Mkfilter

Introduction

`mkfilter` is a program which designs an infinite impulse response digital filter from parameters specified on the command line, using the bilinear transform method.

The source code of the program (in C) is at

```
http://dcpu1.cs.york.ac.uk:6666/fisher/software/mkfilter
```

and there is a World Wide Web form-based front end at

```
http://dcpu1.cs.york.ac.uk:6666/fisher/mkfilter
```

In normal use, the output of the program is a set of S- and Z-plane pole and zero positions, the filter recurrence relation, and some other information in human-readable form. By specifying the `-l` parameter, computer-readable output is produced which can be piped into other programs. In particular, the program `gencode` takes the `-l` output from `mkfilter` and generates a piece of C code which implements the filter, and the program `genplot` takes the `-l` output and generates a `gif` file containing phase and magnitude graphs. `gencode` has no parameters. `genplot` has one, the name of the output `gif` file.

`genplot` uses the `gd` gif manipulation library from Quest Protein Database Center, <http://siva.cshl.org/gd/gd.html>.

A warning about higher-order Bessel filters: although an analogue Bessel filter has an excellent approximation to a linear phase characteristic, the pre-warping inherent in the bilinear transform design method upsets this, and the corresponding digital filter often deviates significantly from linear phase. This is particularly evident in the higher-order Bessel filters (say $N > 2$). You're advised to study the graphs produced by `genplot` before you use any Bessel filter generated by this package.

Mkfilter Command Line Parameters

"[]" means "optional".

The following parameters are required:

`-Be`
`-Bu`
`-Ch` *r*
	Filter type: Bessel, Butterworth or Chebyshev, respectively. Exactly one of these options must be specified. The parameter *r* is the passband ripple in dB, meaningful for Chebyshev designs only. NB: $r < 0$.

`-Lp`
`-Hp`
`-Bp`
	Pass type: Low-pass, High-pass or Band-pass, respectively. Exactly one of these options must be specified.

`-a` a_1 [a_2] Corner (–3dB) frequency/ies, as a fraction of the sampling rate; i.e. 0.5 is the Nyquist frequency. For low-pass and high-pass filters, only a_1 is required. For band-pass, both a_1 (the lower corner frequency) and a_2 (the upper corner frequency) are required. NB: $0 < a_1 < 0.5$; $0 < a_2 < 0.5$; $a_1 < a_2$.

`-o` N Order of filter. NB: $1 \le N \le 10$. The attenuation in the stopband is $6N$ dB per octave. The number of S- and Z-plane poles and zeros is related to N as follows.

	S-poles	S-zeros	Z-poles	Z-zeros
Low-pass:	N	—	N	N at –1
High-pass	N	N at 0	N	N at +1
Band-pass:	$2N$	N at 0	$2N$	N at –1, N at +1

Note in particular that in a bandpass filter the number of poles is twice the order.

The following parameters are optional:

`-e` Execute filter. Instead of printing a filter design, the program accepts a stream of real numbers on stdin, executes the filter, and prints the filtered output stream on stdout. Input and output values are presented one to a line.

`-l` List output in a form suitable for piping to another program. Instead of printing a filter design, the program prints the following: the command line; the number P of Z-plane poles; and $2(P+1)$ further numbers which are the coefficients of x_i and y_i in the recurrence relation. A trial run, specifying the same parameters both with and without `-l`, will make this clear.

`-p` p_1 p_2 … Select only poles p_1, p_2, …, where $0 \le p_i \le N$. Sometimes it is convenient (e.g. to reduce roundoff error) to partition a filter into sections. This option allows you to choose a subset of S-plane poles from the complete set. The subset selected must itself be partitioned into complex conjugate pairs. Some trial and error is required to achieve this! You will be told if you have not got it right.

`-w` Don't pre-warp the corner frequencies. This is useful when you're interested in the S-plane pole positions for an *analogue* filter. If you specify `-w` the generated digital filter will be *wrong*.

Examples

`mkfilter -Bu -Lp -o 4 -a 0.2`
Generate a 4-pole Butterworth low-pass filter with corner frequency 0.2 f_S; display pole & zero positions and filter recurrence relation

`mkfilter -Bu -Lp -o 4 -a 0.2 -l | gencode`
Generate C code for the above filter

`mkfilter -Bu -Lp -o 4 -a 0.2 -l | genplot graph.gif`
Generate phase & magnitude graphs for the above filter

Contact

Dr Anthony J. Fisher
Dept of Computer Science
The University of York
York YO1 5DD, U.K.

`fisher@minster.york.ac.uk`
`http://dcpu1.cs.york.ac.uk:6666/fisher`

19th August 1995