5. MORE ABOUT SPECIFICATIONS AND BOUNDS

In Section 2.3 two examples of how specifications are introduced into Qsyn were given: a reference step response specification using the command `rsrs` that approximately translates a desired step response envelope to a frequency domain envelope from which the tolerance specification emanates, and a constant 6 dB sensitivity specification that was entered manually inte the `odsrs_w` variable of the specification file, so that the `fodsrs.m` criterion file which computes the sensitivity function would automatically be used during the bound computation.

In Section 3.3 the structure and variables of the specification files were explained, and the commands that operate on specification files were listed. We introduced the criterion function m-file which is the function that operates with the appropriate transfer function on a specification for the benefit of the bound computation algorthm. In particular, we showed that the user may define her own criterion function.

Bound computation was illustrated in Section 2.4, and the structure, variables and commands belonging to bound files were explained in Section 3.4.

In this chapter more examples of the definition and computation of specifications and bounds will be given. In particular it will demonstrate how bounds are refined by the elegant interactive command `bndupd`.

But first we will first show one more example of how a standard specification is created (output disturbance step response specification, `odsrs`). We will compare this specification with the sensitivity specification in Section 2.3.2, and show how these two specifications are combined into one dominant odsrs-specification.

The user may insert her own specifications, and write her own criteria functions. Here examples will be given how error coefficients and stiffness coefficients enter naturally into QFT in the Qsyn framework.

An arbitrary specification in the frequency domain can be used with an arbitrary criterion function in the `cbnd` command. This holds for those three specifications (`rsrs, odsrs, idsrs`) for which Qsyn includes commands that aid the user to translate time domain specifications to the frequency domain, and for user defined criteria functions.

Most often a disturbance is not a step, but can often be characterized as a filtered step. We will show how a specification is modified when one knows the gain part of the frequency function of the filter. Sometimes the Power Spectral Density (PSD) of the disturbance is known. A signal with a given PSD can however always be created by letting an impulse through a filter whose magnitude Bode plot is the same as the PSD. And since an impulse is filtered step, we are back in the first case.

When both the disturbance and the desired outputs have limits given as PSD it is trivial to construct the specification by simply dividing the two PSDs. One example shows how that is done. A particular case is when a disturbance is concentrated to a narrow frequency range, and the desired attenuation is specified.

In many cases one may achieve the same modification by either changing the specification variable or by changing the equation in a criterion function. It should be emphasized very strongly that the criterion function is evalued for each template point for each nominal open loop candidate that the bound computation algorithm tests for compliance with the specification. The number of criterion function evaluations is thus very large, typically hundreds of thousands of times to get a set of bounds. Therefore it is *essential to keep the code of the criterion function as short as possible*. So, when possible, modify the specification variable!

5.1 Combination of specifications to get a dominant specification

Consider Example 2.1 in Chapter 2. Assume that, in addition to the 6 dB sensitivity specification in the specification file `ex2 1a.spc` of Section 2.3.2, a time domain output disturbance step response specification is required as defined in the lower part of Figure 5.1 as displayed by the command

```
odsrs('ex5 1','odsrs',[0.3 0.1 50],50,1.5,[],logspace(-1,2),2);
```

5.1.1 Combination by matrix manipulation

Clearly, the resulting frequency domain specification (upper part of Figure 5.1) supercedes 6 dB for some frequencies, see Figure 5.2. In order to minimize the bound computation effort, it would be reasonable to combine the odsrs specifications of `ex2 1a.spc` and `ex5_1.spc`. Proceed as follows:

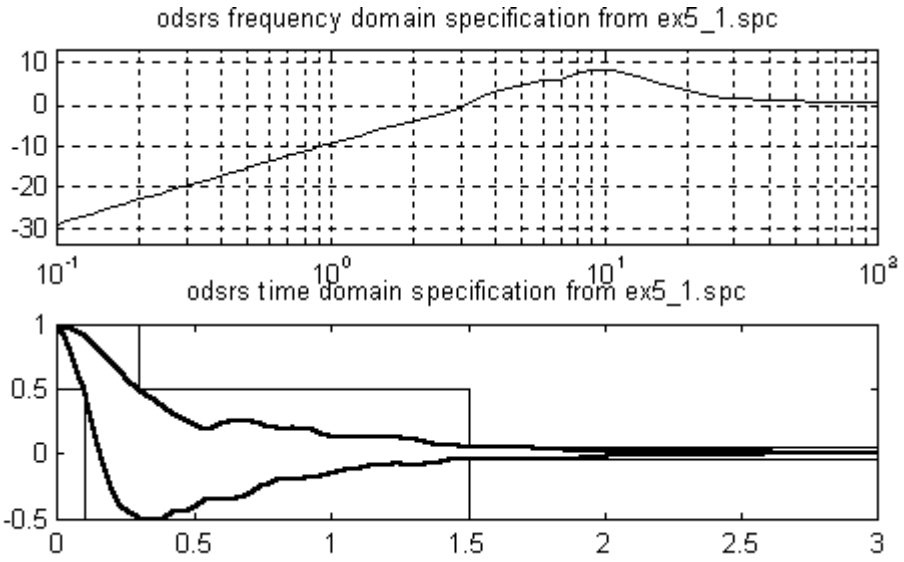

Figure 5.1. Time and frequency domain specifications resulting from the command
`odsrs('ex5 1','odsrs',[0.3 0.1 50],50,1.5,[],logspace(-1,2),2);`

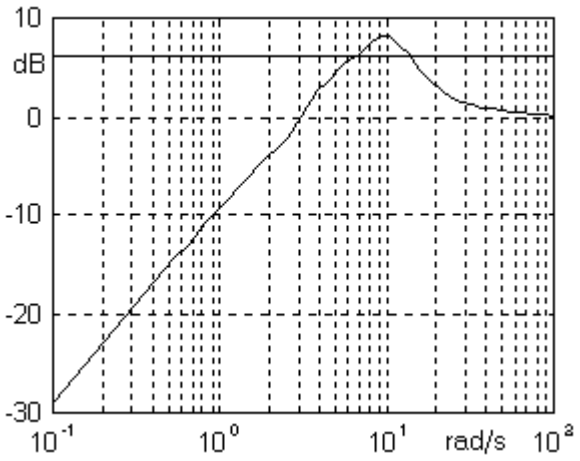

Figure 5.2. The `odsrs_w` *specifications from the files* `ex2_1a.spc` *and* `ex5_1.spc`*, shown with the commands*

```
showspc('ex2 1a','odsrs','freq'); hold on, ...
showspc('ex5 1','odsrs','freq','g',gcf); hold on, ...
axis([0.1 100 -30 10])


odsrs2=getfrom('ex2 1a.spc','odsrs w');
odsrs5=getfrom('ex5 1.spc','odsrs w');
max(abs(odsrs2(:,1)-odsrs5(:,1)))   % check that frequencies are
  ans =                             %  the same
       0
odsrs_new=[odsrs2(:,1) min(odsrs2(:,2),odsrs5(:,2))]; % new spec
!copy ex2_1a.spc ex5_1a.spc         % new spec file
insert('ex5_1a.spc',odsrs_new,'odsrs_w','r');  % add2spc could also
                                               %  have been used
showspc('ex5 1a','odsrs','freq')    % Figure 5.3
```

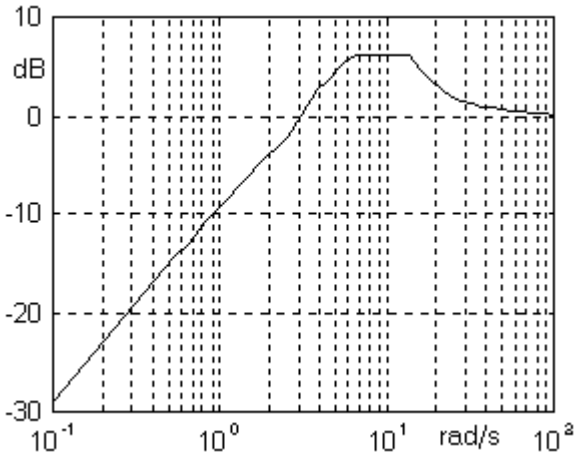

Figure 5.3. Composite odsrs specification found as the minimum of the specifications in Figure 5.2.

The new odsrs specification is found as the minimum of the two specifications, after confirming that the specification frequencies are the same. If the frequencies are not identical, the minimum can be chosen for the frequencies of one of the specifications, using interpolated values of the other specification with the help of the Matlab interpolation function `table1`.

5.1.2 Combination using the command `spcupd`

Alternatively one may proceed as follows, using the graphical updating possibility of the command `spcupd`, in particular if the frequency vectors are not equal. Update the specification vector by manually assigning the minimum of the 6 dB and the odsrs specification in `ex5 1.spc`.

```
!copy ex5_1.spc ex5_1b.spc                % make safety copy
spcupd('ex5_1b','odsrs',[0.1 100 -30 10]); % follow instructions in
      %  the graphic window and confirm update when you are satisfied
odsrs new=getfrom('ex5 1b.spc','odsrs w');
!copy ex2 1a.spc ex5 1b.spc               % copy basic spec file
add2spc('ex5 1b','odsrs',odsrs new(:,1),odsrs new(:,2));
                          % insert could also have been used
showspc('ex5 1b','odsrs','freq');
```

You will possibly notice a small difference between the odsrs frequency specification in `ex5 1a.spc` and `ex5 1b.spc`, depending on how steady your hand was when handling the mouse during the `spcupd` updating. Notice that `spcupd` and `add2spc` take `'odsrs'` as the specification name argument, without the extension `'_w'` , since these commands by default only work with frequency specifications. The specification name variable within the specification file `ex5 1b.spc` is however `odsrs w` .

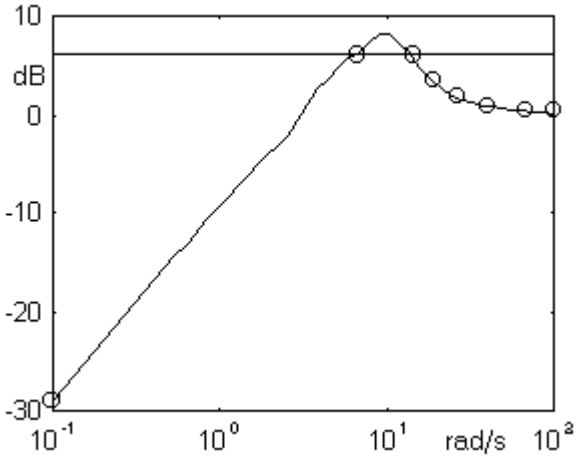

Figure 5.4. The `odsrs_w` *specifications from the files* `ex2_1a.spc` *and* `ex5_1.spc`*, shown with the commands*

```
showspc('ex2 1a','odsrs','freq');
showspc('ex5 1','odsrs','freq','g',gcf); hold on,
axis([0.1 100 -30 10])
```

followed by the command `odsrs_new=mspc(1);` *where* o *denote the locations of the mouse clicks to define the new specification* `odsrs_new`.

5.1.3 Combination using the command `mspc`

A third possibility to change the odsrs specification is the use of the graphical command `mspc`, which works with mouse clicks in an existing figure window:

```
showspc('ex2 1a','odsrs','freq'); ...
hold on, showspc('ex5 1','odsrs','freq','g',gcf); hold on
axis([0.1 100 -30 10])
odsrs new=mspc(1);                        % Figure 5.4
!copy ex2_1a.spc ex5_1c.spc               % copy basic spec file
add2spc('ex5_1c','odsrs',odsrs_new(:,1),odsrs_new(:,2));
showspc('ex5_1c','odsrs','freq');
```

Notice that in this case `odsrs new` will have the same number of rows as there are mouse clicks in Figure 5.4, something that does not matter since the bound computation routine interpolates the specification value with respect to the logarithmic frequency scale.

5.2 Manufacture of other specifications

In Section 2.3.2 the command `add2spc` was used to create a constant sensitivity specification. In the above Section 5.1, the use of the commands `add2spc`, `spcupd`, and `mspc` was demonstrated. In this section we will give examples how other non-predefined specifications are manufactured.

5.2.1 Graphical definition of a specification using `makespc`

Example: iosrs specification

An arbitirary specification with one or two specification vectors can be manufactured graphically with the command `makespc`, e.g.

```
makespc('ex5 2a','iosrs',2,[0.01 100 -30 10]);  %Fig. 5.5
      INSERT: New datafile ex5 2a.spc created
```

where an input-output step response specification for a one degree-of-freedom system (referring to the closed loop transfer function *PG*/(1+*PG*), see the criterion function `fiosrs.m`) was specified by two mouse clicks for each frequency. The sequency of clicks is concluded by pressing Return. It is important to click in the same order for each frequency, i.e. first the upper and then the lower specification, since otherwise the two specification vectors will be mixed up. Check your result with

```
showspc('ex5_2a','iosrs');
```

If you wish to be aided by an existing amplitude Bode plot when specifying the new specificaiton, use the command `mspc`, see Section 5.1.3 and Figure 5.4.

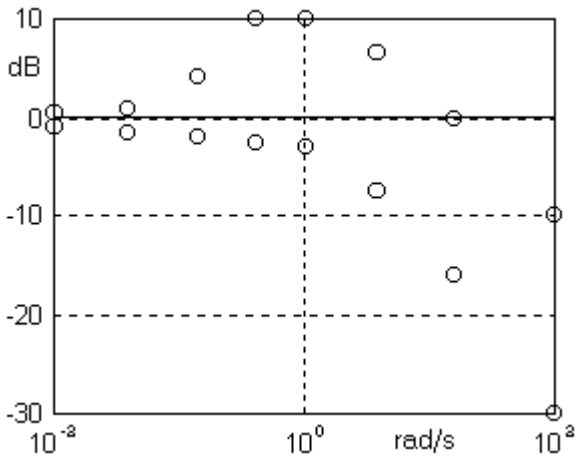

Figure 5.5. Creating a specification with two vectors, with the command `makespc('ex5_2a','iosrs',2,[0.01 100 -30 10]);`.

Example: error coefficient specification

Assume that we wish to ensure that the error coefficients for the closed loop system satisfy

$$e_0 = 0, \quad e_1 \le 0.1 \tag{5.1}$$

which imply that the low frequency asymptotes for all open loop cases L_i must lie above a Bode gain line whose slope is -20 dB/dec and whose intersection with the 0 dB axis occurs at 10 rad/s. Such a specification can be included in the file `ex5 2a.spc` with another use of `makespc`:

```
makespc('ex5 2a','errcoeff',1,[0.1 100 -60 40]);
showspc('ex5 2a','errcoeff');                    % Fig. 5.6
```

Notice in Figure 5.6 that the low frequency asymptote is defined for low frequencies only with a low, non active specification for higher frequencies, as not to impose an irrelevant specification for the cross-over and high frequencies. I

One may define the specification for the limited frequency range [0.1, 1] rad/s only, but then the user *must* assign the argument `w` in the `cbnd` command to be equal to the template frequencies in that range.

The error coefficient specification `errcoeff` will be used in Section 5.x with a user written criterion function to compute error coefficient Horowitz bounds.

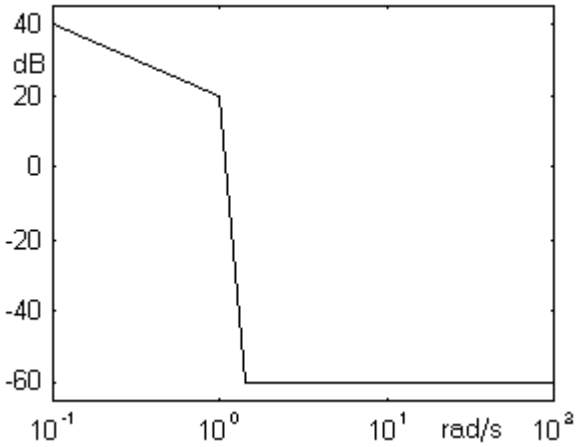

Figure 5.6. Error coefficient specification created with the command `makespc('ex5_2a','errcoeff',1,[0.1 100 -60 40]);` *and displayed with* `showspc('ex5 2a','errcoeff');`

Example: noise rejection specification

A third, almost identical example is the introduction of a noise rejection specification at the plant input, referring to the closed loop transfer function $-G/(1+PG)$, see the criterion function `fodsrsc.m`. Assume that it is known that measurement noise occurs at frequencies in the range [40, 60] rad/s, and that an attenuation of -40 dB at the plant input is required. One way to introduce such a specification is the command

```
makespc('ex5 2a','odsrsc',1,[40 60 -60 -20]); %click on -40 dB
showspc('ex5 2a','odsrsc'); axis ([1 100 -60 -20])
```

and another way is

```
add2spc('ex5 2a','odsrsc',[40 60],-40);
showspc('ex5 2a','odsrsc'); axis ([1 100 -60 -20]).
```

Notice that a limited frequency range was used, and that the argument `w` in the `cbnd` command must then be assigned.

5.2.2 Filtered specifications

Assume it is known that the output step disturbance (d_2 in Figure 1.1) enters the summing junction in a filtered way that can be modelled with a certain filter $H(s)$ as in Figure 5.7.

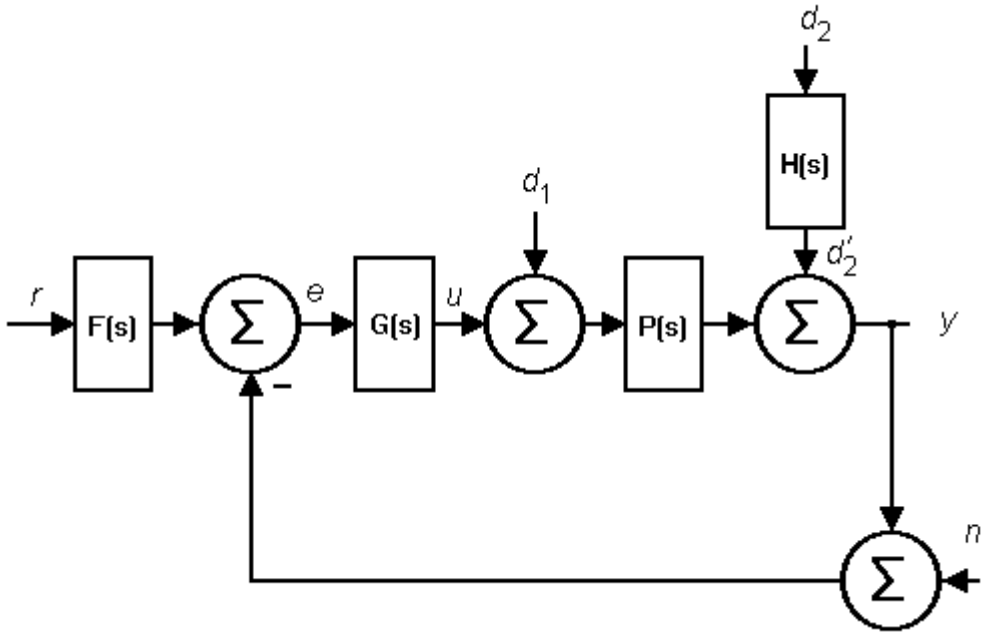

Figure 5.7. Two degree-of-freedom system with filtered output disturbance.

Since $H(s)$ takes a part of the filtering burden, the closed loop frequency domain specification for the transfer function from d_2' to y (whose generic Qsyn name is `odsrs_w`) may be modified by multiplication with $H^{-1}(s)$.

Example: filtered odsrs specificaton

As an example, consider the odsrs specification in `ex5 1.spc`, see Figure 5.1. Assume that $H(s) = 1/(1+s/25)$. Then the new odsrs specification can be constructed in the following way:

```
odsrs51=getfrom('ex5 1.spc','odsrs w');   % get original odsrs w
% compute gain (dB) of H(s) for frequencies in original odsrs w:
add2spc('ex5_2a','odsrs',odsrs51(:,1),[1],[1/25 1]);
showspc('ex5_2a','odsrs');    % check that you got it right
odsrs52=getfrom('ex5_2a.spc','odsrs_w');  % extract |H(jw)| (dB)
% divide original odsrs w (dB) with  |H(jw)|, which is equivalent
%   to logarithmic subtraction:
odsrs52(:,2)=odsrs51(:,2)-odsrs52(:,2);
add2spc('ex5 2a','odsrs',odsrs52(:,1),odsrs52(:,2)); % insert
showspc('ex5 2a','odsrs');          % check, Fig. 5.8
```

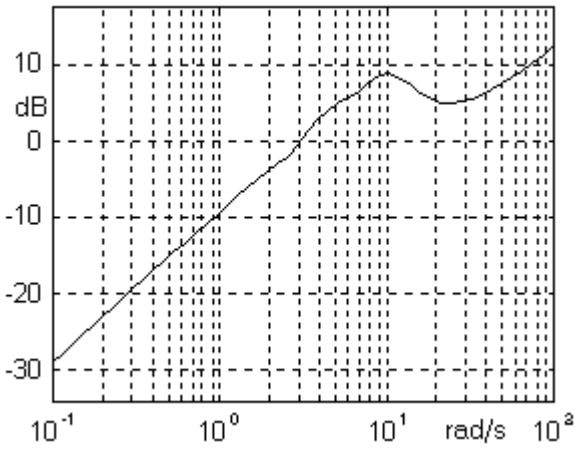

Figure 5.8. odsrs frequency domain specification from `ex5_2a.spc`.

Notice that the odsrs frequency domain specification in `ex5_2a.spc` should be combined with a reasonable sensitivity specification (e.g. 6 db) as in Section 5.1, for closed loop stability reasons.

Example: PSD based idsrs specification

Assume that a plant input disturbance, d_1 in Figure 5.7, has a known Power Spectral Density (PSD), modelled as $|1/(1+s/25)|$ (dB). It is required that the output *y* as a result of the input disturbance should have a PSD not exceeding $|0.1s/((s+10)(1+s/20))|$ (dB). This means that the closed loop system the closed loop transfer function *P*/(1+*PG*) (see the criterion function `fidsrs.m`) should act as a filter with an upper gain envelope defined by $(0.1s(1+s/25))/((s+10)(1+s/20))$. Such a specification is easily included, under e.g. the appropriate standard name `idsrs`,

```
add2spc('ex5_2a','idsrs',logspace(-1,2),[.1/25 0.1 0],[1/20 1.5 10]);
showspc('ex5_2a','idsrs'); grid            % Figure 5.9
```

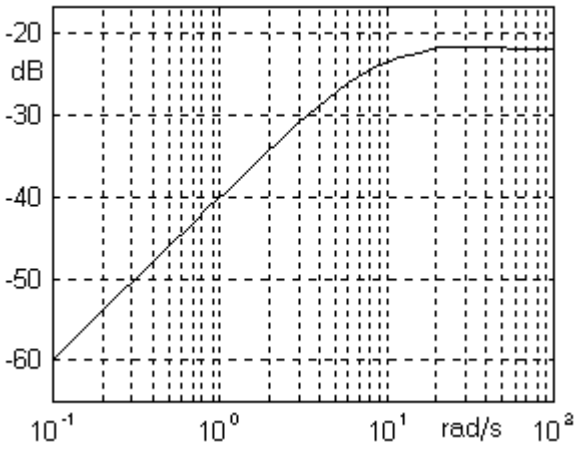

Figure 5.9. idsrs frequency domain specification from `ex5_2a.spc`.

Other specifications

It should be noted that all specification vectors presented in Chapter 2 and in Sections 5.1 and 5.2 have dB units. Nothing prevents the user to define specifications with respect to phase, or some combination of phase, gain, and frequency. A specification matrix may have arbitrary many columns, with the restriction that the first, leftmost column, holds frequency.

The only other requirement is that the user writes a criterion function adapted to her specification.

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

% FERRCOEF   Error coefficient criterion function
%            [Tmax]=ferrcoef(tpl_nom,tpl,Lnom,spec,par_nom,par)
%
% computes the maximum gain value of the open loop G*P = L
%
% 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 error coefficient specification scalar in [dB].
%            It equals the element  2 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)
%
% G = (Lnom - tpl_nom);    % Each column is constant, and each row
%                          % contains the n different feedback compensator
%                          % candidates, deg + j*dB
%
%
% L = Lnom - tpl nom + tpl       % Each column contains an open loop
%                                % candidate, deg + j*dB
%
  Tmax = min(imag(Lnom - tpl_nom + tpl)) - spec(1);
                          % Note tham min works columnwise
                          % spec(1) contains the error coeff criterion
                          % for low frequencies. The specification is
                          % satisfied for the k:th value of Lnom iff
                          % Tmax(k) >= 0.
```

Figure 5.10. Criterion function `ferrcoef.m` *for the error coefficient specification in Section 5.2.1.*

5.3 User defined criteria functions

In Section 3.3.2 the standard criterion function `frsrs.m` was shown in Figure 3.6a and a model for user defined criteria functions, `fuser.m`, in Figure 3.6b. The file `fuser.m` contains two examples of user defined criteria functions, one for sensitivity and one for stiffness coefficients.

Example: criterion function for error coefficients

Here, we will author a criterion function for the error coefficient specification in Section 5.2.1, given by equation (5.1) translated to the specification on the open loop, `errcoeff`, in the file `ex5 2a.spc`, as shown in Figure 5.6. The file `fuser.m` is copied into the file name `ferrcoef.m` in the working directory, and edited as shown in Figure 5.10. It is essential that the executable code is kept as compact as possible, since the criterion function is evaluated so many times. The criterion function is very simple

$$T_{\max} = \min\left(\left|L_i\right|\right) - \text{spec}, \text{ [dB]} \tag{5.2}$$

where $\{L_i\}$ is the compensated open loop candidate template for some candidate compensator value $G(j\omega)$ or equivalently some candidate nominal open loop value $L_{\text{nom}} = P_{\text{nom}}G(j\omega)$, and spec is given in Figure 5.6. The criterion is satisfied for

$$T_{\max} \geq 0 \tag{5.3}$$

for the relevant frequencies that define the low frerquency asymptote in Figure 5.6.

5.4 Updating a bound file

In Section 2.4 it was shown how Horowitz bounds are computed, and how one easily inserts bounds of different types into the same bound file. In this section two other ways of updating a bound file will be considered: adding a bound for a new frequency to a set of existing bounds of a given type; and re-computing bounds with a higher accuracy.

Consider the bound file `ex2 1a.bnd` in Section 2.4. The command

```
bndinf('ex2_1a')

   ODSRS bound exists for [rad/sec]:
-------------------------------------------
  Columns 1 through 7
    0.2000    0.5000    1.0000    2.0000    5.0000   10.0000   20.0000
  Column 8
   50.0000

   RSRS bound exists for [rad/sec]:
-------------------------------------------
  Columns 1 through 7
    0.2000    0.5000    1.0000    2.0000    5.0000   10.0000   20.0000
  Column 8
   50.0000
```

reveals for what frequencies the bounds exist. The column number equals the second column in `odsrs w` and `rsrs w`, respectively and indicates the index in the bound variable name, as explained in Section 2.4.2. Also recall from Section 2.4.2 that the bound variables `rsrs 5`, `rsrs 6`, `rsrs 7`, and `rsrs 8` contain a `NaN` each.

5.4.1 Adding a bound for a new frequency to a bound file

However we found, in Section 2.5 and 2.8 that we might want to compute bounds also for 7 rad/s. As a preparation we computed, in Section 4.6, the template for 7 rad/s for the plant `ex2_1a.m` was computed.

The rsrs and odsrs bounds for 7 rad/s with respect to the specification file `ex2 1a.spc` are now easily computed:

```
cbnd('ex2_1a','rsrs',7);
      Calculating bounds for rsrs specification, with the
      templatefile ex2 1a.tpl
      w---> 7 [rad/s]
      No bound found, use larger search area, or higher accuracy

cbnd('ex2 1a','odsrs',7);
      Calculating bounds for odsrs specification, with the
      templatefile ex2_1a.tpl
      w---> 7 [rad/s]
```

Not surprisingly (see Section 2.4.1) there is no rsrs bound for this frequency. Its value is found by the following sequence of commands,

```
bndinf('ex2_1a')

   ODSRS bound exists for [rad/sec]:
-------------------------------------------
  Columns 1 through 7
    0.2000   0.5000    1.0000    2.0000    5.0000   10.0000   20.0000
  Columns 8 through 9
   50.0000    7.0000

   RSRS bound exists for [rad/sec]:
-------------------------------------------
  Columns 1 through 7
    0.2000   0.5000    1.0000    2.0000    5.0000   10.0000   20.0000
  Columns 8 through 9
   50.0000    7.0000
```

which reveals that the new bounds for 7 rad/s are called `rsrs 9` and `odsrs 9`, respectively, and

```
getbnd('ex2_1a','rsrs', 7)
  ans =
        NaN.
```

The new odsrs bound is, as usual, displayed by

```
showbnd('ex2_1a',[],7,'odsrs');     % Figure 5.11
```

Figure 5.11. The odsrs bound for 7 rad/s, `odsrs_9`*, from* `ex2_1a.bnd`.

5.4.2 Refining a bound with the command `bndupd`

In Figure 5.11 we notice the annoying dummy bound around the instability point -1, and that the bound has a gap, due to the accuracy with which it was originally computed. Using the command `bndupd`, we may refine this bound, by deleting unwanted portions (right mouse button), and recalculating appropriate bound segments with a higher accuracy (left mouse button):

```
bndupd('ex2 1a','odsrs',7,[],[],[],[],[],[.5 1]);
      Recalculating bounds for odsrs specification, with the
      templatefile ex2_1a.tpl
      w---> 7 [rad/s]

      Continue (ENTER) or quit (q):
```

Letting the frequency argument `w` of the command `bndupd` equal `[]`, all bounds of the odsrs type in `ex2_1a.bnd` may be updated. The resulting odsrs bound for 7 rad/s is given in Figure 5.12.

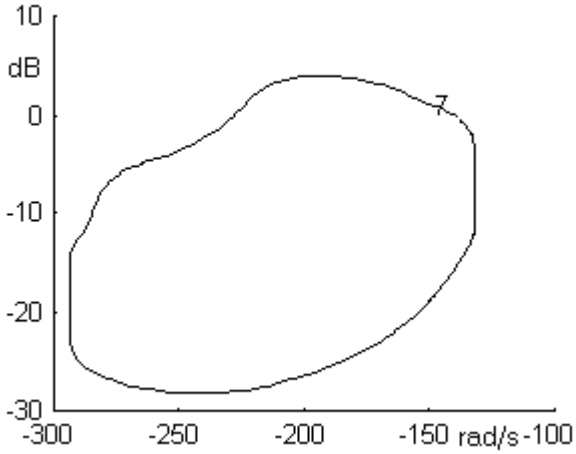

Figure 5.12. The odsrs bound for 7 rad/s, `odsrs_9`*, from* `ex2_1a.bnd`*, as displayed by the command* `showbnd('ex2_1a',[],7,'odsrs')`.

5.5 Composite bounds

In Section 2.4.3 it was shown how different bound types were dominant for different frequencies, and how the dominant bounds were displayed together. Sometimes, however, bounds of different types interect for a given frequency, and a composite bound would consist of segments from the original bounds.

Consider the odsrs specification in `ex5_1a.spc`, see Figure 5.3. Based on this specification and on the template file `ex2 1a.tpl`, compute the odsrs bound for 2 rad/s:

```
cbnd('ex5 1a','odsrs',2,[],'ex5 1a','ex2 1a')
      Calculating bounds for odsrs specification, with the
      templatefile ex2 1a.tpl
      w---> 2 [rad/s]
      GETFROM: The file ex5_1a.bnd does not exist
      INSERT: New datafile ex5 1a.bnd created
```

Display this bound in Figure 5.13, together with the rsrs-bound for 2 rad/s from `ex2 1a.bnd` (compare Figure 2.14).

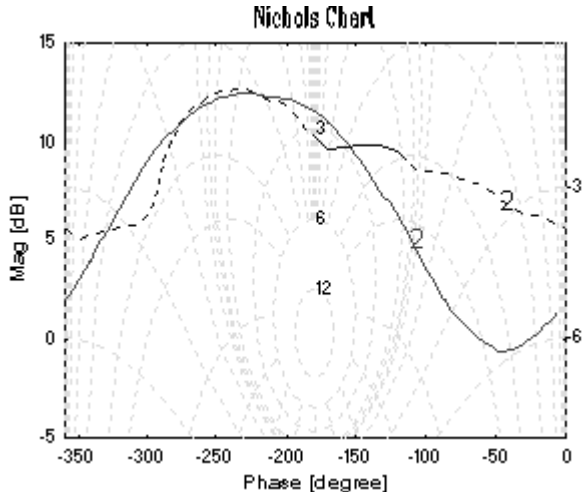

Figure 5.13. The odsrs bound for 2 rad/s from `ex5_1a.bnd` *(solid grey) and the rsrs bound for 2 rad/s from* `ex5_1a.bnd` *(dashed black), displayed with the command sequence*
`hngrid,showbnd('ex5 1a',gcf,2,'odsrs','m');`
`showbnd('ex2 1a',gcf,2,'rsrs','b:'); axis([-360 0 -5 15]);`

Clearly the dominant composite bound consists of the bound segments that have maximum gain for a given phase in Figure 5.13. The dominant composite bound cannot easily be constructed by vector manipulation, since the two original bound row vectors have different lengths, have been computed for different phases, and consist of different contiguous segments with an NaN element inbetween. The structure of the bound vectors may be studied after having extracted them from their bound files:

```
rsrs2=getbnd('ex2_1a','rsrs',2);
odsrs2=getbnd('ex5 1a','odsrs',2);
```

It is hence easier to define the composite bound graphically with the Matlab command `ginput` by mouse clicking in Figure 5.13, and e.g. inserting it into the new boundfile `ex5 5.bnd` under the name `bound`:

```
[deg, dB]=ginput;       % click on desired points from left to right,
                        % and finish with Enter
bound2 = deg+j*dB;
add2bnd('ex5 5',2,bound2.','bound'); % IMPORTANT: insert row vector
      GETFROM: The file ex5 5.bnd does not exist
      INSERT: New datafile ex5_5.bnd created
```

The new bound is displayed together with its constituents in Figure 5.14.

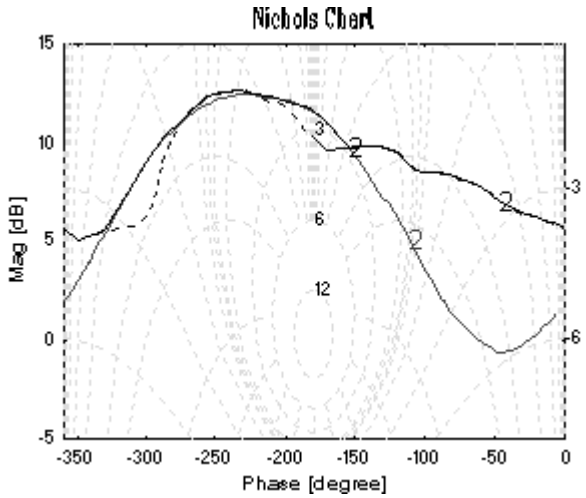

Figure 5.14. The odsrs bound for 2 rad/s from `ex5_1a.bnd` *(solid grey), the rsrs bound for 2 rad/s from* `ex5_1a.bnd` *(dashed black), and the composite bound for 2 rad/s from* `ex5_5.bnd` *(solid black) displayed with the command sequence*

```
hngrid,showbnd('ex5 1a',gcf,2,'odsrs','m');
showbnd('ex2 1a',gcf,2,'rsrs','b:');
showbnd('ex5 5',gcf,2,'bound','r'); axis([-360 0 -5 15]);
```

It demands some work to create composite bounds in the above way. Moreover, if one is forced to violate a bound during the feedback design, it is important to know from which specification it emanates. Therefore the need to use composite bounds should be carefully assesed, and in any case should the original bounds be kept.

5.6 Bounds from a user defined specification and criterion

Consider the error coefficient specification in Section 5.2.1: equation (5.1), the specification `errcoeff` in `ex5 2a.spc`, and Figure 5.6. For this specification, we wrote a special criterion function, `ferrcoef.m`, in Section 5.3. Let us now compute the bounds emanating from `errcoeff` and `ferrcoef.m` for the plant templates in `ex2_1a.tpl`. Since the specification is active only for frequencies ≤ 1 rad/s, we restrict the bound computation to the template frequencies 0.2, 0.5, and 1 rad/s.

```
cbnd('ex5_6','errcoeff',[0.2 0.5 1], ...
   [],'ex5_2a','ex2_1a',[],'ferrcoef');
      Calculating bounds for errcoeff specification, with the
      templatefile ex2_1a.tpl
      w---> 0.2 [rad/s]
      GETFROM: The file ex5 6.bnd does not exist
      INSERT: New datafile ex5 6.bnd created
      w---> 0.5 [rad/s]
      w---> 1 [rad/s]
```

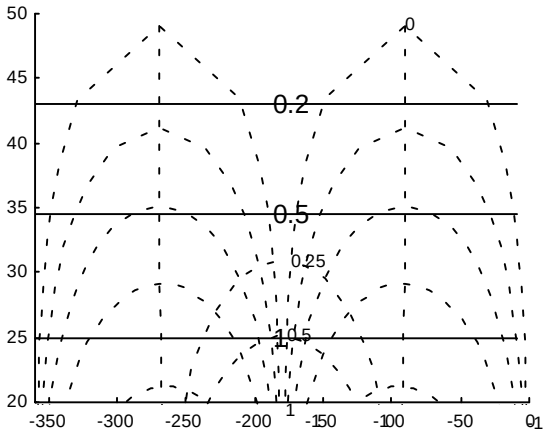

Figure 5.15. Error coefficient bounds from `ex5_6.bnd` *displayed in a Nichols chart with the command sequence*

```
showbnd('ex5 6',[],[],'errcoeff'); hold on; axis([-360 0 20 50]);
hngrid
```

The `cbnd` command demonstrates how one combines different template, specification, and criteria function files and names. Not surprisingly does the result in Figure 5.15 show that Horowitz bounds emanating from an error coefficient specification are not phase dependent.