WHOI-80-23

Copy 2

Woods Hole Oceanographic Institution

TIME SERIES ANALYSIS PROGRAMS

by

Alan D. Chave

May 1980

TECHNICAL REPORT

*Prepared for the National Science Foundation
under Grants OCE 77-8633, OCE 77-82255, and
OCE 79-19258 and for the WHOI Ocean Indus-
tries Program.*

WOODS HOLE, MASSACHUSETTS 02543

WHOI-80-23

TIME SERIES ANALYSIS PROGRAMS

by

Alan D. Chave

WOODS HOLE OCEANOGRAPHIC INSTITUTION
Woods Hole, Massachusetts 02543

May 1980

TECHNICAL REPORT

*Prepared for the National Science Foundation under Grants
OCE 77-8633, OCE 77-82255, and OCE 79-19258 and for the
WHOI Ocean Industries Program.*

*Reproduction in whole or in part is permitted for any pur-
pose of the United States Government. This report should
be cited as: Woods Hole Oceanographic Institution Technical
Report WHOI-80-23.*

Approved for Distribution

John I. Ewing, Chairman
Department of Geology & Geophysics

ABSTRACT

This report documents a set of Fortran subroutines for manipulating one dimensional, complex-valued time series information. The programs are divided into two libraries: TSLIB for ordinary Fourier analysis and ACLIB for more specialized applications. The latter includes interpolation routines, maximum entropy spectral analysis programs, and subroutines for advanced mathematical operations. The programs are designed for efficiency and speed with a minimum of hidden operations and memory accesses. These subroutines have been used extensively in the analysis of magnetotelluric and other geophysical data.

TABLE OF CONTENTS

TABLE OF PROGRAMS	2
INTRODUCTION	3
OVERVIEW	3
APPENDIX I: PROGRAMMING EXAMPLES	6
APPENDIX II: DESCRIPTION OF TSLIB SUBROUTINES	9
APPENDIX III: DESCRIPTION OF ACLIB SUBROUTINES	14
APPENDIX IV: LISTINGS FOR TSLIB SUBROUTINES	17
APPENDIX V: LISTINGS FOR ACLIB SUBROUTINES	27
APPENDIX VI: COMPATIBILITY	52
REFERENCES	53

TABLE OF PROGRAMS

<u>SUBROUTINE</u>	<u>DESCRIPTION</u>	<u>LISTING</u>
AMPLIFY	9	18
BANDPASS	9	18
BURG	14	30
CARTESIAN	9	18
CLEAR	9	18
COMPLEX	10	19
CONJUGATE	10	19
CONVOLVE	10	19
COPY	10	19
CORRELATE	10	19
CPOLY	15	37
CSVD	15	32
DETREND	10	20
DIVIDE	10	20
FFT	10	20
FFTINVERSE	10	20
FORK	11	21
GEOWINDOW	11	22
IBINARY	11	23
IMAGINARY	11	23
INTERPOLATE	14	28
LENCRIT	14	31
LOGPOWER	11	23
MINUS	11	23
PADZERO	11	23
PEF	14	30
PLUS	12	24
POLAR	12	24
POWER	12	24
POWERDB	12	24
REALPART	12	25
SETVARIANCE	12	25
SPLINE	14	29
SWAP	12	25
TIMES	12	25
WINDOW	12	26
ZEROMEAN	13	26

INTRODUCTION

Geophysical and oceanographic data are often acquired as long records of a variable determined at known intervals of space or time. Examples of such data series include measurements of the natural electromagnetic fields as a function of time and measurements of a potential field like gravity as a function of position along a ship track. These sequences are given the collective name of time series.

Time series analysis is the application of quantitative statistical and mathematical transformations to these data sequences. They are usually examined as discrete functions, with a constant increment of time between samples. While most of the mathematical procedures associated with time series analysis are conceptually simple, the computational methods required to implement them are often tedious and fraught with the potential for error. The development of a system of rules for data storage and handling, and a set of computer programs to implement them, can improve productivity in the routine analysis of data.

This report describes a set of FORTRAN subroutines for time series analysis that have found extensive use in the reduction of magnetotelluric and paleomagnetic data. The creation of this processor was motivated by a desire for efficiency and speed with a minimum of hidden actions on the data. This approach differs significantly from that used in the PATSY processor (Denham, 1977). The programs in this document are not intended for use by beginning or inexperienced programmers, as PATSY serves that function well. Some care by the programmer in storage allocation and starting address computation is necessary, but the resulting improvement in throughput and speed makes these routines applicable to large data sets where cost and time become important factors.

OVERVIEW

This report divides logically into two sections: documentation of a general time series analysis routine package called TSLIB and a second discussion of a special package called ACLIB. The former follows the name convention in PATSY closely and is useful for most routine time series analysis applications. The second package contains

interpolation routines, a set of programs for maximum entropy spectral analysis, and two subroutines for manipulation of complex data.

TSLIB provides the user with a complete package for ordinary time series analysis. All data arrays in TSLIB are complex and all computation with them is done in place. While this requires care by the programmer to save intermediate results for later use, the saving in core storage and improvement in speed over other approaches is significant. TSLIB contains utility routines for real to complex data conversion and array duplication. At the heart of the processor is an efficient fast Fourier transform subroutine and auxiliary routines for detrending, demeaning, filtering, windowing, and power computation are provided. No input or output routines are included.

ACLIB contains some unrelated subroutines for data manipulation. A one-dimensional cubic spline interpolation routine is included. Cubic splines offer the smoothest way to interpolate irregularly spaced data, as the continuity of the data and its first and second derivatives are preserved. Additional information can be found in de Boor (1978).

The second set of programs in ACLIB are designed for use in maximum entropy spectral analysis. The maximum entropy approach is data adaptive and minimizes some of the problems of ordinary Fourier analysis when treating short, noisy data sections and few cycles of the periods of interest. The resolution of maximum entropy is much higher than conventional approaches provide and the spectra are smoothed in an optimal manner. An excellent review of the maximum entropy method appears in Haykin (1979).

The final two subroutines in ACLIB are taken from the published literature and are included due to their wide applicability and rarity. The first routine performs the singular value decomposition of an arbitrary complex valued matrix. This matrix transformation is applicable in geophysical inverse theory (Jupp and Vozoff, 1975), transfer function estimation (Jupp, 1978; Chave et al., 1980), and least squares problems (Lawson and Hanson, 1974). The second routine finds the roots or zeroes of a complex polynomial of any degree. Precision is maintained by a scaling process and it has successfully been applied to polynomials of degree 50 or more. This subroutine

has been applied to the choice of prediction error filter length in the maximum entropy spectral method (Chave and Denham, 1980).

The remainder of this document consists of a set of appendices presenting programming examples, descriptions of the subroutines, and complete FORTRAN listings. A final appendix on compatibility is included to facilitate use at other institutions. A tutorial approach to time series analysis is beyond the scope of this report. Interested readers are referred to textbooks like Otnes and Enochson (1972) and Bath (1974).

APPENDIX I: PROGRAMMING EXAMPLES

This appendix contains three examples of computer programs using the routines in TSLIB and ACLIB. Details of calling conventions and data storage conventions are contained in Appendices II and III. In all of the examples data is input to the program by subroutine INPUT and output from it by subroutine OUTPUT. These routines are assumed to be user supplied.

Example 1: Bandpass filter a data set in the frequency domain and convert back to the time domain.

```

        COMPLEX WORK(512)
        REAL*4 DATA(512)
C      INPUT THE DATA
        CALL INPUT(DATA,NPTS,K1,K2,K3,K4)
C      ZERO OUT THE COMPLEX WORK AREA
        CALL CLEAR(WORK,512)
C      TRANSFER THE DATA INTO THE WORK AREA
        CALL COMPLEX(DATA,WORK,NPTS)
C      DETREND AND ZEROMEAN THE DATA
        CALL DETREND(WORK,NPTS)
        CALL ZEROMEAN(WORK,NPTS)
C      COMPUTE THE FOURIER TRANSFORM
        MPTS=IBINARY(NPTS)
        CALL FFT(WORK,NPTS,MPTS)
C      BANDPASS FILTER USING FILTER PARAMETERS INPUT BEFORE
        CALL BANDPASS(WORK,MPTS,K1,K2,K3,K4)
C      CONVERT BACK TO THE TIME DOMAIN
        CALL FFTINVERSE(WORK,MPTS)
C      OBTAIN THE REAL PART AND OUTPUT
        CALL REALPART(DATA,WORK,MPTS)
        CALL OUTPUT(DATA,MPTS)
        END

```

Remarks: It is up to the programmer to ensure that the indices K1-K4, corresponding to corner frequencies for the filter, are within the Nyquist band.

Example 2: Compute the auto and cross spectrum for two data sets.

```

        COMPLEX WORK(1536)
        REAL*4 DATA(512)
        CALL CLEAR(WORK,1536)
C      INPUT THE DATA TO TWO PARTS OF ARRAY WORK
        CALL INPUT(DATA,NPTS)

```

```

      CALL COMPLEX(DATA,WORK,NPTS)
      CALL INPUT(DATA,NPTS)
      MPTS=IBINARY(NPTS)
      CALL COMPLEX(DATA,WORK(MPTS+1),NPTS)
C     ZEROMEAN AND WINDOW THE DATA
      CALL ZEROMEAN(WORK,NPTS)
      CALL WINDOW(WORK,NPTS,1,10,NPTS-10,NPTS)
      CALL ZEROMEAN(WORK(MPTS+1),NPTS)
      CALL WINDOW(WORK(MPTS+1),NPTS,1,10,NPTS-10,NPTS)
C     COMPUTE THE FOURIER TRANSFORMS
      CALL FFT(WORK,NPTS,MPTS)
      CALL FFT(WORK(MPTS+1),NPTS,MPTS)
C     COMPUTE THE CROSS POWER
      CALL COPY(WORK,WORK(2*MPTS+1),MPTS)
      CALL CONJUGATE(WORK(MPTS+1),MPTS)
      CALL TIMES(WORK(2*MPTS+1),WORK(MPTS+1),MPTS)
C     COMPUTE THE AUTO POWERS
      CALL POWER(WORK,MPTS)
      CALL POWER(WORK(MPTS+1),MPTS)
C     OUTPUT THE RESULTS
      CALL REALPART(DATA,WORK,MPTS)
      CALL OUTPUT(DATA,MPTS)
      CALL REALPART(DATA,WORK(MPTS+1),MPTS)
      CALL OUTPUT(DATA,MPTS)
      CALL POLAR(WORK(2*MPTS+1),MPTS)
      CALL REALPART(DATA,WORK(2*MPTS+1),MPTS)
      CALL OUTPUT(DATA,MPTS)
      CALL IMAGINARY(DATA,WORK(2*MPTS+1),MPTS)
      CALL OUTPUT(DATA,MPTS)
      END

```

Remarks: This program illustrates the use of a long working array as a series of shorter ones. Care must be taken to allocate space within the array and include room for calculation of the Fourier transform. Note that ZEROMEAN is called before WINDOW. This is necessary to avoid large jumps at the ends of the data.

Example 3; Interpolate an irregularly spaced data set to a new sample interval using cubic splines.

```

      REAL*4 INITX(100),INITY(100),FINALX(100),FINALY(100)
      REAL*4 WORK(300)
      CALL INPUT(INITX,INITY,NPTS,SAMP)
      FINALX(1)=INITX(1)
      DO 10 I=2,NPTS
10  FINALX(I)=FINALX(I-1)+SAMP
      CALL INTERPOLATE(INITY,INITX,FINALY,FINALX,WORK,

```



```
$NPTS,NPTS,IBAD,NF,NL)  
IF (IBAD.EQ.1) THEN  
    CALL OUTPUT(FINALY(NF),NL-NF+1)  
ELSE  
    CALL OUTPUT(FINALY,NPTS)  
END IF  
END
```

Remarks: Note that the error flag IBAD is checked to ensure that FINALX contains no values outside the range in INITX. If it does, only those locations in FINALY containing interpolated values are output.

APPENDIX II: DESCRIPTION OF TSLIB SUBROUTINES

This appendix contains an alphabetical listing of the general time series routines in TSLIB. The FORTRAN coding for these routines is included as Appendix IV.

Some conventions used in this appendix include:

1. WORK, WORK1, etc. refer to complex arrays.
2. Integer quantities begin with the letters I-N.
3. All angles are in degrees with internal conversion to radians as needed.
4. All Fourier transforms are stored in a special way. The frequencies are arranged from DC, lowest positive frequency, ...Nyquist frequency, highest negative frequency, ...lowest negative frequency. Many of the subroutines assume this storage scheme and care must be taken not to change it before calling some programs.

SUBROUTINE AMPLIFY(WORK,NPTS,GAIN)

This subroutine multiplies all NPTS elements of WORK by the real gain factor GAIN. The routine is used to scale data sets. For example, a power spectrum computed by POWER could be converted to power spectral density by using AMPLIFY with $GAIN=2*SAMP$, where SAMP is the data sample interval.

SUBROUTINE BANDPASS(WORK,NPTS,K1,K2,K3,K4)

The data in WORK is assumed to be a Fourier transform with the usual storage convention. The real and imaginary parts of the positive and negative frequencies are multiplied separately by a cosine tapered filter. The filter has a value of zero from indices 1 to K1, is cosine tapered to a value of one from K1 to K2, has a value of one from K2 to K3, is cosine tapered from K3 to K4, and is zero from indices K4 to the Nyquist. Note that K1-K4 are the indices of the frequencies in the Fourier transform rather than the frequencies themselves. It is up to the user to ensure that K1-K4 lie in the Nyquist band. Note that setting $K1=K2=1$ results in a low pass filter, and similar techniques can be used to produce a high pass filter.

SUBROUTINE CARTESIAN(WORK,NPTS)

This subroutine converts the values in WORK to Cartesian form on a point by point basis. Initially, a given entry in WORK is assumed to have the form $CMPLX(AMP,PHASE)$ and is converted to $CMPLX(X,Y)$ where $X=AMP*\cos(PHASE)$ and $Y=AMP*\sin(PHASE)$. POLAR is the inverse operation to CARTESIAN.

SUBROUTINE CLEAR(WORK,NPTS)

This subroutine sets each value of WORK to $CMPLX(0,0)$. It is used to initialize arrays.

SUBROUTINE COMPLEX(RARRAY,WORK,NPTS)

This subroutine places each element of the real array RARRAY in the corresponding real part of WORK and sets the imaginary part of WORK to zero. This routine is commonly used to input data to the complex working array for further processing.

SUBROUTINE CONJUGATE(WORK,NPTS)

This subroutine replaces each element $\text{CMPLX}(X,Y)$ in WORK with its complex conjugate $\text{CMPLX}(X,-Y)$. This routine is used in computing the cross spectrum of A and B, where the cross spectrum is A times the conjugate of B.

SUBROUTINE CONVOLVE(WORK1,WORK2,WORK3,NPTS,MPTS)

This subroutine convolves the array WORK2 with WORK1. The result is stored in WORK3. The input arrays WORK1 and WORK2 have lengths of NPTS and MPTS and the output array WORK3 has a length of $\text{NPTS}+\text{MPTS}-1$.

SUBROUTINE COPY(WORK1,WORK2,NPTS)

This subroutine copies the data in WORK1 to WORK2. The routine is useful for saving intermediate results for later use.

SUBROUTINE CORRELATE(WORK1,WORK2,WORK3,NPTS,MPTS)

This subroutine computes the correlation of WORK2 with WORK1 and places the result in WORK3. The two input vectors WORK1 and WORK2 have lengths of NPTS and MPTS respectively. The result has a length of $\text{NPTS}+\text{MPTS}-1$. This routine is useful in computing the auto and cross correlation functions to get the spectrum by the old Blackman-Tukey method. The autocorrelation is computed by copying the data in WORK1 to WORK2 before calling CORRELATE.

SUBROUTINE DETREND(WORK,NPTS)

This subroutine fits a complex least squares straight line to the data in WORK. The slope is removed from the data and the mean value is unaffected. All calculations are performed in double precision. This routine is useful in removing instrumental drift or long term changes from a data set to examine shorter term behavior.

SUBROUTINE DIVIDE(WORK1,WORK2,NPTS)

This subroutine divides WORK1 by WORK2 on a term by term basis. If an element of WORK2 is zero, then the result is set to the maximum allowable and no error message is output.

SUBROUTINE FFT(WORK,NPTS,MPTS)

This subroutine is the calling routine for subroutine FORK to compute the fast Fourier transform. NPTS is the number of elements in WORK. MPTS is the user supplied power of 2 that is greater than or equal to NPTS and may be provided using function IBINARY. FFT will pad the data in WORK with zeroes from $\text{NPTS}+1$ to MPTS before calling FORK. MPTS must be a power of 2 or memory access violations will result. Note that the dimensions of WORK must be MPTS, not NPTS.

SUBROUTINE FFTINVERSE(WORK,NPTS)

This subroutine is the calling routine for subroutine FORK to

compute the inverse Fourier transform. NPTS must be a power of 2. This routine performs the inverse operation to subroutine FFT.

SUBROUTINE FORK(NPTS,WORK,SIGNI)

This subroutine is a fast Fourier transform program that operates on complex data in place. NPTS is the dimension of the complex array WORK and must be a power of 2. SIGNI is -1 for the forward transform and +1 for the inverse. FORK preserves length and alternate calls will return the original data. This subroutine was originally written by J. Claerbout of Stanford University and subsequently rewritten by A. Chave to use the block IF construction on the VAX 11/780 computer, resulting in an approximate doubling in speed. This routine is not normally called by a user since FFT and FFTINVERSE take care of the preliminaries.

SUBROUTINE GEOWINDOW(WORK,NPTS,ITYPE)

This subroutine performs windowing of spectra in the Fourier domain using two special weighting functions. The windowing is accomplished by closed convolution with a seven weight filter. ITYPE selects the type of filter. If ITYPE=1, then Goodman-Enochson-Otnes weights are used. This filter is designed to minimize long distance leakage and is useful when band averaging power or cross spectra. If ITYPE=2, then Goodman weights are used. This filter is designed to minimize leakage between nearby frequencies and is useful for spectral section averaging. For details see Otnes and Enochson (1972). These spectral windows are far superior to cosine tapering in the time domain (see subroutine WINDOW).

FUNCTION IBINARY(NPTS)

This function returns the integer power of 2 that is greater than or equal to its argument. IBINARY can be used in place of MPTS in the call to FFT.

SUBROUTINE IMAGINARY(RARRAY,WORK,NPTS)

This subroutine places the imaginary part of the array WORK in the real array RARRAY. WORK is not affected by the call.

SUBROUTINE LOGPOWER(WORK,NPTS,FNYQ)

This subroutine computes the base 10 logarithm of the power spectral density. WORK is assumed to contain a Fourier transform in Cartesian form. The power is calculated by squaring each Fourier coefficient and converted to power spectral density by dividing by the Nyquist frequency FNYQ. Finally, the logarithm of each element in WORK is obtained.

SUBROUTINE MINUS(WORK1,WORK2,NPTS)

This subroutine subtracts WORK2 from WORK1 on a term by term basis. WORK2 is not changed.

SUBROUTINE PADZERO(WORK,NPTS,MPTS)

This subroutine pads the array WORK with zeroes from NPTS+1 to MPTS. The routine is called by FFT and is not normally accessed directly by the user.

SUBROUTINE PLUS(WORK1,WORK2,NPTS)

This subroutine adds the vectors WORK1 and WORK2 on a term by term basis. The result is stored in WORK1 and WORK2 is unchanged.

SUBROUTINE POLAR(WORK,NPTS)

This subroutine converts the elements of the array WORK from Cartesian to polar form. Each element $CMPLX(X,Y)$ is changed to $CMPLX(AMP,PHASE)$ where $AMP=\sqrt{X^2+Y^2}$ and $PHASE=ATAN2(Y,X)$ in degrees. All angles are placed on the open interval $(0,360)$. The singular point $X=Y=0$ is given a phase of 0. The routine is the inverse of CARTESIAN and is normally used for plotting of Fourier transforms or cross spectra.

SUBROUTINE POWER(WORK,NPTS)

This subroutine computes the power in the array WORK, which is assumed to contain a Fourier transform. Each element of WORK is squared and the result is stored in the real part of WORK. The imaginary part of WORK is set to zero.

SUBROUTINE POWERDB(WORK,NPTS)

This subroutine computes the power of each element of the Fourier transform in WORK and converts it to relative decibels. The largest value in the array is found and used to scale the remaining elements by division. The result is converted to db by taking the base 10 logarithm and multiplying by 10. If any original element of WORK is zero then the resulting entry is set to -100.

SUBROUTINE REALPART(RARRAY,WORK,NPTS)

This subroutine puts the real part of WORK into the real array RARRAY. WORK is not changed.

SUBROUTINE SETVARIANCE(WORK,NPTS,VAR)

This subroutine sets the variance of WORK to the real value VAR. This routine is useful when unitvariancing many channels of data to remove weighting effects.

SUBROUTINE SWAP(WORK,NPTS)

This subroutine rearranges the positive and negative frequencies of the Fourier transform in WORK. The output from subroutine FORK is stored in the order DC, positive frequencies, Nyquist frequency, and decreasing negative frequencies. SWAP changes this to decreasing negative frequencies through the DC term to increasing positive frequencies to the Nyquist frequency. This routine is useful in plotting Fourier transforms. Two calls to SWAP will return the original data.

SUBROUTINE TIMES(WORK1,WORK2,NPTS)

This subroutine multiplies the arrays WORK1 and WORK2 on a term by term basis. The result is stored in WORK1 and WORK2 is unchanged.

SUBROUTINE WINDOW(WORK,NPTS,K1,K2,K3,K4)

This subroutine multiplies the data in WORK by a cosine tapered window. The window is zero from 1 to K1, is cosine tapered from K1 to K2, has a value of one from K2 to K3, is cosine tapered from K3 to K4,

and is zero from K4 to NPTS. This routine is used in time domain tapering of data prior to taking the Fourier transform. Note that K1-K4 refer to locations in the array WORK.

SUBROUTINE ZEROMEAN(WORK,NPTS)

This subroutine sets the mean value of WORK to a value of zero. It is commonly used prior to taking the Fourier transform to avoid contamination of the low frequency portion of the spectrum.

APPENDIX III: DESCRIPTION OF ACLIB SUBROUTINES

This appendix contains descriptions of the subroutines in the ACLIB package. Complete program listings are reproduced in Appendix V.

SUBROUTINE INTERPOLATE(Y,X,YO,XO,WORK,N,NO,IBAD,NF,NL)

This subroutine performs a natural cubic spline interpolation of a given real data set. The input data vectors X and Y of length N contain abscissa values X and ordinate values Y that may be irregularly spaced. The entries in X must be monotonic. The user supplied vector XO contains NO new monotonic abscissa values for which new ordinate values YO will be found. The spline coefficients are determined by subroutine SPLINE and stored in the vector WORK, which must be dimensioned 3*N. The subroutine then checks the range of XO to ensure that it lies within the range of X. If XO is a subset of X, then IBAD=0, NF=1, and NL=NO, and the entire vector YO contains interpolated values. If XO contains values outside of X, then IBAD=1 and only those entries in YO from NF to NL contain interpolated values. The remaining entries in YO are set to zero.

SUBROUTINE SPLINE(NPTS,X,Y,B,C,D)

This subroutine computes the coefficients of a cubic natural spline given NPTS input abscissa values X and ordinate values Y. The entries in X must be monotonic. The output vectors B, C, and D contain the spline coefficients. If a new value XX lies between X(I) and X(I+1), a new interpolated value YY can be found using the exact cubic spline fit to the original data. Letting $T = XX - X(I)$, the new value is $YY = ((D(I)*T + C(I))*T + B(I))*T + Y(I)$. Note that C(NPTS)=0 while B(NPTS) and D(NPTS) are not defined. This subroutine is called by INTERPOLATE and is not normally accessed by the user.

SUBROUTINE BURG(X,REFCO,ERRPOW,N,M,WORK)

This subroutine calculates the M complex reflection coefficients REFCO given the N complex data X. M must be smaller than N. The reflection coefficients are used to compute a prediction error filter for calculation of the maximum entropy spectrum. BURG also returns the M real error power values ERRPOW for use in the call to LENCRT. A complex workspace WORK of length 2*N is required. The algorithm used in this subroutine is from BURG(1975) and does not require calculation of the autocorrelation function.

SUBROUTINE PEF(PEFC,LG1,LG2)

This is actually an entry point in subroutine BURG that computes a prediction error filter (PEF) from the reflection coefficients in REFCO and an old PEF. A call to BURG must precede the call to PEF. The complex PEF coefficients are contained in PEFC. The two integers LG1 and LG2 control the length or order of the PEF. LG1 is the length of the old PEF and should initially be set to one. LG2 is the length of the desired PEF and must be smaller than M.

SUBROUTINE LENCRT(ERRPOW,FPERR,CAT,AIC,M,N,NZ)

This subroutine computes three length criteria given the error

power values ERRPOW from subroutine BURG. The final prediction error (FPERR), Aikake information theoretic criterion (AIC), and convolutional autoregressive transfer function (CAT) values are returned. M is the number of entries in ERRPOW, FPERR, CAT, and AIC, and N is the number of data X in the call to BURG. NZ=1 indicates that the data were not zero-meaned while NZ=0 indicates that they were.

Remarks: The previous 3 subroutines and entry points are used in maximum entropy spectral analysis. The description deliberately does not indicate how they are applied. Interested users are referred to Burg (1975), Ulrych and Bishop (1975), Haykin (1979), and Chave and Denham (1980) for details.

SUBROUTINE CSVD(A,M,MAX,N,P,NU,NV,S,U,V)

This subroutine finds the singular value decomposition of the M by N complex matrix A, where M is greater than or equal to N. The dimensions of A are MMAX by NMAX. The N singular values are stored in the real vector S. The subroutine also finds the first NV columns of the N by N complex unitary matrix V and the first NU columns of the M by M complex unitary matrix U such that $\|A - USV^H\|$ is negligible compared to $\|A\|$. The only permitted values for NU are 0, N, and M, and the only permitted values for NV are 0 and N. The transformation may also optionally be applied to the N+1 through P columns of A if P is nonzero.

This subroutine is useful in solving least squares problems and in finding pseudoinverses for arbitrary matrices. For the overdetermined system $AX=B$, the pseudoinverse solution is derived by calling CSVD with $NV=N$ and the right hand side B in the N+1 through P columns of A. The rank r of A must be estimated from the singular values in S. After zeroing those singular values from r+1 to N, the solution is given by $X=VSU^HB$, where U^HB is furnished by CSVD in the N+1 through P positions of A.

The subroutine can also be used to find the minimum Euclidean distance approximation to a given matrix. The matrix B of rank r less than N which most closely approximates A is given by the singular value decomposition of A with the r+1 through N singular values zeroed out.

The subroutine is also useful in principal components and factor analysis. If the appropriate row or column operations are applied to A, the eigenvectors of the correlation or variance-covariance matrix contained in U and V while S contains the positive square root of their eigenvalues. Note that this is accomplished without ever finding the correlation matrix, improving the accuracy for large values of M and N.

Subroutine CSVD is taken from Businger and Golub (1969).

SUBROUTINE CPOLY(OPR,OPI,DEGREE,ZEROR,ZEROI,FAIL)

This subroutine finds the zeroes or roots of a complex polynomial. OPR and OPI are double precision vectors of the real and imaginary parts of the polynomial coefficients arranged in order of decreasing powers. DEGREE is the integer order of the polynomial. The subroutine returns the real and imaginary parts of the zeroes in the double precision vectors ZEROR and ZEROI. FAIL is a logical quantity that is

set .TRUE. if the leading coefficient is zero or if CPOLY finds too few zeroes. CPOLY is limited to polynomials of degree 149 or less, although precision is degraded for polynomials over degree 50. CPOLY is the calling subroutine for a set of 12 other subroutines. CPOLY was taken from Jenkins and Traub (1972).

APPENDIX IV: LIST OF FORTRAN CODING FOR TSLIB

This appendix contains an alphabetical listing of the source code for TSLIB. No attempt has been made to arrange these subroutines in a hierarchical manner, and some computer systems may require rearrangement of them so that calling subroutines precede called ones. All of the statements conform to VAX-11 Fortran IV-Plus for use on the Digital Equipment Corporation VAX 11/780 computer system. Appendix VI discusses some of the differences and potential incompatibilities with other machines.


```

SUBROUTINE AMPLIFY(A,N,GAIN)
  COMPLEX A(1)
  DO 10 I=1,N
10 A(I)=GAIN*A(I)
  RETURN
  END

```

```

SUBROUTINE BANDPASS(A,N,K1,K2,K3,K4)
  COMPLEX A(1),ZERO
  PARAMETER PI=3.1415926
  ZERO=CMPLX(0.,0.)
  DO 10 I=2,K2
  A(N-I+2)=ZERO
10 A(I)=ZERO
  IF(K1.NE.K2)THEN
    DO 20 I=K1+1,K2
    X=.5*(1.-COS(PI*(I-K1)/(K2-K1)))
    A(N-I+1)=X*A(N-I+1)
20 A(I)=X*A(I)
  END IF
  IF(K3.NE.K4)THEN
    DO 40 I=K3,K4
    X=.5*(1.+COS(PI*(I-K3)/(K4-K3)))
    A(N-I+1)=X*A(N-I+1)
40 A(I)=X*A(I)
  END IF
  IF(K4.LT.N/2+1)THEN
    DO 50 I=K4+1,N/2+1
    A(N-I+1)=ZERO
50 A(I)=ZERO
  END IF
  RETURN
  END

```

```

SUBROUTINE CARTESIAN(A,N)
  COMPLEX A(1)
  PARAMETER PI=3.1415926,RAD=PI/180.
  DO 10 I=1,N
12 X=REAL(A(I))*COS(AIMAG(A(I))*RAD)
11 Y=REAL(A(I))*SIN(AIMAG(A(I))*RAD)
10 A(I)=CMPLX(X,Y)
  RETURN
  END

```

```

SUBROUTINE CLEAR(A,N)
  COMPLEX A(1)
  DO 10 I=1,N
10 A(I)=0.0
  RETURN
  END

```

```

SUBROUTINE COMPLEX(A,B,N)
  REAL A(1)
  COMPLEX B(1)
  DO 10 I=1,N
10 B(I)=CMPLX(A(I),0.0)
  RETURN
  END

```

```

SUBROUTINE CONJUGATE(A,N)
  COMPLEX A(1)
  DO 10 I=1,N
10 A(I)=CONJG(A(I))
  RETURN
  END

```

```

SUBROUTINE CONVOLVE(A,B,C,N,M)
  COMPLEX A(1),B(1),C(1),ZERO
  ZERO=CMPLX(0.,0.)
  DO 5 I=1,N+M-1
5 C(I)=ZERO
  DO 10 I=1,N
  DO 10 J=1,M
10 C(I+J-1)=C(I+J-1)+A(I)*B(J)
  RETURN
  END

```

```

SUBROUTINE COPY(A,B,N)
  COMPLEX A(1),B(1)
  DO 10 I=1,N
10 B(I)=A(I)
  RETURN
  END

```

```

SUBROUTINE CORRELATE(A,B,C,N,M)
  COMPLEX A(1),B(1),C(1),ZERO
  ZERO=CMPLX(0.,0.)
  DO 5 I=1,N+M-1
5 C(I)=ZERO
  DO 10 I=1,N
  DO 10 J=1,M
10 C(I+J-1)=C(I+J-1)+A(I)*CONJG(B(M+1-J))
  RETURN
  END

```

MADE IN U.S.A.

12

11

10

9

8

7

6

5

4

3

SUBROUTINE DETREND(A,N)

COMPLEX A(1)

REAL*8 X,X2,YR,YI,XYR,XYI,SLR,SLI

X=0.

X2=0.

YR=0.

YI=0.

XYR=0.

XYI=0.

DO 10 I=1,N

YR=YR+DBLE(REAL(A(I)))

YI=YI+DBLE(AIMAG(A(I)))

XYR=XYR+X*DBLE(REAL(A(I)))

XYI=XYI+X*DBLE(AIMAG(A(I)))

X2=X2+X*X

10 X=X+1.

SLR=(N*XYR-X*YR)/(N*X2-X*X)

SLI=(N*XYI-X*YI)/(N*X2-X*X)

X=0.

DO 20 I=1,N

T1=REAL(A(I))-SNGL(SLR*X)

T2=AIMAG(A(I))-SNGL(SLI*X)

20 A(I)=CMPLX(T1,T2)

RETURN

END

SUBROUTINE DIVIDE(A,B,N)

COMPLEX A(1),B(1),ZERO

ZERO=CMPLX(0.,0.)

DO 10 I=1,N

IF(B(I).EQ.ZERO)THEN

A(I)=CMPLX(1.E+38,1.E+38)

GO TO 10

ELSE

A(I)=A(I)/B(I)

END IF

10 CONTINUE

RETURN

END

SUBROUTINE FFT(A,N,M)

COMPLEX A(1)

CALL PADZERO(A,N,M)

CALL FORK(M,A,-1.0)

RETURN

END

SUBROUTINE FFTINVERSE(A,N)

COMPLEX A(1)

CALL FORK(N,A,+1.0)

RETURN

END

SUBROUTINE FORK(LX,CX,SIGNI)
 COMPLEX CX(LX),CARG,CH,CTEMP,CONST
 PARAMETER PI=3.1415926
 CONST=SIGNI*CMPLX(0.,1.)

J=1

SC=SQRT(1./LX)

DO 5 I=1,LX

IF(I.LE.J)THEN

CTEMP=CX(I)*SC

CX(I)=CX(J)*SC

CX(J)=CTEMP

M=LX/2

ELSE

M=LX/2

END IF

3 CONTINUE

IF(J.GT.M)THEN

J=J-M

M=M/2

IF(M.GE.1)THEN

GO TO 3

ELSE

GO TO 5

END IF

ELSE

GO TO 5

END IF

5 J=J+M

L=1

6 ISTEP=2*L

DO 8 M=1,L

CARG=CONST*PI*(M-1)/L

CH=CEXP(CARG)

DO 8 I=M,LX,ISTEP

CTEMP=CH*CX(I+L)

CX(I+L)=CX(I)-CTEMP

8 CX(I)=CX(I)+CTEMP

L=ISTEP

IF(L.LT.LX)THEN

GO TO 6

ELSE

RETURN

END IF

END

MADE IN U.S.A.

$$SQR = \sqrt{HPTS}$$

$$RECIP = 1/HPTS$$

$$\frac{1}{M}$$


```

SUBROUTINE GEOWINDOW(X,N,ITYPE)
C   CONVOLUTION WINDOWING ROUTINE USING SELECTED WEIGHTS OF:
C   ITYPE=1 GIVES GOODMAN-ENOCHSON-OTNES WEIGHTS FOR BAND AVERAGING
C   ITYPE=2 GIVES GOODMAN WEIGHTS FOR SPECTRAL AVERAGING
C   REFERENCE: OTNES AND ENOCHSON, DIGITAL TIME SERIES ANALYSIS
COMPLEX X(1),S(3),T(3),U(4),Y
REAL A(2,3)
DATA A/-.1817,-.35,-.1707,-.875,-.1476,-.0625/
CALL SHAP(X,N)
DO 10 I=1,3
S(I)=X(N-I+1)
10 T(I)=X(I)
DO 50 K=1,N-3
DO 30 I=1,3
30 Y=X(K)+A(ITYPE,I)*(S(I)+X(K+I))
DO 40 I=3,2,-1
40 S(I)=S(I-1)
S(1)=X(K)
50 X(K)=Y
U(1)=X(N-1)
U(2)=X(N)
U(3)=T(1)
DO 100 K=N-2,N
DO 80 I=1,3
80 Y=X(K)+A(ITYPE,I)*(S(I)+U(I))
IF(K.EQ.N)THEN
X(K)=Y
CALL SHAP(X,N)
RETURN
ELSE
U(4)=T(K-N+4)
DO 90 I=1,3
90 U(I)=U(I+1)
DO 95 I=3,2,-1
95 S(I)=S(I-1)
S(1)=X(K)
END IF
100 X(K)=Y
RETURN
END

```

FUNCTION IBINARY(N)

DO 10 I=1,16

10 IF(N.LE.2**I)GO TO 15

15 IBINARY=2**I

RETURN

END

SUBROUTINE IMAGINARY(A,B,N)

REAL A(1)

COMPLEX B(1)

DO 10 I=1,N

10 A(I)=AIMAG(B(I))

RETURN

END

SUBROUTINE LOGPOWER(A,N,FNYQ)

COMPLEX A(1)

DO 10 I=1,N

X=CABS(A(I))**2

IF(X.EQ.0.)THEN

A(I)=CMPLX(ALOG10(1.E-30/FNYQ),0.)

GO TO 10

ELSE

A(I)=CMPLX(ALOG10(X/FNYQ),0.)

END IF

10 CONTINUE

RETURN

END

SUBROUTINE MINUS(A,B,N)

COMPLEX A(1),B(1)

DO 10 I=1,N

10 A(I)=A(I)-B(I)

RETURN

END

SUBROUTINE PADZERO(A,N,M)

COMPLEX A(1),ZERO

ZERO=CMPLX(0.,0.)

IF(N.EQ.M)THEN

RETURN

ELSE

DO 10 I=N+1,M

10 A(I)=ZERO

END IF

RETURN

END

MADE IN U.S.A


```

SUBROUTINE PLUS(A,B,N)
COMPLEX A(1),B(1)
DO 10 I=1,N
10 A(I)=A(I)+B(I)
RETURN
END

```

```

SUBROUTINE POLAR(A,N)
COMPLEX A(1)
PARAMETER PI=3.1415926,DEG=180./PI
DO 10 I=1,N
AMP=CABS(A(I))
X=REAL(A(I))
Y=AIMAG(A(I))
IF(X.EQ.0.)THEN
  IF(Y.EQ.0.)THEN
    PHASE=0.
    GO TO 10
  END IF
ELSE
  PHASE=DEG*ATAN2(Y,X)
  IF(PHASE.LT.0.)THEN
    PHASE=PHASE+360.
  END IF
END IF
10 A(I)=CMPLX(AMP,PHASE)
RETURN
END

```

```

SUBROUTINE POWER(A,N)
COMPLEX A(1)
DO 10 I=1,N
10 A(I)=CMPLX(CABS(A(I))**2,0.0)
RETURN
END

```

```

SUBROUTINE POWERDB(A,N)
COMPLEX A(1)
12 PEAK=-1.0E+38
DO 10 I=1,N
11 A(I)=CMPLX(CABS(A(I))**2,0.0)
10 PEAK=MAX(PEAK,REAL(A(I)))
DO 20 I=1,N
9 IF(REAL(A(I)).EQ.0.)THEN
8 A(I)=CMPLX(-100.,0.)
GOTO 20
ELSE
7 A(I)=CMPLX(10.*ALOG10(REAL(A(I))/PEAK),0.)
END IF
6 20 CONTINUE
5 RETURN
END

```



```

SUBROUTINE REALPART(A,B,N)
  REAL A(1)
  COMPLEX B(1)
  DO 10 I=1,N
10 A(I)=REAL(B(I))
  RETURN
  END

```

```

SUBROUTINE SETVARIANCE(A,N,VAR)
  COMPLEX A(1),SUM,AVG
  SUM=CMPLX(0.0,0.0)
  DO 10 I=1,N
10 SUM=SUM+A(I)/N
  AVG=SUM
  SUM=CMPLX(0.0,0.0)
  DO 20 I=1,N
20 SUM=SUM+(A(I)-AVG)*CONJG(A(I)-AVG)
  VOLD=REAL(SUM)/N
  SCALE=SQRT(VAR/VOLD)
  DO 30 I=1,N
30 A(I)=(A(I)-AVG)*SCALE+AVG
  RETURN
  END

```

```

SUBROUTINE SWAP(A,N)
  COMPLEX A(N),B
  DO 10 I=1,N/2
  B=A(I)
  A(I)=A(N/2+I)
10 A(N/2+I)=B
  RETURN
  END

```

```

SUBROUTINE TIMES(A,B,N)
  COMPLEX A(1),B(1)
  DO 10 I=1,N
10 A(I)=A(I)*B(I)
  RETURN
  END

```

MADE IN U.S.A.

12

11

10

9

8

7

6

5

4

3

2

SUBROUTINE WINDOW(A,N,K1,K2,K3,K4)

COMPLEX A(1),ZERO

PARAMETER PI=3.1415926

ZERO=CMPLX(0.,0.)

IF(K1.GE.1)THEN

DO 10 I=1,K1

10 A(I)=ZERO

END IF

IF(K1.NE.K2)THEN

DO 20 I=K1+1,K2

20 A(I)=A(I)*.5*(1.-COS(PI*(I-K1)/(K2-K1)))

END IF

IF(K3.NE.K4)THEN

DO 30 I=K3,K4-1

30 A(I)=A(I)*.5*(1.+COS(PI*(I-K3)/(K4-K3)))

END IF

IF(K4.LE.N)THEN

DO 40 I=K4,N

40 A(I)=ZERO

END IF

RETURN

END

SUBROUTINE ZEROMEAN(A,N)

COMPLEX A(1),SUM

SUM=CMPLX(0.0,0.0)

DO 10 I=1,N

10 SUM=SUM+A(I)/N

DO 20 I=1,N

20 A(I)=A(I)-SUM

RETURN

END

12

11

10

9

8

7

6

5

APPENDIX V: LIST OF FORTRAN CODING FOR ACLIB

SUBROUTINE INTERPOLATE(Y,X,YO,XO,WORK,N,NO,IBAD,NF,NL)
 C INTERPOLATION BY CUBIC NATURAL SPLINES
 C INPUT DATA X,Y, I=1,...,N AND NEW XO, I=1,...,NO
 C OUTPUT YO, I=NF,...,NL
 C IBAD=0 NF=1,NL=NO
 C IBAD=1 SOME XO OUTSIDE INPUT RANGE, OUTPUT ONLY FROM NF TO N
 C OTHER YO SET TO ZERO
 REAL*4 X(N),Y(N),XO(NO),YO(NO),WORK(3*N)
 CALL SPLINE(N,X,Y,WORK,WORK(N+1),WORK(2*N+1))
 IBAD=0
 IF(XO(1).LT.X(1))THEN
 DO 10 I=1,NO
 IF(XO(I).GE.X(1))THEN
 NF=I
 IBAD=1
 GO TO 20
 ELSE
 YO(I)=0.
 END IF
 10 CONTINUE
 ELSE
 NF=1
 END IF
 20 IF(XO(NO).GT.X(N))THEN
 DO 30 I=NO,1,-1
 IF(XO(I).LE.X(N))THEN
 NL=I
 IBAD=1
 GO TO 40
 ELSE
 YO(I)=0.
 END IF
 30 CONTINUE
 ELSE
 NL=NO
 END IF
 40 DO 50 I=1,N
 IF(X(I).GT.XO(NF))THEN
 12 K1=I-1
 GO TO 60
 11 END IF
 50 CONTINUE
 10 60 DO 70 I=N-1,1,-1
 IF(X(I).LT.XO(NL))THEN
 9 K2=I+1
 GO TO 80
 8 END IF
 70 CONTINUE
 80 T=XO(NF)-X(K1)
 YO(NF)=((WORK(2*N+K1)*T+WORK(N+K1))*T+WORK(K1))*T+Y(K1)
 M=NF+1
 DO 100 I=K1,K2-1
 5 90 CONTINUE
 IF(XO(M).LE.X(I+1))THEN
 T=XO(M)-X(I)
 YO(M)=((WORK(2*N+I)*T+WORK(N+I))*T+WORK(I))*T+Y(I)
 M=M+1
 GO TO 90
 END IF
 100 CONTINUE


```

SUBROUTINE SPLINE(NPTS,X,Y,B,C,D)
C  COMPUTATION OF THE COEFFICIENTS OF A CUBIC NATURAL SPLINE
REAL*4 X(1),Y(1),B(1),C(1),D(1)
M2=NPTS-1
S=0.
DO 10 K=1,M2
D(K)=X(K+1)-X(K)
R=(Y(K+1)-Y(K))/D(K)
C(K)=R-S
10 S=R
R=0.
S=0.
C(1)=0.
C(NPTS)=0.
IF(M2.LT.2)GO TO 35
DO 20 K=2,M2
C(K)=C(K)+R*C(K-1)
B(K)=(X(K-1)-X(K+1))*2.-R*S
S=D(K)
20 R=S/B(K)
DO 30 K=2,M2
KK=M2-K+2
30 C(KK)=(D(KK)*C(KK+1)-C(KK))/B(KK)
35 DO 40 K=1,M2
B(K)=(Y(K+1)-Y(K))/D(K)-(2.*C(K)+C(K+1))*D(K)
D(K)=(C(K+1)-C(K))/D(K)
40 C(K)=3.*C(K)
RETURN
END

```

MADE IN U.S.A.

12

11

10

9

8

7

6

5

4

3


```

SUBROUTINE BURG(X,REFCO,ERRPOW,N,M,WORK)
C SUBROUTINE TO CALCULATE COMPLEX REFLECTION COEFFICIENTS AND
C PREDICTION ERROR FILTER GIVEN COMPLEX DATA X, I=1,...,N
C REFCO CONTAINS THE M REFLECTION COEFFICIENTS
C PEFC CONTAINS THE M PEF TERMS
C ERRPOW CONTAINS THE M ERROR POWER TERMS
C WORK IS A COMPLEX WORK SPACE OF LENGTH 2*N
C M MUST BE SMALLER THAN N
C A. CHAVE DECEMBER 1979 WHOI
COMPLEX X(N),REFCO(M),PEFC(1),WORK(2*N)
COMPLEX RO,DPM,EPI
REAL*4 ERRPOW(M)
C COMPUTE MEAN POWER OR ZERO LAG AC COEFF
RO=CMPLX(0.,0.)
DO 10 I=1,N
10 RO=RO+X(I)*CONJG(X(I))/N
REFCO(1)=RO
C COMPUTE REFLECTION COEFFICIENTS
DO 20 I=1,N
WORK(I)=X(I)
20 WORK(N+I)=X(I)
DO 40 J=2,M
DEN=0.
DPM=CMPLX(0.,0.)
DO 30 I=J,N
DEN=DEN+CABS(WORK(N+I))*CABS(WORK(N+I))+CABS(WORK(I-J+1))*
3 CABS(WORK(I-J+1))
30 DPM=DPM+WORK(N+I)*CONJG(WORK(I-J+1))
REFCO(J)=-2.*DPM/DEN
DO 40 I=J,N
EPI=WORK(N+I)
WORK(N+I)=WORK(N+I)+REFCO(J)*WORK(I-J+1)
40 WORK(I-J+1)=WORK(I-J+1)+CONJG(REFCO(J))*EPI
C COMPUTE ERROR POWER
ERRPOW(1)=RO
DO 50 I=2,M
50 ERRPOW(I)=ERRPOW(I-1)*(1.-CABS(REFCO(I))**2)
RETURN
12 C COMPUTE PREDICTION ERROR FILTER
ENTRY PEF(PEFC,LG1,LG2)
11 PEFC(1)=CMPLX(1.,0.)
LG1=LG1+1
10 DO 60 I=LG1,LG2
60 PEFC(I)=CMPLX(0.,0.)
9 DO 80 I=LG1,LG2
DO 70 J=2,I
70 WORK(J)=PEFC(J)+REFCO(I)*CONJG(PEFC(I-J+1))
DO 80 J=2,I
80 PEFC(J)=WORK(J)
LG1=LG2
RETURN
END

```

MADE IN U.S.A.

SUBROUTINE LENCRT(ERRPOW,FPERR,CAT,AIC,M,N,NZ)

REAL*4 ERRPOW(M),FPERR(M),AIC(M),CAT(M)

FPERR(1)=1.

IF(NZ.EQ.0.)THEN

DO 10 I=2,M

10 FPERR(I)=ERRPOW(I)/ERRPOW(1)*FLOAT(N+I)/FLOAT(N-1)

ELSE

DO 20 I=2,M

20 FPERR(I)=ERRPOW(I)/ERRPOW(1)*FLOAT(N+I+1)/FLOAT(N-I-1)

END IF

DO 30 I=1,M

30 AIC(I)=ALOG(ERRPOW(I)/ERRPOW(1))+2.*I/N

DO 50 I=1,M

SUM=0.

DO 40 J=1,I

40 SUM=SUM+FLOAT(N-J)/(N*ERRPOW(J)/ERRPOW(1))

50 CAT(I)=SUM/N-FLOAT(N-I)/(N*ERRPOW(I)/ERRPOW(1))

RETURN

END


```

SUBROUTINE CSVD(A,MMAX,NMAX,M,N,P,NU,NV,S,U,V)
C   SINGULAR VALUE DECOMPOSITION OF AN M BY N COMPLEX MATRIX A
C   WHERE M.GT.N. THE SINGULAR VALUES ARE STORED IN THE VECTOR S.
C   THE FIRST NU COLUMNS OF THE M BY M UNITARY MATRIX U AND THE FIR
C   NV COLUMNS OF THE N BY N UNITARY MATRIX V THAT MINIMISE
C   DET(A-USV*) ARE ALSO COMPUTED
C   REFERENCE: BUSINGER AND GOLUB, COMM. A.C.M. 12, 564(1969)
COMPLEX A(MMAX,NMAX),U(MMAX,MMAX),V(NMAX,NMAX),Q,R
INTEGER P
REAL S(NMAX),B(100),C(100),T(100)
PARAMETER ETA=1.2E-7,TOL=2.4E-32
NP=N+P
N1=N+1
C   HOUSEHOLDER REDUCTION
C(1)=0.
K=1
10 K1=K+1
C   ELIMINATION OF A(I,K), I=K+1,...,M
Z=0.
DO 20 I=K,M
20 Z=Z+CABS(A(I,K))*+2
B(K)=0.
IF(Z.LE.TOL)GO TO 70
Z=SQRT(Z)
B(K)=Z
W=CABS(A(K,K))
Q=CMPLX(1.,0.)
IF(W.NE.0.)Q=A(K,K)/W
A(K,K)=Q*(Z+W)
IF(K.EQ.NP)GO TO 70
DO 50 J=K1,NP
Q=CMPLX(0.0,0.)
DO 30 I=K,M
30 Q=Q+CONJG(A(I,K))*A(I,J)
Q=Q/(Z*(Z+W))
DO 40 I=K,M
40 A(I,J)=A(I,J)-Q*A(I,K)
50 CONTINUE
12 C   PHASE TRANSFORMATION
Q=-CONJG(A(K,K))/CABS(A(K,K))
11 DO 60 J=K1,NP
60 A(K,J)=Q*A(K,J)
10 C   ELIMINATION OF A(K,J), J=K+2,...,N
70 IF(K.EQ.N)GO TO 140
Z=0.
DO 80 J=K1,N
80 Z=Z+CABS(A(K,J))*+2
C(K1)=0.
IF(Z.LE.TOL)GO TO 130
Z=SQRT(Z)
C(K1)=Z
W=CABS(A(K,K1))
Q=CMPLX(1.,0.)
IF(W.NE.0.)Q=A(K,K1)/W
A(K,K1)=Q*(Z+W)
DO 110 I=K1,M
Q=CMPLX(0.,0.)
DO 90 J=K1,N
90 Q=Q+CONJG(A(K,J))*A(I,J)
Q=Q/(Z*(Z+W))
DO 100 J=K1,N

```


100 A(I,J)=A(I,J)-Q*A(K,J)
 110 CONTINUE
 C PHASE TRANSFORMATION
 Q=-CONJG(A(K,K1))/CABS(A(K,K1))
 DO 120 I=K1,M
 120 A(I,K1)=A(I,K1)*Q
 130 K=K1
 GO TO 10
 C TOLERANCE FOR NEGLIBLE ELEMENTS
 140 EPS=0.
 DO 150 K=1,N
 S(K)=B(K)
 T(K)=C(K)
 150 EPS=AMAX1(EPS,S(K)+T(K))
 EPS=EPS*ETA
 C INITIALIZATION OF U AND V
 IF(NU.EQ.0)GO TO 180
 DO 170 J=1,NU
 DO 160 I=1,M
 160 U(I,J)=CMPLX(0.,0.)
 170 U(J,J)=CMPLX(1.,0.)
 180 IF(NV.EQ.0)GO TO 210
 DO 200 J=1,NV
 DO 190 I=1,N
 190 V(I,J)=CMPLX(0.,0.)
 200 V(J,J)=CMPLX(1.,0.)
 C QR DIAGONALIZATION
 210 DO 380 KK=1,N
 K=N1-KK
 C TEST FOR SPLIT
 220 DO 230 LL=1,K
 L=K+1-LL
 IF(ABS(T(L)).LE.EPS)GO TO 290
 IF(ABS(S(L-1)).LE.EPS)GO TO 240
 230 CONTINUE
 C CANCELLATION OF B(L)
 240 CS=0.
 SN=1.
 12 LI=L-1
 DO 280 I=L,K
 11 F=SN*T(I)
 T(I)=CS*T(I)
 10 IF(ABS(F).LE.EPS)GO TO 290
 H=S(I)
 9 W=SQRT(F*F+H*H)
 S(I)=W
 8 CS=H/W
 7 SN=-F/W
 IF(NU.EQ.0)GO TO 260
 DO 250 J=1,N
 6 X=REAL(U(J,LI))
 5 Y=REAL(U(J,I))
 U(J,LI)=CMPLX(X*CS+Y*SN,0.)
 250 U(J,I)=CMPLX(Y*CS-X*SN,0.)
 260 IF(NP.EQ.N)GO TO 280
 DO 270 J=N1,NP
 Q=A(LI,J)
 R=A(I,J)
 A(LI,J)=Q*CS+R*SN
 270 A(I,J)=R*CS-Q*SN
 280 CONTINUE


```

C      TEST FOR CONVERGENCE
290  W=S(K)
      IF(L.EQ.K)GO TO 360
C      ORIGIN SHIFT
      X=S(L)
      Y=S(K-1)
      G=T(K-1)
      H=T(K)
      F=((Y-H)*(Y+H)+(G-H)*(G+H))/(2.*H*Y)
      G=SQRT(F*F+1.)
      IF(F.LT.0.)G=-G
      F=((X-H)*(X+H)+(Y/(F+G)-H)*H)/X
C      QR STEP
      CS=1.
      SN=1.
      L1=L+1
      DO 350 I=L1,K
      G=T(I)
      Y=S(I)
      H=SN*G
      G=CS*G
      W=SQRT(H*H+F*F)
      T(I-1)=W
      CS=F/W
      SN=H/W
      F=X*CS+G*SN
      G=G*CS-X*SN
      H=Y*SN
      Y=Y*CS
      IF(NV.EQ.0)GO TO 310
      DO 300 J=1,N
      X=REAL(V(J,I-1))
      W=REAL(U(J,I))
      V(J,I-1)=CMPLX(X*CS+W*SN,0.)
300  V(J,I)=CMPLX(W*CS-X*SN,0.)
310  W=SQRT(H*H+F*F)
      S(I-1)=W
      CS=F/W
      SN=H/W
      F=CS*G+SN*Y
      X=CS*Y-SN*G
      IF(NU.EQ.0)GO TO 330
      DO 320 J=1,N
      Y=REAL(U(J,I-1))
      W=REAL(U(J,I))
      U(J,I-1)=CMPLX(Y*CS+W*SN,0.)
320  U(J,I)=CMPLX(W*CS-Y*SN,0.)
330  IF(N.EQ.NP)GO TO 350
      DO 340 J=N1,NP
      Q=A(I-1,J)
      R=A(I,J)
      A(I-1,J)=Q*CS+R*SN
340  A(I,J)=R*CS-Q*SN
350  CONTINUE
      T(L)=0.
      T(K)=F
      S(K)=X
      GO TO 220
C      CONVERGENCE
360  IF(W.GE.0.)GO TO 380
      S(K)=-W

```


IF(NV.EQ.0)GO TO 380
 DO 370 J=1,N
 370 V(J,K)=-V(J,K)
 380 CONTINUE
 C SORT SINGULAR VALUES
 DO 450 K=1,N
 G=-1.
 J=K
 DO 390 I=K,N
 IF(S(I).LE.G)GO TO 390
 G=S(I)
 J=I
 390 CONTINUE
 IF(J.EQ.K)GO TO 450
 S(J)=S(K)
 S(K)=G
 IF(NV.EQ.0)GO TO 410
 DO 400 I=1,N
 Q=V(I,J)
 V(I,J)=V(I,K)
 400 V(I,K)=Q
 410 IF(NU.EQ.0)GO TO 430
 DO 420 I=1,N
 Q=U(I,J)
 U(I,J)=U(I,K)
 420 U(I,K)=Q
 430 IF(N.EQ.NP)GO TO 450
 DO 440 I=N1,NP
 Q=A(J,I)
 A(J,I)=A(K,I)
 440 A(K,I)=Q
 450 CONTINUE
 C BACK TRANSFORMATION
 IF(NU.EQ.0)GO TO 510
 DO 500 KK=1,N
 K=N1-KK
 IF(B(K).EQ.0.)GO TO 500
 Q=-A(K,K)/CABS(A(K,K))
 12 DO 460 J=1,NU
 460 U(K,J)=Q*U(K,J)
 11 DO 490 J=1,NU
 Q=CMPLX(0.0,0.)
 10 DO 470 I=K,M
 470 Q=Q+CONJG(A(I,K))*U(I,J)
 9 Q=Q/(CABS(A(K,K))*B(K))
 8 DO 480 I=K,M
 480 U(I,J)=U(I,J)-Q*A(I,K)
 7 490 CONTINUE
 500 CONTINUE
 6 510 IF(NV.EQ.0)GO TO 570
 IF(N.LT.2)GO TO 570
 5 DO 560 KK=2,N
 K=N1-KK
 4 K1=K+1
 IF(C(K1).EQ.0.)GO TO 560
 Q=-CONJG(A(K,K1))/CABS(A(K,K1))
 DO 520 J=1,NV
 520 V(K1,J)=Q*V(K1,J)
 DO 550 J=1,NV
 Q=CMPLX(0.,0.)
 DO 530 I=K1,N


```
530 Q=Q+A(K,I)*V(I,J)
    Q=Q/(CABS(A(K,K1))*C(K1))
    DO 540 I=K1,N
540 V(I,J)=V(I,J)-Q*CONJG(A(K,I))
550 CONTINUE
560 CONTINUE
570 RETURN
    END
```

MADE IN U.S.A.

12

11

10

9

8

7

6

5

4

3

2

1


```

SUBROUTINE CPOLY(OPR,OPI,DEGREE,ZEROR,ZEROI,FAIL)
C FINDS THE ZEROES OF A COMPLEX POLYNOMIAL
C TAKEN FROM M.A. JENKINS AND J.F. TRAUB, COMM. A.C.M. 15,97-99(1972)
C OPR, OPI--DOUBLE PRECISION VECTORS OF REAL AND IMAGINARY PARTS
C OF THE COEFFICIENTS IN ORDER OF DECREASING POWERS
C DEGREE--INTEGER DEGREE OF THE POLYNOMIAL
C ZEROR, ZEROI--OUTPUT DOUBLE PRECISION VECTORS OF REAL AND IMAGINARY
C PARTS OF THE ZEROES
C FAIL--OUTPUT LOGICAL PARAMETER, TRUE IF LEADING COEFFICIENT IS ZERO
C OR IF CPOLY HAS FOUND FEWER THAN DEGREE ZEROES.
C THIS PROGRAM HAS BEEN WRITTEN TO REDUCE THE CHANCE OF OVERFLOW
C OCCURRING. IF IT DOES OCCUR, THERE IS STILL A POSSIBILITY THAT THE
C ZEROFINDER WILL WORK PROVIDED THE OVERFLOWED QUANTITY IS REPLACED
C BY A LARGE NUMBER.
COMMON/GLOBAL/PR,PI,HR,HI,QPR,QPI,QHR,QHI,SHR,SHI,SR,SI,TR,TI,
SPVR,PVI,ARE,MRE,ETA,INFIN,NN
REAL*8 SR,SI,TR,TI,PVR,PVI,ARE,MRE,ETA,INFIN
REAL*8 PR(150),PI(150),HR(150),HI(150),QPR(150),QPI(150),QHR(150)
REAL*8 QHI(150),SHR(150),SHI(150)
C TO CHANGE THE SIZE OF THE POLYNOMIALS CHANGE THE SIZE OF THE ARRAYS
C IN COMMON STORAGE
REAL*8 XX,YY,COSR,SINR,SMALNO,BASE,XXX,ZR,ZI,BND
REAL*8 OPR(1),OPI(1),ZEROR(1),ZEROI(1),CMOD,SCALE,CAUCHY,DSORT
LOGICAL*4 FAIL,CONV
INTEGER*4 DEGREE,CNT1,CNT2
C INITIALIZATION OF CONSTANTS
CALL MCON(ETA,INFIN,SMALNO,BASE)
ARE=ETA
MRE=2.00*DSORT(2.00)*ETA
XX=.7071067800
YY=-XX
COSR=-.06075647400
SINR=.9975640500
FAIL=.FALSE.
NN=DEGREE+1
C ALGORITHM FAILS IF LEADING COEFF IS ZERO
IF(OPR(1).NE.0.00.OR.OPI(1).NE.0.00)GO TO 10
FAIL=.TRUE.
RETURN
C REMOVE THE ZEROES AT THE ORIGIN, IF ANY
10 IF(OPR(NN).NE.0.00.OR.OPI(NN).NE.0.00)GO TO 20
IDNN2=DEGREE-NN+2
ZEROR(IDNN2)=0.00
ZEROI(IDNN2)=0.00
NN=NN-1
GO TO 10
C MAKE A COPY OF THE COEFFICIENTS
20 DO 30 I=1,NN
PR(I)=OPR(I)
PI(I)=OPI(I)
SHR(I)=CMOD(PR(I),PI(I))
30 CONTINUE
C SCALE THE POLYNOMIAL
BND=SCALE(NN,SHR,ETA,INFIN,SMALNO,BASE)
IF(BND.EQ.1.00)GO TO 40
DO 35 I=1,NN
PR(I)=BND*PR(I)
PI(I)=BND*PI(I)
35 CONTINUE
C START THE ALGORITHM FOR ONE ZERO
40 IF(NN.GT.2)GO TO 50
```



```

C CALCULATE THE FINAL ZERO AND RETURN
  CALL CDIVID(-PR(2),-PI(2),PR(1),PI(1),ZEROR(DEGREE),
  $ ZEROI(DEGREE))
  RETURN
C CALCULATE BND, A LOWER BOUND ON THE MODULUS OF THE ZEROES
  50 DO 60 I=1,NN
    SHR(I)=CMOD(PR(I),PI(I))
  60 CONTINUE
  BND=CAUCHY(NN,SHR,SHI)
C OUTER LOOP TO CONTROL 2 MAJOR PASSES WITH DIFFERENT SEQUENCES
C OF SHIFTS
  DO 100 CNT1=1,2
C FIRST STAGE CALCULATION, NO SHIFT
  CALL NOSHFT(5)
C INNER LOOP TO SELECT A SHIFT
  DO 90 CNT2=1,9
C SHIFT IS CHOSEN WITH MODULUS BND AND AMPLITUDE ROTATED BY 94 DEGREES
C FROM THE PREVIOUS SHIFT
    XXX=CSR*XX-SINR*YY
    YY=SINR*XX+CSR*YY
    XX=XXX
    SR=BND*XX
    SI=BND*YY
C SECOND STAGE CALCULATION, FIXED SHIFT
    CALL FXSHFT(10*CNT2,ZR,ZI,CONV)
    IF(.NOT.CONV)GO TO 80
C THE SECOND STAGE JUMPS DIRECTLY TO THE THIRD STAGE ITERATION
C IF SUCCESSFUL THE ZERO IS STORED AND THE POLYNOMIAL DEFLATED
    IDNN2=DEGREE-NN+2
    ZEROR(IDNN2)=ZR
    ZEROI(IDNN2)=ZI
    NN=NN-1
  70 DO 70 I=1,NN
    PR(I)=QPR(I)
    PI(I)=QPI(I)
  70 CONTINUE
  GO TO 40
  80 CONTINUE
12 C IF THE ITERATION IS UNSUCCESSFUL ANOTHER SHIFT IS CHOSEN
  90 CONTINUE
11 C IF 9 SHIFTS FAIL, THE OUTER LOOP IS REPEATED WITH ANOTHER SEQUENCE
C OF SHIFTS
10 100 CONTINUE
  C THE ZEROFINDER HAS FAILED ON TWO MAJOR PASSES
  C RETURN EMPTY HANDED
  FAIL=.TRUE.
  RETURN
  END

```


SUBROUTINE NOSHFT(L1)

C COMPUTES THE DERIVATIVE POLYNOMIAL AS THE INITIAL H

C POLYNOMIAL AND COMPUTES L1 NO-SHIFT H POLYNOMIALS

COMMON/GLOBAL/ PR,PI,HR,HI,OPR,QPI,QHR,QHI,SHR,SHI,

SSR,SI,TR,TI,PVR,PVI,ARE,MRE,ETA,INFIN,NN

REAL*8 SR,SI,TR,TI,PVR,PVI,ARE,MRE,ETA,INFIN

REAL*8 PR(150),PI(150),HR(150),HI(150),QPR(150),QPI(150)

REAL*8 QHR(150),QHI(150),SHR(150),SHI(150)

REAL*8 XNI,T1,T2,CMOD

N=NN-1

NM1=N-1

DO 10 I=1,N

XNI=NN-I

HR(I)=XNI*PR(I)/FLOAT(N)

HI(I)=XNI*PI(I)/FLOAT(N)

10 CONTINUE

DO 50 JJ=1,L1

IF(CMOD(HR(N),HI(N)).LE.ETA*10.DO*CMOD(PR(N),PI(N)))GO TO 30

CALL CDIVID(-PR(NN),-PI(NN),HR(N),HI(N),TR,TI)

DO 20 I=1,NM1

J=NN-I

T1=HR(J-1)

T2=HI(J-1)

HR(J)=TR*T1-TI*T2+PR(J)

HI(J)=TR*T2+TI*T1+PI(J)

20 CONTINUE

HR(1)=PR(1)

HI(1)=PI(1)

GO TO 50

C IF THE CONSTANT TERM IS ESSENTIALLY ZERO, SHIFT H COEFFICIENTS

30 DO 40 I=1,NM1

J=NN-I

HR(J)=HR(J-1)

HI(J)=HI(J-1)

40 CONTINUE

HR(1)=0.00

HI(1)=0.00

50 CONTINUE

RETURN

END


```

SUBROUTINE FXSHFT(L2,ZR,ZI,CONV)
C COMPUTES L2 FIXED-SHIFT H POLYNOMIALS AND TESTS FOR CONVERGENCE
C INITIATES A VARIABLE-SHIFT ITERATION AND RETURNS WITH THE
C APPROXIMATE ZERO IF SUCCESSFUL
C L2--LIMIT OF FIXED SHIFT STEPS
C ZR, ZI--APPROXIMATE ZERO IF CONV IS .TRUE.
C CONV--LOGICAL INDICATING CONVERGENCE OF STAGE 3 ITERATION
COMMON/GLOBAL/PR,PI,HR,HI,QPR,QPI,QHR,QHI,SHR,SHI,SR,SI,
STR,SI,TR,PI,PVR,PVI,ARE,MRE,ETA,INFIN,NN
REAL*8 SR,SI,TR,PI,PVR,PVI,ARE,MRE,ETA,INFIN
REAL*8 PR(150),PI(150),HR(150),HI(150),QPR(150),QPI(150)
REAL*8 QHR(150),QHI(150),SHR(150),SHI(150)
REAL*8 ZR,ZI,OTR,OTI,SVSR,SVSI,CMOD
LOGICAL*4 CONV,TEST,PASD,BOOL
N=NN-1
C EVALUATE P AT S
CALL POLYEV(NN,SR,SI,PR,PI,QPR,QPI,PVR,PVI)
TEST=.TRUE.
PASD=.FALSE.
C CALCULATE FIRST T=-P(S)/H(S)
CALL CALCT(BOOL)
C MAIN LOOP FOR ONE SECOND STAGE STEP
DO 50 J=1,L2
    OTR=TR
    OTI=TI
C COMPUTE NEXT H POLYNOMIAL AND NEW T
    CALL NEXTH(BOOL)
    CALL CALCT(BOOL)
    ZR=SR+TR
    ZI=SI+TI
C TEST FOR CONVERGENCE UNLESS STAGE 3 HAS FAILED ONCE OR THIS
C IS THE LAST H POLYNOMIAL
    IF(BOOL.OR..NOT.TEST.OR.J.EQ.L2)GO TO 50
    IF(CMOD(TR-OTR,PI-OTI).GE..500*CMOD(ZR,ZI))GO TO 40
    IF(.NOT.PASD)GO TO 30
C THE WEAK CONVERGENCE TEST HAS BEEN PASSED TWICE, START THE
C THIRD STAGE ITERATION, AFTER SAVING THE CURRENT H POLYNOMIAL
C AND SHIFT
    DO 10 I=1,N
        SHR(I)=HR(I)
        SHI(I)=HI(I)
    CONTINUE
    SVSR=SR
    SVSI=SI
    CALL VRSHFT(10,ZR,ZI,CONV)
    IF(CONV)RETURN
C THE ITERATION FAILED TO CONVERGE. TURN OFF TESTING AND RESTORE
C H,S,PV, AND T
    TEST=.FALSE.
    DO 20 I=1,N
        HR(I)=SHR(I)
        HI(I)=SHI(I)
    CONTINUE
    SR=SVSR
    SI=SVSI
    CALL POLYEV(NN,SR,SI,PR,PI,QPR,QPI,PVR,PVI)
    CALL CALCT(BOOL)
    GO TO 50
30    PASD=.TRUE.
    GO TO 50
40    PASD=.FALSE.

```


50 CONTINUE
C ATTEMPT AN ITERATION WITH FINAL H POLYNOMIAL FROM SECOND STAGE
CALL YRSHFT(10,ZR,ZI,CONV)
RETURN
END

MADE IN U.S.A.

12

11

10

9

8

7

6

5

4

3

2


```

SUBROUTINE VRSHT(L3,ZR,ZI,CONV)
C CARRIES OUT THE THIRD STAGE ITERATION
C L3--LIMIT OF STEPS IN STAGE 3
C ZR, ZI--ON ENTRY CONTAINS THE INITIAL ITERATE, IF THE
C ITERATION CONVERGES IT CONTAINS THE FINAL ITERATE ON EXIT
C CONV--.TRUE. IF ITERATION CONVERGES
COMMON/GLOBAL/PR,PI,HR,HI,QPR,QPI,QHR,QHI,SHR,SHI,SR,SI,
$TR,TI,PVR,PVI,ARE,MRE,ETA,INFIN,NN
REAL*8 SR,SI,TR,TI,PVR,PVI,ARE,MRE,ETA,INFIN
REAL*8 PR(150),PI(150),HR(150),HI(150),QPR(150),QPI(150)
REAL*8 QHR(150),QHI(150),SHR(150),SHI(150)
REAL*8 ZR,ZI,MP,MS,DMP,RELSTP,R1,R2,CMOD,DSQRT,ERREV,TP
LOGICAL*4 CONV,B,BOOL
CONV=.FALSE.
B=.FALSE.
SR=ZR
SI=ZI
C MAIN LOOP FOR STAGE THREE
DO 60 I=1,L3
C EVALUATE P AT S AND TEST FOR CONVERGENCE
CALL POLYEV(NN,SR,SI,PR,PI,QPR,QPI,PVR,PVI)
MP=CMOD(PVR,PVI)
MS=CMOD(SR,SI)
IF(MP.GT.20.DO*ERREV(NN,QPR,QPI,MS,MP,ARE,MRE))GO TO 10
C POLYNOMIAL VALUE IS SMALLER IN VALUE THAN A BOUND ON THE ERROR
C IN EVALUATING P, TERMINATE THE ITERATION
CONV=.TRUE.
ZR=SR
ZI=SI
RETURN
10 IF(I.EQ.1)GO TO 40
IF(B.OR.MP.LT.OMP.OR.RELSTP.GE..05DO)GO TO 30
C ITERATION HAS STALLED. PROBABLY A CLUSTER OF ZEROES. DO 5 FIXED
C SHIFT STEPS INTO THE CLUSTER TO FORCE ONE ZERO TO DOMINATE.
TP=RELSTP
B=.TRUE.
IF(RELSTP.LT.ETA)TP=ETA
R1=DSQRT(TP)
12 R2=SR*(1.DO+R1)-SI*R1
11 SI=SR*R1+SI*(1.DO+R1)
SR=R2
CALL POLYEV(NN,SR,SI,PR,PI,QPR,QPI,PVR,PVI)
10 DO 20 J=1,5
9 CALL CALCT(BOOL)
CALL NEXTH(BOOL)
20 CONTINUE
8 OMP=INFIN
7 GO TO 50
C EXIT IF THE POLYNOMIAL VALUE INCREASES SIGNIFICANTLY
30 IF(MP*.1DO.GT.OMP)RETURN
40 OMP=MP
C CALCULATE NEXT ITERATE
50 CALL CALCT(BOOL)
CALL NEXTH(BOOL)
CALL CALCT(BOOL)
IF(BOOL)GO TO 60
RELSTP=CMOD(TR,TI)/CMOD(SR,SI)
SR=SR+TR
SI=SI+TI
60 CONTINUE
DETIDN

```


SUBROUTINE CALCT(BOOL)

C COMPUTES $T = -P(S)/H(S)$

C BOOL--LOGICAL, SET TRUE IF H(S) IS ESSENTIALLY ZERO

COMMON/GLOBAL/PR,PI,HR,HI,QPR,QPI,QHR,QHI,SHR,SHI,SR,SI,
\$TR, TI, PVR, PVI, ARE, MRE, ETA, INFIN, NN

REAL*8 SR, SI, TR, TI, PVR, PVI, ARE, MRE, ETA, INFIN

REAL*8 PR(150), PI(150), HR(150), HI(150), QPR(150), QPI(150)

REAL*8 QHR(150), QHI(150), SHR(150), SHI(150)

REAL*8 HVR, HVI, CMOD

LOGICAL*4 BOOL

N=NN-1

C EVALUATE H(S)

CALL POLYEV(N, SR, SI, HR, HI, QHR, QHI, HVR, HVI)

BOOL=CMOD(HVR, HVI).LE.ARE*10.DO*CMOD(HR(N), HI(N))

IF(BOOL)GO TO 10

CALL CDIVIDI(-PVR, -PVI, HVR, HVI, TR, TI)

RETURN

10 TR=0.00

TI=0.00

RETURN

END

MADE IN U.S.A.

12

11

10

9

8

7

6

5

4

3

2


```

SUBROUTINE NEXTH(BOOL)
C CALCULATES THE NEXT SHIFTED H POLYNOMIAL
C BOOL LOGICAL, IF .TRUE. H(S) IS ESSENTIALLY ZERO
COMMON/GLOBAL/PR,PI,HR,HI,QPR,QPI,QHR,QHI,SHR,SHI,SR,SI,
$TR, TI, PVR, PVI, ARE, MRE, ETA, INFIN, NN
REAL*8 SR, SI, TR, TI, PVR, PVI, ARE, MRE, ETA, INFIN
REAL*8 PR(150), PI(150), HR(150), HI(150), QPR(150), QPI(150)
REAL*8 QHR(150), QHI(150), SHR(150), SHI(150), T1, T2
LOGICAL*4 BOOL
N=NN-1
IF(BOOL)GO TO 20
DO 10 J=2,N
    T1=QHR(J-1)
    T2=QHI(J-1)
    HR(J)=TR*T1-TI*T2+QPR(J)
    HI(J)=TR*T2+TI*T1+QPI(J)
10 CONTINUE
    HR(1)=QPR(1)
    HI(1)=QPI(1)
    RETURN
C IF H(S) IS ZERO REPLACE H WITH QH
20 DO 30 J=2,N
    HR(J)=QHR(J-1)
    HI(J)=QHI(J-1)
30 CONTINUE
    HR(1)=0.00
    HI(1)=0.00
    RETURN
END

```

MADE IN U.S.A.

12

11

10

9

8

7

6

5

4


```
SUBROUTINE POLYEV(NN,SR,SI,PR,PI,QR,QI,PVR,PVI)
C EVALUATES A POLYNOMIAL P AT S BY THE HORNER RECURRENCE
C PLACING THE PARTIAL SUMS IN Q AND THE COMPUTED VALUE IN PV
REAL*8 PR(NN),PI(NN),QR(NN),QI(NN),SR,SI,PVR,PVI,T
QR(1)=PR(1)
QI(1)=PI(1)
PVR=QR(1)
PVI=QI(1)
DO 10 I=2,NN
    T=PVR*SR-PVI*SI+PR(I)
    PVI=PVR*SI+PVI*SR+PI(I)
    PVR=T
    QR(I)=PVR
    QI(I)=PVI
10 CONTINUE
RETURN
END
```

MADE IN U.S.A.

12

11

10

9

8

7

6

5

4

3

2


```
REAL*8 FUNCTION ERREV(NN,QR,QI,MS,MP,ARE,MRE)
C BOUNDS THE ERROR IN EVALUATING THE POLYNOMIAL BY THE HORNER RECURRENC
C QR,QI--THE PARTIAL SUMS
C MS--MODULUS OF THE POINT
C MP--MODULUS OF THE POLYNOMIAL VALUE
C ARE,MRE--ERROR BOUND ON COMPLEX ADDITION AND MULTIPLICATION
REAL*8 QR(NN),QI(NN),MS,MP,ARE,MRE,E,CMOD
E=CMOD(QR(1),QI(1))*MRE/(ARE+MRE)
DO 10 I=1,NN
    E=E*MS+CMOD(QR(I),QI(I))
10 CONTINUE
ERREV=E*(ARE+MRE)-MP*MRE
RETURN
END
```

12

11

10

9

8

7

6

5


```

REAL*8 FUNCTION CAUCHY(NN,PT,Q)
C CAUCHY COMPUTES A LOWER BOUND ON THE MODULI OF THE ZEROES OF A
C POLYNOMIAL--PT IS THE MODULUS OF THE COEFFICIENTS
REAL*8 Q(NN),PT(NN),X,XM,F,DX,DF,DABS,DEXP,DLOG
PT(NN)=-PT(NN)
C COMPUTE UPPER ESTIMATE OF BOUND
N=NN-1
X=DEXP((DLOG(-PT(NN))-DLOG(PT(1)))/FLOAT(N))
IF(PT(N).EQ.0.00)GO TO 20
C IF NEWTON STEP AT THE ORIGIN IS BETTER, USE IT
XM=-PT(NN)/PT(N)
IF(XM.LT.X)X=XM
C CHOP THE INTERVAL (0,X) UNTIL F<=0
20 XM=X*.100
F=PT(1)
DO 30 I=2,NN
F=F*XM+PT(I)
30 CONTINUE
IF(F.LE.0.00)GO TO 40
X=XM
GO TO 20
40 DX=X
C DO NEWTON ITERATION UNTIL X CONVERGES TO TWO DECIMAL PLACES
50 IF(DABS(DX/X).LE..00500)GO TO 70
Q(1)=PT(1)
DO 60 I=2,NN
Q(I)=Q(I-1)*X+PT(I)
60 CONTINUE
F=Q(NN)
DF=Q(1)
DO 65 I=2,N
DF=DF*X+Q(I)
65 CONTINUE
DX=F/DF
X=X-DX
GO TO 50
70 CAUCHY=X
RETURN
END

```



```

      REAL*8 FUNCTION SCALE(NN,PT,ETA,INFIN,SMALNO,BASE)
C RETURNS A SCALE FACTOR TO MULTIPLY THE COEFFICIENTS OF THE
C POLYNOMIAL. THE SCALING IS DONE TO AVOID OVERFLOW AND TO
C AVOID UNDETECTED UNDERFLOW INTERFERING WITH THE CONVERGENCE
C CRITERION. THE FACTOR IS A POWER OF THE BASE.
C PT--MODULUS OF THE COEFFICIENTS OF P
C ETA,INFIN,SMALNO,BASE,--CONSTANTS DESCRIBING THE
C FLOATING POINT ARITHMETIC.

```

```

      REAL*8 PT(NN),ETA,INFIN,SMALNO,BASE,HI,LO,MAX,MIN,X,SC
      REAL*8 DSQRT,DLOG

```

```

C FIND LARGEST AND SMALLEST MODULI OF COEFFICIENTS

```

```

      HI=DSQRT(INFIN)

```

```

      LO=SMALNO/ETA

```

```

      MAX=0.00

```

```

      MIN=INFIN

```

```

      DO 10 I=1,NN

```

```

         X=PT(I)

```

```

         IF(X.GT.MAX)MAX=X

```

```

         IF(X.NE.0.00.AND.X.LT.MIN)MIN=X

```

```

      10  CONTINUE

```

```

C SCALE ONLY IF THERE ARE VERY LARGE OR VERY SMALL COMPONENTS.

```

```

      SCALE=1.00

```

```

      IF(MIN.GE.LO.AND.MAX.LE.HI)RETURN

```

```

      X=LO/MIN

```

```

      IF(X.GT.1.00)GO TO 20

```

```

         SC=1.00/(DSQRT(MAX)*DSQRT(MIN))

```

```

         GO TO 30

```

```

      20 SC=X

```

```

         IF(INFIN/SC.LT.MAX)SC=1.00

```

```

      30 L=DLOG(SC)/DLOG(BASE)+.5

```

```

      SCALE=BASE**L

```

```

      RETURN

```

```

      END

```

12

11

10

9

8

7

6

5

4


```
SUBROUTINE CDIVID(AR,AI,BR,BI,CR,CI)
C COMPLEX DIVISION C=A/B, AVOIDING OVERFLOW.
REAL*8 AR,AI,BR,BI,CR,CI,R,D,T,INFIN,DABS
IF(BR.NE.0.DO.OR.BI.NE.0.DO)GO TO 10
C DIVISION BY ZERO, C=INFINITY
CALL MCON(T,INFIN,T,T)
CR=INFIN
CI=INFIN
RETURN
10 IF(DABS(BR).GE.DABS(BI))GO TO 20
R=BR/BI
D=BI+R*BR
CR=(AR*R+AI)/D
CI=(AI*R-AR)/D
RETURN
20 R=BI/BR
D=BR+R*BI
CR=(AR+AI*R)/D
CI=(AI-AR*R)/D
RETURN
END
```

MADE IN U.S.A.

12

11

10

9

8

7

6

5

4

3

2

1

E

```
      REAL*8 FUNCTION CMOD(R,I)
      C MODULUS OF A COMPLEX NUMBER AVOIDING OVERFLOW
      REAL*8 R,I,AR,AI,DABS,DSQRT
      AR=DABS(R)
      AI=DABS(I)
      IF (AR.GE.AI) GO TO 10
      CMOD=AI*DSQRT(1.00+(AR/AI)**2)
      RETURN
10  IF (AR.LE.AI) GO TO 20
      CMOD=AR*DSQRT(1.00+(AI/AR)**2)
      RETURN
20  CMOD=AR*DSQRT(2.00)
      RETURN
      END
```


SUBROUTINE MCON(ETA,INFINY,SMALNO,BASE)

C MCON PROVIDES MACHINE CONSTANTS USED IN VARIOUS PARTS OF THE PROGRAM
 C THE USER MAY EITHER SET THEM DIRECTLY OR USE THE STATEMENTS BELOW
 C TO COMPUTE THEM. THE MEANING OF THE FOUR CONSTANTS ARE--
 C ETA--MAXIMUM RELATIVE REPRESENTATION ERROR WHICH CAN BE DESCRIBED
 C AS THE SMALLEST POSITIVE FLOATING POINT NUMBER SUCH THAT $1.00 + \text{ETA}$
 C IS GREATER THAN 1.00
 C INFINY--THE LARGEST FLOATING POINT NUMBER
 C SMALNO--THE SMALLEST POSITIVE FLOATING POINT NUMBER
 C BASE-- THE BASE OF THE FLOATING POINT NUMBER SYSTEM USED
 C LET T BE THE NUMBER OF BASE DIGITS IN EACH FLOATING POINT NUMBER
 C (DOUBLE PRECISION). THEN ETA IS EITHER $.5 * \text{BASE}^{-(1-T)}$ OR $\text{BASE}^{-(1-T)}$
 C DEPENDING ON WHETHER ROUNDING OR TRUNCATION IS USED.
 C LET M BE THE LARGEST EXPONENT AND N THE SMALLEST EXPONENT IN THE
 C NUMBER SYSTEM. THEN INFINY IS $(1 - \text{BASE}^{-(1-T)}) * \text{BASE}^M$ AND SMALNO
 C IS BASE^N .
 C THE VALUES FOR BASE,T,M,N BELOW CORRESPOND TO VAX 11/780
 C A. CHAVE WHOI DEC 1979

REAL*8 ETA,INFINY,SMALNO,BASE

INTEGER*4 M,N,T

BASE=16.00

T=10

M=31

N=-32

ETA=BASE**((1-T))

INFINY=BASE*(1.00-BASE**((-T)))*BASE**(M-1)

SMALNO=(BASE**(N+3))/BASE**3

RETURN

END

MADE IN U.S.A.

12

11

10

9

8

7

6

5

4

3

2

APPENDIX VI: COMPATIBILITY

All of the programs contained in this report, with the exception of CPOLY and CSVD, were written in VAX-11 Fortran IV-Plus for use in a VAX/VMS operating environment on the Digital Equipment Corporation VAX 11/780 computer. Some of the statements used are not implemented on older machines and earlier versions of Fortran. While no attempt was made to provide portable coding, changes in only a few statements will result in a compatible set of routines. These include:

1. The PARAMETER statement is used to initialize constants at compile time. Replace this statement with either an explicit run time assignment or a DATA statement.
2. The block IF or IF-THEN-ELSE statement is used extensively. Replace this with a series of logical IF statements and numbered program blocks.
3. Generic function names are used and no attempt to maintain variable type was made. Replace with the appropriate library function.
4. All angles in library functions are expressed in radians. Care should be taken to ensure that other computers use this convention.

REFERENCES

- Bath, M., 1974. Spectral Analysis in Geophysics, New York: Elsevier, 563 pp.
- Burg, J.P., 1975. Maximum Entropy Spectral Analysis, Ph.D. Thesis, Stanford University, 123 pp.
- Businger, P.A. and G.H. Golub, 1969. Singular value decomposition of a complex matrix, Comm. A. Com. Mach. 12, 564-565.
- Chave, A.D. and C.R. Denham, 1980. Geomagnetic secular variation recorded in sediments from the northeast Atlantic Ocean, submitted to J. geophys. Res.
- Chave, A.D., R.P. Von Herzen, K.A. Poehls, and C.S. Cox, 1980. Electromagnetic induction fields in the deep ocean northeast of Hawaii and implications for mantle conductivity, submitted to Geophys. J. Roy. astr. Soc.
- De Boor, C., 1978. A Practical Guide to Splines, New York: Springer Verlag, 392 pp.
- Denham, C.R., 1977. PATSY: A Time-Series Processor, Technical Report, WHOI Ref. 77-17, Unpublished Manuscript.
- Haykin, S., 1979. Nonlinear Methods of Spectral Analysis, New York: Springer Verlag, 247 pp.
- Jenkins, M.A. and J.F. Traub, 1972. Zeroes of a complex polynomial, Comm. A. Com. Mach. 15, 97-99.
- Jupp, D.L.B., 1978. Estimation of the magnetotelluric impedance functions, Phys. E. Pl. Int. 17, 75-82.
- Jupp, D.L.B. and K. Vozoff, 1975. Stable iterative methods for the inversion of geophysical data, Geophys. J. Roy. astr. Soc. 42, 957-976.
- Lawson, C.L. and R.J. Hanson, 1974. Solving Least Squares Problems, New Jersey: Prentice-Hall, 340 pp.
- Otnes, R.K. and L. Enochson, 1972. Digital Time Series Analysis, New York: John Wiley, 467 pp.
- Ulrych, T.J. and T.N. Bishop, 1975. Maximum entropy spectral analysis and autoregressive decomposition, Rev. Geophys. 13, 183-200.

REPORT DOCUMENTATION PAGE		1. REPORT NO. WHOI-80-23	2.	3. Recipient's Accession No.
4. Title and Subtitle TIME SERIES ANALYSIS PROGRAMS				5. Report Date May 1980
7. Author(s) Alan D. Chave				6.
9. Performing Organization Name and Address Woods Hole Oceanographic Institution Woods Hole, MA 02543				8. Performing Organization Rept. No.
12. Sponsoring Organization Name and Address National Science Foundation				10. Project/Task/Work Unit No.
15. Supplementary Notes				11. Contract(C) or Grant(G) No. (C) OCE77-8633, OCE77-82255 (G) OCE79-19258
16. Abstract (Limit: 200 words) This report documents a set of Fortran subroutines for manipulating one dimensional, complex-valued time series information. The programs are divided into two libraries: TSLIB for ordinary Fourier analysis and ACLIB for more specialized applications. The latter includes interpolation routines, maximum entropy spectral analysis programs, and subroutines for advanced mathematical operations. The programs are designed for efficiency and speed with a minimum of hidden operations and memory accesses. These subroutines have been used extensively in the analysis of magnetotelluric and other geophysical data.				13. Type of Report & Period Covered Technical
17. Document Analysis a. Descriptors 1. Time series analysis 2. Fourier analysis 3. Computer programs b. Identifiers/Open-Ended Terms c. COSATI Field/Group				14.
18. Availability Statement		19. Security Class (This Report)		21. No. of Pages 53
		20. Security Class (This Page)		22. Price

Woods Hole Oceanographic Institution WHOI-80-23	TIME SERIES ANALYSIS PROGRAMS by Alan D. Chave. May 1980. 53 pages. Prepared for the National Science Foundation under Grants OCE77-8633, OCE77-82255, and OCE79-19258, and for the WHOI Ocean Industries Program.	1. Time series analysis 2. Fourier analysis 3. Computer programs I. Chave, Alan D. II. OCE77-8633 III. OCE77-82255 IV. OCE79-19258	Woods Hole Oceanographic Institution WHOI-80-23	TIME SERIES ANALYSIS PROGRAMS by Alan D. Chave. May 1980. 53 pages. Prepared for the National Science Foundation under Grants OCE77-8633, OCE77-82255, and OCE79-19258, and for the WHOI Ocean Industries Program.	1. Time series analysis 2. Fourier analysis 3. Computer programs I. Chave, Alan D. II. OCE77-8633 III. OCE77-82255 IV. OCE79-19258	1. Time series analysis 2. Fourier analysis 3. Computer programs I. Chave, Alan D. II. OCE77-8633 III. OCE77-82255 IV. OCE79-19258
	This report documents a set of Fortran subroutines for manipulating one dimensional, complex-valued time series information. The programs are divided into two libraries: TSLIB for ordinary Fourier analysis and ACLIB for more specialized applications. The latter includes interpolation routines, maximum entropy spectral analysis programs, and subroutines for advanced mathematical operations. The programs are designed for efficiency and speed with a minimum of hidden operations and memory accesses. These subroutines have been used extensively in the analysis of magnetotelluric and other geophysical data.	This card is UNCLASSIFIED		This report documents a set of Fortran subroutines for manipulating one dimensional, complex-valued time series information. The programs are divided into two libraries: TSLIB for ordinary Fourier analysis and ACLIB for more specialized applications. The latter includes interpolation routines, maximum entropy spectral analysis programs, and subroutines for advanced mathematical operations. The programs are designed for efficiency and speed with a minimum of hidden operations and memory accesses. These subroutines have been used extensively in the analysis of magnetotelluric and other geophysical data.	This card is UNCLASSIFIED	This card is UNCLASSIFIED
Woods Hole Oceanographic Institution WHOI-80-23	TIME SERIES ANALYSIS PROGRAMS by Alan D. Chave. May 1980. 53 pages. Prepared for the National Science Foundation under Grants OCE77-8633, OCE77-82255, and OCE79-19258, and for the WHOI Ocean Industries Program.	1. Time series analysis 2. Fourier analysis 3. Computer programs I. Chave, Alan D. II. OCE77-8633 III. OCE77-82255 IV. OCE79-19258	Woods Hole Oceanographic Institution WHOI-80-23	TIME SERIES ANALYSIS PROGRAMS by Alan D. Chave. May 1980. 53 pages. Prepared for the National Science Foundation under Grants OCE77-8633, OCE77-82255, and OCE79-19258, and for the WHOI Ocean Industries Program.	1. Time series analysis 2. Fourier analysis 3. Computer programs I. Chave, Alan D. II. OCE77-8633 III. OCE77-82255 IV. OCE79-19258	1. Time series analysis 2. Fourier analysis 3. Computer programs I. Chave, Alan D. II. OCE77-8633 III. OCE77-82255 IV. OCE79-19258
	This report documents a set of Fortran subroutines for manipulating one dimensional, complex-valued time series information. The programs are divided into two libraries: TSLIB for ordinary Fourier analysis and ACLIB for more specialized applications. The latter includes interpolation routines, maximum entropy spectral analysis programs, and subroutines for advanced mathematical operations. The programs are designed for efficiency and speed with a minimum of hidden operations and memory accesses. These subroutines have been used extensively in the analysis of magnetotelluric and other geophysical data.	This card is UNCLASSIFIED		This report documents a set of Fortran subroutines for manipulating one dimensional, complex-valued time series information. The programs are divided into two libraries: TSLIB for ordinary Fourier analysis and ACLIB for more specialized applications. The latter includes interpolation routines, maximum entropy spectral analysis programs, and subroutines for advanced mathematical operations. The programs are designed for efficiency and speed with a minimum of hidden operations and memory accesses. These subroutines have been used extensively in the analysis of magnetotelluric and other geophysical data.	This card is UNCLASSIFIED	This card is UNCLASSIFIED