Digital Signal Processing

Exstrom Labs Home

Free Software

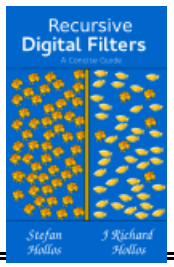
All the software on this page is free and distributed under the terms of the GNU General Public License. It is written in ANSI C and should compile with any C compiler.

Contact: Stefan or Richard

Digital signal generation software can be found here.

Please consider supporting further development of this software by buying our book:

Recursive Digital Filters: A Concise Guide



The book provides a quick introduction to recursive digital filters. The goal is to get the reader to the point where he can understand and use these filters as quickly as possible. The amount of mathematical background material is kept to a minimum and many examples are included.

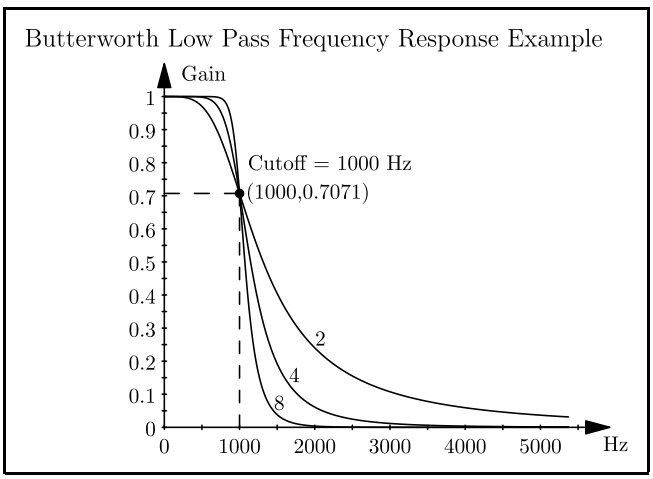
More information ...

Butterworth and Chebyshev Filters

The following are command line filter programs. They read data from stdin and write to stdout. Both input and output streams are in ascii format.

- bwlpf.c - 2, 4, 6,... order lowpass filter.

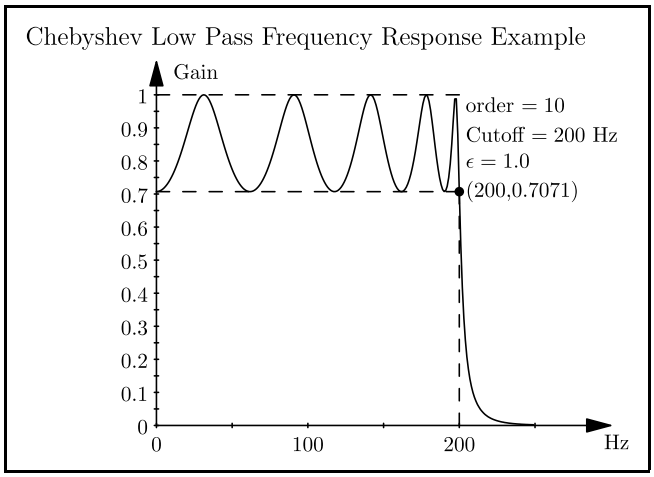
```
Usage: bwlpf n s f
Butterworth lowpass filter.
  n = filter order 2,4,6,...
  s = sampling frequency
  f = half power frequency
```



- cheblpf.c - 2, 4, 6,... order lowpass filter.

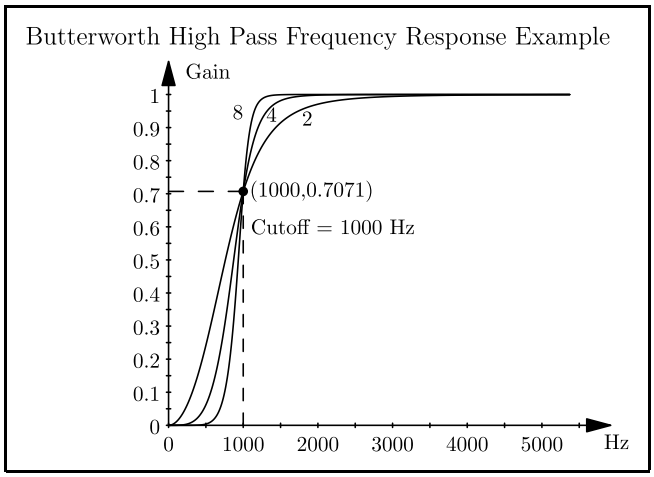
```
Usage: cheblpf n e s f
Chebyshev lowpass filter.
  n = filter order 2,4,6,...
```

```
e = epsilon [0,1]
s = sampling frequency
f = cutoff frequency
```



- bwhpf.c - 2, 4, 6,... order highpass filter.

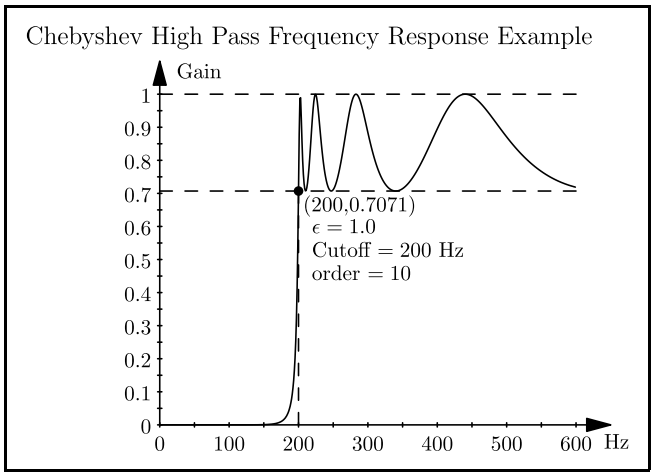
```
Usage: bwhpf n s f
Butterworth highpass filter.
  n = filter order 2,4,6,...
  s = sampling frequency
  f = half power frequency
```



- chebhpf.c - 2, 4, 6,... order highpass filter.

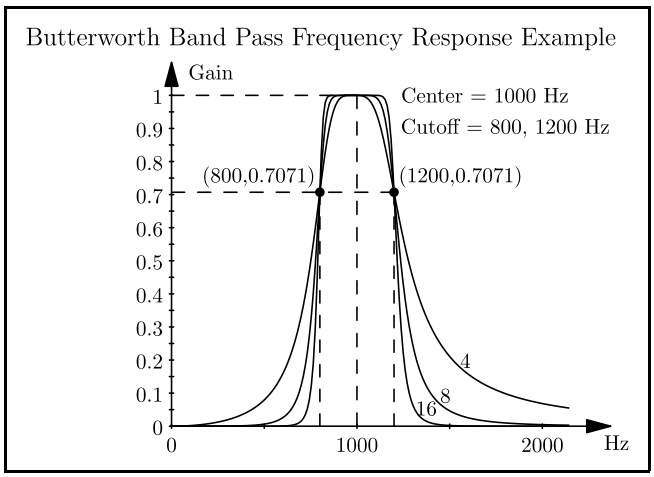
```
Usage: chebhpf n e s f
Chebyshev highpass filter.
  n = filter order 2,4,6,...
  e = epsilon [0,1]
  s = sampling frequency
  f = cutoff frequency
```

Coffee with Fourier?



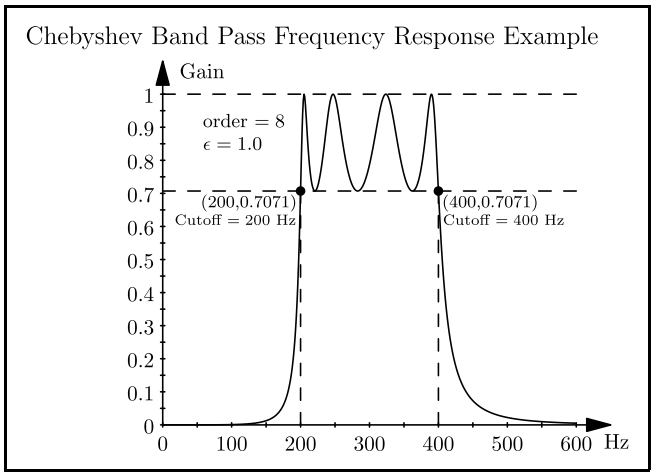
- bwbpf.c - 4, 8, 12,... order bandpass filter.

```
Usage: bwbpf n s f1 f2
Butterworth bandpass filter.
  n = filter order 4,8,12,...
  s = sampling frequency
  f1 = upper half power frequency
  f2 = lower half power frequency
```



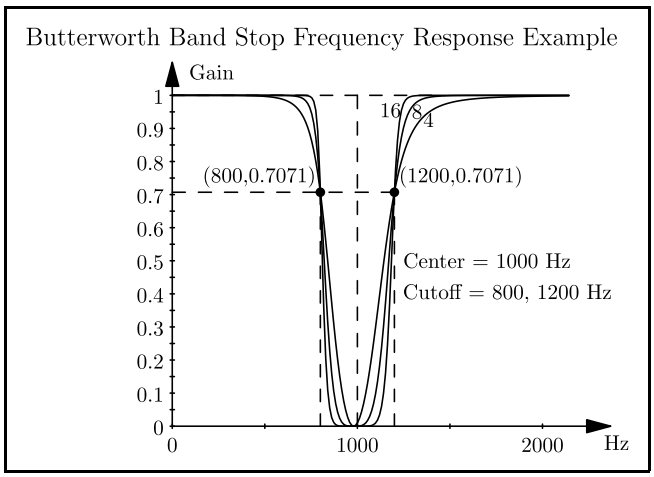
- chebbpf.c - 4, 8, 12,... order bandpass filter.

```
Usage: chebbpf n e s f1 f2
Chebyshev bandpass filter.
  n = filter order 4,8,12,...
  e = epsilon [0,1]
  s = sampling frequency
  f1 = upper half power frequency
  f2 = lower half power frequency
```



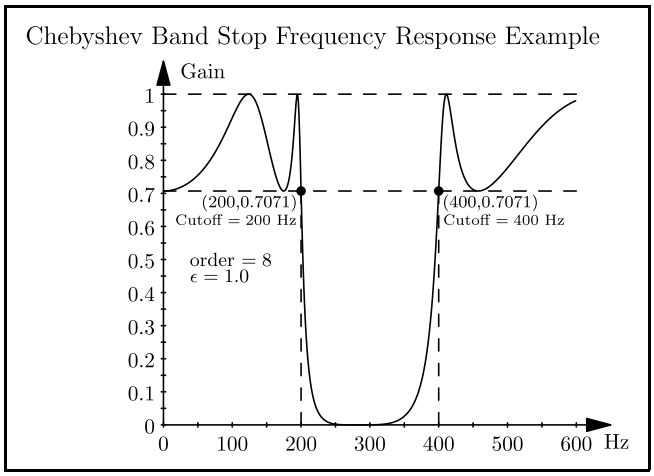
- bwbsf.c - 4, 8, 12,... order bandstop filter.

```
Usage: bwbsf n s f1 f2
Butterworth bandstop filter.
  n = filter order 4,8,12,...
  s = sampling frequency
  f1 = upper half power frequency
  f2 = lower half power frequency
```



- chebbsf.c - 4, 8, 12,... order bandstop filter.

```
Usage: chebbsf n e s f1 f2
Chebyshev bandstop filter.
  n = filter order 4,8,12,...
  e = epsilon [0,1]
  s = sampling frequency
  f1 = upper half power frequency
  f2 = lower half power frequency
```



Butterworth Filter Coefficients

The following files are for a library of functions to calculate Butterworth filter coefficients. There are functions for lowpass, bandpass, highpass, and bandstop filters. If you just want an efficient implementation of these filters then see the programs listed above.

- liir.c - source code
- iir.h - header file
- iir.txt - documentation

The following are example programs that use liir.c.

- bwlp.c - lowpass filter coefficient calculator.
- bwhp.c - highpass filter coefficient calculator.
- bwbp.c - bandpass filter coefficient calculator.
- bwbs.c - bandstop filter coefficient calculator.
- Makefile - creates libiir.a and compiles example programs.

The example programs take the order of the filter and the cutoff frequencies as command line input parameters. Run the programs with no parameters to get a usage message showing how to list the parameters. The output is a file that lists the `c` and `d` coefficients of the filter (see the documentation in iir.txt for an explaination of what this means)

Calculating cutoff frequencies: The cutoff frequencies are specified as a fraction of π. If f_s is the frequency at which the data is sampled (samples per second) and `f` is the cutoff frequency you want, then the parameter you would input to the program is $2*f/f_s$. For example if your data is sampled at 5000 samples/sec and you want a bandpass filter from 500 to 1000 Hz the lower cutoff would be 2*500/5000=0.2 and the upper cutoff would be 2*1000/5000=0.4. Here's a calculator that will calculate this for you.

Samples/second: 300000 Cutoff freq. (Hz): 55000

Calculate Digital cutoff: 0.36666667

Other recursive filter programs:
rffr.c - Calculates the frequency response of a recursive filter using the coefficient file created by one of the above programs.
rdf.c - Filters data from an input file, using a coefficient file generated by one of the above programs.

FIR Nonrecursive Digital Filters

A general nonrecursive filter is implemented as follows:

$y[n] = c_0*x[n] + c_1*x[n-1] + ... + c_M*x[n-M]$

The n^{th} output is a linear function of the n^{th} input and the previous M inputs. The c_k coefficients are calculated to give the filter a specific frequency response.

The three programs listed below calculate the c_k coefficients for implementing a lowpass, bandpass, and highpass filter. The filters are simple (no smoothing) approximations to ideal filters, and have linear phase. The programs take command line input parameters. The parameters are the number of coefficients and the cutoff frequencies. The more coefficients, the sharper the cutoff. The cutoff frequencies are specified as a fraction of Pi. See the discussion under "Recursive Digital Filters" for how to calculate the cutoff frequencies. An additional parameter is the number of zeros to pad on to the end of the coefficient list. This is useful if you want to take the FFT of the coefficients in order to apply the filter in the frequency domain, or if you just want to see what the frequency response of the filter is. You can set this parameter to zero. Running the programs with no parameters will print out a usage message showing the correct order in which the parameters should be given.

tdlpf.c - Calculates lowpass filter coefficients.
tdbpf.c - Calculates bandpass filter coefficients.
tdhpf.c - Calculates highpass filter coefficients.

Spectral Analysis

goertzel.c - A program for doing spectrum calculations using the Goertzel algorithm. It allows spectrum calculation at arbitrary points, whereas the FFT calculates the spectrum at evenly spaced points.

fft.c - FFT program. Takes the FFT of a data (time domain) file, and outputs to file the complex FFT data. The input file needs to be a text (ascii) file. All lines at the top of the input file that start with # are ignored. One data point per line is assumed.

extract.c - Extracts the real or imaginary parts or the magnitude or the phase of an FFT file that was created by the program fft.

Articles and Notes

A Simple Recursive Digital Filter (pdf, 7 pages, 132 KB, Oct 3 2006)
Calculating the Frequency Spectrum of a Signal (pdf, 10 pages, 157 KB)
Fourier Transform of a Sampled Signal (pdf, 5 pages, 93 KB)
Spectrum Magnification: When An FFT Is Not Enough (pdf, 9 pages, 147 KB)
Fourier Series Expansion of Functions in Two or More Dimensions (pdf, 3 pages, 79 KB)
Chebyshev Polynomials (pdf, 6 pages, 154 KB)