Principles and Practice of Clinical Electrophysiology of Vision

Editors

John R. Heckenlively, M.D.
Professor of Ophthalmology
Jules Stein Eye Institute
Los Angeles, California

Geoffrey B. Arden, M.D., Ph.D.
Professor of Ophthalmology and Neurophysiology
Institute of Ophthalmology
Moorfields Eye Hospital
London, England

Associate Editors

Emiko Adachi-Usami, M.D.
Professor of Ophthalmology
Chiba University School of Medicine
Chiba, Japan

G.F.A. Harding, Ph.D.
Professor of Neurosciences
Department of Vision Sciences
Aston University
Birmingham, England

Sven Erik Nilsson, M.D., Ph.D.
Professor of Ophthalmology
University of Linköping
Linköping, Sweden

Richard G. Weleber, M.D.
Professor of Ophthalmology
University of Oregon Health Science Center
Portland, Oregon

Mosby Year Book

St. Louis Baltimore Boston Chicago London Philadelphia Sydney Toronto

Mosby Year Book

Dedicated to Publishing Excellence

Sponsoring Editor: David K. Marshall
Assistant Director, Manuscript Services: Frances M. Perveiler
Production Project Coordinator: Karen E. Halm
Proofroom Manager: Barbara Kelly

A Year Book Medical Publishers imprint of Mosby-Year Book, Inc.

Mosby-Year Book, Inc.
11830 Westline Industrial Drive
St. Louis, MO 63146

1 2 3 4 5 6 7 8 9 0 CL CL MV 95 94 93 92 91

Library of Congress Cataloging-in-Publication Data
Principles and practice of visual electrophysiology / [edited by] John R. Heckenlively, Geoffrey B. Arden.
p. cm.
Includes bibliographical references.
Includes index.
ISBN 0-8151-4290-0
1. Electroretinography. 2. Electrooculography. 3. Visual evoked response. I. Heckenlively, John R. II. Arden, Geoffrey B. (Geoffrey Bernard)
[DNLM: 1. Electrooculography. 2. Electrophysiology. 3. Electroretinography. 4. Evoked Potentials, Visual. 5. Vision Disorders—physiopathology. WW 270 P957]
RE79.E4P75 1991 91-13378
617.7 1547—dc20 CIP
DNLM/DLC
for Library of Congress

26

A Digital Band-Pass Filter for Electrophysiology Recording Systems*

Kamiar Khani-Oskouee
Paul A. Sieving

This article provides the computer source code for a finite impulse response (FIR), band-pass (BP) filter of general utility for signal processing by electrophysiologists. The electroretinogram (ERG) contains responses from many retinal sources that contribute waves of different frequencies to the ERG. Some ERG components can be effectively isolated by frequency filtering, such as the oscillatory potentials, which are small waves of predominantly 100 to 150 Hz that lie on the ascending limb of the b-wave. For ERG recording systems it is useful to have filters to accentuate individual ERG components and thereby separate them from other responses on the basis of their predominant frequencies.

Other uses of digital filters include eliminating high-frequency noise *post hoc* after the recordings are complete or slowing down a signal channel to match the low-frequency function characteristic of very high impedence, ion-selective electrodes.

Despite the prevalence of digital electrophysiological recording systems, there are few published programs for digital filters of a general nature. Our program allows the user to change the low and high frequencies of the pass-band in software as required for each individual application. Additionally, the user specifies the number of points in the digitized file and the total sampling time. The program then calculates the required coefficients for the FIR-BP filter. Thus, the implimentation of this FIR-BP filter is not limited to specific BP frequencies, nor is it limited to a specific computer system.

Copies of the programs mentioned in this article (see Appendix at the end of this chapter) are available from the authors on IBM-PC–compatible, DOS-format 5 1/2-in. disks (send blank disk in preaddressed and postage-paid mailer). The authors hereby grant permission for personal use of these programs by individuals only and not for commercial gain. Commercial users may contact the authors.

COMPUTER PROGRAM FOR A NONRECURSIVE FINITE IMPULSE RESPONSE BAND-PASS FILTER

Program 1 (see Appendix) gives the source code for a FIR-BP filter that we wrote in Turbo Pascal (Borland International, Inc, Scotts Valley, Calif). The user specifies the low- and high-frequency half-amplitude values of the desired pass-band, the number of points in the data files, and the sampling duration. Program 1 then calculates the filter coefficients for these specifications. The filter uses these coefficients and convolves them with the digitized data file. The filter has zero phase shift in the pass-band.

Technically, the coefficients are derived by taking the inverse discrete time Fourier transform of the BP window in the frequency domain. A Hamming window is applied to the coefficients to minimize ringing of the filter. End effects of filtering are minimized by extending the data file with zeros backward and forward in time beyond the first and last points. The pass-band of the filter becomes more square and has a sharper cutoff outside the pass-band as the number of coefficients is increased. For

further discussion of FIR filters, consult Oppenheim and Willsky[2] or Williams.[3]

Figure 26–1 demonstrates two FIR-BP filters operating on a sweep frequency file of 0 to 500 Hz. The test file contains low frequencies on the left and high frequencies on the right. Applying a 75- to 300-Hz BP-FIR filter from program 1 causes complete attenuation of low and high frequencies, with half-amplitude attenuation at 75 and 300 Hz. Within the pass-band, frequencies of 118 to 253 Hz are passed at unity gain (i.e., neither amplified nor reduced). Ninety percent pass-band amplitudes are 103 to 270 Hz; 60-Hz line noise is attenuated to 27%. This filter is useful in isolating the oscillatory potentials from the cone ERG response while also reducing high-frequency noise, as shown in Figure 26–2.

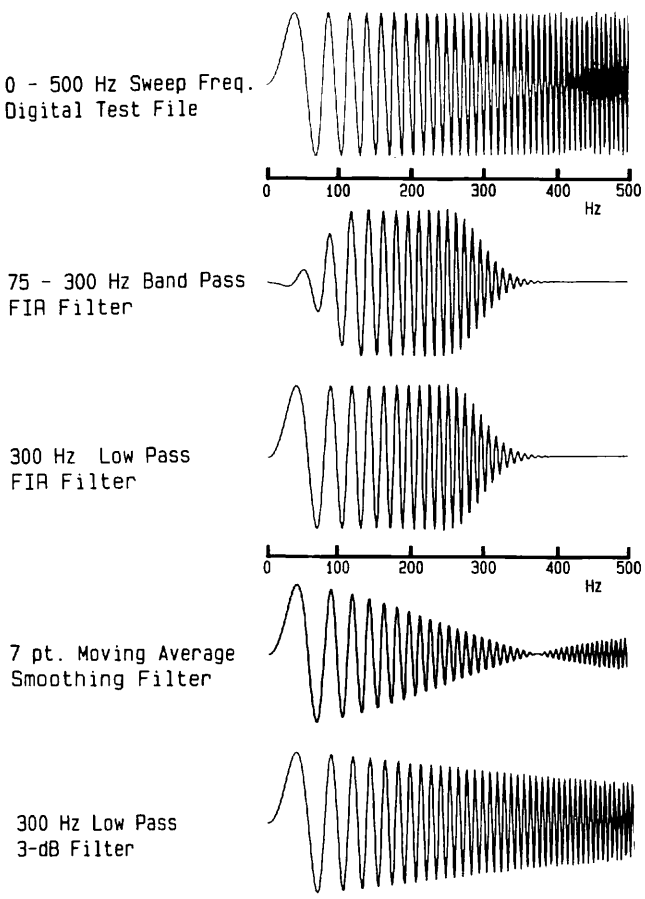

FIG 26–1.
0- to 500-Hz sweep frequency file. A data file was created by program 2 to serve as the standard test for checking our digital filters. The calibration bar shows the approximate frequency at positions along this trace. The higher frequencies appear noisy due to under representation by the 512 point file. **75 to 300-Hz BP-FIR filter:** created by program 1 using 50 coefficients with half-amplitude frequencies of 75 and 300 Hz. The filter has unity gain. **300 Hz low-pass FIR filter:** created by program 1 by specifying 0 Hz as the low-frequency and 300 Hz as the high-frequency half-amplitude values. Note that unity gain extends down to 0 Hz. **Seven-point moving average smoothing filter.** This attenuates *high* frequencies, but it has nonunity gain in the primary pass-band, and high frequencies leak through with 180-degree phase shift. **300 Hz low-pass 3-dB filter.** This filter attenuates to 70.7% (−3 dB) at 300 Hz and 10% (−21 dB) by 3000 Hz, *equivalent* to an analog one-pole RC filter.

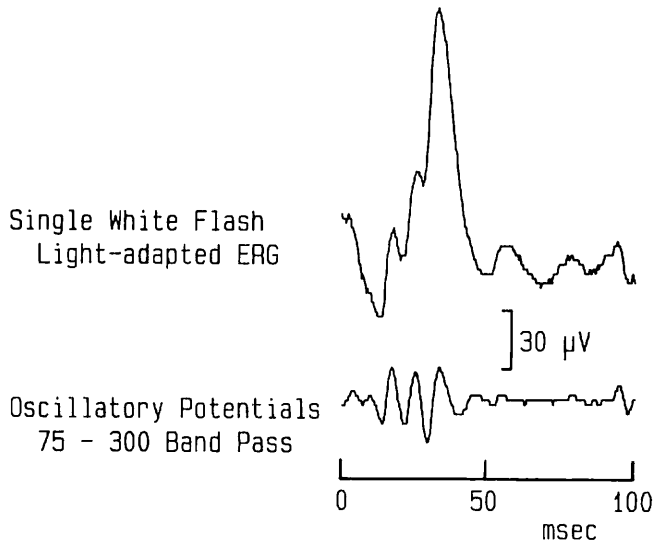

FIG 26–2.
Human light-adapted ERG response to a 10-μs single white flash of 0.73 cd-sec/m^2 per flash with 35 cd/m^2 photopic adapting background. The oscillatory potentials were obtained by filtering this ERG response with the 75- to 300-Hz FIR-BP filter shown in Figure 26–1. Note the clean isolation of the oscillatory potentials and the attenuation of high-frequency noise.

Figure 26–1 shows that by setting the low-frequency cutoff to 0 Hz, the FIR-BP filter becomes a low-pass filter with unity gain down to 0 Hz. Since higher frequencies are attenuated, this makes a good noise filter. By setting the high-frequency half-amplitude point quite low, this filter could be used to match two microelectrode channels that have electrodes with different time constants.

COMPARISON WITH A DIGITAL SMOOTHING FILTER

A smoothing filter based on a moving average technique is shown in Figure 26–1 for comparison. For a seven-point filter, three points to either side are averaged with the central data point, and the average becomes the data point in a new file (nonrecursive filter). Smoothing filters are inherently low

pass since high-frequency noise is eliminated. However, as seen in Figure 26–1, smoothing filters have poor low-pass characteristics. They have no region of unity gain. They also have considerable ripple, and consequently they allow unwanted high-frequency noise to pass through above the cutoff frequency (near 370 Hz in Fig 26–1). Above this first cutoff frequency, phase shifts by 180 degrees.

The sharpness and the frequency of the cutoff zone are mutually interdependent since the number of points in the smoothing routine affects both characteristics. The cutoff band can be made sharper, but only at the expense of making the half-amplitude frequency lower. This makes "designing" a smoothing filter quite arbitrary.

High-pass smoothing filters are created by subtracting the low-pass filtered data from the original file. Such filters generally exhibit ringing and amplify some frequencies while attenuating others in the pass-band. To our knowledge, as of the time of writing, at least one commercial manufacturer of ERG equipment used a high-pass smoothing filter to isolate the oscillatory potentials. Smoothing filters cannot be BP, and the high frequencies must be eliminated by analog filters prior to digitizing the signal.

COMPARISON WITH A 3-DB FILTER

Analog filters are often specified by the frequency of 3-dB attenuation (i.e., to 70.7% amplitude). Further, simple resistor-capacitor (RC) time-constant analog filters (i.e., one-pole filters) attenuate by 21 dB (to 10% amplitude) at one decade higher (at ten times the 3-dB frequency). The low-pass 3-dB filter in Figure 26–1 shows the effect on the sweep frequency test data. The high-frequency attenuation is more gradual than even the smoothing filter. Although this 3-dB filter is equivalent to a simple RC analog filter, it was implemented as a recursive digital filter as explained in Horowitz and Hill.[1] Most analog amplifiers have only one-pole, simple RC filters.

HINTS ON TESTING DIGITAL FILTERS

Because not all digital filters are identical, it is problematic to directly compare ERG data (e.g., oscillatory potentials) recorded by different manufacturers with different and usually unspecified digital filters. Consequently, results from different laboratories may differ in implicit times and amplitudes in the same way that different analog filters will affect ERG responses.

Just as with analog filters, it is wise for a user of commercial digital filters to know its frequency characteristics, gain uniformity in the frequency pass region, and the potential to introduce phase shifts. One convenient method to test a digital filter is with a sweep-frequency data file, as in Figure 26–1. Such test files can be made by recording the output of a sweep-frequency generator. A sweep-frequency test file can also be created mathematically by using the following equation, which is implemented in program 2, (see Appendix).

$$\text{Test-File(j)} = \sin\left[\frac{\pi \times F_{max} \times \text{Dur}}{(512 - 1)^2} \times (j-1)^2\right]$$

where j = 1 to 512 for a test file with 512 data points, F_{max} = the maximum frequency of sweep (i.e., 500 Hz in Figure 26–1), and Dur = the duration of the test file (i.e., 0.200 seconds for 512 points in Figure 26–1). The values of (j, F_{max}, and Dur) can be changed to meet different applications.

REFERENCES

1. Horowitz P, Hill W: *The Art of Electronics.* New York, Cambridge University Press, 1980, pp 446–447.
2. Oppenheim AV, Willsky AS: *Signals and Systems.* Englewood Cliffs, NJ, Prentice-Hall International, Inc, 1983.
3. Williams CS: *Designing Digital Filters.* Englewood Cliffs, NJ, Prentice-Hall International, Inc, 1986.

Appendix: Digital Band-Pass Filter Programs

PROGRAM 1

```
(* Program for a Finite Impulse Response (FIR) Band-Pass (BP) Digital Filter.*)
(* User specifies low- and high-frequency half-amplitude values,          *)
(*      and the duration and number of points in the data file.           *)
(* Authors: Kamiar Khani-Oskouee and Paul A. Sieving. 1989.               *)
(* Copyright by: Kamiar Khani-Oskouee and Paul A. Sieving. 1989.          *)
(* Permission is hereby given for personal use by individuals only and not *)
(*      for commercial gain. Commercial users may contact the authors.    *)
Program FIR_filter:
  const     Num_Data_Pts = 512;
  type      int_data_array = array [1..Num_Data_Pts] of integer;
            real_data_array = array [1..Num_Data_Pts] of real;
  var       Filt_Coeffs : real_data_array;
            Num_Filt_Coeffs : integer;
FUNCTION hamming_A_coefficient (m :integer): real;
   begin
     case m of
          0..7   : hamming_A_coefficient : = 1.124e-3 * m + 0.2100;
          7..10  : hamming_A_coefficient : = 1.136e-3 * m + 0.2209;
          10..18: hamming_A_coefficient : = 2.604e-4 * m + 0.2302;
          ELSE   hamming_A_coefficient : = 0.235;
       end;
   end;
FUNCTION hamming (a: real; i, imax: integer): real   ;
   var b : real;
   begin
     b : = 1.0 - 2.0 * a;
     IF i < = imax THEN hamming := 2*a*cos((π * i) / i_max) + b
        ELSE hamming : = 0;
   end;
FUNCTION impulse (i: integer; a, b: real) : real;
   begin
     IF i <> 0 THEN impulse : = (1.0/(π * i)) * (sin(b * i) - sin(a * i))
        ELSE impulse : = (1.0/π) * (b-a);
   end;
PROCEDURE calculate_filter_coefficients(var h:real_data_array; var m:integer);
   var Low_Cut, Hi_Cut, Duration : real;
       i : integer;
       fn, a, LC, HC : real;
   begin
     Low_Cut : = 75.0;    (* Hz *)     (* These are default filter values. *)
     Hi_Cut : = 300.0;    (* Hz *)     (* They can be changed below    *)
```

**Turbo Pascal 3.0, Borland Int., Inc.*

```
    Duration : = 0.200;  (* Sec *)     (*   by the user                *)
    m : = 50; (*# of coefficients*)     (*   by entering new values.    *)
    ClrScr;
    write ('Enter the lower cut-off frequency ::: '); readln (Low_Cut);
    write ('Enter the higher cut-off frequency ::: '); readln(Hi_Cut);
    write ('Enter the number of filter coefficients ::: '); readln(m);
    write ('Enter collection duration (seconds) ::: '); readln(Duration);
    fn : = ((Num_Data_Pts - 1) / Duration) / 2.0;
    IF Low_Cut = 0 THEN LC : = 0 ELSE LC : = abs((Low_Cut / fn) * pi);
    IF Hi_Cut > = fn THEN HC : = pi ELSE HC : = (Hi_Cut / fn) * pi;
    a : = hamming_A_coefficient(m);
    FOR i : = -m TO m DO h[m+i+1] : = impulse(i,lc,hc) * hamming(a,i,m);
  end;
PROCEDURE convolve(var data: int_data_array; N: integer;
                                h: real_data_array; m: integer);
      (* This Procedure enacts the filtering of an integer digital data file, *)
      (*  using the filter-coefficients calculated by a different procedure. *)
      (* It is called by "Filter_the_data" which reads and writes files, etc.*)
  var y : real_data_array;
      i, j, il     : integer;
  begin
    il : = data [1];
    FOR j : = 1 TO N DO data[j] : = data [j] - il;
    FOR j : = 1 TO N DO
      begin
        y[j]  : = 0;
        IF j  <= m THEN FOR i : = 1 TO j+m DO y[j] : = y[j] + data [i] * h[j+m+1]
        ELSE IF J > N - m THEN
          begin
            FOR i : =  0 TO (N-j+m) DO y[j] : = y[j] + data[j-m+i] * h[2*m+1-i];
            FOR i : = (N-j+m+1) TO 2*m DO y [j] : = y[j] + data[N] * h[2*m+1-i];
          end
        ELSE FOR i : = 0 TO 2*m DO y[j] : = y[j] + data[j-m+i] * h[2*m+1-i]
      end;
    FOR i : = 1 TO N DO data[i] : = round(y[i]);
  end;
PROCEDURE Filter_The_Data(h : real_data_array; m : integer);
    var i : integer;
        data : int_data_array;
        In_Data_File, Out_Data_File; string[80];
        File_Int : file of integer;
    begin
      write('Enter name of input file ::: '); readln(In_Data_File);
      write('Enter name of output file ::: '); readln(Out_Data_File);
      assign(File_Int, In_Data_File); reset(File_Int);
      FOR i : = 1 TO Num_Data_Pts DO read(File_Int,data[i]);
      close(File_Int);
      Convolve(data, Num_Data_Pts, h, m);
      assign(File_Int, Out_Data_File); rewrite(File_Int);
      FOR i : = 1 TO Num_Data_Pts DO write(File_Int,data[i]);
      close(File_Int);
    end;
```

```
BEGIN     (*This is the MAIN PROGRAM. It calls the procedures above.*)
  Calculate_Filter_Coefficients(Filt_Coeffs, Num_Filt_Coeffs);
  Filter_The_Data(Filt_Coeffs, Num_Filt_Coeffs);
END.
```

PROGRAM 2

```
(* Program to create a sweep-frequency data file to test digital filters.*)
(*      Frequencies are 0 - Fmax Hz.                                      *)
(* Authors:  Kamiar Khani-Oskouee and Paul A. Sieving. 1989.              *)
(*           University of Michigan, Ann Arbor, MI, USA.                 *)
Program Sweep Generation:
  const Num_Data_Pts = 512;  (* Default number of data points. *)
        Time = 0.200;        (* Default duration (sec.) of test file. *)
        Amplitude = 10000.0; (* Used to preserve significance of sin function. *)
        File Name = 'sweep'; (* Output file name. *)
var   r : file of integer;
      n, int : integer;
      Fmax, phase : real;
begin
  Fmax : = 500;                                    (* Max sweep frequency = 500 Hz (default value).*)
  Write('Enter maximum sweep-frequency ::: '); ReadIn(Fmax);
  phase : = (π * Fmax * Time) / sqr(Num_Data_Pts - 1.0);
  assign(r, file_name);
  rewrite(r);
  FOR n : = 0 to Num_Data_Pts DO
     begin
       int : = round(Amplitude * sin(phase * n * n));
       write(r,int);
     end;
  close(r);
end.
```