```
Calculating output functions ...
Resetting options ...
Done!
```

The output parameter `SupGamma` is less than 1. In a disk - type problem this means that there is a solution that satisfies the (smooth) constraints. The figure NED is a measure for numerical error or noise, and OT1, OT2 are diagnostics for optimality: OT near 0 means the calculated solution is nearly optimal. We say more on this later.

Another important output is a discrete set of values of the optimal closed loop, which can be recalled with the symbol `T` .

Conclusion

The conclusion we draw with respect to the OPTDesign run above is that there are solutions to the modified Design problem obtained after the envelope was smoothed, and that accuracy of the output is acceptable.

5.3.1 Generating a rational closed loop

A low order rational representation of the closed loop transfer function T can be generated after an OPTDesign[] run with the function RationalModel[]. By default it finds a rational that approximates T.

```
Trat = RationalModel[]

Error = 0.171731

 -5 + s     (-4.67853 - 1.92117 s) (-5 + s)
--------- + -------------------------------
        2              2
(2. + s)       (2. + s)  (3.769 + 1. s)
```

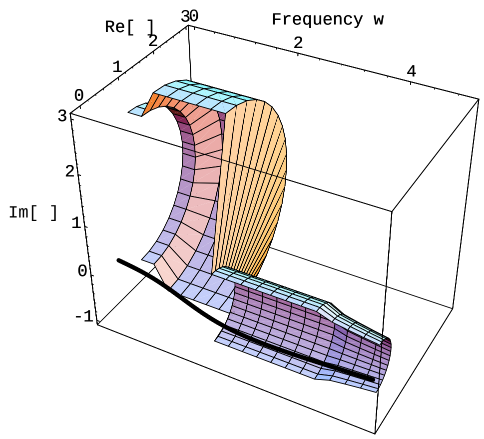
RationalModel[] produces a rational of the form

$$T_{rat}(s) = A(s) + B(s)T_{1rat}(s)$$

The proper rational functions A, B depend on the plant and incorporate automatically the internal stability requirements on T. The rational T_{1rat} is produced by the optimization procedure. If you run OPTDesign[] first, you do not have to specify A, B or $T1$ as an input to `RationalModel[ ]`.

The Error figure above refers to the fit of $T1$ and not to the fit of T. Usually we can get information from a plot. With the command shown below we obtain a 3D picture of the envelope, together with the plot of our choice of rational closed loop.

```
EnvelopePlot3D[Radius->r0,Center->k0,ClosedLoop->Trat]
```




```
-Graphics3D-
```

5.4 Producing a rational compensator

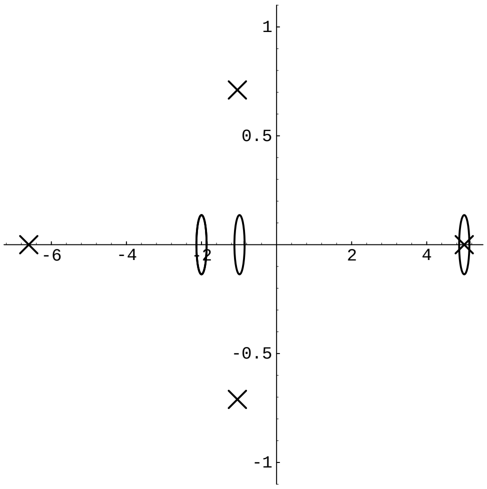
We obtain the compensator from the formula $C = (1/P) * T/(1-T)$:

```
comp1 = Together[ 1/p[s] Trat/(1-Trat) ]
```

```
                 2                               2
  1.15147 (2 + s)  (-4.93682 - 4.01264 s + 1. s )
---------------------------------------------------
                                         2       3
(-5 + s) (10.5283 + 15.3797 s + 8.69017 s  + 1. s )
```

The compensator seems to have a pole at s = 5, but this shouldn't be the case since OPTDesign guarantees an internally stable system. The zeros and poles of **comp1** are shown below:

```
PlotZP[comp1]
```




```
-Graphics-
```

The figure shows that comp1 has a pole - zero pair at s=5. We verify this now:

```
comp1zeros = s /. NSolve[ Numerator[comp1]==0,s]

{-2., -2., -0.987364, 5.}

comp1poles = s /. NSolve[ Denominator[comp1]==0,s]

{5, -6.60224, -1.04396 - 0.710493 I, -1.04396 + 0.710493 I}
```

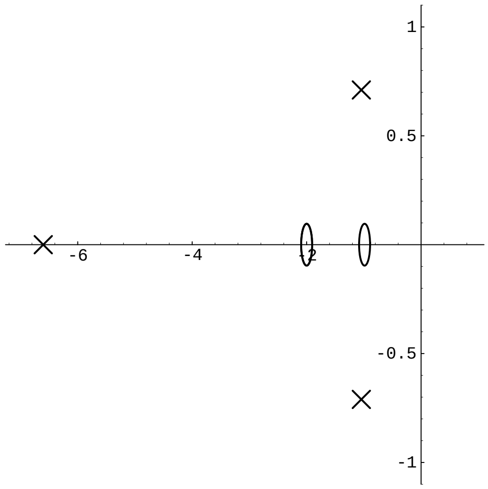
The standard Mathematica functions do not cancel terms with decimal notation. To do it we can use a function provided with OPTDesign:

```
comp2 = Cancel2[comp1,s - 5,s]

                                2             3
4.54766 + 9.15353 s + 5.74278 s  + 1.15147 s
--------------------------------------------
                                  2      3
  10.5283 + 15.3797 s + 8.69017 s  + 1. s
```

Hence `comp2` results from the simplification of `comp1`.

```
PlotZP[comp2]
```




```
-Graphics-
```

5.5 How good is the answer that we obtained?

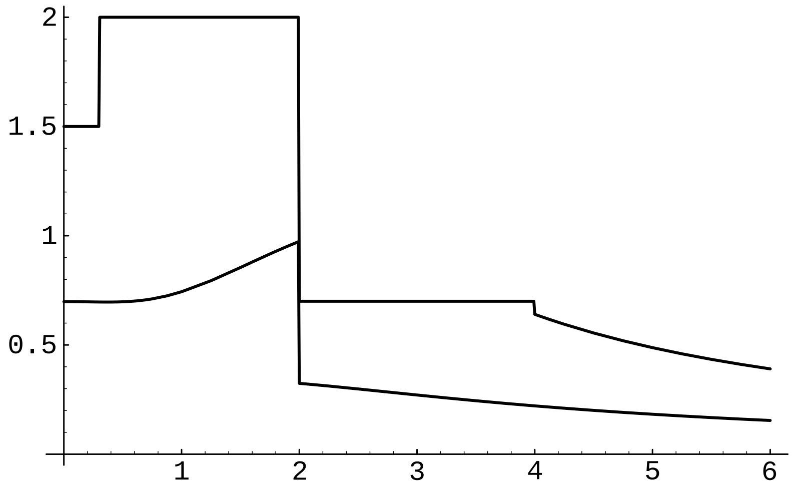
OPTDesign smooths out the envelope automatically, thus distorts it a bit near points of sharp changes. Another source of error is the rational approximation to the answer given by OPTDesign[ ]. An important question is, do we obtain in the end a closed loop or a compensator that solve the original problem?

It is clear from the plots obtained above that the compensator is proper and does not cancel RHP poles or zeros of the plant. Also Trat is stable, thus for this choice of compensator the system is internally stable.

To verify that we have met the performance requirements we can do it in the following manner:

```
closedlp[w_] = ( Trat /. s-> I w );

Show[ Plot[Abs[k0[w]-closedlp[w]],{w,0,6},
                DisplayFunction->Identity],
      Plot[r0[w],{w,0,6},
                DisplayFunction->Identity],
     DisplayFunction->$DisplayFunction]
```




```
-Graphics-
```

5.5.1 Plots of the calculated closed loop function

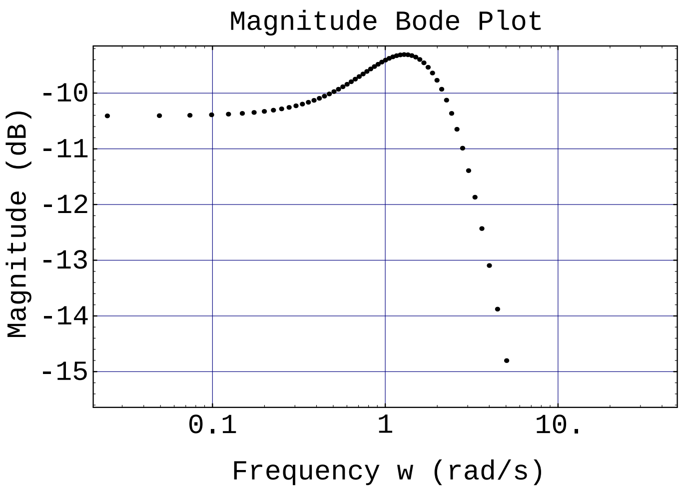
Plots can be produced with the objective of evaluating the choice of grid-size and smoothing input parameters versus the results of the run and the frequency bands of interest. If the results are not satisfactory the problem has to be re-run with modified input parameters.

First the rational closed loop is sampled on the grid used in the calculations above.

```
Tdisc = Discretize[closedlp];
```

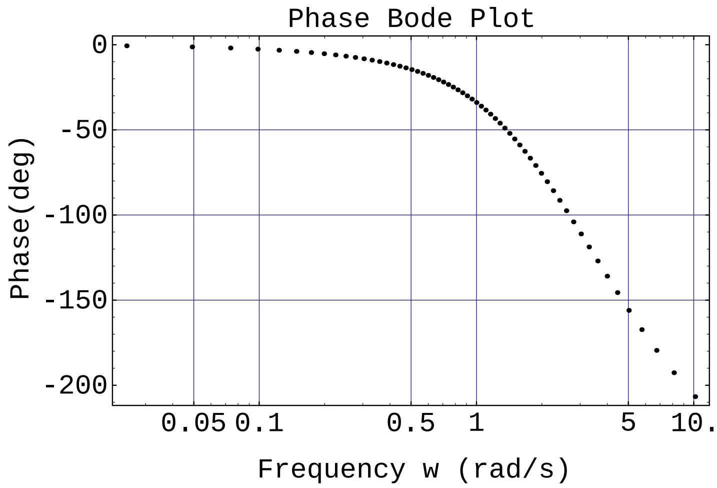
The Bode plots are generated as follows.

```
bm = BodeMagnitude[Tdisc,PlotLabel->"Magnitude Bode Plot"];
```



One can see in the magnitude plot of the closed loop that most of the sample points are located in the frequency band $0.1 < \omega < 10.$, which we judge (bassed on the fact that both the input functions k0 and r0 change mostly in this band) to be acceptable for the design problem we are considering. We will see in a later Section how to give input parameters to get enough sample points in the frequency bands of our choice. Here is a plot of the phase of T:

```
BodePhase[Tdisc, PlotLabel->"Phase Bode Plot"]
```




```
-Graphics-
```

Plots for the open loop L = p c can be produced in similar fashion.

5.6 Reducing the error

The rational approximation entails an error, as does the smoothing of the envelope of constraints by OPTDesign. Re-running OPTDesign with modified grid or with more smoothing may be necessary in some cases to improve the answer.

There are two input parameters that specify the grid. One parameter is the number of gridpoints, specified with the option $Ngrid \rightarrow nn$. Another parameter is a scaling, which is specified as $GridSpread \rightarrow nn$. Here are the values used in the run above:

```
{Ngrid, GridSpread} /. Options[OPTDesign]
```

```
{128, 1.}
```

Therefore 128 gridpoints were used, and most of the points are located near the frequency $\omega = 1$, as the Bode plots above indicate.

Another input parameter is the amount of smoothing, specified as $Nsmth \rightarrow nn$. We used above

```
Nsmth /. Options[OPTDesign]
```

```
1
```

The following is an example of what you would type to re-run Anopt, with more gridpoints, more smoothing and gridpoints accumulating near $\omega = 5$.

```
OPTDesign[p,Center->k0,Radius->r0,
           Ngrid->256,
           Nsmth->5,
           GridSpread->5
          ];
```

5.6.1 Improving the fit

The order of the rational model of T can be increased to obtain a better fit:

```
Trat2 = RationalModel[DegreeOfDenominator->5]
```

```
Error = 0.122737

                                                     2            3
 -5 + s     (-5 + s) (-4.80837 - 4.65535 s - 5.22911 s  - 1.87092 s )
--------- + ---------------------------------------------------------
       2           2                                    2       3
(2. + s)      (2. + s)  (3.74005 + 3.45048 s + 3.89256 s  + 1. s )
```

One sees that the improvement is small in this example.

There are a couple of things to keep in mind when using **RationalModel**: 1. The routine RationalModel[] may fail to work satisfactorily when (a.) the numerical error diagnostic in the optimization procedure is not small, or, (b.) the DegreeOfDenominator option specifies a large number.

2. Recall that if the plant is strictly proper, then the relative degrees of the plant, closed loop and of the compensator functions satisfy

$$d(T_{rat}) = d(P) + d(C)$$

Both $d(P)$ and $d(T)$ are set in advance as input to OPTDesign[], when P is given explicitly, and when a roll off constraint is specified on T_{rat}. This determines $d(C)$ once and for all.

5.6.2 When the relative degree of the compensator is not zero

In the run above we obtained a compensator with relative degree $= 0$, that it, the compensator is proper but not strictly proper.

If you want to specify the relative degree of the compensator to be $nn = 1$ or larger, set the roll off constraint so that the radius function goes to 0 asymptotically with $1/w^{(nn+pp)}$, where pp is the relative degree of the plant.

Here is an example where the relative degree of the compensator is specified as $nn = 2$:

```
k0[w_]= Which[ Abs[w] < wb, 1.0,
                       True, 0.0];

r0[w_] = Which[Abs[w] < wt,  alphat,
               Abs[w] < wb,  alphagpm,
               Abs[w] < wr,  alphab,
                     True ,  alphar*Abs[p[I w]/w^2 ]
                ];
OPTDesign[p,Center->k0,Radius->r0];
```

5.7 Other performance functions (including weights)

5.7.1 Modifying the performance

OPTDesign[ ] is powerful enough to take a great variety of performance functions as input. In the sections above we discussed the class of disk performance functions. These have the form

```
Abs[ (center - z )/ radius ]^2
```

where "z" represents the closed loop transfer function. By choosing the center and radius you can solve many problems. On occasions you may want to use performance functions which are not of this type. We now show how to do this via an example.

```
p[w_] = 2/(1 - s/2)^2;

OPTDesign[p,Performance->Abs[(1+I w)/(4+w^2)*(1-z)]^2 +
                                 Abs[(w^4)/(4+w^2)*z]^2]
```

For OPTDesign[] to work without problems, a requirement is that performance functions should be positive, differentiable functions of z. Note that weight functions can be made part of the performance when the latter is defined as above.

In our example above we specified a performance of the form

```
Abs[ W1 (s-z)]^2 + Abs[ W2 z]^2,
```

where W1 is not proper. This is not a problem: in fact, it is necessary in order for the problem to be well defined as a mathematical optimization problem! This is a delicate point and we direct the reader that wants more details on this to a later chapter in this book.

5.7.2 Non-rational weight functions

Weights can also be easily defined as functions of frequency. You can even define them by pieces, as the following example shows. First you define weight functions:

```
fn1[w_] := If[  Abs[w] < 1, 2., 2./Abs[w] ] ;

fn2[w_]:= Which[ Abs[w] < 1,    Abs[w]^2,
                 Abs[w] < 2,
```

```
Show[Plot[fn1[w],{w,0,6}],Plot[fn2[w],{w,0,6}]]
```

Now you are ready for the OPTDesign[] run.

```
OPTDesign[ p,Weights -> { fn1, fn2},
           Performance ->  W[1] * Abs[  1 - z ]^2 +
                              W[2]* Abs[z]^2  ]
```

OPTDesign[ ] requires that the weight functions defined in this way be bounded on the frequency axis. However, if the weight functions enter the performance function directly as rational function, they may not be bounded, as we mentiond before.

5.8 OPTDesign[] output

5.8.1 Interpretation of output diagnostics

As the computer run of OPTDesgin progresses, some numbers are printed on the screen under the labels OT1, OT2, SupGamma. These give information on the progress of the run. There is more output that is not printed, e.g., the discrete optimal closed loop T. The diagnostics on accuracy and performance are discussed below.

SupGamma. In the run above the output SupGamma = 0.75 is less than 1, thus it implies that there are solutions to the Design problem (after smoothing). However, this conclusion must be analysed under the light of other output diagnostics NED, OT1 and OT2. Note: Recall that the input to OPTDesign[] defines a performance function Gamma that depends on the frequency and on the designable transfer function T. In this chapter, this function has the form

$$Gamma(w,T) = 1/radius(Iw)Abs[center[Iw] - T[Iw]]^2.$$

The number

$$supgamma(T) := sup_w Gamma(w,T)$$

is a performance index: a real number that measures in a specific way "how good" a designable T is. If supgamma(T) < 1, then T meets all the specifications. The number SupGamma output to OPTDesign corresponds to the value at the optimal T. In a sense, this number tells you what is the best you can do with the given specifications.

NED The error diagnostic NED is 0.02 approximately. This figure says that the internal optimization routine Anopt encountered numerical noise of this magnitude when calculating the solution to the problem. We will consider this acceptable since NED is whithin 2This numerical noise is frequently associated with lack of smoothness of the data to the problem; for example, NED is large if functions such as the radius or center have jump discontinuities or sharp corners. The automatic smoothing done by OPTDesign helped in obtaining here a figure which is not "too large". Smaller values of NED are obtained with choices of grid on the I w axis with more points, or with more smoothing. This will be discussed in a different section.

OT1, OT2. The error diagnostics OT1 = 0.006 and OT2 = 0 give (loosely speaking) an indirect measure of how close to optimal the solution is. The best result possible is OT1 = 0 and OT2 = 0. In our example this was not obtained because of a nonzero NED.

5.9 Controlling the numerical error

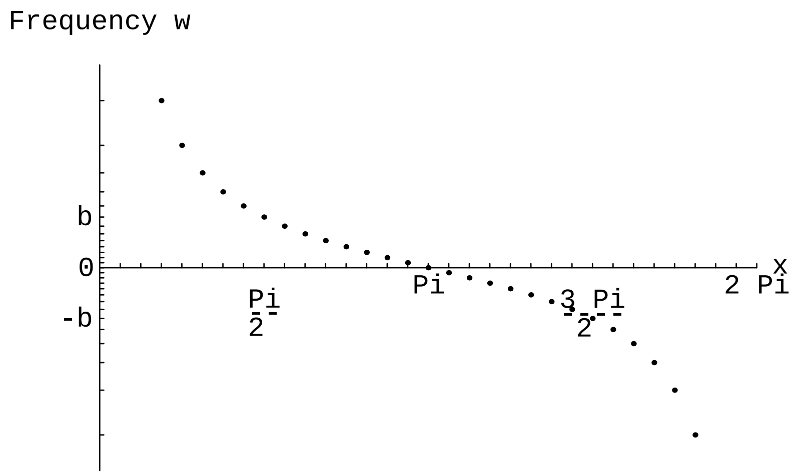
To reduce the numerical error as shown in NED, you can try two things:

1. Modify the grid, by increasing the number of points or by changing the normalization used.

2. Increase the amount of smoothing done automatically on functions.

5.9.1 Options Ngrid and GridSpread

A grid on the frequency axis is produced by OPTDesign by taking a grid of equally spaced points "x" on the interval [0,2Pi), and then calculating samples on the frequency axis via the formula w = Cot[x / 2]. The point x = 0 corresponds to infinite frequency, and "b" is a normalization constant that can be set by the user. Below is a plot of angles x vs. frequency w.



Observe how the gridpoints on the w axis accumulate near the origin. Actually, roughly half of the gridpoints will be located in the interval [-b , b], the two endpoints of the interval corresponding to x = Pi/2 and x=3 Pi/2. It is important to remember this, since good numerical behavior of computer runs depend a great deal on the choice of the grid, i.e., on the choice of the number of gridpoints and of the normalization parameter "b".

The number of gridpoints is set with the option Ngrid, and the normalization constant "b" is set with the option GridSpread.

The parameter "b" has to be chosen so that [0 , b] includes approximately what you consider low and midrange frequencies. A rule of thumb for choosing "b" is to set it equal to the bandwidth.

5.9.2 Option Nsmth

In terms of pictures, increasing the degree of smoothness corresponds to "rounding the corners" in the modified envelope of solutions (roughly speaking). But one must make sure that doing this results in an envelope that changes the original envelope by an acceptable amount.

OPTDesign has the ability to automatically "smooth out" the input data, so that the user avoids some additional work. The price paid by the user is less control on the envelope, but if smoothing is used carefully it will give the desired results. On the other hand, in some cases one must resort to setting the amount of smoothing in order to obtain good numbers.

This automatic smoothing is specified with the option `Nsmth`. The input `Nsmth-` >`1` indicates a small amount of automatic smoothing; more smoothing is achieved with `Nsmth-` >`5` or `Nsmth-` >`10`. The input `Nsmth-` >`0` corresponds to no automatic smoothing. Some precautions must be taken when setting this input since smoothing modifies the envelope.

Part II

Advanced Design

Chapter 6

Examples

Three examples are presented in this chapter. They illustrate the theory developed in previous chapters, as well as the use of some techniques for solving problems in practice. Each example is preceded by a chapter with material related to the example.

The design process involves extensive use of computers and algorithms, so some time spent studying their efficient use is beneficial. We discuss briefly in Section 6.1 numerical issues that arise in a wide class of algorithms for solving optimization problems. The first control example we solve, the design of a control of a slide drive, is presented in Section 6.2.

Many practical situations lead to control design problems with requirements on the time domain functions associated with the system. To use frequency domain methods, one must translate time domain information to the frequency domain. We take a first look at time domain requirements in Section 6.3. The requirements overshoot and the settling time and a method for translating information to the frequency domain are introduced in this section. The method is applied in our solution to a design problem in Section 6.4. A more sophisticated method for translating time domain information is presented in Chapter 6.

The generic problem in control, as in many other disciplines, is to achieve an objective subject to certain constraints. Achieving the objective usually involves a tradeoff: one constraint cannot be attained to perfection without the rest doing badly. These are called "competing constraints". In the design problems studied here, this situation appears when different constraints are active at the same frequency. In Section 6.5 we discuss ways to treat competing constraints with our methods; this is used to solve the design problem in Section 6.6.

6.1 Numerical Practicalities

Several difficulties may arise when OPT problems are solved by computer, because algorithms for solving *OPT* will not work, or will work in an unsatisfactory manner, if any of the following occur:

- The performance function Γ is not continuous. In circular performance case, this occurs when the radius function or the center function is not continuous.
- The value of the performance $\Gamma(\omega, T(j\omega))$ is extremely large at some frequencies ω, regardless of the value of T. In circular performance problems, this is the case when the radius gets very close to 0 at one or more frequencies.
- Points on the $j\omega$ axis for sampling functions are badly chosen.
- At certain frequency ω, the value of the performance $\Gamma(\omega, T(j\omega))$ depends on ω but does not depend on T.

We discuss these issues below. The emphasis is on circular performance functions, but the comments apply, for the most part, to other performance functions as well.

6.1.1 Discontinuous functions and smoothing

Depending on the way in which requirements are put together, the performance function may not be continuous. For example, suppose that you have a case in which the only two constraints active near frequency $\omega = 1$ are: a gain-phase constraint active for $\omega < 1$, and a bandwidth constraint active for $\omega \geq 1$. A circular performance problem formed with these constraints has a center function with values that drop from 1 to 0 near frequency $\omega = 1$. This is undesirable; most algorithms for solving OPT have numerical problems when dealing with discontinuous performance functions. Even if the optimal designable function T could be computed, in many cases it wouldn't be acceptable because it is not continuous.

To circumvent this problem in the circular performance case, it is necessary to "smooth " R and K before forming the performance, which modifies these functions to make them continuous and differentiable. Smoothing changes the problem being solved, but it still should produce good approximate solutions if handled with care. The trade-off is closeness to the original function vs. good numerical properties. Smoothing is usually based on convolution or spline interpolation. Figs. 6.1 and 6.2 illustrate the effect of smoothing.

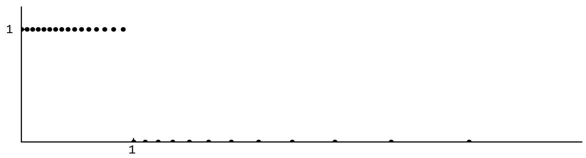


Figure 6.1: Sample values of a discontinuous function

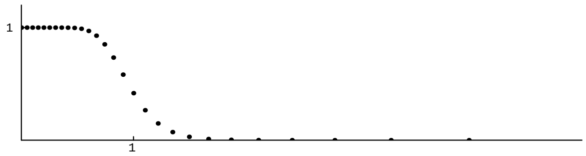


Figure 6.2: Smoothed sample values of a discontinuous function

6.1.2 Vanishing radius function

When the radius in circular performance problems vanishes, the mathematics for solving OPT break down. However, we show below that a vanishing radius is not a difficulty if the problem is set up correctly.

A radius formed with the performance measures given in Chapter 3 will vanish at infinite frequency because of the closed loop-roll off constraint. This constraint is

$$|T(j\omega)| \leq \alpha_r \frac{|P(j\omega)|}{|\omega|^{d_c}}, \qquad \text{for large } |\omega| \tag{6.1}$$

where d_c is the degree of the desired compensator, as specified by the designer. Consider now the parameterization of all T obtained from the requirement of internal stability:

$$T = A + BT_1$$

where A and B are $\mathcal{RH}^\infty$ functions that satisfy certain interpolation conditions at RHP zeros and poles of the plant (see Theorem 9). Because of the way it is chosen, B has relative degree given by $d(B) = d(P) + d_c$. Therefore the right-hand-side in the inequality

$$\left| \frac{K(j\omega) - A(j\omega)}{B(j\omega)} - T_1(j\omega) \right| \leq \frac{|P(j\omega)|}{|w|^{d_c}|B(j\omega)|} \tag{6.2}$$

does not vanish at infinite frequency. That is, when the problem is stated in terms of the new parameter T_1, the new radius function does not vanish at infinite frequency.

A similar situation occurs when the radius has a zero at certain frequency ω_0. This happens when the plant has a zero or a pole on the $j\omega$ axis, so there is a plant bound or compensator bound requirement in the design problem. We leave as an exercise for the reader to show how the technique of reparameterization can produce a circular performance function whose radius does not vanish.

6.1.3 Sampling functions on the $j\omega$ axis

To solve OPT by computer one must choose a grid of points on the $j\omega$ axis. These points are needed for sampling functions. To have an adequate discrete representation of functions, the grid must contain many points in the frequency bands where the functions being sampled change alot. For circular performance functions, the center K and the radius R determine which choices of grid points are appropriate. Figs. 6.3 and 6.4 illustrate different selections of grid.

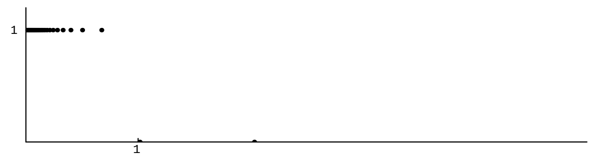


Figure 6.3: High frequency undersampled

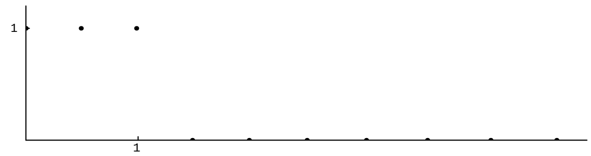


Figure 6.4: Low frequency undersampled