4. MORE ABOUT PLANTS AND TEMPLATES

In Section 2.1 we explained the structure of plant description files using the standard file `plant.m` in the Qsyn library. We used it to define the plant (2.1) of Example 2.1. In Section 2.2 we computed the desired templates of (2.1) using the real factored form method, the recursive edge grid method, and the grid method.

We compared the resulting templates, and noticed the considerable differences in computing time: 0.5 minutes for rff, 28 minutes for the recursive edge grid method, and 6 minutes for the grid method. A large part of the difference is accounted for by the fact that in the grid methods, the parameter vector belonging to each saved template point is stored as a column in the template file variables `par 1`, `par 2`, etc, while no parameter vectors are stored in the rff-method. Another secret behind the speed of the rff-method is the fact that it by construction uses so called Tree Structured Decomposition (Ackermann, 1993), i.e. computes the elementary template for each transfer function factor and then multiplies the factors together (Gutman, Neumann, Baril, 1994). In this chapter we will examplify how tree structured decompositon can be used with the grid methods, by computing the sub-templates for plant parts that are parametrically independent from other parts, and then using the commands `tplfop` and `tplprune` or `prune` to compose the total templates. However the user must decide if she does not prefer to take a (long) coffee break while the computer sweats over the calculation of the total templates, instead of manually defining partial plants, issuing several `ctpl` commands, composing the total templates, and then inspecting them before pruning.

This chapter includes examples of the recursive grid computation method and the random grid method. We also demonstrate a case when the Edge theorem does not hold whence it follows that the recursive edge grid method gives erroneous templates. Examples with unstructured uncertainty, uncertain delay, an uncertain number of integrators, plants with dependent parameters in different transfer function parts, and plants that are unstable for some parameter combinations, and stable for others are included.

An example how measured frequency function data is transformed to templates is given, on the basis of which interactive plant modelling gives rise to an uncertain plant description in form of a plant description file. The model, and the measured templates are joined into a composite template.

It is also demonstrated how an additional template is inserted into a template file, and how a dense rff-computation followed by a thinning out of template points forms a fast and exact way to compute templates.

4.1 Unstructured uncertainty, delay, and a number of integrators. Tree Structured Decomposition

Consider the plant (2.1). Assume we would like to include unstructured uncertainty, an uncertain delay, and an uncertain number of integrators: 0 or 1. Then we get the e.g. the plant description

$$\begin{cases} P(s)=\dfrac{ke^{-\tau s}}{s^r}\cdot\dfrac{s+a}{1+2\varsigma s/\omega_n+s^2/\omega_n^2}\left(1+M(s)\right), \quad \tau\in[0,0.1],\ r\in\{0,1\}, \\ k\in[2,5],\ a\in[1,3],\ \varsigma\in[0.1,0.6],\ \omega_n\in[4,8],\ |M(s)|\le m(s)<1 \end{cases} \tag{4.1}$$

where $m(s)$ is given for $s=j\omega$ in Figure 4.1. Since all factors in (4.1) are in real factored form, a straightforward way to compute the templates is to edit `ex2_1a.m` in Figure 2.2 to

include the new factors. The new file, `ex4 1a.m`, is presented in Figure 4.2. The command `ctpl('ex4 1a')` computes, during 1.6 minutes, the templates with the real factored form method. When studying the templates with `showtpl` (do that!), one notices that some templates, e.g. the one for 0.2 rad/s consists of two disjoint sets each due to the two integrator cases, while in other templates the two distinct sets overlap and form simply connected set with interior template points.

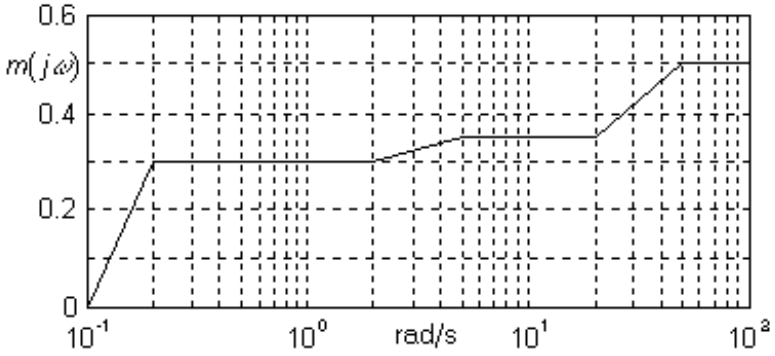

Figure 4.1. Magnitude of the unstructured uncertainty m(s) in (4.1).

```
function   [Par,w_tpl,w_nom,method,P_num,P_den, ...
            n_dif,Uns_Par] = ex4_1a

%  Plant_name :  Example 4.1a. Rff method

% Definition of the parameters
 Par = [                %'p1=[p1min,p1max,p1nom,number of cases]'
         'k=[2, 5, 2, 8]' , ...                  % uncertain gain
         'a=[1, 3, 3, 8]', ...                   % zero: s+a
         'zet=[0.3, 0.6, 0.6, 8]', ...           % complex pole
         'wn=[4, 8, 4, 8]',...
         'tau=[0, 0.1, 0, 3]',...                % delay
         ];

% Multiplicative unstructured uncertainty:
Uns Par=[0.1 0.2 0.5  1   2   5    10   20   50  100;
         0   0.3 0.3  0.3 0.3 0.35 0.35 0.35 0.5 0.5];

% Definition of the frequency vectors [rad/sec]
w_tpl = [0.2 0.5 1 2 5 10 20 50];   %template frequencies [rad/s]
w_nom = logspace(-1,2);             %nominal frequencies  [rad/s]

% Definition of the template computation method
method = 'rff [1,1]';

% Plant definition: Real Factored Form Structure
P num='(gain,k)(delay,tau)(hf,a)';
P_den='(dc,wn,zet)';

% number of differentiators/integrators
n_dif = [-1 0 0]; %one or zero integrators, nominal = zero
```

Figure 4.2. Plant description file `ex4_1a.m` *for the full description of (4.1).*

An alternative way to compute the templates for (4.1) is to exploit the fact that the templates of (2.1) are already computed and stored in `ex2 1a.tpl`. That would be an example of Tree Structured Decompositon. What is needed is a plant description file, `ex4 1b.m` in Figure 4.3, for

```
function   [Par,w tpl,w nom,method,P num,P den, ...
            n dif,Uns Par] = ex4 1b

%  Plant name :  Example 4.1b. Rff method

% Definition of the parameters
 Par = [                %'p1=[p1min,p1max,p1nom,number of cases]'
          'tau=[0, 0.1, 0, 3]',...               % delay
       ];


% Multiplicative unstructured uncertainty:
Uns Par=[0.1 0.2 0.5  1   2   5    10   20   50  100;
         0   0.3 0.3  0.3 0.3 0.35 0.35 0.35 0.5 0.5];

% Definition of the frequency vectors [rad/sec]
w_tpl = [0.2 0.5 1 2 5 10 20 50];   %template frequencies [rad/s]
w_nom = logspace(-1,2);             %nominal frequencies  [rad/s]

% Definition of the template computation method
method = 'rff [1,1]';

% Plant definition: Real Factored Form Structure
P_num='(delay,tau)';
P_den='[1]';

% number of differentiators/integrators
n_dif = [-1 0 0]; %one or zero integrators, nominal = zero
```

Figure 4.2. Plant description file `ex4_1b.m` *for (4.2).*

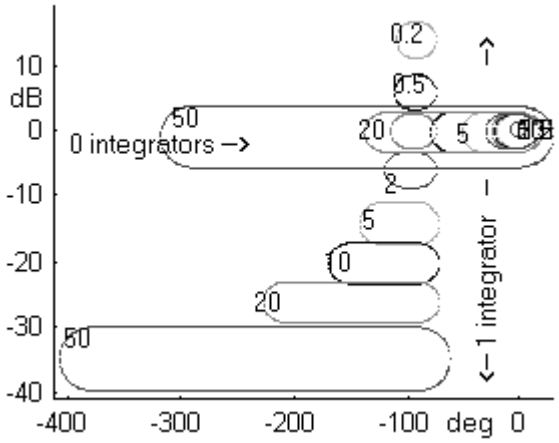

Figure 4.3. Templates generated by `ctpl('ex4_1b'); showtpl('ex4_1b');` *The templated frequencies are printed on the templates where unambiguous. The nominal points lie all in* 1 *(0 dB, 0 deg). The template for each frequency* ω_k *is disjoint: one ellipse-like set for 0 integrators, and an identical set "moved" by* $1/(j\omega_k)$*. The 0 integrator template parts all include the nominal point, and are pointed to by the 0 integrators arrow. The 1 integrator template parts are moved -90 degrees and the appropriate amount of dB, and are indicated by the 1 integrator mark.*

$$Q(s)=\frac{e^{-\tau s}}{s^{r}}\left(1+M(s)\right)\,. \qquad (4.2)$$

The template file for (4.2) is produced by `ctpl('ex4_1b')` (0.6 min) and the result is displayed in Figure 4.3 by the command `showtpl('ex4 1b')`. The reader may investigate the templates of the individual factors of (4.2) taking note that the plant description of the real factored form demands at least one uncertain parameter in the transfer function, which may be an infinitesimally uncertain gain, see `plant.m`.

Alternatively, the template file for (4.2) may be computed by the Recursive Edge Grid method. When this method is applicable, which is the case here, it is advantageous over the rff method in two respects: (*i*) it computes the true template vertex points also when a low accuracy is used (rff approximates to the nearest phase in a phase grid given by the accuracy vector `dist`), and (*ii*) it stores the parameters that were used to compute the final template point, except for unstructured uncertainty and number of integrators. The second point is however not particularly important for uncertain delay. So, the command `ctpl('ex4 1br','ex4 1b','aedgrid [5,5]');` `(3 min)` is used to get the template file `ex4 1br.tpl`. The command `showtpl('ex4 1br')` reveals the same templates as in Figure 4.3, but uglier, and with about one third of the number of points in comparison to `ex4_1b.tpl`.

We now choose to multiply the templates of `ex2 1a.tpl`, (2.1), and `ex4 1b.tpl`, (4.2) to get a set of templates for (4.1) to be compared with those of `ex4_1a.tpl`. (The use of `ex4 1br.tpl` is left as an exercise for the reader, see below.) The command to use is `tplfop`. This command includes, as an option (`prune op` in the input argument list) to have the resulting template pruned and/or reduced, actions that can be done seperately with the commands `tplprune` and `tplreduc`, respectively. One has to be careful, though, and only profit from these options when one is sure that the resulting template is suitable. In our case, with an uncertain number of integrators, we cannot be sure which, if any, of the templates will be simply connected and hence satisfying one condition for prunability. We therefore choose to multiply the template files with the default option of neither pruning nor reducing. Afterwards we may use, also for templates that are not simply connected, the interactive command `tplupd` (template update) which enables us to inspect the templates, and selectively prune them.

Note that our example as usual also serves as a warning example. Template multiplication means that all points in the first template are multiplied by all points in the second template. That means that (*i*) it would have been wiser to save the inclusion of the one-integrator case to the very end, which also would have made it possible to use the pruning option of `tplfop` (which is much faster than first using `tplfop` without pruning, and then `tplprune`); and (*ii*) it is wise to choose as low an accuracy of the multiplicands as allowed. If the user chooses to redo the example, she is advised to recompute `ex4 1b.tpl` with lower accuracy than `[1,1]` used above (or to use `ex4_1br.tpl)` and to wait with the uncertain integrator.

```
tplfop('ex4 1c','*',[],'ex2 1a','ex4 1b');  % 6 Mbytes, ca 15 min!!!
showtpl('ex4 1c');      % another 10 minutes since the templates
                        % are almost solidly filled: don't do it!
!copy ex4_1c.tpl ex4_1d.tpl   % prepare a safety copy
showtpl('ex4_1d',0.2);        % Figure 4.4a
tplupd('ex4 1d',[2 2]);       % Interactive pruning. Takes one hour
                              % for the whole template file.
showtpl('ex4 1d',.2)          % Fig 4.4b shows the pruned template
                              % for 0.2 rad/s.
showtpl('ex4 1a',.2,'ro',gcf);% compare with Fig 4.4b, find out that
                              % the templates are equal. Do this!
```

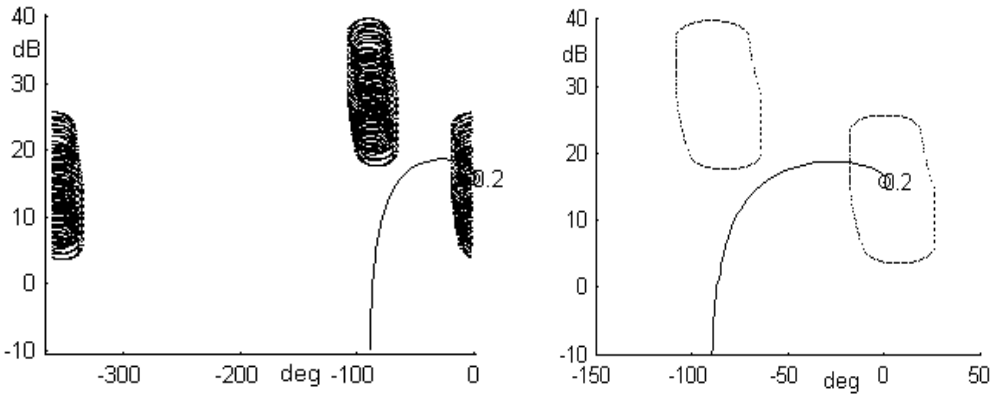

Figure 4.4. (a) the left figure shows the template for 0.2 rad/s in the template file `ex4_1c.tpl`, *which was created by multipling the template for (2.1) and for (4.2). (b) the right figure shows the template to the left after pruning (fromin* `ex4_1d.tpl`*), see the code above. The user should check that the corresponding template in* `ex4_1d.tpl` *is identical. Also compare with Figure 2.6 where no unstructured uncertainty, no delay and no integrators are present.*

4.1.1 Conclusion

Like many other examples in this manual, this example also demonstrates some pitfalls:

1. If the rff method can be used directly, do it. Note however that the rff method does not save the parameters of the stored cases, cf. Section 2.8.
2. When multplying templates, chose as low a resolution of the factors as possible, otherwise the *curse of dimensionality* will haunt you.
3. Prune in an appropriate way, using `prune` to prune individual templates, `tplprune` to prune all templates in a template file, and `tplupd` for a graphical, interactive interface to prune, and "clean" , and complement a template.
4. Tree Structured Decomposition (TSD) means that independent parts of a transfer function have their templates computed seperately, with the total template composed from the parts. It is essential that the individual templates have as low a resolution as possible or as few points as possible, either in the original computation, or by pruning them afterwards. In this example TSD was not necessary, in others it is mandatory. TSD is often worthwhile when template computation methods other than rff are used. In some cases TSD is not recommendable e.g. when the plant has a feedback structure with lowly damped resonances.
5. Factors including an uncertain gain, uncertain delay, or unstructured uncertainty are often a part of a realistic plant description. Their templates (one or together) are efficiently and accurately computed with the recursive edge grid method (`aedgrid`) or the rff method, for use in Tree Structured Decomposition. The rff method is much faster for the same resolution, but the recursive edge grid method gives correct template vertices even with a low resoluiton. A compromise is to compute the templates with high resolution rff, and then prune which is rather a fast operation.
6. Uncertain integraors should be saved as the last `tplfop` operation. It is advisable to keep the templates simply connected as long as possible and uncertain integrators usually destroy this property. Multiplication and division with `tplfop` can always be followed by automatic pruning as part of the `tplfop` command when the templates are simply connected. This combined operation is usually much faster than pruning seperately afterwards with `tplprune`.

4.2 The Edge Theorem

Unfortunately there are no necessary and sufficient conditions for when it is sufficient to compute a template from the edges in the parameter space, only. see Ackermann (1993). Sufficient conditions exist for certain types of transfer functions, e.g. those having so called Kharitonov polynomials for numerator and denominator. See Ackermann (1993).

Here we will warn the user to uncritically use the recursive edge grid method by providing a simple example of its failure. Consider the uncertain transfer function

$$P(s) = \frac{1}{(s+a)(s+b)}, \quad a \in [1,10],\ b \in [1,10] \tag{4.3}$$

described by the plant description file in Figure 4.5. Since (4.3) is in real factored form we use that description although primarily other template computation methods will be used. Notice that we have to denote `method = 'rff'` in oder to give the compiler the correct information about what transfer function structure is used, even though other template computation methods will be desired in the `ctpl` command. The template for 3 rad/s using each of the available Qsyn method is computed:

```
ctpl('ex4 2rg','ex4 2a','adgrid [1 1]');
                              % recursive grid (1013 points):27 min!
ctpl('ex4 2e','ex4 2a','aedgrid [1 1]');
                              % rec edge grid (333 points):  2.8 min.
ctpl('ex4 2g','ex4 2a','grid');% grid (64 points):           0.17 min
ctpl('ex4_2ra','ex4_2a','rgrid');
                              % random grid (64 points):    0.17 min
ctpl('ex4_2rff','ex4_2a');    % rff (228 points):          0.14 min
```

```
function   [Par,w tpl,w nom,method,P num,P den, ...
            n dif,Uns Par] = ex4 2a

%  Plant_name :  Example 4.2.

% Definition of the parameters
 Par = [                %'p1=[p1min,p1max,p1nom,number of cases]'
         'k=[1]', ...                          % gain=1
         'a=[1, 10, 10, 8]' , ...              % pole: 1/(s+a)
         'b=[1, 10, 10, 8]', ...               % pole: 1/(s+b)
        ];

% Multiplicative unstructured uncertainty:
Uns_Par=[];

% Definition of the frequency vectors [rad/sec]
w tpl = [3];      %template frequencies [rad/s]
w nom = logspace(0,1);              %nominal frequencies  [rad/s]

% Definition of the template computation method
method = 'rff_[1,1]';

% Plant definition: Real Factored Form Structure
P num='[k]';
P den='(hf,a)(hf,b)';

% number of differentiators/integrators
n_dif = [ 0 0];   %
```

Figure 4.5. Plant description file `ex4_2a.m` *for f (4.3).*

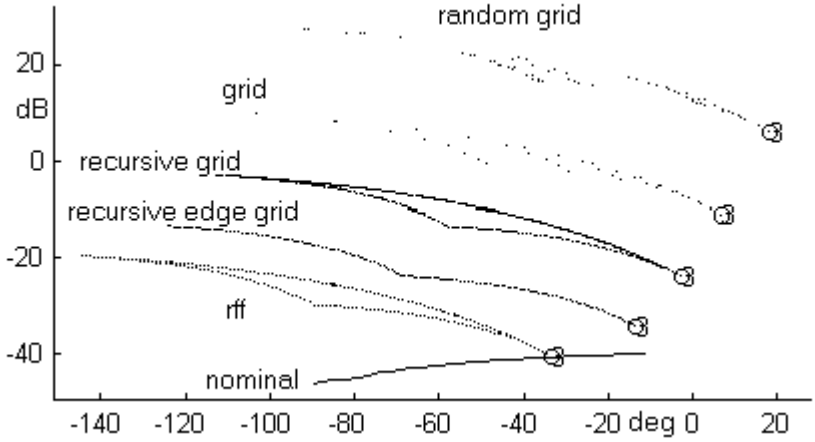

Figure 4.6. The template at 3 rad/s for (4.3) computed with the five template computation methods in Qsyn. Notice that the template computed by the recursive edge grid method is wrong in principle since the Edge Theorem does not hold for (4.3).

Using `showtpl` with the point option, all templates are displayed in the same diagram, see Figure 4.6. Clearly the recursive grid method gives an erroneous template. The reason is that the upper template border is given by *a=b* in (4.3), i.e. the diagonal in parameter space, which is not covered by the any edge search.

4.2.1 Conclusions

If real factored form cannot be used, compute templates as follows:

1. Consider Tree Structured Decomposition.
2. For each frequency, compare the templates computed by the recursive edge grid, grid, and random grid methods. If it seems that the recursive edge grid method correctly computes the template border, use that method. Otherwise, compute the union of the templates computed by the recursive edge grid method, and finely gridded grid and random methods. Prune.
3. When the transfer function form and parameter ranges are finally known, recompute the templates using the recursive grid method. Use a low accuracy except for critical frequencies.

4.3 Unstable and non-minimum phase plants

Consider the transfer function

$$P(s)=\frac{\left(1+2\varsigma_n\omega_n+\omega_n^2\right)}{\left(1+2\varsigma_d\omega_d+\omega_d^2\right)},\quad \varsigma_n\in[-0.1,0.1],\ \omega_n\in[1,10]\ \varsigma_d\in[-0.1,0.1],\ \omega_d\in[1,10] \quad (4.4)$$

This is a very ugly plant having two poles that may be either stable or unstable, and two zeros that may be minimum or non-minimum phase. Moreover, cancellation occurs for some parameter combinations. A plant description file for (4.4) is found in Figure 4.7. The result of `ctpl('ex4_3'); showtpl('ex4_3')` is seen in Figure 4.8. As expected the template covers the whole complex plane, which is approximated here by the gain band between `eps` and `1/eps`. The user is invited to change the parameter ranges, and study the effect.

```
function   [Par,w tpl,w nom,method,P num,P den, ...
            n dif,Uns Par] = ex4 3

%  Plant name :  Example 4.3.

% Definition of the parameters
 Par = [                %'p1=[p1min,p1max,p1nom,number of cases]'
         '
         'zn=[-0.1, 0.1, 0.1, 8]' , ...          % complex zero
         'wn=[1, 10, 10, 8]', ...                %
         'zd=[-0.1, 0.1, 0.1, 8]' , ...          % complex zero
         'wd=[1, 10, 10, 8]', ...                %
         ];

% Multiplicative unstructured uncertainty:
Uns_Par=[];

% Definition of the frequency vectors [rad/sec]
w tpl = [3];      %template frequencies [rad/s]
w nom = logspace(0,1);              %nominal frequencies  [rad/s]

% Definition of the template computation method
method = 'rff [2,5]';

% Plant definition: Real Factored Form Structure
P_num='(dc,wd,zd)';
P_den='(dc,wn,zn)';

% number of differentiators/integrators
n_dif = [ 0 0];   %
```

Figure 4.7. Plant description file `ex4_3.m` *for (4.4).*

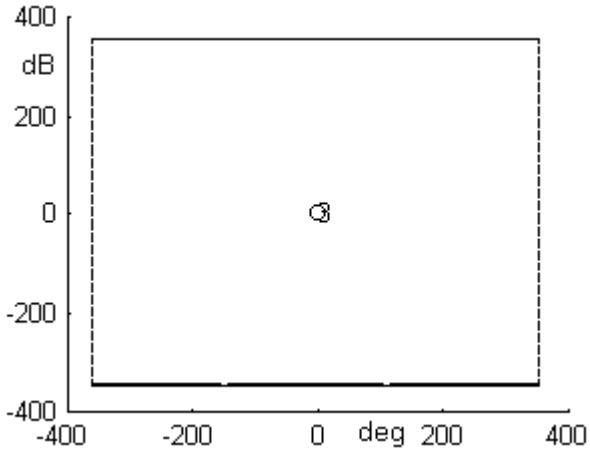

Figure 4.8. The template for 3 rad/s for (4.4).

4.4 Dependent parameters

Consider the following model of a two-mass system connected with a flexible shaft (Cohen, Nordin and Gutman, 1995), where the input is motor torque and the output is motor speed:

$$P(s) = \frac{J_l s^2 + ds + k}{J_l J_m s^2 + (J_l + J_m) d s^2 + (J_l + J_m) k s} \tag{4.5}$$

where $J_l \in [5.6, 8]$ kgm^2 is the load moment of intertia, $d \in [30, 300]$ Nm/(rad/s) is the shaft damping, $k \in [5880, 5900]$ Nm/rad is the shaft elasticity, and J_m = 0.4 kgm^2 is the motor moment of inertia. Clearly, the rff method cannot be used here since the polynomial coefficients are dependent. A plant description file for (4.5) is found in Figure 4.9. The command sequence, using the recursive edge grid and recursive grid methods,

```
ctpl('ex4 4e','ex4 4','adedge [5,2]',10*pi);    % 0.16 min
tplprune('ex4_4e',[],[10 2])              % to smooth the borders
showtpl('ex4_4e',[],[],'-');
ctpl('ex4_4ag','ex4_4','adgrid_[5,2]',10*pi)    % 0.7 min
tplprune('ex4 4ag',[],[10 2])
showtpl('ex4 4ag',[],[],':',gcf);
```

produces the template of 5 Hz as seen in Figure 4.10. We notice that also in this realistic example, the edge grid method is unreliable.

```
function   [Par,w_tpl,w_nom,method,P_num,P_den, ...
            n dif,Uns Par] = ex4 4


%  Example 4.4: Two mass system

% Definition of the parameters
% =====================
   Par = [
         'J_m=[0.4,0.4,0.4,1]' , ... % [minvalue,maxvalue, nomvalue]
         'J l=[5.6,8,5.6,4]', ...
         'd m=[0,0,0,1]', ...
         'd l=[1,1,1,1]',...
         'k s=[5880,5900,5900,2]',...
         'd s=[30,300,30,8]',...
         ];

% Definition of the frequency vectors [rad/sec]
% =====================================

% Template frequency vector.
 w tpl=2*pi* ...
 [0.5000;1.0000;1.3800;1.5600;2.0000;2.2700;2.4200;
  3.1000;3.1700;3.3700;3.4400;3.9000;4.0600;4.4100;
  4.8000;4.9000;5.0000;5.2100;5.4300;5.6600;5.7800;
  5.9000;6.2800;6.6800;7.1100;7.4200;7.8900;8.5700;
  8.9400;9.5100;9.9100;10.3000;10.8000;11.5000;12.2000;
  12.7000;13.3000;13.8000;14.4000;15.0000;16.0000;17.0000;
  17.4000;18.1000;19.3000;20.0000;30.0000;40.0000;50.0000];

w nom=2*pi*5*logspace(-1.5,1,300);  % Nominal frequency vector.

% Definition of the template computation method
% =============================================
  method='adgrid_[5,5]';

% Polynomial Structure
% ================
  P num ='2.5235*(J l*s^2+(d s+d l)*s+k s)';
  P den=['(J l*J m*s^2+(J l*(d m+d s)+J m*(d l+d s))*',...
            ' s+k s*(J l+J m))*(s+(d m+d l)/(J m+J l))'];

  % note the way strings are written over more than one line
  % in Matlab

% =============
  Uns_Par=[];
  n_dif=[];
```

Figure 4.9. Plant description file `ex4_4.m` *for (4.5).*

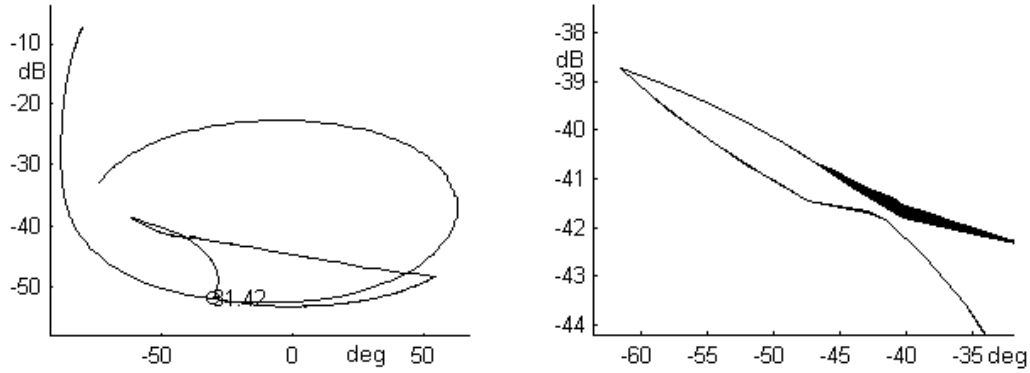

Figure 4.10. The nominal frequency function for (4.5) with its nominal as defined in Figure 4.9, together with the template for 5 Hz, computed with both the recursive edge grid method and the recursive grid method. The zoomed picture on the right shows in heavy black the part of the template that the edge grid method overllooked.

4.5 Templates from measured data and interactive plant modelling

Qsyn enables the user to measure frequency function data, import them into Matlab, and reformat them into a matrix with the frequencies as column number 1, the (arbitrary) nominal case as column number 2, and the other measured cases as the remaining columns. Note that every frequency value point must be stored as `deg + j*dB`.

Using the command `pmodel`, the user can then try to fit different transfer functions, in controller function m-file form, to the measured data, in a Bode or Nichols chart.

When one has a feeling for the possible transfer function structure and parameter ranges, the data matrix is translated to a template file with the help of the command `mffd` or `mat2tpl`. Then the user defines a new plant description file, computes its template file (`ctpl`) and compares it in the Nichols chart with the template file emanating from the measured data, using `showtpl`. The comparison can be made in the Bode plot with the help of the command `cases`, and a Bode display of the original data matrix. Clearly the plant description file parameters will have to be changed a few times until a satisfactory fit is achieved.

4.5.1 Example

Assume that we have measured 49-frequency frequency function for each of 46 load cases a two mass system similar to the one described in Section 4.4.. We have imported the data as a matrix formatted as mentioned above. The Bode and Nichols diagrams of the 46 cases, Figure 4.11, is displayed with the following command sequence, where `tpl_meas` is the measured data matrix, with the nominal case in column 2 equalling the case in column 3.

```
w_meas=tpl_meas(:,1);                % freq  vector in first column
tpl_meas=tpl_meas(:,3:48);           % measured cases w/o nominal
figure                               %
subplot(211);                        % Bode diagram in Figure 4.11
semilogx(w_meas,imag(tpl_meas(:,3:48)),'r');
subplot(212);
semilogx(w_meas,real(tpl_meas(:,3:48)),'r');
```

```
plot(tpl meas(:,3:48));                % Nichols diagram in Figure 4.11
```

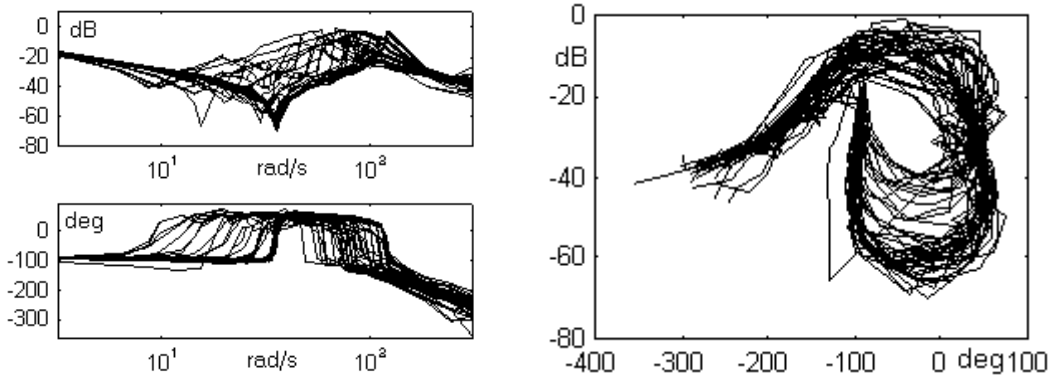

Figure 4.11. Bode and Nichols diagram of 46 measued frequency functions from different load cases a two-mass system.

Using our deep insight of linear transfer functions and their Bode and Nichols diagram we immediately believe that a suitable model is the one in Figure 4.12, an m-file called `model.m` realised in controller function form (see Section 3.5).

```
function [P]=model(s)

% model for example 4.5

  J m=0.4; J l=5.8; d m=0; d l=1; k s=5900; d s=2; delay=0.01;


P_num =2.5235*(J_l*s.^2+(d_s+d_l).*s+k_s).*exp(-s*delay);


P den=(J l*J m.*s.^2+(J l*(d m+d s)+J m*(d l+d s)).*s+k s*(J l+J m)).
*(s+(d m+d l)/(J m+J l));

P = P_num./P_den;
```

Figure 4.12. Suggested transfer function model

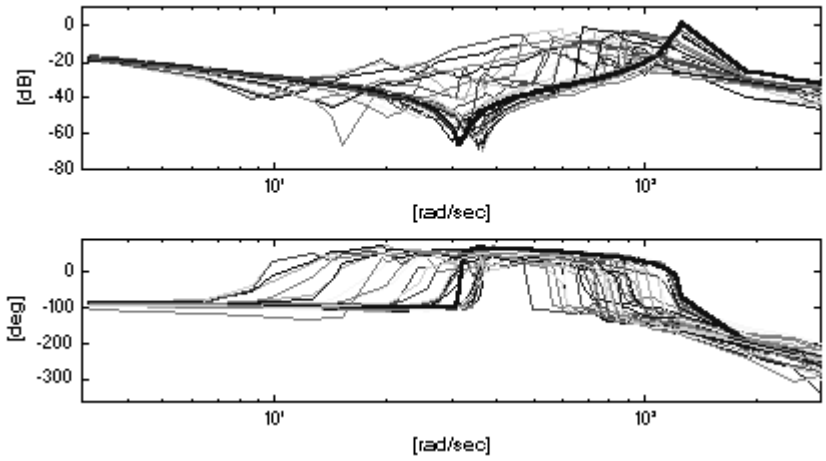

Figure 4.13. The result of `pmodel('model',tpl_meas,tpl_meas(:,1),1);`*, where the measured data in the matrix* `tpl_meas` *is displayed in Figure 4.11, and plant model* `model.m` *is defined in Figure 4.12. The frequency function of* `model.m` *is drawn in heavy black.*

The command

```
pmodel('model',tpl meas,tpl meas(:,1),1);
```

produces Figure 4.13, and we happily note that our mental identifier worked quite well, since the model frequency function lies right inside the set of measured frequency functions! After playing around with the parameters in `model.m` to get a feeling for the ranges we feel confident enough to propose an uncertain model, having the same structure as `ex4_4.m` in Figure 4.9, but with the parameters given in Figure 4.14. The new plant description file is called `ex4_5.m`.

```
Par = [
       'J_m=[0.4,0.4,0.4,1]' , ... % [minvalue,maxvalue, nomvalue]
       'J_l=[5.8,5.8,5.8,1]', ...
       'd_m=[0,0,0,1]', ...
       'd_l=[1,1,1,1]',...
       'k_s=[600,7500,5900,8]',...
       'd_s=[2,20,2,8]',...
       'delay=[0.007,0.01,0.01,2]',...
       ];
```

Figure 4.15. Parameter vector in the plant description file `ex4_5.m`*. The remaining part of* `ex4_5.m` *is identical to* `ex4_4.m` in Figure 4.9.

A template file `ex4_5.tpl` is created from the matrix `tpl_meas` with the command

```
mat2tpl('ex4_5',tpl_meas,'data_960214');
```

whereby we also included a comment variable in the template file. The template file for `ex4_5.m` should be computed the same way as the templates for `ex4_4.m` in Section 4.4. Here we compute the modelled template for frequency number 35 and display it together with the measured template for the same frequency in Figure 4.16 :

```
w35 = tpl_meas(35,1);
ctpl('ex4_5ag','ex4_5','adgrid [5,2]',w35)
                                % pruning included in adgrid
showtpl('ex4_5ag'); showtpl('ex4_5',w35,[],'o',gcf); % Figure 4.16
```

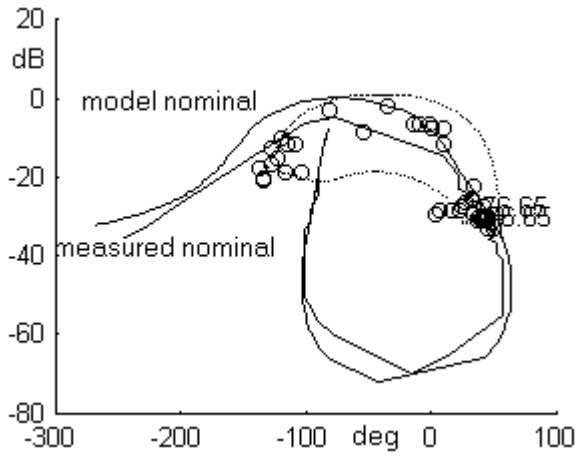

Figure 4.16. Nominals for plant model `ex4_5.m` *and in the measured template file* `ex4_5.tpl`. *The border of the template for 76.65 rad/s in* `ex4_5ag.tpl` *emanating from* `ex4_5.m` *is shown dotted. The measured template for the same frequency is shown as* o.

In Figure 4.16 and corresponding figures for other frequencies we notice that the correspondence between the measured templates and the model templates is good but not perfect. In particular it seems as if some model parameter combinations are not encountered in the measured plant.

The Bode plots of the modelled transfer functions (for all cases in `ex4_5.m`) are produced as follows, and shown in Figure 4.17. Compare with Figure 4.11 and 4.13 that contain the measured cases and our first modelling attempt!

```
cases('ex4_5','all',[],1);
```

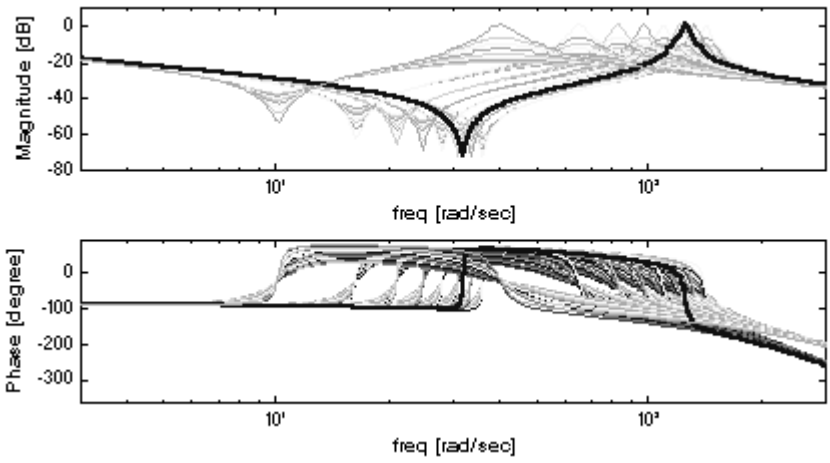

Figure 4.17. Bode plots for the cases defined in the plant definition file `ex4_5.m`.

4.6 Adding a template into an existing template file

Qsyn offers the opportunity to add an extra template into a template file. Consider the example in Chapter 2 with its plant definition file `ex2_1a.m` and its template file `ex2_1a.tpl`. We noticed that we need an extra template for 7 rad/s. Let us first see what templates are included in `ex2_1a.tpl`. The command

```
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
```

To compute the template for 7 rad/s (with the default template computation method and accuracy) the following command is issued:

```
ctpl('ex2_1a',[],[],7);
```

which as usual gives calculation information on screen

```
Calculating templates using the Real Factored Form method

--> for w=7 [rad/sec]

 Computing time  :  [min]  =  0.1793
```

Notice that we did not have to set the `ctpl` argument `option`, since there was no previous template at 7 rad/s in the template file `ex2 1a.tpl`. If there had existed such a template, we could either choose to have it replaced (`option='r'`), which is the default, or keeping the union of the old and new template (`option='u'`). The new contents of the template file is revealed by

```
tplinf('ex2 1a')
```

giving the answer

```
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 |
|   7.000   |    t_w 9       |   287x1 |
|  10.000   |    t_w 6       |   181x1 |
|  20.000   |    t_w 7       |    99x1 |
|  50.000   |    t_w 8       |    60x1 |
+-----------+----------------+---------+
Number of Differentiators : 0
```

Note that if one has a template as a vector in the workspace, e.g. after having extracted it from some template file with the command `gettpl`, then one may add that template to some other template file with the command `add2tpl`, which also has the replace/union option. The command `add2tpl` should be used rather than `insert`. The user should make sure that `w_tpl`, the template frequency vector, correctly reflects the templates in the template file, by e.g. using the command `tplinf`. Notice that inserting a new template by `insert` does not update `w tpl`, which then has to be done with `gettpl`, vector editing, and inserting the new `w tpl`. The command `add2tpl` updates `w tpl` and `w nom`, however.

4.7 Creating the union of two template files

Qsyn offers the opportunity to form the union of two template files in the sense that templates for frequencies occur in in only one of the parent template files are copied unchanged into the daughter file, while the resulting template for a common frequency will become the union of the two parent templates. Consider the example in Section 4.5. One template file, `ex4_5.tpl`, holds the experimental data. Another template file, `ex4 5ag.tpl`, holds one template (for 76.6549 rad/s) generated from the plant description file `ex4 5.m`. For design purposes it is clever to let the template be the union of measured and modelled template points:

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

If we regard e.g. the template for 76.6549 rad/s (`showtpl('ex4_7',76.6549)`), see Figure 4.16, we notice that some of the measured points are outside the border of the model template. We may leave it that way; an alternative, however, is to manually include the convex hull of the union of the measured modelled template points, i.e. exactly the templates in `ex4_7.tpl`, and then prune the convex hull. This is achieved using the command `tplupd`. Since `tplupd` updates the file it is working on, it is often wise to make a safety copy:

```
!copy ex4_7.tpl ex4_7saf.tpl
                        % Unix users note: this is the DOS command.
```

Then we issue the command

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

which has three mouse clicking options: single left click and holding the button down adds new template points along the the line the user draws, double left click and holding the button down prunes the template within the rectangular area the user marks, and single right click and holding the button down deletes points within the rectangular area the user marks.

We start by adding points along lines connecting the measured template points and the model template border. Then, just after adding all new desired points, the figure window looks like Figure 4.18a, (compare with Figure 4.16). Then we ask for pruning, and after that the template looks like in Figure 4.18b.

The user is invited to chech the template sizes in the different template files with the command `tplinf`, just like in the preceding section.

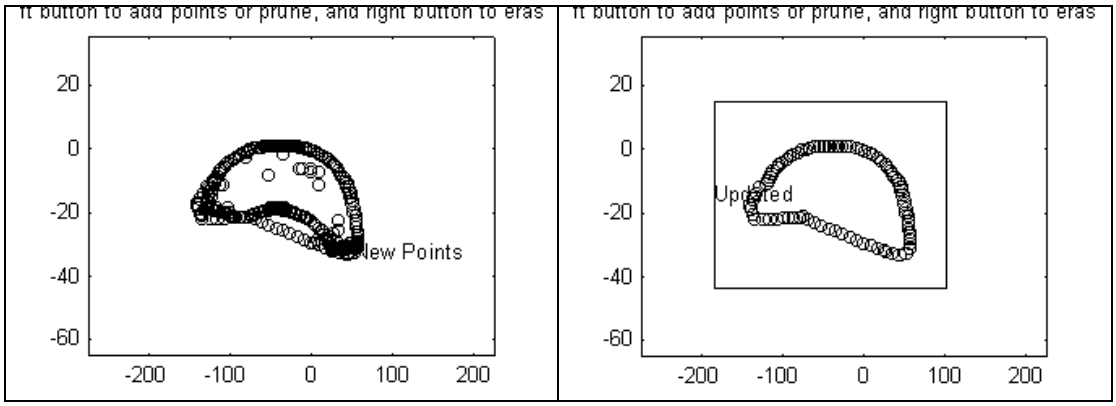

Figure 4.18. The union of the template files `ex4_5.tpl` *and* `ex4_5ag.tpl` *was computed with the command* `tplunion('ex4_7','ex4_5','ex4_5ag');` *Then the command* `tplupd('ex4_7',[10 2]);` *was issued. Figure 4.16 shows the union of the two templates for* 76.6549 *rad/s. On the left in this figure (a) the template is shown after adding points with the* `tplupd` *command, in order to "fill out" the experimental points outside the model template. On the right, (b), the template is shown after pruning.*

4.8 RFF with accuracy enhancement and reduction

We refer to Section 2.2.3 and Figure 2.8, where a template computed with the real factored form method was "thinned" using the command `tplreduc`. For all the templates in a template file, the same procedure is performed as follows. Consider the plant description `ex2 1a.m` in Figure 2.2. Compute the templates with 10 times higher phase accuracy:

```
ctpl('ex4 8','ex2 1a','rff [0.1 1]');
...
 Computing time  :  [min]  =  10.17
```

Then issuing the command `tplfop` with the template reduction option (`op` = 3), and a demanded accuracy of [2 1] by the command

```
tplfop('ex4 8a','*',[],'ex4 8',[],[],[],3,[2 1]);
```

results, after about 3 minutes of computing time, in templates whose size is about 1/20 of the original ones, and with a *de facto* resolution of about 1 degree and 1 dB. The difference the templates of `ex4 8a.tpl` and `ex2 1a.tpl` of Chapter 2 is that the edge accuracy is 10 times better. The user is invited to check this by the command `showtpl`, as in Section 2.2.2.

A faster alternative to enhance the accuracy of the *extreme* template edge points of the templates is the following: Compute one template file with the real factored form with moderate resolution, say [1 1], and a second template file with the recursive edge method with low resolution, say [10 10]. Form the union of these template files, with `tplunion`, as in Section 4.7, and possibly reduce the number of template points with `tplfop` or `tplprune`.