfa etc) have to be taken from defaults

fixerr

```
fix errors in mixture-estimation while TIME<101  
fixerr(Mix)
```

Mix : processed mixture

designed : PN
updated : MK July 2004
project : post-ProDaCTool

fullscreen

Set a figure size to completely fill the screen

```
fullscreen(h)
```

fullscreen h = handle of the current figure

h : the handle of the figure that will fill completely the screen

designed : JA

updated : July 2004

[gauss1](#)

evaluate values of one-dimensional Gaussian pdf on a grid
`p = gauss1(x,m,R)`

`p` : values of the pdf
`x` : grid of arguments on which values are asked
`m` : mean value defining the pdf
`R` : variance defining pdf

Design : P. Nedoma

Updated : August 2001, June 2002, MK

Project : Prodactools

See also: [gaussn](#)

[gaussn](#)

values of multivariate Gaussian density pdf

`p = GausN(x,m,R)`

`p` : vector of values

`x` : matrix vector of argument: i-th column = i-th argument

`m` : column of the mean defining the pdf

`R` : L'DL decomposition of the covariance matrix defining pdf

Design : MK

Updated : August 01, June 2002, MK

Project : Prodoctools

See also: [gauss1](#)

genmixe

```
generates mixture for identification (type 24)
Mix = genmixe(ncom, ychns, str, ndat, diagCth, diagcove, dfm)
Mix = genmixe(ncom, ychns, str, ndat, diagCth, diagcove) dfm=100
Mix = genmixe(ncom, ychns, str, ndat, diagCth)          diagcove=0.01
Mix = genmixe(ncom, ychns, str, ndat)                  diagCth=1
Mix = genmixe(ncom, ychns, str)                        ndat=size(DATA,1)
Mix = genmixe(ncom, ychns)                             str=[0;1]
Mix = genmixe(ncom)                                    ychns=1:size(DATA,1)
Mix = genmixe;                                          ncom = 1;
```

```
Mix      : generated mixture (type 24)
ncom     : number of components
ychns    : modeled channels
str      : common component structure
ndat     : size of data sample
diagCth: diagcove  setting of Cth and cove
```

```
Design : P. Nedoma
Updated: January 2002, MK July 2004
Project: ProDaCTools
```

[genmixel](#)

generates initial mixture for identification (type 24) at random positions
suits for normalized data

```
Mix = genmixe(ncom,ychns,str,ndat,diagCth,diagcove,dfm,dfcs)
Mix = genmixe(ncom,ychns,str,ndat,diagCth,diagcove,dfm) dfcs    =0.1ndat
Mix = genmixe(ncom,ychns,str,ndat,diagCth,diagcove)      dfm     =100
Mix = genmixe(ncom,ychns,str,ndat,diagCth)              diagcove=0.0001
Mix = genmixe(ncom,ychns,str,ndat)                      diagCth  =10000
Mix = genmixe(ncom,ychns,str)                           ndat    =size(DATA,2)
Mix = genmixe(ncom,ychns)                               str     =[0;1]
Mix = genmixe(ncom)                                     ychns   =1:size(DATA,1)
Mix = genmixe                                           ncom    = 1
```

Mix : generated mixture, type = 24
ncom : number of components
ychns : modeled channels
str : common structure of all components
ndat : size of data sample
diagCth : diagonal value of LS covariance factor
diagcove : diagonal value of LS estimate of noise covariance
dfm : degrees of freedom of factors (common)
dfcs : degrees of freedom of component weights

Design : P. Nedoma
Updated : January 2002, July 2004
Project : post-ProDaCTools
See also: [genmxi](#), [defaults](#)

[genmxi](#)

```
generates mixture for initialization (type 24)
Mix = genmxi(ychns, str, ndat)
Mix = genmxi(ychns, str)      ndat=size(DATA,2)
Mix = genmxi(ychns)          str=[0;1]
Mix = genmxi                  ychns=1:size(DATA,1)
```

```
Mix    : constructed mixture, type = 24
ychns  : modeled channels
str     : common component structure
ndat    : size of data sample
```

```
Design  : I. Nagy, P. Nedoma
Updated : January 2002
Project : ProDaCTools
See also: genmixe
```

genstr

generate model structure for given model order, channels and common delay
str=genstr(ord,ychns,npr)
str=genstr(ord,ychns) npr = 0

str : model structure, offset always present
ord : common model order
ychns : vector of channels; if scalar then the number of channels
npr : common transportation delay

Design : I. Nagy, P. Nedoma
Updated: August 2002, MK July 2004
Project: post-ProDaCTools

gentab

generate Markov transition table for Metropolis' algorithm
tab = gentab(dim, dia)

tab : Markov transition table
dim : size of tab, dim>1, number of components
dia : common diagonal value of tab, dia in (0,1)
 dia = 0.99 if it is not specified within this interval

designed = PN
updated = July 2004, MK
project = post-PorDaCTool

getdvect

get data vector used in factor
dvect = facgetdv(Fac)

dvect : data vector made of current data values according to Fac.stru
Fac : either directly stru or the considered factor

Design : P. Nedoma
Updated : June 2003, MK July 2004
Project : post-ProDaCTools

getflds

```
get fields from cell vector of structures  
flds = getflds(C, fld)
```

```
C      : cell vector of structures  
fld    : field (character string)  
flds   : cell vector of the fields
```

Designed = P. Nedoma

Updated = May 2000, MK August 2001, July 2004

Project = post-Prodactools

[getth](#)

construction of parameter estimates from "Et" produced by Profix
th=GetTh(Et)

th : parameter estimates

Et : the first output returned by Profix

designed : PN

updated : July 2004

project : post-ProDaCTool

See also : [profix](#)

[inisyn](#)

initialize advisory design for normal mixture

[aMix,aMixu] = inisyn(Mix, Mixu, varargin)

aMix : advised mixture of the type ARX LS enriched on the following control states:

 strc : common control structure

 ufc : normalised vector qualifying components:

 dangerous component (0), not dangerous (positive number)

 kc : lift of quadratic forms

 UDc : cell vector of u'du decompositions of KLD kernels

 udca : u'du decomposition of average KLD kernel in UDc

 kca : average lift of quadratic forms kc

aMixu : desired mixture of the type ARX LS with control states

Mix : mixture estimate, any type

Mixu : desired mixture, any type

input argument varargin may contain either:

 Chns : cell vector of channels descriptions

 or

 pochn : list of channels with o-innovations (vector)

 uchn : list of channels with recognisable actions (vector)

Design : J. Bohm

Updated : August, 2002, TG October 2002, MK July 2004

Project : post-ProDaCTools

See also : [synmxi](#)

isdimeq

check whether dimensions of 2 arrays are equal
ok = isdimeq(a,b)

a,b : checked arrays (or cell vectors or structures)
ok : 1 - dimensions of a and b are equal
 0 - dimensions are not equal

Design : P. Nedoma

Updated: April 2001, MK July 2004

Project: post-ProDaCTools

[isstatic](#)

check whether the mixture is static one
`is = isstatic(Mix)`

Mix : inspected mixture
is : 1 - mixture is static
 0 - mixture is dynamic

Design : P. Nedoma

Updated : April 2001, September 2001, MK July 2004

Project : post-ProDaCTools

Overloaded methods

[help frd/isstatic.m](#)

[help lti/isstatic.m](#)

[help ss/isstatic.m](#)

[help tf/isstatic.m](#)

[help zpk/isstatic.m](#)

See also: [isfactor](#)

iterplot

plot evolution of mixture estimate during iterative learning
iterplot(Mix0, Mix, iter, time)
iterplot(Mix0, Mix, iter) time = TIME-1

Mix0 : mixture before iteration
Mix : mixture after iteration
iter : iteration counter
time : time at which is drawing done

designed : PN
updated : MK July 2004
project : post-Prodactool

[kldcom](#)

KL divergence of components from initial components

```
dist = kldcom(Mix, Mix0)
```

```
dist = kldcom(Mix)          Mix0 = mixflat(Mix)
```

dist : vector KL divergences of components

Mix : estimated mixture

Mix0 : initial mixture

Designed : P. Nedoma

Updated : June 2002, MK July 2004

Project : post-ProDaCTools

[kldiscom](#)

Kullback-Leibler divergence of components in data space

```
KLcoms = kldiscom(Mix, ndat)
```

KLcoms: matrix of values of KL divergences between pairs of components

Mix : inspected mixture estimator

ndat : the number of processed data

Designed: L. Tesar , based on kldiscom.c by P. Nedoma

Updated : LT Jan 2004, MK July 2004

Project : post-Prodactools

Calls : facdpred

[kldisdir](#)

Kulback-Leibler divergence of Dirichlet pdf

```
kld = kldisdir(V, V0)
```

kld : Kullback-Leibler distance

V, V0 : sufficient statistics of Dirichlet pdfs

Designed : M. Karny

Updated : May 2002, July 2004

Project : post-ProDaCTools

Source : Theory section \Markov_learning

kldist

Kullback-Leibler (KL) divergence of normal mixtures and factors (type = 1)

kld = kldist(fac, Mix, Mix0) divergence of a factor in mixtures

kld = kldist(0, Fac, Fac0) divergence of factors

[kldistc](#)

provides KL divergence of components in normal mixture

```
[distances,KLi,KLj] = kldistc(Mix)
  [distances,KLi] = kldistc(Mix)
    distances = kldistc(Mix)
```

distances: matrix of KL divergences between individual components
in parametric space

KLi, KLj : index

Mix : evaluated mixture

Design : P. Nedoma

Updated : May 2002, MK June 2002, July 2004

Project : ProDaCTools

See also : [kldist](#), [kldistcom](#)

ld2ld

convert L'DL decomposition described by a source structure strs
to another corresponding to a target structure strt

LD = ld2ld(LD, strs, strt)

LD = ld2ld(L,D) composes the factors L and D into the matrix LD

[L,D] = ld2ld(LD) decomposes the matrix LD into the factors L and D

LD : L'DL decomposition of a positive definite matrix
LD ... D placed instead of the unit diagonal of L

strs: source structure

strt: target structure

Designed : M. Karny, P. Nedoma

Updated : April 2000, MK July 2004

Project : post-ProDaCTool

ld2ls

```
convert L'DL decomposition of information matrix to LS representation
[Eth, Cth, cove] = ld2ls(LD, dfm, ny)
[Eth, Cth, cove] = ld2ls(LD,dfm)      ny=1
```

LD : lower triangular matrix L with unit diagonal replaced by diagonal D,
both from L'DL decomposition of and extended information matrix
dfm : degrees of freedom
ny : number of outputs
Eth : (ny,length(LD)-1)-matrix of estimates of regression coefficients
Cth : L'DL decomposition (in LD form) of covariance matrix of Eth
cove: L'DL decomposition (in LD form) of estimate of noise covariance

Designed: M. Karny

Updated : September 2001, MK July 2004

Project : post-ProDaCTools

ld2ud

```
convert L'DL decomposition, L lower triangle  
to      L'DL decomposition, L upper triangle  
use: [L, D] = lt2ut(LD)  
      LD      = lt2ut(LD)
```

Autor : J. Bohm, design P. Nedoma
Updated: March 2000
Project: ProDaCTools

ld2v

convert LD decomposition into extended information matrix
 $V = \text{lt2v}(\text{LD})$

$V = L'DL$

LD = L'DL decomposition of information matrix in packed form
i.e. with unit diagonal of the lower triangular matrix L
replaced by diagonal matrix D

Designed: P. Nedoma

Updated : April 2000, MK July 2004

Project : post-ProDaCTools

ldchg

change a part of L'DL decomposition

```
LD1 = ldchg(LD, str, LD1, str1)
```

LD1 : target LD decomposition (its structure corresponds to "str")

LD : source LD decomposition

str : structure of the source LD

LD1 : LD to be substituted into the original LD

str1 : structure of the substituted LD1

Design : P. Nedoma

Updated: July 2002, July 2004

Project: post-ProDaCTools

ldinv

inversion of L'DL decomposition, lower triangular representation
[L, D] = ldinv(LD)
LD = ldinv(LD)

L, D: inversion of the source original LDL' decomposition in the separated form
LD : inversion of the source original LDL' decomposition in the packed form
LD : source LD decomposition, triangular L matrix with unit diagonal
replaced by diagonal D

Author : P. Nedoma
Updated: May 2000, July 2004
Project: post-ProDaCTools
Calls : utinv,ud2ld,ld2ld

ldperm

LDPERM permute LD decomposition after permuting regressor entry

```
LD = ldperm(LD, i)
```

LD : packed permuted form of the L'DL decomposition of the extended information matrix

LD : packed form of the L'DL decomposition of the extended information matrix

i : the row number to be moved to 1st row

Author : L. Tesar, Jan 2004

Updated: Jan 2004, MK July 2004

Project: post-ProDaCTool

[ldupdt](#)

```
update LD decomposition by the weighted dyad  $w*r'*r$   
LD = ldupdt(LD,r,w)  
LD = ldupdt(LD,r)          w = 1  
[LD,ep] = ldupdt(LD,r)    call for r = data vector
```

LD: resulting $L'DL$ decomposition with diagonal D stored on diagonal L
ep: prediction error for r = data vector and $w = 1$ only
LD: source $L'DL$ to be modified
 r : row vector forming the dyad
 w : scalar weight, possibly negative until result is positive definite

Designed : January 2004 LT, using MK code partially
Last updated : March 2004
Project : ProDaCTool
See also : [dydrs](#)

ldupdtp

update L'DL weighted by w1 by the weighted dyad $w \cdot \text{Psi}' \cdot \text{Psi}$

[LD,hate,zeta1,Eth] = ldupdt1(LD,Psi,w)

[LD,hate,zeta1,Eth] = ldupdt1(LD,Psi) w = 1

LD : resulting L'DL decomposition with diagonal D stored on diagonal L

hate : prediction error

zeta1: LS gain = $1 + w \cdot \text{regression_vector}' \cdot \text{LS_covariance} \cdot \text{regression_vector}$

Eth : point estimate of parameters: only when NARGOUT=4

LD : source L'DL to be modified

Psi : row vector forming the dyad

w : scalar weight, possibly negative until result is positive definite

Designed : MK

Updated : November 2003

Project : ProDaCTool

Reference: overshoot2.tex

linplot

%%

loglik

```
compute v-likelihood or its increment
vll = loglik(LD,dfm,LD0,dfm0)    % increment of v-log likelihood
vll = loglik(LD,dfm)             % absolute value of v-log likelihood
vll = loglik(Mix, Mix0)

vll : v-likelihood for use vll = loglik(LD,dfm),
      otherwise increment of v-likelihood
LD   : L'DL decomposition of extended information matrix
dfm  : degrees of freedom    (\nu-2)
LD0  : L'DL decomposition of prior extended information matrix
dfm0 : prior degrees of freedom (\nu_{0}-1)
Mix  : posterior mixture
Mix0 : prior mixture

Design : M. Karny
Updated : April 2002
Project : ProDaCTools
```

ls2ld

convert least squares representation of information matrix
to L'DL decomposition of extended information matrix
use : `LD = ls2ld(Eth, Cth, cove, dfm)` packed version of L'DL
or `[L,D] = ls2ld(Eth, Cth, cove, dfm)` explicit version of L'DL

`Eth` : matrix of LS estimates of regression coefficients
`Cth` : covarinace matrix of `Eth` in L'DL form packed so that
 diagonal `D` replaced unit diagonal of lower triangular `L`
`cove` : LS estimate of noise covariance in the packed L'DL form
`dfm` : degrees of freedom

Author : M. Karny
Updated : September 2001
Project : ProDaCTools
Calls : `utinv`, `ud2ld`

ltdl

L'DL decomposition of positive definite symmetric matrix

use: `LD = ltdl(C)`

or : `[L, D] = ltdl(C)`

`C` : matrix to be decomposed

`LD` : $C=L'DL$ where `L` is lower triangular matrix with unit diagonal
and `D` is non-negative diagonal matrix; `LD` packed version
of `L` and `D`: units are replaced by `D`

Author : P. Nedoma

Updated : August 2001

Project : ProDaCTools

Calls : `ulchol`

[mix2arxc](#)

convert normal mixture into array of ARX components
Coms = mix2arxc(Mix)

Coms : array of components
Mix : the converted mixture

Design : P. Nedoma
Updated : July 2000, MK August 2001, July 2004
Project : post-ProDaCTools

Calls : arx2arx, fac2arxc

[mix2mix](#)

convert mixture to a specified form

```
Mix = mix2mix(Mix, form)
```

Mix : mixture of any form

form : coded form 21 22 23 or 24; (+100 for estimator)

Designed: P. Nedoma

Updated : June 2001, MK August 2001, July 2004

Project : post-ProDaCTools

Calls : arx2arx, com2com, mixconst

[mix2mix0](#)

```
create initial mixture dimensioned as a pattern
Mix0 = mix2mix0(Mix, dfcs)
Mix0 = mix2mix0(Mix)          dfcs are set to default values
```

Designed: P. Nedoma

Updated : July 2000, MK August 2001, August 2004

Project : post-ProDaCTools

Calls : defaults, facarx, facarxl, mixconst

[mix2mixm](#)

```
get data-marginal projector
pMix = mix2mixm(Mix, pchns)
pMix = mix2mixm(Mix)    predicted channels coincide with the modelled ones
```

```
pMix    : mixture predictor
Mix      : mixture of any form, preferably 22, 122
pchns    : predicted channels
```

```
Designed: P. Nedoma
Updated  : June 2001, August 2004
Project  : post-ProDaCTools
```

`mix2pro`

```
make/re-build mixture projector
pMix = mix2pro(Mix, pchns, cchns)
pMix = mix2pro(Mix, pchns)          no channels in condition
pMix = mix2pro(Mix)                all modelled channels predicted and
                                   no channels in conditions
```

```
pMix    : mixture predictor
Mix      : mixture or p-mixture of any form
pchns    : modelled channels
cchns    : channels in condition
```

```
Design   : P. Nedoma
Updated  : June 2001, MK August 2001, August 2004
Project  : ProDaCTools
Calls    : mix2mix, com2pro
```

`mixbrow`

GUI window for viewing of mixture parameters
`mixbrow(Mix)`

Mix : an arbitrary mixture

Designed: Vasek Smidl

Updated : 3 Sep 2001, MK August 2004

Project : post-ProDaCTools

`mixconst`

normal-mixture constructor

```
[Mix, maxtd] = mixconst(Facs,coms,dfcs)
```

```
Mix          = mixconst(Coms, dfcs)
```

Mix : mixture or p-mixture

maxtd : maximum time delay of factors

Facs : array of factors

Coms : array of components

coms : table components-by-factors

dfcs : degrees of freedom of components

Designed: P. Nedoma

Updated : June 2001, MK August 2001, 2004

Project : post-ProDaCTools

[mixcopy](#)

copies of ARX or ARX LS statistics, states and type preserved
`Mix1 = mixcopy(Mix,Mix0)`

`Mix1` : updated mixture
`Mix` : source of statistics
`Mix0` : mixture to be updated

Note: current limitation to 21, 22 and 122 mixtures

Design : P. Nedoma
Updated: April 2002, MK August 2004
Project: post-ProDaCTools

[mixcut](#)

```
cancel components with small dfcs  
Mix = mixcut(Mix, level)  
Mix = mixcut(Mix)          level = 0.001
```

```
Mix  : updated mixture  
Mix  : source mixture  
level: percentage of sum(Mix.dfcs)
```

Note : type 21, 22, 122 supported only

Design : P. Nedoma
Updated: October 2002, MK July 2002, August 2004
Project: post-ProDaCTools
See also: [comdel](#)

[mixdfms](#)

sum of dfm and dfm0 of mixture factors
[s, s0] = mixdfms(Mix)

s : sum of dfms
s0 : sum of dfms0
Mix : mixture

Designed: P. Nedoma
Updated : April 2001, MK August 2004
Project : post-ProDaCTools

`mixdump`

dump the mixture to the disk file

```
mixdump(Mix,filename)
```

Mix : dumped mixture

filename : the target file

Note:

machineformat: used machine format, fixed at machineformat=1

prec : adopted precision, fixed at prec=double

Designed : B. Kovar, M. Tichy

Updated : September 2000, MK August 2001, 2004

Project : post-ProDaCTools

Calls : dump

Problems : in "dump" prec='double' thus prec is not optional

[mixest](#)

iterative Bayesian ARX mixture estimation

```
LHS = mixest(Mix0, frg, ndat, niter, method)
LHS = mixest(Mix0, frg, ndat, niter)          method = quasi-Bayes (q)
LHS = mixest(Mix0, frg, ndat)                 niter  = 1 for q,f; 10 for b
LHS = mixest(Mix0, frg)                       ndat   = lenght(DATA)
LHS = mixest(Mix0)                            frg    = defaults
```

```
LHS      : Mix or [Mix, mixlls]
Mix      : estimated ARX mixture,           type = 21
Mix0     : initial estimate of ARX mixture of any type
mixlls   : trajectory of mixll, i.e. v-log-likelihood of the mixture
           (used for experiments only)
frg      : forgetting rate (alternative forgetting for method (f))
ndat     : size of data sample or data sample (copied to global DATA)
niter    : the number of iterations used in iterative methods
method   : character string, the 1st significant character means:
           'q' : iterative          quasi-Bayes mixture estimation
           'b' : iterative batch quasi-Bayes mixture estimation
           'f' : iterative mixture estimation with forgetting branching
           If the string 'method's' 2nd column is 's' : states are
               evaluated by mixstats
```

Designed: P. Nedoma

Updated : October 2001, MK June 2002, August 2004

Project : post-ProDaCTools

Note: DEBUG mechanism is exploited for inspecting intermediate results

DEBUG is set in Pro dini

See also: [mixestqb](#), [mixestq](#), [mixestbb](#), [mistats](#)

[mixestbb](#)

iterative mixture estimation by forgetting branching

```
function [Mix, mixlls] = mixestbb(Mix0, frg, ndat, niter)
```

```
    LHS = mixestbb(Mix0, frg, ndat, niter)
```

```
    LHS = mixestbb(Mix0, frg, ndat)    niter = 1
```

```
    LHS = mixestbb(Mix0, frg)          ndat  = length of global DATA
```

```
    LHS = mixestbb(Mix0)               frg    = defaults('frg')
```

LHS : Mix or [Mix, mixlls]

Mix : estimated mixture, type = 21

mixlls: trajectory of mixll, i.e. v-log-likelihood of the mixture

Mix0 : initial mixture

frg : a low forgetting rate

ndat : size of data sample or data sample (copied to global DATA)

niter : the number of iterations

Note : by setting global DEBUG>1: the initial and estimated
mixtures are displayed in each iteration

Designed: P. Nedoma

Updated : November 2001, MK June 2002, August 2004

Project : post-ProDaCTools

See also: [mixestbb1](#), [mixest](#), [mixestqb](#), [mixestim](#), [mixestbq](#), [mixestem](#)

[mixestbq](#)

iterative mixture estimate by batch quasi-Bayes estimation

function [Mix, mixlls] = mixestbq(Mix0, frg, ndat, niter)

LHS = mixestbq(Mix0, frg, ndat, niter)

LHS = mixestbq(Mix0, frg, ndat) niter = 10

LHS = mixestbq(Mix0, frg) ndat = size of global DATA

LHS = mixestbq(Mix0) frg = defaults('f')

LHS : Mix or [Mix, mixlls]

Mix : estimated mixture, type = 21

mixlls: trajectory of mixll, i.e. v-log-likelihood of mixture

Mix0 : initial mixture, any type

frg : forgetting rate

ndat : size of data sample or data sample (copied to global DATA)

niter : number of iterations

Note : by setting global DEBUG>1 the initial and estimated mixtures
are displayed in each iteration

Design : P. Nedoma

Updated : October 2001, MK June 2002, August 2004

Project : post-ProDaCTools

See also: [mixestqb](#), [mixest](#), [mixestim](#), [mixestbb](#)

[mixestem](#)

normal mixture estimation by EM algorithm

```
Mix = mixestem(Mix0, ndat, niter)
```

```
Mix = mixestem(Mix0, ndat)      niter = 10
```

```
Mix = mixestem(Mix0)          ndat  = length of global DATA
```

Mix : estimated mixture (point estimate is produced only), type = 21

Mix0 : initial mixture

ndat : size of data sample or data sample (copied to global DATA)

niter: number of iterations

Designed: P. Nedoma

Updated : January 2000, MK June 2002, August 2004

Project : post-ProDaCTools

See also: [mixestbq](#)

`mixestfe`

```
[Mix, Mixs, Mixs1] = mixestfe(Mix0, frg, ndat, Mixs, Mixs1);  
mixture estimate + filtration error for each factor  
called in covering function mixestee.m
```

```
Mix   : estimated mixture  
Mixs  : cell vector of mixtures, 2 components  
Mixs1: cell vector of mixtures, 1 component
```

```
Mix0 : initial mixture  
frg   : forgetting rate      (default: default forgetting rate)  
ndat  : size of data sample  (default: whole data sample)
```

```
Design : P. Nedoma  
Updated: May 2002  
Project: ProDaCTools
```

```
=====*/
```

`mixestim`

```
quasi-Bayes estimation of ARX mixture
Mix = mixestim(Mix0, frg, ndat, Mixa)
Mix = mixestim(Mix0, frg, ndat)      % stabilized forgetting is not used
Mix = mixestim(Mix0, frg)           % recursive estimation
Mix = mixestim(Mix0)                 % use default forgetting rate
```

```
Mix      : estimated mixture
Mix0     : initial mixture, any type
frg      : forgetting rate
ndat     : sample size
Mixa     : alternative mixture for stabilized forgetting
```

Designed: P. Nedoma

Updated : January 2003, MK August 2004

Project : post-ProDaCTools

`mixestimp`

```
projection-based estimation of ARX mixture
Mix = mixestimp(Mix0, frg, ndat, Mixa)
Mix = mixestimp(Mix0, frg, ndat)      % stabilized forgetting is not used
Mix = mixestimp(Mix0, frg)           % recursive estimation
Mix = mixestimp(Mix0)                % use default forgetting rate
```

```
Mix      : estimated mixture (type 21)
Mix0     : initial mixture   (any type)
frg      : forgetting rate
ndat     : sample size
Mixa     : alternative mixture for stabilized forgetting
```

Designed: P. Nedoma

Updated : January 2003, MK August 2004

Project : post-ProDaCTools

`mixestims`

```
quasi-Bayes estimation of ARX mixture with stopping
[Mix, tstop, Qs, mixlls] = mixestims(Mix0, frg, ndat, Mixa)
[Mix, tstop, Qs, mixlls] = mixestims(Mix0, frg, ndat) % stabilized forgetting is not used
[Mix, tstop, Qs, mixlls] = mixestims(Mix0, frg)        % recursive estimation
[Mix, tstop, Qs, mixlls] = mixestims(Mix0)            % use default forgetting rate
```

```
Mix      : estimated mixture (type 21)
tstop    : time moment when estimation was stopped
Qs       : value of the test statistics according to which the stopping is made
mixlls   : time trajectory of the mixture log-likelihood
Mix0     : initial mixture (any type)
frg      : forgetting rate
ndat     : sample size
Mixa     : alternative mixture for stabilized forgetting
```

Designed: P. Nedoma

Updated : January 2003, MK August 2004

Project : post-ProDaCTools

[mixestmt](#)

```
function Mix = mixestmt(Mix0, frg, ndat, niter)
quasi-Bayes iterative estimation with fixed variances
```

```
Mix = mixestmt(Mix0, frg, ndat, niter)
Mix = mixestmt(Mix0, frg, ndat)      niter = 1
Mix = mixestmt(Mix0, frg)           ndat  = length of global DATA
Mix = mixestmt(Mix0)                frg   = defaults('frg')
```

```
Mix    : estimated mixture, type = 21
Mix0   : initial mixture, any type
frg    : forgetting rate
ndat   : size of data sample or data sample (copied to global DATA)
niter  : number of iterations
```

Notes :

- without iterations equivalent to batch mixestim
- by setting global DEBUG>1: prior and posterior mixture are displayed in each iteration

Design : P. Nedoma

Updated : June 2002, MK August 2004

Project : post-ProDaCTools

See also: [mixestim](#), [mixest](#), [mixestbq](#), [mixestqb](#)

`mixestpb`

```
function [Mix, mixlls] = mixestpb(Mix0, frg, ndat, niter)
projection-based iterative estimation of normal mixture

    LHS = mixestpb(Mix0, frg, ndat, niter)
    LHS = mixestpb(Mix0, frg, ndat)      niter = 1
    LHS = mixestpb(Mix0, frg)           ndat  = length of global DATA
    LHS = mixestpb(Mix0)                frg    = defaults('frg')
```

LHS : Mix or [Mix, mixlls]
Mix : estimated mixture
mixlls: trajectory of mixll, i.e. log-likelihood of the mixture
Mix0 : initial mixture, any type
frg : forgetting rate
ndat : size of data sample or data sample (copied to global DATA)
niter : number of iterations

Note : by setting global DEBUG>1:
in each iteration, start and end mixture is displayed

Designed: P. Nedoma
Updated : October 2001, MK August 2001, 2004
Project : post-ProDaCTools

`mixestqb`

```
function [Mix, mixlls] = mixestqb(Mix0, frg, ndat, niter)
quasi-Bayes iterative estimation of normal mixture

    LHS = mixestqb(Mix0, frg, ndat, niter)
    LHS = mixestqb(Mix0, frg, ndat)      niter = 1
    LHS = mixestqb(Mix0, frg)           ndat  = length of global DATA
    LHS = mixestqb(Mix0)                frg    = defaults('frg')
```

LHS : Mix or [Mix, mixlls]
Mix : estimated mixture, type 21
Mix0 : initial mixture, any type
mixlls: trajectory of mixll, i.e. log-likelihood of the mixture
frg : forgetting rate
ndat : size of data sample or data sample (copied to global DATA)
niter : number of iterations

Note : by setting global DEBUG>1:
in each iteration, start and end mixture is displayed

Design : P. Nedoma
Updated : October 2001, MK August 2001, 2004
Project : post-ProDaCTools

`mixestqbs`

```
function [Mix, mixlls] = mixestqbs(Mix0, frg, ndat, niter, threshold)
quasi-Bayes iterative estimation of normal mixture with a stopping rule
```

```
    LHS = mixestqbs(Mix0, frg, ndat, niter, threshold)
    LHS = mixestqbs(Mix0, frg, ndat, niter, []) default threshold
    LHS = mixestqbs(Mix0, frg, ndat)      niter = 1
    LHS = mixestqbs(Mix0, frg)           ndat  = length of global DATA
    LHS = mixestqbs(Mix0)                 frg   = defaults('frg')
```

```
LHS   : Mix or [Mix, mixlls]
Mix    : estimated mixture, type 21
mixlls: trajectory of mixll, i.e. log-likelihood of the mixture
Mix0   : initial mixture, any type
frg    : forgetting rate
ndat   : size of data sample or data sample (copied to global DATA)
niter  : number of iterations
threshold: stopping threshold
```

```
Note   : by setting global DEBUG>1:
          in each iteration, start and end mixture is displayed
```

```
Designed: P. Nedoma
Updated  : October 2001, MK August 2001, 2004
Project  : post-ProDaCTools
```


[mixflat](#)

mixture flattening

```
Mix0 = mixflat(Mix)
```

```
Mix0 = mixflat(Mix, rate)
```

```
Mix0 = mixflat(Mix, rate, rate1)
```

```
Mix0 = mixflat(Mix, rate, rate1, MixA)
```

Mix : input estimated mixture

Mix0 : flattened mixture

rate : factor flattening rate

rate1 : component-weights flattening rate

MixA : alternative flattening mixture

Design : M. Karny, P. Nedoma

Updated : May 2001

Project : ProDaCTools

See also: [mixflatv](#)

[mixflatv](#)

mixture flattening with variable flattening rate

```
[Mix0, handle] = mixflat(Mix, niter, ndat)    initialization  
                                                (1st estimation step)  
[Mix0, handle] = mixflat(Mix, handle)       otherwise
```

```
Mix      : input estimated mixture  
Mix0     : output flattened mixture  
handle   : 1st step: No of iterations  
          otherwise run time states (structure)
```

```
Design   : P. Nedoma  
Updated  : November 2001  
Project  : ProDaCTools  
See also: mixflat
```

[mixfrg](#)

```
mixture forgetting
Mix = mixfrg(Mix, rate, rate1, Mixa)
Mix = mixfrg(Mix, rate, rate1)      flat "bar" alternative
Mix = mixfrg(Mix, rate)            rate1 = rate
```

```
Mix  : forgotten mixture
Mix  : updated mixture
rate : factor flattening rate
rate1: component-weights forgetting rate
Mixa : stabilezed forgetting
```

```
Design  : P. Nedoma
Updated : March 2003
Project : ProDaCTools remake
```

`mixgmean`

geometric mean of mixtures

```
Mix = mixgmean(Mix1, Mix2, [lambda1 lambda2])
```

```
Mix = mixgmean(Mix1, Mix2, lambda) lambda2 = 1-lambda
```

Mix1, Mix2: prior mixtures

lambda: weight(s)

sum can be <1 (for geometric mean of several mixtures)

Design : P. Nedoma

Updated : April 2003

Project : ProDaCTools remake

`mixgrid`

conversion of ARX mixture to pdf. coordinates

```
[x,y,z] = mixgrid(Mix, pchns, cchns, psi0, pre, n, r)
[x,y,z] = mixgrid(Mix, pchns, cchns, psi0, pre, n) r    = []
[x,y,z] = mixgrid(Mix, pchns, cchns, psi0, pre)      n    = []
[x,y,z] = mixgrid(Mix, pchns, cchns, psi0)           no scaling
[x,y,z] = mixgrid(Mix, pchns, cchns)                 psi0 = []
[x,y,z] = mixgrid(Mix, pchns)                        cchns = []
[x,y,z] = mixgrid(Mix)                              pchns = modelled channels
```

Mix : mixture, any form
pchns : predicted channels
cchns : channels in condition
psi0 : values of channels in condition
 if empty, the values are extracted from DATA
pre : preprocessing list (scaling)
n : grid density (densities)
 if empty: 100 for 1 dim, [50,50] for 2 dimensional case
r : grid range (ranges)
 if empty: automatic value computed from means and noise covariances

Design : P. Nedoma
Updated : March 2002
Project : ProDaCTools

[mixinit](#)

NEW mixinit

initialization of mixture estimation

```
[Mix, Mix0] = mixinit(Mix0, frg, ndat, niter, options, belief)
[Mix, Mix0] = mixinit(Mix0, frg, ndat, niter, options) belief is not used
[Mix, Mix0] = mixinit(Mix0, frg, ndat, niter) defaults are used
[Mix, Mix0] = mixinit(Mix0, frg, ndat) niter=5
[Mix, Mix0] = mixinit(Mix0, frg) size(DATA, 2)
[Mix, Mix0] = mixinit(Mix0) default forgetting rate
```

Mix : estimated mixture, type 21

Mix0 : initial mixture, any type internally converted to 21

frg : forgetting rate

ndat : size of data sample

it can be cell vector for buffered data processing

niter : number of iterations

options: initialization options

character string or character string followed an number

belief : cell vector of beliefs for individual channels

--- options for estimation ---

q : iterative quasi-Bayes mixture estimation

b : iterative batch quasi-Bayes mixture estimation

f : iterative mixture estimation based on forgetting branching

m : iterative quasi-Bayes with fixed variances

n : number of iterations for iterative estimation, a number follows;
default is 10 iterations

Note: if iterative estimation is not explicitly specified,
mixestim is used

--- option for structure estimation ---

h : number of runs for structure estimation (integer follows)

--- options that modify processing ---

c : do not make the final housekeeping, default: make it

g : number of initial steps when all factors are split default: 2

k : number of steps when components are not merged or erased, default: 1

Design : M. Karny, P. Nedoma

Updated: September 2002

Project: ProDaCTools

See also: [commerge](#), [mixerase](#), [mixest](#)

[mixmaxtd](#)

```
get maximum time delay in mixture  
maxtd = mixmaxtd(Mix)
```

Design : P. Nedoma
Updated: January 2003
Project: ProDaCTools

[mixmesh](#)

```
mixture contour plot
mixmesh(Mix, pchns, cchns, psi0, pre, n, r)
mixmesh(Mix, pchns, cchns, psi0, pre, n) r      = []
mixmesh(Mix, pchns, cchns, psi0, pre)      n      = []
mixmesh(Mix, pchns, cchns, psi0)           no scaling
mixmesh(Mix, pchns, cchns)                 psi0 = [] (values from DATA)
mixmesh(Mix, pchns)                       cchns = [] (marginal pdf)
mixmesh(Mix)                             pchns = modelled channels[1 2] or 1
```

```
Mix      : mixture, any form
pchns    : predicted channels
cchns    : channels in condition
psi0     : values of channels in condition
           if empty, the values are extracted from DATA
pre      : preprocessing list (scaling)
n        : grid density (densities)
           if empty: 100 for 1 dim, [50,50] for 2 dimensional case
r        : grid range (ranges)
           if empty: automatic value computed from means and noise covariances
```

```
Design  : P. Nedoma
Updated : March 2002
Project : ProDaCTools
```


[mixplot](#)

```
mixture contour plot
mixplot(Mix, pchns, cchns, psi0, pre, n, r)
mixplot(Mix, pchns, cchns, psi0, pre, n) r      = []
mixplot(Mix, pchns, cchns, psi0, pre)      n      = []
mixplot(Mix, pchns, cchns, psi0)            no scaling
mixplot(Mix, pchns, cchns)                  psi0 = [] (values from DATA)
mixplot(Mix, pchns)                        cchns = [] (marginal pdf)
mixplot(Mix)                              pchns = modelled channels[1 2] or 1
```

```
Mix      : mixture, any form
pchns    : predicted channels
cchns    : channels in condition
psi0     : values of channels in condition
           if empty, the values are extracted from DATA
pre      : preprocessing list (scaling)
n        : grid density (densities)
           if empty: 100 for 1 dim, [50,50] for 2 dimensional case
r        : grid range (ranges)
           if empty: automatic value computed from means and noise covariances
```

```
Design   : P. Nedoma
Updated  : March 2002
Project  : ProDaCTools
```

`mixplotc`

```
mixture contour plot in 2 subplots
mixplot(Mix, pchns, cchns, psi0, pre, n, r)
mixplot(Mix, pchns, cchns, psi0, pre, n) r    = []
mixplot(Mix, pchns, cchns, psi0, pre)    n    = []
mixplot(Mix, pchns, cchns, psi0)          no scaling
mixplot(Mix, pchns, cchns)                psi0 = [] (values from DATA)
mixplot(Mix, pchns)                       cchns = [] (marginal pdf)
mixplot(Mix)                             pchns = modelled channels[1 2] or 1
```

```
Mix      : mixture, any form
pchns    : predicted channels
cchns    : channels in condition
psi0     : values of channels in condition
           if empty, the values are extracted from DATA
pre      : preprocessing list (scaling)
n        : grid density (densities)
           if empty: 100 for 1 dim, [50,50] for 2 dimensional case
r        : grid range (ranges)
           if empty: automatic value computed from means and noise covariances
```

```
Design  : P. Nedoma
Updated : March 2002
Project : ProDaCTools
```

`mixpro`

```
get mixture projection (prediction on conditioned marginal pdf)
LHS = mixpro(Mix, pchns, cchns, psi0, pre)
LHS = mixpro(Mix, pchns, cchns, psi0) no scalling
LHS = mixpro(Mix, pchns, cchns)          psi0 = []
LHS = mixpro(Mix, pchns)                  cchns = []
LHS = mixpro(Mix)                         pchns = modelled channels
LHS = pMix : mixture prediction or LHS = [Eths, coves, dfcs]
        with mixture prediction sum dfcs * N(Eths,coves)
```

```
Mix    : mixture or p-mixture, any form
pchns  : modelled channels
cchns  : channels in condition
psi0   : values of channels in condition
pre    : preprocessing list (scaling)
```

```
Design : P. Nedoma
Updated : June 2001
Project : ProDaCTools
```

[mixreorg](#)

reorganize mixture estimator

```
Mix1 = mixreorg(Mix, chns)
```

Mix1 : reorganized mixture

chns : new order of channels in components

Design : P. Nedoma

Updated: January 2003

Project: ProDaCTools continued

[mixscan](#)

```
scan mixture for 2 dim. clusters
mixscan(Mix,chns,pre)
mixscan(Mix,chns)           no data scaling
mixscan(Mix)                all channels
```

Design : P. Nedoma
Updated: July 2002
Project: ProDaCTools

[mixshow](#)

```
function plots mixture pMix;  
1D supported
```

[mixsimul](#)

```
mixture simulation
use: mixsimul(Sim)      - recursive
    mixsimul(Sim,ndat)  - batch
```

```
Sim:  mixture simulator
ndat: size of data sample
```

```
Autor  : P. Nedoma
Updated: April 2000
Project: ProDaCTools
```

`mixstats`

```
compute estimation statistics for a fixed mixture  
Mix = mixstats(Mix, ndat)      batch  
Mix = mixstats(Mix)           recursive
```

Design : P. Nedoma

Updated: March 2003

Project: ProDaCTools remake

`mixstrid`

```
estimate mixture structure
[Mix, Mix0, flag] = mixstrid(Mix, Mix0, belief, nruns)

Mix    : estimated mixture
Mix0   : initial mixture
flag   : 0 if not changed, otherwise 1
belief: cell vector of believes for individual channels
nruns  : number of runs for factor structure estimation

Design : P. Nedoma
Updated: April 2002
Project: ProDaCTools
```

`mixupdt`

```
one step of mixture update  
Mix = mixupdt(Mix0, flag, weight)  
Mix = mixupdt(Mix0, flag)    weight=1;  
Mix = mixupdt(Mix0)          flag = 0, i.e. physical updating
```

```
Mix      : estimated mixture  
Mix0     : initial mixture  
flag     : 1 only trial step  
          0 physical updating  
w        : weight of the data vector used, weight of the time
```

Designed: P. Nedoma

Updated : March 2003, MK August 2004

Project : post-ProDaCTools

`mixupdtF`

one step of ARX mixture update for projection based estimation

```
Mix = mixupdt(Mix0, flag)
```

```
Mix = mixupdt(Mix0)           % flag = 0, i.e. physical update
```

Mix : estimated mixture, type 21

Mix0 : initial mixture, type 21

flag : 1 only trial step
 : 0 physical update

Designed: P. Nedoma

Updated : March 2003, MK August

Project : post-ProDaCTools

[mversion](#)

No help comments found in mversion.m.

noise

random number generator scaled to mean=0, variance=1 (except Cauchy)

rn = noise(m,n,type)

m,n : dimensions of the generated matrix sample

type : 1 gauss

2 uniform

3 lognormal

4 Cauchy

rn : generated sample

Author : P. Nedoma

Updated: August 2001

Project: ProDaCTools

realized as MEX function

perm

PERM swaps information matrix in LD decomposition

```
[LD] = perm(LD,i);
```

LD ... LD decomposition of information matrix
i ... index of line to be swapped with the next one

Description:

$$\text{Lout}' * \text{diag}(\text{dout}) * \text{Lout} = P(i,i+1) * L' * \text{diag}(d) * L * P(i,i+1);$$

Where permutation matrix $P(i,j)$ permutes columns if applied from the right and line if applied from the left.

Design : L. Tesar, Feb 2003

Updated : Jan 2004

Project : post-ProDaCTool

Reference: dydrs

preaux

auxiliary function used by preproc.m
designed to be equal wit the dll version

preaux1

PREAUX1 Auxiliary function for data pre-processing. Should be run through preproc and prest

```
Preout = preaux1(method_nr, time, Pre);
```

method_nr method number

time time array

Pre data pre-processing structure Pre. Preout is the output.

preinit

initialization of preprocessing

```
pre = preinit(pre)
```

pre : input preprocessing requirements
output run-time preprocessing list

pre is the celllist of the type

{'algorithm' parameters}

parameters is a cell list of of individual parameters or
numerical matrix

Available pre-processing algorithms:

boundary	signal limits
olymean	mean outlier removal filter (window based)
olymedian	median outlier removal filter (window based)
olymeanf	mean outlier removal filter (forgetting based)
olymedianf	median outlier removal filter (forgetting based)
mean	simple mean filter (window based)
median	simple median filter (window based)
meanf	simple mean filter (forgetting based)
medianf	simple median filter (forgetting based)
outliers	remove outliers
smooth	filter for smoothing

General parameters:

c	list of channels
startup_period	is a scalar to adjust the initial period of algorithm where the DATA matrix is not modified at all, to allow clean startup of

Specific parameters:

boundary	lower upper
olymean	window_size level m0 s0
olymedian	window_size level m0 s0
olymeanf	forgetting_constant level m0 s0
olymedianf	forgetting_constant level m0 s0 importance_threshold
mean	window_size m0 s0
median	window_size m0 s0
meanf	forgetting_constant m0 s0
medianf	forgetting_constant m0 s0 importance_threshold

Design : L. Tesar, P. Nedoma
Updated: January 2002
Project: ProDaCTools

preproc

=====

prestep

PRESTEP, data pre-processing run-time function

```
function [out]=prestep(in,nt);
```

```
function [out]=prestep(in);
```

Data pre-processing is done on the global data matrix DATA. If nt parameter is omitted, the time parameter is taken from the global variable TIME. Otherwise data are pre-processed from 1 to nt.

in ... cellvector describing the data pre-processing to be done
out ... cellvector, that is to be used in the next call of the
function preproc. It includes run-time information, and
status.

prior

prior knowledge processing
Facs = priproc(Facs0, pri)

Facs : fictitious ARX factor or cell vector of factors
Facs0 : flat ARX factor or cell vector of factors
pri : prior knowledge list

Design : P. Nedoma
Updated: February 2003, February 2003 MK
Project: DESIGNER
Call : priproc

pristr

```
structure estimation with prior knowledge
lhs = pristr(Facs0, pri, beliefs, nbest, nrep)
lhs = pristr(Facs0, pri, beliefs, nbest) nrep = 50
lhs = pristr(Facs0, pri, beliefs) nbest = 30
lhs = pristr(Facs0, pri) belief is not used

lhs      : Facs | [Facs, vlls]
Facs     : resulting estimated factor or cell vector
           of factors
vlls     : cell vector of information or cell vector of the information
           for individual outputs
vll{1}   - vlls nested prior
vll{2}   - vlls with prior
vll{3}   - "best" structures in cell array
vll{4}   - stuctures in block

Facs0    : richest initial factor or cell vector of factors
pri       : prior knowledge list (can be empty)
beliefs  : belief of cell vector of belifs or empty matrix
nbest    : number of "best" regressor maintained
nrep     : number of repetitions of search in space of regressors

Design   : M. Karny, P. Nedoma
Updated  : July 2004
Project: GA CR 102/03/0049, AV CR S1075351, S1075102, DESIGNER
+ maxfac +priestim + primerge
```

pristrd

display results of structure estimation
with prior knowledge
pristrd(Fac, Fac0, vll, sub)

Fac : estimated factor
Fac0: reachest factor
vll : result of pristr
sub : display weights and weights with prior
110 for full display

Design : P. Nedoma
Updated: April 2003
Project: ProDaCTools remake

pro2pre

convert predictor to prediction component

Facs : ARX LS component (array of ARX LS factors)

comaux: { matstr, pcove, ih, ic, pchns, cchns, itd0 }
build by com2pro

psi0 : values of channels in condition and zero-delayed not modelled channels

Design : P. Nedoma

Updated: June 2003

Project: ProDaCTools remake

pro2str

add additional information about pointers to external structure
pMix = pro2str(pMix, str)

pMix: mixture projector
str : structure to be used for prediction

Design : P. Nedoma
Updated: November 2001
Project: ProDaCTools

prodini

```
standard start of Mixtools session that
    cleans everything
    defines global variables needed in management
    specifies debugging level
    fixes seeds of random generators
use Proдини
```

```
Designed : P. Nedoma
Updated  : August 2001
Project  : Prodictools
```

profix

get projection from projector

```
OUTPUTS = profix(pMix, psi0, pre)
```

```
OUTPUTS = profix(pMix, psi0)      pre =[] (data are not normalized)
```

OUTPUTS

```
pMix1          or [pMix1, scales] or  
[EthS, coves, dfcs] or [EthS, coves, dfcs, scales]
```

pMix : mixture predictor

psi0 : values of channels in condition and zero-delayed not modelled channels

pre : pre-processing list if data are scaled

OUTPUTS

pMix1 : mixture prediction

EthS : cell vector or vector of means of the mixture component

coves : cell vector or vector of noise covariances of the mixture component

dfcs : transformed mixture dfcs respecting conditioning

scales: approximate data dependent component weights

for components -----

```
Com1 = profix(Com, comaux, psi0)
```

```
(internal function pro2pre is called)
```

```
Com : predictor component
```

```
Com1 : prediction component
```

```
comaux: states related to components (build by com2pro)
```

```
psi0 : values of channels in condition and zero-delayed not modelled
```

Design : P. Nedoma

Updated: May 2002

Project: ProDaCTools

Calls : pro2pre (internal), getflds

Updated : August 2001, MK comments added and problems searched for

Problems: solved by PN September 1001

remains: pro2pre should be complemented by comments

profixn

```
prediction n-steps ahead
OUTPUTS = profixna(pMix, psi0, pre, nstep)
OUTPUTS = profixna(pMix, psi0, pre) nstep==1
OUTPUTS = profixna(pMix, psi0)      no scaling
OUTPUTS = profixna(pMix)           psi0 from DATA
```

```
OUTPUTS: --- data dependent case ---
[Eths, coves, scales, weights] | [pMix1, weights]
      --- normal case ---
[Eths, coves, scales] | pMix1
```

Design : P. Nedoma
Updated: October 2002
Project: ProDaCTools

profixn1

```
prediction n-steps ahead
OUTPUTS = profixn(pMix, psi0, pre, nstep)
OUTPUTS = profixn(pMix, psi0, pre) nstep==1
OUTPUTS = profixn(pMix, psi0)      no scaling
OUTPUTS = profixn(pMix)            psi0 from DATA
```

```
OUTPUTS: --- data dependent case ---
[Eths, coves, dfcs, scales] | [pMix1, scales]
      --- normal case ---
[Eths, coves, dfcs] | pMix1
```

Design : P. Nedoma
Updated: October 2002
Project: ProDaCTools

protest

check projection arguments
auxiliary undocumented function
protest(Mix, pchns, cchns)

pchns : predicted channels
chns : channels in condition

Design : P. Nedoma
Updated: October 2001
Project: ProDaCTools

prt

debugging prints

use : prt(mes , X)

prt(mes)

mes : message

X : cell vector, structure matrix, LS normal mixture, normal factor

Design : P. Nedoma

Updated: August 2001

Project: ProDaCTools

prtstr

print results of facstrid

[psi](#)

evaluates psi function, i.e. 1st derivative of `gammaln`
z = positive argument
psi = value of the function

Design : M. Karny
Updated : May 2000
Project : Prodatools
See also : [kldistc](#), [kldistcom](#)

randnm

```
generate sample from Normal distribution  
r= randnm
```

```
r      : generated sample
```

```
%
```

```
Design : J. Andrysek
```

```
Project : BadDyr
```

```
Comments: used to generate exactly the same numbers in .m and .dll  
versions
```

randun

generate sample from Uniform distribution
r= randun

r : generated sample
SEED : global variable SEED

Design : J. Andrysek

Project : BadDyr

Comments: used to generate exactly the same numbers in .m and .dll
versions

relep

```
[se, yp] = relep(Mix, ndat, use_wgs)
```

se: relative prediction error

yp: prediction trajectory

Mix: estimated mixture or mixture predictor

ndat: portion of the data sample to be processed

use_wgs: use additional components weights

Design: P. Nedoma

Project: GA CR 102/03/0049, AV CR S1075351, S1075102, DESIGNER

relepn

```
[se, yp, dd] = relepn(Mix, nstep)
Mix:  estimated mixture
nstep: number of prediction steps
se:   relative prediction error
yp:   prediction
dd:   shifted data
```

Design: P. Nedoma

Project: GA CR 102/03/0049, AV CR S1075351, S1075102, DESIGNER

resizefig

SETFIG Set a figure size and position relatively to the screen size

```
h=resizefig(s,x,y,h)
h=resizefig(s,x,y)    h=gcf
h=resizefig(s,x)      y=0
h=resizefig(s)        x=0
```

s : new size of the figure given as percentage of the screen size (0-100)
x : new x-coordinate of the left bottom corner of the figure given as
 percentage of the screen size (0-100)
y : new y-coordinate of the left bottom corner of the figure given as
 percentage of the screen size (0-100)
h : handle of the figure, which size should be changed

Design : J.Andrysek

Project : The BOOK

restore

```
Xs = mixrest(filename)                                machineformat='l'
```

```
filename      : file where the object is stored  
machineformat: see MATLAB fopen  
prec          : see MATLAB fopen  
Xt            : object of the same type as the restored one  
                used to restore field names  
                not required for Mixtools structures
```

```
Desing  : P. Nedoma  
Updated : September 2001  
Project : ProDaCTools
```

ricexp

```
disp('exp')  
keyboard
```


ricpen

noise contribution of channels with o-innovations

ricpenu

noise contribution of channels with o-innovations

ricshift

```
ric = [Rici(nychn+1:nPsi,nychn+1:nPsi) zeros(npsi,nychn); zeros(nychn,nPsi)];
```

scalepri

scale prior knowledge list

ychn - first channel

setaxis

No help comments found in setaxis.m.

setdbg

```
set in dialog "dbstop" in a function  
setdbg('function')
```

The function is checked for presence of the comment
%> any comment
at the line beginning (line number is "line_number").
The comment(s) is (are) displayed and one selected in dialog.
Then the command
"dbstop in function at line_number"
is evaluated.
No action is done when only Enter is pressed.

Author : P. Nedoma
Updated: December 2001
Project: ProDaCTools

setfig

setting plot windows

use: setfig(x)

x - number of windows, 1-2

autor: I. Nagy

setting position:

(x,y) left corner

Position: [x y width height]

sigscan

scan signal

[sim2pdf](#)

convert mixture predictor to estimator close to data pdf (edits Cth,dfm etc.)

```
Mix = sim2mix(Sim, ndat)
```

Mix : complete mixture, type = 22

Sim : predictor mixture

ndat : number of data items

Design : P. Nedoma

Updated : June 2000

Project : ProDaCTools

See also: [mixestem](#)

simeval

repeated simulations

```
[res, tstop] = simeval(Sim, chns, nrep, ndat, threshold)
[res, tstop] = simeval(Sim, chns, nrep, ndat) threshold = 0.01
[res, tstop] = simeval(Sim, chns, nrep) size(DATA,2)
[res, tstop] = simeval(Sim, chns) 100
[res, tstop] = simeval(Sim) all modelled channels
```

```
res    : processing results: cell vector for each channel of chns
        res{chn}.stats - confidence interval for range and increments
        res{chn}.tra   - trajectories of individual simulation runs
tstop  : time where trajectories are stopped
Sim    : simulator | task
chns   : list of channels to be evaluated
nrep   : maximum number of repetitions
ndat   : maximum length of trajectories
threshold : threshold value, default = 0.01;
```

soptim

soptim performs simultaneous advisory design for normal mixture

```
[aMix] = soptim(aMix, aMixu, ufc, nstep, chis)
[aMix] = soptim(aMix, aMixu, ufc, nstep) chis = 1
[aMix] = soptim(aMix, aMixu, ufc)          nstep = [200, 1]
```

aMix : advised mixture of the type ARX LS enriched on following control states:

strc : common control structure

ufc : normalised vector qualifying components:

dangerous component (0), not dangerous (positive number)

kc : lift of quadratic forms

UDc : cell vector of u'du decompositions of KLD kernels

udca : u'du decomposition of average KLD kernel in UDc

kca : average lift of quadratic forms kc

aMixu : desired mixture (user's target) of the type ARX LS with control states

ufc : vector qualifying components: 0 - dangerous component, (1) - not

nstep : parameters [ns1,per] determining design horizon, i.e. horizon = ns1*per;

ns1 : number of block repetition

per : horizon of a block

if nstep is defined by parameter ns1 only then per is set to 1

chis : indicates strategy chosen: chis=1 for receding horizon (default) and chis=-1 for

Design : J. Bohm

Updated : June, 2002

Project : ProDaCTools, IST-1999-12058

See also : [udupdt](#), [getdvect](#), [facchng](#), [facarxls](#)

statgrid

```
[x,y,z] = statgrid(Eths, coves, alphas, n, r)
[x,y,z] = statgrid(pMix, n, r)
```

x,y,z : coordinates
Eths : vector or cell-vector of means
coves : vector or cell vector of noise variances
alphas : normalized degrees of freedom of components
n : grid density - number of points (or densities)
r : grid range (or ranges)
pMix : mixture projection

Design : P. Nedoma
Updated: June 2001
Project: ProDaCTools

statmesh

interactive 2D static mixture or data display

```
[x,y,z] = statmesh(Mix)
```

```
[x,y,z] = statmesh(chns)
```

x,y,z : coordinates

Mix : mixture or p-mixture, any form

chns : channels to be displayed

Design : P. Nedoma

Updated: June 2001

Project: ProDaCTools

statplot

plot centers and 2-sigma borders of components of a mixture

statplot(Mix, arg)

Mix : mixture, any form

arg : character string for plot (e.g. 'r--');

1 - standard

2 - standard, instead of 'o' in center, component No is displayed

missing: standard, before plot: clf; hold off

plot ends by hold on

Design : P. Nedoma

Updated: September 2001

Project: ProDaCTools

statsim

[illegible]

stopstac

stopping of a time series at stationary mode

Fac : updated factor

Q : value of relevant statistics

Design : P. Nedoma

Updated: October 2003

Project: GA CR 102/03/0049, AV CR S1075351, S1075102, DESIGNER

reference= stationarity.tex

straux1

STRAUX1 Advanced structure estimation based on LD decomposition

This m/mex file is internally called by facstr, IT IS NOT TO BE CALLED BY USER!! Documentation guiven for reference.

```
[strout, Lout, dout, nuout] = straux1(L, d, nu, L0, d0, nu0, belief, nbest, max_nrep, lambda)
```

L : Actual LD decomposition based on data
d : Actual LD decomposition based on data
nu : Actual data amount
L0 : prior information
d0 : prior information
nu0 : prior data amount
belief: user's belief on maximum structure items
 (1 items must be present, 2 items are probably present
 4 items must not be present, 3 items are probably not present)
 2 and 3 is the same
nbest : how many "best" regressors are maintained
strout : structure estimated (of the regressor, richest is 2:length(d))
Lout : output lower triangular matrix
dout : output diagonal vector of diagonal matrix D
nuout : posterior amount of data
max_nrep : maximal number of random starts in search for the best
 structure
lambda : stooping rule threshold
order_k : order of k

Design : L. Tesar
Updated : Feb-Apr 2003
Project : post-ProDaCTool
References: (only local inline functions)

Todo: in add_new, we need to implement structure comparison, instead of
loglikelihood comparison: ~any(logliks == new.loglik)

streq

are two structures equal ?

is = streq(str1, str2)

str1, str2: structures to be compared

is: 0 if not equal

1 if identical

-1 if equal but in different ordering

Design : P. Nedoma

Updated: May 2002

Project: ProDaCTools

[strmax](#)

auxiliary function used in mixture structure estimation

Design : L. Berec

Updated : 1998

Project : ProDaCTool

See also: [facstrid](#)

[student](#)

No help comments found in student.m.

`synmixi`

```
aMix = synmixi(Mix,uchn,strt)
aMix = synmixi(Mix,uchn), strt = []
aMix = synmixi(Mix), uchn = [], strt = []
```

`synmixi` transforms mixture estimate `Mix` to the control form `aMix` needed for design of advises

```
aMix    : advised mixture of the type ARX LS + following control states:
          strt : common control structure
          ufc  : normalised vector qualifying components:
                  dangerous component (0), not dangerous (positive number)
          kc   : lift of quadratic forms
          UDc  : cell vector of u'du decompositions of KLD kernels
          udca : u'du decomposition of average KLD kernel in UDc
          kca  : average lift of quadratic forms kc
Mix      : mixture estimate, any form
uchn     : list of channels with recognisable actions
strt     : common control structure
```

Design : J. Bohm
Updated: 06.03.02
Project: ProDaCTools

calls : `mix2mix`, `mix2pro`, `pro2str`
problems:

[target](#)

TARGET creates desired mixture (user target) and build list of modelled channels in component in the form [o-innovations, surplus p+, recognisable actions]. Channels inside of each group is ordered according to ascending priority set by the user

```
[Mixu, Chns, ychns] = target(Chns)
```

Mixu : user target
ychns : list of modelled channels in component
Chns : channels descriptions

Design : T.V.Guy
Updated : September 2002
Project : ProDaCTools
See also : [chnget](#), [chnset](#), [aloptim](#), [inisynd](#), [soptim](#)

tukinit

obtain number of data samples in the input data of estimation algorithms,
set gloal variable DATA if need
ndat=tukinit(Ndat)

Ndat : scalar containing number of data samples to be processed or a
pair containing data filename and rowcount of data stored in
the file

ndat : scalar containing number of data samples to be processed

Design : J. Andrysek

Created: September 2004

Project: BadDyr

Note : C-version of tukinit is used to process the data contained in the file
filename in parts. This m-version was made for compatibility purposes
only. It loads the whole data file in memory at once.

ud2ld

convert U'DU decomposition to its equivalent L'DL decomposition

```
[L, D] = ud2ld(UD)
```

```
LD      = ud2ld(UD)
```

UD = upper triangular matrix with unit diagonal replaced by diagonal % D

L = lower triangular matrix with unit diagonal

D = diagonal matrix

LD = lower triangular matrix with unit diagonal replaced by diagonal
D

Design : P. Nedoma

Updated: November 2000

Project: ProDaCTools

udform

UDFORM restores matrix factorized (canonical) form of ARX LS component

```
[Eth1, cove1] = udform(Eth, cove)
```

Eth, cove - input relevant part of the component

Eth1, cove1 - output component

Design : L. Tesar, Jan 2004

Updated: January 2004

Project: post-ProDaCTool

Note: to exchange 2 rows of the component, change
the columns of the input component and call this function

udin

inversion of $U'DU$ decomposition

$UD = \text{Udin}(UD)$

$[U, D] = \text{udin}(UD)$

UD = upper triangular matrix U with unit diagonal replaced by
the diagonal D

UD = the resulting inversion organised as inputs

U, D = the resulting inversion with full form of U and D

Autor : P. Nedoma

Updated: November 2000

Project: ProDaCTools

udupdt

```
update U'DU decomposition update
U'DU := U'DU+w*r'*r
UD      = udupdt(UD, r, w)
UD      = input upper triangular matrix with unit diagonal replaced
          by D
w       = weight of the updating dyad
r       = the modifying data vector
```

Design : L. Tesar Jan 2004

Updated: Jan 2004

Project: ProDaCTool

Calls : dydrs

[ufcgen](#)

```
Mixc = stedopt(Mixc,Mixc0)
```

stedopt computes losses of the components for the
given criterion (desired mixture Mixc0)

```
Mixc   : mixture advised of the type ARX LS + following control states:
        strc : common control structure
        ufc  : normalised vector qualifying components:
                dangerous component (0), not dangerous (positive number)
        kc   : lift of quadratic forms
        UDc  : cell vector of u'du decompositions of KLD kernels
        udca : u'du decomposition of average KLD kernel in UDc
        kca  : average lift of quadratic forms kc
Mixc0  : mixture desired type ARX LS + control states
```

utdu

utdu = decompose positive definite g into u'du decomposition and store it into g
use : g = utdu(g)

Autor : J. Bohm

Updated: April 2000

Project: ProDaCTools

utinv

```
utrinv = Upper Triangle matrix INVersion  
use : b = utinv(a)  
    a : inverted upper triangular matrix  
    b : resulting inversion
```

```
Autor  : I. Nagy  
Updated: April 2000  
Project: ProDaCTools
```

TIME processing time

DATA data sample

`ndat` length of data

[Ndat](#) specification for buffered processing

psi regressor vector

Psi data vector

`npsi` length of regression vector

n_{Ψ} length of data vector

`str` structure of regression vector

Fac factor

Facs array of factors

`fac` position of a factor in an array of factors

ychn modeled channel

dfm degrees of freedom of a factor

[LD](#) L'DL decomposition of the extended information matrix

L triangular part of L'DL decomposition

D diagonal part of L'DL decomposition of extended information matrix

Eth point estimate of regression coefficients

Cth covariance of regression coefficients

`cove` point estimate of noise covariance

com component

`coms` array of components

`dfcs` vector of degrees of freedom of components

`dfcs0` initial degrees of freedom of components

α normalized vector of degrees of freedom of components

Com matrix ARX or ARX LS component

Coms array of matrix ARX or ARX LS components

[ychns](#) modeled channels in component

`nychn` number of modeled channels

Mix mixture estimate

[Sim](#) mixture simulator

pMix mixture predictor

[pMixfix](#) mixture prediction

[facs](#) list of factors

`nfac` number of active factors

`ncom` number of components

`nchn` number of modeled channels

frg forgetting rate

`frgd` default forgetting rate

rate mixture flattening rate

`maxtd` maximum time delay of factors in a mixture

`nruns` number of runs in iterative mixture estimation

`relerr` relative error

`maxerr` maximum possible error

[pchns](#) predicted channels

[echns](#) channels in condition

ψ_0 value of zero-delayed regressor

[nsk](#) extent of data grouping

[faclls](#) virtual factor predictions

[comlls](#) component predictions

[mixl](#) posterior data likelihood (mixture prediction)

`comwgs` component weights

facwgs factor weights

`maxstr` guess of the richest structure

maxFac richest factor

maxMix richest mixture

[belief](#) belief on a guess of richest structure

[chbelief](#) belief on factors of a channel

`nbest` number of "best" MAP structures stored

`nrep` number of random starts

irep iteration

MAPstr MAP estimate of the factor structure

[pre](#) preprocessing requirements

aMixc advised mixture of the type ARX LS + control states

[aMixer](#) desired mixture of the type ARX LS + control states

`strc` common control structure

ufc normalised vector qualifying components

[kc](#) lift of quadratic forms

UDc cell vector of u'du decompositions of KLD kernels

[udca](#) u'du decomposition of average KLD kernel in UDc

[kca](#) average lift of quadratic forms kc

[uchn](#) list of channels with recognisable actions

[poch](#) list of channels with o-innovations

[outs](#) list of channels with innovations

`npochn` number of channels with o-innovations

[chis](#) strategy of control design

DEBUG global debugging flag

PLOTNO control results of iterative computation

chn channel (data row)

`std` standard deviation

pdf probability density function

[ll](#) log of posterior likelihood on data: v-log-likelihood

[kld](#) Kullback-Leibler distance

niter number of iterations

[options](#) computational options

`seed` seed of random generator

`sig` standard deviation of output noise

Can component in matrix factorized ARX LS form

[Cans](#) array of components in matrix factorized ARX LS form

CUMTAB transition table of components

ACTIVE active component

[Chms](#) channel descriptions

Cryptonyms

Data management	
TIME	processing time
DATA	data sample
ndat	length of data
psi	create regression vector
Psi	data vector
npsi	length of regression vector
nPsi	length of data vector
str	structure of regression vector
Factors	
Fac	factor
Facs	array of factors
fac	position of a factor in an array of factors
ychn	modeled channel
str	structure of regression vector
dfm	degrees of freedom of a factor <i>standard ARX factors</i>
LD	L'DL decomposition of the extended information matrix
L	triangular part of L'DL decomposition
D	diagonal part of L'DL decomposition of extended information matrix
V	information matrix <i>ARX factors in least squares representation</i>
Eth	point estimate of regression coefficients
Cth	covariance of regression coefficients
cove	point estimate of noise covariance
Components	
com	component
coms	array of components
dfcs	vector of degrees of freedom of components
dfcs0	initial degrees of freedom of components
alphas	normalized vector of degrees of freedom of components
Com	matrix ARX or ARX LS component
Coms	array of matrix ARX or ARX LS components
Can	component in matrix factorized ARX LS form
Cans	array of components in matrix factorized ARX LS form
ychns	modeled channels in component
nychn	number of modeled channels
Mixtures	

Mix	mixture estimate
Sim	mixture simulator
pMix	mixture predictor
pMixfix	mixture prediction
facs	list of factors
nfac	number of active factors ^a
ncom	number of components
nchn	number of modeled channels
^a dimensions are computed as : [ncom, nchn] = size(Mix.coms); nFacs = length(Mix.Facs); nfac = length(Mix.states.facs);	

Mixture estimation	
frg	forgetting rate
frgd	default forgetting rate
rate	mixture flattening rate
maxtd	maximum time delay of factors in a mixture
nruns	number of runs in iterative mixture estimation
relerr	relative error
maxerr	maximum possible error
	<i>states in mixture estimation ^a</i>
faclls	trial factor predictions $\log(f(d_{t+1} fac, t+1))$
comlls	component predictions $\log(f(d_t com))$
mixll	mixture prediction $\log(f(d_t mix))$
comwgs	component weights
facwgs	factor weights
^a refer to mixupdt.m for meaning of the statistics	

Mixture projection	
pchns	predicted channels
cchns	channels in condition
psi0	value of zero-delayed regressor

Advisory system design	
aMixc	advised mixture of the type ARX LS + control states
aMixu	desired mixture of the type ARX LS + control states
strc	common control structure
kc	lift of quadratic forms
UDc	cell vector of u'du decompositions of KLD kernels
udca	u'du decomposition of average KLD kernel in UDc
kca	average lift of quadratic forms kc
uchn	list of channels with recognisable actions
pochn	list of channels with o-innovations
outs	list of channels with innovations
npochn	number of channels with o-innovations
udca	u'du decomposition of average KLD kernel in UDc
ufc	normalised vector qualifying components

Structure estimation	
----------------------	--

<code>maxstr</code>	guess of the richest structure
<code>maxFac</code>	richest factor
<code>maxMix</code>	richest mixture
<code>belief</code>	belief on a guess of richest structure
<code>chbelief</code>	belief on factors of a channel
<code>nrep</code>	number of random starts
<code>MAPstr</code>	MAP estimate of the factor structure

General cryptonyms	
<code>DEBUG</code>	global debugging flag
<code>chn</code>	channel (data row)
<code>std</code>	standard deviation
<code>pdf</code>	probability density function
<code>kld</code>	Kullback-Leibler distance
<code>ll</code>	log of posterior likelihood on data: v-log-likelihood
<code>niter</code>	number of iterations
<code>opt</code>	option
<code>options</code>	computational options
<code>seed</code>	seed of random generator
<code>uchn</code>	list of channels with recognisable actions
<code>sig</code>	standard deviation of output noise
<code>CUMTAB</code>	transition table of components
<code>ACTIVE</code>	active component