20.M APPENDIX: MATLAB SCRIPTS

MATLAB scripts used to generate simulations for this chapter are described here. They can be downloaded for a more complete understanding. Some scripts have features that could be incorporated in others where that is found to be desirable.

20.M.1 Simulating Bandlimited Noise

These scripts filter the noise. They vary in the prefilter used and in the order of the loop. They all gather statistics. `NB1bw` and `NB2bw` can also simulate time responses, providing the same kinds of displays as does `NLPhN` (Appendix 17.M.1). In these simulations, the condition

$$f_{\max} >> f_L \tag{13.26}$$

is not commonly met so the use of dual noise sources, according to the model of Fig. 13.5, becomes important.

20.M.2 First-Order Loop with First-Order Prefilter

`NB1` simulates a first-order loop preceded by a first-order filter in which the signal is centered. The filter has a complex pair of poles but the equivalent baseband (after the PD) response is that of a single-pole low-pass filter. The program output gives information on the variance of the noise sources as well as the filtered variances. The net noise bandwidth of the filter (video equivalent) and the loop combined is computed and, if we specify variance rather than noise density, the variance is set in this net bandwidth.

The sampling period (1/`SmpRate`) must be short compared to the filter time constant (reciprocal of video bandwidth in rad/sec) for the filter to be accurately simulated by the method used. Therefore, a sampling rate that is adequate for loop simulation may have to be increased for filters that are much wider than the loop. We can judge the adequacy of the sampling rate by determining that the results do not change significantly when it is increased; this method is widely applicable.

20.M.3 First-Order Loop with Higher-Order Prefilter

`NB1bw` simulates a first-order loop preceded by a fourth-order Butterworth (maximally flat) filter. Here we can choose to set the noise in either the prefilter or the net noise bandwidth. The net bandwidth is determined by integration of the loop's power transfer response from zero to the prefilter's noise bandwidth, which is the ripple bandwidth, adjusted slightly based on the noise power that typically comes through the filter.

Three parameters are iterated in the script `NB1bwi` and the output is written both to the screen and to a file, `runrec.txt`. The latter feature becomes very important for recovering results if the computer should crash during a long run.

The sampling rate may again have to be increased, beyond what is required for the loop, when filter-to-loop bandwidth ratios are high. That is to prevent aliased passbands from providing significant noise power at frequencies to which the loop will have significant response.

There can also be a problem at the other extreme, where the filter function may fail to perform properly because the filter bandwidth becomes too narrow relative to the sampling rate. For this reason the filter is sampled at a slower rate than the loop at narrow relative bandwidths. The sampling-rate reduction factor, `SRd`, is computed automatically but the criterion might not always be optimum.

In order to alleviate this problem, a first-order data hold is employed in `NB1ch1` (first-order loop, Chebychev filter, first-order hold). The data holds otherwise employed are zero-order holds. They hold the sampled data steadily until the next sample, but a first-order hold predicts the sampled value based on a straight line passing through the last two samples [Franklin, et. al, 1990, p. 119]. The advantage for us is that the frequency response of the first-order hold is lower in the vicinity of the sampling harmonics, thus reducing the aliased filter energy for narrow filters and allowing the sampling rate to be reduced (which moves more aliased pass bands close to the filter bandwidth).

The even-order Chebychev filter was problematical at narrow loop bandwidths because the noise density falls below nominal at the center frequency (zero frequency equivalent at baseband), producing a lower-than-expected loop output variance. This led to the use of Butterworth (maximally flat) filters. The problem might also have been alleviated by changing the filter order by one.

20.M.4 Second-Order Loop with First-Order Prefilter

In `NB2,` a first-order prefilter precedes a second-order loop. Here the net noise bandwidth is determined by contour integration of the net power transfer function, which is enabled by MATLAB'S ability to find poles and residues.

20.M.5 Second-Order Loop with Higher-Order Prefilter

`NB2bw` and `NB2bwi` simulate a second-order loop with a Butterworth prefilter, the latter iteratively. Here we must choose to set the linear phase variance either in the prefilter or the loop bandwidth. (The script does not find the net noise bandwidth.) Again, the iterated script outputs to a file as well as the screen.

20.M.6 Simulating Limiting

By setting the variable `Limiting` to 1, we can cause the scripts with higher-order Butterworth prefilters (`NB1bw`, `NB2bwi`, etc.) to simulate a loop following a limiter. In that case, the two independent noise sources are used to generate the input phase of the signal plus noise, according to the model in Fig. 16.9. Only the phase of that vector sum is retained. The amplitude is that corresponding to K'_p. This simulates hard limiting of a signal in the presence of noise.

20.M.7 Simulating AGC

Script `NB1bw` will simulate AGC when the variable `Limiting` is set to 2. This is an additive noise simulation (as it is with `Limiting` set to 0) but the gain K is multiplied by η_A from Eq. (16.8), which is a function of the ρ_i, the S/N into the prefilter, so it simulates the reduction of gain as a function of ρ_i. Thus `NB1bw` may be the most versatile of these scripts, since it can simulate additive noise, limiting, or AGC and can produce either statistical output or time response. Of course, its complexity is correspondingly great.

20.M.8 Simulating an Eccentric Noise Band

Setting `OoverL` = o to a nonzero value in `NB2bw` or `NB2bwi` (both described above) allows simulation of a noise band that is centered at a frequency separation from the input signal of $o[B_n]_{Loop}$. If o is set to a large value, the sample rate might need to be increased.