NOTES ON THE SCHREIER TOOL SET

Disclaimer

I originally wrote this software for my own private consumption, so you will find that the documentation is generally sparse, that the commands are often unforgiving, and that there are a few bugs. Nonetheless, I find the programs indispensible and I hope you will too. Insert standard legal disclaimers here absolving me of liability regarding the true utility of these programs.

First of all

To be able to use my software, you need to do the following:

`ftp` the file `delta-sigma.tar.Z` from ftp from `next242.ece.orst.edu` (`128.193.48.65`). Please send me (`schreier@ece.orst.edu`) e-mail if you obtain a copy. That way, I can keep you posted regarding any updates and improvements that are made.

Compile the programs by typing `make` in the src.delta-sigma directory. Either move the executables to a directory which is in your search path or add the bin subdirectory to your search path.

Copy the files in the `examples` subdirectory into one of your own directories.

After doing this, the following commands should work (from within the directory into which you just copied the example files).

Some examples

`% sine N=256 f=3 A=-20dB`	Output a cosine wave, 256 samples long with 3 cycles and an amplitude of -20 dB (0.1).
`% noise N=8 rms=-20`dB	Output 8 samples of uniformly distributed white noise with an average power of -20 dB.
`% modulator 1stOrder`	Simulate a delta-sigma modulator whose noise transfer function is given in the file `1stOrder`. You type numbers to it (the input samples), and it responds with the output of the modulator. Type anything that is not a number and the program will stop. Likewise with `modulator 2ndOrder` etc.

And now the fun begins. Try

```
% sine N=1024 f=3 | fft
```

This outputs the magnitude of the bins in a 1024-point FFT of a cosine wave. All bins are (nearly) zero, except for bins 3 and -3. (The fft starts with bin 0, counts up to bin *N*/2, which is equivalent to bin -*N*/2, and then continues counting up to bin -1.)

Next, try

```
% sine N=1024 f=3 A=-20dB | modulator 1stOrder | fft
```

You get a stream of data: the (magnitude of the) FFT of the output of the first-order modulator with a sine wave input. (Graph this with gnuplot, if you like.) Notice the low numbers in the low-frequency bins, except of course for bin 3 (and -3), and the larger numbers (shaped quantization noise) in the higher bins.

If we take the oversampling ratio to be 32, we can get an idea of the signal-to-noise ratio for this modulator by comparing the signal power to the in-band noise (the non-signal components in bins 0 to 16). I have a program which does this. To try it, type

```
% sine N=1024 f=3 rms=-20dB | modulator 1stOrder | spectralAnalysis f0=3 f1=0
f2=16 N=1024
```

It should respond with the signal-to-noise ratio (in dB). From the linear model, we would expect a noise power of about -45 dB; a signal power of -20 dB thus implies an SNR of 25 dB. I get 31.5 dB SNR by simulation, 6 dB better than expected.

A rectangularly-windowed FFT has problems with spectrum smearing, and these can cause severe errors when estimating the magnitude of small components surrounded by strong ones. I have implemented a Hann window for the spectralAnalysis program and it is invoked with the `-w` option, as follows:

```
% sine N=1024 f=3 rms=-20dB | modulator 1stOrder | spectralAnalysis -w f0=3
f1=0 f2=16 N=1024
```

These commands are a nuisance to type, especially if one wants to sweep the amplitude or frequency. Use the shell script `snrCheck` to automate this process. You can supply arguments to change its defaults, or you can make a copy of it (it is in `~schreier/bin`) and edit it to your heart's content. Try

```
% snrCheck 2ndOrder
```

This produces three files:
1) A `2ndOrder.p` file which contains SNR vs. input power data.
2) A `2ndOrder.f` file which contains SNR vs. frequency data.
3) A `2ndOrder.r` file which lists the parameters (frequency, amplitude, oversampling ratio…) that were used to create the above files.

To suppress the generation of the `2ndOrder.f` file, use the `-f` option. You can override the default values used in the simulation on the command line by making parameter assignments. Both options and parameter assignments must be after the modulator name. For example,

```
% snrCheck 1stOrder f=5 p=-10
```

uses the modulator whose NTF is given in the file `1stOrder`, changes the default frequency to 5 cycles and the default input power to -10 dB.

To generate a NTF, use the `synthesizeNTF` command:

```
% synthesizeNTF n=6 Hinf=1.6 R=32 w0=0.5 -opt > myModulator
```

This command generates a NTF file readable by the modulator program. The NTF is sixth-order, has an out-of-band gain of 1.6, uses an oversampling ratio of 32, has a center frequency of 0.5π, and uses optimized zeros. The default values for these parameters are 3, 1.5, 64, and 0 with non-optimized zeros.

To map the NTF onto the cascade-of-resonators topology, use

```
% realize myModulator
```

The program responds with the required coefficient values. (At the moment, this program only works for even-order modulators.) We will learn how to interpret these numbers later in the term.

To check the sensitivity of the SNR to the coefficients in the cascade-of-resonators topology, use

```
% sensitivity myModulator trials=1000 tol=.01
```

This command varies the coefficients randomly by 1% for 1000 trials and evaluates the in-band noise power for each perturbed modulator. It prints the 95th percentile and the largest values of N_0^2 that resulted.