OregonDSP: A Basic Collection of FIR and IIR Digital Filters

Author: David B. Harris
Deschutes Signal Processing LLC

OregonDSP is a collection of 44 Java classes that design and implement finite impulse response (FIR) and infinite impulse response (IIR) digital filters. Supporting classes implement complex and real fast Fourier transforms (FFTs) and the overlap-add algorithm for filtering a continuous stream in a series of consecutive, contiguous blocks with an FIR filter.

Documentation at this point is limited to this brief README file and a set of Java docs included with the distribution. Many of the Java docs have extensive comments, including references to papers describing the algorithms.

The filters and FFTs implemented by this collection of classes generally replace (and frequently improve upon) Fortran and C filter implementations found in SAC (Seismic Analysis Code). However, they are not translations of SAC subroutines, but represent completely new development with substantially different implementations. These new routines often are significantly faster than their SAC counterparts.

The root package is com.oregondsp.signalprocessing. This top-level package contains classes for windowing signals (Hamming and Hanning windows are supported at present) and a Sequence utility class that contains low-level methods used by the rest of the classes. The remaining classes are divided among the following subpackages:

com.oregondsp.signalprocessing.fft

This package contains two public classes (CDFT, RDFT) and three package private support classes (CDFTsr, CDFTsr8, CDFTsr16). The latter are not intended to be instantiated by the user. The FFT classes recursively implement a hierarchy of discrete Fourier transforms of decreasing size until reaching length-8 and length-16 transforms. These latter two transforms are hand-coded for efficiency. Throughout, the design goal was to use the constructors of these classes to precompute and “hardwire” indexing and cosine table calculations, so that these are not performed during evaluation of the DFTs. This philosophy leads to a time-efficient implementation that may produce large images for larger-size DFTs (of length 16384 and greater). There are provisions for sharing the DFT instances among multiple calculations (if, for example, the FFTs are called repeated to process data from a continuous stream), though care must be taken to do this in a thread-safe manner.

The real DFT class (RDFT) uses the well-established trick of computing a real length-N DFT with two length-N/2 real DFTs implemented with a length-N/2 complex DFT.

com.oregondsp.signalprocessing.filter

This package contains two sub-packages implementing IIR and FIR filters and three utility classes (LagrangePolynomial, Polynomial and Rational) useful in designing these filters.

com.oregondsp.signalprocessing.filter.iir

This package contains classes for implementing IIR digital filters and their analog prototypes. The digital filters are designed by transformation of the analog prototypes with an impulse-invariant transformation (see Oppenheim and Schafer, Digital Signal Processing). As of this writing, the package implements Butterworth filters and Chebyshev types I and II filters in lowpass, bandpass and highpass types. In addition, it implements a general Allpass filter class and a specialized class for Thiran allpass filters used to implement subsample interpolation (equivalently subsample delays).

com.oregondsp.signalprocessing.filter.fir

This package contains the OverlapAdd class, which implements the overlap-add algorithm for filtering a continuous stream with an FIR filter in consecutive, contiguous blocks. This algorithm allows filtering of an arbitrary-length sequence and is suitable for filtering real-time streams. The package also contains the Interpolator class, which uses OverlapAdd to implement arbitrary integer-rate interpolation of continuous stream data. Fixed-length sequences (such as cut event segments) also can be processed with these classes. This package also contains a ComplexAnalyticSignal class that can be used to generate complex analytic representations of real signals and envelopes of real signals.

This package also contains the following subpackage implementing equiripple FIR filters designed with the Parks-McClellan algorithm.

com.oregondsp.signalprocessing.filter.fir.equiripple

This package implements the Remez exchange algorithm to design a wide variety of FIR filters characterized by minimizing the maximum deviations from desired frequency responses. At present the package implements lowpass, bandpass and highpass filters, two types of differentiators and Hilbert transformers (even-length and odd-length) and a half-band lowpass filter specialized for interpolation of sequences by a factor of two. Overviews of these filter types and the design method can be obtained in the following references:

A Unified Approach to the Design of Optimum Linear Phase FIR Digital Filters, James H. McClellan and Thomas W. Parks (1973), IEEE Transactions on Circuit Theory, Vol. CT-20, No. 6, pp. 697-701.

FIR Digital Filter Design Techniques Using Weighted Chebyshev Approximation, Lawrence R. Rabiner, James H. McClellan and Thomas W. Parks (1975) Proceedings of the IEEE, Vol. 63, No. 4, pp. 595-610.

Despite the relative antiquity of these references, the Parks-McClellan algorithm is still one of the best design methods available for digital filters.

Comments on use

To get to this point you have already unzipped the distribution file OregonDSP.zip. You should find com and doc subdirectories, which contain the source/executable files and the documentation html files, respectively, as well as copies of the GNU GPL and LGPL licenses. To view the documentation, open the doc subdirectory and click on the file index.html. This should launch your browser and bring up an index page that you can navigate to see documentation for each class.

To use the software, you should put the com directory and all of its subdirectories into the resource directory that you use for Java projects (it could be the directory you point to in the Java's classpath). Alternatively, if you are using an IDE, you can refer to the software as a library using the supplied jar file. Add oregondsp.jar to your list of jar file resources with your IDE.

For bug reports, please use oregondsp@gmail.com.

Dave Harris
Maupin, Oregon
February 3, 2011