3. QSYN FLOW CHART, COMMAND OVERVIEW, AND DATA STRUCTURES

In Section 2.10 a summary of the commands for a SISO design was given. The basic design procedure, for a SISO system (see Figure 1.1), can be compactly condensed in the flow chart in Figure 3.1.

First user input: Plant description

If the user has an analytic transfer function expression for her uncertain plant, she may formalize it, together with various computational options in a *Plant description file*, a commented model of which is present in the Qsyn library under the name `plant.m`. How to do it is well explained in Section 2.1. With the help of the command `ctpl`, the plant description is transformed to a *template file*, a Matlab mat-file with extension `tpl`, that contains templates for user selected frequencies.

Template computations are discussed in Sections 2.1.4 and Section 2.2, with further examples in Chapter 4. If the uncertain plant is described with the help of a set of measured, or otherwise collected (e.g. simulated) transfer functions, the commands `mffd` (make template from frequency function description) or `mat2tpl` (matrix-to-template) facilitate the creation of the corresponding template file; see the example in Section 4.5. Several plant descriptions may contribute to the same template file, and even to the same template, see the example in Section 4.7. The template file structure, and a list of the commands that operate on template files are given below in Section 3.2, including the command `tplfop`, that is used to generate the template file of the complementary sensitivity function, *PG*/(1+*PG*).

Second user input: Loop specifications

The second user input is the closed loop specifications. Qsyn includes commands, such as `rsrs`, `iosrs`, `odsrs`, etc, that help the user to approximately translate, to the frequency domain, step response specifications for members of a set of *predefined* closed loop transfer functions. The time and frequency domain specifications are collected as variables in a *specification file*, a Matlab mat-file with extension `spc`. Of course the user may insert a frequency domain specification directly into the specification file. Examples are given in Section 2.3.

If the the user includes a specification that does not refer to one of the predefined transfer functions, then she must also write an m-file executing the computation of the actual c*riterion function* to serve in the computation of the Horowitz bounds. See `fuser.m` in the Qsyn library. Examples of criteria functions for non-predefined specifications are found in Section x.x. Below, in Section 3.3, the commands to generate specifications of the predefined type, and their criteria functions are listed, together with an explanation of the specification file structure and the commands that operate on them.

Resulting bounds

Based on a plant template file, a specification file and its corresponding criterion function m-file (which may be predefined such as `frsrs.m`, or user written such as `fuser.m` (to be copied by the user into his working directory, edited and given another name, into e.g. `fuser1.m`), the command `cbnd` computes the Horowitz bounds for the template frequencies, and collects them into a Matlab mat-file with extension `bnd`, called a *bound file*. Two examples are given in Section 2.4. The bound file structure, and a list of the commands that operate on bound files are given below in Section 3.4, including the command

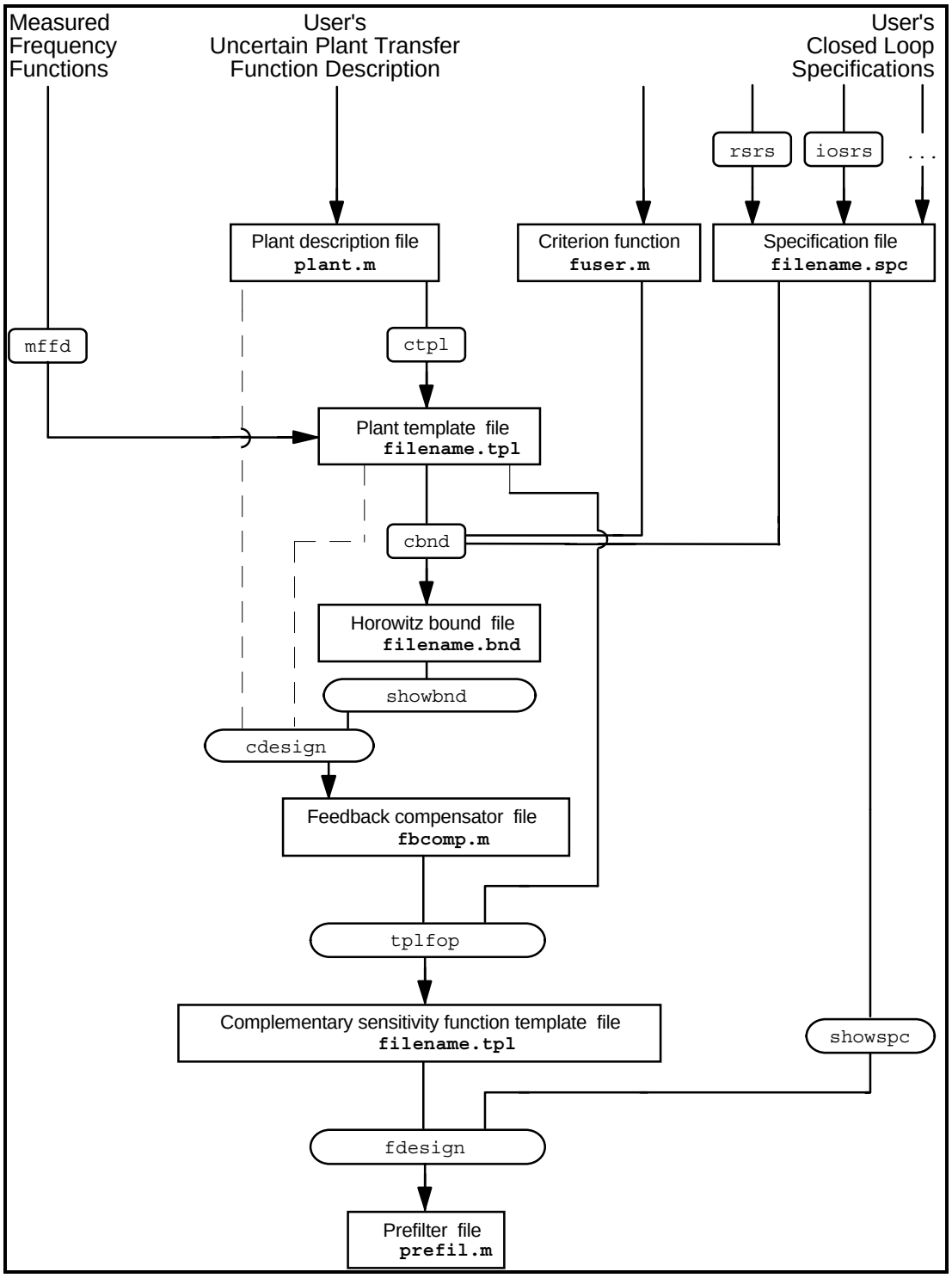

Figure 3.1. ***Basic Qsyn Flowchart****. Qsyn commands are placed in boxes with rounded vertices, with inputs entering from above and outputs marked with arrows. Qsyn files are placed in square boxes. Standard file names in the Qsyn library, e.g.* `prefil.m`*, are given; other filenames are marked by the generic* `filename`*. The file name extension* `xxx` *denotes the filetype. The user is expected to use his own filenames with the appropriate extension. The dashed lines from the Plant description file and Plant template file denote alternative plant nominal inputs to the* `cdesign` *command. After completing the design, the user is expected to simulate the closed loop in the frequency and time domains. The work flows for Cascaded and MIMO designs contain iterations of this flowchart, see the example of Cascaded design in Chapter 7.*

`showbnd`, that is used to display the bounds in a Nichols chart as a preparation for the feedback compensator design.

Design functions

The result of the design process, the feedback compensator and the prefilter, are user written m-files called *Controller functions* that have to adhere to a defined input-output convention. Two examples of compensator functions, `fbcomp.m` for the feedback compensator, and `prefil.m` for the prefilter, are found in the Qsyn library. These model files are used for Example 2.1 in Section 2.5 and 2.7, respectively. The controller function structure is briefly recapitulated in Section 3.5, together with auxiliary design commands such as `showbnd`, `cdesign`, `showspc`, and `fdesign`. More examples of controller functions, including their use for digital control, are found in Chapter 6

Cascaded and MIMO designs.

The design procedure for Cascaded and MIMO systems is more complicated than that of Figure 3.1. An example of Cascaded SISO design is given in Chapter 7. The user has to write his own application functions for MIMO design.

3.1 Plant description files and related commands

Qsyn plant description files enable a convenient description of parametrically uncertain, continuous time, rational (or irrational) transfer functions, with optional delay and/or multiplicative unstructured uncertainty. Their structure, and file head are explained in Section 2.1, and in the comments of the model file `plant.m` in the Qsyn library, see Figure 2.1. More examples of various plant definition options are given in Chapter 4.

Here follows a list of commands that operate on plant description files or have a plant description file as an input. The commands are divided into two groups. *Primary commands* would be used in the normal course of design. *Auxiliary commands* are either secondary commands that serve as subroutines to the primary commands, or are suitable for special or tailormade applications.

Primary commands for Plant description files	
CASES	plant frequency domain simulation for user selected cases
CCASES	closed loop frequency function simulation for user selected cases
CDESIGN	enables interactive feedback compensator design
EDIT	editing the plant description file with the default editor (Matlab command)
FDESIGN	enables interactive prefilter design
PLANT	model of plant description file to be copied and edited by the user
PLNT	invokes a default editor to edit a plant description file
PMODEL	interactive plant modelling function for measured frequency function data

Auxiliary commands for Plant description files	
BUILDF	extracts transfer fcn factors from the rff string format in a plant file
MULT	unstructured uncertainty fcn for all template comp methods except rff
PLANT_ID	compiles a plant description file and creates an auxiliary m-file
PNOMINAL	computes freq function values of the nominal in a plant description file

Figure 3.2. Commands connected with plant description files.

Remarks

1. The commands `cdesign` and `fdesign` can be used to design compensators and filters for a plant with no uncertainty that is represented as the nominal in a plant description file.
2. The m-file `plant.m` can also be called, and then gives output variables.
3. The command `plant-id(plant)` produces an auxiliary file in the current directory called `~plant.m`. It contains some of the information in `plant.m` in a form more convenient for subsequent commands. Some Qsyn functions, such as `ctpl` use `plant-id` as a subroutine.

3.2 Template files and their commands

Template files are mat-files, with extention `.tpl` that contain the variables listed in Figure 3.3. The commands having template files as inputs or outputs are listed in Figure 3.4.

Template file variables			
horcad_comment	string		information how the template file was generated the first time
filename	string		filename, not including '.tpl'
info	string		information if the file was changed
w_nom	row vector	[rad/s]	nominal frequencies
nom	row vector	[deg+j*dB]	nominal frequency function, with length(nom)=length(w_nom)
par_nom	col vector		parameters of the nominal plant
w_tpl	matrix	[rad/s, index]	1st col: sorted template frequencies 2nd col: index in template name
t_w1, t_w2, t_w3, ...	col vectors	[deg+j*dB]	templates whose indices 1, 2, 3 ... are frequency pointers w r t w_tpl
par_1, par_2, ...	matrices		column *k* in par_1 holds the parameters that gave rise to t_w1(k), ...

Figure 3.3. Template file variables

Remarks

1. Since a template file is a mat-file, all its variables may be loaded into the workspace by the Matlab command `load filename.tpl -mat`.
2. Sometimes spurious variables, e.g. `ans`, enter a template file when it is created or changed. This is a known but benevolent bug that does not influence any computation.
3. The indices of `w_tpl(:,2)` work as follows: Let

```
w tpl = [0.2   2;    0.5    1;     0.7    3];
```

   Then `t_w1` is the template for 0.5 rad/s, `t_w2` the template for 0.2 rad/s, and `t_w3` the template for 0.7 rad/s. Corresponding parameter matrices are `par 1`, `par 2`, and `par 3`.
4. The parameters in `par nom` and the columns of `par 1`, `par 2`, etc are ordered in order of appearance of the uncertain parameters in the vector `Par` of the plant description file from which the template file was computed.
5. The parameter variables `par nom` and `par 1`, `par 2`, etc are not created when the template file is computed with the rff-method or by the command `mffd` or `mat2tpl`.

6. Be careful when changing or adding templates in a template file. Insert e.g. your own comments as a string variable to log the change, since the variable `info` is not always created. In particular, if a template with a parameter matrix is exchanged for a template without (e.g. created by rff) then the old parameter vector is not automatically removed. In particular, make sure that `w_tpl` correctly reflects the templates (user the commands `tplinf` or `gettpl`), and if necessary change it by the commands `gettpl` and `insert`.

Primary commands for Template files	
ADD2TPL	adds/replaces/combines a new template into a template file
CTPL	calculates a template file from data in a plant definition file
GETFROM	lists and/or copies variables from a mat-file
GETTPL	copy a a template or other data variable from a template file
INSERT	inserts/replaces variable in mat-file
LOOK	lists the names of the variables stored in a mat-file
MAT2TPL	converts a matrix to a template file
MFFD	creates template file from a freq vector, nominal, and template matrix
PRUNE	removes interior points from a connected template
REMOVE	removes a variable from a mat-file
SHOWTPL	displays templates from a template file in a Nichols diagram
TPL2MAT	converts a template file from tpl-format to a matrix
TPLFOP	template file operation routines, such as +, *
TPLINF	displays information about the templates in a template file
TPLOP	single template operation routines, such as +, *
TPLPRUNE	prunes templates in a template file
TPLREDUC	reduces/interpolates points in a sorted and ordered template
TPLUNION	computes the union of two template files
TPLUPD	interactive, graphical template updating

Auxiliary commands for Template files	
ADEDGE	calculates templates with the recursive edge grid method
ADGRID	calculates templates with the recursive grid method
CLTMP	interpolates points in a sorted and pruned template
FVALUE	get template from tpl-file or freq fcn value from controller file (subroutine)
ISIN	checks if a variable exists in a mat-file
LOAD	loads the variables into the work space from a mat-file
MULT	unstructured uncertainty fcn for all template comp methods except rff
PRUNE1	test version of PRUNE that displays each step of the iteration
RECEDGE	subroutine used by the template computation functions ADEDGE
RECGRID	subroutine for the template computation function ADGRID
RFFCPZ	computes a complex pole/zero pair template in real factored form
RFFEL	pure gain, delay, unstructured uncertainty, or integrators rff template
RFFMUL	multiples two templates in real factored form
RFFPZ	produces a real pole or real zero template in real factored form
RFFUTIL1	utility function for RFFCPZ to compute template edge points
RFFUTIL3	utility function for RFFCPZ to sort, clean, and complement edges
TPLCASC	creates equivalent template for 2nd step in cascaded design
UPDTPL	subroutine for interactive template updating in TPLUPD
UPDTPL1	subroutine, script m-file, only for use by UPDTPL

Figure 3.4. Commands connected with templates and template files.

3.3 Specification files and criteria functions, and their commands

3.3.1 Specification files

Specification files are mat-files, with extention `.spc` that contain time domain and/or frequency domain envelopes into which some closed loop frequency response must fit. A separate m-file, a so called criterion function, with a given structure defines how the closed loop frequency response is to be computed. The variables of a specification file are presented in Figure 3.5, and a criteria functions are displayed in Figure 3.6.

Specification file variables with default names			
comment	string		information how the specification file was generated the first time
filename	string		filename, including '.spc'
specname_tab	matrix	[sec, max, min]	Original time domain specification for relevant specification variables 1st col: time 2nd/3rd col: upper/lower spec
idsrs_t idsrs_w	matrix matrix	[sec, value] [rad/s,dB]	**Input Disturbance Step Response Spec** $P/(1+L)$ 1st col: time/frequency 2nd col: upper envelope Created by command **idsrs**
odsrs_t odsrs_w	matrix matrix	[sec, value] [rad/s,dB]	**Sensitivity** $1/(1+L)$ **Output Disturbance Step Response Specification** 1st col: time/frequency 2nd col: upper envelope Created by command **odsrs**
rsrs_t rsrs_w	matrix matrix	[sec, max, min] [rad/s,dBmax,dBmin]	**Tolerance Specification** $FL/(1+L)$ **Reference Step Response Spec** 1st col: time/frequency 2nd col: upper envelope 3rd col: lower envelope Created by command **rsrs**

Specification file variables with other names			
iosrs_t iosrs_w	matrix matrix	[s, max, min] [rad/s,dBmax,dBmin]	**Complementary Sensitivity** $L/(1+L)$ **Input Output Step Response Spec for one degree-of-freedom systems** 1st col: time/frequency 2nd col: upper envelope 3rd col: lower envelope May be created by cmnd **rsrs**
odsrsc_t odsrsc_w	matrix matrix	[s, value] [rad/s,dB]	**Output Disturbance Step Response Spec at Controller Output** $G/(1+L)$ 1st col: time/frequency 2nd col: upper envelope May be created by commands **idsrs**/**odsrs**
userspec1_t userspec1_w	matrix matrix	[sec, val1, val2, ...] [rad/s, val1, val2, ...]	**User defined specification # 1**
... 	matrix matrix	[sec, val1, val2, ...] [rad/s, val1, val2, ...]	**User defined spec # 2, 3, ...**

Figure 3.5. Specification file variables

Remarks

1. Since a specification file is a mat-file, all its variables may be loaded into the workspace by the Matlab command `load filename.spc -mat`.
2. Not all variables have to be present in one or any specification file: the user decides which variables are relevant, and in which specification file they belong.
3. Variables ending with `t` are time domain specifications. Variables ending with `w`, are frequency domain specifications. Only frequency domain specifications are used in the bounds computations, see Chapters 1 and 2. The time domain specifications are optional for the design, and used for comparison with the (simulated) closed loop responses.
4. The specifications `idsrs`, `odsrs`, and `rsrs` are predifined in the sense that Qsyn includes commands with the very same names that aid the user to describe the time domain specifications and approximately translate them to the corresponding frequency domain specifications. The corresponding criteria functions, with the names `fidsrs`, `fodsrs`, and `frsrs` are also included in Qsyn, see below in Section 3.3.3.
5. The specifications `iosrs`, and `odsrsc` are predefined in the sense that their respective criteria functions, `fiosrs`, and `fodsrsc` are included in Qsyn. There are however no dedicated commands to create these time and frequency domain specification variables. As indicated in Figure 3.5, other existing commands may be utilized.
6. The user has to make sure that the specification frequency range spans the template frequencies, so that the bound computation routine may find a specification for each template frequency, if necessary by interpolation.
7. User frequency specifications may have arbitrary many columns, with the convention that the first column denotes rad/s. The units may be arbitrary, e.g. phase units. The specification variable names are chosen by the user. An existing criterion function may be used for the ensuing bound computation, if appropriate, else the user could write a new criterion function, see below.
8. When using the commands `idsrs`, `odsrs`, and `rsrs` to create or update a specification file, the computed specification variables will get the default name (= the name of the commmand) if not assigned otherwise in the command line.
9. The user may create her own frequency domain specification variable in several ways: the following way: *i)* by the command `makespc` whereby specifications are entered graphically; *ii)* computing a specification variable with one of the commands `idsrs`, `odsrs`, or `rsrs`, and then updating it graphically with the command `spcupd`; *iii)* with the command `add2spc` that enables the insertion (or replacement) of a constant specification, or a specification given as a filter amplitude function, or an arbitrary specification in matrix form.
10. Note that unless otherwise demanded in the command `cbnd`, the names of the bounds in a bound file computed with the use of a specification variable called `name` will be called `name_1`, `name_2`, `name_3`, ..., where the indices 1, 2, 3, ..., are pointers to the bound frequency vector `name w` in the bound file, see Section 3.4.
11. "`specname`" in `specname tab` can be any relevant specification variable name, such as `rsrs`.

3.3.2 Criteria functions

A criterion function (or specification function) is realized in an m-file, and computes the feedback controller dependent quantity to be compared with a specification variable during the bounds computation.

Some criteria functions are predifined, such as `fiosrs.m`, `fodsrsc.m`, `fidsrs.m`, `fodsrs.m`, and `frsrs.m`. These criteria functions are meant to operate on the frequency specification variables `iosrs w`, `odsrsc w`, `idsrs w`, `odsrs w`, `rsrs w`, respectively, as defined in Figure 3.5. They may of course operate on any other frequency specification variable that has the correct number of columns.

For the 2nd step of cascaded design there are two special predefined criteria functions: `fcasc r.m`, and `fcasc s.m`. They operate on specifications of the type `rsrs w` and `odsrs w`, respectively. See Chapter 7

The user may let an existing specification variable, say `rsrs_w,` be evaluated by a user-written critererion function, say `fuser.m`, during the bound computation. She may also introduce a new specification variable, say `lirt w`, into an existing or new specificaiton file, and let it be evaluated by an predefined criterion function, say `fodsrs.m` during the bound computation. Of course she may introduce her new specification variable, `itta_w`, and let it be evaluated by her new criterion function. She is then advised (but not obliged) to follow the the name convention to call her criterion function `flirt.m`. The bound variable names will in any case be named after the specification variable: `lirt_1`, `lirt_2`, `lirt_3`, etc.

One of the predefined criteria functions in the Qsyn library, `frsrs.m,` is reproduced in Figure 3.6a. A "user written" criterion function, `fuser.m`, is included in the Qsyn library. It is printed in Figure 3.6b and explained below. The user may study the other predefined criteria functions. Note the compulsory structure of the criterion function file head and and its input and output variables.

The existing criteria functions are listed among the commands connected to specification files in Figure 3.7, and among commands connected to bound files in Figure 3.9.

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

%FRSRS      Reference step response specification criterion fcn
%           function[Tmax]=frsrs(tpl nom,tpl,GP,spec,par nom,par)
%
%Criterion fcn for max|GP/(1+GP)|/max|GP/(1+GP)| < spec(1)/spec(2)
%
% Output and input variables are explained in fuser.m
% Note that GP = Lnom

L=n2c(GP+tpl-tpl nom);
Tmax=20*log10(max(abs(L)./abs(1+L)));
Tmin=20*log10(min(abs(L)./abs(1+L)));
Tmax=Tmax-Tmin-spec(1)+spec(2);
```

Figure 3.6a. Predefined criterion function `frsrs.m` *from the Qsyn library. Other predefined criteria functions include* `fiosrs.m`, `fidsrs.m`, `fodsrsc.m`, `fodsrs.m`, *and* `fcasc_r.m` *and* `fcasc_s.m` *for cascaded design.*

```
function[Tmax]=fuser(tpl_nom,tpl,Lnom,spec,par_nom,par)

% FUSER      Model file to create user defined specification criteria
%            [Tmax]=fuser(tpl_nom,tpl,Lnom,spec,par_nom,par)
%
% create your own specifications by copying/editing this file.
%
% 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.
%
% ALL INPUT VARIABLES ARE GIVEN BY THE CALLING BOUND COMPUTATION FCN
%
% tpl_nom =  the nominal template point, a scalar i [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 have the same dimension.)
%
% 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 =     the 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 belongs
%            to the frequency for which bounds are now computed).
%
% par_nom = the nominal parameters of the plant (if existing)
%
% par     = parameter matrix that produced the template (if existing)
%
% EXAMPLE 1: 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
%
% EXAMPLE 2: Stiffness coefficient c0=0, c1<=c1spec. Assume that the
% user defined the second column of 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);
%
% OTHER EXAMPLES: The user is advised to study the predefined
% criteria functions FRSRS, FIDRS, FODRS, FODSRSC, FIOSRS

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

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

Figure 3.6b. Predefined criterion function `fuser.m` *from the Qsyn library. Other predefined criteria functions include* `fiosrs.m`, `fidsrs.m`, `fodsrsc.m`, `fodsrs.m`, *and* `fcasc_r.m` *and* `fcasc_s.m` *for cascaded design.*

3.3.3 Commands connected to specification files

Qsyn includes commands that aids the user to define some commonly used specifications. They are `iosrs`, `idsrs`, `odsrsc`, `odsrs`, and `rsrs`, listed among the primary commands for specificaiton files in Figure 3.7. The resulting specification variables get standard names as listed in Figure 3.5. Connected with these are predefined criteria functions that have the same name as the commands, with a preceding "f". The criteria functions are described in Section 3.3.2, and listed among the auxiliary commands in Figure 3.7.

Primary commands for Specification files	
ADD2SPC	adds/replaces specification in a specification file
FUSER	user written criterion function for user defined specification
GETFROM	lists and/or copies variables from a mat-file
IDSRS	computes input disturbance step response specifications
INSERT	inserts/replaces variable in mat-file
LOOK	lists the names of the variables stored in a mat-file
MAKESPC	graphical definition of a specification variable
ODSRS	computation of output disturbance step response specification
REMOVE	removes a variable from a mat-file
RSRS	calculation of reference step response specification
SHOWSPC	displays a specification from a specification file
SPCUPD	graphical updating of a freq domain specification in a specification file

Auxiliary commands for Specification files	
FCASC_R	criterion fcn for rsrs for the 2nd step of cascaded design
FCASC_S	criterion fcn for odsrs or sensitivity of the 2nd step of cascaded design
FIDSRS	criterion fcn for plant input dist step response specification
FIOSRS	criterion fcn for input step response spec/complementary sensitivity
FODSRSC	criterion fcn for output dist step response specification at the plant input
FODSRS	criterion function for output dist step response specification or sensitivity
FRSRS	criterion function for reference step response specification or tolerance
FUSER	criterion function file to be copied and edit for user defined criterion fcn
ISIN	checks if a variable exists in a mat-file
LOAD	loads the variables into the work space from a mat-file
MSPC	auxiliary function to MAKESPC
SPC_ID2	Subroutine used by IDSRS
SPC_OD2	specification calculation for 2:nd order output disturbance step
SPC_OD3	specification calculation for 3:rd order output disturbance step
SPC_OD31	specification calc for 3:rd order output disturbance step (alternative grid)
SPC_RS2	specification calculation for 2:nd order reference step
SPC_RS3	specification calculation for 3:rd order reference step
SPC_RS31	specification calculation for 3:rd order reference step (alternative grid)
UPDSPC	subroutine for interactive specification updating in SPCUPD
UPDSPC1	subroutine, script m-file, only for use by UPDSPC

Figure 3.7. Commands connected with specifications and specification files.

3.4 Bound files and their commands

Bound files are mat-files, with extention `.bnd` that contain the variables listed in Figure 3.8. The commands having bound files as inputs or outputs are listed in Figure 3.9.

Bound file variables			
comment	string		information how the template file was generated the first time
filename	string		filename, including '.bnd'
info	string		information if the file was changed
name1_w	matrix	[rad/s, index]	1st col: frequencies for bound # 1 2nd col: index in bound name
name1_1, name1_2, name1_3, ...	row vectors	[deg+j*dB] or NaN	bounds of type # 1 whose indices 1, 2, 3 ... are frequency pointers w r t name1_w
name2_w	matrix	[rad/s, index]	1st col: frequencies for bound # 2 2nd col: index in bound name
name2_1, name2_2, name2_3, ...	row vectors	[deg+j*dB] or NaN	bounds of type # 2 whose indices 1, 2, 3 ... are frequency pointers w r t name2_w
...			

Figure 3.8. Bound file variables

Remarks

1. Since a bound file is a mat-file, all its variables may be loaded into the workspace by the Matlab command `load filename.bnd -mat`.
2. Sometimes spurious variables, e.g. `ans`, enter a template file when it is created or changed. This is a known but benevolent bug that does not influence any computation.
3. The variables `comment` and `info` are not always present.
4. The indices of `name1 w(:,2)`, `name2 w(:,2)`, `name2 w(:,2)`, ... work as follows: Let

```
name1 w = [0.2    2;    0.5    1;    0.7    3];
```

   Then `name1_1` is the bound for 0.5 rad/s, `name1_2` the bound for 0.2 rad/s, and `name1_3` the bound for 0.7 rad/s. Compare with `w tpl` in Figure 3.3.
5. When the command `cbnd` is called, one type of bounds is computed. Its `name` becomes the same name as the name of the specification variable used for the bound computation, e .g. `odsrs`, unless otherwise stated in the command line. A second call of `cbnd` may be used to replace the bounds of an existing bound type in an existing bound file, include another bound type into the existing bound file, or to create a new bound file with the freshly computed bounds.
6. Be careful when changing bounds in a bounds file, e.g. manually with the command `insert`. Insert e.g. your own comments as a string variable to log the change, since the variables `comment` and `info` may not be updated.
7. How to combine two bounds, i.e. to replace them by a dominant bound defined by the dominant segments from each, is shown in Chapter 5.

Primary commands for Bound files	
ADD2BND	adds/replaces bound in a bound file
BNDINF	displays information about a bound file
BNDUPD	interactive recalculation of bounds with higher accuracy
CBND	calculates bounds from given template file, specification file and criterion
FUSER	user written criterion function for user defined specification
GETBND	copies a bound from a bound file
GETFROM	lists and/or copies variables from a mat-file
LOOK	lists the names of the variables stored in a mat-file
REMOVE	removes a variable from a mat-file
SHOWBND	plots bounds from a bound file for selected frequencies, in a Nichols char

Auxiliary commands for Bound files	
EXTRACT	extracts bounds from a contour given by the command CONTOURC
FCASC_R	criterion fcn for rsrs for the 2nd step of cascaded design
FCASC_S	criterion fcn for odsrs or sensitivity of the 2nd step of cascaded design
FIDSRS	criterion fcn for plant input dist step response specification
FIOSRS	criterion fcn for input step response spec/complementary sensitivity
FODSRSC	criterion fcn for output dist step response specification at the plant input
FODSRS	criterion function for output dist step response specification or sensitivity
FRSRS	criterion function for reference step response specification or tolerance
ISIN	checks if a variable exists in a mat-file
LOAD	loads the variables into the work space from a mat-file (Matlab cmd)
MAKEBND	calculates bounds on a grid in the Nichols chart, subroutine to BNDUPD
UDBND	recalculates a previosly calculated bound, subroutine to BNDUPD
UPDBND	refining of a previously calculated bound, subroutine to BNDUPD
UPDBND1	subroutine, script m-file, only for use by UPDBND

Figure 3.9. Commands connected with bound files.

3.5 Controller functions and their commands

Controller functions are m-files, in which the user designs the feedback compensators and prefilters, and which are used to compute their frequency functions. The Qsyn library holds a model of a feedback compensator function in real factored form, called `fbcomp.m`, see Figure 2.15. Edited versions of `fbcomp.m` are shown in Figures 2.16 and 2.18. The user is advised to study the full help text of fbcomp (`help fbcomp`), where it is also stated how the file should be edited to represent a digital controller. An example of a digital controller is given in Chapter 6.

A model of a prefilter function in real factored form, `prefil.m`, is also present in the Qsyn library. An edited version of it, for Example 2.1, is shown in Figure 2.20.

We show here, in Figure 3.10, a controller function in polynomial form. The structure of a controller function file head is emphasized in a remark.

The commands connected to controller functions and control design are listed in Figure 3.11.

```
function [F] = control(s)

% control.m Controller function example in polynomial form for
% feedback compensator and prefilter design
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%

  b1 = 0; b2 = 0; b3 = 0; b4 = 0; b5 = 2; b6 = 2;
  a1 = 0; a2 = 0; a3 = 0; a4 = 0; a5 = 1; a6 = 0;


  Fnum = b1*s.^5 + b2*s.^4 + b3*s.^3 + b4*s.^2 + b5*s + b6;
  Fden = a0*s.^6 + a1*s.^5 + a2*s.^4 + a3*s.^3 + a4*s.^2 + a5*s + a6;

  F    = Fnum./Fden;
```

Figure 3.10. A controller function example realizing the transfer function $F(s)=2(s+1)/s$.

Remarks

1. A controller function file head must always be

```
function [F] = filename(s)
```

   where `filename` is a user chosen file name. `s=jw` is the imaginary frequency vector for which the frequency function `F` is computed. Since `s` is a vector, elementwise operations must be used, i.e. a point (.) precedes an operator acting on a vector.
2. A controller function may be used to represent a plant transfer function without uncertainty. The command cdesign and fdesign can be used to design controllers for such a plant representation. However, the user must himself write the code to close the loop, that will be similar to the some of the Matlab commands in the third paragraph in Section 2.8. An example of a design for a certain system is found in Chapter 6.

Primary commands for Controller files and Control Design	
CDESIGN	enables interactive feedback compensator design
FDESIGN	enables interactive prefilter design
PLANT	model of plant description file to be copied and edited by the user
SHOWBND	plots bounds from bound file for selected frequencies, in a Nichols chart
SHOWSPC	displays a specification from a specification file

Auxiliary commands for Controller files and Control Design	
FVALUE	get template from tpl-file/freq fcn value from controller file (subroutine)
HNGRID	draws L/(1+L) or 1/(1+L) loci in a Nichols diagram
HZOOM	puts a zoom menu in the current figure
MGRID	draws user defined grid lines in the current figure window

Figure 3.11. Commands connected with controller functions and controller design.

3.6 Commands for simulation

As mentioned in Section 2.8, Qsyn does not include commands for time domain simulations. The user should resort to other simulation tools or programs, such as those present in the Matlab Control System Toolbox, or Matlab Simulink, or other simulation programs such as Simnon or Visisim. Some auxiliary Qsyn functions do exist, however, and they are listed below in Figure 3.12.

Frequency domain simulations of selected closed loop cases are easily performed with Qsyn commands and regular Matlab algebra. An example is given in Section 2.8. Relevent commands are listed in Figure 3.12.

Primary commands for Simulation	
CASES	plant frequency domain simulation for user selected cases
CCASES	closed loop frequency function simulation for user selected cases
FBCOMP	feedback function model file, to be copied and edited by the user
PREFIL	prefilter function model file, to be copied and edited by the user
SHOWSPC	displays a specification from a specification file

Auxiliary commands for Simulation	
FVALUE	get template from tpl-file or freq fcn value from controller file (subroutine)
HNGRID	draws L/(1+L) or 1/(1+L) loci in a Nichols diagram
HZOOM	puts a zoom menu in the current figure
MGRID	draws user defined grid lines in the current figure window
PZ2S	computes freq function values/SIMULINK block for factored transfer fcn
SPECIF	SIMULINK block diagram for user's simulations

Figure 3.11. Commands connected with simulation.

3.7 Commands for cascaded design

Cascaded design for a two section SISO system is treated by a detailed example in Section x.x. To facilitate cascaded design, Qsyn includes a number of special commands in addition to those listed above: see Figure 3.12.

Commands for cascaded design	
FCASC_R	criterion fcn for rsrs for the 2nd step of cascaded design
FCASC_S	criterion fcn for odsrs or sensitivity of the 2nd step of cascaded design
TPLCASC	creates equivalent template for 2nd step in cascaded design

Figure 3.12. Commands connected with cascaded design

3.8 Utility commands

Qsyn includes a number of commands useful for matrix manipulation, conversion between Nichols and complex representation, etc. Notice that transfer functions without uncertainty can be displayed in Nichols and Bode charts with the commands `fdesign` and `cdesign`. The utility commands are listed in Figure 3.13.

Qsyn utility commands	
ADD2MAT	combines two matrices
C2N	converts a matrix from complex to Nichols form
CDESIGN	enables interactive feedback compensator design
CMPVEC	compares two vectors of different sizes, gives indices of equal elements
DEFILE	identifies file type (controller, plant, tpl-file, bnd-file)
E_TABLE	error table
FDESIGN	enables interactive prefilter design
GETFROM	lists and/or copies variables from a mat-file
HNGRID	draws L/(1+L) or 1/(1+L) loci in a Nichols diagram
HZOOM	puts a zoom menu in the current figure
INSERT	inserts/replaces variable in mat-file
ISIN	checks if a variable exists in a mat-file
LOAD	loads all variables into the work space from a mat-file (Matlab command)
LOOK	lists the names of the variables stored in a mat-file
MGRID	draws user defined grid lines in the current figure window
N2C	converts a matrix from Nichols form to complex form
PARGRID	creates vectors with all combinations of the elements of the input vectors
PGRID	grids a parameter vector elementwise and produces all combinations
PUTP	put a point before '^' , '/' , '*' in a string
QUNWRAP	unwraps templates in Nichols form over several Riemann surfaces
QGRID	makes a grid out of two vectors
READP	reads one variable from a mat-file (subroutine)
REMOVE	removes a variable from a mat-file
RMDBLP	removes doublets from a sorted vector
RNDSPACE	generates a row vector with randomly spaced elements
STDFON	change the fonts used in standard plots
WRAP	wraps angles of a matrix in Nichols form into desired Riemann surface
ZBOX	graphic utility for figure window

Figure 3.14. Qsyn utility commands

Appendix: Qsyn commands alphabetically

ADD2BND	adds/replaces bound in a bound file
ADD2MAT	combines two matrices
ADD2SPC	adds/replaces specification in a specification file
ADD2TPL	adds/replaces/combines a new template into a template file
ADEDGE	calculates templates with the recursive edge grid method
ADGRID	calculates templates with the recursive grid method
BNDINF	displays information about a bound file
BNDUPD	interactive recalculation of bounds with higher accuracy
BUILDF	extracts transfer fcn factors from the rff string format in a plant file
C2N	converts a matrix from complex to Nichols form
CASES	plant frequency domain simulation for user selected cases
CCASES	closed loop frequency function simulation for user selected cases
CBND	calculates bounds from given template file, specification file and criterion
CDESIGN	enables interactive feedback compensator design
CLTMP	interpolates points in a sorted and pruned template (for rff)
CMPVEC	compares two vectors of different sizes, returns indices of equal elements
CTPL	calculates a template file from data in a plant definition file
DEFILE	identifies file type (controller, plant, tpl-file, bnd-file)
E_TABLE	error table
EXTRACT	extracts bounds from a contour given by the Matlab command CONTOURC
FBCOMP	feedback function model file, to be copied and edited by the user
FCASC_R	criterion fcn for rsrs for the 2nd step of cascaded design
FCASC_S	criterion fcn for odsrs or sensitivity of the 2nd step of cascaded design
FDESIGN	enables interactive prefilter design
FIDSRS	criterion fcn for plant input dist step response specification
FIOSRS	criterion fcn for input step response specification/ complementary sensitivity
FODSRSC	criterion function for output dist step response specification at the plant input
FODSRS	criterion function for output dist step response specification or sensitivity
FRSRS	criterion function for reference step response specification or tolerance
FUSER	criterion function file to be copied and edit for user defined criterion fcn
FVALUE	get template from tpl-file or freq fcn value from controller file (subroutine)
GETBND	copies a bound from a bound file
GETFROM	lists and/or copies variables from a mat-file
GETTPL	copy a a template or other data variable from a template file
HNGRID	draws L/(1+L) or 1/(1+L) loci in a Nichols diagram
HZOOM	puts a zoom menu in the current figure
IDSRS	computes input disturbance step response specifications ??
INSERT	inserts/replaces variable in mat-file
ISIN	checks if a variable exists in a mat-file
LOAD	loads the variables into the work space from a mat-file (Matlab command)
LOOK	lists the names of the variables stored in a mat-file
MAKEBND	calculates bounds on a grid in the Nichols chart, subroutine to BNDUPD
MAKESPC	graphical definition of a specification variable
MAT2TPL	converts a matrix to a template file
MFFD	creates template file from a frequency vector, nominal, and template matrix
MGRID	draws user defined grid lines in the current figure window
MSPC	auxiliary function to MAKESPC
MULT	unstructured uncertainty function for all template comp methods except rff
N2C	converts a matrix from Nichols form to complex form
ODSRS	computation of output disturbance step response specification
PARGRID	creates vectors with all combinations of the elements of the input vectors
PGRID	grids a parameter vector elementwise and produces all combinations
PLANT	model of plant description file to be copied and edited by the user
PLANT_ID	compiles a plant description file and creates an auxiliary m-file
PLNT	invokes a default editor to edit a plant description file
PMODEL	interactive plant modelling function for measured freq function data
PNOMINAL	computes the freq function values of the nominal in a plant description file

PREFIL	prefilter function model file, to be copied and edited by the user
PRUNE	removes interior points from a connected template
PRUNE1	test version of PRUNE that displays each step of the iteration
PUTP	put a point before '^' , '/' , '*' in a string
PZ2S	computes freq function values/SIMULINK block for factored transfer fcn
QGRID	makes a grid out of two vectors
QUNWRAP	unwraps templates in Nichols form over several Riemann surfaces
READP	reads one variable from a mat-file (subroutine)
RECEDGE	subroutine used by the template computation functions ADEDGE
RECGRID	subroutine for the template computation function ADGRID
REMOVE	removes a variable from a mat-file
RFFCPZ	computes a complex pole/zero pair template in real factored form
RFFEL	pure gain, delay, unstructured uncertainty, or integrators rff template
RFFMUL	multiples two templates in real factored form
RFFPZ	produces a real pole or real zero template in real factored form
RFFUTIL1	utility function for RFFCPZ to compute template edge points
RFFUTIL3	utility function for RFFCPZ to sort, clean, and complement edges
RMDBLP	removes doublets from a sorted vector
RNDSPACE	generates a row vector with randomly spaced elements
RSRS	calculation of reference step response specification
SHOWBND	plots bounds from a bound file for selected frequencies, in a Nichols chart
SHOWSPC	displays a specification from a specification file
SHOWTPL	displays templates from a template file in a Nichols diagram
SPC_ID2	Subroutine used by IDSRS
SPC_OD2	specification calculation for 2:nd order output disturbance step
SPC_OD3	specification calculation for 3:rd order output disturbance step
SPC_OD31	specification calc for 3:rd order output disturbance step (alternative grid)
SPC_RS2	specification calculation for 2:nd order reference step
SPC_RS3	specification calculation for 3:rd order reference step
SPC_RS31	specification calculation for 3:rd order reference step (alternative grid)
SPCUPD	graphical updating of a frequency domain specification in a specification file
SPECIF	SIMULINK block diagram for user's simulations
STDFON	change the fonts used in standard plots
TPL2MAT	converts a template file from tpl-format to a matrix
TPLCASC	creates equivalent template for 2nd step in cascaded design
TPLFOP	template file operation routines, such as +, *
TPLINF	displays information about templates in a template file
TPLOP	single template operation routines, such as +, *
TPLPRUNE	prunes templates in a template file
TPLREDUC	reduces/interpolates points in a sorted and ordered template
TPLUNION	computes the union of two template files
TPLUPD	interactive, graphical template updating
UDBND	recalculates a previosly calculated bound, subroutine to BNDUPD
UPDBND	refining of a previously calculated bound, subroutine to BNDUPD
UPDBND1	subroutine, script m-file, only for use by UPDBND
UPDSPC	subroutine for interactive specification updating in SPCUPD
UPDSPC1	subroutine, script m-file, only for use by UPDSPC
UPDTPL	subroutine for interactive template updating in TPLUPD
UPDTPL1	subroutine, script m-file, only for use by UPDTPL
WRAP	wraps angles of a matrix in Nichols form into desired Riemann surface
ZBOX	graphic utility for figure window

Appendix 2: Commands for MIMO design

Design for a 2-input-2-output MIMO system, with basically non-interacting reference response specifications is treated in a detailed example in Chapter 8. To facilitate MIMO design, Qsyn includes a number of special commands in addition to those listed above: see Figure 3.13.

Commands for MIMO design	
BD11	computation of bd11 bound for 2x2 MIMO design
BDB	subroutine to BDBOUND
BDBOUND	computation of bd bounds for 2x2 MIMO design
BP11	computes bp11 bound for 2x2 MIMO design
CR_W11	computes w11 for 2x2 MIMO design
CR_W22E	computes w22e for 2x2 MIMO design
FT12	subroutine to MIMOBND1
FT12	subroutine to MIMOBND2
FT22	subroutine to MIMOBND2
MIMOBND1	computes bounds for the first step in a 2x2 MIMO design
MIMOBND2	computes bounds for the second step in a 2x2 MIMO design
MIMOINF	computation of 2x2 MIMO plant inverse and equivalent plant
SHOWBD	display bd11 bounds for the first step prefilter design (MIMO 2x2)

Figure 3.13. Commands connected with MIMO design