Qsyn

the Toolbox for
Robust Control Systems Design
for use with Matlab

Reference Guide

July 1996

ADD2BND

Purpose

Adds/replaces bound in a bound file

Syntax

```
add2bnd(boundfile,w,bound,boundname);
```

Inputs

`boundfile`	name file to save the bound in, string with or without the extension `.bnd`.
`w`	the frequency, rad/s.
`bound`	the bound vector, deg + j*dB.
`boundname`	the name of the bound, string variable, e.g. `'odsrs'`, `'idsrs'`, `'rsrs'` ...

Example

```
hngrid           % generate empty Nichols diagram

[deg, dB]=ginput;% click on desired points from
                 % left to right, to define the
                 % new bound, and finish with
                 % Enter.

bound2 = deg+j*dB;      % Note: this is a column
                        % vector.

add2bnd('ex5_5',2,bound2,'bound'); % IMPORTANT:
                                   % insert row
                                   % vector.
```

Add the row vector `bound2` into the bound file `ex5 5.bnd` as the bound for 2 rad/s under the name `bound`. The actual bound variable name will be `bound` index2 where index2 is the index associated with 2 rad/s. See BNDINF.

See also

SHOWBND BNDINF GETBND

ADD2MAT

Purpose

combines two matrices

Syntax

`v=add2mat(v1,v2,x)`

Outputs

`v`	resulting matrix

Inputs

`v1, v2`	vectors. Each can be either a row or column vector, independently of the other.
`x`	switch If `x`=1: `v` = [`v1 v2`] Otherwise: `v` = [`v1`;`v2`] with empty elements in `v` filled with `NaN`

Example

```
>>v1 = [1 2]; v2 = [3; 4; 5]; v = add2mat(v1,v2,1)
        v =
            1      2      3
            NaN    NaN    4
            NaN    NaN    5

>>v = add2mat(v1,v2,-1)
        v =
            1      2
            3    NaN
            4    NaN
            5    NaN
```

ADD2SPC

Purpose

Adds/replaces specification in a specification file

Syntax

`add2spc(specfile,specname,w,spec);`

or

`add2spc(specfile,specname,w,num,den);`

Inputs

`specfile`	name of specification file to save the spec in, with or without the extension `.spc`.
`specname`	name of specificatione e.g. `'odsrsc'`,`'idsrs'`....
`w`	the frequency vector, rad/s.
`spec`	the specification. Can be 1: a constant (c.f. a 6 dB sensitivity bound) 2: a vector of the same length as `w` 3: a matrix with as many rows as there are elements in `w` and as many columns as you like
`num,den`	two vectors including the constants of filter parameters. The filter is `num(s)/den(s)` where `num` and `den` are the numerator and denominator polynomials, respectively, in Matlab notation. The specification will be the magnitude in dB of the frequency function `|num(j*w)/den(jw)|` evaluated at the frequencies `w`.

Example

Create a 6 dB sensitivity specification and show it:

```
add2spc('test','sens',logspace(-2,2),6);
showspc('test','sens');
```

Create a sensitivity specification from the filter s/(s+1), and show it:

```
add2spc('test','sens',logspace(-2,2),[1 0],[2 1]);
showspc('test','sens');
```

Remarks

The units of the specification should correspond to the criterion function it is going to be used for. If a standard criterion function is going to be used (e.g. `frsrs.m`) the unit is dB.

ADD2TPL

Purpose

Adds/replaces/combines a new template into a template file

Syntax

```
add2tpl(tplfile,w,tpl,tnom,op,par)
```

Description

adds a new template to a tpl file. If there is already a template at the frequency `w`, then it replaces it or creates the union with the existing template, according to the input argument `op`.

Inputs

`tplfile`	name of template file into which the new template enters, string variable with or without the extension `.tpl`.
`w`	frequency (rad/s).
`tpl`	template vector. Each element is of the form deg+j*dB. Empty if you just want to add a nominal point.
`tnom`	the nominal point (deg + j*dB).
`op`	one of the following: `'r'` = replacement operation; (default) `'u'` = union operation.
`par`	(optional) `kxN` matrix of the parameter combinations that produced the template. `k` is the number of uncertain parameters in the transfer function, and `N` the number of points in `tpl`. When appropriate, the parameters in each column of par should be ordered is in the plant description file that generated the template. If `tpl=[]`, then `par` is assumed to be the nominal parameters and must be a `k` by `1` matrix.

Remarks

If `tnom` is empty in a union operation, the nominal point will remain unchanged, otherwise `tnom` will be the new nominal point.
If the templates file does not exist, a new templates file is created, for both alternatives of `op`.

Example

```
tvec = [0+j*10;0+j*5;-10+j*10;-10+j*5];
add2tpl('tex1',10,tvec,-5+j*7,'r')
```

puts the templates vector tvec into the templates file `tex1.tpl`, for for 10 rad/s. The nominal point is -5+j*7 (deg+j*dB). If a template for 10 rad/s already existed in `tex1.tpl`, it is replaced.

See also

INSERT, CMBTPL, CTPL, TPLINF, SHOWTPL, PNOMINAL

ADEDGE

Purpose

Calculates templates with the recursive edge grid method

Syntax

`[T,Qpar]=adegdge(trf,s,qpar,Tacc,n);`

Description

Subroutine to CTPL

Template generation for transfer functions with parametric uncertainty. The method recursively makes a grid over the edges of the parameter space finer and finer until the prescribed accuracy is achieved. The final result (in Nichols form), after pruning, is a set of points representing the border of the value set (see the m-function `T=prune(t,Tacc)`).

Outputs

`T`	final template vector, with elements in Nichols form (dB + j*dB).
`Qpar`	parameter matrix for the final template points. Each column represents one plant case. The parameters are in the same order as in `qpar`.

Inputs

`trfun`	user defined m-function `[t]=trfun(q,s)` that gives the frequency response (in Nyquist form) for all parameter combinations in a column of q. `s` is j*w.
`s`	j*w
`qpar`	[`qmin`,`qmax`] The range of the uncertain parameters. The first column is the minimum values, and second column is the maximum values. If `qmin(i)=qmax(i)` the algorithm will detect that this is a constant parameter.
`Tacc`	[`degree_accuracy` , `dB_accuracy`], prescribed 2-norm accuracy in Nichols form.
n	(Optional) Initial grid of the parameterspace. If `n(i)`=1 then parameter `i` will be considered to be constant and the mean value of `qmin(i)` and `qmax(i)` will be used.

Remarks

WARNING: Note that it is not generally true that the the edges of the parameter box gives the full value set, see [J. Ackermann, Robust Control: Systems with uncertain physical Parameters, Springer Verlag, 1993], where the edge theorem is investigated. If you are unsure, compare the result with that given by `T=adgrid(trf,s,qpar,Tacc,n)`, which however is slower.

ADGRID

Purpose

Template computation with the recursive grid method

Syntax

```
[T,Qpar]=adgrid(trfun,s,qpar,Tacc,n,plot_on);
```

Description

Subroutine to CTPL

Template generation for transfer functions with parametric uncertainty. The method recursively makes a grid over the parameter space finer and finer until the prescribed accuracy is achieved. The final result (in Nichols form) is a set of points representing the border of the value set, after pruning (see the m-function `T=prune(t,Tacc)`.

Outputs

`T`	final template vector, with elements in Nichols form (dB + j*dB)
`Qpar`	parameter matrix for the final template points. Each column represents one plant case. The parameters are in the same order as in qpar.

Inputs

`trfun`	user defined m-function `[t]=trfun(q,s)` that gives the frequency response (in Nyquist form) for all parameter combinations in a column of `q`. `s` is j*w.
`s`	j*w
`qpar`	[`qmin`,`qmax`] The range of the uncertain parameters. The first column is the minimum values, and second column is the maximum values.If `qmin(i)=qmax(i)` the algorithm will detect that this is a constant parameter.
`Tacc`	[`degree_accuracy` , `dB_accuracy`], prescribed 2-norm accuracy in Nichols form.
`n`	(Optional) Initial grid of the parameterspace. If `n(i)`=1 then parameter `i` will be considered to be constant and the mean value of qmin(i) and qmax(i) will be used. Default: `n`=3*(number of uncertain parameters)
`plot_on`	`plot_on`=1, plot the entire recursion. Default: [] (=plot off)

Remarks

Reference:

[B. Cohen, M. Nordin and P.-O. Gutman, Recursive Grid Methods to Compute Value Sets for Uncertain Transfer Functions, ACC 1995]

BD11SPC

Purpose

Calculates bd11-modified servo specs for first step of 2x2 MIMO.

Syntax

```
bd11spc(specfile1,specname2,bd11name,bndfile,tplfi
le,specname1,w,specfile2)
```

Inputs

`specfile1` File name (string) of the file to contain the modified servo specification. NO EXTENSION ALLOWED, `.spc` added automatically.

`specname2` Name (string) of specification variable that holds the original servo specification.

`bd11name` Name of bd11-matrix (string) in `bndfile` (excluding _index, see remarks).

`bndfile` File name (string) of the file that contains the bd11-matrices.NO EXTENSION ALLOWED, `.bnd` added automatically. Default: `bndfile=specfile1`

`tplfile` File name (string) of the file that contains the open loop templates.NO EXTENSION ALLOWED, `.tpl` added automatically.Default:
`tplfile=specfile1`.

`specname1` Name (string) of specification variable that is to hold the modified servo specification.Default: `specname1 = [specname2,'m']`.

`w` the frequencies for which to calculate the modified servo specification.Default: the frequencies for
which the bd11-matrices exist in `bndfile`.

`specfile2` File name (string) of the file containing the original servo specification. NO EXTENSION ALLOWED, `.spc` added automatically.Default: `specfile2 = specfile1`.

Remarks

It is imperative that the bd11-matrices is saved in `table2.m`-format:
`[x,GPdeg(:)';GPdB(:), [values of bd11]];`
where `x` is any finite or infinite number or NaN,
`GPdeg` is a ordered vector spanning at least [-360,0] deg, whose elements have the unit deg, `GPdB` is an ordered vector (dB), and that the value bearing elements of the bd11-matrix have absolute units (not dB).
The internal name of the bd11-matrix in `bndfile` will _index appended,exactly as the internal bound names, where the index refers to thefrequency, see the command `add2bnd`.
The open loop template file is easily constructed with the command `tplfof`, prior to using `bd11spc`.

Author: P-O Gutman

BNDINF

Purpose

displays information about a bound file

Syntax

```
bndinf(bndf,name1,name2,name3,name4,name5,name6);
```

Description

Displays on screen information about a bound file.
It always displays information about the standard bounds:
`odsrs`, `idsrs`, `odsrsc`, `iosrs`, `rsrs`.
If there are other (user defined) bounds, you must give the names of these bounds. To check which names are included you can use e.g. the command `look('ex1.bnd');`

Inputs

`bndf` bound file name, string variable with or without extension `.bnd`.

`name1, name2,...`
strings containing user defined bound names, e.g. `'rsrs1'`

Example

```
bndinf('ex6');
filename = 'ex6'; bndinf(filename);
bndinf('ex6','sens','rsrs1','rsrs2')
```

BNDUPD

Purpose

Recalculates bounds with higher accuracy INTERACTIVELY.

Syntax

```
bndupd(bndfile,specname,w,plot off,specfile, ...
tplfile,bndname,specfun,Tacc)
```

Description

For each frequency, the user selects areas on screen where the bound should be refined. Pressing *enter* takes you to the next frequency, pressing *q* means quit.

Inputs

`bndfile` File that shall contain the bounds, NO EXTENSION ALLOWED, `'.bnd'` added automatically.

`specname` The specification name, predefined names are:

- `iosrs` GP/(1+GP) sensor noise gain, complementary sensitivity.
- `idsrs` P/(1+GP) input disturbance to plant output.
- `odsrs` 1/(1+GP) output disturbance to plant output, sensitivity.
- `odsrsc` G/(1+GP) output disturbance to control output/plant input.
- `rsrs` max(GP/(1+GP))/min(GP/(1+GP)) servo specification.

`w` the frequencies for which to calculate bounds, must be a subset of the template frequencies. default is all of them.

`plot off` flag to supress (`plot off=1`) plotting of calculated bounds. Default is to plot.

`specfile` File that contains the specifications, default is `specfile=bndfile`. NO EXTENSION ALLOWED, `'.spc'` added automatically

`tplfile` file that contains the template, default is `tplfile=bndfile`. NO EXTENSION ALLOWED, `.tpl` added automatically

`bndname` the name of the new bound, default is `bndname=specname`.

`specfun` The name of the criterion function required to calculate a certain bound.default is [`'f'`,`specname`] e.g. the function required for the sensitivity specification `'odsrs'` calls the function `'fodsrs'`

`Tacc` The required accuracy in [deg dB], default is [3 1].

Example

```
bndupd('ex1','odsrs')
```

calculates sensitivity bounds for all templates in `ex1.tpl`, given the specifications in `ex1.spc` and stores the result in `ex1.bnd`.

BUILDF

Purpose

Extracts transfer fcn factors from the rff string format in a plant file.

Syntax

```
[Known_Factors,UnKnown_Factors,Un_templates]=buildf(P)
```

Description

extract and build factors from rff string format. Subroutine for rff template computations.

Purpose

Converts a complex matrix from complex to Nichols form.

Syntax

```
y = c2n(x,op)
```

Outputs

`y`	resulting matrix in Nichols form, i.e. each element is given as deg + j*dB.

Inputs

`x`	original matrix in complex form, i.e. each element is given as a + j*b.	
`op`	one of the following:	
	`'wrap'`	for each matrix element, the phase, deg, is wrapped into the range [-360,0] degrees (default);
	`'unwrap'`	the phase is unwrapped columnwise, i.e.continuous over Riemann surface boundaries, if possible;
	real number, `c`	the phase is unwrapped columnwise, i.e.continuous over Riemann surface boundaires, if possible, with the phase of the (1,1)-matrix element belonging to the Riemann surface [c-180,c+180] degrees.

Example

```
a =      [1.0000 + 0.1000i   1.0000 - 0.1000i
          1.0000 - 0.1000i  -1.0000 - 0.1000i
         -1.0000 - 0.1000i  -1.0000 + 0.1000i
         -1.0000 + 0.1000i   1.0000 + 0.1000i];

y=c2n(a)

y =
          1.0e+002 *
         -3.5429 + 0.0004i  -0.0571 + 0.0004i
         -0.0571 + 0.0004i  -1.7429 + 0.0004i
         -1.7429 + 0.0004i  -1.8571 + 0.0004i
         -1.8571 + 0.0004i  -3.5429 + 0.0004i

y=c2n(a,'unwrap')

y =
          1.0e+002 *
          0.0571 + 0.0004i  -0.0571 + 0.0004i
         -0.0571 + 0.0004i  -1.7429 + 0.0004i
         -1.7429 + 0.0004i  -1.8571 + 0.0004i
         -1.8571 + 0.0004i  -3.5429 + 0.0004i

y=c2n(a,-179)

y =
          1.0e+002 *
         -3.5429 + 0.0004i  -3.6571 + 0.0004i
         -3.6571 + 0.0004i  -5.3429 + 0.0004i
         -5.3429 + 0.0004i  -5.4571 + 0.0004i
         -5.4571 + 0.0004i  -7.1429 + 0.0004i
```

See also

Matlab function UNWRAP, N2C.

CASES

Purpose

Plant frequency domain simulation for user selected cases.

Syntax

```
[T,freq,nom]=cases(Plant,Par,freq,nic bode)
```

Outputs

`T`	The simulation result, in the form of an `(m x length(freq))` matrix, where each row represents one plant case. Each element of `T` is of the form [deg+j*dB]. `T` does not include the nominal case (unless it happens to belong to `Par`).
`freq`	frequencies for which `T` and `nom` was computed. Row vector, [rad/s].
`nom`	plant nominal frequency function. Row vector, [deg+j*dB].

Inputs

`Plant`	Plant description file name, string with or without extension `.m`, (see `plant.m`).
`Par`	Matrix that contains the cases to be simulated. The dimention of this matrix is `nxm` where `n` is the number of uncertain parameters (in the plant description file) and `m` is the number of cases to be simulated. Each column of `Par` contains one plant case, i.e. one parameter combination. Notice that unstructured uncertainty is not considered. If `Par` = `'all'`, the number of cases, and the cases themselves, are taken from `Plant`. If `Par` = [], only the nominal plant is displayed (and `T`=[]).
`freq`	Row vector of simulation frequencies [rad/s]. Default when `freq`=[] or when `freq` does not exist: {`w nom U w tpl`} in `Plant` (`U` stands for union).
`nic_bode`	A flag that indicates how the simulation will be presented. `nic_bode` = 0: in a Nichols diagram; `nic_bode` = 1: in a Bode diagram; `nic bode` = -1: No diagram. Default (`nic_bode` non existing): `nic_bode` = -1.

Example

```
[T,tfreq,nom,nfreq] = cases('ex2 1b','all',[],1);
```

gives a Bode plot of all cases of plant `ex2_1b.m` for frequencies {`w_nom U w_tpl`}, and the simulation results as outputs.

```
[T,tfreq,nom,nfreq] =cases('ex2_1a',[2 5; 1 3; .3
.3; 8 4],[.1 1 10 100],1);
```

gives a Bode diagram for the plant `ex2 1a.m` of the two defined plant cases for the frequencies [.1 1 10 100], and the nominal plant for the frequencies {`w nom U w tpl`}, and the simulation results.

```
T = cases('ex2 1a',[2 5; 1 3; .3 .3; 8 4],[],0);
```

gives a Nichols diagram of the two defined plant cases and the nominal plant from plant `ex2_1a.m` for the frequencies {`w_nom U w_tpl`} and matrix `T` only.

```
[T,tfreq] = cases('ex2_1a',[2 5; 1 3; .3 .3; 8
4]);
```

gives `T` for the two defined plant cases from plant `ex2_1a.m` for the frequencies `tfreq =` $\{$`w nom U w tpl`$\}$, and no graphic output.

```
T = cases('ex2 1a',[],[],0);
```

gives a Nichols diagram of the nominal plant from `ex2 1a.m` for the frequencies $\{$`w_nom U  w_tpl`$\}$, and an empty `T`.

```
cases('ex2 1a',[],[],1);
```

gives a Bode diagram of the nominal plant from `ex2 1a.m` for the frequencies $\{$`w_nom U  w_tpl`$\}$

See also

FDESIGN, PGRID, PARGRID, CTPL, PNOMINAL, CCASES.

CBND

Purpose

Calculate bounds from given templates and specifications.

Syntax

```
cbnd(bndfile,specname,w,plot off,specfile,...
tplfile,bndname,specfun,Tacc,dBlimit,par_nom,...
par,case)
```

Inputs

`bndfile` File that shall contain the bounds, default is `bndfile=tplfile`. NO EXTENSION ALLOWED, `'.bnd'` is added automatically

`specname` The specification name, predefined names are:

- `iosrs` GP/(1+GP) sensor noise gain, complementary sensitivity
- `idsrs` P/(1+GP) input disturbance to plant output
- `odsrs` 1/(1+GP) output disturbance to plant output, sensitivity
- `odsrsc` G/(1+GP) output disturbance to control output/plant input
- `rsrs` max(GP/(1+GP))/min(GP/(1+GP)) servo specification

`w` the frequencies for which to calculate bounds, must be a subset of the template frequencies. Default is all of them.

`plot_off` flag to supress (`plot_off`=1) plotting of calculated bounds. Default is to plot

`specfile` File that contains the specifications, default is `specfile=bndfile`. String. NO EXTENSION ALLOWED, `'.spc'` added automatically.

`tplfile` file that contains the template String. NO EXTENSION ALLOWED, `.tpl` added automatically

`bndname` the name of the new bound, default is `bndname=specname`.

`specfun` The name of the criterion function required to calculate a certain bound. Default is [`'f'`,`specname`] e.g. the function required for the sensitivity specification `'odsrs'` calls the function `'fodsrs'`.

`Tacc` The required accuracy in [deg dB], default is [3 1]

`dBlimit` Decides the search area in which to find the bounds, in [min max] db, default is [-50 50] dB.

`par_nom,par` auxiliary arguments for the criterion function `specfun`, (see `case`).

`case` = `'siso'` standard SISO case (incl. Cascaded SISO, and some MIMO bound computations), (default).
= `'mimo2'` the cross coupling bound computation in the second step of the standard servo 2x2 MIMO bound computation. In this case, `par` = the name of the template file containing the (w21/w22e)*(L10/w11)*F11/(1+L10/w11)-templates, string WITHOUT EXTENSION, `'.tpl'` added automatically if needed.

Example

```
cbnd('ex1','odsrs')
```

Calculates sensitivity bounds for all templates in `ex1.tpl`, given the specifications in `ex1.spc` and stores the result in `ex1.bnd`.

CCASES

Purpose

Closed loop frequency function display for user selected cases.

Syntax

```
[T,freq,nom]=ccases(Plant,Par,op,G,F,freq,plot op)
```

Outputs

`T`	The simulation result, in the form of an `(m x length(freq))` matrix, where each row represents one closed loop plant case such as e.g. GP/(1+GP) or FGP/(1+GP). Each element of `T` is of the form [deg+j*dB]. `T` does not include the nominal case (unless it happens to belong to `Par`).
`freq`	frequencies for which `T` and `nom` was computed. Row vector, [rad/s].
`nom`	plant nominal frequency function. Row vector, [deg+j*dB].

Inputs

`Plant`	plant description file name, string with or without extension `.m`, (see `plant.m`).
`Par`	Matrix that contains the cases to be simulated. The dimension of this matrix is `nxm` where `n` is the number of uncertain parameters (in the plant description file) and `m` is the number of cases to be simulated. Each column of `Par` contains one plant case, i.e. one parameter combination. Notice that unstructured uncertainty is not considered. If `Par` = `'all'`, the number of cases, and the cases themselves,are taken from `Plant`.If `Par` = [], only the nominal plant is displayed (and `T`=[])
`op`	the closed loop operation, one of

`'ol'`	GP the open loops
`'iosrs'`	GP/(1+GP) sensor noise gain, complementary sensitivity.
`'idsrs'`	P/(1+GP) plant input disturbance to plant output.
`'odsrs'`	1/(1+GP) output disturbance to plant output, sensitivity.
`'odsrsc'`	G/(1+GP) output disturbance to controller output/plant input
`'rsrs'`	FGP/(1+GP)) servo specification you can also enter your own expression e.g. op='`P.*G./(1+G)`'

`G,F`	controller feedback and feedforward controller file names, strings with or without extension `.m`. When `G` or `F` are not given or set to [], the default values are 1 (O dB).
`freq`	Row vector of simulation frequencies [rad/s]. Default when `freq`=[] or when `freq` does not exist: {`w_nom U w tpl`} in P (`U` stands for union).
`plot_op`	plot option , one of the following:

`'none'`:	no plot, default
`'bode','nichols','mag','phase'`:	Bode, Nichols,magnitude and phase plots, respectively.

Example

```
figure, showspc('ex2 1a','odsrs','freq');
ccases('ex2_1a',par,'odsrs','g1','f1',...
```

```
logspace(-1,2,120),'mag');
```

This example plots, in a Bode magnitude diagram, the odsrs frequency domain specification together with the closed loop sensitivity function 1/(1+GP), where the plant P is defined by `ex2_1a.m`, the feedback controller G by `g1.m`, and the plant cases by the matrix par. The prefilter `'f1' (f1.m)` could have been omitted.

See also

FDESIGN, PGRID, PARGRID, CTPL, CASES.

CDESIGN

Purpose

Interactive feedback compensator design.

Syntax

```
[gh]=cdesign(P,G,h,color,t color,w tick,w)
```

Description

The function `cdesign` plots the nominal open loop Pnom*G in a Nichols chart. Use with `showbnd` to design the feedback G. The function cdesign can also be used to display any frequency function in controller form in a Nichols chart.

Inputs

`P`	one of: (1) a template file name, string with extension `.tpl`, e.g. `plant.tpl` (2) a plant file name, string with extension '.m'. Only the nominal case is used. (3) a controller file name, string with extension `.m`. (4) [], (i.e. empty), which means a constant of 0 dB.
`G`	one of: (1) a controller file name , string with or without the extension '.m' (2) [], which means a constant of 0 dB.
`h`	handle or handles to previous cdesign plots, which should be deleted in the current figure, before the new plot is put on. If `phandle=[]`, it plots in the current figure, if `phandle='new'` a new figure will open.
`color`	the color and line style of the plot, e.g. `'r:'`
`t color`	the COLOR ONLY of the tick marks, e.g. `'r'` or `'c5'`.
`w_tick`	vector containing the tick mark frequencies.Default is given by plant- or template files `P`. If `P=[]`, then `w` must be given.
`w`	frequency vector for the open loop. Default is given by plant- or template files `P`. If `P=[]`, then `w` must be given.

Remarks

This command works only with files in the current working directory.

Example

plot the nominal loop for two controllers together with bounds.

```
showbnd('ex1,[],[],'rsrs')
h0=cdesign('ex1.m');            % plot the  nominal
                                % open loop with G=1
h1=cdesign('ex1.m','g1');       % plot the nominal
                                % open loop with
                                %
roller g1
h2=cdesign('ex1.m','g2',h0);    % erase the nominal
                                % open loop with G=1,
                                % and  plot the
                                % nominal open loop
                                % with controller g2.
```

See also

SHOWBND, FDESIGN, SHOWTPL.

CLTMP

Purpose

Interpolates points in a sorted and pruned template (subroutine).

Syntax

```
[nt]=cltmp(t,dist)
```

Description

closes the distance between two neighbouring points in the template to a desired distance, by adding new template points along the line between them in the Nichols chart. Subroutine to `rff`.

Outputs

`nt`	new template vector with elements in Nichols form [deg + j * dB].

Inputs

`t`	pruned and sorted template vector with elements in Nichols form [deg + j * dB].
`dist`	a two elements row vector [phase distance in degrees , gain distance in dB] denoting the maximal distance (2-norm) between two neighbouring template points in the new template

CMPVEC

Purpose

Compares elements of two vectors of different sizes.

Syntax

```
[a index,b index]=cmpvec(a,b,option)
```

Description

compares two vectors of different sizes, returns indices of those elements that have an equal/no equal counterpart in the other vector.

Outputs

`a index`	indeces of those elements in `a` that have an equal/no equal counterpart in `b`.
`b index`	indeces of those elements in `b` that have an equal/no equal counterpart in `a`.

Inputs

`a`	complex row or column vector
`b`	complex row or column vector
`option`	equal/no equal switch `option` = 0: no equal. `option` = 1: equal (default).

CTPL

Purpose

Calculates a template file from a plant definition file.

Syntax

`ctpl(tplf,QPlant,u_method,w_op,option,plot_on)`

Inputs

`tplf`	Template file name, within ' ' and not including '`.tpl`'. NOTE: NO EXTRA SPACES ARE ALLOWED IN ANY NAMES. The template file, which is the output, contains the computed templates and other information.
`QPlant`	Plant definition file name, within `''` and not including `'.m'`. The plant definition file contains the plant definition and data for the template calculation, see `plant.m`. Default: The plant definition file name = template file name.
`u_method`	Template calculation method. Available methods are listed in `plant.m`. The name of the method must be given within `''`. Default: When `u_method==[]` or empty, then `u_method` = the template calculation method in `Qplant`. **WARNING**: If the plant transfer function in `Qplant` is given in polynomial form, then `u_method= rff [deg,dB]` is not a legal choice. Although a template file will be produced, it only contains rubbish. Remark: `plant.m` can be inspected with any editor, or invoked with the command: `plnt newfilename` where newfilename is a new name, without `.m`
`w_op`	Vector of frequencies [rad/s] for which the templates are to be computed. Default: When `w_op==[]` or empty, then `w_op` = `w_tpl` in `Qplant`.
`option`	`'r'` (default) replaces old templates, `'u'` produces the union. This makes it possible to get the union of templates calculated with different methods
`plot_on`	`plot_on`==1, invokes plotting for the Recursive Grid Method.

Example

`ctpl('ex2_1a');`
creates the template file `'ex2_1a.tpl'` using the data in the plant file `'ex2_1a.m'`.

`ctpl('ex2_1b','ex2_1a','aedgrid [1,1]');`
creates the template file `'ex2_1b.tpl'` using the data in the plant file `'ex2_1a.m'`, with the exception that the Recursive Edge Grid template computation method is used.

`ctpl('ex2_1c1','ex2_1a','grid',[0.3 3 30]);`
creates the template file `'ex2_1c1.tpl'` using the data in the plant file `'ex2_1a.m'`,with the exception that the grid template computation method is used for the frequencies [0.3 3 30] rad/s.

`ctpl('ex2_1c2','ex2_1a',[],[0.3 3 30]);`

creates the template file `'ex2_1c2.tpl'` using the data in the plant file `'ex2 1a.m'`, with the exception that templates are computed for the frequencies [0.3 3 30] rad/s.

See also

AGRID, AEDGRID, RFFPZ, RFFCPZ, RFFEL, SHOWTPL, TPL2MAT, MAT2TPL, LOOK, GETFROM, REMOVE, INSERT, CASES.

DEFILE

Purpose

Identifies file type (controller, plant, tpl-file, bnd-file).

Syntax

```
[flag]=defile(file name)
```

Outputs

`flag`	= 0: Unknown.
	= 1: Controller function
	= 2: Plant description file
	= 3: tpl file
	= 4: bnd file
	= -1: an error occur while trying to open the file, or the file does not exist.

Inputs

`file_name`	string, with or without extension.

Remarks

If `file name` is given withour extension, and if there are more than one files having the first name `file_name`, then the length of flag will become greater than one.

E_TABLE

Purpose

Error table.

Syntax

```
[err]=e table(error number)
```

Description

Subroutine to many functions.

EXTRACT

Purpose

Extracts bounds from a contour given by the Matlab command CONTOURC.

Syntax

```
[bnd]=extract(cont);
```

Description

Subroutine to CBND and BNDUPD.

Outputs

`bnd`	extracted bound vector, with elements in Nichols form.

Inputs

`cont`	contour vector produced by the Matlab command CONTOURC.

FBCOMP

Purpose

Feedback function model file, to be copied and edited by the user.

Syntax

`[G] = fbcomp(s)`

Description

Copy the file into the workspace, change its filename, and name in its head above, and edit the file to reflect the current feedback Compensator. `fbcomp` is called by `cdesign` to plot the current nominal open loop frequency function.

Outputs

`G` G is the only output argument. `G` is a complex vector, equalling the feedback compensator frequency function.

Inputs

`s` s is the only input argument. `s` = j*w, where w is the frequency vector [rad/s]

Remarks

The equation of the frequency function may be written in whatever form the user chooses, with the restriction that all vector operations involving `s` be elementwise, i.e. use a point, `.` , in front of arithmetic operators such as `*`, `/`, `^`, etc.

A Real Factored Form of `G` is suggested below. Notice that the parameters whose absolute value is `1/eps` gives factors whose value equals 1.

If `G` is to represent a digital controller, include e.g the following statement `z = exp(s*T)`; where `T` is the sampling period [s], and define `G` as a function of `z`. If appropirate, include the frequency function of a continuous time pre-sampling filter (as a function of s), and the transfer function of a zero order hold:

```
zoh = (1-exp(-T*s))/(Ts);
```

Some of the parameters below are preset to `1/eps` or `-1/eps`, where `eps` is a Matlab variable denoting the machine precision. Such parameters cancel their respective frequency function factors to 1, e.g. `p2` = `-1/eps`; makes the single pole factor in `G` `pole2` = `1./(1-s/p2)` = `1`. If `pole2` is needed in `G`, then the user sets `p2` to the desired pole location, e.g. `p2 = -2.5;` See the code below.

Normally the user who likes working with feedback compensator transfer functions in DC real factored form, will assign appropriate values to those paramaters `k`, `n`, `p1`, ..., `z1`, ... that represent needed feedback compensator factors. If necessary, the user may add more parameters and factors, such as e.g. `p6 = -10.5;` and the code `pole6 = 1./(1-s/p6);` and also change
`G = ... pole5.*zero1 ...;` to
`G = ... pole5.*pole6.*zero1 ...;`

Body

```
[G] = fbcomp(s)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% User's comment: feedback compensator no. for
% plant
%
% DC-gain
% =======
```

```
  k = 1;
%
% Number of integrators
% =====================
  n = 0;
%
% Real Poles
% ==========
  p1 = -1/eps;   p2 = -1/eps;      p3 = -1/eps;
     p4 = -1/eps;      p5 = -1/eps;
%
% Real Zeros
% ==========
  z1 = -1/eps;   z2 = -1/eps;      z3 = -1/eps;
     z4 = -1/eps;      z5 = -1/eps;
%
% Complex Poles
% =============
  zp1 = 0; zp2 = 0;    zp3 = 0;    zp4 = 0;    zp5
= 0;
  wp1 = 1/eps;   wp2 = 1/eps;      wp3 = 1/eps;
     wp4 = 1/eps;      wp5 = 1/eps;
%
% Complex Zeros
% =============
  zz1 = 0; zz2 = 0;    zz3 = 0;    zz4 = 0;    zz5
= 0;
  wz1 = 1/eps;   wz2 = 1/eps;      wz3 = 1/eps;
     wz4 = 1/eps;      wz5 = 1/eps;
%
% -----------------------------------------------
%
pole1 = 1./(1-s/p1);   pole2 = 1./(1-s/p2);
pole3 = 1./(1-s/p3);   pole4 = 1./(1-s/p4);
pole5 = 1./(1-s/p5);

zero1 = 1-s/z1;  zero2 = 1-s/z2; zero3 = 1-s/z3;
zero4 = 1-s/z4;  zero5 = 1-s/z5;

cpole1 = 1./(1 + (2*zp1 + s/wp1).*s/wp1);
cpole2 = 1./(1 + (2*zp2 + s/wp2).*s/wp2);
cpole3 = 1./(1 + (2*zp3 + s/wp3).*s/wp3);
cpole4 = 1./(1 + (2*zp4 + s/wp4).*s/wp4);
cpole5 = 1./(1 + (2*zp5 + s/wp5).*s/wp5);

czero1 = 1 + (2*zz1 + s/wz1).*s/wz1;
czero2 = 1 + (2*zz2 + s/wz2).*s/wz2;
czero3 = 1 + (2*zz3 + s/wz3).*s/wz3;
czero4 = 1 + (2*zz4 + s/wz4).*s/wz4;
czero5 = 1 + (2*zz5 + s/wz5).*s/wz5;

G=(k./s.^n).*pole1.*pole2.*pole3.*pole4.*pole5...
.*zero1.*zero2.*zero3.*zero4.*zero5.*cpole1...
.*cpole2.*cpole3.*cpole4.*cpole5.*czero1...
.*czero2.*czero3.*czero4.*czero5;
```

See also

CDESIGN FDESIGN.

Purpose

Criterion function for 2x2 MIMO bd11 specification.

Syntax

```
function[Smax]=fbd11(tpl nom,tpl,GP,spec,par)
```

Description

Subroutine to MCBND11, MBNDUPD.
Criterion function for
max|(L10/W11)/(1+(L10/W11))|/min|(L10/W11)/(1+(L10/W11))| = V (abs, not dB)
abd |(W12/W11)*b21/(1+L10/W11)|=bd (abs, not dB):

Outputs

`V` value of the servo tolerance criterion, absolute value (not dB).

`bd` value of bd11 candidate, absolute value (not dB).

Inputs

ALL INPUT VARIABLES ARE GIVEN BY THE CALLING BOUND COMPUTATION FCN

`tpl_nom` plant template nominal, Pnom=1/W11nom, scalar in Nichols form, deg+j*dB.

`tpl` plant template matrix for 1/W11, in Nichols form. Matrix with equal columns, where each column equals the template 1/W11.

`GP` open loop nominal candidate for Lnom=L10*Pnom, in Nichols form, deg+j*dB. Matrix with equal rows, where each row equals the candidates for Lnom=L10*Pnom.

`spec` specification scalar, dB.
`spec`=b21

`par` W12/W11 - template, deg + j*dB.
Matrix with equal columns, where each column equals the template W12/W11. Matrix of the same size as `tpl`, where each element corresponds to the same plant parameter combination, see `mtpl11.m`.

Remarks

Author: P-O Gutman, M Nordin.

Purpose

`rsrs` criterion fcn for the 2nd step of cascaded design.

Syntax

```
function[Tmax]=fcasc r(tpl nom,tpl,GP,spec,...
par_nom,par)
```

Description

Subroutine to CBND. Computes the creterion value for an implicit frequency that is not explicitly used in this function and is therefore not explicitly among the input arguments. CBND keeps track of the frequency.

Outputs

`Tmax`	value of tolerance criterion.

Inputs

ALL INPUT VARIABLES ARE GIVEN BY THE CALLING BOUND COMPUTATION FCN.

`tpl_nom`	inner plant template nominal, `P2nom`, scalar in Nichols form, deg+j*dB.
`tpl`	inner plant template `P2`, in complex form.
`GP`	inner open loop nominal candidate, `G*P2nom`, in Nichols form, deg+j*dB.
`spec`	`rsrs` specification vector [upper lower], dB.
`par_nom`	[], not in use
`par`	outer feedback bcompensator * outer template, `G1*P1=L1`, in complex form.

FCASC_S

Purpose

Sensitivity criterion fcn for the 2nd step of cascaded design.

Syntax

```
function[Smax]=fcasc s(tpl nom,tpl,GP,spec,...
par_nom,par)
```

Description

Subroutine to CBND. Computes the creterion value for an implicit frequency that is not explicitly used in this function and is therefore not explicitly among the input arguments. CBND keeps track of the frequency.

Outputs

`Smax`	value of criterion

Inputs

ALL INPUT VARIABLES ARE GIVEN BY THE CALLING BOUND COMPUTATION FCN.

`tpl_nom`	inner plant template nominal, `P2nom`, scalar in Nichols form, deg+j*dB.
`tpl`	inner plant template `P2`, in complex form.
`GP`	inner open loop nominal candidate, `G*P2nom`, in Nichols form, deg+j*dB
`spec`	sensitivity specification value, dB
`par_nom`	[], not in use
`par`	outer feedback compensator * outer template, `G1*P1`, in complex form.

FCOUPLE1

Purpose

Criterion function for first step 2x2 MIMO cross coupling specification.

Syntax

```
function[Smax]=fcouple1(tpl_nom,tpl,GP,spec,par)
```

Description

Subroutine to MCBNDxx, MBNDUPD.
Criterion function for |(W12/W11)*b21/(1+L10/W11)| < bd11.

Outputs

`Smax` value of the criterion, dB.

Inputs

ALL INPUT VARIABLES ARE GIVEN BY THE CALLING BOUND COMPUTATION FCN

`tpl nom` plant template nominal, Pnom=1/W11nom, scalar in Nichols form, deg+j*dB.

`tpl` plant template matrix for 1/W11, in Nichols form. Matrix with equal columns, where each column equals the template 1/W11.

`GP` open loop nominal candidate for Lnom=L10*Pnom, in Nichols form, deg+j*dB. Matrix with equal rows, where each row equals the candidates for Lnom=L10*Pnom.

`spec` specification vector, dB. `spec`(1) = b22; `spec`(2) = b12. (for the frequency in question).

`par` W12/W11 - template, deg + j*dB. Matrix with equal columns, where each column equals the template W12/W11. Matrix of the same size as `tpl`, where each element corresponds to the same plant parameter combination, see `mtpl11.m`.

Remarks

Author: P-O Gutman, M Nordin.

FCOUPLE2

Purpose

Criterion function for second step 2x2 MIMO cross coupling specification.

Syntax

```
function[Smax]=fcouple2(tpl_nom,tpl,GP,spec,...
par nom,par)
```

Description

Subroutine to CBND.
Criterion function for
|(w21/w22e)*(L10/w11)*F11/(1+L10/w11)| | wwc1 |
|--| = |--------------------| <= b21
| 1 + L20/w22e | | 1 + L20/w22e|

Outputs

`Smax` value of the criterion, dB.

Inputs

ALL INPUT VARIABLES ARE GIVEN BY THE CALLING BOUND COMPUTATION FCN

`tpl nom` plant template nominal, Pnom=1/W22enom, scalar in Nichols form, deg+j*dB/

`tpl` plant template matrix for 1/W22e, in Nichols form. Matrix with equal columns, where each column equals the template 1/W22e.

`GP` open loop nominal candidate for Lnom=L20*Pnom, in Nichols form, deg+j*dB. Matrix with equal rows, where each row equals the candidates for Lnom=L20*Pnom.

`spec` specification scalar, dB. `spec` = b21; (for the frequency in question)

`par nom` = [] (not in use)

`par` wwc1 - template, deg + j*dB, where wwc1 = (w21/w22e)*(L10/w11)*F11/(1+L10/w11). Matrix with equal columns, where each column equals the template wwc1. Matrix of the same size as `tpl`, where each element corresponds to the same plant parameter combination, see `mtpl2.m`.

Purpose

Interactive prefilter design in the Bode plot.

Syntax

```
fdesign(Pclosed,Fc,phandle,plot_op,w_nom,col_nom,...
w_tpl,col_tpl)
```

Description

This program shows the transfer function `F*Pclosed` in a Bode plot. Both the nominal case, and the gains and/or phases or their extents of the templates (if existing) are shown.

`Pclosed` should typically be the closed loop template GP/(1+GP), but can be arbitrary.

`fdesign` can also be used to plot templates in Bode plots, or to do nominal open loop design in Bode plots.

Output

`ph`	row vector containing handles to all graphical objects (curves and text) added in the current figure by the current command Inputs.

Inputs

`Pclosed`	Either: (1) a template file, string with extension `.tpl`, e.g. `iotpl.tpl`. (2) a plant file, string with extension `.m`, only the nominal case is used. (3) a controller file, string with extension `.m`. (4) [], (i.e. empty), which means a constant of 0 dB.
`Fc`	a controller file, with or without the extension `.m` or [], which means a constant of 0 dB.
`phandle`	handle or row vector of handles to objects added by previous `fdesign` plots, which are to be deleted in the current figure, before the current object is added. NOTE: trying to delete the same object twice will cause an error! If `phandle=[]`, plotting will take place in the current figure,if `phandle='new'`, a new figure will be opened.
`plot_op`	One of the following: `'mag'` magnitude plot in dB, default. `'phase'` phase plot in deg. `'bode'` a Bode plot, both magnitude and phase. `'mag_all','phase_all','bode_all'` the same as above but plots all template points, rather than only the extreme ones.
`w_nom`	vector for the nominal case; however, if `Pclosed` is a template file, its nominal frequencies will be used.
`col_nom`	color and linestyle of nominal case, default `'b-'`.
`w_tpl`	frequency vector for template display.
`col_tpl`	color and linestyle of template display, default `'ro'`.

Remarks

This command works only with files in the current working directory.

Example

Plot different prefilter*(closed loop) designs together with the specifications:

```
h0=fdesign('ioex1.tpl',[],'new');   % without
```

```
                                               % prefilter.

showspc('ex1','rsrs','freq','r',gcf);
h1=fdesign('ioex1.tpl','f1');% add the
                                  % prefiltered design
                                  % to the plot.

h2=fdesign('ioex1.tpl','f2',[h1]); % new prefilter
                                      % f2,delete the
                                      % old design
                                      % (object h1)
                                      % in the
                                      % current plot.
h3=fdesign('ioex1.tpl','f3',[h0 h2]);
                                      % new prefilter
                                      % f3, new.
                                      % delete the
                                      % old objects
                                      % h0 and h2.
```

Remarks

It is possible to delete an old object with the command delete, e.g.: `delete(h3)`

See also

CDESIGN, SHOWSPC, SHOWTPL, TPLFOP.

FIDSRS

Purpose

Criterion function for input disturbance step response specification.

Syntax

```
[PSmax]=fidsrs(tpl nom,tpl,GP,spec,par nom,par)
```

Description

Subroutine for `cbnd`, `bndupd`.

Criterion function for |P/(1+GP)| < `spec`.

Outputs

`PSmax` value of the criterion.

Inputs

ALL INPUT VARIABLES ARE GIVEN BY THE CALLING BOUND COMPUTATION FCN.

`tpl nom`	plant template nominal, `Pnom`, scalar in Nichols form, deg+j*dB.
`tpl`	plant template `P`, in Nichols form
`GP`	open loop nominal candidate, `Lnom=G*Pnom`, in Nichols form, deg+j*dB.
`spec`	specification value, dB, which equals the row, belonging to the appropriate frequency, of the current specification.
`par_nom`	[], not in use
`par`	[], not in use

Purpose

Criterion fcn for input step response spec/complementary sensitivity.

Syntax

```
[Tmax]=fiosrs(tpl_nom,tpl,GP,spec,par_nom,par)
```

Description

subroutine for CBND, BNDUPD criterion function for spec(2) < |GP/(1+GP)| < spec(1) criterion function also for 1 d-o-f servo specifications.

Outputs

`Tmax`	value of the criterion

Inputs

ALL INPUT VARIABLES ARE GIVEN BY THE CALLING BOUND COMPUTATION FCN.

`tpl nom`	plant template nominal, Pnom, scalar in Nichols form, deg+j*dB.
`tpl`	plant template P, in Nichols form.
`GP`	open loop nominal candidate, Lnom=G*Pnom, in Nichols form, deg+j*dB.
`spec`	specification values = [upper limit, lower limit], dB, which equals the row, belonging to the appropriate frequency, of the current specification.
`par_nom`	[], not in use
`par`	[], not in use

Purpose

Criterion fcn output disturbance step response spec at plant input.

Syntax

```
[PSmax]=fodsrsc(tpl nom,tpl,GP,spec,par nom,par)
```

Description

Subroutine to CBND, BNDUPD Criterion function for |G/(1+GP)| < spec (dB).

Outputs

`PSmax`	value of the criterion function for the n instances of the nominal open loop candidates to be tested that are contained in `GP`. Tmax is a row vector of length n with real elements. The Horowitz bound is the locus of those GP-candidates, for which `Tmax` = 0.

Inputs

ALL INPUT VARIABLES ARE GIVEN BY THE CALLING BOUND COMPUTATION FCN.

`tpl nom`	the nominal plant template point, a scalar in Nichols form.

FODSRS

Purpose

Criterion function for output dist step resp specification or sensitivity.

Syntax

```
function[Smax]=fodsrs(tpl_nom,tpl,GP,spec,...
par nom,par)
```

Description

Subroutine to CBND, BNDUPD.
Criterion function for |1/(1+GP)| < spec(1).

Outputs

`Smax`	value of the criterion.

Inputs

ALL INPUT VARIABLES ARE GIVEN BY THE CALLING BOUND COMPUTATION FCN.

`tpl_nom`	plant template nominal, Pnom, scalar in Nichols form, deg+j*dB.
`tpl`	plant template `P`, in Nichols form.
`GP`	open loop nominal candidate, Lnom=G*Pnom, in Nichols form, deg+j*dB.
`spec`	sensitivity specification value, dB.
`par nom`	[], not in use.
`par`	[], not in use.

FRSRS

Purpose

Criterion fcn for reference step response specification.

Syntax

```
[Tmax]=frsrs(tpl_nom,tpl,GP,spec,par_nom,par)
```

Description

Subroutine to CBND, BNDUPD.
Criterion fcn for:
20*log10(max|GP/(1+GP)|/max|GP/(1+GP)|)<spec(1)-spec(2) [dB].

Outputs

`Tmax`	value of the criterion function for the n instances of the nominal open loop candidates to be tested that are contained in `GP`. `Tmax` is a row vector of length n with real elements. The Horowitz bound is the locus of those GP-candidates, for which `Tmax` = 0.

Inputs

ALL INPUT VARIABLES ARE GIVEN BY THE CALLING BOUND COMPUTATION FCN .

`tpl_nom`	the nominal plant template point, a scalar in Nichols form [deg + j*dB].
`tpl`	a m*n plant template matrix where each of the n columns contains the same template, i.e. n identical columns (of length m) are stacked side by side. Each element is in Nichols form [deg + j*dB]. (This means each of the m rows have equal elements. This is done to simplify matrix computation, the input variables `tpl` and `GP` have the same dimension.)
`GP`	A m*n matrix where each column is constant, and each row contains the n different nominal open loop candidates [deg + j*dB].
`spec`	the specification vector = [upper spec dB, lower spec dB, frequency rad/s] corresponding to the frequency for which bounds are currently computed).
`par_nom`	`[]`, not in use.
`par`	`[]`, not in use.

FUSER

Purpose

Model file to create user defined specifications .

Syntax

```
[Tmax]=fuser(tpl nom,tpl,Lnom,spec,par nom,par)
```

Description

User supplied subroutine to CBND, and BNDUPD.
Create your own criterion function by copying/editing this file.

Outputs

`Tmax` value of the criterion function for the n instances of the nominal open loop candidates to be tested that are contained in `Lnom`. `Tmax` is a row vector of length n with real elements. The Horowitz bound is the locus of those Lnom-candidates, for which `Tmax` = 0.

Inputs

ALL INPUT VARIABLES ARE GIVEN BY THE CALLING BOUND COMPUTATION FCN.

`tpl_nom` the nominal plant template point, a scalar in Nichols form [deg + j*dB].

`tpl` a m*n matrix where each of the n columns contains the same template, i.e. n identical columns (of length m) are stacked side by side. Each element is in Nichols form [deg + j*dB]. (This means each of the m rows have equal elements This is done to simplify matrix computation, the input variables `tpl` and `Lnom` has the same dimension.) `tpl` contains the plant template.

`Lnom` A m*n matrix where each column is constant, and each row contains the n different nominal open loop candidates [deg + j*dB].

`spec` e specification vector (or scalar), e.g. in [dB]. It equals the elements 2, 3, ... in one row of the specification variable specname that the user used as an input in her CBND or BNDUPD command. (The row corresponds to the frequency for which bounds are currently computed).

`par_nom` the nominal parameters of the plant (if existing).

`par` the parameter matrix that produced this template (if existing).

Body

```
% Author: M Nordin, P-O Gutman
% Copyright P-O Gutman 1996



if nargin==0;
     disp('Tmax)=fuser(tpl_nom,tpl,Lnom,spec,par_n
om,par)');
     break;
end;

%%% WRITE YOUR OWN SPECIFICATION BELOW....
```

Examples

Due to the matrix form of `tpl` and `Lnom` it is now easy to form for instance the sensitivity criterion by

```
L=n2c(Lnom+tpl-tpl nom);  %the open loop in
% Nyquist format. Each column contains
%the template multiplied by one of the feedback
%compensator candidates S=1./(1+L);
% The sensitivity in Nyquist format

Tmax=20*log10(max(abs((S))); %The maximum of the
% sensitivity in Nichols form Note that max
% operates column wise.

Tmax=Tmax-spec(1);        % It is assumed that
% spec(1) contains the sensitivity spec in dB.
% this is the case if it was generated by the
% command odsrs Now the specification holds for
% the k:th value of Lnom, iff Tmax(k) <= 0

*****
```

Stiffness coefficient `c0`=0, `c1<=c1spec`. Assume that the user defined the second column of the relevant specification variable specname as the low frequency asymptote [dB] above which `G` must lie.

```
G = min(imag(Lnom - tpl_nom));  % n feedback
% compensator candidates, row vector [deg+j*dB]

Tmax = G - spec(1);       % Now the specification
% holds for the k:th value of Lnom, iff Tmax <= 0
```

OTHER EXAMPLES: The user is advised to study the predefined criteria functions FRSRS, FIDRS, FODRS, FODSRSC, FIOSRS

FVALUE

Purpose

Copies template from tpl-file/freq fcn value from controller file.

Syntax

`fvalue(file name,w,flag ,value)`

Outputs

`t`	the template variable, in complex form, from the template file for the frequency `w`, or the frequency fcn value, in complex form, for the frequency `w` from the controller file, or `t` = `value` (the third input argument), see below.

Inputs

`file name`	file name, string with or without extension. See remark.
`w`	frequency, rad/s.
`flag_`	file type/output selector. flag_ = 0: t = value; flag_ = 1: file_name is the name of a controller file flag_ = 2: file_name is the name of a template file flag_ = []: file type is determined through a call to the function `defile`, see remark.
`value`	complex variable

Remarks

If `flag` =[] and `file name` contains no extension, then the function DEFILE searches, in the current directory, for file candidates in the following order of priority:[controller file, template file].

DEFILE output = 1:   controller file
DEFILE output = 2:   template file

See also

DEFILE.

Purpose

Copies a bound from a bound file.

Syntax

```
bound=getbnd(bondfile,boundname,w);
```

Outputs

`bound`	bound vector, with elements in Nichols form, bound frequency vector, depending on input `w`.

Inputs

`boundfile`	Boundfile name, string with or without extension.
`boundname`	the name of the bound, string, e.g.. `'odsrs'`, `'rsrs'....`
`w`	the frequency [rad/s] of the bound, or `w=='w'`, which then gives the bound frequency vector.

Example

```
rsrs7 = getbnd('ex2_1a','rsrs', 7);
% copy the rsrs bound for 7 rad/s from ex2_1a.bnd
% into the vector7.
```

See also

GETFROM ADD2BND REMOVE.

GETFROM

Purpose

Lists and/or copies variables from a mat-file.

Syntax

`[res1,res2,res3...]=getfrom(Matfil,var1,var2,var3...)`

Outputs

`res1, ...`	names of variables in workspace.

Inputs

`Matfil`	full name of mat-file within ''.
`var1, ...`	names of variables (within '') to be copied from Matfil.

Remarks

tpl-, spc-, and bnd-files are mat-files from which variables may be copied.

At most nine variables can be copied in one go.

If neither output arguments nor input arguments beyond Matfil are given, then all variable names inside Matfil are listed.

If one or more output variable names are given, but no input input variable names, then all variable names within Matfil are listed, and the user is prompted to select the variables she wants to have copied. Note that in this case the variable names must not be within ''.

If input variable names as well as output variable names are given, then their number must be the same.

No warning is given if an input variable name does not exist. The corresponding output variable will become empty.

Examples

```
getfrom('ex2_1a.tpl');
 % lists the variables in ex2_1a.tpl

[tw1,tw2]=getfrom('ex2_1a.tpl');
% lists the variables in ex2_1a.tpl, and
% prompts you to name (without '') the
% names of two variables to be copied.

[tw1,tw2]=getfrom('ex2_1a.tpl','t_w1','t_w2');
% copies the variable t_w1 and t_w2 from
% 'ex2_1a.tpl' to tw1, and tw2, respectively.
```

See also

LOOK REMOVE INSERT.

GETTPL

Purpose

Gets a variable from a templatefile.

Syntax

```
[tpl,par]=gettpl(tplfile,w);
```

Outputs

`tpl`	if the input `w` is a frequency, then `tpl` is the requested template row vector, otherwise see the options of the input argument `w`.
`par`	parameter matrix whose columns are the parameter vector belonging to each template point (if it exists). Otherwise empty. See also the options of the input argument `w`.
`parnom`	the nominal parameter vector (if it exists) when `w` = `'nom'`, otherwise empty.

Inputs

`tplfile`	The template file name, string with or without extension
`w`	either the frequency [rad/s] of the template to be copied, or one of the following options: `'w_tpl'` the output tpl will become a column vector containing the template frequencies, while the other outputs are empty. `'nom'` the output tpl becomes the nominal frequency vector, the output par becomes the nominal frequency response, and the output parnom becomes the nominal parameter vector.

Examples

```
[tpl,par]=gettpl('ex1',5)
```

gets the template, and corresponding parameters for 5 [rad/s] from the template file `'ex1.tpl'` .

```
[w_tpl]=gettpl('ex1','w_tpl');
```

gets the template frequency row vector

```
[w_nom,nom,par_nom]=gettpl('ex1','nom');
```

gets the nominal frequency, the nominal frequency vresponse, and the nominal parameters.

HNGRID

Purpose

Draws L/(1+L) or 1/(1+L) loci in a Nichols diagram.

Syntax

`hngrid(mm,pp,op)`

Inputs

`mm`	vector of desired magnitude loci [dB]. Default: When `mm` is non-existant, or `mm`==[], the default is displayed (try it!)
`pp`	vector of desired phase loci [deg]. Default: When `pp` is non-existant, or `pp`==[], the default is displayed (try it!)
`op`	Loci selector. `op`=0: L/(1+L) loci `op` nonzero: 1/(1+L) loci Default: `op`=0

Examples

hngrid
% displays the default L/(1+L) loci

hngrid([],[],1)
% displays the default 1/(1+L) loci

hngrid([6],[-90 90])
% displays the 6 dB, and the -90 and 90 degree L/(1+L) loci

See also

MGRID.

HZOOM

Purpose

Puts a zoom menu in the current figure.

Syntax

`hzoom`

Remarks

Warning: user, do not use this command with input arguments!

IDSRS

Purpose

Input Disturbance Step Response Specification.

Syntax

```
[spec w,spec t,tab]=idsrs(spcfile,specname,Ts,...
Max,M,td,w,tf,zmin,plt,dt,n)
```

Description

Transfers time-domain input disturbance step specifications to the frequency domain, using simulations of a 2:nd order system c*s/(s^2+2*z*w0*s+w0^2) and gridding over its parameters. This is the input disturbance closed loop P/(1+PG) transfer function obtained when the plant is the first order system c/(s+a) and the controller is a pure integrator k/s.

Outputs

`spec w`	Two-column matrix, where the first column contains the frequency vector and the second column contains the upper specification in the frequency domain.
`spec t`	Three-column matrix, where the first column contains the time vector, the second column contains the upper envelope of the step responses and the third column contains the lower envelope.
`tab`	Three-column matrix, where the first column contains the time vector corresponding to the time-domain specifications. The second column contains the upper limit corresponding to the time-domain specifications and the third column contains the corresponding lower limit.

Inputs

`spcfile`	the name of the .spc-file to store the specification in. String,without extension.
`specname`	the name under which the specifications are stored, string without '_w' or '_t', Default is `'idsrs'`
`Ts`	[Settling time,upper deviation (percent),lower deviation (percent)]. The default value for the lower deviation is the upper deviation and the default value for the upper deviation is 5%.
`Max`	the maximum allowed value (absolute value, not dB, not percent) of the step response.
`M`	the undershoot given in percent of Max.
`td`	Minimum rise time. Default value ts/5. The step responces will be below the line from the origin to (`td`,`Max`). This imposes a limitation of the derivative at t=0.
`w`	frequency vector. The default value is: 'logspace(log10(2*pi/(Final-time)/20),log10(20*pi/(Minimum rise-time)*2))'.
`tf`	final time. The default value is 2*Settling-time.
`zmin`	minimum damping of the poles, default is `zmin`=0.01.
`plt`	plot option. Logical variable for switching on and off plotting information of step responses. Can be 0 or 1. Default is 1 (=plotting).
`dt`	time increment for the simulation; The default value is 'Final-time/100'.

`n`	number of grid-points in the space of parameters of the simulated system. The default value is n=[40,40] for the 2:nd order system, n=[10,10,10,10] for the 3:rd order system and n=[10,10,10,3] for the alternative gridding. The number of grid-points becomes 'prod(n)'.

Setting any input variable to [] gives the default value of that parameter.

Example

```
idsrs('platform',[],[0.1 1], 0.5, 10);
```

Insert into the specification file `platform.spc` a specification, called idsrs, that is generated by an input disturbance step response that settles within 0.1 second within 1%, and whose maximum is <= 0.5, and whose undershoot is less than 10% of that, i.e. 0.05.

See also

RSRS, ODSRS, SPC_ID2.

INSERT

Purpose
Inserts/replaces variable in mat-file.

Syntax
`insert(matf,value,name,op)`

Description
Inserts variable into mat-file, or replaces variable value in a mat-file.

Inputs

`matf`	full name of mat-file within ' ', including extension.
`value`	variable to be inserted.
`name`	variable name (within '') in matf, into which value is to be inserted.
`op`	replace/not replace switch.
	`op` not given: an existing variable (name in matf) will not be replaced. If name does not exist, then it will be created in matf with the assigned value. (default)
	`op=='r'` an existing variable (name in matf) will be replaced. If name does not exist, then it will be created in matf with the assigned value.

Remarks
The command always leaves a dummyvariable called 'filename' containing the filename in the file.

Examples
`insert('ex0.tpl',tw1,'t_w1')`
if `t_w1` does not exist in `ex0.tl`, then it will be created with `t_w1=tw1`, else `t_w1` will retain its previous value.

`insert('ex0.tpl',tw1,'t_w1', 's')`
this misprint will have the same effect as
`insert('ex0.tpl',tw1,'t_w1')`

`insert('ex0.tpl',tw1,'t_w10','r')`
if `t_w10` does not exist in `ex0.tl`, then it will be created with `t_w10=tw1`, else `t_w10=tw1`

`insert('ex0.tpl',[1 2],'t_w10','r')`
if `t_w10` does not exist in `ex0.tl`, then it will be created with `t_w10`=[1 2], else `t_w10`=[1 2]

`insert('ex0.tpl','bla','t_w10','r')`
if `t_w10` does not exist in `ex0.tl`, then it will be created with `t_w10='bla'`, else `t_w10='bla'`

See also
LOOK GETFROM REMOVE.

ISIN

Purpose

Checks if a variable exists in a mat-file.

Syntax

`[flag]=isin(matf,name)`

Description

ISIN returns 1 if the variable name 'name' exists in the mat file 'matf', otherwise 0. If the mat file does not exist then the output is -1

Inputs

`matf`	mat-file name, string with extension.
`name`	string.

LOAD

Purpose

Retrieve variables from disk (Matlab command).

Description

`LOAD fname` retrieves the variables from the MAT-file name.mat'.
`LOAD`, by itself, loads from the file named 'matlab.mat'.
`LOAD xxx.yyy` reads the ASCII file xxx.yyy, which must contain a rectangular array of numeric data, arranged in m lines with n values in each line. The result is an m-by-n matrix named xxx.

To load an ASCII file that does not have a filename extension, use `LOAD fname -ascii`. Otherwise MATLAB adds the extension '.mat' and tries to load it as a MAT-file. To load a MAT-file that does NOT have a '.mat' extension, use `LOAD fname.ext -mat`.

If fname is `"stdio"`, `LOAD` reads from standard input.

See also

SAVE, SPCONVERT, FSCANF, FPRINTF.

.

LOOK

Purpose

Lists the names of the variables stored in a mat-file.

Syntax

```
look(matfilname)
```

Inputs

`matfilename`

the full name of the mat-file, string variable. Note that tpl-files, spc-files, and bnd-files are mat-files whose contents can be inspected with look.

Examples

```
look('exercise.mat')

look('ex2_1a.tpl')
```

See also

GETFROM REMOVE INSERT.

MAKEBND

Purpose

Calculates bounds for a general specification (subroutine to CBND).

Syntax

```
function[bound]=makebnd(tpl_nom,tpl,specfun,spec,...
GPdeg,GPdB,par_nom,par,case);
```

Description

Calculates bounds for a general specification, over a grid in the Nichols plane.

The algorithm is based on brute force gridding of the Nichols chart and the contour algorithm, to extract the bounds. If the problem is large (many points in the template and/or dense gridding) the program solves a sequence of smaller problems.

Inputs

`tpl_nom`	nominal plant, scalar in Nichols form.
`tpl`	template, (value set), vector in Nichols form.
`specfun`	name of criterion function, example: `specfun='frsrs'`.
`GPdeg`	degree grid, default is -360:20:0.
`GPdB`	dB grid, default is -50:5:50.
`spec`	vector containing the specification for the frequency in question.
`par_nom,par`	optional parameters for the criterion function, typically the uncertain physical parameters of the templates. Also useful for SIMO, MISO and MIMO systems.
`case`	= `'siso'` standard SISO case (including Cascaded SISO, and some MIMO bound computations) (default); = `'mimo2'` the cross coupling bound computation in the second step of the standard servo 2x2 MIMO bound computation. In this case, `par` = the (w21/w22e)*(L10/w11)*F11/(1+L10/w11)-template

Remarks

Basic subroutine for bound computation. Computes bound for one frequency.

MAKESPC

Purpose

Graphical design of specification.

Syntax

```
[vf]=makespc(specfile,specname,Vtype,naxis)
```

Outputs

`vf` Specification matrix.
The format of `vf` is columnwise:
for `vf` with one specification column (e.g. sensitivity specification), [frequency (rad/s), specification (db)];
for `vf` with two specification columns
e.g. servo specification),
[frequency (rad/s), upper_spec (db), lower_spec (db)].

Inputs

`specfile` the specification file name (string with or without extension '.spc')

`specname` the name of the specification, string

`Vtype` The number of specification columns in `vf`. This number must be 1 or 2.

`naxis` axis of the Bode plot in format [w-min w_max dB_min dBmax]. Default is [0.01 100 -50 50];

Remarks

The specification matrix is entered by mouse clicking in a Bode plot.

Example

```
makespc('ex5 2a','errcoeff',1,[0.1 100 -60 40]);
```

Inserts into the specification file `ex5 1a.spc` a frequency domain specification called `errcoeff` that has one specification column only, to be defined graphically in a Bode plot with axes
[0.1 100] rad/s and [-60 40] dB.

See also

SPCUPD GETFROM INSERT SHOWSPC RSRS ODSRS IDSRS MSPC.

MAT2TPL

Purpose

Converts a matrix to a template file on Qsyn tpl-format.

Syntax

```
mat2tpl(tplf,mat,user comment)
```

Inputs

`tplf` — output template file name, without .tpl, but within ''. The variables of tplf have standard Qsyn template file variable names. If a template file with the same name already exists, then it is overwritten.

`mat` — matrix of the form [`wtpl nom templ`], with

- `wtpl` column vector of template frequencies [rad/s], in a non-decreasing sequence;
- `nom` column vector of nominal frequency function, where `nom`(i) is the nominal for `wtpl`(i), on the form (deg + j*dB);
- `templ` template matrix, where row number i is the template for `wtpl`(i) padded with NaN such that all rows have the same number of elements as the longest template, and each non-NaN element is on the form deg + j*dB.

`user comment` — user's comment, within '', which becomes the variable `user comment` in `tplf`.

Remarks

NaN is an allowed element in nom, but not advisable.

If `wtpl`(i)==`wtpl`(i+1)==...==`wtpl`(i+m), then `nom`(i+1), ..., `nom`(i+m) are ignored and the template for `w_tpl`(i) in `tplf`, `t_wi`, becomes the union of `templ`(i,:), `templ`(i+1,:), ...,`templ`(i+m,:). Note that subsequent indeces are shifted, i.e. `w_tpl`(i+1) = `wtpl`(i+m+1), etc

The templates in `tplf`, `t_w1`, `t_w2`, ... do not contain any elements equal to NaN.

Note that in `tplf, w nom=w tpl` which equals `wtpl` with adjacent multiplets removed.

To include a denser nominal frequency vector, `wnomdens`, with its frequency function, `nomdens`, one may use the command INSERT. Make sure, however, that `wnomdens` includes `w tpl` - otherwise some commands acting on templates, like SHOWTPL, will not work.

Note that this command may be used to create Qsyn templates from measured data, see the command MFFD

Examples

```
mat2tpl('ex000',m,'data 960103')

insert('ex000.tpl',...
sort([m(:,1)' logspace(-1,2)]),'w nom','r');
```

replaces `w_nom` in `ex000.tpl` with a dense nominal frequency vector that includes `w tpl`.

```
insert('ex000.tpl',...)

c2n([10./(j*sort([m(:,...
1)' logspace(-1,2)])+1)]), 'nom','r');
```

replaces nom in ex000.tpl with a dense nominal frequency function defined for the above frequency vector `w nom`.

See also

MFFD TPL2MAT INSERT ADD2TPL GETTPL.

MCBND11

Purpose

Calculate bounds from given templates and specifications, first step of 2x2 MIMO.

Syntax

```
mcbnd11(bndfile,specname1,specname2,w,plot_off,...
specfile,tplfile1,tplfile2,bndname,specfun1,Tacc,.
..
dBlimit)
```

Inputs

`bndfile` File that shall contain the bounds, default is `bndfile=tplfile`. NO EXTENSION ALLOWED, `.bnd` added automatically

`specname1`, `specname2`
The specification names, within `' '`, allowed names are:
`specname1`= 1,1-servo specification name,
`specname2` = name of b21 spec for the first step servo/bd11 bounds,or
`specname1` = name of b12 spec,
`specname2` = name of b22-spec for the first step cross-coupling bound.

`w` the frequencies for which to calculate bounds, must be a subset of the template frequencies. default is
all of them.

`plot_off` flag to supress (`plot_off=1`) plotting of
calculated bounds. Default is to plot.

`specfile` name of file that contains the specifications, default
is `specfile=bndfile`; String variable. NO EXTENSION ALLOWED, `.spc` added
automatically.

`tplfile` name of file that contains the template 1/W11.
String.
NO EXTENSION ALLOWED, `.tpl` added automatically.

`tplfile2` name of file that contains the template W12/W11. String.NO EXTENSION ALLOWED, `.tpl` added automatically.

`bndname` the name of the new bound, default is `bndname='servo1'` if first step servo
bounds/bd11 bounds are calculated, and `'couple1'` if
first step cross-coupling bounds are calculated.

`specfun1` The name of the specification function required to calculate the bound. Defaults: `'fbd11'`, for the servo/bd11 bound calculation case (`case=1`); `'fcouple1'` for the cross coupling bounds calculation case (`case=2`).

`Tacc` The required accuracy in [deg dB], default is [3 1]

`dBlimit` Decides the search area in which to find the bounds, default is [-50 50] dB.

Remarks

MCBND11 recognizes that the first step servo bound and bd11 bound is to be calculated, by the fact that the specification `specname1` contains 3 columns and `specname2` contains 2 columns.

MCBND11 recognizes that the first step cross coupling bound is to be calculated, by the fact that the specification `specname1` (= b12) contains 2 columns and specname2 contains 3 columns.

All other combinations of number of columns will be considered illegal.

In the servo/bd11 bound computation case, the output bound file `[bndfile,'.bnd']` will also get, for each frequency, a `bd11` matrix, computed during the initial coarse bound computation step. The value bearing elements of the `bd11` matrix have absolute units (not dB). bd11 is saved in `table2.m`-format:
`bd11 = [0,GPdeg(:)';GPdB(:), [values of bd11]];`

The name of the `bd11` matrix will be analog to its bound, i.e. `[bndname,'bd11']`, onto which `_index` is, as usual, appended in the `add2bnd` command that stores the `bd11` matrix.

Purpose

Makes template file from a freq vector, nominal and template matrix.

Syntax

`mffd(tplf,freq,nom,t,comment)`

Inputs

`tplf`	output template file name, without .tpl, but within ' '. The variables of `tplf` have standard Qsyn template file variable names. If a template file with the same name already exists, then the user is prompted to accept overwriting.
`freq`	vector of template frequencies [rad/s], in a non-decreasing sequence.
`nom`	vector of nominal frequency function, where `nom`(i) is the nominal for `freq`(i), on the form (deg + j*dB);
`m`	template matrix, where row number i is the template for `freq`(i) padded with NaN such that all rows have the same number of elements as the longest template, and each non-NaN element is on the form deg + j*dB.
`comment`	user's comment, within ' ', which becomes the variable `user_comment` in `tplf`.

Remarks

This function is a user friendlier shell for the command MAT2TPL.

Example

```
» mffd('ex0',frequency,nominal,template,'data
951212')
Are you sure you want to replace the existing
file?
Hit any key to confirm or ^c to abort
```

replaces the template file `'ex0.tpl'` with a new one, with `w_nom=w_tpl`=frequency,`nom`=nominal, `t w1, t w2, ...` = the rows of template with NaN removed, and `user_comment` = `'data 951212'`.

See also

For more details, see reference guide for MAT2TPL or issue the command `help mat2tpl`.

MGRID

Purpose

Draws user defined grid lines in the current figure window.

Syntax

```
mgrid(nx,ny,add x,add y,label)
```

Inputs

`nx`	the number of segments the current x-axis will be divided into The number of grid lines perpendicular to the x-axis will be `nx`-1.
`ny`	the number of segments the current y-axis will be divided into. The number of grid lines perpendicular to the y-axis will be `ny`-1.
`add_x`	vector of x-values at which grid lines are to be drawn.
`add_y`	vector of y-values at which grid lines are to be drawn.
`label`	tick label selector. If `label`<=0 or non-existant, then the x- and y- values of the new grid lines are not written (default). If `label`>0, then then the x- and y-values of the new grid lines are written; note, however, that the regular Matlab tick labels are not erased.
`fig`	line type and color selector, within ''. `fig` may be one the line type and color combinations defined in the Matlab function plot (`help plot`, for assistance). Default: `':k'`, i.e. black dotted grid

Remarks

An input parameter set to [] is ignored.

If `nx` is assigned, then `ny` must also be assigned, but one or both may be set to []. The other input parameters are optional.

`mgrid` without input parameters draws the regular Matlab grid.

Examples

```
mgrid(5,1)
mgrid(5,[])
```

each of these commands will divide the x-axis into 5 segments (4 grid lines), and no grid lines perpendicular to the y-axis.

```
mgrid(5,2,[ ],20)
```

divides the x-axis into 5 segments (4 grid lines), and the y-axis into 2 segments (1 grid line), and in addition draws a grid line at y=20, perpendicular to the y-axis.

```
hngrid, mgrid(12,10,[],[],[],'-y')
```

gives a Nichols chart with the default closed loop grid in grey (see hngrid), and an open loop grid in yellow for [0 -30 -60 ... -360] deg, and [-60 -50 ... 40] dB.

See also

HNGRID PLOT.

MSPC

Purpose

Auxiliary function to MAKESPC.

Syntax

```
[vf] = mspc(cmnd)
```

Description

The specification matrix is entered by mouse clicking on an existing plot.

Outputs

`vf`	Specification matrix.

Inputs

`cmnd`	The number of specification columns in the desired specification matrix. This number must be 1 or 2.

MULT

Purpose

Unstructured uncertainty function for all template comp methods except rff.

Syntax

```
y=~mult(Qbox,s))
```

Description

Subroutine to CTPL.

Outputs

`y`	edge of multiplicative uncertainty template, `y`= 1+r.*exp(i*fi);

Inputs

`Qbox`	parameter matrix, such that fi=`Qbox`(1,:); and r=`Qbox`(2,:);
`s`	j*w, w = frequency, rad/s.

N2C

Purpose

Converts a matrix from Nichols form to complex form.

Syntax

```
[y]=n2c(z);
```

Outputs

`y`	resulting matrix in complex form, i.e. each element is given as a + j*b.

Inputs

`x`	original matrix in Nichols form, i.e. each element is given as deg + j*dB.

ODSRS

Purpose

Computation of output disturbance step response specification.

Syntax

```
[spec_w,spec_t,tab]=odsrs(spcfile,specname,Tr,...
M,Ts,Td,w,ordr,Ks,tf,plt,dt,n)
```

Description

Transfers time-domain output disturbance step specifications to the frequency domain, using simulations of a 2:nd or 3:rd order system and gridding over its parameters.

Outputs

`spec_w`	two-column matrix, where the first column contains the frequency vector (rad/s), the second column contains the upper bound (dB).
`spec_t`	two-column matrix, where the first column contains the time vector (sec), the second column contains the upper envelope of the step responses and the third column contains the lower envelope.
`tab`	Three-column matrix, where the first column contains the time vector corresponding to the time-domain specifications. The second column contains the upper bound corresponding to the time-domain specifications and the third column contains the corresponding lower bound.

Inputs

`spcfile`	the file in which the spec is stored
`specname`	the name under which the `spec` is stored, default is `'odsrs'`
`Tr`	[Maximum rise-time, Minimum rise-time, level (percent)]; Default level is 10%.
`M`	undershoot (percent). The default value is `M`=10%.
`Ts`	[Settling time, upper deviation (percent),lower deviation (percent)]. The default value for the lower deviation is the upper deviation and the default value for the upper deviation is 5%. The default value for the Settling time is 5*Rise-time.
`Td`	[Delay-time,level (percent)]. Default value for the level is 90% and the default value for the delay time is the upper rise-time.
`w`	frequency vector. The default value is 'logspace(log10(2*pi/(Final-time)/2),log10(2*pi/(Lower rise-time)*2))'.
`ordr`	system order. Can be 2, 3 or 3.1. The default value is 2. If `ordr` is set to 3.1 an alternative gridding for the 3:rd order system is performed.
`Ks`	Initial gain. The default value is `Ks`=1.
`tf`	final time. The default value is 2*Settling-time.
`plt`	plot option. Logic variable for switching on and off plotting information of step responses. Can be 0 or 1. Default is 1.
`dt`	time increment; The default value is 'Final-time/100'.
`n`	number of grid-points. The default value is `n`=[40,40] for the 2:nd order system, `n`=[10,10,10,10] for the 3:rd order system and `n`=[10,10,10,3] for the alternative gridding. The number of grid-points becomes 'prod(`n`)'. See `spc_od2`, `spc_od3`, `spc_od31` for information about the parametrization.

Setting any input variable to [] gives the default value of that parameter.

Example

```
odsrs('ex5_1','odsrs',[0.3 0.1 50],50,1.5,...
[],logspace(-1,2),2);
```

Inserts into the specification file `ex5_1.spc` a specification called odsrs, based on simulated output disturbance step responses that have a rise time of between 0.1 0.3 seconds to the level 50%, a maximum undershoot of 50%, a settling time of 1.5 seconds to within 5%. (% refer to % of disturbance step amplitude).
The frequency domain specification is defined for the frequencies logspace(-1,2). The simulation system order of 2 is used.

See also

RSRS, ODSRS, IDSRS, SPC_OD2, SPC_OD3, SPC_OD31.

PARGRID

Purpose

Creates vectors with all combinations of the elements of the input vectors.

Syntax

```
[a1,a2,...,an]=pargrid(a1,a2,...,an);
              or
[p out]=pargrid(p in) in matrix form
```

Description

Subroutine to PGRID, etc.
Like meshdom for n vectors. The outputs are row vectors.

Outputs

`p_out` The columns of `p_out` contain all combinations of elements from the rows of `p in`. `p out` has as many rows as `p_in`.

`a1,a2,...,an`
correspond to the rows of `p_out`.

Inputs

`p_in` matrix, such that `p_out`(k,i) holds an element from `p_in`(k,:).

`a1,a2,...,an`
correspond the rows of `p_out`.

Remarks

The function call is implemented for a maximum of nine vectors. Change the first line above if you need more than nine, or use the matrix form.

Example

```
p=pargrid([2 3;20 30; 300 200])
p =
     2     2     2     2     3     3     3     3
     30    30    20    20    30    30    20    20
     200   300   200   300   200   300   200   300

p1=[2 3]; p2=[20 30]; p3=[300 200];
[a1,a2,a3]=pargrid(p1,p2,p3)
a1 =
     2     2     2     2     3     3     3     3
a2 =
     30    30    20    20    30    30    20    20
a3 =
     200   300   200   300   200   300   200   300
```

See also

PGRID.

PGRID

Purpose

Grids a parameter vector elementwise and produces all combinations.

Syntax

```
[p]=pgrid(Par,dm,rnd_flag)
```

Description

Subroutine to CTPL and other commands.

Outputs

`p`	matrix where each column contains a parameter value combination. `p` contains as many rows as `Par`, and the value of `p`(k,i) is taken from the parameter interval [min(`Par`(k,1)), max(`Par`(k,2))] according to `dm` and `rnd_flag`.

Inputs

`Par`	matrix with 2 columns, where `Par`(k,1) and `Par`(k,2) are the endpoints of the parameter interval number `k`.
`dm`	scalar, or vector with as many elements as there are rows in `Par`. (`dm`(k)- 1) denotes into how many parts the interval [min(`Par`(k,1)), max(`Par`(k,2))] should be divided, i.e. `dm`(k) denotes how many values should be selected from parameter interval number `k`.
`rnd_flag`	= 1, random gridding of each parameter range = 0, equidistant gridding of each parameter range . (default)

Remarks

Example

```
p=pgrid([2 3;20 30; 300 200],2)
p =
     2     2     2     2     3     3     3     3
     30    30    20    20    30    30    20    20
     200   300   200   300   200   300   200   300
```

PLANT

Purpose

Model of plant description file to be copied and edited by the user.

Syntax

```
plant      % evaluate the output variables
```

```
plnt new_plant
```

If `new plant.m` does not exist in your current directory:
invokes default editor with this file as a model. The file will be saved in your current directory under the name `new_plant.m`
If `new plant.m` exists in your current directory:
invokes default editor with `new_plant.m`.

```
edit plant
```

invokes default editor. Do not forget to save the edited file under a new name in your current directory.

Body

```
--------------------------------------------------
Plant_name :  Plant description


% Definition of the parameters
% ====================
Par = [
       'p1=[p1min,p1max,p1nom,# of cases]',...
        % uncertain parameters
       'p2=[p2min,p2max,p2nom]', ...
       'p3=[p3min,p3max,p3nom]', ...
       'c1=[c1value]',...    % constant parameters
       'c2=[c2value]',...
         ];

% Multiplicative unstructured uncertainty:
% uncertainty circle radius m(w), in [0,1), as a %
% function of frequency w [rad/s]
% ============
Uns_Par=[];      % Uns_Par is either
                 % (1) empty --> no unstructured
                 % uncertainty;
                 % (2) one real number, m, in the
                 % range [0,1) --> m(w) = m for
                 % all w;
                 % (3) a matrix with two rows and
                 % at least two columns, where the
                 % upper row contains the
                 % frequencies w [rad/s], and the
                 % lower row m(w), in [0,1), the
                 % unstructured uncertainties,
                 % --> m(w_tpl) is computed by
                 % linear interpolation with
                 % respect to the logarithmic
                 % frequency scale.


% Definition of the frequency vectors [rad/sec]
% ====================================
w_tpl=[ wmin ... wmax];% Template frequency
                       % vector.

w nom=[ wmin ... wmax];% Nominal frequency vector.
                       % w_nom will automatically
                       % include all points in
                       % w_tpl.

% Definition of the template computation method
```

```
% ==============================================
method = 'rff_[1,1]';
                 % (1) grid_dist   = Grid.
                 % (2) rgrid dist  = Random Grid.
                 % (3) adgrid_dist = Recursive
                 % Grid.
                 % (4) aedgrid_dist= Recursive
                 % Edge Grid.
                 % (5) rff dist    = Real Factored
                 % Form.

% dist=[Max phase_distance [deg], max
% magnitude distance [dB]]; maximum
% resolution of template computation, 2-norm, in
% the Nichols chart for adgrid, aedgrid, rff
% methods grid, rgrid: resolution is given by
% parameter grid, while dist gives the
% Unstructured Uncertainty resolution only, if Uns
% Uncertainty is present.In RFF, the Max phase
% distance must be chosen such that 360 is a
% multiple integer of dist(1). Example:
% dist(1) = 1,2,5,8,0.8,4/3,etc are OK but not
% 7,0.33, etc.
% WARNING: IF RFF STRUCTURE IS USED BELOW, THEN
% method MUST BE 'rff', EVEN THOUGH OTHER METHODS
% MAY BE REQUESTED IN THE ctpl COMMAND.

% Plant definition
% ================

% Polynomial Structure
% ================
P num ='p1';          % numerator
P_den='(s)*(s+p2)* ...
       ((s^2)/(wn^2)+2*zet*s/wn + 1)*(s+c1)';
                                   % denominator
       % include more factors, etc as needed
  % PLEASE INCLUDE THE CERTAIN INTEGRATORS IN
  % P_den.

% Real Factored Form Structure
% ============================
% WARNING: AT LEAST ONE UNCERTAIN PARAMETER MUST
% BE PRESENT IN THE LANT DEFINITION.
% NUMERATOR AND DENOMINATOR MUST EACH HOLD >= ONE
% PARAMETER
% (CERTAIN OR UNCERTAIN).  IF THE PLANT IS
% CERTAIN, DEFINE E.G. AN UNCERTAIN GAIN
% 'k=[1,1+eps,1,1]' IN Par

P_num='(gain,p1)(delay,p3)';
P_den='(hf,p2)[1 c1 0]';

% include more factors, etc as needed

% Certain polynomial factors: [  ] with Matlab
% syntax, incl integrators
     % Uncertain rff factors: (  )
     % (gain,k)=k    (dc,a)=(1+s/a)
     % (hf,a)=(s+a)
     % (delay,tau)=exp(-s*tau)
     % (dc,wn,zet)=(1 + 2*zet*s/wn + s^2/wn^2)
     % (hf,wn,zet)=(s^2 + 2*zet*wn*s + wn^2)
     % Note that the uncertainty in each factor
     % will be treated as independent even if the
     % same parameter is used.
     % PLEASE INCLUDE THE CERTAIN INTEGRATORS IN
     % P_den AS [1 0], etc.
```

```
% number of additional uncertain differentiators
% ========================
n_dif=[0 0];
     % n dif = []; or n dif not given
     %      <==> n_dif=0 ,
     % the nominal case is 0, and
     % there are no uncertain differentiators in
     % the templates. In all other cases, n_dif
     % has at least two elements. The last element
     % of n_dif denotes the number of
     % differentiators in the nominal case which
     % may be outside the templates. The other
     % elements of n dif denote the uncertain
     % differentiator cases of the templates.
     % Examples:
     % n_dif = [d1 d2 d3];
     % the nominal case has d3
     % differentiators, while a template is
     % the union of T*s^d2 and T*s^d3, where
     % T is the template without uncertain
     % differentiators defined above.
     % n_dif = [-2 -2]
     % the nominal case has 2 integrators, while
     % the templates also have two integrators, so
     % indeed we have a case of a certain number
     % of integrators, that could have been
     % treated in P_den above, with n_dif = [0 0];
     %
     % Note: negative numbers denote integrators
```

PLANT_ID

Purpose

Compiles a plant description file and creates an auxiliary m-file.

Syntax

```
[Par_range,Par_name,Par_nom,Par_points,Par_const,...
n_dif,Uns_Par,w_tpl,w_nom,method,...
Kn1,Un1,Un_t1,Kn2,Un2,Un_t2]=plant_id(Plant)
```

Description

Compilation of the Plant description file [Plant,'.m'], and creates the auxiliary file ['~',Plant,'.m'] in the current directory.

Inputs

`Plant`	Name of a Plant description file, a string variable without '.m'.

Purpose

Invokes a default editor to edit a plant description file.

Syntax

`plnt file name`

Description

Invokes a default editor with the plant file `file name`.m If `file_name`.m does not exist in your current directory, a standard model plant file (plant.m in the Qsyn library) is placed in `file_name`.m for editing by the user.

Example

`plnt ex1`

If `ex1.m` exists in the work space, then `ex1.m` is invoked by the default editor. If `ex1.m` does not exist in the current directory, then `plant.m` is copied into `ex1.m` and opened for editing.

PMODEL

Purpose

Interactive plant modelling function for measured freq function data.

Syntax

```
pmodel(plant,data,freq,nic_bode)
```

Description

Compares a measured frequency function with a modelled one in a Bode or Nichols diagram.

Inputs

`plant`	name of controller file (see `fbcomp.m` or `prefil.m`) computing the transfer function of one plant model candidate. String variable without '.m'.
`data`	matrix that contains the measured data, whose format is as follows: first column is frequency vector (rad/s) and the other columns hold the measured data in Nichols form, [degree + dB*j];
`freq`	frequency vector [rad/s] to be used for the plant model simulation.
`nic bode`	A flag that indicates if the frequency functions will be presented in a Nichols or Bode diagram.
`nic bode`	= 0: Nichols diagram ~= 0: Bode diagram (default)

Example

```
pmodel('model',tpl_meas,tpl_meas(:,1),1);
```

The Bode plot of the frequency function in `model.m` is displayed together with the frequency function data in `tpl_meas` for the frequencies defined by the first column in `tpl meas`.

See also

FDESIGN, CDESIGN, CASES, CCASES.

PNOMINAL

Purpose

Computes the `freq` function values of the nominal in a plant description file.

Syntax

`Pnom=P(Plant,w_op);`

Description

Subroutine that computes the frequency function values in complex form of the nominal plant case defined in Plant.m.
The user is advised to use CASES to display nominal plant.

Outputs

`Pnom`	row vector, cantaining the nominal frequency function values in complex form (Nyquist form).

Inputs

`Plant`	name of plant description file, string without extension '.m'
`w_op`	optional frequency vector, rad/s, for which `Pnom` is to be computed. Default: nominal frequency vector `w_nom` in Plant.m. Remember that `w_nom` always is recomputed to include `w_tpl` (the template frequencies) in Plant.m.

Purpose

Prefilter function model file, to be copied and edited by the user.

Syntax

Description

Use: Copy the file into the workspace, change its filename, and name in its head above, and edit the file to reflect the current prefilter.

`prefil` is called by fdesign.m to plot the current closed loop frequency function magnitude.

Outputs

`F`	`F` is the only output argument. `F` is a complex vector, equalling the prefilter frequency function.

Inputs

`s`	`s` is the only input argument. `s` = j*w, where w is the frequency vector [rad/s].

Remarks

The equation of the frequency function may be written inwhatever form the user chooses, with the restriction that all vector operations involving s be elementwise, i.e. use a point, . , in front of arithmetic operators such as *, /, ^, etc.

A Real Factored Form of F is suggested below. Notice that the parameters whose absolute value is 1/eps gives factors whose value equals 1.

If F is to represent a digital controller, include e.g the following statement:

```
z = exp(s*T);
```

where T is the sampling period [s], and define F asa function of z.

Some of the parameters below are preset to 1/eps or -1/eps, where eps is a Matlab variable denoting the machine precision. Such parameters cancel their respective frequency function factors to 1, e.g.

```
p2 = -1/eps;
```

makes the single pole factor in F

```
pole2 = 1./(1-s/p2) = 1
```

If pole2 is needed in F, then the user sets p2 to the desired pole location, e.g.

```
p2 = -2.5;
```

See the code below.

Normally the user who likes working with feedback compensator transfer functions in DC real factored form, will assign appropriate values to those paramaters k, n, p1, ..., z1, ... that represent needed prefilter factors. If necessary, the user may add more parameters and factors, such as e.g.

```
p6 = -10.5;
```

and the code

```
pole6 = 1./(1-s/p6);
```

and also change

```
F = ... pole5.*zero1 ...;
```

to

```
F = ... pole5.*pole6.*zero1 ...;
```

Body

```
[F] = prefil(s)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% User's comment: prefilter no. for plant
%
% DC-gain
% =======
  k = 1;
%
% Number of integrators
% =====================
  n = 0;
%
% Real Poles
% ==========
  p1 = -1/eps;   p2 = -1/eps;      p3 = -1/eps;
     p4 = -1/eps;      p5 = -1/eps;
%
% Real Zeros
% ==========
  z1 = -1/eps;   z2 = -1/eps;      z3 = -1/eps;
     z4 = -1/eps;      z5 = -1/eps;
%
% Complex Poles
% =============
  zp1 = 0; zp2 = 0;    zp3 = 0;    zp4 = 0;    zp5
= 0;
  wp1 = 1/eps;   wp2 = 1/eps;      wp3 = 1/eps;
     wp4 = 1/eps;      wp5 = 1/eps;
%
% Complex Zeros
% =============
  zz1 = 0; zz2 = 0;    zz3 = 0;    zz4 = 0;    zz5
= 0;
  wz1 = 1/eps;   wz2 = 1/eps;      wz3 = 1/eps;
     wz4 = 1/eps;      wz5 = 1/eps;
%
% -------------------------------------------------
------------------------
%
  pole1 = 1./(1-s/p1); pole2 = 1./(1-s/p2);
     pole3 = 1./(1-s/p3);
                 pole4 = 1./(1-s/p4);    pole5 =
1./(1-s/p5);

  zero1 = 1-s/z1;zero2 = 1-s/z2;         zero3 =
1-s/z3;
                 zero4 = 1-s/z4;         zero5 =
1-s/z5;

  cpole1 = 1./(1 + (2*zp1 + s/wp1).*s/wp1);
     cpole2 = 1./(1 + (2*zp2 + s/wp2).*s/wp2);
  cpole3 = 1./(1 + (2*zp3 + s/wp3).*s/wp3);
     cpole4 = 1./(1 + (2*zp4 + s/wp4).*s/wp4);
                                   cpole5 = 1./(1
+ (2*zp5 + s/wp5).*s/wp5);

  czero1 = 1 + (2*zz1 + s/wz1).*s/wz1;
     czero2 = 1 + (2*zz2 + s/wz2).*s/wz2;
  czero3 = 1 + (2*zz3 + s/wz3).*s/wz3;
     czero4 = 1 + (2*zz4 + s/wz4).*s/wz4;
                                   czero5 = 1 +
(2*zz5 + s/wz5).*s/wz5;

  F =
(k./s.^n).*pole1.*pole2.*pole3.*pole4.*pole5.*zero
1.*zero2.*zero3.*zero4.*zero5 ...
```

```
.*cpole1.*cpole2.*cpole3.*cpole4.*cpole5.*czero1.*
czero2.*czero3.*czero4.*czero5;
```

See also

CDESIGN FDESIGN fbcomp.

Purpose

Removes interior points from a connected template.

Syntax

```
[T,t index]=prune(t,Tacc,upper);
```

Outputs

`T`	pruned template vector, with each element of the form deg + j*dB.
`t index`	tindex is the indices of the points in t that are contained in the final pruned set. Useful for parameter handling, see the specification of the template file.

Inputs

`t`	template vector to be pruned, with each element of the form deg + j*dB
`Tacc`	[degree_accuracy , dB_accuracy], given 2-norm accuracy in Nichols form. A larger value gives a smoother appearance.
`upper`	`upper==1` only gives the upper border of a template with phase uncertainty larger than 360 degrees. default: upper=0.

Remarks

Removes interior points of a value set in Nichols form deg+j*dB. The points must be connected, i.e. the union of all ellipses (x-x(i))^/Tacc(1)^2+(y-y(i))^/Tacc(2)^2 where t(i)=x(i)+j*y(i) must be connected.

Connected templates are produced by e.g. adgrid, adedge, or rff. PRUNE works also on a non-connected template (e.g. one created with the grid method) but not very well. A non-connected template can be made connected with the help of the command CLTMP

The algorithm handles both wrapping over the Riemann surfaces and phase uncertainty larger than 360 degrees.

T=prune(t,Tacc,1) only gives the upper border of a template with phase uncertainty larger than 360 degrees.

Example

```
redge5=getfrom('ex2_1b.tpl','t_w5'); % gettpl can
     %also be used!
```

gets a template out of `ex2_1b.tpl` which was created with the Recursive Edge Grid method.

```
Tredge5=prune(redge5,[2 2]);
plot(Tredge5,'ro') % prune and plot

grid5=getfrom('ex2_1c.tpl','t_w5');
Tgrid5=prune(grid5,[2 2]);
plot(Tgrid5,'bx')
```

the same for a template generated with the Grid method!

See also

TPLPRUNE TPLREDUC.

PRUNE1

Purpose

Testversion of PRUNE that displays each step of the iteration.

Syntax

```
[T]=prune1(t,Tacc,upper);
```

Description

THIS IS A TESTVERSION OF PRUNE THAT DISPLAYS EACH STEP OF THE ITERATION!
See PRUNE.
Removes interior points of a value set in Nichols form deg+j*dB.
The points must be connected, i.e. the union of all ellipses
(x-x(i))^/Tacc(1)^2+(y-y(i))^/Tacc(2)^2 where T(i)=x(i)+j*y(i)
must be connected.
Such sets are produced by e.g. `adgrid` or `adedge`.

Inputs

`t`	the value set (in Nichols form) to be pruned.	
`Tacc`	[degree_accuracy , dB_accuracy], Given accuracy in Nicholls form. A larger valu gives a smother appearance.	
`upper`	`upper`==1	only gives the upper border of a template with phase uncertainty larger than 360 degrees.
	default: `upper`=0.	

Remarks

The algorithm handles both wrapping over the Riemann surfaces and phase uncertainty larger than 360 degrees.

`T=prune1(t,Tacc,1)` only gives the upper border of a template with phase uncertainty larger than 360 degrees.

PUTP

Purpose

Put a point before '^' , '/' , '*' in a string (subroutine).

Syntax

```
[nfunc]=putp(func)
```

Outputs

`nfunc` string expression with points before '^' , '/' , '*'.

Inputs

`func` string expression without points before '^' , '/' , '*'.

Purpose

Computes freq function values/SIMULINK block for factored transfer fcn.

Syntax

```
[P]=pz2s(s,gain,zeros,poles,czeros,cpoles,int,...
diff,delay,sname)
```

Description

User: This function is provided as a service only.
There is no guarantee or support of it.

Outputs

`P` frequency function for the frequencies in s = j*w

Inputs

`s` = j*w = j*[w1 w2 w3 ....]; frequency vector [rad/s]

`zeroes, poles, czeros, cpoles`
matrices, where each row holds the frequency response for `s`, for a first or second order factor, see example below.

`gain, int, diff, delay`
constants, representing gain, number of integrators, number of differentiators, and delay (seconds).

`sname` optional name of Simulink block to be created, string variable.

Remarks

The vector `s` must have at least two elements.

Example

For a plant that has two zeroes: (s+a)*(s+b) the matrix zeroes looks as follows:

```
zeroes = [ (j*w1+a) (j*w2+a) ...
         (j*w1+b) (j*w2+b) ...                ];
```

Purpose

Makes a grid out of two vectors .

Syntax

```
[outgrid]=qgrid(n,qmin,qmax);
```

Description

First grids rowwise between qmin(i) and qmax(i) with n(i) grid points, and then produces all vector combinations over i, such that each column in [outgrid] is a combination.

Outputs

`[outgrid]`	matrix where each column is a combination. `[outgrid]` has as many rows as n has elements.

Inputs

`n`	integer vector. Each element `n`(i) tells how many grid points there will be in the interval [`qmin`(i) `qmax`(i)].
`qmin`	vector of the same size as `n`, denoting the left endpoints.
`qmax`	vector of the same size as `n`, denoting the left endpoints.

Example

```
Q=qgrid([3,2];[1 10],[2 20]),
Q= [  1.0   1.5   2.0    1.0    1.5    2.0
     10.0  10.0  10.0   20.0   20.0   20.0 ]
```

See also

PGRID, PARGRID.

QUNWRAP

Purpose

Unwraps templates in Nichols form over several Riemann surfaces.

Syntax

```
[T]=qunwrap(t,tol);
```

Outputs

`T`	unwrapped template, row vector in Nichols form.

Inputs

`t`	the template to unwrap, row vector in Nichols form.
`tol`	the tolerance angle when unwrapping, default is 180 deg, see UNWRAP.

Remarks

If t is a matrix, the unwrapping takes place column wise.

See also

UNWRAP, WRAP.

READP

Purpose

Reads one variable from a mat-file (subroutine).

Syntax

```
[x,name,pos e]=readp(matf,pos s)
```

Outputs

`x`	value of the extracted variable.
`name`	name of the extracted variable, string.
`pos e`	final position pointer of the variable in the file.

Inputs

`pos s`	optional pointer to the initial reading position. Default = 0 (beginning of the file).
`matf`	name of mat-file, string with extension.

RECEDGE

Purpose

Subroutine used by ADEDGE.

Syntax

```
[Tnew,Qpar]=recedge(trf,s,qmin,qmax,Tmin,Tmax,...
Tacc,qdist);
```

RECGRID

Purpose

Subroutine used by ADGRID.

Syntax

```
[T,Qpar]=recgrid(trfun,s,n,qmin,qmax,Tgrid,Tacc,
phandle,indgrid,Nmul,qconst);
```

Remarks

This code is difficult to read, due to the complicated handling of n-dimensional grids, without support for them in Matlab.

REMOVE

Purpose

Removes a variable from a mat-file.

Syntax

`remove(matf,name)`

Inputs

`matf`	full name of mat-file within ' '.
`name`	name (within ' ') of variable to be removed.

Remarks

tpl-, spc-, and bnd-files are mat-files from which variables may be removed.

No warning is given if `matf` does not exist.

No warning is given if `name` does not exist in `matf`.

No warning is given that `name` is not removed from `matf`,in case of name being the only variable in `matf`.

The function always leave a variable 'filename' that contains the file name itself.

Example

`remove('ex0.tpl','t w1')`

removes the variable `t_w1` from `'ex0.tpl'`, if `'ex0.tpl'` exists, and if `t_w1` exists in `'ex0.tpl'`, and if `t_w1` is not the only variable in `'ex0.tpl'`. Otherwise no action is taken and no warning is given.

See also

LOOK GETFROM INSERT.

RFFCPZ

Purpose

Computes a complex pole/zero pair template in real factored form..

Syntax

`[T]=rffcpz(a1,a2,w,form,pzf,dist)`

Outputs

`T` column vector with an even number of elements. The first half the vector `T` is the low gain template edge, the second half of the vector `T` is the high gain template edge. Each element is of the form degree + j*dB. Each template edge is sorted in ascending order with respect to angle. The same angles occur in both edges. `T` contains angles in the interval (-180, 180] deg, but may have angles outside this interval in order to make the angle sequence contiguous. Each angle is a multiple of `dist` [deg], see below.

Inputs

`a1` When uncertain, a1 is a vector with two real elements in the form `a1`=[zmin zmax]. zmin and zmax represent the minimum and maximum values of the uncertain relative damping z. When the relative damping is certain, either zmin = zmax, or `a1` contains one element only. The elements of `a1` are real numbers.

`a2` When uncertain, `a2` is a vector with two positive real elements in the form `a2`=[wnmin wnmax]. wnmin and wnmax represent the minimum and maximum values of the uncertain resonance frequency wn [rad/s]. When wn is certain, either wmin = wmax, or `a2` contains one element only. The elements of `a2` are positive real numbers.

`w` [rad/s], non-negative real number, the freqeuncy for which the is computed.

`form` The form of the factor, `'dc'` or `'hf'`.

- `'dc'` Indicates dc form (1 + 2*z*s/wn + s^2/wn^2) or 1/(1 + 2*z*s/wn + s^2/wn^2) (default).
- `'hf'` Indicates high frequency form (s^2 + 2*z*wn*s + wn^2) . or 1/(s^2 + 2*z*wn*s + wn^2).

`pzf` Pole/zero flag, `'z'` or `'p'`.

- `'z'` Indicates zero factor (1 + 2*z*s/wn + s^2/wn^2) or (s^2 + 2*z*wn*s + wn^2) (default).
- `'p'` Indicates pole factor 1/(1 + 2*z*s/wn + s^2/wn^2) or 1/(s^2 + 2*z*wn*s + wn^2).

`dist` The template edges are computed for angles that are multiples of `dist` [deg] which must be a positive real number such that 360 may be divided by dist without remainder The default of dist is 1 [deg]. The angular distance between neigbouring edge points is a multiple of dist.

Remarks

End point phase rounding is performed as follows: If a true template end point is nearer a phase grid point outside the the true template, that grid point is included in the computed template, with a gain value equal to that of the true template end point.

Reference: Gutman, P-O, Baril C, Neumann L: "An algorithm for computing value sets of uncertain transfer functions in factored real form." IEEE Transactions on Automatic Control, vol 29, no 6, 1268-1273, June 1994.

See also

RFFPZ, RFFEL, RFFMUL.

Purpose

Pure gain, delay, unstructured uncertainty, or integrators rff template.

Syntax

```
[T] = rffel(element,a,w,dist)
```

Description

produces a template in real factored form for uncertain gain, uncertain delay, multiplicative unstructured uncertainty, and an uncertain number of integrators.

Outputs

`T`	column vector with an even number of elements. The first (upper, with lowest indeces) half the vector `T` denotes the low gain template border, the second half of the vector `T` denotes the low gain template border. Each vector element is of the form degree + j*dB (Nichols chart representation).

Inputs

`element`	`'gain', 'delay', 'uns', or 'int'`. See below.
`a`	Vector with two or more real elements. See below.
`w`	frequency [rad/s], non-negative real number, for which the template is computed.
`dist`	The phases of the computed template values are integer multiples of `dist` [deg]. The angular distance between two neighbouring computed template point is `dist` [deg] (default = 1). `dist` must be such that 360 is divisible by `dist` without remainder.

`[T] = rffel('gain',a,w,dist)` produces a magnitude template.

`a`	Vector with two real elements in the form of `a`=[amin amax]. amin represents the minimum gain [absolute value] and amax represents the maximum gain [absolute value]. Maximum and minimum gain must have the same non-zero sign.

`[T] = rffel('delay',a,w,dist)` produces a delay template.

`a`	Vector with two non-negative real numbers in the form of `a`=[amin amax]. amin represents the minimum delay [seconds] and amax represents the maximum delay [seconds].

`[T] = rffel('uns',m,dist,w)` produces an unstructured multiplicative uncertainty template.

`m`	Matrix with two rows, or alternatively, one real number in [0,1) . If `m` is a matrix, then `m` must contain at least two columns. The first row, containing non-negative real numbers holds frequencies [rad/s] in increasing order. The second row, consisting of real numbers in the interval [0, 1), contains the unstructured multiplicative uncertainty radii [absolute value], for each of the frequencies in the first row, respectively. The uncertainty radius is then linearly interpolated, or end point constant extrapolated, respectively, with respect to frequency. If `m` is a single number, then it denotes the uncertainty radius for all frequencies.

`[T] = rffel('int',a,w,dist)` produces a templete for an uncertain number of differentiators/integrators.

`a`	Vector of two integer numbers, in the form of `a`=[amin amax]. amin represents the minimum number of integrators, and amax represents the maximum maximum number of integrators.(-amin represents the maximum number of differentiators, and -amax represents the minimum number of differentiators)

Remarks

End point phase rounding is performed as follows: If a true template end point is nearer a phase grid point outside the the true template, that grid point is included in the computed template, with a gain value equal to that of the true template

Reference: Gutman, P-O, Baril C, Neumann L:"An algorithm for computing value sets of uncertain transfer functions in factored real form." IEEE Transactions on Automatic Control, vol 29, no 6, 1268-1273, June 1994..

See also

RFFPZ, RFFCPZ, RFFMUL.

RFFMUL

Purpose

Multiples two templates in real factored form.

Syntax

```
[T]=rffmul(t1,t2,dist)
```

Outputs

`T`	column vector with an even number of elements. The first half the vector `T` is the low gain template edge, the second half of the vector `T` is the high gain template edge. Each element is of the form degree + j*dB. Each template edge is sorted in ascending order with respect to angle. The sequence of angles is not necessarily contiguous. Each angle is a multiple of dist [deg]. `T` is found as the upper and lower edges of the vector addition (concatenation) of `t1` and `t2` in the Nichols chart corresponding to the multiplication of the templates in the Nyquist chart

Inputs

`t1,t2`	column vectors representing templates in real factored form, of the same structure as `T`, with the relaxation that for each edge the angles do not have to be sorted, nor do they have to be contiguous (see `rffpz`).
`dist`	The template edges are computed for angles that are multiples of dist [deg] which must be a positive real number such that 360 may be divided by dist without remainder The default of dist is 1 [deg]. The angular distance between neigbouring edge points is a multiple of dist.

Remarks

Reference: Gutman, P-O, Baril C, Neumann L: "An algorithm for computing value sets of uncertain transfer functions in factored real form." IEEE Transactions on Automatic Control, vol 29, no 6, 1268-1273, June 1994.

See also

RFFPZ, RFFEL, RFFCPZ.

RFFPZ

Purpose

Produces a real pole or real zero template in real factored form.

Syntax

```
[T] = rffpz(a,w,form,pzf,dist)
```

Outputs

`T` column vector with an even number of elements. The upper half the vector `T` is the low gain template edge, the second half of the vector `T` denotes the high gain template edge (equal in the case of a first order factor). Each element is of the form degree + j*dB. (Nichols chart representation). Each edge is sorted in ascending order w r t angles, in the interval (-180, 180] deg. In some cases the angles are not contiguous (e.g. when the parameter a defines both positive and negative numbers).

Inputs

`a` When the factor is uncertain, a is a vector with two real elements in the form `a`=[min max] min and max represent the minimum and maximum values of the uncertain parameter. When the factor is certain, either min=max, or a contains one element only.

`w` frequency [rad/s], non-negative real number. form: The form of the factor, `'dc'` or `'hf'`.

`'dc'` Indicates dc form (1 + s/a) or 1/(1 + s/a) (default).

`'hf'` Indicates high frequency form (s+a) or 1/(s+a).

`pzf` Pole zero flag, `'z'` or `'p'`.

`'z'` Indicates zero factor (1 + s/a) or (s+a) (default).

`'p'`: Indicates pole factor 1/(1 + s/a) or 1/(s+a).

`dist` The template edges are computed for angles that are multiples of dist [deg] which must be a positive real number such that 360 may be divided by dist without remainder The default of dist is 1 [deg]. The angular distance between neigbouring edge points is a multiple of dist.

Remarks

End point phase rounding is performed as follows: If a true template end point is nearer a phase grid point outside the the true template, that grid point is included in the computed template, with a gain value equal to that of the true template.

Reference: Gutman, P-O, Baril C, Neumann L:"An algorithm for computing value sets of uncertain transfer functions in factored real form." IEEE Transactions on Automatic Control, vol 29, no 6, 1268-1273, June 1994.

See also

RFFCPZ, RFFEL, RFFMUL.

Purpose

Utility function for RFFCPZ to compute template edge points.according to given edge case for a complex pole or zero pair.

Syntax

```
[T]=rffutil1(w,phi,zmin,zmax,wmin,wmax,form,...
pzf,case)
```

Outputs

`T`	vector of edge template points of the same dimension as phi. Each element is of the form degree + j*dB, with degree in the interval (-180,180] deg.

Inputs

`w`	frequency [rad/s], for which the template is computed.
`phi`	column vector with phase values [deg] for which the edge points are computed.
`zmin`	minimum relative damping
`zmax`	minimum relative damping
`wmin`	minimum natural frequency [rad/s]
`wmax`	maximum natural frequency [rad/s]
`form`	`'dc'` or `'hf'`, dc/hf flag
`pzf`	`'p'` or `'z'`, pole/zero flag
`case`	= 1, 2, 3, or 4, border segment case.

See also

RFFCPZ.

RFFUTIL3

Purpose

Utility function for RFFCPZ to sort, clean, and complement edges.

Syntax

`[Tnew]=rffutil3(Tleft,T,Tright,edge,dist)`

Description

The edge `T` is first sorted w r t ascending angle. Then the sorted edge, `T`, and which has groups of more than one element with the same phase, select, from each group, the highest gain member(s), if `edge=='hi'`, and the lowest gain member if `edge=='lo'`. Finally, the edge is complemented with one or two new endpoints if these are angularly nearer its true endpoints.

Angular elimination may occur for -90, 0, 90, 180 degrees,when an edge sits or "almost" sits on an axis, or at the vertices of elementary edges. Phase rounding in described in RFFCPZ.

Outputs

`Tnew`	sorted, cleaned, and end point complemented edge.

Inputs

`Tleft`	Two exact endpoints of left (low angle) segment of `T.`
`T`	edge to be corrected, row or column vector in Nichols form (degree + j*dB)
`Tright`	Two exact endpoints of right (high angle) segment of `T.`
`edge`	`'hi'` or `'lo'.`
`dist`	current angular resolution, degrees

See also

RFFCPZ.

RMDBLP

Purpose

Removes doublets from a sorted vector.

Syntax

```
[vnew,index]=rmdblp(vold)
```

Outputs

`vnew`	vector with doublets removed.
`index`	index vector of positions in vold that were retained in `vnew`.

Inputs

`vold`	sorted vector with doublets.

RNDSPACE

Purpose

Generates a row vector with randomly spaced elements.

Syntax

`y = rndspace(p1, p2 , n)`

Outputs

`y`	row vector with `n` elements. The left element is `p1`, the right element is `p2`, and inbetween there are `n`-2 randomly spaced points.

Inputs

`p1`	left endpoint, scalar.
`p2`	right endpoint, scalar.
`n`	number of elements in `y`.

Purpose

Reference Step Response Specification Calculation.

Syntax

```
[spec w,spec t,tab]=rsrs(spcfile,specname,Tr,M,...
Ts,Td,w,wco,ordr,Ks,tf,plt,dt,n)
```

Description

Transfers time-domain reference step specifications to the frequency domain, using simulations of 2:nd or 3:rd order systems.

Outputs

`spec_w`	Three-column matrix, where the first column contains the frequency vector, the second column contains the upper bound and the third column contains the lower bound.
`spec t`	Three-column matrix, where the first column ontains the time vector, the second column contains the upper envelope of the step responses and the third column contains the lower envelope.
`tab`	Three-column matrix, where the first column contains the time vector corresponding to the time-domain specifications. The second column contains the upper bound corresponding to the time-domain specifications and the third column contains the corresponding lower bound.

Inputs

`spcfile`	the file in which the spec is stored
`specname`	the name under which the spec is stored, default is `'rsrs'`.
`Tr`	=[maximum-rise-time, minimum-rise-time, level in percent]; Default level is 90%.
`M`	=overshoot (percent). The default value is `M`=10%.
`Ts`	=[Settling time,upper deviation in percent,lower deviation in percent]. The default value for the lower deviation is the upper deviation and the default value for the upper deviation is 5%. The default value for the Settling time is 5*Rise-time.
`Td`	=[Delay-time,level in percent]. Default value for the level is 10% and the default value for the delay time is the upper rise-time.
`w`	frequency vector. The default value is `logspace(...` `log10(2*pi/(Final-time)/20),...` `log10(2*pi/(Lower-rise-time)*20))`.
`wco`	cut-off frequency. The lower bound of the frequency specification takes the value '20*log(eps)' for all angular frequencies above `wco`. The default value is `wco=Inf`.
`ordr`	system order. Can be 2, 3 or 3.1. The default value is 2. order 2 simulates 1/(s^2/w0^2+2*z*s/w0+1). Order 3 simulates s+b)/[(s+a)*(s^2/w0^2+2*z*s/w0+1)] with appropriate limits on a,b,w0 and z to avoid resonance peaks, it includes also second order systems order 3.1 is an alternative gridding of order 3, described in Horowitz: QFT vol 1, 1993. Use instead of 3 for higher speed, it does not include all the cases of order 2. See `spc rs2,spc rs3,spc rs31` for more information.

Ks	Static gain. The default value is Ks=1.
tf	final time. The default value is 2*Setting-time.
plt	plot option. Logic variable for switching on and off plottinginformation of step responses. Can be 0 or 1. Default is 1.
dt	time increment, the default value is 'Final-time/100'.
n	number of grid-points. The default value is n=[40,40] for the 2:nd order system, n=[10,10,10,10] for the 3:rd order system and n=[10,10,10,3] for the alternative gridding. The number of grid-points becomes 'prod(n)'. See the mfiles spc rs2, spc_rs3, spc_rs31, too see what the different parameterizations are, in order to choose another gridding.

Setting any input variable to [] gives the default value of that parameter.

Example

```
rsrs('ex2 1a',[],[1.2 0.2],10,1.5,[],...
logspace(-1,2),2.85,3.1);
```

Inserts into the specification file ex2 1.spc a specification called rsrs, based on simulated reference step responses that have a rise time of between 0.2 1.2 seconds to the default level 50%, a maximum overshoot of 10%, a settling time of 1.5 seconds to within 5%. (% refer to % of disturbance step amplitude).
The delay time is the defualt value. The frequency domain specification is defined for the frequencies logspace(-1,2). A cut-off frequency of 2.85 rad/s is required. The special simulation system order of 3 is used.

See also

RSRS, ODSRS, IDSRS, SPC_RS2, SPC_RS3, SPC_RS31.

SHOWBND

Purpose

Plots bounds from a bound file for selected frequencies, in a Nichols chart.

Syntax

```
showbnd(bndfile,phandle,w1,bnd1,c1,w2,bnd2,c2,...
w3,bnd3,c3,w4,bnd4....)
```

Outputs

`phandle`	handle to the figure.

Inputs

`bndfile`	the boundfile name, string with or without extension '.bnd', containing the bounds
`phandle`	handle to the figure, if =[] a new figure is invoked.
`w1,w2,...`	the frequencies for which to plot each bound, [] gives all freqs.
`bnd1,bnd2,...`	names of the bounds to plot, strings, e.g. `'odsrs'`, `'rsrs'`
`c1,c2,c3,...`	the color of the bound plot, must be either a standard Matlab color option e.g `'r:'` or `'b--'` or `['roll',x]` where x is a matlab linestyle e.g. `':'` The roll options automatically rolls over different colors.

Remarks

When phandle ~= [], no warning is given if a bound falls outside current axis.

Example

showbnd('ex2_1a',gcf,[],'rsrs');
Show, in the current figure, all rsrs bounds from `ex2_1a.bnd`.

See also

BNDINF.

Purpose

Displays specification from a specification file.

Syntax

```
[phandle]=showspc(spcfile,spec,plot op,color,...
phandle)
```

Outputs

`phandle`	A handle to the figure.

Inputs

`spcfile`	Name of specification file, string with or without extension '.spc', from which specifications are to be displayed.	
`spec`	The specification to be displayed. Can be any name, but typically one of the following:	
	`'iosrs'`	GP/(1+GP) sensor noise gain, complementary sensitivity.
	`'idsrs'`	P/(1+GP) input disturbance to plant output.
	`'odsrs'`	1/(1+GP) output disturbance to plant output, sensitivity.
	`'odsrsc'`	G/(1+GP) output disturbance to control output/plant input.
	`'rsrs'`	max(GP/(1+GP))/min(GP/(1+GP)) servoL specification, tolerance.
	`plot_op`	`'both'` (default) plot both frequncy and time domain specifications, `'time'` and `'freq'` plots only time and frequency domain specifications respectively.
`color`		color of the plots e.g. `'r:'` or `'B--'`,...., default is `'r-'`.
`phandle`		A handle to the figure you want to plot in, use `phandle`=gcf if you want to plot in the current fig. An empty phandle invokes a new figure (default).

Remarks

Don't forget to command hold on before SHOWSPC. You may zoom in on details of the template display by either the Matlab command zoom, or by the Qsyn command hzoom, that puts a zoom menu on the current figure toolbar, and which also lets you zoom out\beyond the borders of the original picture.

Example

```
showspc('ex5 1','odsrs','freq','g',gcf);
```

Plot the frequency domain odsrs specification from the file `ex5_1.spc` in the current figure (`gcf`) in green.

See also

RSRS, ODSRS, SPCUPD, IDSRS.

SHOWTPL

Purpose

Displays templates from a template file in a Nichols diagram.

Syntax

```
[phandle]=showtpl(tplf,w_op,option,line style,...
phandle)
```

Outputs

`phandle` A handle to the figure.

Inputs

`tplf` Name of template file, from which templates are to be displayed.

`w_op` Vector of frequencies [rad/s], for which templates are to be displayed. A frequency without template is ignored. Default: If `w_op` is non-existant, or `w_op==[]`, all templates in `tplf` are presented.

`option` Selector for how the templates are presented. `Option` can be one of the following:

- `'nom'` The nominal plant is displayed in a heavy red linestyle, and the templates are drawn correctly relative their nominal points.
- `'point'` The user clicks with his mouse on the Nichols chart for the location of the nominal point of the next template, and the template is drawn correctly relative to that point.

Default: `option='nom'`.

`linestyle` the color/linestyle of the bound plot,

- `'fill','fill roll',` filled templates, with a new color for each template.
- `'fill_r','fill_g',` filled templates the same color e.g. red or green.
- `'roll','roll_:','roll_-',` new color for each template, with your own line style, or the default '.'.

`phandle` A handle to the figure you want to plot in,use `phandle=gcf` if you want to plot in the current figure. An empty phandle invokes a new figure (default).

Remarks

During the execution of the display under the `'point'` option, pressing the left button gives the next point and pressing the right button terminates `showtpl`

If phandle is a figure with no axes (e.g. after you have opened a new figure with the command figure), then PLEASE issue the command `hngrid` before `showtpl`, so that the axes will be suitable for the display of the templates.

You may zoom in on details of the template display by either the Matlab command `zoom`, or by the Qsyn command `hzoom`, that puts a zoom menu on the current figure toolbar, that also lets you zoomout beyond the borders of the original picture.

Examples

```
showtpl('ex2_1a');
```

plots all the templates in ex2_1a.tpl along the nominal.

```
showtpl('ex2 1a',[],'point');
```

lets the user place all templates in `ex2 1a.tpl` by mouse clicks in the current figure if it exists, or in a new first figure whose axes were drawn by the Qsyn command `hngrid`.

```
showtpl('ex2_1a',[.15 .2]);
```

plots the templates for [.15 .2] rad/s in `ex2_1a.tpl`, if they exist, along the nominal.

See also

HNGRID MGRID CTPL HZOOM.

SPC_ID2

Purpose

Subroutine used by IDSRS.

Syntax

```
[spec w,spec t,tab]=spc id2(spc tab,w,dt,plt,...
zmin,n)
```

Description

Sub-function called by `idsrs`. Can be used separately for advanced use. The function calculates frequency domain specifications given time domain specifications for an input disturbance step. The calculations are done by gridding the parameters of a second order system.

Outputs

`spec_w`	Two-column matrix, where the first column contains the frequency vector and the second column contains the upper bound.
`spec t`	Three-column matrix, where the first column contains the time vector, the second column contains the upper envelope of the step responses and the third column contains the lower envelope.
`tab`	Three-column matrix, where the first column contains the time vector corresponding to the time-domain specifications. The second column contains the upper bound corresponding to the time-domain specifications and the third column contains the corresponding lower bound.

Inputs

`spc tab`	a 3-column matrix containing time domain specifications for a reference step. The first column contains the times, second column contains the upper bound and the third contains the lower bounds. The upper and lower bounds are assumed to be LINEARLY interpolated between the values in the second and third columns, respectively.
`w`	the frequency vector, preferably created by logspace.
`dt`	the time increment. (If you want to have a denser or sparser time gridding
`plt`	a logic variable for switching on and off plotting. The default value is 1.
`zmin`	the minimum relative damping of the system. The default value is 1/sqrt(2).
`n`	the number of grid-points. The default value is `n`=[10,10,40] and the number of grid-points becomes 'prod(`n`)'. The first grid variable corresponds to 'phi', the second, to 'x', and the third to 'log10(c)' where the second order system is given by c*s/(s^2+2*z*w0*s+w0^2) where w0=2*pi/x, and z=cos(phi). This is the transfer function corresponding to the input disturbance transfer function when the plant is a general first order system and the controller is a pure integration.

See also

RSRS, ODSRS, IDSRS, SPC_ID2.

SPC_OD2

Purpose

Specification calculation for 2:nd order output disturbance step.

Syntax

```
[spec_w,spec_t,tab]=spc_od2(spc_tab,w,dt,plt,n)
```

Description

Sub-function called by odsrs. Can be used separately for advanced use. The function calculates frequency domain specifications given time domain specifications for a output disturbance reference step. The calculations are done by gridding the parameters of a second order system.

Outputs

`spec_w` Three-column matrix, where the first column contains the frequency vector, the second column contains the upper bound and the third column contains the lower bound.

`spec_t` Three-column matrix, where the first column contains the time vector, the second column contains the upper envelope of the step responses and the third column contains the lower envelope.

`tab` Three-column matrix, where the first column contains the time vector corresponding to the time-domain specifications. The second column contains the upper bound corresponding to the time-domain specifications and the third column contains the corresponding lower bound.

Inputs

`spc_tab` a 3-column matrix containing time domain specifications for a reference step. The first column contains the times, second column contains the upper bound and the third contains the lower bounds. The upper and lower bounds are assumed to be LINEARLY interpolated between the values in the second and third columns, respectively.

`w` the frequency vector, preferably created by logspace.

`dt` the time increment.

`plt` a logic variable for switching on and off plotting. The default value is 1.

`n` the number of grid-points. The default value is `n`=[40,40] and the number of grid-points becomes 'prod(`n`)'. The first grid variable corresponds to 'phi' and the second, to 'x', where the second order system is given by 1-1/(s^2/w0^2+2*z*s/w0+1) where w0=10^x, and z=cos(phi).

See also

RSRS, ODSRS, IDSRS, SPC_OD3, SPC_OD31.

SPC_OD3

Purpose

Specification calculation for 3:rd order output disturbance step.

Syntax

```
[spec w,spec t,tab]=spc od3(spc tab,w,dt,plt,n)
```

Description

Sub-function called by `odsrs`. Can be used separately for advanced use. The function calculates frequency domain specifications given time domain specifications for an output disturbance reference step. The calculations are done by gridding the parameters of a third order system.

Outputs

spec_w	Three-column matrix, where the first column contains the frequency vector, the second column contains the upper bound and the third column contains the lower bound.
spec_t	Three-column matrix, where the first column contains the time vector, the second column contains the upper envelope of the step responses and the third column contains the lower envelope.
tab	Three-column matrix, where the first column contains the time vector corresponding to the time-domain specifications. The second column contains the upper bound corresponding to the time-domain specifications and the third column contains the corresponding lower bound.

Inputs

spc_tab	a 3-column matrix containing time domain specifications for a reference step. The first column contains the times, second column contains the upper bound and the third contains the lower bounds. The upper and lower bounds are assumed to be LINEARLY interpolated between the values in the second and third columns, respectively.
w	the frequency vector, preferably created by logspace.
dt	the time increment.
plt	a logic variable for switching on and off plotting. The default value is 1.
n	the number of grid-points. The default value is `n`=[10,10,10,10] and the number of grid-points becomes 'prod(`n`)'. The first grid variable corresponds to '2*pi/p', the second to '2*pi/b', the third to 'log10(u)' and the fourth to 'log10(v)', where the third order step-response is given by alpha*exp(-b*t)+beta*exp(-p*t)*sin(ws*t+Phi) where alpha=(u-v)/sqrt(2)+1, beta=(u+v)/sqrt(2). ws and Phi are the corresponding frequency and phase respectively.

See also

RSRS, ODSRS, IDSRS, SPC_OD2, SPC_OD31.

SPC_OD31

Purpose

Specification calculation for 3:rd order output disturbance step (alternative grid).

Syntax

```
[spec_w,spec_t,tab]=spc_od31(spc_tab,w,dt,plt,n)
```

Description

Sub-function called by `odsrs`. Can be used separately for advanced use. The function calculates frequency domain specifications given time domain specifications for an output disturbance reference step. The calculations are done by gridding the parameters of a third order system using a different gridding than that used in spc_rs3.

Outputs

`spec w`	Three-column matrix, where the first column contains the frequency vector, the second column contains the upper bound and the third column contains the lower bound.
`spec t`	Three-column matrix, where the first column contains the time vector, the second column contains the upper envelope of the step responses and the third column contains the lower envelope.
`tab`	Three-column matrix, where the first column contains the time vector corresponding to the time-domain specifications. The second column contains the upper bound corresponding to the time-domain specifications and the third column contains the corresponding lower bound.

Inputs

`spc_tab`	a 3-column matrix containing time domain specifications for a reference step. The first column contains the times, second column contains the upper bound and the third contains the lower bounds. The upper and lower bounds are assumed to be LINEARLY interpolated between the values in the second and third columns, respectively.
`w`	the frequency vector, preferably created by logspace.
`dt`	`t`he time increment.
`plt`	a logic variable for switching on and off plotting. The default value is 1.
`n`	is the number of grid-points. The default value is `n`=[10,10,10,3] and the number of grid-points becomes 'prod(`n`)'. The first grid variable corresponds to 'log10(w0)', the second to 'phi', the third to 'log10(lambda)' and the fourth to 'log10(mu)', where the third order system is given by 1-(s/a+1)/((s/b+1)*(s^2/w0^2+2*z*w/w0+1)), where z=cos(phi), a=lambda*z*w0, b=mu*w0.

See also

RSRS, ODSRS, IDSRS, SPC_OD2, SPC_OD3, SPC_OD31.

SPC_RS2

Purpose

Specification calculation for 2:nd order reference step.

Syntax

```
[spec w,spec t,tab]=spc rs2(spc tab,w,dt,plt,n)
```

Description

Sub-function called by `rsrs`. Can be used separately for advanced use. The function calculates frequency domain specifications given time domain specifications for a reference step. The calculations are done by gridding the parameters of a second order system.

Outputs

`spec_w`	Three-column matrix, where the first column contains the frequency vector, the second column contains the upper bound and the third column contains the lower bound.
`spec_t`	Three-column matrix, where the first column contains the time vector, the second column contains the upper envelope of the step responses and the third column contains the lower envelope.
`tab`	Three-column matrix, where the first column contains the time vector corresponding to the time-domain specifications. The second column contains the upper bound corresponding to the time-domain specifications and the third column contains the corresponding lower bound.

Inputs

`spc tab`	a 3-column matrix containing time domain specifications for a reference step. The first column contains the times, second column contains the upper bound and the third contains the lower bounds. The upper and lower bounds are assumed to be LINEARLY interpolated between the values in the second and third columns, respectively.
`w`	the frequency vector, preferably created by logspace.
`dt`	the time increment.
`plt`	`a` logic variable for switching on and off plotting. The default value is 1.
`n`	the number of grid-points. The default value is `n`=[40,40] and the number of grid-points becomes 'prod(`n`)'. The first grid variable corresponds to 'phi' and the second, to 'x', where the second order system is given by 1/(s^2/w0^2+2*z*s/w0+1) where w0=10^x, and z=cos(phi).

See also

RSRS, ODSRS, IDSRS, SPC_RS3, SPC_RS31.

SPC_RS3

Purpose

Specification calculation for 3:rd order reference step.

Syntax

```
[spec w,spec t,tab]=spc rs3(spc tab,w,dt,plt,n)
```

Description

Sub-function called by `rsrs`. Can be used separately for advanced use. The function calculates frequency domain specifications given time domain specifications for a reference step. The calculations are done by gridding the parameters of a third order system.

Outputs

`spec_w`	Three-column matrix, where the first column contains the frequency vector, the second column contains the upper bound and the third column contains the lower bound.
`spec_t`	Three-column matrix, where the first column contains the time vector, the second column contains the upper envelope of the step responses and the third column contains the lower envelope.
`tab`	Three-column matrix, where the first column contains the time vector corresponding to the time-domain specifications. The second column contains the upper bound corresponding to the time-domain specifications and the third column contains the coresponding lower bound.

Inputs

`spc_tab`	a 3-column matrix containing time domain specifications for a reference step. The first column contains the times, second column contains the upper bound and the third contains the lower bounds. The upper and lower bounds are assumed to be LINEARLY interpolated between the values in the second and third columns, respectively.
`w`	the frequency vector, preferably created by logspace.
`dt`	the time increment.
`plt`	a logic variable for switching on and off plotting. The default value is 1.
`n`	the number of grid-points. The default value is `n`=[10,10,10,10] and the number of grid-points ecomes 'prod(`n`)'. The first grid variable corresponds to '2*pi/p', the second to '2*pi/b', the third to 'log10(u)' and the fourth to 'log10(v)', where the third order step-response is given by 1-alpha*exp(-b*t)-beta*exp(-p*t)*sin(ws*t+Phi) where alpha=(u-v)/sqrt(2)+1, beta=(u+v)/sqrt(2). ws and Phi are the corresponding frequency and phase respectively.

See also

RSRS, ODSRS, IDSRS, SPC_RS2, SPC_RS31.

SPC_RS31

Purpose

Specification calculation for 3:rd order reference step (alternative grid).

Syntax

`[spec_w,spec_t,tab]=spc_rs31(spc_tab,w,dt,plt,n)`

Description

Sub-function called by rsrs. Can be used separately for advanced use. The function calculates frequency domain specifications given time domain specifications for a reference step. The calculations are done by gridding the parameters of a third order system usinga different gridding than that used in spc_rs3.

Outputs

`spec_w` Three-column matrix, where the first column contains the frequency vector, the second column contains the upper bound and the third column contains the lower bound.

`spec t` Three-column matrix, where the first column contains the time vector, the second column contains the upper envelope of the step responses and the third column contains the lower envelope.

`tab` Three-column matrix, where the first column contains the time vector corresponding to the time-domain specifications. The second column contains the upper bound corresponding to the time-domain specifications and the third column contains the corresponding lower bound.

Inputs

`spc_tab` a 3-column matrix containing time domain specifications for a reference step. The first column contains the times, second column contains the upper bound and the third contains the lower bounds. The upper and lower bounds are assumed to be LINEARLY interpolated between the values in the second and third columns, respectively.

`w` the frequency vector, preferably created by logspace.

`dt` the time increment.

`plt` a logic variable for switching on and off plotting. The default value is 1.

`n` is the number of grid-points. The default value is `n`=[10,10,10,3] and the number of grid-points becomes 'prod(`n`)'. The first grid variable corresponds to 'log10(w0)', the second to 'phi', the third to 'log10(lambda)' and the fourth to 'log10(mu)', where the third order system is given by (s/a+1)/((s/b+1)*(s^2/w0^2+2*z*w/w0+1)), where z=cos(phi), a=lambda*z*w0, b=mu*w0.

See also

RSRS, ODSRS, IDSRS, SPC_RS2, SPC_RS3.

Purpose

Graphical updating of a freq domain specification in a specification file.

Syntax

```
[spec]=spcupd(specfile,specname,newaxis);
```

Description

Instructions to use this command are given in the figure window. Pressing return twice takes you back to the Matlab command window.

Outputs

`spec`	Updated specification matrix, see remark.

Inputs

`specfile`	Specification file name (string with or without extension '.spc').
`specname`	Name of the specification, string, e.g. `'rsrs'`, `'odsrs'`.
`newaxis`	Optional axis vector in Matlab notation of the specification updating figure.

Remarks

The format of `spec` is columnwise:
For `spec` with one specification (e.g. sensitivity specification), [frequency (rad/s), specification (db)] for `spec` with two specifications (e.g. servo specification), [frequency (rad/s), upper_spec (db), lower_spec (db)]
For `spec` with more than two user defined specifications: [frequency (rad/s), user_spec1, user_spec2, ...]

Since the specification file specfile is updated, the user is advised to make a copy before invoking this command.

WARNING: As all Matlab graphical commands using Matlab's zoom, this is a fragile command. Some mouse action or combination of mouse actions may cause Matlab to abort. You are advised to use the full screen figure window, zoom once if necessary, complete all your updating actions and quit (press return twice).

Example

```
spcupd('ex5_1b','odsrs',[0.1 100 -30 10]);
% Update the frequency domain odsrs specification
% in the file ex5 1b.spc in a magnitude Bode plot
% whose axis are [0.1 100 ] rad/s and [-30 10] dB.
```

See also

MAKESPC ADD2SPC SHOWSPC RSRS ODSRS IDSRS.

SPECIF

Purpose

Simulink block diagram for user's simulations.

Syntax

```
[ret,x0,str]=specif(t,x,u,flag);
```

Description

User: this function is provided as a service only. There is no guarantee or support of it.

M-file description of the SIMULINK system named SPECIF. The block-diagram can be displayed by typing: SPECIF.
SYS=SPECIF(T,X,U,FLAG) returns depending on FLAG certain system values given time point, T, current state vector, X, and input vector, U.

FLAG is used to indicate the type of output to be returned in SYS. Setting FLAG=1 causes SPECIF to return state derivatives, FLAG=2 discrete states, FLAG=3 system outputs and FLAG=4 next sample time. For more information and other options see SFUNC.

Calling SPECIF with a FLAG of zero:
[SIZES]=SPECIF([],[],[],0), returns a vector, SIZES, which contains the sizes of the state vector and other parameters.

SIZES(1) number of states
SIZES(2) number of discrete states
SIZES(3) number of outputs
SIZES(4) number of inputs.

For the definition of other parameters in SIZES, see SFUNC.

See also

TRIM, LINMOD, LINSIM, EULER, RK23, RK45, ADAMS, GEAR.

STDFON

Purpose

Change the fonts used in standard plots.

Syntax

`stdfon(fname,sz)`

Inputs

`fname`	a Matlabfont name.
`sz`	a vector of two elements, [label_size title_size];

TPL2MAT

Purpose

Converts a template file from tpl-format to a matrix.

Syntax

```
[matrix]=tpl2mat(tplf)
```

Outputs

`matrix` has the form [`wtpl nom templ`], with

- `wtpl` column vector of template frequencies [rad/s];
- `nom` column vector of nominal frequency function, where `nom`(i) is the nominal for `wtpl`(i), on the form deg + j*dB;
- `templ` template matrix, where row number i is the template for `wtpl`(i) padded with NaN such that all rows have the same number of elements as the longest template, and each non-NaN element is on the form deg + j*dB.

Inputs

`tplf` template file name, without '.tpl', but within ''.

Remarks

If `tplf` includes standard names, then

```
matrix =   [w_tpl(1) nom(w_tpl(1)) [t_w1 NaN ...];
           w_tpl(2) nom(w_tpl(2)) [t_w2 NaN ...];
           ...
           w_tpl(n) nom(w_tpl(n)) [t_wn NaN ...]]
```

Note that the nominal is converted only for the template frequencies. If there is no nominal value for some template frequency, NaN is included instead. The full nominal frequency vector and full nominal frequency funcction may be extracted with the command GETFROM or GETTPL.

Examples

```
mex2_1a=tpl2mat('ex2_1a');
```

extracts the template frequencies, the templates, and the nominal at those template frequencies where it exists.

```
[wnom,nom]=getfrom('ex2_1a.tpl','w_nom','nom');
```

extracts the nominal frequency vector `w_nom`, and the nominal frequency function nom from `ex2_1a.tpl`, if they exist.

See also

MAT2TPL GETFROM GETTPL.

TPLCASC

Purpose

Creates equivalent template for 2nd step in cascaded design.

Syntax

```
tplcasc(Pcasc,P1,P2,G1,w)
```

Description

Combines templates `P1`,`P2` and controller `G1` into template `Pcasc`, in order to fit the 2nd step of cascaded design

Inputs

`Pcasc`	Name of file to save the new templates in, string, with or without '.tpl'.
`P1`	Name of outer plant template file, string, with or without '.tpl'.
`P2`	Name of inner plant template file, string, with or without '.tpl'.
`G1`	Outer feedback compensator controller file, string, with or without '.m'.
`w`	Frequencies for which the new templates are to be computed. Default or `w`=[]: all frequencies in `P2`.

Remarks

All combinations of the templates are created, especially for CBND to work with the criterion functions `fcasc s`, `fcasc r`.

Note: this is a taylor made command for the standard cascaded SISO design with criteria functions `fcasc_s` and `fcasc_r`. The user should study these files carefully, and create her own functions, if the design problem is non-standard.

Note: the templates are stored in complex form, not the usual Nichols form.

Example

```
tplcasc('Pcasc','P1','P2','g1');
```

Combine, for all template frequencies, the templates of the outer plant template file `P1.tpl`, the templates of the inner plant template file `P2.tpl`, and the output feedback compesator function `G1.m` into a template suitable for the calculatation of cascaded bounds for inner loop design. The resulting templates are stored in complex form in the template file `Pcasc.tpl`.

TPLFOP

Purpose

General template operation function, operates on template files.

Syntax

`tplfop(tplfile,op,w,P,H,G,F,prune on,Tacc)`

Description

This function multiplies templates, closes loops, and a lot of other operations on template files, controller files and scalar complex numbers. Typical applications are multiplications of templates, or calculating different closed loop transfer function. It allows you to also consider sensor dynamics (H).

Inputs

`tplfile`	a string containing the name of the resulting template file, with or without '.tpl'.
`op`	the operation to be performed, predefined are `'+'` P+H+G+F. `'-'` P-H-G-F. `'*'` P*H*G*F. `'/'` P/H/G/F. `'iosrs'` GPH/(1+GPH) sensor noise gain, complementary sensitivity. `'idsrs'` P/(1+GPH) input disturbance to plant output. `'odsrs'` 1/(1+GPH) output disturbance to plant output, sensitivity. `'odsrsc'` GH/(1+GPH) output disturbance to control output/plant input. `'rsrs'` FGP/(1+GPH)) servo transfer function.
`w`	the frequncies for which to perform the operation, default is the intersection of the frequencies in the template files `P` and `H`. If `P` (`H`) is empty then the frequencies are taken from `H` (`P`).
`P,H`	template file name (string with our without extension), controller file name (with extension '.m'), or a scalar (in complex form). Default: If `P`, `H`, is not given or is [], it is assigned the scalar 0 if the operation `op` includes `+,-`, else 1.
`G,F`	controller file name (with or without extension '.m'), or a scalar (in complex form). Default: If `G, F`, is not given or is [], it is assigned the scalar 0 if the operation `op` includes `+,-`, else 1.
`prune_on`	prune option: 0 = no pruning, or tpl-reduction (default) 1 = prune and preform a tplreduc operation 2 = prune only 3 = `tplreduc` operation only
`Tacc`	the 2-norm accuracy of prune and tplreduc operations, [deg dB]. Default [5 5].

Remarks

One of `P` and `H` must be a template file.

Also the nominal case is calculated, and stored.

IMPORTANT: `tplfop` first transforms all templates (that, as you know, all are in in Nichols form) to Nyquist form by the Qsyn function `n2c`, and `op` above relates to the Nyquist form. Note that controller functions always are in Nyquist form. For the DEFAULT operands above, the resulting template is transformed back to Nichols form. For user defined operations, the result is stored in Nyquist form, unless `c2n` is included in the user's expression, e.g. 'c2n(A.^2)'.

`op` = 'A.*A' is identical to `op` = 'A.^2' i.e. elementwise multiplication between the elements of the templates in, say, the template file 'tpl1.tpl'.

If you wish all combinations, write
`tplfop(tplfile,'*',[], tpl1, tpl1)`.

See TPLOP for more information how the expression in `op` is evaluated.

Example

```
tplfop('cloop','rsrs',[],'plant','sensor','g1',...
'f1')
```

calculates the rsrs transfer function and stores it in `'cloop.tpl'`.

See also

TPLOP.

TPLINF

Purpose

Displays on screen information about a template file.

Syntax

`tplinf(tplf)`

Description

Displays original plant description file name and template computation method (this information may be obsolete if the template file was changed subsequently), template frequencies, template names, and sizes.

Inputs

`tplf` template file name, string with or without '.tpl'.

Example

`tplinf('ex2 1a')`

prints the following useful information on screen (notice that the space after `w` in the template name is spurious and due to Matlab's string formatting):

```
 QPlant file :  ex2_1a   Method : rff_[1,1]
+-----------+----------------+---------+
|   Freq    |    template    |   size  |
+-----------+----------------+---------+
|   0.200   |     t_w 1      |    51x1 |
|   0.500   |     t_w 2      |    77x1 |
|   1.000   |     t_w 3      |   113x1 |
|   2.000   |     t_w 4      |   147x1 |
|   5.000   |     t_w 5      |   326x1 |
|  10.000   |     t_w 6      |   181x1 |
|  20.000   |     t_w 7      |    99x1 |
|  50.000   |     t_w 8      |    60x1 |
+-----------+----------------+---------+
Number of Differentiators : 0
```

TPLOP

Purpose

Low-level template operation routine, such as +, *.

Syntax

```
[T]=tplop(expr,A,B,C,D);
```

Description

Low-level template operation routine. Performs user defined template operations on two templates A and B. Useful for tree structured uncertainties.

Outputs

`T`	resulting template vector.

Inputs

`expr`	a string containing an expression in A,B,C,D of the operation to be performed . Note upper case of A,B
`A,B`	Template vectors
`C,D`	Complex scalars.

Remarks

It is the users responsibility to know if the templates are in Nichols or Nyquist form. The deafalt is Nichols form.
Note that the size of the output is equal to the product of the size of `A` and `B` respectively. This means that the result can be a very large vector. It is often worthwile to prune the resulting vector, if this is possible.

Examples

```
tpl3=tplop('A+B',tpl1,tpl2)
```

calculates the complex MULTIPLICATION of the two value sets `tpl1` and `tpl2`, if they are in Nichols form (deg+j*dB).

```
tpl3=tplop('c2n(A+B)',n2c(tpl1),n2c(tpl2))
```

calculates the complex ADDITION of the two value sets `tpl1` and `tpl2`, provided they are given in in Nichols form.

```
FT=tplop('c2n(C.*B.*A/(1+B.*A))',n2c(P),n2c(G),...
n2c(F))
```

calculates `FPG/(1+GP)` provided `P`,`G`,`F` are in Nichols form.

TPLPRUNE

Purpose

Prunes templates in a template file.

Syntax

`tplprune(outfile,infile,Tacc,w)`

Inputs

`outfile`	The name of the template file to save the new templates in, string with or without extension '.tpl'.
`infile`	The existing template file name, string with or without extension '.tpl'. Default is outfile.
`Tacc`	The required pruning 2-norm accuracy in Nichols form, [degree_accuracy , dB_accuracy]. A larger value gives a smoother appearance.
`w`	The frequencies to prune, default or `w`=[] means the pruning of all templates in infile

Remarks

The parameter matrix of the retained cases after pruning is saved in outfile.

If outfile does not equal infile, it is the user's responsibilty to copy w_nom, nom, par_nom to outfile, if needed. It is often easier to copy the whole templatefile first and then perform TPLPRUNE with `infile==outfile`.

Example

```
tplprune('ex4_4e',[],[10 2]);
```

All templates in `ex4_4e.m` are pruned to the accurace 10 deg and 2 dB.

See also

PRUNE.

TPLREDUC

Purpose

Reduces/interpolates points in a sorted and ordered template.

Syntax

```
[Treduced]=tplreduc(T,Tacc,acc,interpol)
```

Description

This function reduces/interpolates points in a SORTED template, e.g. one produced by adgrid or adedge, or any template that has been pruned.It also works on a template produced by rff.
The algorithms remove points if the accuracy loss is less than `acc*Tacc`. If the distance between two consecutive points is larger than `Tacc`, then it adds extra points by linear interpolation.

Outputs

`Treduced`	new template, row vector in Nichols form, deg + j*dB.

Inputs

`T`	template, row vector in Nichols form, deg + j*dB. It must be ORDERED or PRUNED, and it does not handle phase wrapping! If the template is wrapped (on one Riemann surface) use QUNWRAP to wrap it before attempting TPLREDUC;
`Tacc`	The accuracy distance [deg dB] (2-norm)
`acc`	relative accuracy, the relative tolerance of points to be removed. Default: `acc`=0.1;
`interpol`	If interpol is 1 (default) interpolation takes place, otherwise not.

Example

```
redrff5=tplreduc(rff5,[10 2]);
                TPLREDUC:size reduction ratio 91.11%
```

The template border of the rff computed template vector `rff5` is thinned/complemented to a 2-norm accuracy of at least 10 deg and 2 dB, with the result placed in the vector `redrff5`.

See also

TPLFOP TPLPRUNE.

TPLUNION

Purpose

Computes the union of the templates of two template files.

Syntax

```
tplunion(tplnew,tpl1,tpl2,w);
```

Description

This function calculates the union of the templates, and associated parameter vectors (if they exist), in `tpl1` and `tpl2` and stores them in `tplnew`. If the optional frequency vector w is given, only the templates for those frequencies are stored. The nominal case, and nominal frequency vector, is taken from `tpl1`.

Inputs

`tplnew`	name of the new template file, string with or without '.tpl'
`tpl1,tpl2`	names of the two original template files, strings with or without '.tpl'
`w`	vector of frequencies of templates to be saved in `tplnew`. Default: all frequencies in `tpl1` and `tpl2`.

Remarks

If for a given frequency only one of the templates exist, then that template (and parameter matrix) is inserted into `tplnew`.

Example

```
tplunion('ex4 7','ex4 5','ex4 5ag');
```

The union of the templates in `ex4_5.tpl` and `ex4_5ag.tpl`, for all fequencies is saved in the template file `ex4_7.tpl`

See also

TPLFOP.

TPLUPD

Purpose

Interactive, graphical template updating.

Syntax

```
tplupd(tplfile,Tacc,w)
```

Description

This command lets you interactively update each template, by adding points, deleting points and, by pruning subsets of points.

Add Points: Press the LEFT mouse button, keep it pressed and draw a line, release the button. If you press return, new points will be added by linear interpolation along the line, with accuracy `Tacc`.

Delete Points: Press the RIGHT mouse button, keep it pressed and draw a rectangular area by stretching the diagonal. Release the button. If you press return, all points in the chosen area will be deleted.

Prune points: DOUBLECLICK the LEFT mouse button, keep it pressed, and draw a rectangular area by stretching the diagonal. Release the button. If you press return, all points in the chosen area are pruned with accuracy `Tacc`.

Regret: You may regret an action before you have pressed return: Release the button if it is pressed, and choose another action.

Continue: An extra return takes you to the template of the next frequency.

Quit: When prompted in the Matlab Command Window, press 'q' on the keyboard to quit.

Inputs

`tplfile`	Name of the tpl-file to update and to save the new templates in, string valued, with or without the extension '.tpl'
`Tacc`	the accuracy (2-norm) for adding points and for pruning, [deg dB]. Default is [5 5]
`w`	vector of frequencies [rad/s] for which the templates are to be updated. Default or `w`=[] means all templates in the template file.

Remarks

Using TPLUPD destroys the parameter structure of the templates: All parameters are kept as they were before tplupd was performed

To keep the original templatefile, it is wise to first copy the whole template file and then run TPLUPD on the copy.

Example

```
tplupd('ex4_7',[10 2]);
```

All templates in `ex4_7.tpl` are displayed for updating with an accuracy of 10 deg and 2 dB.

See also

TPLPRUNE TPLREDUC TPLFOP.

UDBND

Purpose

Recalculates a previosly calculated bound, subroutine to BNDUPD.

Syntax

```
[bound]=udbnd(tpl nom,tpl,specfun,spec,oldbnd,...
new_acc,old_acc,par_nom,par,title)
```

Outputs

`bound` updated bound.

Inputs

`tpl nom` nominal plant, scalar in Nichols form.

`tpl` template, value set, vector in Nichols form.

`specfun` name of specification function. ex: `specfun='frsrs'.` The specification is satisfied for all values of GPnom such that frsrs(tpl_nom,tpl,GP, spec)<0. See the m-file frsrs.m for an example.

`oldbnd` old bound to be updated.

`new_acc` new accuracy. i.e. density of the grid in the Nichols chart.

`old_acc` old accuracy on previously calculated bound.

`par nom, par`
optional parameters to specification function, typically the uncertain physical parameters of the templates, also useful for SIMO, MISO and MIMO systems.

`title` title of optional plot window for displaying the results. If `title`==[] or not given, plotting is supressed.

UPDBND

Purpose

Refining of a previously calculated bound, subroutine to BNDUPD.

Syntax

```
[bound]=updbnd(tpl_nom,tpl,specfun,spec,oldbnd,...
new_acc,par_nom,par,figname)
```

Outputs

`bound` refined bound.

Inputs

`tpl_nom` nominal plant, scalar in Nichols form.

`tpl` template, value set, vector in Nichols form.

`specfun` name of specification function. ex: `specfun='fodsrs'`.

`oldbnd` old bound to be updated.

`new_acc` new accuracy. i.e. how dense the grid on the Nichols chart should be.

`par_nom, par`
optional parameters to specification function, typically the uncertain physical parameters of the templates, also useful for SIMO, MISO and MIMO systems.

`figname` name of figure window (optional).

UPDBND1

Purpose

Subroutine, script m-file, only for use by UPDBND.

Syntax

```
updbnd1
```

UPDSPC

Purpose

Subroutine for interactive specification updating in SPCUPD.

Syntax

```
[new spec]=updspc(spec,figname, new axis);
```

Inputs

`spec`	the specification to update, in correct standard format, see `add2spc`.
`figname`	optional figurename.
`new_axis`	optional figure axis, see SPCUPD.

UPDSPC1

Purpose

Subroutine, script m-file only to be used by UPDSPC.

Syntax

```
updspc1
```

UPDTPL

Purpose

Subroutine for interactive template updating in TPLUPD.

Syntax

```
[tpl]=updtpl(tpl,tpl nom,Tacc,figname)
```

Outputs

`tpl`	template in Nichols form, deg + j*dB.

Inputs

`tpl`	template in Nichols form.
`tpl_nom`	nominal plant frequency function value, in Nichols form.
`Tacc`	accuracy for adding and pruning points, see TPLUPD.
`figname`	optional current window caption, string.

UPDTPL1

Purpose

Subroutine, script m-file, only for use by UPDTPL.

Syntax

```
updtpl1
```

WRAP

Purpose

Wraps angles of a matrix in Nichols form into desired Riemann surface.

Syntax

```
[y]=wrap(x,deg)
```

Outputs

`y`	matrix in Nichols form on the Riemann surface [deg-360,deg] degrees.

Inputs

`x`	matrix in Nichols form over one or more Riemann surfaces, assuming than min(arg(x)) > - 5*360 deg. (If this assumption is violated, the user is invited to change '5' in the code below to any desired larger integer.)
`deg`	right angular limit, degrees, of the Riemann surface that spans [deg-360,deg]

ZBOX

Purpose

Graphic utility for figure window (subroutine).

Syntax

```
zbox(flag)
```

Description

subroutine to HZOOM. Not to be used by user.