The `mkfilter` Digital Filter Generation Program

(`mkfilter` vsn. 4.5 and friends)

1. Introduction

`mkfilter` is a program which designs an infinite impulse response digital filter from parameters specified on the command line. Lowpass, highpass, bandpass and bandstop filters, with Butterworth, Bessel or Chebyshev characteristics, are designed using the bilinear transform or matched z-transform method. For most applications the bilinear transform method is recommended. The program can also design resonators with bandpass, bandstop or allpass characteristics. A companion program, `mkshape`, designs raised-cosine finite-impulse-response filters and Hilbert transformers. Other programs generate "C" code (in a variety of formats) from the compiled filter specification, and generate various graphs in "gif" format.

The source code of the programs (in C++) is at

`http://www-users.cs.york.ac.uk/~fisher/software/mkfilter`

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

`http://www-users.cs.york.ac.uk/~fisher/mkfilter`

The WWW front end is recommended. For most applications, it is the most convenient way to use the `mkfilter` package.

2. Mkfilter

In normal use, the output of `mkfilter` 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 or C++ code which implements the filter, and the program `genplot` takes the `-l` output and generates a "gif" file containing a frequency-domain or time-domain response graph.

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, and to consider using the matched z-transform for Bessel filters.

`mkfilter` can also design resonators. A resonator is a digital equivalent of a tuned circuit. There are three varieties:

1. the *bandpass* resonator, which has a high gain at its centre frequency and low gain elsewhere;
2. the *bandstop* resonator (or *notch filter*), which has zero gain ($-\infty$ dB) at its centre frequency and about unity (0 dB) elsewhere;
3. the *allpass* resonator, which has unity gain (0 dB) everywhere, with a phase shift which varies with frequency.

The phase response of the bandpass resonator approximates to $+\pi/2$ at frequencies below the centre and $-\pi/2$ at frequencies above the centre, and is exactly zero at the centre. The bandstop and allpass resonators both have approximately zero phase shift except at the centre frequency, at which the phase shift is nominally $\pm\pi$; however in the case of the bandstop resonator, since the gain is zero at the centre frequency, the phase shift at that frequency is not defined.

In both respects (magnitude and phase) the resonator behaves like a "real" analogue tuned circuit.

If you want a narrow bandpass or bandstop filter, a resonator is often more efficient and better behaved than a traditional (e.g. Butterworth) filter.

All types of resonator are designed directly in the z-plane. The bilinear transform is not used here. A bandpass resonator is constructed first; if you asked for one of the other types, the bandpass resonator is transformed accordingly.

The number of poles is fixed at 2, initially at $z = r \exp \pm j\theta$, where r is close to 1. Two zeros are added at $z = \pm 1$, to ensure zero response at d.c. and h.f.

The presence of the conjugate poles affects the response slightly: the "correct" pole positions are not exactly where you would expect them to be. Consequently, the initial pole positions are next refined iteratively, to place the peak as close as possible to where you said you wanted it.

If you asked for a bandstop or allpass resonator, the zeros at $z = \pm 1$ are then removed. For a bandstop design, new zeros are added on the unit circle at $z = \exp \pm j\theta$, where θ is the *unrefined* initial value of θ. This gives a zero response at the precise centre frequency. For an allpass design, zeros are added at $(1/r) \exp \pm j\theta$, where θ this time is the *refined* value, to balance the existing poles.

You may specify a value of `Inf` as the Q (quality factor) of a bandpass resonator, in which case you will get an oscillator (with poles exactly on the unit circle in the z-plane).

2.1. Mkfilter Command Line Parameters

"[ ]" means "optional"; "|" means "or". "*(Not Res)*" means that the parameter is neither required nor allowed for resonators, i.e. if `−Re` has been specified.

The following parameters are *required:*

`−Be` | `−Bu` | `−Ch` *r* | `−Re` *Q*

Filter type: Bessel, Butterworth, Chebyshev or Resonator, respectively. Exactly one of these options must be specified. The parameter r is the passband ripple in dB, meaningful for Chebyshev designs only. (NB: $r < 0$.) Q is the Q-factor of the resonator: the higher the Q, the narrower the peak. Values in the range 10 … 1000 are typical. The special value `Inf` specifies an oscillator.

`−Lp` | `−Hp` | `−Bp` | `−Bs` | `−Ap`

Pass type: Lowpass, Highpass, Bandpass, Bandstop or Allpass, respectively. Exactly one of these options must be specified. For resonators, only `–Bp`, `–Bs` and `–Ap` are allowed. For non-resonators, all pass types *except* `–Ap` are allowed.

`-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 (blt)	z-zeros (mzt)
Lowpass:	N	none	N	N at -1	none
Highpass	N	N at 0	N	N at $+1$	N at $+1$
Bandpass:	$2N$	N at 0	$2N$	N at -1; N at $+1$	N at $+1$
Bandstop:	$2N$	N at $+j\omega_0$; N at $-j\omega_0$	$2N$	$2N$	$2N$

where ω_0 is the geometric mean of the corner frequencies.

Note that in a bandpass or bandstop filter the number of s- or z-plane poles is twice the order. *(Not Res)*

`-a` α_1 [α_2] — Corner (–3 dB) frequency/ies, as a fraction of the sampling rate; i.e. 0.5 is the Nyquist frequency. For lowpass and highpass filters and resonators, only α_1 is required. For non-resonator bandpass and bandstop filters, both α_1 (the lower corner frequency) and α_2 (the upper corner frequency) are required. NB: $0 < \alpha_1 < 0.5$; $0 < \alpha_2 < 0.5$; $\alpha_1 < \alpha_2$.

The following parameters are *optional:*

`-l` — List output in a machine-readable form suitable for piping to another program. The format is described below.

`-p` $p_1\ p_2 \ldots$ — Select only poles p_1, p_2, …, where $0 \le p_i < N$. 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. *(Not Res)*

`-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` without `-z` the generated digital filter will be *wrong*. *(Not Res)*

`-z` — Use the matched z-transform design method. The default if `-z` is not specified is the bilinear transform method. This option implies `-w`. *(Not Res)*

`-Z` α — Add an additional z-plane zero at α times the sampling rate. This gives infinite attenuation at the specified frequency. Unless α specifies a frequency which is well within the stop-band, the filter shape is severely distorted. Check the graphs produced by `genplot` to make sure that the results are what you wanted.

2.2. Format of `–l` output

If the `–l` option is specified, `mkfilter` writes the following to standard output:

1. The command line which invoked `mkfilter`.
2. The magnitude of the gain in the pass-band, defined as follows. Let $H(\alpha)$ be the complex filter gain (transfer function) at frequency αf_s, where f_s is the sampling frequency. Define:

   For a lowpass filter: $g = H(0)$
   For a highpass filter: $g = H(0.5)$
   For a bandpass filter: $g = H((\alpha_1 + \alpha_2) / 2)$
   For a bandstop filter: $g = [H(0)\ H(0.5)]^{0.5}$

   The program outputs the magnitude of g.
3. This is followed by the number Z of z-plane zeros, followed by Z+1 further numbers which are the coefficients of x_i in the recurrence relation.
4. Finally, the number P of z-plane poles, and the P+1 coefficients of y_i, are listed in the same format. The last y coefficient is always –1.

A trial run, specifying the same parameters both with and without `–l`, will make this clear.

3. Mkshape

This program generates a finite impulse response filter with a raised-cosine magnitude response, or a Hilbert transformer. It is run by:

```
mkshape –{cr} α β n [–{lwx}]
mkshape –i n [–{lwx}]
mkshape –h n [–{lw}]
```

where α is the “corner” frequency, β is the excess bandwidth (roll-off factor), and n is the length of the impulse response, in samples.

Although `–l` is an optional parameter, the program doesn’t do anything useful if `–l` is not specified. With `–l`, the output is in the format described in section 2.2 above.

Exactly one of `–c`, `–r`, `–i` or `–h` must be specified. `–c` specifies a raised-cosine response. `–r` specifies a square-root response, i.e. the magnitude of the response at any frequency is proportional to the square root of the response you would have got if you had specified `–c` instead of `–r`. Square-root raised-cosine filters are often used in digital communication systems. `–i` specifies the identity response (constant “1” at all frequencies). `–h` specifies a Hilbert transformer. For `–h` and `–i`, the only parameter required is n, the length of the impulse response.

The optional parameter `–x` (not available for Hilbert transformers) specifies that $x / \sin x$ compensation is to be applied in the frequency domain, to compensate for the $\sin x / x$ response of real-world DACs (which output a sequence of square pulses when, ideally, they should be outputting a sequence of delta functions).

The optional parameter `–w` applies a Hamming window to the impulse response. This can significantly improve (reduce the amplitude of) the sidelobes, at the cost of some distortion in the passband response shape for short filters.

The corner frequency is defined for raised-cosine filters as the frequency at which the response is –6 dB relative to the response at 0 Hz if `-c` is specified, or –3 dB if `-r` is specified.

For further information on raised-cosine filters, see

    http://www-users.cs.york.ac.uk/~fisher/mkfilter/rcdoc/rcdoc.ps.gz

4. Gencode and genplot

The program `gencode` takes the `-l` output from `mkfilter` or `mkshape` and generates a piece of C or C++ code which implements the filter. With the exception of code generated by `-f` or `-xyc` (see below), the code is meant primarily to be read, not executed; however it is syntactically correct and complete except for input and output code, which you will have to supply.

`gencode` and `genplot` do not work with oscillators designed by `mkfilter` (i.e. band-pass resonators with infinite *Q*), because the "gain" of an oscillator is infinite.

The usage is:

`gencode` [`-ansic` | `-xyc` | `-f`]

The parameters control the format of the output. The default is `-ansic`, which specifies Ansi "C". `-xyc` causes just a single line to be output, containing the following data separated by tab characters: the pass-band gain g (see section 2.2); the Z+1 x coefficients; the P+1 y coefficients.

`-f` generates code required by Fisher's experimental "Filter-Filter" program. This option is intended for internal use only.

The program `genplot` takes the `-l` output from `mkfilter` or `mkshape` and generates a "gif" file containing a graph of either the phase and magnitude (frequency-domain) response, or the impulse (time-domain) response. The usage is

`genplot` [`-i` *n* | `-s` *n* | `-a` α_1 α_2] [`-log` *min*] [`-d`] *fn.gif*

fn.gif is the name of the output "gif" file. `-i` selects an impulse response graph and `-s` selects a unit-step response graph; *n* is the number of samples along the time axis. If neither `-i` nor `-s` is given then a frequency-domain graph is produced, in which case the optional parameters α_1 and α_2 are lower and upper limits on the range of frequency values to be plotted, expressed as a fraction of the sampling rate. The default values are $\alpha_1 = 0$ and $\alpha_2 = 0.5$, i.e. the response is plotted from zero frequency up to the Nyquist frequency. If the `-log` option is specified, the magnitude scale is logarithmic and labelled in dB from *min* to zero. (*min* must be negative.) If the `-log` option is omitted, the magnitude scale is linear from 0.0 to 1.0.

The `-d` option modifies the phase part of the frequency-response graph. Normally, the phase decreases monotonically wth frequency, because all causal filters have a strictly positive overall signal (group) delay. If `-d` is specified, `genplot` tries to guess the group delay and plots the phase relative to this figure. This is useful in the case of a linear-phase finite-impulse-response filter (e.g. raised-cosine or Hilbert transformer), in which case the delay is half the number of zeros, and likely to be confusing for other filter types.

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

5. Examples

`mkfilter -Bu -Lp -o 4 -a 0.2`

Generate a 4-pole Butterworth lowpass 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

`mkfilter -Re 1000 -Bp -a 0.3`

Generate a bandpass resonator with Q = 1000 and centre frequency 0.3 f_s; display pole & zero positions and filter recurrence relation

6. Contact

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

fisher@minster.york.ac.uk
http://www-users.cs.york.ac.uk/~fisher

13 Dec 1999