UNITED STATES DEPARTMENT OF THE INTERIOR
GEOLOGICAL SURVEY

FORTRAN COMPUTER PROGRAMS FOR RUNNING MEDIAN FILTERS AND A GENERAL DESPIKER

By

John R. Evans

Open-File Report
81-1091

This report is preliminary and has not been reviewed for conformity with Geological Survey editorial standards

INTRODUCTION

This report consists mainly of listings of the FORTRAN subroutines MEDFLT, an odd-length running median filter, and DESPIK, a general despiker and interpolator. The use and behavior of these time-domain digital filters are explained in Evans (1981); a brief description follows here. Punched card copies of the routines are available from the author at

345 Middlefield Road, MS-77
Menlo Park, CA 94Ø25.

Running Median Filters:

Running Median Filters (RMF's) are non-linear and cannot yet be described analytically. Their behavior is, however, predictable with the following main characteristics:

1. The "transfer behavior" for gaussian white noise is similar to that of running means (ie. convolution with a boxcar; Figure 1). Side-lobes and zeros are at the same frequencies, but RMF's have slightly higher side-lobes.
2. RMF's produce more "numerical noise" than their linear counterparts. Since a median is exactly equal to one of the numbers given it, there is no "smearing out" of small errors in those numbers.
3. RMF's are useful because they effectively destroy spikes (brief but large excursions of the data; Figure 2). An RMF with a window length of m points will destroy any spike of $(m-1)/2$ or fewer points. Linear filters change one-sided spikes into broad bumps ruining otherwise good data near the spike.
4. RMF's pass locally monotonic functions unchanged. An RMF with a window width of m points will not affect any signal which is monotonic on every m length segment. In particular, a square wave with no peak or trough narrower than $(m+1)/2$ will be unaffected.

One of the best direct applications of odd-length RMF's is filtering noisy square-wave time codes (Figure 3). Spikes are removed but the basic signal is only rarely affected, and then only where it is ambiguous.

The briefest segment of the square-wave code (2ØØ ms for WWVB radio time code) determines the longest RMF window that can be used without damaging the underlying signal:

$$m_{max} = 2(N/S - 3/2)$$

where m_{max} is odd, N=duration of shortest square-wave segment, and S=sampling interval of digitizer. For N=2ØØ ms and S=5 ms (2ØØ samples per second), m_{max}=77. Normally a much shorter window is used to save computation time.

Despiker:

If the numerical noise of an RMF is troublesome or if minimal impact on the time series is desired, the general despiker DESPIK can be used. It affects the trace only where spikes are found and is the better routine to apply to seismograms.

DESPIK can be fooled by data oscillating rapidly between good and bad values. If you tell DESPIK to detect one-point-long spikes and gives it a series such as "++++++/+/+/+/+//////" (+=good point, /=bad point) then"/+/+/+/+" will all be called bad and "//////" will be called good as long as all the latter points are all about the same value. If the value 99999, for example, identifies known bad points, this extra information can be given to DESPIK by using the statement beginning with "C:" in the source code. Otherwise you may find the routine interpolating from a section of good data, across a section of oscillating data, to a section of 99999's and producing ugly results.

General:

One of the annoying things about DESPIK and MEDFLT is that they require you to understand your data. You must choose a window width and (in DESPIK) a minimum spike height. To make such choices, you must know what sort of good data to expect and what sort of spiking behavior; they should not overlap. You can effectively despike when the spikes are of much shorter duration than the data events. These routines cannot save bad data, but they can be very powerful tools for saving good data contaminated by a few spikes and sometimes can help badly spiked data.

DESPIK and MEDFLT are not the whole story in despiking--use all the information available to you to accomplish the task--but they are very powerful general tools. Velleman and Hoaglin (1978) give a set of slower but more general robust filtering routines. Those routines do odd- or even-length RMF's as well as some other specialized operations of the sort Tukey (1977) describes for handling array ends and such things. They are useful to people with scatter plots which need smoothing.

MEDFLT

The running median filter algorithm implemented by this routine is Ken Anderson's*, takes O(m) operations and is the fastest I know of. The most direct algorithm fully sorts values in the window each time it produces a new output point; it requires $O(m^2)$ operations.

CALL MEDFLT (ARRAY,LN,M)

-ARRAY(LN) is the integer or real data (see comments). Both an input and an output.

-M is the (odd) window length (3.LE.M.LE.101; the upper limit can be changed either direction easily).

*Now at Lincoln Labs.

```
      SUBROUTINE MEDFLT (ARRAY,LN,M)
C-----ODD-LENGTH, UNWEIGHTED RUNNING MEDIAN FILTER----------------------------
C-----NON-LINEAR -- REMOVES SPIKES OF WIDTH (M-1)/2 OR LESS; LEAVES
C     SHARP STEPS UNAFFECTED (W.R.T. TIME); OTHERWISE BEHAVES ABOUT LIKE
C     UNWEIGHTED RUNNING MEAN.
C-----EACH POINT IS REPLACED BY THE MEDIAN OF THE NEAREST M POINTS OF
C     THE ORRIGINAL SERIES.
C-----M MUST BE ODD FOR THIS ALGORITHM (3.LE.M.LE.1Ø1 IN THIS VERSION).
C-------------------------------------------------------------------------------
C     PARAMETER LIST:
C     "ARRAY" BOTH INPUT AND OUTPUT TIME SERIES
C     "LN" - DIMENSION OF "ARRAY"
C     "M" - WINDOW SIZE FOR RUNNING MEDIAN
C-------------------------------------------------------------------------------
C-----("ARRAY" TYPE CAN BE CHOSEN BY APPROPRIATE USE OF ONE OF THE
C     DECLARATION STATEMENTS AND APPROPRIATE IF STATEMENTS PRECEEDED
C     BY "C*".)
C-----USES KEN ANDERSON'S ALGORITHM FOR ORDER M EXECUTION TIME AS
C     ENCODED AND SLIGHTLY MODIFIED BY
C     JOHN R. EVANS
C     U. S. GEOLOGICAL SURVEY
C     345 MIDDLEFIELD ROAD, MS-77
C     MENLO PARK, CA  94Ø25
C-----Ø8/Ø1/79
C-------------------------------------------------------------------------------
      REAL SORT(1Ø1),S,ARRAY(LN)
C*    INTEGER SORT(1Ø1),S,ARRAY(LN)
C*    COMPLEX SORT(1Ø1),S,ARRAY(LN)
      INTEGER SUBS(1Ø1),IS
      WRITE (6,1) M,LN
    1 FORMAT (/," STARTING MEDFLT. WINDOW LENGTH=",I3,
     *".  ARRAY LENGTH=",I1Ø,".")
      M1=M-1
      N=(M-1)/2
      N1=N+1
      NS=LN-N
      IOLD=Ø
      INEW=M
C-----TEST FOR ERRORS---------------------------------------------------------
      IF (M.LT.3) GOTO 2
      IF (M.GT.LN.OR.M.GT.1Ø1) GOTO 4
      IF ((2*N+1).NE.M) GOTO 6
      GOTO 1Ø
    2 WRITE (6,3) M
    3 FORMAT (" WINDOW LENGTH TOO SMALL. M=",I5)
      GOTO 8
    4 WRITE (6,5) M
    5 FORMAT (" WINDOW LENGTH TOO GREAT. M=",I5)
      GOTO 8
    6 WRITE (6,7) M
    7 FORMAT (" WINDOW MUST BE ODD LENGTH. M=",I5)
    8 WRITE (6,9)
    9 FORMAT (" -----ERROR IN SUBROUTINE MEDFLT-----",/,
     *"      (TIME SEARIES RETURNED UNFILTERED.)")
      GOTO 1ØØ
```

```
C-------------------------------------------------------------------------
C-----FILTER-----------------------------------------------------------------
C-------------------------------------------------------------------------
C     "IO" IS SUBSCRIPT OF "ARRAY"
C     "I" IS FOR LOCAL USE INCLUDING "OLDEST VALUE" SEARCH
C          OF "SORT" (AND "SUBS")
C     "IOLD" IS SUBSCRIPT OF OLDEST ELEMENT OF
C          "ARRAY" STILL HELD IN "SORT".  "SUBS"
C          IS SEARCHED FOR IT.
C     "INEW" IS SUBSCRIPT OF "ARRAY" FOR ELEMENT TO BE
C          ADDED TO "SORT" AND "SUBS"
C-------------------------------------------------------------------------
C-----FILL SORTING ARRAY FOR FIRST TIME AND BUBBLE SORT (WITH INCREASING SIZE
C     TO RIGHT)
   1Ø IO=N1
      DO 2Ø I=1,M
      SUBS(I)=I
   2Ø SORT(I)=ARRAY(I)
   3Ø KEY=Ø
      DO 4Ø I=1,M1
      IF (SORT(I)-SORT(I+1)) 4Ø,4Ø,35
C*    IF (REAL(SORT(I))-REAL(SORT(I+1))) 4Ø,4Ø,35
C*    ANY OTHER COMPARISON OF COMPLEX NUMBERS APPROPRIATE TO THE
C*    SPECIFIC APPLICATION
   35 S=SORT(I)
      SORT(I)=SORT(I+1)
      SORT(I+1)=S
      IS=SUBS(I)
      SUBS(I)=SUBS(I+1)
      SUBS(I+1)=IS
      KEY=1
   4Ø CONTINUE
      IF (KEY) 1Ø1Ø,5Ø,3Ø
C-------------------------------------------------------------------------
C     PUT RESULT (MIDDLE POINT OF "SORT") INTO INPUT "ARRAY" (NOTE THAT
C     NO OVERWRITE OF INPUT OCCURS BECAUSE INPUT IS SAVED IN "SORT" AS
C     LONG AS IT IS NEEDED)
C-------------------------------------------------------------------------
   5Ø ARRAY(IO)=SORT(N1)
C-----STEP SUBSCRIPTS-------------------------------------------------------
      IO=IO+1
      IF (IO.GT.NS) GOTO 1ØØ
      INEW=INEW+1
      IOLD=IOLD+1
C-----FIND OLD SUBSCRIPT----------------------------------------------------
      I=Ø
   6Ø I=I+1
      IF (SUBS(I)-IOLD) 1Ø2Ø,65,6Ø
C-----DECIDE WHICH WAY TO MOVE NEW ENTRY-------------------------------------
   65 S=ARRAY(INEW)
      IF (I-1) 1Ø3Ø,7Ø,67
   67 IF (SORT(I-1)-S) 7Ø,8Ø,77
C* 67 IF (REAL(SORT(I-1))-REAL(S)) 7Ø,8Ø,77
C*    OR ANY OTHER APPROPRIATE COMPARISON OF COMPLEX #'S
```

```
C-----MOVE NEW ENTRY UP UNTIL IT FITS-----------------------------------------
   7Ø IF (I-M) 72,8Ø,1Ø4Ø
   72 IF (SORT(I+1)-S) 74,8Ø,8Ø
C* 72 IF (REAL(SORT(I+1))-REAL(S)) 74,8Ø,8Ø
C*    OR ANY OTHER APPROPRIATE COMPARISON OF COMPLEX #'S
   74 SORT(I)=SORT(I+1)
      SUBS(I)=SUBS(I+1)
      I=I+1
      GOTO 7Ø
C-----MOVE NEW ENTRY DOWN UNTIL IT FITS---------------------------------------
   75 IF (SORT(I-1)-S) 8Ø,8Ø,77
C* 75 IF (REAL(SORT(I-1))-REAL(S)) 8Ø,8Ø,77
C*    OR ANY OTHER APPROPRIATE COMPARISON OF COMPLEX #'S
   77 SORT(I)=SORT(I-1)
      SUBS(I)=SUBS(I-1)
      I=I-1
      IF (I-1) 1Ø5Ø,8Ø,75
C-----DROP NEW ENTRY INTO ITS CORRECT PLACE-----------------------------------
   8Ø SORT(I)=S
      SUBS(I)=INEW
      GOTO 5Ø
C-----END OF FILTERING--------------------------------------------------------
  1ØØ WRITE (6,11Ø)
  11Ø FORMAT (" RETURN FROM MEDFLT.",/)
      RETURN
C-----UNUSUAL ERROR RETURNS---------------------------------------------------
 1Ø1Ø LINE=84
      GOTO 1Ø6Ø
 1Ø2Ø LINE=97
      GOTO 1Ø6Ø
 1Ø3Ø LINE=1ØØ
      GOTO 1Ø6Ø
 1Ø4Ø LINE=1Ø5
      GOTO 1Ø6Ø
 1Ø5Ø LINE=12Ø
 1Ø6Ø WRITE (6,1Ø7Ø) LINE
 1Ø7Ø FORMAT (//," ------UNUSUAL ERROR RETURN------",/,"     FROM LINE",
     *I3," OF MEDFLT",/," ----CONTACT J.R.EVANS----",//)
      STOP
      END
```

DESPIK

This routine employs MEDFLT to identify spikes in the data, by comparing abs(trace-RMF(trace)) to a threshold, and does linear or piecewise continuous cubic polynomial (PCCP) interpolations across identified spikes (Wiggins, 1976). It can be used to identify spikes and/or interpolate across them (Figure 4).

CALL DESPIK (IDATA,IWORK,SPKFLG,NPTS,MAXDUR,MINSIZ,INTERP)

IDATA(NPTS) is the integer input data (also used for output when INTERP.GT.Ø). The routine can handle real arays with minor changes (see comments).

IWORK(NPTS) is work space for the routine.

SPKFLG(NPTS) is a logical array which is true everywhere except where spikes are identified (an output if INTERP is non-negative; an input if INTERP.LT.Ø).

MAXDUR is the maximum duration (in number of points) of excursions of the data that are to be called "spikes". In the call to MEDFLT the window length M is 2*MAXDUR+1.

MINSIZ is the minimum height or depth of excursions relative to their immediate surroundings that are to be called "spikes". MAXDUR and MINSIZ define "spike"--choose them thoughtfully.

INTERP=Ø to identify but not touch spikes.
=1 or 2 to identify and do a linear or piecewise continuous cubic polynomial interpolation across spikes, respectively.
=-1 or -2 to do only the interpolation (segments to be interpolated are identified to the routine by making the same points in SPKFLG false).

```
      SUBROUTINE DESPIK (IDATA,IWORK,SPKFLG,NPTS,MAXDUR,MINSIZ,INTERP)
C-----"GENTLE" GENERAL DESPIKING ROUTINE.
C-----IF THE USEFUL DATA IN ARRAY "IDATA(NPTS)" IS OF SUBSTANTIALLY
C     LOWER FREQUENCY THAN THE SPIKES TO BE REMOVED, THIS ROUTINE
C     CAN BE USED TO:
C          A)  IDENTIFY SPIKES (INTERP=Ø)
C          B)  IDENTIFY AND INTERPOLATE ACROSS SPIKES (INTERP=1 OR 2)
C          C)  INTERPOLATE ACROSS SPIKES GIVEN "SPKFLG" AS AN INPUT (INTERP=-1
C              OR -2).  IN OTHER WORDS, THE USER HAS IDENTIFIED THE SPIKES
C              BY SOME MEANS ALREADY AND JUST WANTS AN INTERPOLATOR.
C-------------------------------------------------------------------------
C     IN ALL CASES THE LOGICAL ARRAY "SPKFLG(NPTS)" WILL BE TRUE EVERYWHERE
C     EXCEPT WHERE SPIKES ARE.  IN CASE A) "IDATA" IS UNAFFECTED; IN
C     CASE B) "IDATA" IS AFFECTED ONLY WHERE SPIKES WERE FOUND; IN
C     CASE C) "IDATA" IS AFFECTED WHEREVER "SPKFLG" IS .FALSE..
C-------------------------------------------------------------------------
C     INTERP=+1 OR -1 GIVES A LINEAR INTERPOLATION ACROSS SPIKES,
C     INTERP=+2 OR -2 GIVES A CUBIC INTERPOLATION ACROSS SPIKES, USING THE
C          PIECEWISE CONTINUOUS CUBIC POLYNOMIAL INTERPOLATION SCHEME OF
C          WIGGINS (1976).
C-------------------------------------------------------------------------
C     FILTERING DOES NOT OPERATE WITHIN "MAXDUR" POINTS OF EITHER END
C          OF THE DATA ARRAY.
C-------------------------------------------------------------------------
C     THE USER MUST KNOW THE CHARACERISTICS OF THEIR DATA AND SHOULD USE ANY
C     SPECIAL KNOWLEDGE OF IT TO IDENTIFY AND REMOVE SPIKES BEFORE APPLYING
C     THIS GENERAL DESPIKER:
C          A "SPIKE" IS ANY SEGMENT OF DATA NOT MORE THAN "MAXDUR" POINTS LONG
C          WHICH IS AT LEAST "MINSIZ" DIFFERENT IN VALUE FROM ITS NEAREST
C          VALUED NEIGHBORS.  "NEIGHBORS" ARE POINTS NO MORE THAN MAXDUR
C          POINTS FROM THE SPIKE.  FOR D=ANY DATA POINT, N=NEIGHBOR POINT,
C          E=END POINT, S=SPIKE POINT, AND MAXDUR=3 FOR EXAMPLE:
C
C          EEEDDDDDDDDDNNNSSSNNNDDDDDDDDDEEE
C          EEEDDDDDDDDDNNNSSNNNDDDDDDDDDEEE
C          EEEDDDDDDDDDNNNSNNNDDDDDDDDDEEE
C
C          ARE POSSIBLE SPIKE SEQUENCES FOR A SMALL SAMPLE DATA ARRAY.
C-------------------------------------------------------------------------
C-----USES RUNNING MEDIAN FILTER (RMF) AND COMPARES FILTERED AND UNFILTERED
C          ARRAYS TO DETERMINE WHICH DATA IS "SPIKED".
C-----"IWORK(NPTS)" IS A WORKING ARRAY WHICH WILL BE RETURNED WITH THE
C          ABSOLUTE DIFFERENCE BETWEEN "IDATA" AND "IDATA" FILTERED BY AN RMF
C          (WHEN INTERP.GE.Ø).
C-----"MAXDUR" MUST BE NON-NEGATIVE (ZERO WILL CAUSE IMMEDIATE RETURN
C          FROM THE ROUTINE WITHOUT ANY FILTERING).
C-----"MINSIZ" MUST BE POSITIVE.
C-----"NPTS" MUST BE GREATER THAN 2*MAXDUR
C-------------------------------------------------------------------------
C     JOHN R. EVANS,
C     U. S. GEOLOGICAL SURVEY
C     345 MIDDLEFIELD ROAD, MS-77
C     MENLO PARK, CA  94Ø25
C     Ø4/16/8Ø
C-------------------------------------------------------------------------
```

```
C-------------------------------------------------------------------------
C     THE SUBROUTINE STATEMENT AND STATEMENTS FOLLOWING "C;;;;;" COMMENTS CAN
C     BE CHANGED TO HANDLE REAL ARRAYS.
C-------------------------------------------------------------------------
C     THE STATEMENT COMMENTED OUT BY "C:" CAN BE USED TO TELL THE ROUTINE ABOUT
C     KNOWN BAD POINTS (PREVIOUSLY SET TO VALUE 99999).  THEY WILL BE
C     INTERPOLATED ACROSS TOO.
C-------------------------------------------------------------------------
C     USES SUBROUTINE MEDFLT
C-------------------------------------------------------------------------
C;;;;;
      INTEGER IDATA(NPTS),IWORK(NPTS),MINSIZ
      LOGICAL SPKFLG(NPTS)
C-----INITIALIZE ARRAYS AND CHECK INPUT PARAMETERS-----------------------
      IF (NPTS.LT.2*MAXDUR+1) GOTO 4ØØ
      IF (INTERP.LT.Ø) GOTO 22
      DO 1Ø I=1,NPTS
C;;;;;
      IWORK(I)=IDATA(I)
   1Ø SPKFLG(I)=.TRUE.
      IF (MAXDUR.LE.Ø) RETURN
      IF (MINSIZ.LE.Ø) GOTO 3ØØ
C-----FIND WINDOW LENGTH FOR RUNNING MEDIAN FILTER----------------------
      M=2*MAXDUR+1
C-------------------------------------------------------------------------
C-----DO RUNNING MEDIAN FILTER-----------------------------------------
C-------------------------------------------------------------------------
C;;;;;
      CALL MEDFLT (IWORK,NPTS,M)
C-------------------------------------------------------------------------
C-----FIND SPIKES-------------------------------------------------------
C-------------------------------------------------------------------------
      DO 2Ø I=1,NPTS
C;;;;;
      IWORK(I)=IABS(IDATA(I)-IWORK(I))
C:    IF (IDATA(I).EQ.99999) GOTO 15
C;;;;;
      IF (IWORK(I)-MINSIZ) 2Ø,15,15
   15 SPKFLG(I)=.FALSE.
   2Ø CONTINUE
C-----INTERPOLATE ACROSS SPIKES AS REQUESTED----------------------------
   22 IF (INTERP.EQ.Ø) RETURN
      IF (IABS(INTERP).EQ.2) GOTO 2ØØ
      IF (IABS(INTERP).GT.2) GOTO 1ØØ
C-------------------------------------------------------------------------
C-----LINEAR INTERPOLATION----------------------------------------------
C-------------------------------------------------------------------------
      I=1
   3Ø I=I+1
      IF (I-NPTS) 32,31,31
C-----EXIT--------------------------------------------------------------
   31 RETURN
   32 IF (SPKFLG(I)) GOTO 3Ø
      II=I
```

```
   35 I=I+1
      IF (.NOT.SPKFLG(I)) GOTO 35
      I=I-1
      III=I
C;;;;;
      D=FLOAT(IDATA(II-1))
C;;;;;
      Y=FLOAT(IDATA(III+1)-IDATA(II-1))
      X=FLOAT(III-II+2)
      DO 4Ø J=II,III
      DD=D+Y*FLOAT(J-II+1)/X
      IF (DD) 37,39,39
C;;;;;
   37 IDATA(J)=IFIX(DD-Ø.5)
      GOTO 4Ø
C;;;;;
   39 IDATA(J)=IFIX(DD+Ø.5)
   4Ø CONTINUE
      GOTO 3Ø
C-------------------------------------------------------------------------
C-----USER SUPPLIED INTERPOLATION---------------------------------------
C-------------------------------------------------------------------------
C     USER CAN ADD A SUBROUTINE OPERATING ACROSS AREAS WHERE "SPKFLG" IS .FALSE.
C     (ONLY ONE "GOOD" POINT ON EITHER SIDE OF A SPIKE IS ASSURED, BUT MORE WILL
C     USUALLY BE FOUND BY SCANNING "SPKFLG"--TYPICALLY MAXDUR+1 OR MORE.  SOME
C     RARE CIRCUMSTANCES EXIST IN WHICH "GOOD" POINTS REALLY AREN'T--USE ANY
C     SPECIAL KNOWLEDGE TO WEED THEM OUT BEFORE INTERPOLATING.)
C-------------------------------------------------------------------------
  1ØØ WRITE (6,11Ø)
  11Ø FORMAT (/," ONLY LINEAR AND CUBIC INTERPOLATIONS AVAILABLE NOW
     *(ROUTINE DESPIK)",/," ------DOING CUBIC INTERPOLATION------",/)
C-------------------------------------------------------------------------
C-----PIECEWISE CONTINUOUS CUBIC POLYNOMIAL INTERPOLATION----------------
C-------------------------------------------------------------------------
C     BASED ON AVERAGE SLOPE INTERPOLATION OF WIGGINS (1976):
C          FIND SPIKES AND TWO GOOD POINTS ON EITHER SIDE (IF ONLY ONE GOOD
C          POINT IS FOUND BEFORE END OF ARRAY IS ENCOUNTERED, USE A FAKE POINT
C          OF THE SAME VALUE AS THE LAST GOOD ONE).
C-------------------------------------------------------------------------
  2ØØ I=1
  21Ø I=I+1
      IF (I-NPTS) 22Ø,215,215
C-----EXIT--------------------------------------------------------------
  215 RETURN
  22Ø IF (SPKFLG(I)) GOTO 21Ø
      II=I
  23Ø I=I+1
      IF (.NOT.SPKFLG(I)) GOTO 23Ø
      I=I-1
      III=I
C-----FIND SURROUNDING GOOD DATA POINTS---------------------------------
C;;;;;
      YB=FLOAT(IDATA(II-1))
      XB=FLOAT(II-1)
```

```
C;;;;;
      YC=FLOAT(IDATA(III+1))
      XC=FLOAT(III+1)
      JJ=II-1
  235 JJ=JJ-1
      IF (JJ) 24Ø,24Ø,245
  24Ø YA=YB
      XA=XB-1.
      GOTO 25Ø
  245 IF (.NOT.SPKFLG(JJ)) GOTO 235
C;;;;;
      YA=FLOAT(IDATA(JJ))
      XA=FLOAT(JJ)
  25Ø JJJ=III+1
  255 JJJ=JJJ+1
      IF (JJJ-NPTS) 265,265,26Ø
  26Ø YD=YC
      XD=XC+1.
      GOTO 27Ø
  265 IF (.NOT.SPKFLG(JJJ)) GOTO 255
C;;;;;
      YD=FLOAT(IDATA(JJJ))
      XD=FLOAT(JJJ)
C-----SUBTRACT XA FROM ALL X VALUES TO PREVENT ROUND-OFF ERRORS LATER----------
  27Ø XB=XB-XA
      XC=XC-XA
      XD=XD-XA
      XS=XA
      XA=Ø.
C-----SUBTRACT YA FROM ALL Y VALUES TO PREVENT ROUND-OFF ERRORS----------------
      YB=YB-YA
      YC=YC-YA
      YD=YD-YA
      YS=YA
      YA=Ø.
C-----CALCULATE SLOPES TO FIT CUBIC TO----------------------------------------
      R=(YB-YA)/(XB-XA)
      S=(YC-YB)/(XC-XB)
      T=(YD-YC)/(XD-XC)
      WR=1./AMAX1(R,Ø.ØØØØØ1)
      WS=1./AMAX1(S,Ø.ØØØØØ1)
      WT=1./AMAX1(T,Ø.ØØØØØ1)
      SB=(WR*R+WS*S)/(WR+WS)
      SC=(WS*S+WT*T)/(WS+WT)
C-----CALCULATE THE FOUR COEFFICIENTS OF THE CUBIC----------------------------
      XB2=XB*XB
      XB3=XB2*XB
      XB4=XB3*XB
      XC2=XC*XC
      XC3=XC2*XC
      XC4=XC3*XC
      DETA=XB4-4.*XC*XB3+6.*XC2*XB2-4.*XC3*XB+XC4
      DETC1=2.*(XC-XB)*(YB-YC)+(XB2-2.*XC*XB+XC2)*(SB+SC)
      DETC2=3.*(YB-YC)*(XB2-XC2)-SB*(XB3-3.*XC2*XB+2.*XC3)
     B      -SC*(2.*XB3-3.*XC*XB2+XC3)
```

```
      DETC3=6.*(YB-YC)*(XC2*XB-XC*XB2)+SB*(XC4-3.*XC2*XB2+2.*XC*XB3)
     B      +SC*(XB4-3.*XC2*XB2+2.*XC3*XB)
      DETC4=YB*(XC4-4.*XC3*XB+3.*XC2*XB2)
     B      +YC*(XB4-4.*XC*XB3+3.*XC2*XB2)
     C      +SB*(-XC4*XB+2.*XC3*XB2-XC2*XB3)
     D      +SC*(-XC*XB4+2.*XC2*XB3-XC3*XB2)
      C1=DETC1/DETA
      C2=DETC2/DETA
      C3=DETC3/DETA
      C4=DETC4/DETA
C-----FILL IN INTERPOLATED POINTS-----------------------------------------------
      DO 29Ø J=II,III
      XX=FLOAT(J)-XS
      DD= C1*XX*XX*XX + C2*XX*XX + C3*XX + C4 + YS
      IF (DD) 285,287,287
C;;;;;
  285 IDATA(J)=IFIX(DD-Ø.5)
      GOTO 29Ø
C;;;;;
  287 IDATA(J)=IFIX(DD+Ø.5)
  29Ø CONTINUE
      GOTO 21Ø
C-------------------------------------------------------------------------------
C-----WARNINGS AND ABNORMAL EXIT POINTS-----------------------------------------
C-------------------------------------------------------------------------------
  3ØØ WRITE (6,31Ø)
  31Ø FORMAT (/," SPIKE SIZE TOO SMALL IN ROUTINE DESPIK.",/,
     *" STOPPING RUN.",/)
      STOP
  4ØØ WRITE (6,41Ø) NPTS
  41Ø FORMAT (/," TOO FEW POINTS (",I7,") IN ROUTINE DESPIK.",/,
     *" NO FILTERING DONE.",/)
      RETURN
      END
```

REFERENCES

Evans, John R. (1981). Running median filters and a general despiker, Bull. Seis. Soc. Am., submitted.

Tuckey, John W. (1977). Exploratory data analysis, Addison-Wesley Publishing Company, Reading, Massachusetts, 688 p.

Velleman, Paul F. and David C. Hoaglin (1978). Implementation of some resistant non-linear smoothers, Economic and Social Statistics Technical Report Series, (878/Ø18), New York State School of Industrial and Labor Relations, Cornell University, 38 p.

Wiggins, Ralph A. (1976). Interpolation of digitized curves, Bull. Seis. Soc. Am., vol. 66, pp. 2Ø77-2Ø81.

FIGURE CAPTIONS

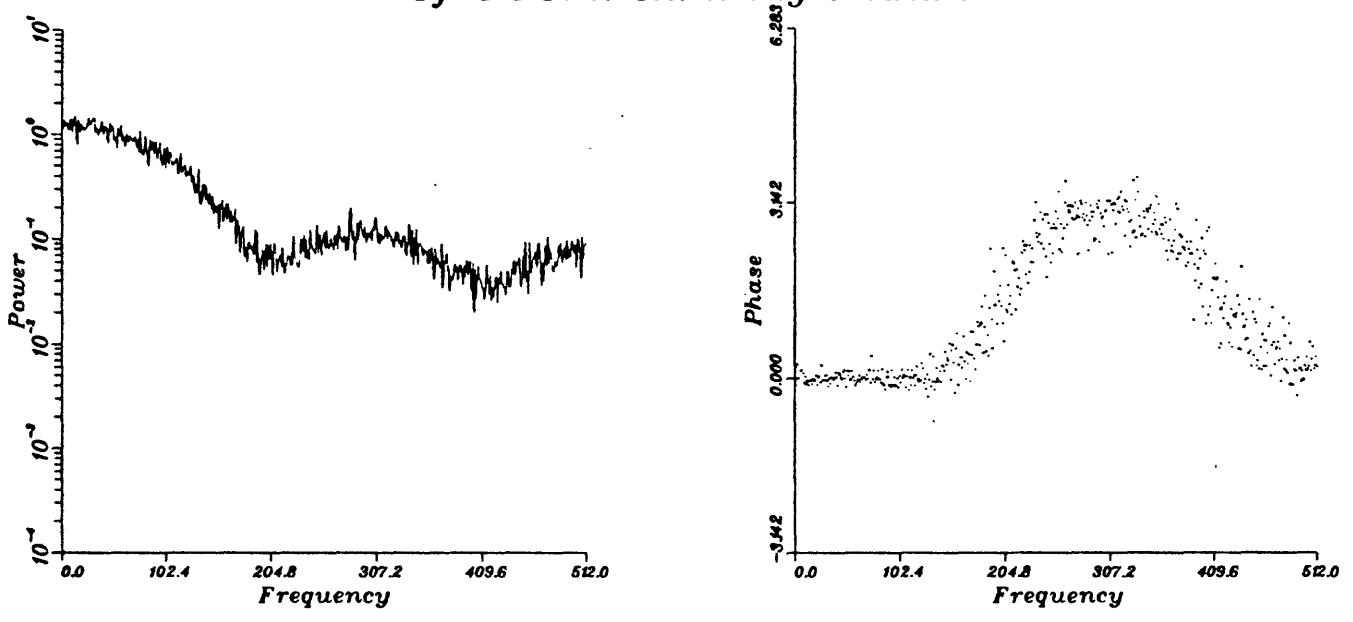
Figure 1. "Transfer behavior" of a running median filter compared to transfer function of a running mean (m=5) for a 1Ø24-point gaussian white noise sample. Each decade of "Power" is 1ØdB. These are ensemble averages of 15 estimates. Since the test data have no associated sample rate (they are just a series of points without reference to time), there are no units attached to frequency. Frequency 512 is nyquist; tick marks between nyquist and zero frequency are at 2Ø% intervals. Phase is in radians.

Figure 2. From top to bottom: 1Ø24-point noise sample; sample filtered by running median (m=7); sample filtered by 7-point running mean. Running median filter removes spikes much better than running mean.

Figure 3. Effect of running medians and running means on WWVB radio time code. Note undesirable slopes introduced into square wave by running means, and points "copied on" at both ends of sample where filters are undefined. Digitized at 2ØØ samples per second.

Figure 4. The effect of the general despiker DESPIK. From top to bottom: synthetic spikes, original seismogram, original plus spikes (input to filter), filtered seismogram, error in recovering original. Total spikes (integral of noise) are reduced by 92%; signal/noise is improved from 1.2 to 2.9.

White Noise Transfer Behavior of 5-Point Running Median



White Noise Transfer Behavior of 5-Point Running Mean

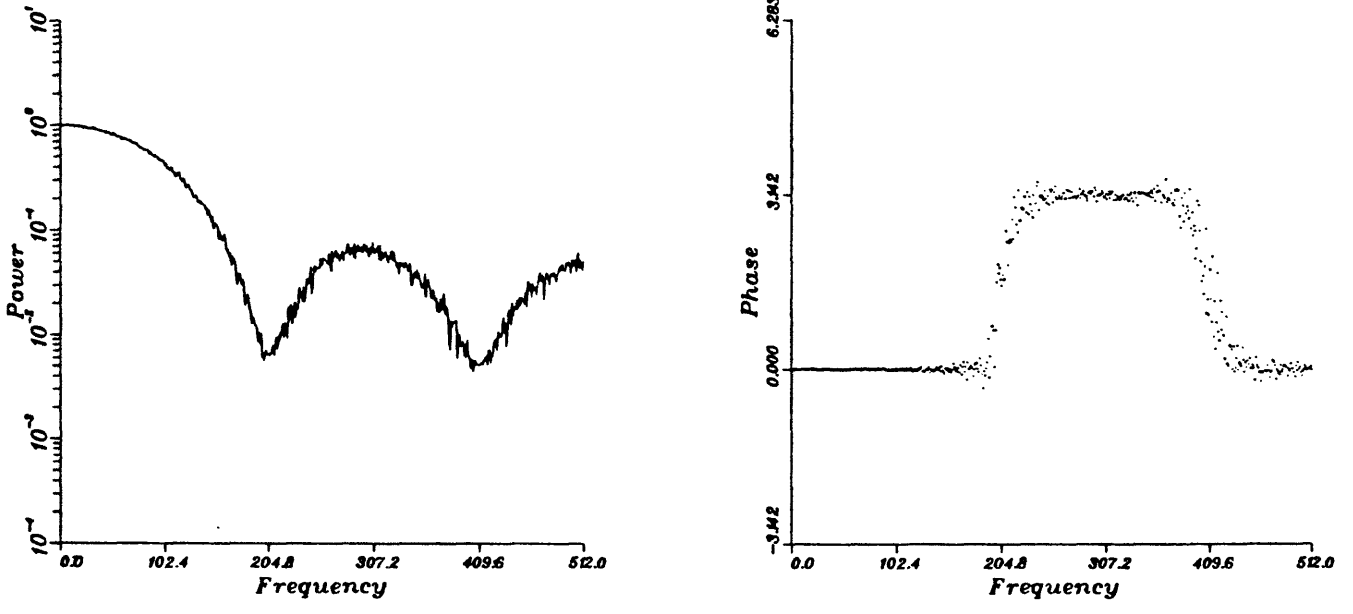


Figure 1

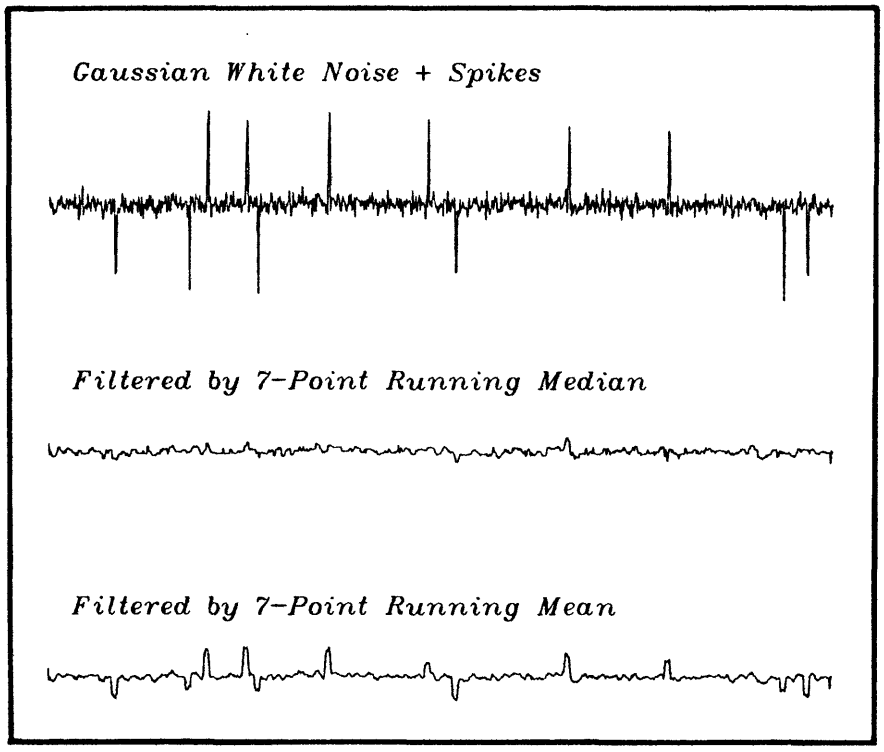

Gaussian White Noise + Spikes
Filtered by 7-Point Running Median
Filtered by 7-Point Running Mean

Figure 2

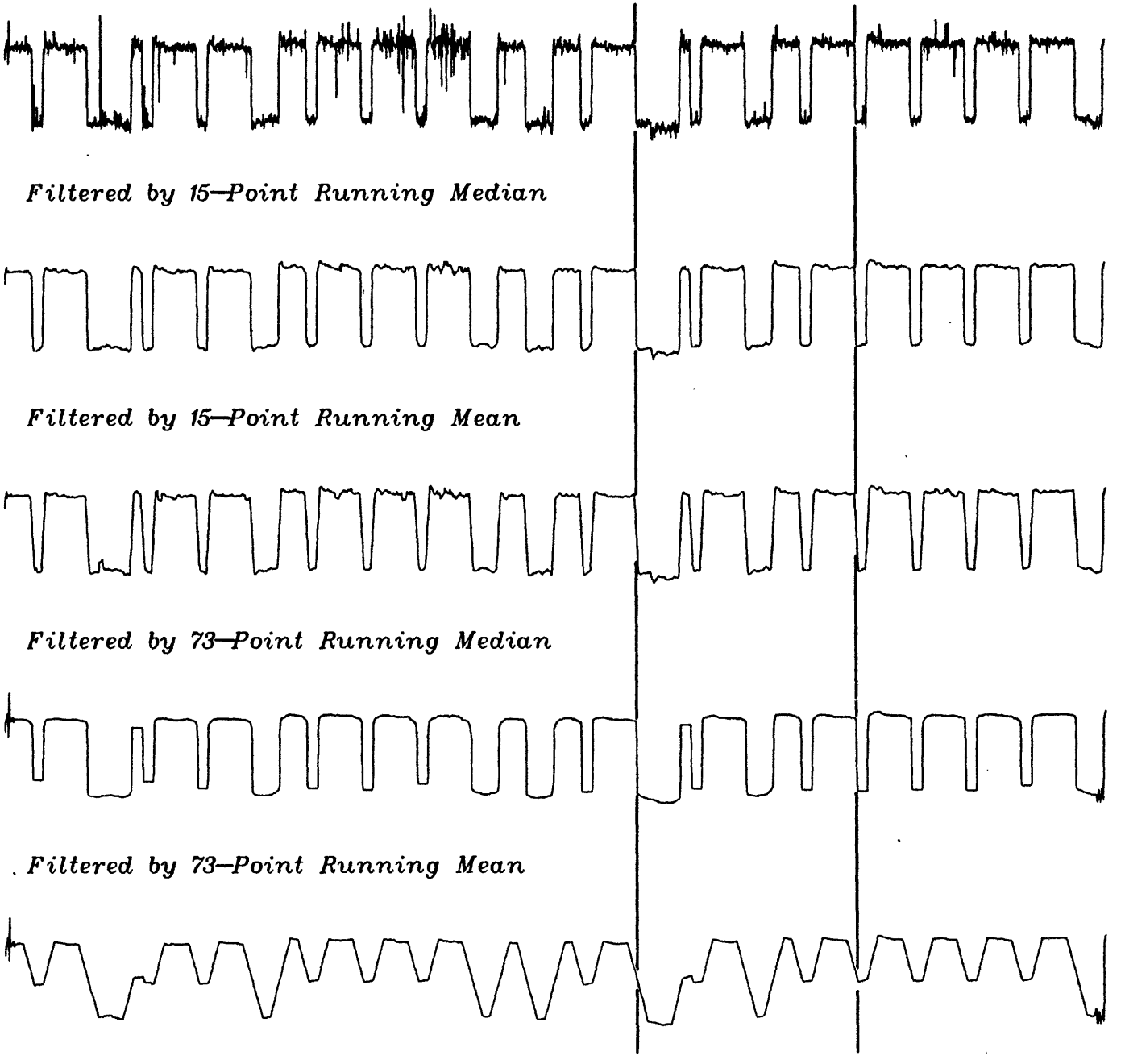

WWVB Time Code (200 Samples per Second)
Filtered by 15—Point Running Median
Filtered by 15—Point Running Mean
Filtered by 73—Point Running Median
Filtered by 73—Point Running Mean

Figure 3

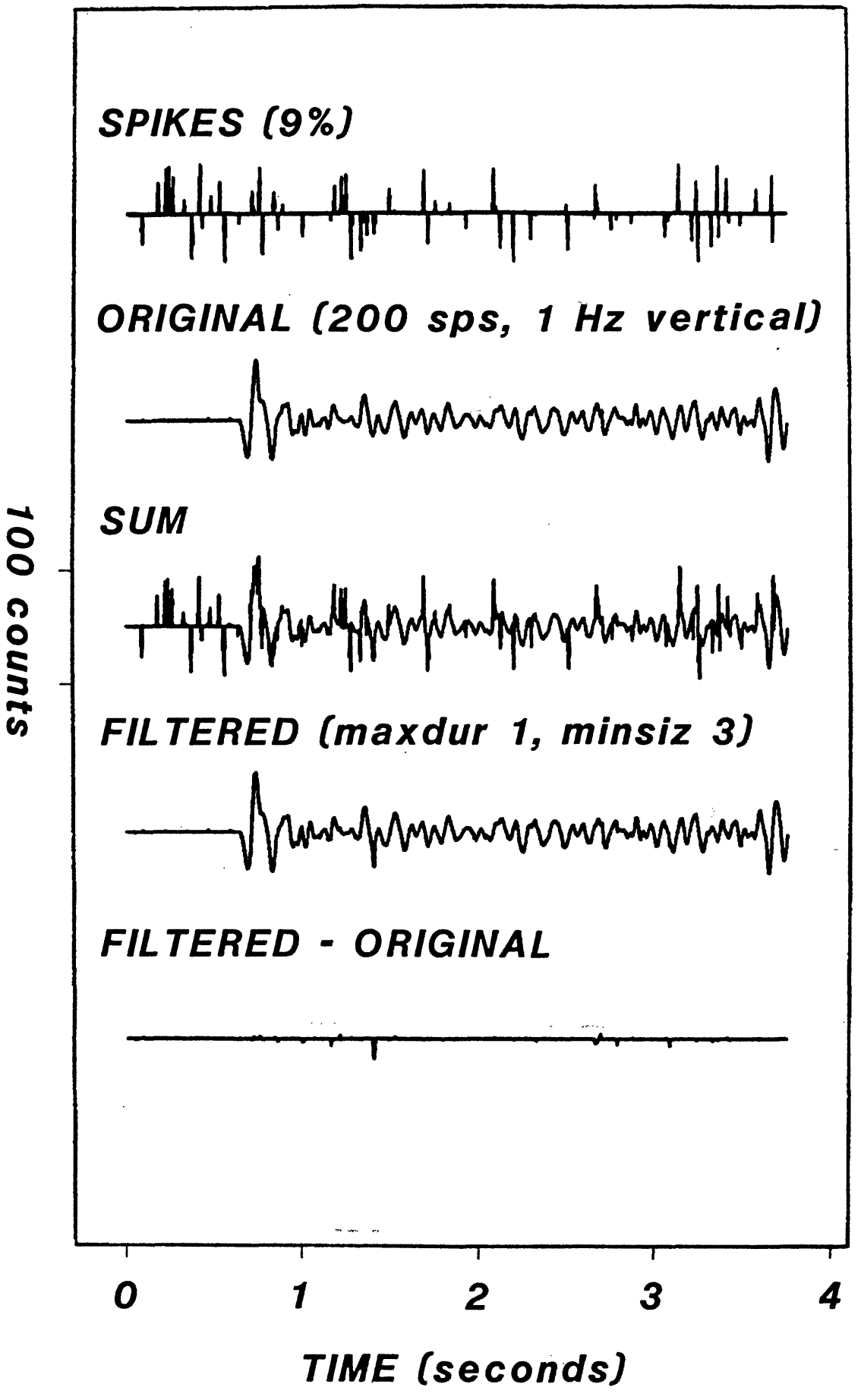

SPIKES (9%)
ORIGINAL (200 sps, 1 Hz vertical)
SUM
FILTERED (maxdur 1, minsiz 3)
FILTERED - ORIGINAL
100 counts
0
1
2
3
4
TIME (seconds)

Figure 4