Two Toolboxes for Systems with Random Delays

Nilsson, Johan

1998

Document Version:
Publisher's PDF, also known as Version of record

Link to publication

Citation for published version (APA):
Nilsson, J. (1998). *Two Toolboxes for Systems with Random Delays*. (Technical Reports TFRT-7572). Department of Automatic Control, Lund Institute of Technology (LTH).

Total number of authors:
1

ISSN 0280–5316
ISRN LUTFD2/TFRT--7572--SE

Two Toolboxes for Systems with Random Delays

Johan Nilsson

Department of Automatic Control
Lund Institute of Technology
April 1998

Department of Automatic Control
Lund Institute of Technology
Box 118
S-221 00 Lund Sweden

Document name

Date of issue
April 1998

Document Number
ISRN LUTFD2/TFRT--7572--SE

Author(s)
Johan Nilsson

Supervisor

Sponsoring organisation

Title and subtitle
Two Toolboxes for Systems with Random Delays

Abstract

This report is the manual for the Matlab toolboxes `pdfbox` and `dlqgbox`. The toolboxes are used for design and analysis of systems with random time delays in the loop. Typically, the delays are transmission delays originating from networks in a distributed real-time control system. The toolbox `pdfbox` contains functions for description of the random delays. The toolbox `dlqgbox` contains functions for analysis of covariances in the closed loop system, and functions for LQG-controller design.

Key words

Classification system and/or index terms (if any)

Supplementary bibliographical information

ISSN and key title
0280–5316

ISBN

Language
English

Number of pages
28

Security classification

Recipient's notes

The report may be ordered from the Department of Automatic Control or borrowed through:
University Library 2, Box 3, S-221 00 Lund, Sweden
Fax +46 46 222 44 22 E-mail ub2@ub2.lu.se

1. Introduction

This is a short manual for two Matlab toolboxes developed to do the calculations in the PhD-thesis Nilsson (1998). The basic problem studied is how to do controller design and analysis of closed loops when there are random delays in the loop. The motivating application for studying random delays is real-time control systems with distributed I/O. Typically, measurements and control signals are sent on a field bus or a computer network. The main toolbox is called `dlqgbox` and is used for design of LQG-controllers and for analysis of covariances in the closed loop. The implementation of `dlqgbox` uses extensive calculations of mathematical expectations. To simplify the implementation an underlying toolbox has been developed, `pdfbox`. The `pdfbox` can be used to build probability distribution functions and to calculate expected values of matrix functions. The toolboxes use some new features in Matlab, which means they require Matlab x, where $x \geq 5$. The report is organized with an example from Nilsson (1998) in Section 2. Sections 4 and 5 contain the reference manuals for `pdfbox` and `dlqgbox` respectively.

Department users

The m-files for the toolboxes are found in the directories

```
/regler/matlab-5/johan/pdfbox
/regler/matlab-5/johan/dlqgbox
```

Correct Matlab paths are set up by the commands

```
>> pdfbox
>> dlqgbox
```

which are functions in `/regler/matlab-5/regler`. The test example in Section 2 can be found in

```
/regler/matlab-5/johan/dlqgbox/testex/testex.m
```

External users

The toolboxes are packed together in the file `rdboxes.tar.Z`. Unpack the toolboxes with the commands

```
> uncompress rdboxes.tar.Z
> tar xvf rdboxes.tar
```

Two directories need to be added to the Matlab path. This is typically done as

```
addpath('.../rdboxes/pdfbox','-end');
addpath('.../rdboxes/dlqgbox','-end');
```

where ... is the install directory. The test example in Section 2 can be found in

```
rdboxes/dlqgbox/testex/testex.m
```

Future work / Bugs

- Theorem 6.2 of Nilsson (1998) is not implemented yet, but starting with `pdfbox`, `finds`, and `lqrdel` it should be a possible generalization.
- During the work a bug in the Matlab function `interp1` was found. The bug has been reported to Mathworks, but no solution is yet released. The bug shows up when giving an interval boundary as argument to `interp1`. The following example shows the buggy behavior.

```
>> xdata=[0:0.006:0.15]; ydata=randn(26,1);
>> interp1(xdata,ydata,0-eps)
ans =
   NaN
>> interp1(xdata,ydata,0)
ans =
    0.8137
>> interp1(xdata,ydata,0.15-eps)
ans =
    1.6360
>> interp1(xdata,ydata,0.15)
ans =
   NaN
```

The function should not return `NaN` for 0.15. When the bug is fixed references to the function `myinterp1`, the temporary fix, can be changed to `interp1`. (Current version is Matlab Version 5.1.0.421, May 25 1997) The temporary solution is used in the functions `M1fun.m`, `M2fun.m`, and `lqrdel.m` in `dlqgbox`.

2. An Example

The example is taken from Section 5.5 of Nilsson (1998). Consider control of the plant

$$\frac{dx}{dt} = \begin{bmatrix} 0 & 1 \\ -3 & -4 \end{bmatrix} x + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u + \begin{bmatrix} 35 \\ -61 \end{bmatrix} \xi \tag{1}$$

$$y = [2 \quad 1]\, x + \eta, \tag{2}$$

where $\xi(t)$ and $\eta(t)$ have mean zero and unit incremental variance. The control objective is to minimize the cost function

$$J = E \lim_{T\to\infty} \frac{1}{T} \int_0^T (x^T H^T H x + u^2) dt, \tag{3}$$

where $H = 4\sqrt{5}\left[\sqrt{35} \quad 1\right]$. The sampling period for the controller is chosen as $h = 0.05$. The following Matlab code sets up the problem data.

```
A=[0,1;-3,-4]; B=[0;1]; Bk=[35;-61]; C=[2,1];
H=4*sqrt(5)*[sqrt(35),1];
Q1c=H'*H; Q2c=1;
R1c=Bk*Bk'; R2c=1;
h=0.05;
```

Synthesis

The continuous time cost function is sampled, Åström and Wittenmark (1997), using the `lqgsamp` function of `lqgbox`.

```
[Phi,Gam,Q1,Q2,Q12,R1,Je] = lqgsamp(A,B,h,Q1c,Q2c,zeros(2,1),R1c);
Q=[Q1,Q12;Q12',Q2];
```

For $\alpha = 1$ the delays, τ_k^{sc} and τ_k^{ca}, are uniformly distributed on the interval $[0, h/2]$. The pdfs are built with the commands.

```
pdftsc=newpdf; pdftca=newpdf;
pdftsc=cpdfadd(pdftsc,'uniform',1,0,h/2);
pdftca=cpdfadd(pdftca,'uniform',1,0,h/2);
```

The LQ-controller of Theorem 5.1 in Nilsson (1998) is calculated by first solving the corresponding Riccati equation. A start value is found by iteration the Riccati recursion some steps.

```
[S0]=startval(A,B,h,Q,pdftsc,pdftca,5);
```

The Riccati equation is then solved using the solver `finds`.

```
[S,PC1,PC2]=finds(A,B,h,Q,pdftsc,pdftca,S0);
```

The function returns the solution `S` together with some precalculations for use in `lqrdel`. The optimal controller

$$u_k = -L(\tau_k^{sc}) \begin{bmatrix} x_k \\ u_{k-1} \end{bmatrix} \tag{4}$$

is calculated using `lqrdel`. The tabular discretization of $L(\tau_k^{sc})$ is here chosen to 100 values.

```
[Lvec,tauvec]=lqrdel(S,Q,100,pdftsc,pdftca,h,PC1,PC2);
```

The resulting $L(\tau_k^{sc})$ is shown in Figure 1. The observer is designed with the

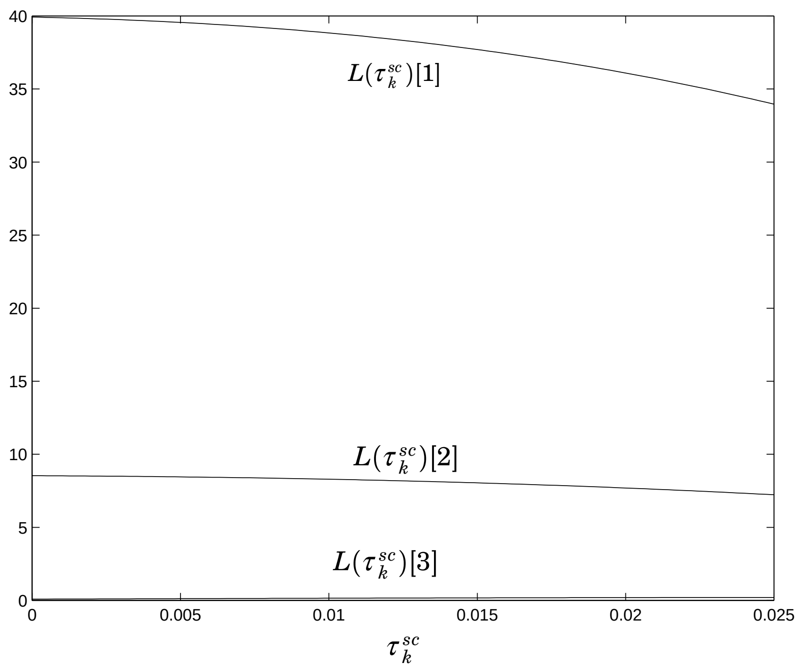


Figure 1 Plot of the elements in $L(\tau_k^{sc})$. The elements are varying smoothly enough for a discretization in 100 points.

standard observer tool `lqed`.

```
[K,Kb,Kv,P,Pf] = lqed(Phi,C,R1,1);
```

Analysis

The idea is to formulate the closed loop system and the use then function `coveval`. Using (5.2) and (5.31) of Nilsson (1998) the closed loop system can be written as

$$z_{k+1} = \begin{bmatrix} \Phi & \Gamma_1() - \Gamma_0()L_2() & -\Gamma_0()L_1() \\ 0 & -L_2() & -L_1() \\ \bar{K}C\Phi & \Gamma_1() - \Gamma_0()L_2() & \Phi - \bar{K}C\Phi - \Gamma_0()L_1() \end{bmatrix} z_k + \begin{bmatrix} \Gamma_v & 0 \\ 0 & 0 \\ \bar{K}C\Gamma_v & \bar{K} \end{bmatrix} e_k, \quad (5)$$

where

$$z_k = \begin{bmatrix} x_k \\ u_{k-1} \\ \hat{x}_{k|k} \end{bmatrix} \quad (6)$$

$$e_k = \begin{bmatrix} v_k \\ w_{k+1} \end{bmatrix}. \quad (7)$$

The arguments of Γ_0, Γ_1, and L have been suppressed. The feedback gain $L(\tau_k^{sc})$ has also been partitioned as $L(\tau_k^{sc}) = [L_1(), L_2()]$. To make e_k have unit variance Γ_v has been introduced as the Cholesky factorization of `R1`. In Matlab this is done as

```
Gammav=chol(R1)';
```

Notice that e_k is of dimension 3. We will analyze the closed loop covariances with the function `coveval`. This needs a function that returns $\Phi(\tau_k^{sc}, \tau_k^{ca}, i)$ and $\Gamma(\tau_k^{sc}, \tau_k^{ca}, i)$. The function `coveval` is designed to handle the situation with a Markov chain postulation the delay distributions. In our case the delays have constant distributions, which corresponds to one Markov state. The delay behavior is described with a *Markov structure*. In this example it is programmed as

```
ms=newms(1,1);
ms=setpdf(ms,1,pdftsc,pdftca);
```

The function describing the closed loop needs some of the matrices describing the process and the controller. This is done by defining these as global variables.

```
global Lvec tauvec Kb A B C h Gammav Q
```

The closed loop is described with the following function named `excl.m`.

```
function [Phii,Gami]= excl(tsc,tca,i);
if isempty(tsc),
  Phii=zeros(5,5);
  Gami=zeros(5,3);
else
  global Lvec tauvec Kb A B C h Gammav
  n=2;
  X=[A,B;zeros(1,n),zeros(1)];
  T1=[eye(n),zeros(n,1)];
```

```
    T2=[zeros(n,1);eye(1)];
    Gam0=T1*expm(X*(h-(tsc+tca)))*T2;
    Gam1=expm(A*(h-(tsc+tca)))*T1*expm(X*(tsc+tca))*T2;
    Phi=expm(A*h);
    L=interp1(tauvec,Lvec,tsc);
    L1=L(1,1:2); L2=L(1,3);

    Phii=[Phi          , Gam1-Gam0*L2, -Gam0*L1;
          zeros(1,2)   , -L2         , -L1;
          Kb*C*Phi     , Gam1-Gam0*L2, Phi-Kb*C*Phi-Gam0*L1];

    Gami=[Gammav     , zeros(2,1);
          zeros(1,2) , 0;
          Kb*C*Gammav, Kb];
  end;
```

For efficiency it would have been advantageous to calculate X, T1, T2, and Phi ones outside the function and use these as global variables. The closed loop covariances are now calculated with

```
Res=coveval('excl',ms);
```

The cost function we are to evaluate can be written as

$$
\begin{aligned}
&\mathrm{E}\begin{bmatrix} x_k \\ u_k \end{bmatrix}^T Q \begin{bmatrix} x_k \\ u_k \end{bmatrix} \\
&\qquad = \mathrm{E}\, z_k^T \begin{bmatrix} I & 0 & 0 \\ 0 & -L_2(\tau_k^{sc}) & -L_1(\tau_k^{sc}) \end{bmatrix}^T Q \begin{bmatrix} I & 0 & 0 \\ 0 & -L_2(\tau_k^{sc}) & -L_1(\tau_k^{sc}) \end{bmatrix} z_k \\
&= \mathrm{tr}\left(\left(\mathop{\mathrm{E}}_{\tau_k^{sc}} \begin{bmatrix} I & 0 & 0 \\ 0 & -L_2(\tau_k^{sc}) & -L_1(\tau_k^{sc}) \end{bmatrix}^T Q \begin{bmatrix} I & 0 & 0 \\ 0 & -L_2(\tau_k^{sc}) & -L_1(\tau_k^{sc}) \end{bmatrix} \right) P \right)
\end{aligned} \tag{8}
$$

The covariance P can be accessed with Res.P. The first part of (8) is calculated using the function expect with the following function named costmat.m.

```
function [out]=costmat(tsc);
if isempty(tsc),
  out=zeros(5,5);
else
  global Lvec tauvec Q
  L=interp1(tauvec,Lvec,tsc);
  L1=L(1,1:2); L2=L(1,3);

  X=[eye(2)     , zeros(2,1), zeros(2,2);
     zeros(1,2) , -L2       , -L1];

  out=X'*Q*X;
end;
```

The expected value is calculated with

```
St=expect(pdftsc,'costmat');
```

The cost can now be evaluated with

```
Cost=trace(St*Res.P)
```

The resulting cost is $8.15 \cdot 10^3$.

3. References

ÅSTRÖM, K. J. and B. WITTENMARK (1997): *Computer-Controlled Systems*, third edition. Prentice Hall.

NILSSON, J. (1998): *Real-Time Control Systems with Delays*. PhD thesis ISRN LUTFD2/TFRT--1049--SE, Department of Automatic Control, Lund Institute of Technology, Lund, Sweden.

4. Reference Guide for `pdfbox`

Function	*Purpose*
cpdfadd	Add a continuous distribution function to a pdf
cuth	Cut all higher than t in a pdf
cutl	Cut all lower than t in a pdf
dpdfadd	Add a Dirac to a pdf
expect	Calculate the expected value of a function
geprob	Calculate the probability that a variable is greater than or equal t
gprob	Calculate the probability that a variable is greater than t
leprob	Calculate the probability that a variable is less than or equal t
lprob	Calculate the probability that a variable is less than t
meanpdf	Calculate the mean value of a pdf
newms	Create a new Markov structure
newpdf	Create a new pdf-variable
plotpdf	Plot a pdf
prob	The probability that a random variable is exactly equal to t.
setpdf	Set pdfs for a Markov structure
varpdf	Calculate the variance of a pdf

cpdfadd

Purpose

Add a continuous distribution function to a pdf

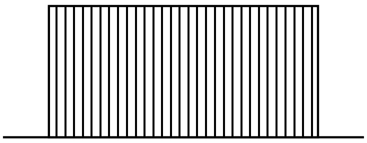
Synopsis

```
opdf=cpdfadd(ipdf,type,ptot,tmin,tmax);
```

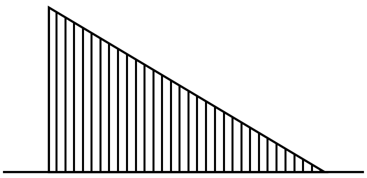
Description

Adds a continuous distribution function function to `ipdf`. The total probability of the added distribution function is `ptot`. If `ipdf` does not contain anything `ptot` must be 1. If `ipdf` already contains a distribution this is scaled down a factor $(1 - pval)$. The added distribution has support on the interval $[tmin, tmax]$. The distribution type is set with the parameter `type`. The following types are implemented.

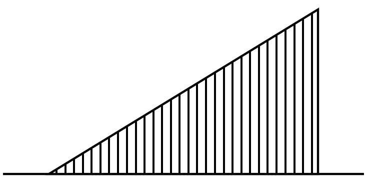
- `'uniform'` Uniform distribution with support on the interval $[tmin, tmax]$, and total probability `ptot`.



- `'ltriang'` Left triangular distribution with support on the interval $[tmin, tmax]$, and total probability `ptot`.



- `'rtriang'` Right triangular distribution with support on the interval $[tmin, tmax]$, and total probability `ptot`.



Example

This command sequence creates a pdf with uniform distribution on $[0, 1]$.

```
pdf=newpdf;
pdf=cpdfadd(pdf,'uniform',1,0,1);
```

See Also

`newpdf dpdfadd`

cuth

Purpose

Cut all higher than t in a pdf

Synopsis

```
[opdf,Ph]=cuth(ipdf,t)
```

Description

Cut all higher than `t` in `ipdf`. The new distribution, `opdf`, is scaled to be a valid pdf. The part cut had the probability `Ph`.

See Also

`cutl`

cutl

Purpose

Cut all lower than t in a pdf

Synopsis

```
[opdf,Pl]=cutl(ipdf,t)
```

Description

Cut all lower than `t` in `ipdf`. The new distribution, `opdf`, is scaled to be a valid pdf. The part cut had the probability `Pl`.

See Also

`cuth`

dpdfadd

Purpose

Add a Dirac to a pdf

Synopsis

```
opdf=dpdfadd(ipdf,tval,pval)
```

Description

Adds a Dirac function to `ipdf`. The probability of the Dirac is `pval`, and it is added for the value `tval`. If `ipdf` does not contain anything `pval` must be 1. If `ipdf` already contains a distribution this is scale down a factor $(1 - pval)$.

Example

This command sequence creates a pdf with probability 0.5 for the values 0 and 1.

```
pdf=newpdf;
pdf=dpdfadd(pdf,0,1);
pdf=dpdfadd(pdf,1,0.5);
```

See Also

`newpdf cpdfadd`

expect

Purpose

Calculate the expected value of a function

Synopsis

```
val=expect(pdf,pfun,p1,p2,p3,...)
```

Description

Calculates the expected value of a matrix valued function `pfun`. The first argument of `pfun` is a random variable with distribution `pdf`. The optional parameters `p1`, `p2`, `p3` etc are passed on to `pfun`. `pfun` is a matrix valued function with the following definition.

```
out=pfun(t,p1,p2,p3,...)
```

If `t` is empty ([]) a matrix with the same size as the usual output for the `pfun` should be returned.

Example

The following is an example `pfun` for calculating the mean of a pdf.

```
function [out]=meanfun(t);

if isempty(t),
  out=0;
else
  out=t;
end;
```

Algorithm

The following integral is calculated numerical.

$$\mathrm{E}(pfun) = \int pdf(s)\, pfun(s)\, ds$$

Diracs are taken out of the integral, and continuous parts of the integral is calculated with `ode23`.

See Also

`meanpdf` `varpdf`

geprob

Purpose

Calculate the probability that a random variable is greater than or equal t.

Synopsis

```
P=geprob(pdf,t)
```

Description

Calculates the probability that a random variable with distribution `pdf` is greater than or equal `t`.

See Also

`gprob leprob lprob prob`

gprob

Purpose

Calculate the probability that a random variable is greater than t.

Synopsis

```
P=gprob(pdf,t)
```

Description

Calculates the probability that a random variable with distribution `pdf` is greater than `t`.

See Also

`geprob leprob lprob prob`

leprob

Purpose

Calculate the probability that a random variable is less than or equal t.

Synopsis

```
P=leprob(pdf,t)
```

Description

Calculates the probability that a random variable with distribution `pdf` is less than or equal `t`.

See Also

`geprob` `prob`

lprob

Purpose

Calculate the probability that a random variable is less than t.

Synopsis

```
P=lprob(pdf,t)
```

Description

Calculates the probability that a random variable with distribution `pdf` is less than `t`.

See Also

`geprob gprob leprob prob`

meanpdf

Purpose

Calculate the mean value of a pdf

Synopsis

```
mean=meanpdf(pdf)
```

Description

Calculates the mean value of `pdf`.

See Also

`expect varpdf`

newms

Purpose

Create a new Markov structure

Synopsis

`ms=newms(s,Q)`

Description

Create a Markov chain structure, `ms`, with two pdfs associated with each Markov state. The pdfs are added with the function `setpdf`.

Parameters

`s`: number of states in the Markov chain

`Q`: transistion matrix for the Markov chain

See Also

`setpdf`

newpdf

Purpose

Create a new pdf-variable

Synopsis

```
pdf=newpdf()
```

Description

Creates a new pdf. The new pdf i not contain anything yet and is not valid for use in calculations.

See Also

`dpdfadd` `cpdfadd`

plotpdf

Purpose

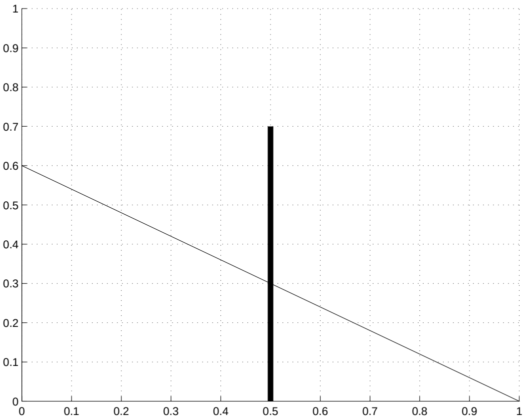
Plot a pdf

Synopsis

```
plotpdf(pdf)
```

Example

```
pdf=newpdf;
pdf=cpdfadd(pdf,'ltriang',1,0,1);
pdf=dpdfadd(pdf,0.5,0.7);
plotpdf(pdf);
```



Description

Plots both continuous and discrete parts of a pdf.

prob

Purpose

Calculate the probability that a random variable is exactly equal to t.

Synopsis

```
P=prob(pdf,t)
```

Description

Calculates the probability that a random variable with distribution `pdf` is exactly equal to `t`. This needs `pdf` to contain a Dirac at `t`.

See Also

`geprob` `leprob`

setpdf

Purpose

Set pdfs for a Markov structure

Synopsis

```
msout=setpdf(msin,i,pdftsci,pdftcai)
```

Description

Set the two pdfs assosiated with Markov state `i`.

Parameters

`msin`**:** Markov structure to modify

`i`**:** Markov state number

`pdftsci`**:** Pdf for τ^{sc}

`pdftcai`**:** Pdf for τ^{ca}

See Also

`newms`

varpdf

Purpose

Calculate the variance of a pdf

Synopsis

```
var=varpdf(pdf)
```

Description

Calculates the variance of `pdf`.

See Also

`expect meanpdf`

5. Reference Guide for `dlqgbox`

Function	*Purpose*
coveval	Covariance matrix evaluation
finds	Solve stochastic Riccati equation
lqrdel	Calculate optimal LQG-controller
startval	Iterate stochastic Riccati equation

coveval

Purpose

Covariance matrix evaluation

Synopsis

```
[Res]=coveval(clfun,ms)
```

Description

Covariance matrix evaluation by Theorem 6.1 in Nilsson (1998). The covariance R is assumed to be an identity matrix. If not, it can be included in the Γ-matrix by Cholesky factorization. The returned covariance matrix is only valid if the closed loop system is stable.

Parameters

`Res`**:** Matlab-structure containing the result as the following parts.

> `P`**:** Stationary covariance matrix for the closed loop system `clfun`
>
> `stable`**:** 1 if system is stable, 0 if not stable
>
> `Ptilde(i).val`**:** $\widetilde{P}_i$-matrices. These can be necessary when later evaluating a quadratic cost function.

`clfun`**:** cl-function describing the closed loop

$$z_{k+1} = \Phi(\tau_k^{sc}, \tau_k^{ca}, i) z_k + \Gamma(\tau_k^{sc}, \tau_k^{ca}, i) e_k, \tag{9}$$

where e_k is white noise with unit covariance. The cl-function should have the following synopsis

```
function [Phii,Gami]=clfun(tsc,tca,i)
```

where `tsc` and `tca` are the delays, and `i` is the Markov state number. If called with both `tsc` and `tca` empty ([]), `Phii` and `Gami` of correct sizes should be returned.

Precalulated variables can be used in `clfun` by declaring them as global in both the function calling `coveval`, and in `clfun`.

`ms`**:** Markov structure describing the underlying Markov chain and delays.

See Also

`pdfbox newms setpdf chol`

finds

Purpose

Solve stochastic Riccati equation

Synopsis

```
[S,PC1,PC2]=finds(A,B,h,Q,pdftsc,pdftca,S0)
```

Description

Solve stochastic Riccati equation in Theorem 5.1 of Nilsson (1998). The time delays τ_k^{sc} and τ_k^{ca} are assumed to have the probability distribution function pdftsc and pdftca. The delay variation may never be greater than h.

Parameters

`A,B`: System matrices

`h`: Sampling period

`Q`: Cost function

`pdftsc,pdftca`: Probability distribution functions

`S0`: Optional argument, initial guess for `S`

`S`: Stationary solution of the Riccati equation

`PC1,PC2`: Precalculated data for use in `lqrdel`

Algorithm

Kleinmann iteration

See Also

`startval` `lqrdel`

lqrdel

Purpose

Calculate optimal LQG-controller

Synopsis

```
[Lvec,tauvec]=lqrdel(S,Q,nvals,pdftsc,pdftca,h,PC1,PC2)
```

Description

Calculate optimal LQG-controller in Theorem 5.1 of Nilsson (1998). The time delays τ_k^{sc} and τ_k^{ca} are assumed to have the probability distribution function pdftsc and pdftca. The delay variation may never be greater than h. `S`, `PC1`, `PC2` are calculated using `finds`.

Parameters

`S`**:** Stationary solution of the Riccati equation

`Q`**:** Cost function

`nvals`**:** number of values in each cont part of the L-tabular

`pdftsc,pdftca`**:** Probability distribution functions

`h`**:** Sampling period

`PC1,PC2`**:** Precalculated data from `finds`

`Lvec`**:** Vector with the feedback L for τ_{sc} in `tauvec`

`tauvec`**:** Values for τ_{sc} in `Lvec`

See Also

`finds`

startval

Purpose

Iterate stochastic Riccati equation

Synopsis

```
S0=startval(A,B,h,Q,pdftsc,pdftca,nit,Sinit)
```

Description

Iterate stochastic Riccati equation in Theorem 5.1 of Nilsson (1998) to find a stabilizing solution. This can be necessary for the Kleinmann iteration done in `finds` to converege. The time delays τ_k^{sc} and τ_k^{ca} are assumed to have the probability distribution function pdftsc and pdftca. The delay variation may never be greater than h.

Parameters

`A,B`**:** System matrices

`h`**:** Sampling period

`Q`**:** Cost function

`pdftsc,pdftca`**:** Probability distribution functions

`nit`**:** Number of iterations to do

`Sinit`**:** Start value for S, default$= 1E6 * eye(n+1)$

`S0`**:** Value for S after `nit` iterations

See Also

`finds` `lqrdel`