LEAST-SQUARES SPECTRAL ANALYSIS REVISITED

D. WELLS
P. VANICEK
S. PAGIATAKIS

November 1985

TECHNICAL REPORT
NO. 84

PREFACE

In order to make our extensive series of technical reports more readily available, we have scanned the old master copies and produced electronic versions in Portable Document Format. The quality of the images varies depending on the quality of the originals. The images have not been converted to searchable text.

LEAST SQUARES SPECTRAL ANALYSIS REVISITED

David E. Wells
Petr Vaníček
Spiros Pagiatakis

Department of Geodesy and Geomatics Engineering
University of New Brunswick
P.O. Box 4400
Fredericton, N.B.
Canada
E3B 5A3

November 1985
Latest Reprinting August 1994

PREFACE

The original version of this report appeared in 1978 under the Report Series (Report BI-R-78-8) of the Bedford Institute of Oceanography, Dartmouth, Nova Scotia, when the first author was working there. It was authored by the first two authors of the present version. The third author has been mainly responsible for the changes and improvements in the present version of the software.

This revised version is being issued for three reasons:

(i) The original report is out of print.

(ii) Students not exposed to functional analysis have had some difficulty in reading the report, so that the original report has been extended to include a more elementary description of least-squares spectral analysis.

(iii) Since 1978 the software has been modified, both to eliminate some "bugs" and to be more versatile.

The changes, program listings of the new version, and a user's guide are in PART B.

The authors are grateful to the Bedford Institute of Oceanography for permission to reprint herewith parts of the original Report BI-R-78-8.

TABLE OF CONTENTS

PART A

LEAST-SQUARES SPECTRAL ANALYSIS

ABSTRACT

An algorithm is described to compute the optimum least-squares spectrum of an unequally or equally spaced generally non-stationary and coloured time series for which some of the shapes of the constituents (systematic noise) are known. Known constituents of four kinds are provided for: datum biases, linear trend, periodic constituents with known periods, and arbitrary user-specified constituents. An alternative, more efficient algorithm is described for piecewise equally-spaced time series with possible gaps.

INTRODUCTION

Observed time series are often composed of constituents that are of interest (which we will call the signal), and constituents that obscure the signal (which we will call the noise). Often we know the general form of a noise constituent, but not its magnitude (we will call such constituents systematic noise).

One class of such time series is that dominated by a few periodic constituents (the systematic noise) and smaller amplitude periodic constituents (the signal) whose presence is obscured by the noise--coloured series. Extracting the signal from this series has been called the hidden periodicity problem. In general, the systematic noise may contaminate but not totally obscure the signal.

If all the constituents of a time series are periodic, then we can say that it is a colour time series; that the noise is coloured noise; that the signal is a coloured signal; and that the extent to which the noise obscures the signal is the extent to which the noise discolours the signal.

A property of a time series is the degree to which it is stationary. Strict sense (or strong) stationarity requires that all statistical properties of the time series (mean value, autocorrelation function, and all higher order moments) be independent of the choice of the time origin. Wide sense (or weak) stationarity requires that only the mean value and autocorrelation function be independent of the choice of the time origin [Bendat and Piersol, 1971]. One common violation of stationarity is the presence of datum shifts in the time series: that is, the mean value is shifted from time to time. Another common violation is the presence of a trend (perhaps linear) in the time series: the "mean" value changes linearly with time.

Both these kinds of non-stationarity-inducing constituents (datum shifts and trends) can also be considered as systematic noise, as long as we know their general form, that is, the time at which datum shifts occurred, and the kind of trend (linear, quadratic, exponential, etc.).

Another property of a time series is its spacing. Are all the data values equally spaced in time? Are there data gaps between otherwise equally-spaced segments of the series? Or are all the values in the time

series unequally spaced in time? Ideally we would like to have only equally spaced time series, but this is rarely the case in practice.

A specific example of this situation is a time series of ocean tide gauge records, from which we are interested in the long period (> 1 year) constituents. Three kinds of systematic noise may contaminate this time series from our point of view. Firstly, there may be step functions due to sudden changes in the tide gauge datum (caused by alterations to the tide gauge or to its supporting structure, or possibly caused by vertical co-seismic displacements). The dates of such step functions are usually well documented, but it can be difficult to document their magnitudes. Secondly, there may be a gradual change in the tide gauge datum (due, for example, to changes in the mean sea level, or to land subsidence including the gauge), which is most simply modelled as a linear trend. Thirdly, the tide gauge time series are dominated by short (in our context) periodic constituents (i.e., tidal constituents) for which the periods are precisely known, but the magnitudes are not. In addition to these problems, there is almost certain to be data gaps due to equipment failures. It is this kind of series that we will analyse at the end of PART A as an example.

To obtain an undistorted spectral image of the signal, we must somehow remove the influence of the systematic noise, both the "colours" and the non-stationarity. The usual way of dealing with this problem is to first find the magnitudes of the components of the noise, subtract the noise from the time series, and perform a spectral analysis on the "corrected" time series. It is known, however, that such a treatment affects the location of spectral peaks arising from the rest of the time series [Taylor and Hamilton, 1972]. We must somehow deal with the data gaps as well. When the data is piecewise equally spaced, as in the example given at the end of PART A, the two usual options are to treat each piece separately, or to somehow manufacture data to fill in the gaps. Neither is satisfactory. The problem is even more difficult when the time series is completely unequally spaced, rather than merely gappy.

An alternative is the least-squares spectral analysis [Vaníček, 1971]. This alternative provides two advantages: systematic noise, including both colours and non-stationarity, can be rigorously accounted for (suppressed) without producing any shift of the existing spectral peaks [Taylor and Hamilton, 1972]; and time series with unequally spaced data can

be analysed [Maul and Yanaway, 1978].

The purpose of this report is to present a brief exposition of the method, for unequally spaced time series, and to describe how the spectrum computation can be made much faster if the time series is equally, or at least piecewise equally, spaced.

Capitalized parameter names longer than one letter refer to identifiers used in FORTRAN subroutines SPECUN and SPECEQ which respectively compute the least-squares spectrum for unequally spaced and equally spaced time series (see PART B for program listings).

SPECTRUM COMPUTATION

There are many definitions of a spectrum, and many ways of computing a spectrum from a time series. Here we simply state the problem in a general way: Given

(a) $\underline{t} = \{t_i\}$, i=1,2,...,n, a vector of observation times,

(b) $\underline{f}(\underline{t}) = \{f_i\} = \{f(t_i)\}$, a vector of observed values,

(c) $\underline{\omega} = \{\omega_j\}$, j=1,2,...,m, a vector of frequencies for which spectral values are desired,

then find

$\underline{s}(\underline{\omega}) = \{s_j\} = \{s(\omega_j)\}$, a vector of spectral values.

Note that

(a) $\underline{f}(\underline{t})$ or $\{f_i, t_i\}$ together define a time series.

(b) $s(\omega_j)$ must be some measure of the fractional content of $\underline{f}(\underline{t})$ which is represented by the frequency ω_j.

Here we consider only one specific technique for computing $s(\omega_j)$, which is the Least-Squares Spectral Analysis (LSSA). This technique is an application of Least-Squares Approximation (LSA) [Vaníček and Wells, 1972], which is closely related to the Linear Least-Squares Parametric Adjustment (LLSPA) [Wells and Krakiwsky, 1971; Vaníček and Krakiwsky, 1982].

LSA and LLSPA use the same algorithm, but they have different purposes, and different interpretations of the quantities involved. Those parts of the algorithm which we are interested in are, in our notation,

$$\underline{\hat{c}} = (\underline{\Phi}^T \underline{W} \underline{\Phi})^{-1} \underline{\Phi}^T \underline{W} \underline{f} \qquad (1)$$

$$\underline{\hat{v}} = \underline{f} - \underline{\Phi} \underline{\hat{c}} \quad . \qquad (2)$$

For LLSPA, we are given

$\underline{f}$ = a vector of observations

$\underline{\Phi}$ = the design matrix which models the physical relationship between the observations $\underline{f}$ and the vector of unknown parameters $\underline{c}$ via the observation equation $\underline{f} = \underline{\Phi}\,\underline{c}$

$\underline{W} = \underline{C}_f^{-1}$ where $\underline{C}_f$ is the covariance matrix of $\underline{f}$, a statistical quantity,

and the problem solved in part by (1) and (2) is to obtain an estimate for some physical parameters $\underline{c}$, based on the observations $\underline{f}$. What we want here is $\underline{\hat{c}}$ (plus its covariance matrix).

For LSA we are given

$\underline{f}$ = a known vector to be approximated, not necessarily based on observations,

$\underline{\Phi}$ = a matrix considered to consist of several column vectors $\underline{\Phi} = [\underline{\phi}_1, \underline{\phi}_2, \ldots, \underline{\phi}_m]$ called base functions, each of which is a known function of the same dimension as $\underline{f}$. Note that $\underline{\Phi}$ does not necessarily model any physical dependence of $\underline{f}$.

$\underline{W}$ = a weight function with no statistical meaning. Here we assume $\underline{W} = \underline{I}$ (often the case in LSA).

For LSA it is usual to rewrite $\underline{f} = \underline{\Phi}\,\underline{c}$ in the form

$$\underline{f} = \sum_{i=1}^{m} c_i \underline{\phi}_i \qquad (3)$$

and to state the LSA problem as finding the best fitting approximant $\underline{p}$ to $\underline{f}$, that is

$$\underline{p} = \sum_{i=1}^{m} \hat{c}_i \underline{\phi}_i \qquad (4)$$

such that the residuals $\underline{\hat{v}} = \underline{f} - \underline{p}$ are minimized in the least-squares sense. Note that for LSA we are more interested in $\underline{p}$ than in the coefficients $\underline{\hat{c}}$, although $\underline{\hat{c}}$ is still given by (1).

Specifically for LSSA we know $\underline{f}(t)$ and we use

$$\phi_1 = \cos \omega_j t$$

$$\phi_2 = \sin \omega_j t \quad . \tag{5}$$

For each ω_j for which we want $s(\omega_j)$ we compute

$$\underline{p}(\omega_j) = \hat{c}_1 \cos \omega_j t + \hat{c}_2 \sin \omega_j t \tag{6}$$

where $\underline{\hat{c}} = \begin{vmatrix} \hat{c}_1 \\ \hat{c}_2 \end{vmatrix}$ is determined from (1), that is

$$\underline{\hat{c}} = (\underline{\Phi}^T \underline{\Phi})^{-1} \underline{\Phi}^T \underline{f} \quad . \tag{7}$$

Now when $\underline{p}(\omega_j)$ fits $\underline{f}$ perfectly ($\hat{\underline{v}} = 0$), then the fractional content of $\underline{f}$ represented by $\underline{p}$ is 1 (all of $\underline{f}$ is represented by $\underline{p}$). On the other hand, it is possible that $\underline{\hat{c}} = 0$ (that is $\underline{f}$ is orthogonal to $\underline{\Phi}$) and $\underline{p} = 0$. In this case, the fractional content of $\underline{f}$ represented by $\underline{p}$ is 0. In general we will see below that the fractional content of $\underline{f}$ represented by $\underline{p}$ can be measured by the ratio

$$s = \frac{\text{length of the orthogonal projection of } \underline{p} \text{ onto } \underline{f}}{\text{length of } \underline{f}} \tag{8}$$

and that this can be computed from

$$s = \frac{\underline{f}^T \underline{p}}{\underline{f}^T \underline{f}} \quad . \tag{9}$$

Note that since from (6) $\underline{p} = \underline{p}(\omega_j)$, so also

$$s(\omega_j) = \frac{\underline{f}^T \underline{p}(\omega_j)}{\underline{f}^T \underline{f}} \tag{10}$$

that is, for each spectral value $s(\omega_j)$ we must separately compute the least-squares approximant $\underline{p}(\omega_j)$. Therefore to compute the least-squares spectrum $\underline{s}(\omega) = \{s_j, \omega_j\}$ we must compute m spectral values $s(\omega_j)$ $j=1,2,\ldots,m$ which involves performing the least-squares approximation m times, each time to get $\underline{p}(\omega_j)$ for a different frequency ω_j.

So far we have dealt only with the problem of computing the spectrum of a complete time series. This is one major application of LSSA. A second major application is to first remove some constituents from the time series, and then to compute the spectrum of the residual time series. This is more complicated, and is more easily described (and hopefully understood) using the language of functional analysis.

REQUISITE ELEMENTS OF FUNCTIONAL ANALYSIS

Functional analysis is the analysis of functionals [Luenberger, 1969; Kreyszig, 1978; Oden, 1979]. A functional is a scalar function of vector quantities. We are interested in three functionals called the scalar product, the norm, and the metric. First we define some spaces.

In this report we will speak solely of spaces of finite dimensions. A vector space L of dimension n = dim L is a space of all possible n-tuples $\{\ell 1, \ell_2, \ldots, \ell_n\}$ of real numbers $\ell_1, \ell_2, \ldots, \ell_3$. It is required that a linear combination of any elements is also an element, that is if $\forall i: \underline{a}_i \in L$ and $\alpha_i \in R$ (scalars), then

$$\underline{b} = \sum_i \alpha_i \underline{a}_i \tag{11}$$

is also from L. A Hilbert (finite) space H is a vector space on which the scalar product is defined. If $\underline{a}, \underline{b} \in H$ then we denote their scalar product by $\langle \underline{a}, \underline{b} \rangle$ and define the norm (or length or magnitude) of $\underline{a} \in H$ as

$$||\underline{a}|| = (\langle \underline{a}, \underline{a} \rangle)^{1/2} \tag{12}$$

and the metric (or distance) between $\underline{a}, \underline{b} \in H$ as

$$d(\underline{a}, \underline{b}) = ||\underline{a} - \underline{b}|| = [\langle (\underline{a} - \underline{b}), (\underline{a} - \underline{b}) \rangle]^{1/2} \quad . \tag{13}$$

There are many ways of specifying a particular expression for the scalar product, some involving weight functions, some involving integrals. Here we use the most familiar and simplest expression

$$\langle \underline{a}, \underline{b} \rangle = \underline{a}^T \underline{b} = \sum_i a_i b_i \tag{14}$$

We will use concepts of linear independence, basis, and manifold. An n-tuple of vectors $\underline{a}_i \in L$ is <u>linearly independent</u> when the equation

$$\sum_{i=1}^{n} \alpha_i \underline{a}_i = 0 \quad , \forall i: \alpha_i \in R \tag{15}$$

is satisfied if and only if $\forall i: \alpha_i = 0$. That is none of the $\underline{a}_i$ can be expressed as a linear combination of the others. Given a linearly independent n-tuple $\{\underline{a}_i, \forall i\} \subset L$ then the set of all vectors

$$\underline{b}_j = \sum_i \alpha_i \underline{a}_i \tag{16}$$

(where all possible combinations of scalar values α_i are used to generate different $\underline{b}_j$) form a <u>manifold</u> S of L, and $\{\underline{a}_i, \forall i\}$ is said to <u>generate</u> S, or to be a <u>basis</u> of S. The number n of vectors in $\{\underline{a}_i, \forall i\}$ is the <u>dimension</u> of S.

We will also use concepts of orthogonality, and orthogonal projection. Let us explore how these concepts are intimately related to the scalar product. For illustration we will consider two vectors $\underline{a}$ and $\underline{b}$ in the real plane. If these two vectors intersect at a right angle then their scalar product is zero, $\langle \underline{a}, \underline{b} \rangle = 0$. In this case we say that $\underline{a}$ and $\underline{b}$ are <u>orthogonal</u> (denoted $\underline{a} \perp \underline{b}$). In more general Hilbert spaces, the following statements also all mean the same thing, although "intersection at a right angle" can no longer be visualized:

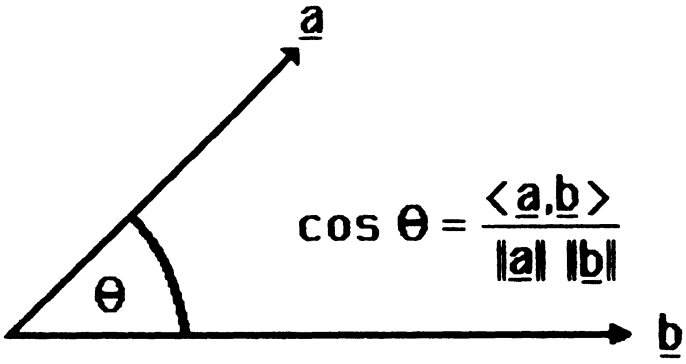
$\langle \underline{a}, \underline{b} \rangle = 0$ means the same as

$\underline{a} \perp \underline{b}$, which means the same as

$\underline{a}$ and $\underline{b}$ are orthogonal.

So much for the special case of orthogonality. In general $\underline{a}$ and $\underline{b}$ will <u>not</u> intersect at a right angle. What then is the geometrical meaning of the scalar product? Let us say they intersect at some angle θ (see Figure 1). We recall

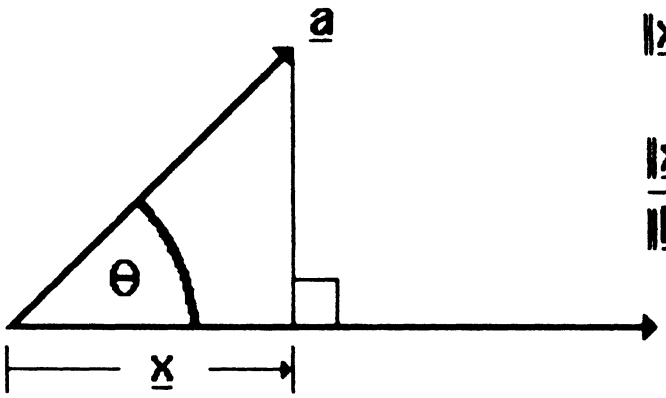
$$\langle \underline{a}, \underline{b} \rangle = ||\underline{a}|| \; ||\underline{b}|| \cos \theta \quad . \tag{17}$$



$$\cos\theta = \frac{\langle \underline{a},\underline{b}\rangle}{\|\underline{a}\|\,\|\underline{b}\|}$$

ORTHOGONAL PROJECTIONS

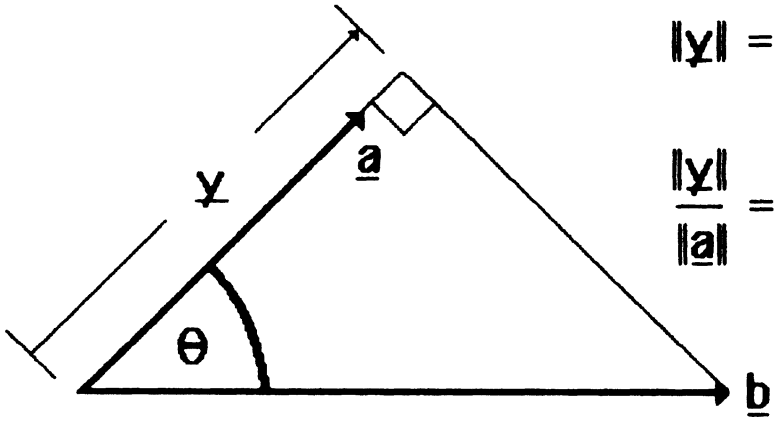
Definition of angle



$$\|\underline{x}\| = \|\underline{a}\|\cos\theta = \frac{\langle \underline{a},\underline{b}\rangle}{\|\underline{b}\|}$$

$$\frac{\|\underline{x}\|}{\|\underline{b}\|} = \frac{\langle \underline{a},\underline{b}\rangle}{\langle \underline{b},\underline{b}\rangle}, \quad \underline{x} = \frac{\langle \underline{a},\underline{b}\rangle}{\langle \underline{b},\underline{b}\rangle}\,\underline{b}$$

Orthogonal Projection of $\underline{a}$ onto $\underline{b}$



$$\|\underline{y}\| = \|\underline{b}\|\cos\theta = \frac{\langle \underline{a},\underline{b}\rangle}{\|\underline{a}\|}$$

$$\frac{\|\underline{y}\|}{\|\underline{a}\|} = \frac{\langle \underline{a},\underline{b}\rangle}{\langle \underline{a},\underline{a}\rangle}, \quad \underline{y} = \frac{\langle \underline{a},\underline{b}\rangle}{\langle \underline{a},\underline{a}\rangle}\,\underline{a}$$

Orthogonal Projection of $\underline{b}$ onto $\underline{a}$

FIGURE 1.

Orthogonality is such a useful concept that even in this case we want somehow to construct a right angle. There are two possibilities. Either we can drop a perpendicular from $\underline{a}$ onto $\underline{b}$ or we can drop a perpendicular from $\underline{b}$ onto $\underline{a}$. In Figure 1, the vector $\underline{x}$ is called the orthogonal projection of $\underline{a}$ onto $\underline{b}$, and $\underline{y}$ is the orthogonal projection of $\underline{b}$ onto $\underline{a}$. From (17) we see that the lengths of these vectors are

$$\begin{aligned} ||\underline{x}|| &= ||\underline{a}|| \cos\theta = \frac{\langle \underline{a}, \underline{b} \rangle}{||\underline{b}||} \quad , \\ ||\underline{y}|| &= ||\underline{b}|| \cos\theta = \frac{\langle \underline{a}, \underline{b} \rangle}{||\underline{a}||} \quad . \end{aligned} \tag{18}$$

To obtain expressions for the vectors themselves, we note that unit vectors in the direction of $\underline{a}$ (and $\underline{y}$) and in the direction of $\underline{b}$ (and $\underline{x}$) are given by $\frac{\underline{a}}{||\underline{a}||}$ and $\frac{\underline{b}}{||\underline{b}||}$ respectively. Hence, using (12), we have

$$\begin{aligned} \underline{x} &= ||\underline{x}|| \frac{\underline{b}}{||\underline{b}||} = \frac{\langle \underline{a}, \underline{b} \rangle}{\langle \underline{b}, \underline{b} \rangle} \underline{b} \quad , \\ \underline{y} &= ||\underline{y}|| \frac{\underline{a}}{||\underline{a}||} = \frac{\langle \underline{a}, \underline{b} \rangle}{\langle \underline{a}, \underline{a} \rangle} \underline{a} \quad , \end{aligned} \tag{19}$$

from which we see that the ratio of the length of $\underline{x}$ to the length of $\underline{b}$ (and similarly of $\underline{y}$ to $\underline{a}$) is given by the ratio of two scalar products. Finally we note that (18) and (19) are not restricted to the simple example here, but are valid in any Hilbert space.

THE PROJECTION THEOREM

The shortest distance between a point and a plane is the perpendicular from the point to the plane. This is the projection theorem.

We can rephrase this theorem, substituting the terms

"minimum norm" for	"shortest distance";
"(vector) element of Hilbert space" for	"point";
"manifold of Hilbert space" for	"plane"; and
"orthogonal" for	"perpendicular".

Given $\underline{f} \in H$ (point) and $S \subset H$ (plane), then of all the elements $\underline{s} \in S$, there is one element $\underline{p} \in S$ such that $d(\underline{f}, \underline{p}) \leq d(\underline{f}, \underline{s})$ (shortest distance). This element $\underline{p}$ is given by the orthogonal projection of $\underline{f}$ onto S, that is $(\underline{f} - \underline{p}) \perp S$ (perpendicular) (see Figure 2).

In order to invoke this theorem, we need first to specify $\underline{f}$ and S. We can specify $\underline{f}$ in several ways, for example, as an ordered sequence of real numbers, or using an analytic functional expression. We can also specify S in many ways. Let us choose to specify S by specifying a basis $\{\underline{\phi}_i, \forall i\}$ which generates S. Figure 2 illustrates the geometrical relationships between $\underline{f}$, S, $\underline{p}$, and $\{\underline{\phi}_i, \forall i\}$ for the simple case of a three-dimensional $\underline{f}$ and a two-dimensional S. Then any $\underline{s} \in S$ can be expressed as

$$\underline{s} = \sum_i c_i \underline{\phi}_i \quad . \tag{20}$$

That is, there is some relationship between each n-tuple $\{c_i, \forall i\}$ and the corresponding $\underline{s}$. Let the particular n-tuple of scalars $\{c_i, \forall i\}$ corresponding to $\underline{p}$ be denoted $\{\hat{c}_i, \forall i\}$. Then

$$\underline{p} = \sum_i \hat{c}_i \underline{\phi}_i \quad . \tag{21}$$

Now we can write the condition we must satisfy, $(\underline{f} - \underline{p}) \perp S \equiv (\underline{f} - \underline{p}) \perp \underline{\phi}_j; \forall j$, in terms of $\{\hat{c}_i, \forall i\}$ corresponding to $\underline{p}$, that is

$$\forall j: \langle (\underline{f} - \sum_i \hat{c}_i \underline{\phi}_i), \underline{\phi}_j \rangle = 0 \tag{22}$$

This can be rewritten

$$\sum_i \hat{c}_i \langle \underline{\phi}_i, \underline{\phi}_j \rangle = \langle \underline{f}, \underline{\phi}_j \rangle \qquad j=1,2,\ldots,n \quad . \tag{23}$$

If we define

$$\underline{N} = \begin{bmatrix} \langle \underline{\phi}_1, \underline{\phi}_1 \rangle & \langle \underline{\phi}_2, \underline{\phi}_1 \rangle & \cdots & \langle \underline{\phi}_n, \underline{\phi}_1 \rangle \\ \langle \underline{\phi}_1, \underline{\phi}_2 \rangle & \langle \underline{\phi}_2, \underline{\phi}_2 \rangle & \cdots & \langle \underline{\phi}_n, \underline{\phi}_2 \rangle \\ \cdot & \cdot & \cdots & \cdot \\ \langle \underline{\phi}_1, \underline{\phi}_n \rangle & \langle \underline{\phi}_2, \underline{\phi}_n \rangle & \cdots & \langle \underline{\phi}_n, \underline{\phi}_n \rangle \end{bmatrix}$$

$$\underline{u} = \begin{bmatrix} \langle \underline{f}, \underline{\phi}_1 \rangle \\ \langle \underline{f}, \underline{\phi}_2 \rangle \\ \cdot \\ \langle \underline{f}, \underline{\phi}_n \rangle \end{bmatrix}$$

$$\underline{\hat{c}} = \begin{bmatrix} \hat{c}_1 \\ \hat{c}_2 \\ \cdot \\ \hat{c}_n \end{bmatrix}$$

then (23) becomes

$$\underline{N}\,\underline{\hat{c}} = \underline{u} \tag{24}$$

the normal equations ($\underline{f} - \underline{p}$ is normal or orthogonal to S). In fact if we define

$$\underline{\Phi} = [\underline{\phi}_1, \underline{\phi}_2, \cdots, \underline{\phi}_n] \tag{25}$$

then $\underline{N} = \underline{\Phi}^T \underline{\Phi}$ and $\underline{u} = \underline{\Phi}^T \underline{f}$ so that

$$\underline{\hat{c}} = \underline{N}^{-1}\,\underline{u} = (\underline{\Phi}^T \underline{\Phi})^{-1}\,\underline{\Phi}^T \underline{f} \tag{26}$$

which was equation (7). The approximant $\underline{p}$ is then

$$\underline{p} = \underline{\Phi}\,\underline{\hat{c}} = \sum_i \hat{c}_i\,\underline{\phi}_i \tag{27}$$

and the residual vector is

$$\underline{\hat{v}} = \underline{f} - \underline{p} = \underline{f} - \underline{\Phi}\,\underline{\hat{c}} = \underline{f} - \underline{\Phi}(\underline{\Phi}^T\underline{\Phi})^{-1}\underline{\Phi}^T\underline{f} \quad . \tag{28}$$

Note that $\underline{\hat{v}} \mid \underline{p}$. This follows from the projection theorem, where $\underline{\hat{v}} = (\underline{f} - \underline{p}) \mid S$, that is $\underline{\hat{v}}$ is orthogonal to all vectors in S, in particular the column vectors of $\underline{\Phi}$ (which generate S). Thus $\underline{\hat{v}}$ is orthogonal to $\underline{p}$ which is a linear combination of $\underline{\Phi}$, and hence also lies in S. Thus the projection theorem (or LSA or LLSPA) decomposes $\underline{f}$ into two orthogonal components $\underline{p}$ (the orthogonal projection of $\underline{f}$ onto S) and $\underline{\hat{v}}$ (the perpendicular from $\underline{f}$ to S).

To compute something akin to the spectral value, we must perform a second orthogonal projection. However, this one is simpler. So far we have projected $\underline{f}$ onto the manifold S, in which case many projections are possible, and we used the minimum norm, or perpendicularity condition to select the one we want. For the second projection, we simply project $\underline{p}$ back onto $\underline{f}$. Figure 3 illustrates this for the simple case corresponding to Figure 2.

The length of this orthogonal projection is, from (18)

$$\frac{\langle \underline{f}, \underline{p} \rangle}{||\underline{f}||} \quad .$$

The ratio of the length of this orthogonal projection to the length of $\underline{f}$ is, from (19)

$$\frac{\langle \underline{f}, \underline{p} \rangle}{\langle \underline{f}, \underline{f} \rangle} = \frac{\underline{f}^T \underline{p}}{\underline{f}^T \underline{f}} \quad . \tag{29}$$

This then is a measure of the fractional part of $\underline{f}$ which is represented by $\underline{p}$.

Since $\underline{p}$ is a special element of S (the orthogonal projection of $\underline{f}$ onto S), this ratio also tells us something about how much of $\underline{f}$ is "contained" in S. The "closer" to S that $\underline{f}$ lies, the closer to 1 will the ratio (29) become. If $\underline{f}$ lies in S, the ratio is 1. If $\underline{f}$ is orthogonal to S, the ratio is 0.

Now let us apply this to spectral analysis. For each spectral frequency ω_j, j=1,m, we have a different manifold S spanned by

$$\underline{\Phi} = [\cos \omega_j t, \quad \sin \omega_j t] \quad . \tag{30}$$

Consequently, the orthogonal projection $\underline{p}(\omega_j)$ of $\underline{f}$ onto S will be different for each ω_j. Due to the properties of the ratio (29) described above, we choose that ratio to be the least-squares spectral value of $\underline{f}$ for frequency ω_j

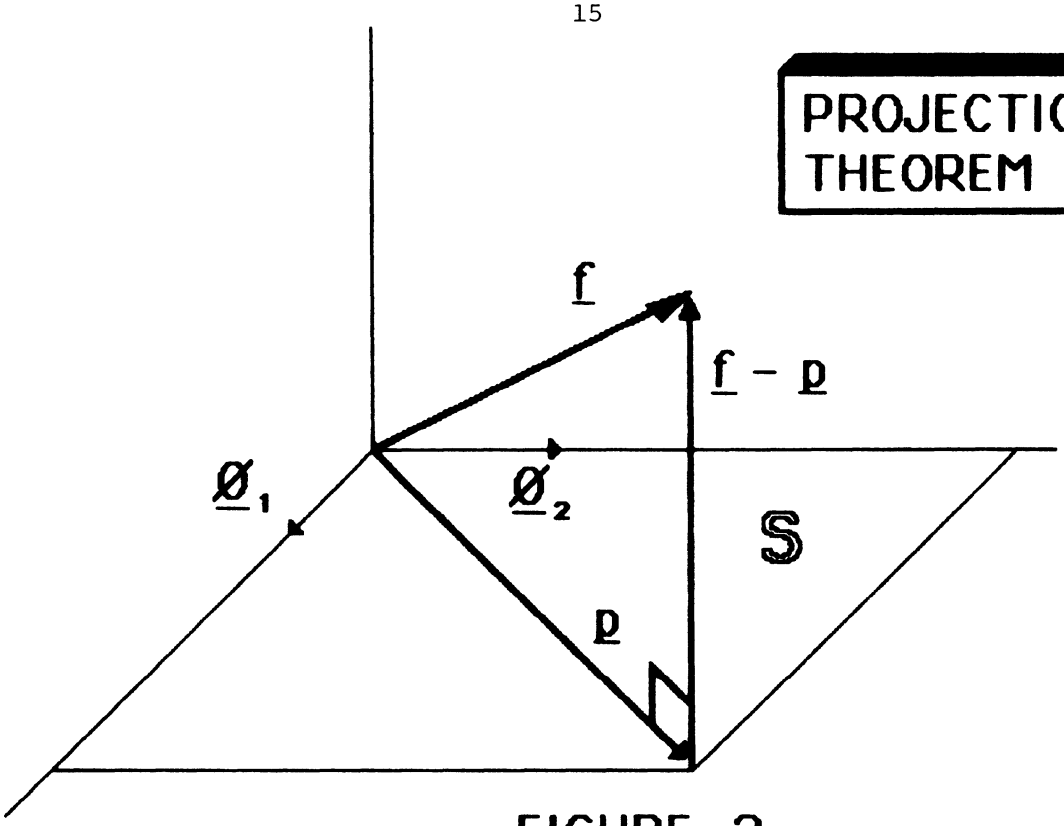


FIGURE 2.

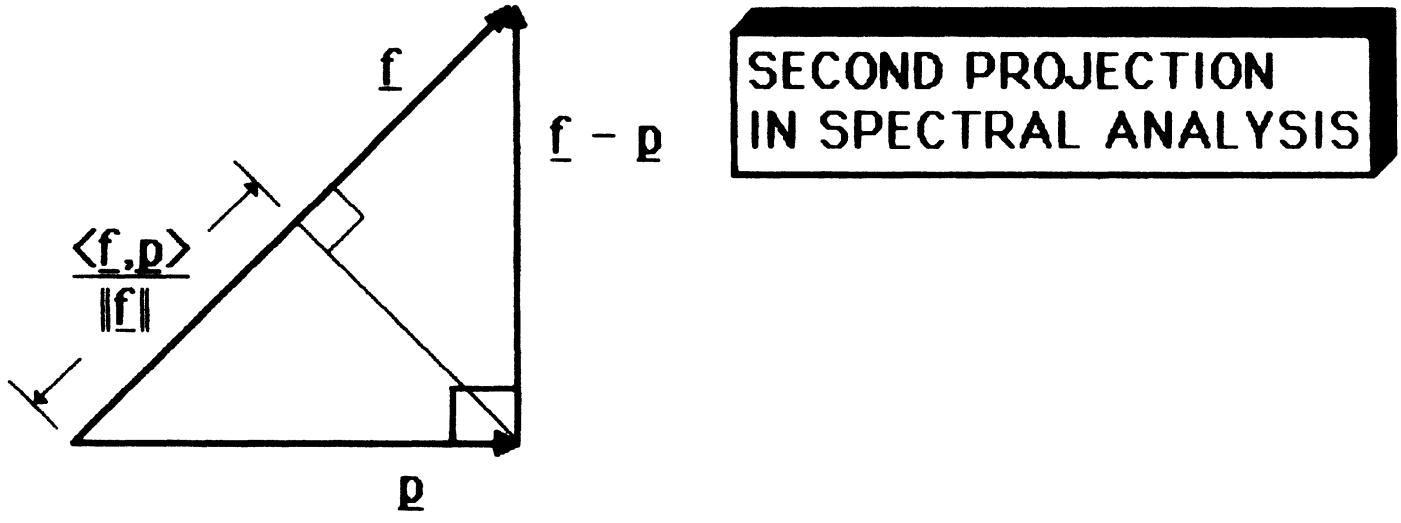


FIGURE 3.

$$s(\omega_j) = \frac{\underline{f}^T \underline{p}\ (\omega_j)}{\underline{f}^T \underline{f}} \tag{31}$$

The least-squares spectrum of $\underline{f}$ is the collection of spectral values for all (desired) frequencies ω_j,

$$\underline{s}(\omega) = \{s(\omega_j);\ j=1,m\} \tag{32}$$

LEAST-SQUARES SPECTRAL ANALYSIS WITH KNOWN CONSTITUENTS

For some applications we can consider a time series as consisting of two kinds of constituents: those which we are interested in studying (and having represented in the spectrum), i.e., the signal, and those which we are not interested in, or which obscure the constituents we want to study, i.e., the noise. The noise can be either periodic, rendering the series "coloured" or other, rendering the series non-stationary, or both.

In both cases, we must know something about the constituent in order to deal with it. Here we restrict ourselves to the case where we know the noise base functions $\underline{\phi}_i(t)$, but do not know what the magnitude of the contribution is to the time series; that is if we represent

$$\underline{f}(t) = \sum_i^{NK} c_i\ \underline{\phi}_i(t) \tag{33}$$

we know $\underline{f}(t)$ and all the $\underline{\phi}_i(t)$, and do not know the coefficients c_i. So far this is similar to the previous case.

Now, however, we partition $\underline{\Phi}$ into the known constituents $\hat{\underline{\Phi}}$, and the spectral functions, cos $\omega_j t$, sin $\omega_j t$, we used before, so that

$$\underline{\Phi} = [\hat{\underline{\phi}}_1, \hat{\underline{\phi}}_2, \ldots, \hat{\underline{\phi}}_{NK}, \cos \omega_j t, \sin \omega_j t] \tag{34}$$

It was shown by Vaníček [1971] that the known constituents do not have to be removed from $\underline{f}$ before evaluating the spectrum. It speeds the computations up, however, if the least-squares estimate $\hat{\underline{p}} = \hat{\underline{c}}\hat{\underline{\Phi}}$ is removed before the

spectrum is evaluated. This simply means we decompose $\underline{f}(t)$ into its orthogonal projection $\hat{\underline{p}} \in \{c\hat{\underline{\Phi}}\}$ and the residual $\underline{f} - \hat{\underline{p}}$ (which is orthogonal to $\hat{\underline{p}}$). We then have the residual time series

$$\underline{g}(t) = \underline{f}(t) - \hat{\underline{p}}(t) \tag{35}$$

and it is this that we compute the spectrum for. We then orthogonally project $\underline{g}$ onto the manifold $M(\Phi)$ spanned by $\underline{\Phi} = [\hat{\underline{\Phi}}, \cos \omega t, \sin \omega t]$ and obtain the projection $\underline{r} = \underline{p} - \hat{\underline{p}}$ and residual $\hat{\underline{v}} = \underline{g} - \underline{r} = \underline{f} - \underline{p}$. Note that whether we project $\underline{f}$ onto $M(\underline{\Phi})$ directly, or $\underline{f}$ onto $M(\hat{\underline{\Phi}})$ and then $\underline{f} - \hat{\underline{p}}$ onto $M(\underline{\Phi})$, we obtain the same final residual. Finally, we orthogonally project $\underline{r}$ onto $\underline{g}$ and compute the ratio of the length of the projection to the length of $\underline{g}$ as our spectral value.

In summary:

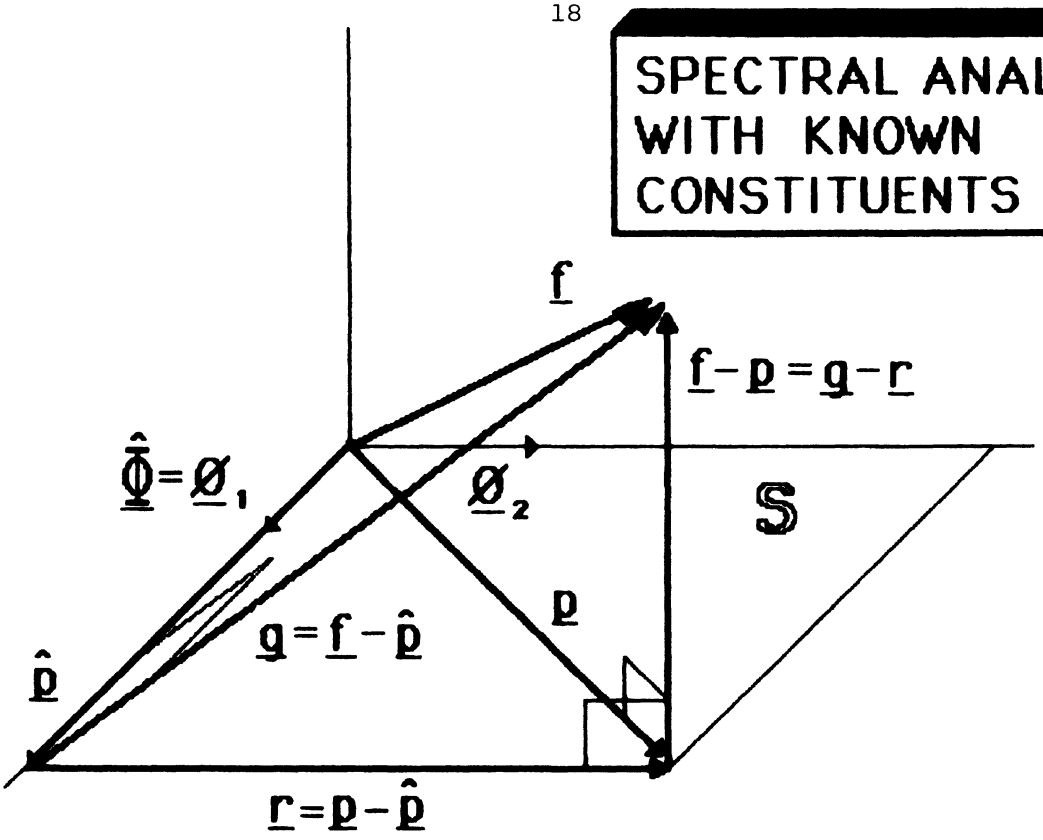
(a) LSA and LLSPA involve one orthogonal projection: $\underline{f}$ onto $M(\underline{\Phi})$.

(b) LLSA with no known constituents involves two orthogonal projections
   (i) $\underline{f}$ onto $M(\underline{\Phi})$ to obtain $\underline{p}$
   (ii) $\underline{p}$ onto $\underline{f}$ to obtain the spectral value.

(c) LSSA with known constituents involves three orthogonal projections
   (i) $\underline{f}$ onto $M(\hat{\underline{\Phi}})$ to eliminate the known constituents and obtain $\underline{g}$
   (ii) $\underline{g}$ onto $M(\underline{\Phi})$ to obtain $\underline{r}$
   (iii) $\underline{r}$ onto $\underline{g}$ to obtain the spectral value.

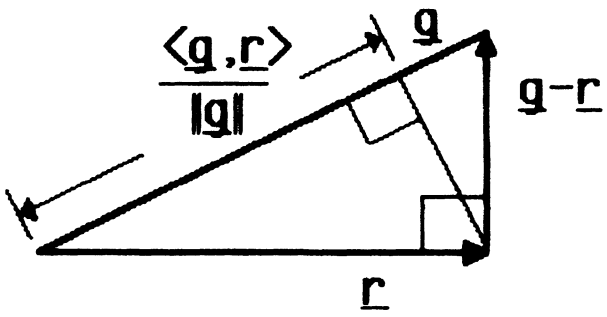
In order to geometrically illustrate the concept of these three orthogonal projections as in Figure 4, we have to unrealistically restrict $\underline{\Phi}$ to two dimensions. If we let $\hat{\Phi} = \underline{\phi}_1$ be one dimensional, that leaves only one dimension, $\underline{\phi}_2$, to represent the spectral functions (of which we have in actuality two). If we can live with this limitation in order to look at the three projections conceptually, then the top part of Figure 4 shows the first and second projections ($\underline{f}$ onto $M(\hat{\underline{\Phi}})$, and $\underline{g}$ onto $M(\underline{\Phi})$), and the bottom part of the figure shows the second and third projections ($\underline{g}$ onto $M(\underline{\Phi})$, and $\underline{r}$ onto $\underline{g}$).

By analogy with (29) the spectral value is



1. $\underline{f}$ onto $\hat{\underline{\Phi}}$ to obtain $\underline{g}$

2. $\underline{g}$ onto $\underline{\Phi}$ to obtain $\underline{r}$



3. $\underline{r}$ onto $\underline{g}$ to obtain spectral value

FIGURE 4.

$$s(\omega_j) = \frac{\underline{g}^T \underline{r}(\omega_j)}{\underline{g}^T \underline{g}} \tag{36}$$

where $\underline{g} = \underline{f} - \hat{\underline{p}}$ and $\underline{r} = \underline{p} - \hat{\underline{p}}$. We must compute $\underline{r}(\omega_j)$ for each spectral frequency ω_j, however we need compute only once the quantities (from (28))

$$\underline{g} = \underline{f} - \hat{\underline{\Phi}} (\hat{\underline{\Phi}}^T \hat{\underline{\Phi}})^{-1} \hat{\underline{\Phi}}^T \underline{f} \tag{37}$$

and $\underline{g}^T \underline{g}$. Then the projection of $\underline{g}$ onto $M(\underline{\Phi})$ has the form

$$\underline{r} = \underline{\Phi} (\underline{\Phi}^T \underline{\Phi})^{-1} \underline{\Phi}^T \underline{g} \tag{38}$$

so that

$$\underline{g}^T \underline{r} = \underline{g}^T \underline{\Phi} (\underline{\Phi}^T \underline{\Phi})^{-1} \underline{\Phi}^T \underline{g} \quad . \tag{39}$$

Now $\underline{g}^T \underline{\Phi}$ can be written

$$\underline{g}^T \underline{\Phi} = \underline{g}^T [\hat{\underline{\Phi}}, \cos \omega_j t, \sin \omega_j t] \quad . \tag{40}$$

But $\underline{g}^T \hat{\underline{\Phi}} = 0$ ($\underline{g}$ is orthogonal to $M(\hat{\underline{\Phi}})$), so that

$$\underline{g}^T \underline{\Phi} = [0, 0, \ldots, 0, \underline{g}^T \cos \omega_j t, \underline{g}^T \sin \omega_j t] \quad . \tag{41}$$

Hence in (39) only the south-east 2 by 2 submatrix of $(\underline{\Phi}^T \underline{\Phi})^{-1}$ need be computed.

More specifically, denoting $\underline{\Phi}^T \underline{\Phi}$ by $\underline{A}$, we have

$$\underline{A} = \begin{array}{|ccc|} \hat{\underline{A}} & \underline{u} & \underline{v} \\ \underline{u}^T & CC & CS \\ \underline{v}^T & CS & SS \end{array}$$

where the NK by NK matrix $\hat{\underline{A}} = \hat{\underline{\Phi}}^T \hat{\underline{\Phi}}$, the NK-vectors $\underline{u}$ and $\underline{v}$ are given by $\underline{u}_j = \underline{\phi}_j^T \underline{\phi}_{NK+1}$, $j = 1,2,\ldots,NK$ and $\underline{v}_j = \underline{\phi}_j^T \underline{\phi}_{NK+2}$, $j = 1,2,\ldots,NK$, and the elements

CC = $\underline{\phi}_{NK+1}^T \underline{\phi}_{NK+1}$, CS = $\underline{\phi}_{NK+1}^T \underline{\phi}_{NK+2}$, and SS = $\underline{\phi}_{NK+2}^T \underline{\phi}_{NK+2}$. (Note that alternatives to the normal equations such as the Householder transformation could be used;however, they would probably involve penalties in computation times over the algorithm we have chosen.) Since the matrix $\underline{\hat{A}}$ is positive definite symmetric, it is most conveniently inverted by the Choleski method. The residual time series then is $\underline{g} = \underline{f} - \underline{\hat{\Phi}}(\underline{\hat{\Phi}}^T\underline{\hat{\Phi}})^{-1}\underline{\hat{\Phi}}^T\underline{f}$ and its quadratic norm is FNORM = $\underline{g}^T\underline{g} = \underline{f}^T(\underline{I} - \underline{\hat{\Phi}}(\underline{\hat{\Phi}}^T\underline{\hat{\Phi}})^{-1}\underline{\hat{\Phi}}^T)\underline{f}$.

The orthogonal projection $\underline{r}$ of $\underline{g}$ onto $\underline{\Phi}$ is $\underline{r} = \underline{\Phi c}$, where the coefficient vector $\underline{c}$ satisfies the normal equations $\underline{Ac} = \underline{b}$, where $\underline{A} = \underline{\Phi}^T\underline{\Phi}$ and b = $\underline{\Phi}^T\underline{g}$ are known. Then $\underline{g}^T\underline{r} = \underline{g}^T\underline{\Phi c} = \underline{b}^T\underline{c} = \underline{b}^T\underline{A}^{-1}\underline{b}$, and $\underline{s}(\omega) = \underline{b}^T\underline{A}^{-1}\underline{b}$/FNORM. From (2) the first NK components of the (NK+2)-vector $\underline{b}$ are zero, the last two being FCOS = $\underline{g}^T\underline{\phi}_{NK+1}$ and FSIN = $\underline{g}^T\underline{\phi}_{NK+2}$. Hence we really need only determine the lower right-hand 2 by 2 submatrix of $\underline{A}^{-1}$.

It is easily shown that the lower right-hand 2 by 2 submatrix of $\underline{A}^{-1}$ is:

$$\frac{1}{DET}\begin{bmatrix} (SS-VAV), & -(CS-UAV) \\ -(CS-UAV), & (CC-UAU) \end{bmatrix},$$

where UAU = $\underline{u}^T\underline{\hat{A}}^{-1}\underline{u}$, VAV = $\underline{v}^T\underline{\hat{A}}^{-1}\underline{v}$, UAV = $\underline{u}^T\underline{\hat{A}}^{-1}\underline{v}$, and DET = (CC-UAU)(SS-VAV) - (CS-UAV)^2. Hence the algorithm for computing the spectrum of $\underline{f}$ is:

$$s(\omega) = [(SS-VAV)FCOS^2-2(CS-UAV)FCOS\cdot FSIN+(CC-UAU)FSIN^2]/(DET\cdot FNORM). \quad (42)$$

RELATIONSHIP TO OTHER SPECTRAL FUNCTIONS

To relate this spectrum to the Fourier spectrum, a basis for other kinds of spectra, note that in the absence of known constituents UAU = VAV =

UAV = 0 and $\underline{g} = \underline{f}$. If the time series is equally spaced and symmetrical about the time origin, then CS = 0 and $s(\omega) = (1/FNORM)[(FCOS^2/CC) + (FSIN^2/SS)]$. Letting the time series length increase beyond all limits as the time series spacing decreases to zero, and introducing the compact definition of the scalar product,

$$\underline{x}^T\underline{y} = \int_{-\infty}^{\infty} \underline{x}(t)\ \underline{y}(t)\ dt,$$

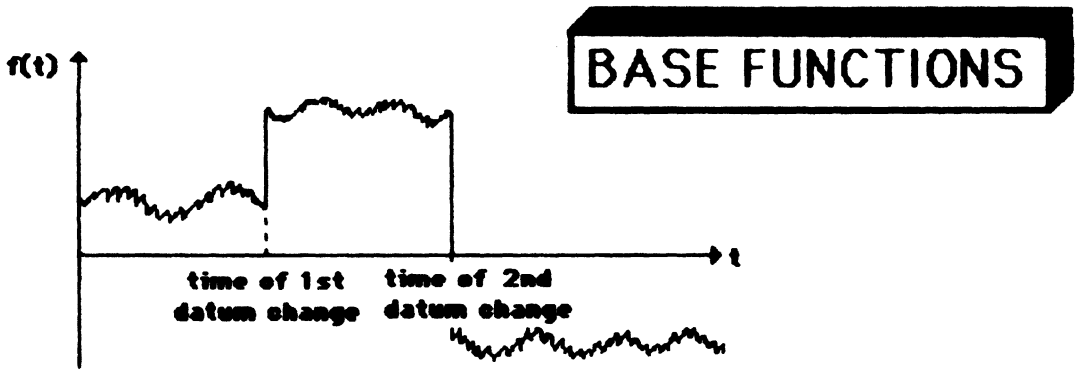
then in our notation the square of the absolute value of the Fourier transform of $\underline{f}$, $|C(\omega)|^2 = (1/2\pi)(FCOS^2 + FSIN^2)$, can be compared with the above expression for the least-squares spectrum.

There are other possibilities to define a least-squares spectrum, namely, $s(\omega) = (\alpha^2 + \beta^2)/||\underline{f}||^2$, where α, β are evaluated (a) from the orthogonal projection $\hat{\Phi}\underline{c} + \alpha\cos(\omega t) + \beta\sin(\omega t)$ of $\underline{f}$ onto $M(\underline{\Phi})$, or (b) from the orthogonal projection $\alpha\cos(\omega t) + \beta\sin(\omega t)$ of $\underline{g} = \underline{f} - \hat{\underline{p}}$ onto the two-dimensional manifold spanned by $\{\cos(\omega t), \sin(\omega t)\}$. In the first case, the spectrum is not defined for values of ω which are present in the known constituents. The second case (equivalent to the standard Fourier analysis approach) distorts the spectrum by forcing it to go to zero for the frequencies present in the known constituents. Both cases are discussed by Taylor and Hamilton [1972] and neither is found to be advantageous from the spectral accuracy point of view.

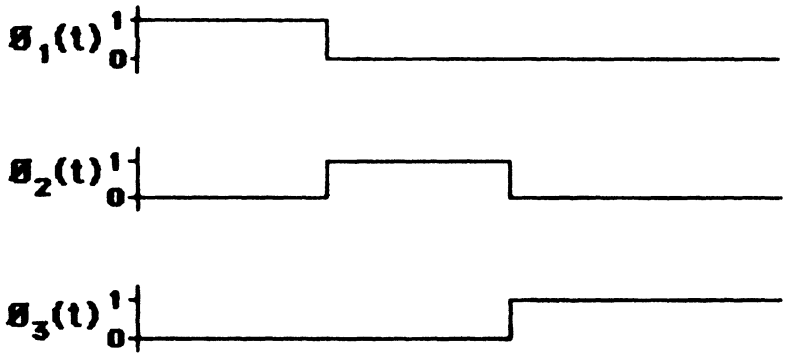
TYPES OF KNOWN CONSTITUENTS

We now turn to the specific software implementation of LSSA documented in this report. In this software, the known constituent base functions $\hat{\underline{\Phi}}$ can be of several types.

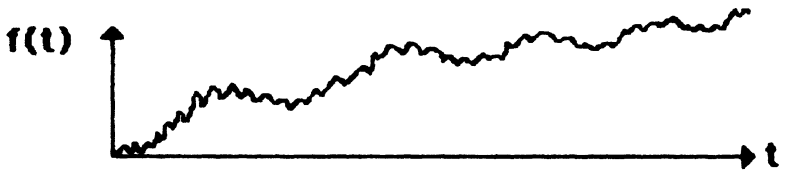
(a) $\phi(t) = 1$ for datum bias. For example, say a tide gauge was moved twice in 10 years and the times of the move were known, but the vertical relationship of the different locations was not known. The time series would look like that shown in the top part of Figure 5. The three datum bias known constituent base functions in Figure 5 would be used.



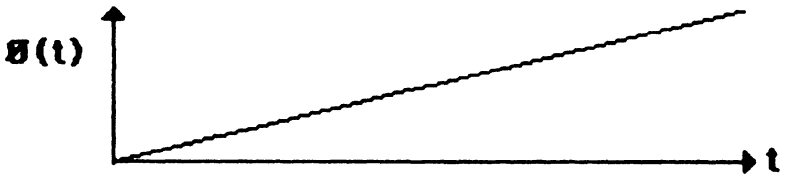
Time Series Containing Datum Biases



Base Functions to Remove Datum Biases



Time Series Containing Linear Trend



Base Function to Remove Linear Trend

FIGURE 5.

(b) $\phi(t) = t$, for linear trend. For example, say a tide gauge was situated on a dock which was slowly (or uniformly) sinking into the seabed. Then the time series would look like that in the bottom part of Figure 5. The linear trend constituent base function in Figure 5 would be used to remove this linear trend.

(c) $\phi_1(t) = \cos \mu_i t$ |
$\phi_2(t) = \sin \mu_i t$ | for forced periodic constituents, with frequencies μ_i.

For example, we know that a given time series contains the tidal frequencies M_2 and K_1 (perhaps from a previous spectral analysis), so we want to remove these peaks from the spectrum and see what is left.

(d) $\phi(t)$ = "anything else" for user defined constituents. For example, instead of a linear trend we may believe that some nonlinear trend (say exponential) exists.

INPUT AND OUTPUT PARAMETERS

The input parameters for computing the spectrum (5) must specify the time series, the limits and density of the spectral band to be produced, and the known constituents $\hat{\Phi}$.

The time series is defined by the vectors (F_i, T_i) i=1,2,...,NF where the values T_i are in units of time, and for SPECUN (the version used for unequidistant series) are unrestricted as to spacing. For SPECEQ (the version used for equidistant series) the time series is assumed to consist of a specific number (NIVL) of subintervals, each of which consists of equally spaced data points separated by a time increment STEP common to all subintervals. The subintervals need not be separated by integral multiples of STEP. Separation of subintervals is specified as detected by the software when two consecutive elements are not separated by STEP. The value for STEP is defined by the difference between the first and second elements in the time series.

The spectrum is defined by the vectors (S_i, P_i) i=1,2,...,NW, where the values P_i are specified spectral periods in the same units as T_i, and the values S_i are the computed spectral values.

The known constituents. Rather than requiring the user to specify the form of $\hat{\underline{s}}$, it is useful to build some common types of known constituents into the algorithm leaving the user free to ignore them and specify his own functions if he so desires. This algorithm, therefore, provides four optional types of known constituents:

(a) Datum Bias. Let the time series consist of NDAT segments, each referred to a different datum. Then for NDAT $\geq$ 1

$$\underline{\phi}_i(\underline{t}) = \begin{cases} 1 & \text{if t is in the ith datum segment} \\ 0 & \text{otherwise} \end{cases} \qquad i=1,2,\ldots,NDAT$$

In the program in this case

NDAT = 3 (number of segments of total time series separated by datum shifts)

DAT(1) = t_o (start time of first datum = start time of time series)

DAT(2) = t_1 (start time of second datum)

DAT(3) = t_2 (start time of third datum).

If there are no datum biases we set NDAT = 0, and the contents of DAT are not used. On input to routines SPECUN and SPECEQ, if NDAT is negative, a warning message is produced, NDAT is set to zero, and the program continues. If NDAT is positive and DAT(1) $\neq t_o$, a fatal message is produced and the program aborts.

(b) Linear Trend. If used, this known constituent is of the form

$\underline{\phi}_i(\underline{t}) = \underline{t}$, i = NDAT + 1 (LT = 1 if used, LT = 0 otherwise) .

On input to routines SPECUN and SPECEQ, if LT is not either 0 or 1, a warning message is produced, LT is set to 0, and the program continues.

(c) Forced Periods. For NPER $\geq$ 1 known periods PER_j (and frequencies $\mu_j = 2\pi/PER_j$), the known constituents are the periodic functions

$\underline{\phi}_i(\underline{t}) = \cos(\mu_j t)$

$i = NDAT + LT + 2j-1$, j=1,2,...,NPER.

$\underline{\phi}_{i+1}(\underline{t}) = \sin(\mu_j t)$

In the program in this case we set

NPER = 2 (number of frequencies to be removed)

PER(1) = 12.42 hours (for M_2) (period of first frequency)

PER(2) = 23.93 hours (for K_1) (period of second frequency).

If we do not want to remove any periodic constituents before computing the spectrum we set NPER = 0, and the contents of PER are not used. On input to routines SPECUN and SPECEQ, if NPER is negative, a warning message is produced, NPER is set to zero, and the program continues. On input to routine SPECEQ only, if NPER is greater than the dimension of the arrays required (NPERDM), a fatal message is produced and the program aborts.

(d) User-specified. These known constituents are of arbitrary form (for example, quadratic trend or exponential trend, a numerical function) chosen by each user

$\underline{\phi}_i(\underline{t})$ = ?, i=NDAT + LT + 2*NPER + j, j=1,2,...,NBASE.

In the example at the end of PART A we set

NBASE = 1 (number of user defined constituents to be removed)

and add the appropriate code in subroutine BASE to implement the user-defined function. This particular user defined function is an exponential trend and the code reads:

BASE = EXP(-T/25.).

If there are no user defined constituents to be removed, set NBASE = 0. On input to routines SPECUN and SPECEQ, if NBASE is negative, a warning message is produced, NBASE is set to zero, and the program continues.

The total number of known constituents then is:

$$NK = NDAT + LT + 2*NPER + NBASE \ , \qquad (43)$$

which may also equal to 0 (for NDAT = LT = NPER = NBASE = 0).

On input to routines SPECUN and SPECEQ, if (43) is not satisfied, a warning message is produced, NK is set equal to the right hand side of (43), and the computation continues. If NK is greater than the dimensions of the arrays required (NKDIM), a fatal message is produced, and the program aborts.

The next two input parameters, MODE and EQORUN, specify whether a sequential or batch solution is desired (MODE) and if SPECEQ or SPECUN should be used. Standard deviations of and correlations between a priori estimates $\underline{\hat{c}}$ are also evaluated from the usual statistical formulae. Their values are printed if they are considered significant. The significance level for standard deviations (in percents) is another input parameter, PCENT; for correlation, the level is called CLEVEL.

Two more statistical parameters are produced by the software: the mean spectral value for white noise (see Vaníček [1971]):

$$RS = 2/(NF - NK) * 100\% \quad , \tag{44}$$

and the critical percentage variance on 95% for detecting statistically significant peaks in the spectrum [Steeves, 1981]:

$$RS95 = (1 - \alpha^{2/(NF-NK-2)}) * 100\% \quad , \tag{45}$$

where $\alpha = 0.95$. These are printed together with the spectrum.

GENERAL SCALAR PRODUCT ALGORITHM FOR EQUALLY SPACED DATA

The spectrum (5) requires evaluation of the scalar products FNORM, FCOS, FSIN, CC, CS, SS, and U_i, V_i (i=1,2,...,NK). For an unequally spaced time series treated by SPECUN, all these scalar products must be evaluated directly from:

$$\underline{x}^T\underline{y} = \langle \underline{x}, \underline{y} \rangle = \sum_{t_i} x(t_i)\, y(t_i) \tag{46}$$

where, for convenience, we now introduce the bracket notation $\langle \underline{x}, \underline{y} \rangle$. Provided that the time series is at least piecewise equally spaced (as described in the previous section for input to SPECEQ), we can use much more efficient formulae to evaluate CC, CS, SS and those elements of the vectors $\underline{U}$ and $\underline{V}$ corresponding to datum bias, linear trend, and forced period known constituents. However, FNORM, FCOS, FSIN and those elements of $\underline{U}$ and $\underline{V}$ corresponding to user-defined known constituents must still be evaluated directly from (44).

For convenience we define the function trig(x) as being either cos(x) or sin(x). We seek, to begin with, an algorithm for the scalar products

$$<1, \text{trig}(\omega t)> = \Sigma_{T_i} \text{trig}(\omega T_i) \quad i=1,2,\ldots,NF. \tag{47}$$

Direct evaluation requires computing NF trigonometric functional values. We can reduce this number considerably by applying the identities [Korn and Korn, 1968, p. 981]:

$$\sum_{k=0}^{n} \text{trig}(2ak + b) = [1/\sin(a)]* \sin(an + a)* \text{trig}(an + b). \tag{48}$$

Let the jth subinterval of the time series F consist of equally spaced data points, separated by the time increment STEP, the first data point occurring at time TA_j and the last at time TB_j. Then setting

$k = (T_i - TA_j)/STEP = 0,1,\ldots,n;$

$n = (TB_j - TA_j)/STEP;$

$a = (\omega/2)*STEP;$ and

$b = \omega*TA_j;$

we have

$$\sum_{T_i=TA_j}^{TB_j} \text{trig}(\omega T_i) = [1/\sin(Q)]* \sin(N_jQ)* \text{trig}(L_jQ) , \tag{49}$$

where $Q = (\omega/2)*STEP$; $N_j = 1 + (TB_j - TA_j)/STEP$; and $L_j = (TB_j + TA_j)/STEP$.

Summing over the NIVL subintervals in F gives us the scalar product:

$$<1, \text{trig}(\omega T)> = \frac{1}{\sin(Q)} \sum_{j=1}^{NIVL} \sin(N_jQ)* \text{trig}(L_jQ) , \tag{50}$$

which requires computing only (2*NIVL+1) trigonometric functional values, where NIVL is the number of subintervals.

SPECIFIC SCALAR PRODUCT EXPRESSIONS

It now simply remains to reduce the scalar products CC, CS, SS, $\underline{U}$, and $\underline{V}$ to the form of (50). Using (46) it is easy to see that CC = (NF/2) +

$(1/2)*$ <1, cos2ωT>; CS = $(1/2)*$ <1, sin2ωT>; and SS = (NF/2) - $(1/2)*$ <1, cos2ωT> where from (50) we can see that:

$$<1,\ \mathrm{trig}(2\omega T)> = \frac{1}{\sin(2Q)} \sum_{j=1}^{NIVL} \sin(2N_jQ)*\ \mathrm{trig}(2L_jQ) \quad . \quad (51)$$

The first NDAT elements of vectors $\underline{U}$ and $\underline{V}$ involve constituents of type (a) (datum biases). Let $INTA_i$ and $INTB_i$ be the first and last subintervals referred to the ith datum. Then

$$U_i = \frac{1}{\sin Q} \sum_{j=INTA_i}^{INTB_i} \sin N_jQ \cos L_jQ$$

$$i=1,2,\ldots,NDAT \quad . \quad (52)$$

$$V_i = \frac{1}{\sin Q} \sum_{j=INTA_i}^{INTB_i} \sin N_jQ \sin L_jQ$$

If LT ≠ 0, the next element of $\underline{U}$ and $\underline{V}$ involves the known constituent of type (b) (linear trend). Then

$$U_{NDAT+1} = \frac{\partial}{\partial\omega} <1,\ \sin\omega T> = \frac{\partial Q}{\partial\omega}\frac{\partial}{\partial Q}\left(\frac{1}{\sin Q}\sum_{j=1}^{NIVL} \sin N_jQ \sin L_jQ\right) \quad (53)$$

$$V_{NDAT+1} = -\frac{\partial}{\partial\omega} <1,\ \cos\omega T> = -\frac{\partial Q}{\partial\omega}\frac{\partial}{\partial Q}\left(\frac{1}{\sin Q}\sum_{j=1}^{NIVL} \sin N_jQ \cos L_jQ\right) \quad .(54)$$

After some development, we get

$$U_{NDAT+1} = \frac{STEP}{2}\frac{1}{\sin Q}\sum_{j=1}^{NIVL} -\cot Q \sin N_jQ \sin L_jQ + N_j \cos N_jQ \sin L_jQ + L_j \sin N_jQ \cos L_jQ \quad (55)$$

$$V_{NDAT+1} = \frac{STEP}{2}\frac{1}{\sin Q}\sum_{J=1}^{NIVL} +\cot Q \sin N_jQ \cos L_jQ - N_j \cos N_jQ \cos L_jQ + L_j \sin N_jQ \sin L_jQ \quad .(56)$$

The next (2*NPER) elements of $\underline{U}$ and $\underline{V}$ involve constituents of type (c) (forced periods). Using (46), it is easy to see that:

$$U_i = \frac{1}{2}\langle 1, \cos(\mu_k + \omega)T\rangle + \frac{1}{2}\langle 1, \cos(\mu_k - \omega)T\rangle \quad (57)$$

$$U_{i+1} = \frac{1}{2}\langle 1, \sin(\mu_k + \omega)T\rangle + \frac{1}{2}\langle 1, \sin(\mu_k - \omega)T\rangle \quad (58)$$

$$V_i = \frac{1}{2}\langle 1, \sin(\mu_k + \omega)T\rangle - \frac{1}{2}\langle 1, \sin(\mu_k - \omega)T\rangle \quad (59)$$

$$V_{i+1} = -\frac{1}{2}\langle 1, \cos(\mu_k + \omega)T\rangle + \frac{1}{2}\langle 1, \cos(\mu_k - \omega)T\rangle \quad (60)$$

$i = NDAT + LT + 2*k-1$

$k = 1,2,\ldots,NPER.$

Letting $P_k = (\mu_k/2)*STEP$, we see from (50) that

$$\langle 1, \text{trig}(\mu_k \pm \omega)T\rangle = \frac{1}{\sin(P_k \pm Q)} \sum_{j=1}^{NIVL} \text{sinN}_j(P_k \pm Q)\ \text{trigL}_j(P_k \pm Q). \quad (61)$$

We note that the functions of sums of angles in (51) and (61) can be expressed in terms of functions of angles only. Hence the scalar products CC, CS, SS and those elements of $\underline{U}$, $\underline{V}$ which refer to known constituents of types (a), (b), and (c) can be computed from the (2*NPER + 4*NPER*NIVL) functions trig(P_k), trig(N_jP_k), trig(L_jP_k) (which need only be computed once for a given $\hat{\underline{\Phi}}$) and from the (2 + 4*NIVL) functions trigQ, trig(N_jQ), trig(L_jQ) (which must be computed for each desired spectral frequency ω).

EXAMPLES

As a model of many time series encountered in practice, we have generated the following time series:

$$f(t) = c_i + 0.01t + 3*\exp(-t/25) + \sum_{j=1}^{5} (a_j\cos\mu_j t + b_j\sin\mu_j t)\ , \quad (62)$$

(where t is in years) that may represent a typical, say geophysical, (coloured) time series. Three hundred values of f were generated spanning 50 years and grouped into four subintervals consisting of equally spaced data, that is $t \in D_k$, k=1,2,3,4, where

$D_1 \equiv [\ 0.1,\ 0.2, \ldots, 10.0]$ years (100 values)
$D_2 \equiv [20.1, 20.2, \ldots, 25.0]$ years (50 values)
$D_3 \equiv [28.1, 28.2, \ldots, 40.0]$ years (120 values)
$D_4 \equiv [47.1, 47.2, \ldots, 50.0]$ years (30 values) .

The datum biases were $c_1 = 1$, $t \in D_1$; $c_2 = -1$, $t \in D_2$; and $c_3 = 3$, $t \in D_3$, D4. The amplitudes and periods of the trigonometric terms were $a_j = 1/2$, 1, 0, 1.2, - 1.4; $b_j = 1$, 1/2, 1, -1, 0; and $p_j = 2\pi/\mu_j = (2.759, 3.636, 5.714, 40, 16)$ years. The graph of this time series is shown in Figure 6.

The time series (62) was analysed using both SPECEQ and SPECUN. In addition, a second unequally spaced time series was generated from (62) by adding to the linearly increasing t a sinusoidal variation of period 50 years and amplitude 0.5 years. The second time series was analysed using SPECUN only.

Nine runs were made for each of these three analyses, increasing the number of known constituents from zero to 15. The SPECEQ results are shown in Figure 7. The top four spectra illustrate the influence the datum biases and the linear and exponential trends, and their removal, have on the spectra. The next four spectra illustrate how the technique can be used in searching for hidden periodicities. By suppressing the effect of the periodic constituent which was the most prominent in the previous run, we enhance the remaining peaks, revealing the existence of "weaker" periodic constituents. The ninth run suppressed the effect of all constituents of (62), in which case the residual time series consisted of round-off error only, no spectrum was computed, and, as expected, the computed amplitudes of the known constituents agreed within round-off with those used in (62). The SPECUN generated results were identical to the SPECEQ generated results. As expected for the unequally spaced SPECUN results, the computed amplitudes of the known constituents and the height of the spectral peaks differed slightly from the equally spaced analyses. However, there was no shift in the position of the spectral peaks.

The execution times of Table 1 were obtained using the FORTRAN 77 compiler on an IBM 3081 computer. The equally spaced SPECEQ and unequally spaced SPECUN execution times were essentially equal for small numbers of

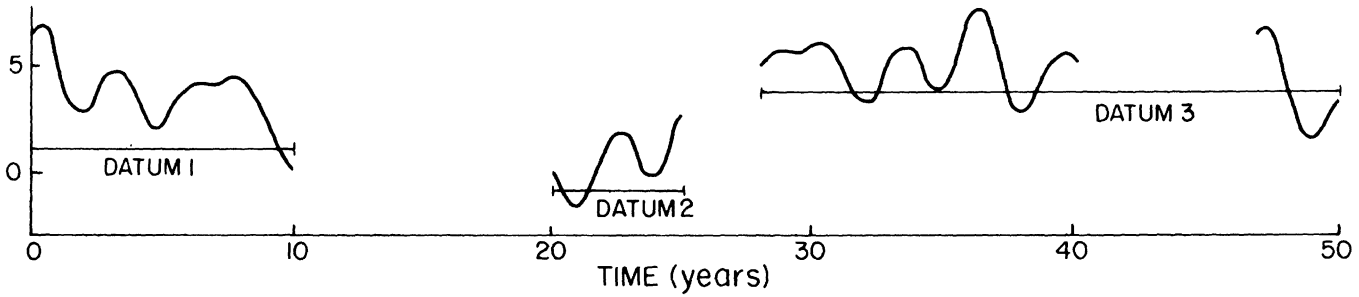


Figure 6. Graph of test time series of 300 values generated by Equation (62).

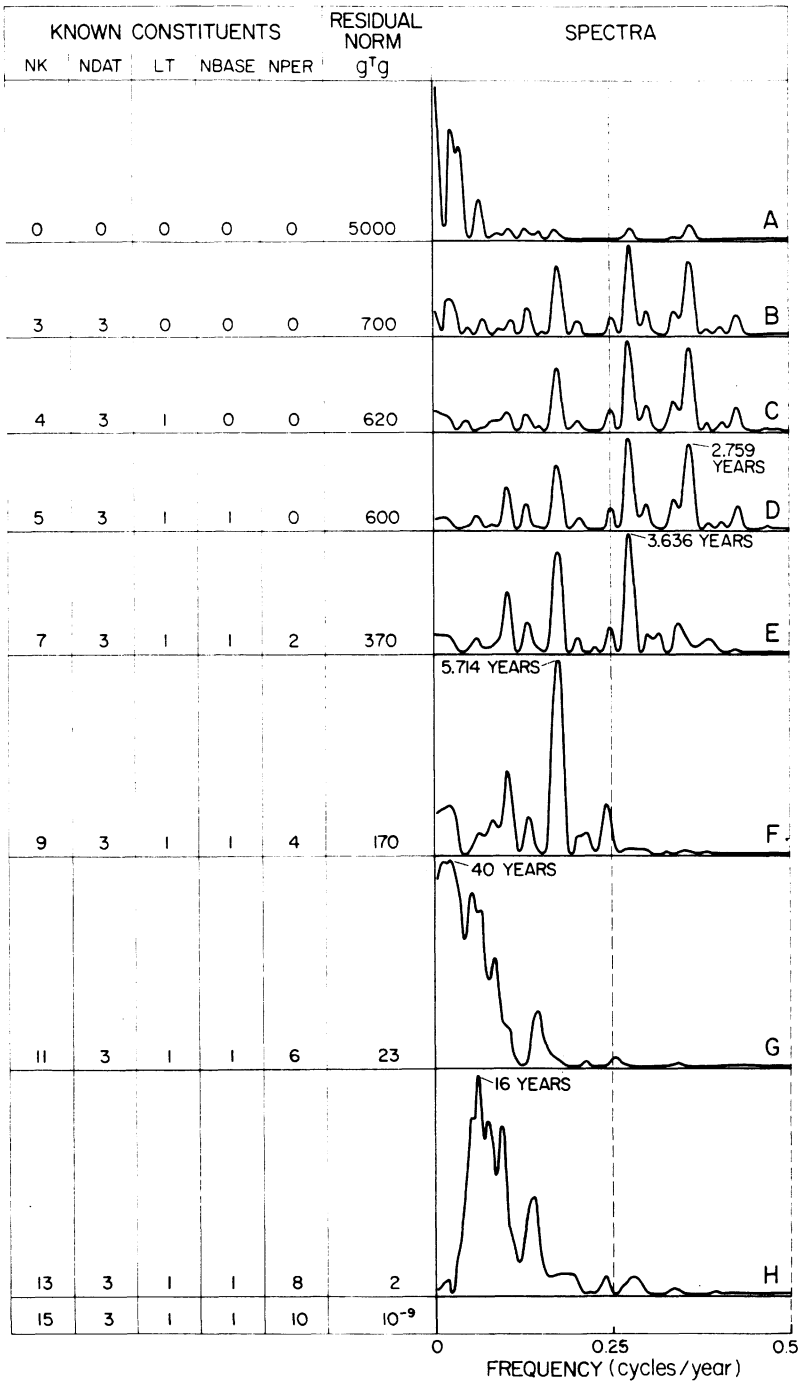


Figure 7. Results from nine runs of program SPECEQ, analyzing the time series of Figure 6. The number of known constituents suppressed in each run are specified by NK, NDAT, LT, NBASE, and NPER, respectively, giving the total number of constituents, the number of datum biases, the linear trend, the number of user-defined constituents, and the number of forced periods.

TABLE 1

IBM 3081 CPU Times for Test Time Series Containing 300 Values.

Number of Known Constituents NK	CPU Times (sec)	
	SPECEQ	SPECUN
0	5.77	5.83
3	5.82	6.95
4	6.04	7.32
5	6.72	8.19
7	6.90	10.23
9	7.04	12.19
11	7.28	14.22
13	7.53	16.35

constituents. However, for longer time series and larger numbers of constituents the difference in execution times increases considerably in favour of equally spaced execution time.

REFERENCES

Bendat, J.S. and A.G. Piersol (1971). Random Data: Analysis and Measurement Procedures. Wiley, New York.

Korn, G.A., and T.M. Korn (1968). Mathematical Handbook for Scientists and Engineers. 2nd ed., McGraw-Hill, Toronto.

Kreyszig (1978). Introductory Functional Analysis with Applications. Wiley.

Luenberger (1969). Optimization by Vector Space Methods. Wiley.

Maul, G.A. and A. Yanaway (1978). "Deep sea tides determination from GEOS-3." NASA Contractor Report 141435, NOAA Atlantic Oceanographic and Meteorological Laboratories, Miami, FL.

Oden, J.T. (1979). Applied Functional Analysis. Prentice-Hall.

Steeves, R.R. (1981). "A statistical test for significance of peaks in the least squares spectrum." Collected Papers, Geodetic Survey, Department of Energy, Mines and Resources. Surveys and Mapping Branch, Ottawa, pp. 149-166.

Taylor, J. and S. Hamilton (1972). "Some tests of the Vanicek method of spectral analysis." Astrophysics and Space Science, 17, pp. 357-367.

Vanicek, P. (1971). "Further development and properties of the spectral analysis by least squares." Astrophysics and Space Science, 12, pp. 10-73.

Vanicek, P. and E.J. Krakiwsky (1982). Geodesy: The Concepts. North Holland, Amsterdam.

Vanicek, P. and D.E. Wells (1972). "The least-squares approximation and related topics." Department of Surveying Engineering Lecture Notes 22, University of New Brunswick, Fredericton, N.B., Canada.

Wells, D.E. and E.J. Krakiwsky (1971). "The method of least-squares." Department of Surveying Engineering Lecture Notes 18, University of New Brunswick, Fredericton.

PART B

USER'S GUIDE AND PROGRAM LISTINGS

INTRODUCTION

This version of the Least-Squares Spectral Analysis software has been modified from the version published with the original version of this report. Some modifications were made to correct errors in the original version. Other modifications were made to expand the information provided on output.

STRUCTURE OF THE SOFTWARE

The software has been modularized into 16 routines, shown in Figure 1. Three of these specify the input:

TSPEC Main program. Calls TIMSER, DRIVER and FPLOT.

TIMSER Reads input time series.

DRIVER Calls SPECUN or SPECEQ.

Five of these compute the known constituents, the spectrum, and the residual time series

SPECUN Computes least squares spectrum of unequally-spaced time series.

SPECEQ Computes least squares spectrum of equally-spaced time series.

BASE Computes known constituent functional values.

RESID Computes residual time series after removing known constituents.

CHOLS Inverts matrix in place using Cholesky decomposition.

EPS Determines smallest ε such that $1 + \varepsilon$ is distinguishable from 1.

Four of these report the results on the lineprinter:

FPLOT Plots input time series.

AMPL Lists sine and cosine least-squares estimated coefficients of known constituents.

AMPHAS Lists least-squares estimated amplitude and phase (and their standard deviations) of known constituents.

COVAR Lists covariance matrix of unknown constituents.

SPLOT Plots output spectrum.

ERROR Prints error message. Stops if fatal error.

The central routine is either SPECUN or SPECEQ. Both have the following parameter list:

T = input vector of time series times $\{t_i\}$

F = input vector of time series values $\{f_i\}$
output vector of residual time series values $\{g_i\}$

NF = input length of T and F

FNORM = output $\underline{g}^T \underline{g}$

NK = input total number of known constituents to be removed from F

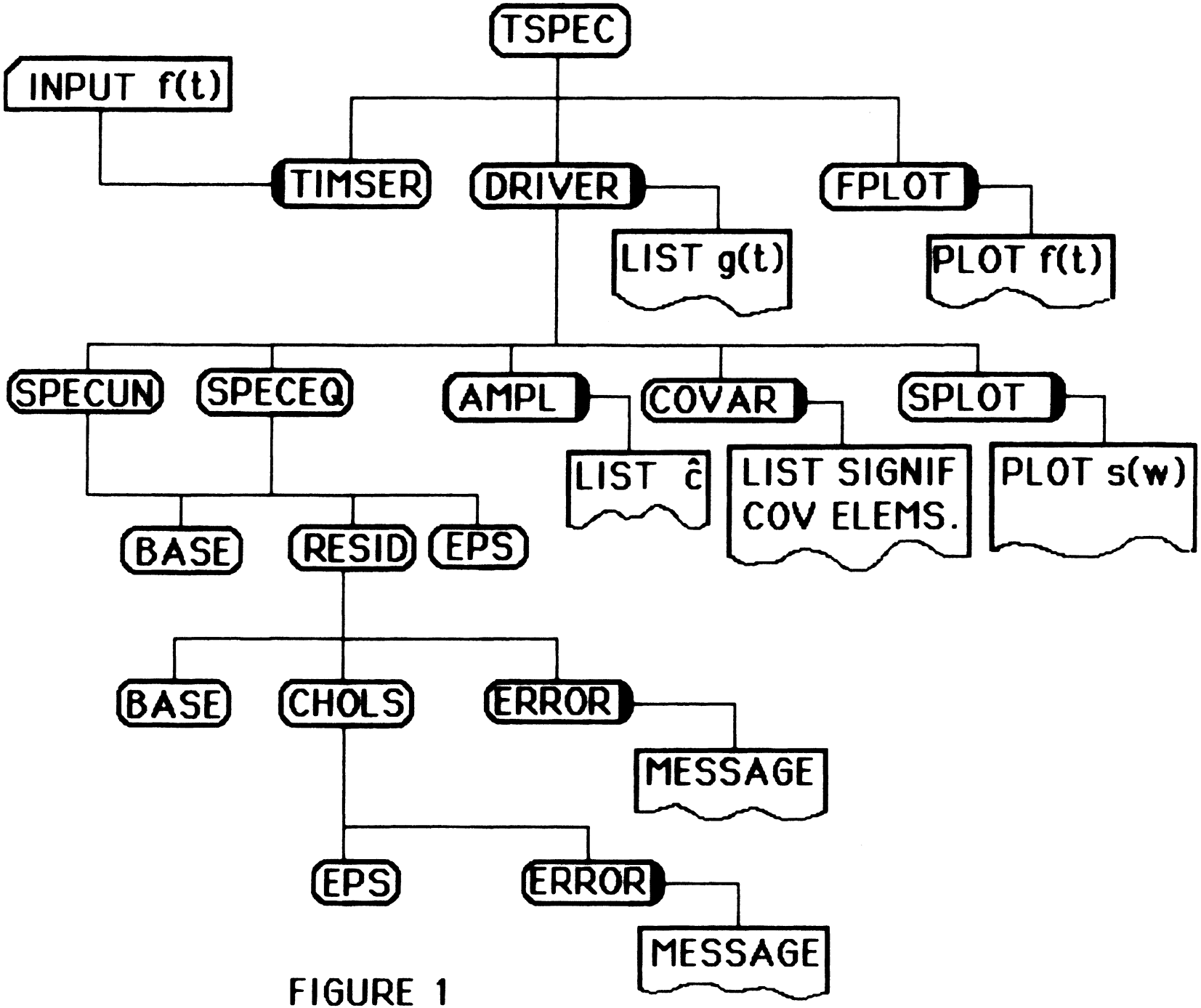

TSPEC
INPUT f(t)
TIMSER
DRIVER
FPLOT
LIST g(t)
PLOT f(t)
SPECUN
SPECEQ
AMPL
COVAR
SPLOT
LIST ĉ
LIST SIGNIF COV ELEMS.
PLOT s(w)
BASE
RESID
EPS
BASE
CHOLS
ERROR
MESSAGE
EPS
ERROR
MESSAGE

FIGURE 1

DAT = input vector of time new datum bias begins

NDAT = input number of datum biases (length of DAT)

LT = input linear trend switch

PER = input forced periods

NPER = input number of forced periods (length of PER)

NBASE = input number of user defined constituents

C = output vector of amplitudes (coefficients) of removed known constituents $\underline{\hat{c}}$. NK values.

P = input vector of periods for which spectral values will be computed

S = output vector of spectral values

NW = input length of P, S.

IB = input spectral band label. If only one spectral band is to be computed, set IB = 1. If more than one spectral band is to be computed from same time series, set IB = 1 for first band, during which $\underline{g}$ and $\underline{g}^T\underline{g}$ are computed. For subsequent bands set IB > 1, and the previous values of $\underline{g}$ and $\underline{g}^T\underline{g}$ are used, rather than recomputing.

Thus SPECUN and SPECEQ accept inputs specifying

(a) the time series $\{t_i, f_i\}$, i=1,2,...,NF

(b) the known constituents $\underline{\hat{\phi}}_i(t)$, i=1,2,...,NK to be removed from $\underline{f}$

(c) the periods P_i, i=1,2,...,NW for which spectral values are wanted

and provides outputs specifying

(a) the residual time series $\{g_i\}$ i=1,2,...,NF and its norm $\underline{g}^T\underline{g}$

(b) the amplitudes of the known constituents $\{\hat{c}_i\}$ i=1,2,...,NK

(c) the spectral values $\{s_i\}$, i=1,2,...,NW.

SPECEQ has four main blocks of code:

(a) error checking

(b) identification of equally spaced subintervals, and precomputation of trigonometric functions

(c) computation of $\underline{g}$ and $\underline{g}^T\underline{g}$ (done by subroutine RESID)

(d) computation of $s(\omega_j) = \underline{g}^T \underline{r}(\omega_j)/\underline{g}^T\underline{g}$ for each ω_j.

SPECUN omits the second of these four blocks of code.

MODIFYING THE SOFTWARE

The only routines that need be changed to accommodate new time series, known constituents, or spectral periods are

TSPEC (main)

TIMSER

DRIVER

SPECUN

BASE

(a) The only change to BASE is to add more user defined base functions, if required.

(b) The only change to SPECUN is to redimension the following arrays if there are more than 15 known constituents (NK > 15):

A(NK,NK)

B(NK)

U(NK)

V(NK)

reset NKDIM = NK

(c) The only changes to TSPEC (Main) are as follows:

Redimension FF(NF), T(NF) if NF > 500

Redimension PER(NPER) if NPER > 5

Redimension DAT(NAT) if NDAT > 3

(d) DRIVER generates the input specification of the periods for which spectral values are wanted, and passes the parameters P, NW, IB to SPECUN. If these are to be changed then the parameters PL, PS, NW, and IB in the DATA statement must be changed.

In addition, DRIVER must be changed under the following circumstances:

Redimension F(NF) for NF > 500

Redimension P(NW), S(NW) for NW > 500

Redimension C(NK) for NK > 15

(e) TIMSER generates the inputs specifications for

(i) the time series, passing T, F, NF to SPECUN

(ii) the known constituents to be removed, passing DAT, NDAT, LT, PER, NPER, NBASE, to SPECUN.

If the time series is to be read in as data, replace the DO 10 loop in TIMSER by a READ statement. If a different artificial time series is to

be generated by TIMSER, change the vectors A, B,C, P, NB, NE, IVL and the scalars NIVL and STEP to appropriate values, taking care to redimension as required.

If different known constituents are to be removed (including none to be removed) change DAT, NDAT, LT, PER, NPER, and NBASE as required. Take care to redimension DAT(NDAT) and PER(NPER) in TSPEC as required.

General redimensioning rules are, for TIMSER, to redimension

C(NK) if NK > 5

A(NPER), B(NPER), P(NPER) if NPER > 5

IVL(NDAT+1), NB(NDAT+1), NE(NDAT+1) if NDAT > 3.

```
      PROGRAM TSPEC                                                          TSPE 001
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)                                    TSPE 002
      CHARACTER *2 EQORUN                                                    TSPE 003
      CHARACTER *5 MODE                                                      TSPE 004
      DIMENSION    DAT(3), FF(500), PER(5), T(500)                           TSPE 005
      DATA         IPR  /6/                                                  TSPE 006
C                                                                            TSPE 007
C FUNCTION:  TSPEC CALLS TIMSER TO GENERATE TEST TIME SERIES                 TSPE 008
C                AND CALLS DRIVER TO COMPUTE SEVERAL                         TSPE 009
C                LEAST SQUARES SPECTRA OF THE TEST                           TSPE 010
C                TIME SERIES                                                 TSPE 011
C                                                                            TSPE 012
C UNIT NUMBER: IPR = 6 = LISTING OF INPUT AND OUTPUT                         TSPE 013
C                                                                            TSPE 014
C EXTERNALS: DRIVER,FPLOT,TIMSER                                             TSPE 015
C                                                                            TSPE 016
C SUMMARY:                                                                   TSPE 017
C   CALL TIMSER                                                              TSPE 018
C   CALL FPLOT TO PLOT INPUT TIME SERIES                                     TSPE 019
C   CALL DRIVER ADDING DATUM BIAS, LINEAR TREND, USER-DEFINED                TSPE 020
C        CONSTITUENTS AND FORCED FREQUENCIES SIMULTANEOUSLY (MODE=BATCH)     TSPE 021
C   CALL DRIVER WITH NO KNOWN CONSTITUENTS                                   TSPE 022
C   CALL DRIVER ADDING DATUM BIAS CONSTITUENTS                               TSPE 023
C   CALL DRIVER ADDING LINEAR TREND CONSTITUENT                              TSPE 024
C   CALL DRIVER ADDING USER-DEFINED CONSTITUENTS                             TSPE 025
C   CALL DRIVER ADDING ONE FORCED FREQUENCY AT A TIME                        TSPE 026
      CALL TIMSER(T, FF, NF, DAT, MDAT, MT, PER, MPER, MBASE, MODE,          TSPE 027
     $            EQORUN, PCENT, CLEVEL)                                   TSPE 028
      CALL FPLOT(T, FF, NF, DAT, MDAT, EQORUN, IPR)                          TSPE 029
      IF (MODE .EQ. 'SQNTL') GO TO 1                                         TSPE 030
      IF (MODE .EQ. 'BATCH')                                                 TSPE 031
     $CALL DRIVER(T, FF, NF, DAT, MDAT, MT, PER, MPER, MBASE, IPR,           TSPE 032
     $            EQORUN, PCENT, CLEVEL)                                   TSPE 033
      STOP                                                                   TSPE 034
    1 NDAT = 0                                                               TSPE 035
      LT = 0                                                                 TSPE 036
      NBASE = 0                                                              TSPE 037
      NPER = 0                                                               TSPE 038
      CALL DRIVER(T, FF, NF, DAT, NDAT, LT, PER, NPER, NBASE, IPR,           TSPE 039
     $            EQORUN, PCENT, CLEVEL)                                   TSPE 040
      IF(MDAT .EQ. 0) GO TO  5                                               TSPE 041
      NDAT = MDAT                                                            TSPE 042
      CALL DRIVER(T, FF, NF, DAT, NDAT, LT, PER, NPER, NBASE, IPR,           TSPE 043
     $            EQORUN, PCENT, CLEVEL)                                   TSPE 044
    5 IF(MT .EQ. 0) GO TO 10                                                 TSPE 045
      LT = MT                                                                TSPE 046
      CALL DRIVER(T, FF, NF, DAT, NDAT, LT, PER, NPER, NBASE, IPR,           TSPE 047
     $            EQORUN, PCENT, CLEVEL)                                   TSPE 048
   10 IF(MBASE .EQ. 0) GO TO 15                                              TSPE 049
      NBASE = MBASE                                                          TSPE 050
      CALL DRIVER(T, FF, NF, DAT, NDAT, LT, PER, NPER, NBASE, IPR,           TSPE 051
     $            EQORUN, PCENT, CLEVEL)                                   TSPE 052
   15 IF(MPER .EQ. 0) GO TO 25                                               TSPE 053
      DO 20 NPER = 1,MPER                                                    TSPE 054
   20   CALL DRIVER(T, FF, NF, DAT, NDAT, LT, PER, NPER, NBASE, IPR,         TSPE 055
     $              EQORUN, PCENT, CLEVEL)                                  TSPE 056
   25 STOP                                                                   TSPE 057
      END                                                                    TSPE 058
```

```
      SUBROUTINE AMPL(A, NF, NK, FNORM, DAT, NDAT, LT, PER, NPER,      AMPL 001
     $                NBASE, C, IPR)                                   AMPL 002
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)                              AMPL 003
      DIMENSION       A(100,100), C(1), DAT(1), PER(1)                 AMPL 004
      DATA PI/3.141592653589793D0/                                     AMPL 005
C                                                                      AMPL 006
C FUNCTION:  AMPL LISTS PRELIMINARY COSINE AND SINE                    AMPL 007
C                COEFFICIENTS.                                         AMPL 008
C                                                                      AMPL 009
C CALLED FROM: DRIVER                                                  AMPL 010
C                                                                      AMPL 011
C ARGUMENTS: A          = INVERTED MATRIX OF NORMAL EQUATIONS          AMPL 012
C            NK         = TOTAL NUMBER OF KNOWN CONSTITUENTS           AMPL 013
C            DAT(NDAT)  = TIMES NEW DATUM BEGINS                       AMPL 014
C            LT         = LINEAR TREND SWITCH (1 = INCLUDED)           AMPL 015
C            PER(NPER)  = FORCED PERIODS                               AMPL 016
C            NBASE      = NUMBER OF USER-DEFINED CONSTITUENTS          AMPL 017
C            C(NK)      = PRELIMINARY AMPLITUDES OF KNOWN              AMPL 018
C                         CONSTITUENTS                                 AMPL 019
C            IPR        = UNIT NUMBER FOR OUTPUT                       AMPL 020
C                                                                      AMPL 021
C EXTERNALS: DSQRT, DATA2, DMOD                                        AMPL 022
      WRITE(IPR,1001) NDAT,LT,NPER,NPER,NPER,NBASE                     AMPL 023
      IF(NDAT .GE. 1) WRITE(IPR,1002) (K,C(K),K=1,NDAT)                AMPL 024
      K = NDAT + 1                                                     AMPL 025
      IF(LT .EQ. 1) WRITE(IPR,1003) K,C(K)                             AMPL 026
      IF(NPER .EQ. 0) GO TO 10                                         AMPL 027
      DO 5 I = 1, NPER                                                 AMPL 028
        K = NDAT + LT + 2 * I - 1                                      AMPL 029
        K1 = K + 1                                                     AMPL 030
        AMP = DSQRT(C(K)*C(K) + C(K1)*C(K1))                           AMPL 031
        GPL = DATAN2(C(K1),C(K)) * 180.D0 / PI                         AMPL 032
        GPL = DMOD(GPL + 360.D0, 360.D0)                               AMPL 033
        ESTSD = DSQRT(FNORM / (NF - NK))                               AMPL 034
        SIGAMP = DSQRT(C(K) * C(K) * A(K,K) + C(K1) * C(K1) *          AMPL 035
     $             A(K1,K1) + 2.D0 * C(K) * C(K1) * A(K,K1)) *         AMPL 036
     $             ESTSD / AMP                                         AMPL 037
        SIGPL = DSQRT(C(K1) * C(K1) * A(K,K) +                         AMPL 038
     $            C(K) * C(K) * A(K1,K1) -                            AMPL 039
     $            2.D0 * C(K) * C(K1) * A(K,K1)) *                    AMPL 040
     $            (ESTSD / (AMP * AMP)) * (180.D0 / PI)               AMPL 041
        WRITE(IPR,1004) K,K1,PER(I),C(K),C(K1),AMP,SIGAMP,GPL,SIGAPL   AMPL 042
    5 CONTINUE                                                         AMPL 043
   10 IF(NBASE .EQ. 0) RETURN                                          AMPL 044
      DO 15 I = 1,NBASE                                                AMPL 045
        K = NDAT + LT + 2 * NPER + I                                   AMPL 046
   15   WRITE(IPR,1005) K, C(K)                                        AMPL 047
      RETURN                                                           AMPL 048
 1001 FORMAT(1H1,2X,31HSOLUTION FOR KNOWN CONSTITUENTS,///,            AMPL 049
     $  14X,5HDATUM,4X,6HLINEAR,4X,6HFORCED,4X,6HCOSINE,6X,            AMPL 050
     $  4HSINE,8X,4HUSER,/,15X,4HBIAS,5X,5HTREND,                      AMPL 051
     $  4X,6HPERIOD,4X,4HTERM,8X,4HTERM,5X,7HDEFINED,2X,               AMPL 052
     $  9HAMPLITUDE,3X,7H(SIGMA),4X,5HPHASE,2X,7H(SIGMA),//,           AMPL 053
     $  2X,6HNUMBER,5X,I5,5(7X,I3),//)                                 AMPL 054
 1002 FORMAT(7X,I3,E11.3)                                              AMPL 055
 1003 FORMAT(7X,I3,10X,E11.3)                                          AMPL 056
 1004 FORMAT(3X,I3,1H-,I3,18X,F11.3,2E11.3,10X,E10.3,1X,1H(,E10.3,     AMPL 057
     $       1H),1X,F6.2,1X,1H(,F6.2,1H))                              AMPL 058
 1005 FORMAT(7X,I3,51X,E11.3)                                          AMPL 059
      END                                                              AMPL 060
```

```
      DOUBLE PRECISION FUNCTION BASE(I, T, DAT, NDAT, LT, PER, NPER)     BASE 001
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)                                BASE 002
      DIMENSION     DAT(1), PER(1)                                       BASE 003
      DATA PI/3.14159265358979300/                                       BASE 004
C                                                                        BASE 005
C FUNCTION: BASE COMPUTES KNOWN CONSTITUENT FUNCTIONAL                   BASE 006
C               VALUES.                                                  BASE 007
C                                                                        BASE 008
C CALLED FROM: RESID                                                     BASE 009
C                                                                        BASE 010
C ARGUMENTS: I          = INDEX OF KNOWN CONSTITUENT TO BE COMPUTED      BASE 011
C            T          = TIME AT WHICH KNOWN CONSTITUENT COMPUTED       BASE 012
C            DAT(NDAT)  = INPUT TIMES NEW DATUM BEGINS                   BASE 013
C            LT         = INPUT LINEAR TREND SWITCH (1 = INCLUDED)       BASE 014
C            PER(NPER)  = INPUT FORCED PERIODS                           BASE 015
C                                                                        BASE 016
C EXTERNALS: DCOS, DEXP, DSIN                                            BASE 017
C                                                                        BASE 018
C LIMITATION: USER MUST SUPPLY CODING TO COMPUTE EACH                    BASE 019
C                USER-DEFINED CONSTITUENT. AS AN EXAMPLE,                BASE 020
C                THIS VERSION CONTAINS THE EXPONENTIAL                   BASE 021
C                TREND EXP(-T/25).                                       BASE 022
C                                                                        BASE 023
C DATUM BIAS                                                             BASE 024
      IF(I .GT. NDAT) GO TO 5                                            BASE 025
      BASE = 1.0D0                                                       BASE 026
      IF(I .EQ. NDAT .AND. T .GE. DAT(I)) RETURN                         BASE 027
      IF(I .LT. NDAT .AND. T .GE. DAT(I)                                 BASE 028
     $                .AND. T .LT. DAT(I+1)) RETURN                       BASE 029
      BASE = 0.0D0                                                       BASE 030
      RETURN                                                             BASE 031
C                                                                        BASE 032
C LINEAR TREND                                                           BASE 033
    5 IF(I .GT. NDAT + LT) GO TO 10                                      BASE 034
      BASE = T                                                           BASE 035
      RETURN                                                             BASE 036
C                                                                        BASE 037
C FORCED PERIODS                                                         BASE 038
   10 IF(I .GT. NDAT + LT + 2 * NPER) GO TO 20                           BASE 039
      IND = (I - NDAT - LT + 1) / 2                                      BASE 040
      IF(I - NDAT - LT .EQ. IND * 2) GO TO 15                            BASE 041
      BASE = DCOS(2.D0 * PI * T / PER(IND))                              BASE 042
      RETURN                                                             BASE 043
   15 BASE = DSIN(2.D0 * PI * T / PER(IND))                              BASE 044
      RETURN                                                             BASE 045
C                                                                        BASE 046
C EXPONENTIAL TREND                                                      BASE 047
   20 IF(I .GT. NDAT + LT + 2 * NPER + 1) GO TO 25                       BASE 048
      BASE = DEXP(-T / 25.D0)                                            BASE 049
      RETURN                                                             BASE 050
C                                                                        BASE 051
C ADD ADDITIONAL USER-DEFINED FUNCTIONS HERE                             BASE 052
   25 BASE = 0.D0                                                        BASE 053
      RETURN                                                             BASE 054
      END                                                                BASE 055
```

```
      SUBROUTINE CHOLS(A, IRDA, NA)                                          CHOL 001
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)                                    CHOL 002
      DIMENSION        A(IRDA, NA)                                           CHOL 003
      DATA             ROUND /500.DO/                                        CHOL 004
C                                                                            CHOL 005
C FUNCTION:  CHOLS INVERTS MATRIX A IN PLACE                                 CHOL 006
C                USING CHOLESKY DECOMPOSITION                                CHOL 007
C                                                                            CHOL 008
C CALLED FROM: RESID                                                         CHOL 009
C                                                                            CHOL 010
C ARGUMENTS: A(IRDA,NA) = ARRAY CONTAINING POSITIVE DEFINITE                 CHOL 011
C                         SYMMETRIC INPUT MATRIX, WITH ROW DIMENSION         CHOL 012
C                         IRDA. THE INPUT MATRIX SIZE IS (NA,NA)             CHOL 013
C                         AND IS INVERTED IN PLACE, DESTROYING THE           CHOL 014
C                         INPUT, RETURNING THE INVERSE.                      CHOL 015
C                                                                            CHOL 016
C EXTERNALS: EPS, ERROR, DSQRT                                               CHOL 017
C                                                                            CHOL 018
C ERROR CONDITIONS:                                                          CHOL 019
C   101 = FATAL. DIMENSION OF A .LT. 1                                       CHOL 020
C   102 = FATAL. NEGATIVE SQUARE ROOT. A PROBABLY SINGULAR.                  CHOL 021
C   103 = FATAL. DIAGONAL ELEMENT OF CHOLESKI DECOMPOSITION                  CHOL 022
C         NEGLIGIBLY SMALL COMPARED TO DIAGONAL ELEMENT OF A.                CHOL 023
C         A PROBABLY SINGULAR.                                               CHOL 024
C         ("NEGLIGIBLY SMALL" MEANS LESS THAN EPS*ROUND,                     CHOL 025
C         WHERE EPS IS THE SMALLEST NUMBER SO THAT                           CHOL 026
C         1. + EPS .GT. 1., AND ROUND ACCOUNTS FOR                           CHOL 027
C         ACCUMULATED ROUNDOFF)                                              CHOL 028
C                                                                            CHOL 029
C MATRIX DIMENSION CHECK                                                     CHOL 030
      IF(NA .LT. 1)                              CALL ERROR(101)             CHOL 031
C                                                                            CHOL 032
C INVERSION OF 1X1 MATRIX                                                    CHOL 033
      IF(NA .GT. 1) GO TO 5                                                  CHOL 034
      A(1,1) = 1.0DO / A(1,1)                                                CHOL 035
      RETURN                                                                 CHOL 036
C                                                                            CHOL 037
C CHOLESKI DECOMPOSITION OF INPUT MATRIX                                     CHOL 038
    5 A(1,1) = DSQRT(A(1,1))                                                 CHOL 039
      DO 10 I = 2, NA                                                        CHOL 040
   10   A(I,1) = A(I,1) / A(1,1)                                            CHOL 041
      DO 30 J = 2, NA                                                        CHOL 042
        SUM = 0.0DO                                                          CHOL 043
        DO 15 K = 2, J                                                       CHOL 044
   15     SUM = SUM + A(J,K-1) ** 2                                         CHOL 045
        IF(A(J,J) .LT. SUM)                  CALL ERROR(102)                 CHOL 046
        SUM = DSQRT(A(J,J) - SUM)                                            CHOL 047
        IF(SUM/A(J,J) .LT. EPS(ARG)*ROUND)   CALL ERROR(103)                 CHOL 048
        A(J,J) = SUM                                                         CHOL 049
        IF(J .EQ. NA) GO TO 30                                               CHOL 050
        J2 = J + 1                                                           CHOL 051
        DO 25 I = J2, NA                                                     CHOL 052
          SUM = 0.0DO                                                        CHOL 053
          DO 20 K = 2, J                                                     CHOL 054
   20       SUM = SUM + A(I,K-1) * A(J,K-1)                                 CHOL 055
   25     A(I,J) = (A(I,J) - SUM) / A(J,J)                                  CHOL 056
   30   CONTINUE                                                            CHOL 057
C                                                                            CHOL 058
C INVERSION OF LOWER TRIANGULAR MATRIX                                       CHOL 059
      DO 35 I = 1, NA                                                        CHOL 060
   35   A(I,I) = 1.0DO / A(I,I)                                             CHOL 061
```

```
      DO 45 J = 2, NA                                                    CHOL 062
        DO 45 I = J, NA                                                  CHOL 063
          SUM = 0.0D0                                                    CHOL 064
          DO 40 K = J, I                                                 CHOL 065
   40       SUM = SUM + A(I,K-1) * A(K-1,J-1)                           CHOL 066
   45     A(I,J-1) = - A(I,I) * SUM                                     CHOL 067
C                                                                        CHOL 068
C CONSTRUCTION OF INVERSE OF INPUT MATRIX                                CHOL 069
      DO 65 J = 1, NA                                                    CHOL 070
        IF(J .EQ. 1) GO TO 55                                            CHOL 071
        DO 50 I = 2, J                                                   CHOL 072
   50     A(I-1,J) = A(J,I-1)                                           CHOL 073
   55   DO 65 I = J, NA                                                 CHOL 074
          SUM = 0.0D0                                                    CHOL 075
          DO 60 K = I, NA                                                CHOL 076
   60       SUM = SUM + A(K,I) * A(K,J)                                 CHOL 077
   65     A(I,J) = SUM                                                  CHOL 078
      RETURN                                                             CHOL 079
      END                                                                CHOL 080
```

```
      SUBROUTINE COVAR(FNORM, NF, NK, A, C, PCENT, CLEVEL, IPR)          COVA 001
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)                                COVA 002
      DIMENSION A(100,1), COV(1,1), C(1)                                 COVA 003
C                                                                        COVA 004
C FUNCTION: COVAR COMPUTES THE VARIANCE-COVARIANCE MATRIX                COVA 005
C               OF THE UNKNOWN CONSTITUENTS, THE CORRELATION             COVA 006
C               MATRIX AND PRINTS RESULTS.                               COVA 007
C                                                                        COVA 008
C CALLED FROM: DRIVER                                                    COVA 009
C                                                                        COVA 010
C EXTERNALS: DSQRT, DABS                                                 COVA 011
C                                                                        COVA 012
      IF (NK .LE. 0) RETURN                                              COVA 013
C                                                                        COVA 014
C COMPUTE THE STANDARD DEVIATIONS STD, CHECK IF                          COVA 015
C   EXCEED PCENT*C(I) AND PRINT ALL OUTSTANDING                          COVA 016
C   STANDARD DEVIATIONS                                                  COVA 017
      SIGMA2 = FNORM / (NF - NK)                                         COVA 018
      WRITE (IPR,1000) PCENT                                             COVA 019
      NSTD = 0                                                           COVA 020
      DO 25 I = 1, NK                                                    COVA 021
        STD = DSQRT(SIGMA2 * A(I,I))                                     COVA 022
        IF (STD .LT. DABS(PCENT*C(I) / 100.0D0)) GO TO 25                COVA 023
        WRITE (IPR,1001) I, STD                                          COVA 024
        NSTD = NSTD + 1                                                  COVA 025
   25 CONTINUE                                                           COVA 026
      IF(NSTD .EQ. 0) WRITE (IPR, 1004)                                  COVA 027
C                                                                        COVA 028
C CHECK IF ANY CORRELATION EXCEEDS CLEVEL AND PRINT                      COVA 029
C   ALL OUTSTANDING CORRELATIONS                                         COVA 030
      WRITE(IPR,1002) CLEVEL                                             COVA 031
      NLEVEL = 0                                                         COVA 032
      DO 35 I = 1, NK                                                    COVA 033
        DO 30 J = 1, NK                                                  COVA 034
          IF(I .GE. J) GO TO 30                                          COVA 035
          COR = A(I,J) / DSQRT(A(I,I) * A(J,J))                          COVA 036
          IF(DABS(COR) .LT. CLEVEL) GO TO 30                             COVA 037
          WRITE (IPR,1003) I, J, COR                                     COVA 038
          NLEVEL = NLEVEL + 1                                            COVA 039
   30   CONTINUE                                                         COVA 040
   35 CONTINUE                                                           COVA 041
      IF(NLEVEL .EQ. 0) WRITE (IPR, 1004)                                COVA 042
 1000 FORMAT(1H1, 5X,                                                    COVA 043
     $       53HOUTSTANDING STANDARD DEVIATIONS OF KNOWN CONSTITUENTS    COVA 044
     $       ,/,5X,12H(LARGER THAN, 1X, F5.1, 1X,                       COVA 045
     $       25H% OF ESTIMATED MAGNITUDE),//, 5X, 6HNUMBER,             COVA 046
     $       2X, 18HSTANDARD DEVIATION,/)                               COVA 047
 1001 FORMAT(6X, I3, 9X, E9.3)                                           COVA 048
 1002 FORMAT(1H1, 5X,                                                    COVA 049
     $       51HOUTSTANDING CORRELATIONS BETWEEN KNOWN CONSTITUENTS,/,   COVA 050
     $       5X, 30H(LARGER IN ABSOLUTE VALUE THAN, 1X, F4.2,1H),//,     COVA 051
     $       6X,6HNUMBER, 6X, 11HCORRELATION,/)                         COVA 052
 1003 FORMAT(5X, I3, 1H-, I3, 5X, F11.8)                                 COVA 053
 1004 FORMAT(10X, 14HNONE WAS FOUND)                                     COVA 054
      END                                                                COVA 055
```

```
      SUBROUTINE DRIVER(T, FF, NF, DAT, NDAT, LT, PER, NPER, NBASE, IPR,DRIV 001
     $                  EQORUN, PCENT, CLEVEL)                          DRIV 002
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)                               DRIV 003
      CHARACTER *2      EQORUN                                          DRIV 004
      DIMENSION         A(100,100), C(100), DAT(1), F(2000), FF(1),     DRIV 005
     $                  P(2000), PER(1), S(2000), T(1)                  DRIV 006
      DATA              PL/ 200.D0/,                                    DRIV 007
     $                  PS/ 2.D0/,                                      DRIV 008
     $                  NW/125/,                                        DRIV 009
     $                  IB/  1/                                         DRIV 010
C                                                                       DRIV 011
C FUNCTION:  DRIVER CALLS SPECEQ OR SPECUN TO COMPUTE A                 DRIV 012
C                LEAST SQUARES SPECTRUM (P,S) FOR THE INPUT             DRIV 013
C                TIME SERIES (T,F).                                     DRIV 014
C                                                                       DRIV 015
C CALLED FROM: TSPEC                                                    DRIV 016
C                                                                       DRIV 017
C ARGUMENTS: T(NF)      = INPUT TIME SERIES TIMES                       DRIV 018
C            FF(NF)     = INPUT TIME SERIES VALUES                      DRIV 019
C            DAT(NDAT)  = INPUT TIMES NEW DATUM BEGINS                  DRIV 020
C            LT         = INPUT LINEAR TREND SWITCH (1 = INCLUDED)      DRIV 021
C            PER(NPER)  = INPUT FORCED PERIODS                          DRIV 022
C            NBASE      = NUMBER OF USER-DEFINED CONSTITUENTS           DRIV 023
C            IPR        = UNIT NUMBER FOR OUTPUT                        DRIV 024
C            EQORUN     = FLAG FOR EQUALLY OR UNEQUALLY SPACED SERIES   DRIV 025
C            PCENT      = PERCENTAGE LEVEL FOR DETECTING OUTSTANDING    DRIV 026
C                         STANDARD DEVIATIONS OF UNKNOWNS               DRIV 027
C            CLEVEL     = CRITICAL LEVEL FOR DETECTING OUTSTANDING      DRIV 028
C                         CORRELATIONS IN THE SOLUTION                  DRIV 029
C                                                                       DRIV 030
C EXTERNALS: AMPL, DFLOAT, SPECEQ, SPECUN, SPLOT, ERROR(108)            DRIV 031
C                                                                       DRIV 032
C SUMMARY:                                                              DRIV 033
C   COMPUTE SPECTRAL PERIODS, P                                         DRIV 034
C      PL = LONGEST PERIOD IN P                                         DRIV 035
C      PS = SHORTEST PERIOD IN P                                        DRIV 036
C      NW = NUMBER OF PERIODS IN P                                      DRIV 037
C   COPY VECTOR F (MODIFIED BY SPECEQ AND SPECUN)                       DRIV 038
C   COMPUTE NK = TOTAL NUMBER OF KNOWN CONSTITUENTS                     DRIV 039
C   CALL SPECEQ OR SPECUN TO COMPUTE SPECTRUM                           DRIV 040
C   CALL AMPL TO LIST KNOWN CONSTITUENT AMPLITUDES                      DRIV 041
C   CALL COVAR TO LIST ALL OUTSTANDING STANDARD DEVIATIONS              DRIV 042
C       AND CORRELATIONS                                                DRIV 043
C   LIST RESIDUAL TIME SERIES AND ITS QUADRATIC NORM                    DRIV 044
C   COMPUTE RS = MEAN SPECTRAL VALUE FOR WHITE NOISE                    DRIV 045
C   COMPUTE RS95 = CRITICAL PERCENTAGE VARIANCE AT 95%                  DRIV 046
C                  CONFIDENCE LEVEL FOR DETECTING STATISTICALLY         DRIV 047
C                  SIGNIFICANT PEAKS IN THE SPECTRUM                    DRIV 048
C   PLOT SPECTRUM                                                       DRIV 049
      DO 5 I = 1,NW                                                     DRIV 050
    5   P(I) = DFLOAT(NW-1)/(DFLOAT(NW-I)/PL + DFLOAT(I-1)/PS)          DRIV 051
      DO 10 I = 1,NF                                                    DRIV 052
   10   F(I) = FF(I)                                                    DRIV 053
      NK = NDAT + LT + NBASE + 2 * NPER                                 DRIV 054
      IF(EQORUN. EQ. 'EQ')                                              DRIV 055
     $CALL SPECEQ(T, F, NF, FNORM,                                      DRIV 056
     $            NK, DAT, NDAT, LT, PER, NPER, NBASE, A, C,            DRIV 057
     $            P, S, NW, IB, ICRIT)                                  DRIV 058
      IF(EQORUN .EQ. 'UN')                                              DRIV 059
     $CALL SPECUN(T, F, NF, FNORM,                                      DRIV 060
     $            NK, DAT, NDAT, LT, PER, NPER, NBASE, A, C,            DRIV 061
```

```
     $            P, S, NW, IB, ICRIT)                                    DRIV 062
      CALL AMPL(A, NF, NK, FNORM, DAT, NDAT, LT, PER, NPER,               DRIV 063
     $          NBASE, C, IPR)                                           DRIV 064
      CALL COVAR(FNORM, NF, NK, A, C, PCENT, CLEVEL, IPR)                 DRIV 065
      RS = 200.0D0 / (NF - NK)                                            DRIV 066
      RS95 = 100.0D0/(1.0D0/(0.05D0**(-2.0D0/(NF-NK-2))-1)+1)             DRIV 067
      WRITE(IPR,1001) (F(I),I=1,NF)                                       DRIV 068
      WRITE(IPR,1002) FNORM                                               DRIV 069
      IF(ICRIT .EQ. 0)                          CALL ERROR(108)           DRIV 070
      CALL SPLOT(P, S, NW, RS, RS95, IB, IPR)                             DRIV 071
      RETURN                                                              DRIV 072
 1001 FORMAT(1H1,9X,20HRESIDUAL TIME SERIES//110(11E10.2/))               DRIV 073
 1002 FORMAT(9X,35HRESIDUAL TIME SERIES QUADRATIC NORM,E15.5)             DRIV 074
      END                                                                 DRIV 075
```

```
      DOUBLE PRECISION FUNCTION EPS(ARG)                                EPS  001
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)                               EPS  002
C                                                                       EPS  003
C FUNCTION: EPS SETS FUNCTIONAL VALUE EPS AND ARGUMENT ARG              EPS  004
C               BOTH EQUAL TO THE SMALLEST NUMBER SO THAT               EPS  005
C               1. + EPS .GT. 1.                                        EPS  006
C                                                                       EPS  007
C CALLED FROM: CHOLS, SPECUN, SPECEQ                                    EPS  008
C                                                                       EPS  009
      EPS = 1.0D0                                                       EPS  010
   10 EPS = EPS / 2.0D0                                                 EPS  011
      IF ((1.0D0 + EPS) - 1.0D0 .EQ. EPS) GO TO 10                      EPS  012
      EPS = EPS + 2.0D0                                                 EPS  013
      ARG = EPS                                                         EPS  014
      RETURN                                                            EPS  015
      END                                                               EPS  016
```

```
      SUBROUTINE ERROR(IER)                                              ERRO 001
      INTEGER          IER,IPR                                           ERRO 002
      DATA             IPR /6/                                           ERRO 003
C                                                                        ERRO 004
C FUNCTION:  ERROR DETECTS WHETHER ERROR IS WARNING                      ERRO 005
C               OR FATAL, AND PRINTS MESSAGE                             ERRO 006
C                                                                        ERRO 007
C CALLED FROM: DRIVER, CHOLS, SPECUN, SPECEQ                             ERRO 008
C                                                                        ERRO 009
C ARGUMENT:  IER = ERROR INDEX                                           ERRO 010
C                WARNINGS HAVE INDICES 1 - 99                            ERRO 011
C                FATAL ERRORS HAVE INDICES 100 AND OVER.                 ERRO 012
C                                                                        ERRO 013
      IF(IER .GT. 100) GO TO 10                                          ERRO 014
      WRITE(IPR,1001) IER                                                ERRO 015
      RETURN                                                             ERRO 016
   10 WRITE(IPR,1002) IER                                                ERRO 017
      STOP                                                               ERRO 018
 1001 FORMAT(11H ***WARNING,I5)                                          ERRO 019
 1002 FORMAT(15H ***FATAL ERROR,I5)                                      ERRO 020
      END                                                                ERRO 021
```

```
      SUBROUTINE FPLOT(T, F, NF, DAT, NDAT, EQORUN, IPR)                FPLO 001
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)                               FPLO 002
      CHARACTER *1      IBLANK, ISTAR, IPLOT                            FPLO 003
      CHARACTER *2      EQORUN                                          FPLO 004
      DIMENSION         DAT(1), F(1), IPLOT(100), T(1)                  FPLO 005
      DATA              IBLANK /' '/,                                   FPLO 006
     $                  ISTAR  /'*'/                                    FPLO 007
C                                                                       FPLO 008
C FUNCTION:  FPLOT PLOTS TIME SERIES F                                  FPLO 009
C                DETECTING TIMES OF NEW DATUM BIASES                    FPLO 010
C                                                                       FPLO 011
C CALLED FROM: TSPEC                                                    FPLO 012
C                                                                       FPLO 013
C ARGUMENTS: T(NF)      = INPUT TIME SERIES TIMES                       FPLO 014
C            F(NF)      = INPUT TIME SERIES VALUES                      FPLO 015
C            DAT(NDAT)  = INPUT TIMES NEW DATUM BEGINS                  FPLO 016
C            IPR        = UNIT NUMBER FOR OUTPUT                        FPLO 017
C                                                                       FPLO 018
C EXTERNALS: DMAX1,DMIN1,IFIX                                           FPLO 019
C                                                                       FPLO 020
C SUMMARY:                                                              FPLO 021
C   INITIALIZE PLOT ARRAY                                               FPLO 022
C   COMPUTE MAXIMUM AND MINIMUM VALUES IN F                             FPLO 023
C   SCAN THROUGH TIME SERIES                                            FPLO 024
C     CHECKING FOR NEW DATUM                                            FPLO 025
C     PLOTTING TIME SERIES VALUES                                       FPLO 026
C                                                                       FPLO 027
      WRITE(IPR,1001)                                                   FPLO 028
      DO 5 I = 1,100                                                    FPLO 029
    5   IPLOT(I) = IBLANK                                               FPLO 030
      FMIN = F(1)                                                       FPLO 031
      FMAX = F(1)                                                       FPLO 032
      DO 10 I = 2,NF                                                    FPLO 033
        FMIN = DMIN1(FMIN,F(I))                                         FPLO 034
   10   FMAX = DMAX1(FMAX,F(I))                                         FPLO 035
      IDAT = 1                                                          FPLO 036
      STEP = T(2) - T(1)                                                FPLO 037
      NGAP = 0                                                          FPLO 038
      DO 20 I = 1,NF                                                    FPLO 039
        IF (I .EQ. 1 .OR. EQORUN .EQ. 'UN') GO TO 12                    FPLO 040
        IF (T(I)-T(I-1) .LT. 1.5D0*STEP) GO TO 12                       FPLO 041
        NGAP = NGAP + 1                                                 FPLO 042
        NPNT = (T(I) - T(I-1)) / STEP - 1                               FPLO 043
        WRITE (IPR, 1004) NGAP, NPNT                                    FPLO 044
   12   IF(NDAT .EQ. 0 .OR. IDAT .GT. NDAT) GO TO 15                    FPLO 045
        IF(DAT(IDAT) .GT. T(I)) GO TO 15                                FPLO 046
        WRITE(IPR,1002) IDAT                                            FPLO 047
        IDAT = IDAT + 1                                                 FPLO 048
   15   KF= 1 + (99.0D0 * (F(I) - FMIN) / (FMAX - FMIN))                FPLO 049
        IF(KF .LT. 1  ) KF = 1                                          FPLO 050
        IF(KF .GT. 100) KF = 100                                        FPLO 051
        IPLOT(KF) = ISTAR                                               FPLO 052
        WRITE(IPR,1003) I,T(I),F(I),IPLOT                               FPLO 053
   20   IPLOT(KF) = IBLANK                                              FPLO 054
      RETURN                                                            FPLO 055
 1001 FORMAT(1H1,9X,11HTIME SERIES//                                    FPLO 056
     $       5X,1HI,8X,4HT(I),8X,4HF(I)/)                               FPLO 057
 1002 FORMAT(/,30X,38(1H-),18HBEGINNING OF DATUM,I5,39(1H-),/)          FPLO 058
 1003 FORMAT(2X,I4,2E12.4,100A1)                                        FPLO 059
 1004 FORMAT(/,30X,37(1H-),5HGAP #,I4,1X,2HOF,I6,1X,6HPOINTS,38(1H-))   FPLO 060
      END                                                               FPLO 061
```

```
      SUBROUTINE RESID(T, F, NF,                                        RESI 001
     $                 NK, DAT, NDAT, LT, PER, NPER, NBASE,             RESI 002
     $                 A, B, C, NKDIM)                                  RESI 003
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)                               RESI 004
      DIMENSION       A(NKDIM,1), B(1), C(1), DAT(1), F(1),             RESI 005
     $                PER(1), T(1)                                      RESI 006
C                                                                       RESI 007
C FUNCTION:  RESID COMPUTES THE RESIDUAL TIME SERIES                    RESI 008
C                AFTER REMOVING THE KNOWN CONSTITUENTS                  RESI 009
C                                                                       RESI 010
C CALLED FROM: SPECUN, SPECEQ                                           RESI 011
C                                                                       RESI 012
C ARGUMENTS: T(NF)        = INPUT TIME SERIES TIMES                     RESI 013
C            F(NF)        = INPUT TIME SERIES VALUES                    RESI 014
C                         = OUTPUT RESIDUAL TIME SERIES VALUES          RESI 015
C            NK           = INPUT TOTAL NUMBER OF KNOWN CONSTITUENTS    RESI 016
C            DAT(NDAT)    = INPUT TIMES NEW DATUM BEGINS                RESI 017
C            LT           = INPUT LINEAR TREND SWITCH (1 = INCLUDED)    RESI 018
C            PER(NPER)    = INPUT FORCED PERIODS                        RESI 019
C            NBASE        = NUMBER OF USER-DEFINED CONSTITUENTS         RESI 020
C            A(NKDIM,NK)  = OUTPUT ARRAY CONTAINING NORMAL              RESI 021
C                           EQUATION COEFFICIENT MATRIX                 RESI 022
C            B(NK)        = OUTPUT NORMAL EQUATION KNOWN VECTOR         RESI 023
C            C(NK)        = OUTPUT NORMAL EQUATION UNKNOWN VECTOR       RESI 024
C                                                                       RESI 025
C EXTERNALS: BASE, CHOLS                                                RESI 026
C                                                                       RESI 027
C SUMMARY:                                                              RESI 028
C   CLEAR NORMAL EQUATION ARRAYS                                        RESI 029
C   CONSTRUCT NORMAL EQUATIONS FOR KNOWN CONSTITUENTS                   RESI 030
C   INVERT NORMAL EQUATION MATRIX USING CHOLESKY ALGORITHM              RESI 031
C   COMPUTE SOLUTION TO NORMAL EQUATIONS                                RESI 032
C   COMPUTE RESIDUAL TIME SERIES                                        RESI 033
C                                                                       RESI 034
      DO 5 I = 1, NK                                                    RESI 035
        B(I) = 0.0D0                                                    RESI 036
        DO 5 J = 1, NK                                                  RESI 037
    5     A(I,J) = 0.0D0                                                RESI 038
      DO 10 I = 1, NF                                                   RESI 039
        DO 10 J = 1, NK                                                 RESI 040
          FUNC = BASE(J, T(I), DAT, NDAT, LT, PER, NPER)                RESI 041
          B(J) = B(J) + FUNC * F(I)                                     RESI 042
          DO 10 K = J, NK                                               RESI 043
   10       A(K,J) = A(K,J) + FUNC *                                    RESI 044
     $                BASE(K, T(I), DAT, NDAT, LT, PER, NPER)          RESI 045
      CALL CHOLS(A, NKDIM, NK)                                          RESI 046
      DO 15 I = 1, NK                                                   RESI 047
        C(I) = 0.0D0                                                    RESI 048
        DO 15 J = 1, NK                                                 RESI 049
   15     C(I) = C(I) + A(I,J) * B(J)                                   RESI 050
      DO 20 I = 1, NF                                                   RESI 051
        DO 20 J = 1, NK                                                 RESI 052
   20     F(I) = F(I) - C(J) *                                          RESI 053
     $             BASE(J, T(I), DAT, NDAT, LT, PER, NPER)            RESI 054
      RETURN                                                            RESI 055
      END                                                               RESI 056
```

```
      SUBROUTINE SPLOT(P, S, NW, RS, RS95, IB, IPR)                    SPLO 001
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)                              SPLO 002
      CHARACTER *1      IBLANK, ISTAR, IPLOT                           SPLO 003
      DIMENSION         IPLOT(100), P(1), S(1)                         SPLO 004
      DATA              IBLANK /' '/,                                  SPLO 005
     $                  ISTAR  /'*'/                                   SPLO 006
C                                                                      SPLO 007
C FUNCTION:  SPLOT PLOTS SPECTRUM S                                    SPLO 008
C                                                                      SPLO 009
C CALLED FROM: DRIVER                                                  SPLO 010
C                                                                      SPLO 011
C ARGUMENTS: P(NW) = INPUT SPECTRAL PERIODS                            SPLO 012
C            S(NW) = INPUT SPECTRAL VALUES                             SPLO 013
C            IPR   = UNIT NUMBER FOR OUTPUT                            SPLO 014
C                                                                      SPLO 015
C EXTERNALS: IFIX                                                      SPLO 016
C                                                                      SPLO 017
C SUMMARY:                                                             SPLO 018
C   INITIALIZE PLOT ARRAY                                              SPLO 019
C   SCAN THROUGH SPECTRUM                                              SPLO 020
C     PLOTTING SPECTRAL VALUES (PERCENTAGE VARIANCES)                  SPLO 021
C                                                                      SPLO 022
      WRITE(IPR,1001) IB, NW, P(1), P(NW), RS, RS95                    SPLO 023
      DO 5 I = 1,100                                                   SPLO 024
    5   IPLOT(I) = IBLANK                                              SPLO 025
      DO 10 I = 1, NW                                                  SPLO 026
      KS=S(I)                                                          SPLO 027
        IF(KS .LT. 1) KS = 1                                           SPLO 028
        IF(KS .GT. 100) KS = 100                                       SPLO 029
        IPLOT(KS) = ISTAR                                              SPLO 030
        WRITE(IPR,1002) P(I), S(I), IPLOT                              SPLO 031
   10   IPLOT(KS) = IBLANK                                             SPLO 032
      RETURN                                                           SPLO 033
 1001 FORMAT(1H1,10X,13HSPECTRAL BAND,I5,//,                           SPLO 034
     $  10X,I5,24H SPECTRAL VALUES BETWEEN,2F14.6,//,                  SPLO 035
     $  12X,35HMEAN SPECTRAL VALUE FOR WHITE NOISE,F11.2,//,           SPLO 036
     $  12X,35HCRITICAL RERCENTAGE VARIANCE AT 95%,/,                  SPLO 037
     $  12X,35HCONFIDENCE  LEVEL   FOR   DETECTING,/,                  SPLO 038
     $  12X,35HSIGNIFICANT  PEAKS  IN THE SPECTRUM,F11.2,//,           SPLO 039
     $  10X,6HPERIOD,6X,19HPERCENTAGE VARIANCE/)                       SPLO 040
 1002 FORMAT(5X,F10.5,F12.3,100A1)                                     SPLO 041
      END                                                              SPLO 042
```

```
      SUBROUTINE TIMSER(T, F, NF, DAT, NDAT, LT, PER, NPER, NBASE,      TIMS 001
     $                  MODE, EQORUN, PCENT, CLEVEL)                    TIMS 002
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)                               TIMS 003
      CHARACTER *2      EQORUN, EQ, UN                                  TIMS 004
      CHARACTER *5      MODE, SQNTL, BATCH                              TIMS 005
      DIMENSION         X(5), Y(5), Z(5), DAT(3), F(500), IVL(4),       TIMS 006
     $                  NB(4), NE(4), P(5), PER(5), T(500)              TIMS 007
      DATA X  /  0.500D0,  1.000D0,  0.000D0,  0.500D0, -0.250D0/,      TIMS 008
     $     Y  /  1.000D0,  0.500D0,  1.000D0, -0.500D0,  0.000D0/,      TIMS 009
     $     Z  /  1.000D0, -1.000D0,  3.000D0,  0.010D0,  3.000D0/,      TIMS 010
     $     P  /  2.759D0,  3.636D0,  5.714D0, 40.000D0, 16.000D0/,      TIMS 011
     $     NB /  1,    201,    281,    471/,                            TIMS 012
     $     NE /100,    250,    400,    500/,                            TIMS 013
     $     IVL/  1,      2,      3,      3/,                            TIMS 014
     $     NIVL/ 4/,                                                    TIMS 015
     $     STEP/  0.1D0/,                                               TIMS 016
     $     PI /  3.141592653589793D0/                                   TIMS 017
      DATA EQ / 'EQ' /,                                                 TIMS 018
     $     UN / 'UN' /,                                                 TIMS 019
     $     SQNTL / 'SQNTL' /,                                           TIMS 020
     $     BATCH / 'BATCH' /                                            TIMS 021
C                                                                       TIMS 022
C                                                                       TIMS 023
C FUNCTION:  TIMSER GENERATES A TEST TIME SERIES WITH                   TIMS 024
C                3 DATUM BIASES, A LINEAR TREND,                        TIMS 025
C                SIN/COSINE TERMS FOR 5 FREQUENCIES,                    TIMS 026
C                AND AN EXPONENTIAL TREND                               TIMS 027
C                                                                       TIMS 028
C CALLED FROM: TSPEC                                                    TIMS 029
C                                                                       TIMS 030
C            F(NF)      = TEST TIME SERIES VALUES                       TIMS 031
C            DAT(NDAT)  = TIMES NEW DATUM BEGINS                        TIMS 032
C            LT         = LINEAR TREND SWITCH (1 = INCLUDED)            TIMS 033
C            PER(NPER)  = PERIODS FOR TRIGONOMETRIC TERMS               TIMS 034
C            NBASE      = NUMBER OF USER-DEFINED CONSTITUENTS           TIMS 035
C            EQORUN     = EQUAL OR UNEQUAL SPACED TIME SERIES           TIMS 036
C                         EQORUN = EQ: EQUAL SPACED TIME SERIES         TIMS 037
C                                      (SUBROUTINE SPECEQ IS USED)       TIMS 038
C                         EQORUN = UN: UNEQUAL SPACED TIME SERIES       TIMS 039
C                                      (SUBROUTINE SPECUN IS USED)       TIMS 040
C            MODE       = BATCH OR SEQUENTIAL FORCING OF UNKNOWNS       TIMS 041
C                         MODE = SQNTL: SEQUENTIAL SOLUTION             TIMS 042
C                         MODE = BATCH: BATCH SOLUTION                  TIMS 043
C            PCENT      = PERCENTAGE LEVEL FOR DETECTING                TIMS 044
C                         OUTSTANDING STANDARD DEVIATIONS OF UNKNOWNS   TIMS 045
C            CLEVEL     = CRITICAL LEVEL FOR DETECTING OUTSTANDING      TIMS 046
C                         CORRELATIONS IN THE SOLUTION                  TIMS 047
C                                                                       TIMS 048
C EXTERNALS: DCOS,DEXP,DFLOAT,DSIN                                      TIMS 049
C                                                                       TIMS 050
      NDAT = 3                                                          TIMS 051
      LT = 1                                                            TIMS 052
      NPER = 5                                                          TIMS 053
      NBASE = 1                                                         TIMS 054
      EQORUN = EQ                                                       TIMS 055
      MODE = SQNTL                                                      TIMS 056
      PCENT = 25.0D0                                                    TIMS 057
      CLEVEL = 0.50D0                                                   TIMS 058
      DO 5 I = 1,NPER                                                   TIMS 059
    5   PER(I) = P(I)                                                   TIMS 060
C                                                                       TIMS 061
```

```
C EQUAL SPACING RUN                                                     TIMS 062
      DT1 = 0.D0                                                        TIMS 063
      DT2 = 1.D0                                                        TIMS 064
C                                                                       TIMS 065
C UNEQUAL SPACING RUN                                                   TIMS 066
C     DT1 = 0.5D0                                                       TIMS 067
C     DT2 = 50.0D0                                                      TIMS 068
      NF = 0                                                            TIMS 069
      DO 10 K = 1,NIVL                                                  TIMS 070
        I1 = NB(K)                                                      TIMS 071
        I2 = NE(K)                                                      TIMS 072
        DO 10 I = I1,I2                                                 TIMS 073
          NF = NF + 1                                                   TIMS 074
          T(NF) = DFLOAT(I) * STEP +                                    TIMS 075
     $            DT1 * DSIN(PI * DFLOAT(I) / DT2)                    TIMS 076
          F(NF) = Z(IVL(K)) + Z(NDAT+1) * T(NF) +                       TIMS 077
     $            Z(NDAT+2) * DEXP(-T(NF) / 25.D0)                    TIMS 078
          DO 10 J = 1,NPER                                              TIMS 079
   10       F(NF) = F(NF) + X(J) * DCOS(2.D0 * PI * T(NF) / P(J))      TIMS 080
     $                    + Y(J) * DSIN(2.D0 * PI * T(NF) / P(J))      TIMS 081
      DAT(1) = T(1)                                                     TIMS 082
      DAT(2) = T(101)                                                   TIMS 083
      DAT(3) = T(151)                                                   TIMS 084
      RETURN                                                            TIMS 085
      END                                                               TIMS 086
```

```
      SUBROUTINE SPECEQ(T, F, NF, FNORM,                                SPCQ 001
     $                  NK, DAT, NDAT, LT, PER, NPER, NBASE, A, C,      SPCQ 002
     $                  P, S, NW, IB, ICRIT)                            SPCQ 003
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)                               SPCQ 004
      DIMENSION        A(100,100), B(100), C(1), CLP(60,50),            SPCQ 005
     $                 CNP(60,50), CP(50), DAT(1), F(1), IVL(60), P(1), SPCQ 006
     $                 PER(1), S(1), SLP(60,50), SNP(60,50), SP(50),    SPCQ 007
     $                 SPMQ(50), SPPQ(50), T(1), U(100), V(100),        SPCQ 008
     $                 XL(60), XN(60)                                   SPCQ 009
      DATA             PI/3.141592653589793D0/,                         SPCQ 010
     $                 ROUND  /100000./,                                SPCQ 011
     $                 NKDIM /100/,                                     SPCQ 012
     $                 NPERDM /50/,                                     SPCQ 013
     $                 IVLDIM /60/                                      SPCQ 014
C                                                                       SPCQ 015
C FUNCTION:  SPECEQ COMPUTES THE LEAST SQUARES SPECTRUM OF               SPCQ 016
C                A PIECEWISE EQUALLY SPACED TIME SERIES                 SPCQ 017
C                AFTER SUPPRESSING KNOWN CONSTITUENTS                   SPCQ 018
C                                                                       SPCQ 019
C CALLED FROM: DRIVER                                                   SPCQ 020
C                                                                       SPCQ 021
C ARGUMENTS:                                                            SPCQ 022
C   SPECIFYING THE INPUT TIME SERIES                                    SPCQ 023
C          T(NF) = INPUT TIME SERIES TIMES                              SPCQ 024
C          F(NF) = INPUT TIME SERIES VALUES                             SPCQ 025
C                = OUTPUT RESIDUAL TIME SERIES VALUES                   SPCQ 026
C          FNORM = OUTPUT QUADRATIC NORM OF RESIDUAL F                  SPCQ 027
C                                                                       SPCQ 028
C   SPECIFYING THE KNOWN CONSTITUENTS                                   SPCQ 029
C          NK               = INPUT TOTAL NUMBER OF KNOWN CONSTITUENTS  SPCQ 030
C          DAT(NDAT)        = INPUT TIMES NEW DATUM BEGINS              SPCQ 031
C          LT               = INPUT LINEAR TREND SWITCH (1 = USE TREND) SPCQ 032
C                                                       (0 = DO NOT USE)SPCQ 033
C          PER(NPER)        = INPUT FORCED PERIODS                      SPCQ 034
C          NBASE            = INPUT NUMBER OF USER-DEFINED CONSTITUENTS SPCQ 035
C          A(NKDIM,NKDIM) = OUTPUT NORMAL EQUATION MATRIX RESULTING     SPCQ 036
C                             FROM SUPPRESSION OF KNOWN CONSTITUENTS    SPCQ 037
C          C(NK)            = OUTPUT PRELIMINARY AMPLITUDES OF KNOWN    SPCQ 038
C                             CONSTITUENTS                              SPCQ 039
C                                                                       SPCQ 040
C   SPECIFYING THE OUTPUT SPECTRUM                                      SPCQ 041
C          P(NW) = INPUT SPECTRAL PERIODS                               SPCQ 042
C          S(NW) = OUTPUT SPECTRAL VALUES                               SPCQ 043
C          IB    = INPUT SPECTRAL BAND LABEL                            SPCQ 044
C          ICRIT = ROUNDOFF FLAG                                        SPCQ 045
C                  (1 = OK. CONTINUE ANALYSIS)                          SPCQ 046
C                  (0 = RESIDUAL TIME SERIES CONSISTS ONLY OF ROUNDOFF) SPCQ 047
C                                                                       SPCQ 048
C EXTERNALS: DABS, DMAX1, BASE, DCOS, EPS, ERROR, DFLOAT, RESID, DSIGN, SPCQ 049
C            DSIN, DSQRT                                                SPCQ 050
C                                                                       SPCQ 051
C ERROR CONDITIONS:                                                     SPCQ 052
C     1 = WARNING. ARGUMENT NDAT .LT. 0.   (SET TO 0.)                  SPCQ 053
C     2 = WARNING. ARGUMENT LT NOT 0 OR 1. (SET TO 0.)                  SPCQ 054
C     3 = WARNING. ARGUMENT NPER .LT. 0.   (SET TO 0.)                  SPCQ 055
C     4 = WARNING. ARGUMENT NBASE .LT. 0.  (SET TO 0.)                  SPCQ 056
C     5 = WARNING. ARGUMENT NK .NE. NDAT+LT+2*NPER+NBASE.               SPCQ 057
C                  (SET TO NDAT + LT + 2 * NPER + NBASE.)               SPCQ 058
C   104 = FATAL. LESS THAN 3 TIME SERIES VALUES INPUT.                  SPCQ 059
C   105 = FATAL. T ELEMENT VALUES NOT MONOTONIC INCREASING              SPCQ 060
C   106 = FATAL. NK TOO LARGE FOR DIMENSIONS OF A,B,U,V                 SPCQ 061
```

```
C                    (LIMITATION NO. 2 BELOW)                          SPCQ 062
C    107 = FATAL. DAT(1) .NE. T(1). (REQUIREMENT NO. 2 BELOW)          SPCQ 063
C    108 = FATAL. RESIDUAL TIME SERIES CONSISTS OF ROUNDOFF            SPCQ 064
C                 (NOW CALLED IN DRIVER)                               SPCQ 065
C    109 = FATAL. NPER TOO LARGE FOR DIMENSIONS OF                     SPCQ 066
C                 CLP,CNP,CP,SLP,SNP,SP,SPMQ,SPPQ.                     SPCQ 067
C    110 = FATAL. NIVL TOO LARGE FOR DIMENSIONS OF                     SPCQ 068
C                 CLP,CNP,IVL,SLP,SNP,XL,XN.                           SPCQ 069
C    111 = FATAL. PER CONTAINS FORCED PERIOD .LT. 2. * STEP.           SPCQ 070
C    112 = FATAL. P CONTAINS SPECTRAL PERIOD .LT. 2. * STEP.           SPCQ 071
C                                                                      SPCQ 072
C CALLING ROUTINE REQUIREMENTS:                                        SPCQ 073
C    1. WHEN NO KNOWN CONSTITUENTS ARE TO BE SUPPRESSED, THE           SPCQ 074
C          CALLING ROUTINE MUST PASS ZERO VALUES FOR NK,NDAT,          SPCQ 075
C          LT,NPER AND NBASE.                                          SPCQ 076
C    2. WHEN NDAT .GT. 0, THE CALLING ROUTINE MUST SET                 SPCQ 077
C          DAT(1) = T(1)                                               SPCQ 078
C    3. THE CALLING ROUTINE MUST SET                                   SPCQ 079
C          NK = NDAT + LT + 2 * NPER + NBASE.                          SPCQ 080
C    4. WHEN NBASE .GT. 0, THE USER MUST SUPPLY CODING IN              SPCQ 081
C          FUNCTION BASE TO COMPUTE EACH USER-DEFINED                  SPCQ 082
C          CONSTITUENT.                                                SPCQ 083
C    5. ON INITIAL CALL, CALLING ROUTINE MUST SET IB = 1 TO            SPCQ 084
C          COMPUTE RESIDUAL TIME SERIES. MANY SPECTRAL BANDS           SPCQ 085
C          FOR THE SAME SPECTRUM CAN THEN BE COMPUTED BY               SPCQ 086
C          SETTING IB .NE. 1, AND CALLING REPEATEDLY,                  SPCQ 087
C          CHANGING ONLY P(NW).                                        SPCQ 088
C    6. CALLING ROUTINE MUST DIMENSION ARGUMENT ARRAYS .GE.            SPCQ 089
C          T(NF),F(NF),DAT(NDAT),PER(NPER),C(NK),P(NW),S(NW).          SPCQ 090
C    7. T ELEMENT VALUES MUST CONSIST OF NIVL SUBINTERVALS,            SPCQ 091
C          EACH SUBINTERVAL CONTAINING EQUALLY SPACED DATA             SPCQ 092
C          SEPARATED BY A TIME INCREMENT STEP COMMON TO ALL            SPCQ 093
C          SUBINTERVALS. THE GAPS BETWEEN SUBINTERVALS NEED            SPCQ 094
C          NOT BE INTEGRAL MULTIPLES OF STEP. THE FIRST DATA           SPCQ 095
C          POINT MUST NOT BE ISOLATED (MUST NOT BE FOLLOWED            SPCQ 096
C          BY A GAP). T ELEMENT VALUES MUST INCREASE                   SPCQ 097
C          MONOTONICALLY. DAT, P AND PER ELEMENT VALUES MUST           SPCQ 098
C          BE IN THE SAME UNITS AS T.                                  SPCQ 099
C    8. THE FORCED PERIODS IN PER AND SPECTRAL PERIODS IN P            SPCQ 100
C          MUST BE SHORTER THAN 2 * STEP, EQUIVALENT TO THE            SPCQ 101
C          MAXIMUM INTERVAL USED IN FOURIER ANALYSIS.                  SPCQ 102
C                                                                      SPCQ 103
C LIMITATIONS:                                                         SPCQ 104
C    1. WHEN CALLED WITH IB = 1, AND NK .GT. 0, THE CONTENTS           SPCQ 105
C          OF THE TIME SERIES F IS REPLACED BY THE RESIDUAL            SPCQ 106
C          TIME SERIES VALUES.                                         SPCQ 107
C    2. WHEN NK .GT. NKDIM, A,B,U AND V MUST BE REDIMENSIONED          SPCQ 108
C          .GE. NK, AND NKDIM CHANGED TO THE NEW DIMENSION.            SPCQ 109
C    3. WHEN NPER .GT. NPERDM THEN SP,CP,SPMQ,SPPQ AND THE             SPCQ 110
C          SECOND INDEX OF CLP,SLP,CNP,SNP MUST BE                     SPCQ 111
C          REDIMENSIONED .GE. NPER, AND NPERDM CHANGED TO THE          SPCQ 112
C          NEW DIMENSION.                                              SPCQ 113
C    4. WHEN NIVL (= NDAT + NUMBER OF GAPS IN T BETWEEN DATUM          SPCQ 114
C          CHANGES) .GT. IVLDIM THEN XN,XL,IVL AND THE FIRST           SPCQ 115
C          INDEX OF CLP,SLP,CNP,SNP MUST BE REDIMENSIONED .GE.         SPCQ 116
C          NIVL, AND IVLDIM CHANGED TO THE NEW DIMENSION.              SPCQ 117
C                                                                      SPCQ 118
      IF(IB .NE. 1) GO TO 65                                           SPCQ 119
C                                                                      SPCQ 120
C PROCESS INPUT ARGUMENTS                                              SPCQ 121
C   CHECK NF .GE. 3                                                    SPCQ 122
```

```
C    CHECK T INCREASES MONOTONICALLY                                        SPCQ 123
C    COMPUTE FMAX = MAXIMUM ABSOLUTE VALUE IN F                             SPCQ 124
C    CHECK VALUES OF NDAT,LT,NPER,NBASE,AND NK                              SPCQ 125
C    CHECK DAT(1) .EQ. T(1)                                                 SPCQ 126
       IF(NF .LT. 3)                                CALL ERROR(104)        SPCQ 127
       DO 5 I = 2,NF                                                       SPCQ 128
         IF(T(I) .LE. T(I-1))                       CALL ERROR(105)        SPCQ 129
     5   CONTINUE                                                          SPCQ 130
       FMAX = DABS(F(1))                                                   SPCQ 131
       DO 10 I = 2,NF                                                      SPCQ 132
    10   FMAX = DMAX1(FMAX,DABS(F(I)))                                     SPCQ 133
       IF(NDAT .GE. 0) GO TO 15                                            SPCQ 134
       CALL ERROR(1)                                                       SPCQ 135
       NDAT = 0                                                            SPCQ 136
    15 IF(LT .EQ. 0 .OR. LT .EQ. 1 ) GO TO 20                              SPCQ 137
       CALL ERROR(2)                                                       SPCQ 138
       LT = 0                                                              SPCQ 139
    20 IF(NPER .GE. 0) GO TO 25                                            SPCQ 140
       CALL ERROR(3)                                                       SPCQ 141
       NPER = 0                                                            SPCQ 142
    25 IF(NBASE .GE. 0) GO TO 30                                           SPCQ 143
       CALL ERROR(4)                                                       SPCQ 144
       NBASE = 0                                                           SPCQ 145
    30 IF(NK .EQ. NDAT + LT + 2 * NPER + NBASE) GO TO 35                   SPCQ 146
       CALL ERROR(5)                                                       SPCQ 147
       NK = NDAT + LT + 2 * NPER + NBASE                                   SPCQ 148
    35 IF(NK .GT. NKDIM)                            CALL ERROR(106)        SPCQ 149
       IF(NDAT .GE. 1 .AND. DAT(1) .NE. T(1)) CALL ERROR(107)             SPCQ 150
       IF(NPER .GT. NPERDM)                         CALL ERROR(109)        SPCQ 151
       EPSARG = EPS(ARG)                                                   SPCQ 152
C                                                                          SPCQ 153
C COMPUTE STEPSIZE IN T                                                    SPCQ 154
       STEP = T(2) - T(1)                                                  SPCQ 155
C                                                                          SPCQ 156
C COMPUTE CRITICAL VALUE OF STEP FOR DETECTING GAPS IN T                   SPCQ 157
       STEP1 = 1.5D0 * STEP                                                SPCQ 158
C                                                                          SPCQ 159
C INITIALIZE ARGUMENTS IDAT, NIVL, NGAP, TA                                SPCQ 160
       IDAT = 1                                                            SPCQ 161
       NIVL = 0                                                            SPCQ 162
       NGAP = 0                                                            SPCQ 163
       TA = T(1)                                                           SPCQ 164
C                                                                          SPCQ 165
C FIND SUBINTERVAL BOUNDARIES (GAPS OR NEW DATUM SHIFTS) IN T              SPCQ 166
       DO 45 N = 2,NF                                                      SPCQ 167
         NEWIVL = 0                                                        SPCQ 168
         IF(N .NE. NF) GO TO 39                                            SPCQ 169
         NEWIVL = NGAP                                                     SPCQ 170
         IF(NEWIVL .EQ. 0) GO TO 45                                        SPCQ 171
         GO TO 42                                                          SPCQ 172
C                                                                          SPCQ 173
C CHECK IF THERE IS GAP AND DATUM SHIFT IN T                               SPCQ 174
    39   IF((T(N) - T(N-1)) .GT. STEP1 .AND.                               SPCQ 175
     $        DABS(DAT(IDAT+1)-T(N)) .LT. EPSARG*ROUND) GO TO 40           SPCQ 176
C                                                                          SPCQ 177
C CHECK IF THERE IS ONLY GAP IN T                                          SPCQ 178
         IF((T(N) - T(N-1)) .GT. STEP1) GO TO 41                           SPCQ 179
C                                                                          SPCQ 180
C CHECK IF THERE IS ONLY DATUM SHIFT IN T                                  SPCQ 181
         IF (DABS(DAT(IDAT+1)-T(N)) .LT. EPSARG*ROUND) GO TO 40            SPCQ 182
         IF (NEWIVL .EQ. 0) GO TO 45                                       SPCQ 183
```

```
      GO TO 42                                                                  SPCQ 184
C                                                                               SPCQ 185
C COMPUTE NEWIVL, IDAT AND NGAP WHEN THERE IS DATUM SHIFT REGARDLESS            SPCQ 186
C OF PRESENCE OF GAP IN T                                                       SPCQ 187
 40   NEWIVL = NGAP + 1                                                        SPCQ 188
      DAT(IDAT+1) = T(N)                                                       SPCQ 189
      IDAT = IDAT + 1                                                          SPCQ 190
      GO TO 42                                                                 SPCQ 191
C                                                                              SPCQ 192
C COMPUTE NEWIVL, IDAT AND NGAP WHEN THERE IS ONLY GAP IN T                    SPCQ 193
 41   NEWIVL = NGAP + 1                                                        SPCQ 194
      NGAP = NGAP + 1                                                          SPCQ 195
      IF(NEWIVL .EQ. 0) GO TO 45                                               SPCQ 196
C                                                                              SPCQ 197
C COMPUTE XL, XN, IVL FOR EACH SUBINTERVAL IN T                                SPCQ 198
 42   TB = T(N-1)                                                              SPCQ 199
      IF (N .EQ. NF) TB = T(NF)                                                SPCQ 200
      NIVL = NIVL + 1                                                          SPCQ 201
      IF(NIVL .GT. IVLDIM)                    CALL ERROR(110)                  SPCQ 202
      XN(NIVL) = 1.0D0 + (TB - TA) / STEP                                      SPCQ 203
      XL(NIVL) =         (TB + TA) / STEP                                      SPCQ 204
      IVL(NIVL) = IDAT - NEWIVL + NGAP                                         SPCQ 205
      TA = T(N)                                                                SPCQ 206
 45   CONTINUE                                                                 SPCQ 207
      IF (NPER.LE.0) GO TO 52                                                  SPCQ 208
C                                                                              SPCQ 209
C COMPUTE TRIG(PK), TRIG(XN*PK), TRIG(XL*PK) FOR EACH SUBINTERVAL IN T         SPCQ 210
      DO 50 I = 1,NPER                                                         SPCQ 211
        IF(PER(I) .LT. 2.0D0 * STEP)          CALL ERROR(111)                  SPCQ 212
        PK = PI * STEP / PER(I)                                                SPCQ 213
        SP(I) = DSIN(PK)                                                       SPCQ 214
        CP(I) = DCOS(PK)                                                       SPCQ 215
        DO 50 J = 1,NIVL                                                       SPCQ 216
          XNPK = XN(J) * PK                                                    SPCQ 217
          XLPK = XL(J) * PK                                                    SPCQ 218
          SNP(J,I) = DSIN(XNPK)                                                SPCQ 219
          CNP(J,I) = DCOS(XNPK)                                                SPCQ 220
          SLP(J,I) = DSIN(XLPK)                                                SPCQ 221
          CLP(J,I) = DCOS(XLPK)                                                SPCQ 222
 50   CONTINUE                                                                 SPCQ 223
C                                                                              SPCQ 224
C CHECK VALUES IF P .GE. 2*STEP                                                SPCQ 225
 52   DO 55 I = 1,NW                                                           SPCQ 226
        IF(P(I) .LT. 2.0D0 * STEP)            CALL ERROR(112)                  SPCQ 227
 55   CONTINUE                                                                 SPCQ 228
C                                                                              SPCQ 229
C SUPPRESS KNOWN CONSTITUENTS                                                  SPCQ 230
C   REPLACE F WITH RESIDUAL TIME SERIES                                        SPCQ 231
C   COMPUTE QUADRATIC NORM OF F                                                SPCQ 232
C   CHECK IF RMS VALUE OF RESIDUAL F IS LESS                                   SPCQ 233
C      THAN EPS * FMAX * ROUND, WHERE                                          SPCQ 234
C      EPS = EPSARG = SMALLEST NUMBER SO 1 + EPS .GT. 1                        SPCQ 235
C      FMAX = MAXIMUM ABSOLUTE VALUE OF ORIGINAL F                             SPCQ 236
C      ROUND  ACCOUNTS FOR ACCUMULATED ROUNDOFF IN                             SPCQ 237
C      COMPUTING RESIDUAL F                                                    SPCQ 238
      IF(NK .GT. 0) CALL RESID(T, F, NF,                                       SPCQ 239
     $                         NK, DAT, NDAT, LT, PER, NPER, NBASE,            SPCQ 240
     $                         A, B, C, NKDIM)                                 SPCQ 241
      FNORM = 0.0D0                                                            SPCQ 242
      DO 60 I = 1,NF                                                           SPCQ 243
        FNORM = FNORM + F(I) ** 2                                              SPCQ 244
```

```
   60 CONTINUE                                                                  SPCQ 245
C                                                                               SPCQ 246
C CHECK IF RESIDUAL F CONSISTS OF ROUNDOFF                                      SPCQ 247
      ICRIT = 1                                                                 SPCQ 248
      IF(DSQRT(FNORM/DFLOAT(NF)) .LT.                                           SPCQ 249
     $    EPSARG*FMAX*ROUND) ICRIT = 0                                         SPCQ 250
C                                                                               SPCQ 251
C FOR EACH SPECTRAL PERIOD P(I),COMPUTE SPECTRAL VALUE S(I)                     SPCQ 252
C   COMPUTE SCALAR PRODUCTS FCOS,FSIN,CC,CS,SS,U,V                              SPCQ 253
C   COMPUTE BILINEAR FORMS UAU,UAV,VAV                                          SPCQ 254
C   COMPUTE PERCENTAGE VARIANCE S                                               SPCQ 255
   65 DO 130 I = 1,NW                                                           SPCQ 256
        OMEGA = 2.0D0 * PI / P(I)                                               SPCQ 257
        FCOS = 0.0D0                                                            SPCQ 258
        FSIN = 0.0D0                                                            SPCQ 259
        CC = 0.0D0                                                              SPCQ 260
        CS = 0.0D0                                                              SPCQ 261
        SS = 0.0D0                                                              SPCQ 262
        IF(NK .EQ. 0) GO TO 75                                                  SPCQ 263
        DO 70 J = 1,NK                                                          SPCQ 264
          U(J) = 0.0D0                                                          SPCQ 265
          V(J) = 0.0D0                                                          SPCQ 266
   70   CONTINUE                                                                SPCQ 267
   75   DO 85 J = 1,NF                                                          SPCQ 268
          WT = OMEGA * T(J)                                                     SPCQ 269
          COSWT = DCOS(WT)                                                      SPCQ 270
          SINWT = DSIN(WT)                                                      SPCQ 271
          FCOS = FCOS + F(J) * COSWT                                            SPCQ 272
          FSIN = FSIN + F(J) * SINWT                                            SPCQ 273
          IF(NBASE .EQ. 0) GO TO 85                                             SPCQ 274
          DO 80 L = 1, NBASE                                                    SPCQ 275
            K = NDAT + LT + 2 * NPER + L                                        SPCQ 276
            FUNC = BASE(K, T(J), DAT, NDAT, LT, PER, NPER)                      SPCQ 277
            U(K) = U(K) + FUNC * COSWT                                          SPCQ 278
            V(K) = V(K) + FUNC * SINWT                                          SPCQ 279
   80     CONTINUE                                                              SPCQ 280
   85   CONTINUE                                                                SPCQ 281
        Q = 0.5D0 * OMEGA * STEP                                                SPCQ 282
        SQ = DSIN(Q)                                                            SPCQ 283
        CQ = DCOS(Q)                                                            SPCQ 284
        IF(NPER .EQ. 0) GO TO 95                                                SPCQ 285
        DO 90 J = 1,NPER                                                        SPCQ 286
          SPMQ(J) = SP(J) * CQ - CP(J) * SQ                                     SPCQ 287
          IF(DABS(SPMQ(J)) .LT. EPSARG) SPMQ(J) = DSIGN(EPSARG,SPMQ(J))         SPCQ 288
          SPPQ(J) = SP(J) * CQ + CP(J) * SQ                                     SPCQ 289
          IF(DABS(SPPQ(J)) .LT. EPSARG) SPPQ(J) = DSIGN(EPSARG,SPPQ(J))         SPCQ 290
   90   CONTINUE                                                                SPCQ 291
   95   DO 115 J = 1,NIVL                                                       SPCQ 292
          XNQ = XN(J) * Q                                                       SPCQ 293
          XLQ = XL(J) * Q                                                       SPCQ 294
          SNQ = DSIN(XNQ)                                                       SPCQ 295
          CNQ = DCOS(XNQ)                                                       SPCQ 296
          SLQ = DSIN(XLQ)                                                       SPCQ 297
          CLQ = DCOS(XLQ)                                                       SPCQ 298
          CC = CC + SNQ * CNQ * CLQ * CLQ - SNQ * CNQ * SLQ * SLQ               SPCQ 299
          CS = CS + SNQ * CNQ * SLQ * CLQ                                       SPCQ 300
          IF(NK .EQ. 0) GO TO 115                                               SPCQ 301
          IF(NDAT .EQ. 0) GO TO 100                                             SPCQ 302
          K = IVL(J)                                                            SPCQ 303
          U(K) = U(K) + SNQ * CLQ / SQ                                          SPCQ 304
          V(K) = V(K) + SNQ * SLQ / SQ                                          SPCQ 305
```

```
100      IF(LT .EQ. 0) GO TO 105                                                                                SPCQ 306
         K = NDAT + 1                                                                                           SPCQ 307
         SSCS = - SNQ * SLQ * CQ / SQ                                                                           SPCQ 308
         SCCS =   SNQ * CLQ * CQ / SQ                                                                           SPCQ 309
         XNCS =   XN(J) * CNQ * SLQ                                                                             SPCQ 310
         XNCC = - XN(J) * CNQ * CLQ                                                                             SPCQ 311
         XLSC = XL(J) * SNQ * CLQ                                                                               SPCQ 312
         XLSS = XL(J) * SNQ * SLQ                                                                               SPCQ 313
         STSQ = 0.5DO * STEP / SQ                                                                               SPCQ 314
         U(K) = U(K) + STSQ * (SSCS + XNCS + XLSC)                                                              SPCQ 315
         V(K) = V(K) + STSQ * (SCCS + XNCC + XLSS)                                                              SPCQ 316
105      IF(NPER .EQ. 0) GO TO 115                                                                              SPCQ 317
         DO 110 L = 1,NPER                                                                                      SPCQ 318
           K= NDAT + LT + 2 * L - 1                                                                             SPCQ 319
           SNPMQ = (SNP(J,L) * CNQ - CNP(J,L) * SNQ) / SPMQ(L)                                                  SPCQ 320
           SNPPQ = (SNP(J,L) * CNQ + CNP(J,L) * SNQ) / SPPQ(L)                                                  SPCQ 321
           SINM  = (SLP(J,L) * CLQ - CLP(J,L) * SLQ) * SNPMQ                                                    SPCQ 322
           SINP  = (SLP(J,L) * CLQ + CLP(J,L) * SLQ) * SNPPQ                                                    SPCQ 323
           COSM  = (CLP(J,L) * CLQ + SLP(J,L) * SLQ) * SNPMQ                                                    SPCQ 324
           COSP  = (CLP(J,L) * CLQ - SLP(J,L) * SLQ) * SNPPQ                                                    SPCQ 325
           U(K)   = U(K)   + 0.5DO * ( COSP + COSM)                                                             SPCQ 326
           U(K+1) = U(K+1) + 0.5DO * ( SINP + SINM)                                                             SPCQ 327
           V(K)   = V(K)   + 0.5DO * ( SINP - SINM)                                                             SPCQ 328
           V(K+1) = V(K+1) + 0.5DO * (-COSP + COSM)                                                             SPCQ 329
110      CONTINUE                                                                                               SPCQ 330
115    CONTINUE                                                                                                 SPCQ 331
       SQCQ = SQ * CQ                                                                                           SPCQ 332
       CCSQCQ = CC / SQCQ                                                                                       SPCQ 333
       SS = 0.5DO * (DFLOAT(NF) - CCSQCQ)                                                                       SPCQ 334
       CC = 0.5DO * (DFLOAT(NF) + CCSQCQ)                                                                       SPCQ 335
       CS = CS / SQCQ                                                                                           SPCQ 336
       UAU = 0.0DO                                                                                              SPCQ 337
       UAV = 0.0DO                                                                                              SPCQ 338
       VAV = 0.0DO                                                                                              SPCQ 339
       IF(NK .EQ. 0) GO TO 125                                                                                  SPCQ 340
       DO 120 J = 1,NK                                                                                          SPCQ 341
         DO 120 K = 1,NK                                                                                        SPCQ 342
           UAU = UAU + U(J) * A(J,K) * U(K)                                                                     SPCQ 343
           UAV = UAV + U(J) * A(J,K) * V(K)                                                                     SPCQ 344
           VAV = VAV + V(J) * A(J,K) * V(K)                                                                     SPCQ 345
120    CONTINUE                                                                                                 SPCQ 346
125    S(I) = 0.0DO                                                                                             SPCQ 347
       DET = (CC-UAU) * (SS-VAV) - (CS-UAV) * (CS-UAV)                                                          SPCQ 348
       IF(DABS(DET) .LT. EPSARG) GO TO 130                                                                      SPCQ 349
       S(I) = 100.0DO * ((SS - VAV) * FCOS * FCOS  -                                                            SPCQ 350
     $            2.0DO *  (CS - UAV) * FCOS * FSIN  +                                                        SPCQ 351
     $                     (CC - UAU) * FSIN * FSIN) /                                                        SPCQ 352
     $                     (DET * FNORM)                                                                      SPCQ 353
130  CONTINUE                                                                                                   SPCQ 354
     RETURN                                                                                                     SPCQ 355
     END                                                                                                        SPCQ 356
```

```
      SUBROUTINE SPECUN(T, F, NF, FNORM,                                SPUN 001
     $                  NK, DAT, NDAT, LT, PER, NPER, NBASE, A, C,      SPUN 002
     $                  P, S, NW, IB, ICRIT)                            SPUN 003
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)                               SPUN 004
      DIMENSION         A(100,100), B(100), C(1), DAT(1), F(1), P(1),   SPUN 005
     $                  PER(1), S(1), T(1), U(100), V(100)              SPUN 006
      DATA              PI     /3.141592653589793D0/,                   SPUN 007
     $                  ROUND  /100000./,                               SPUN 008
     $                  NKDIM /100/                                     SPUN 009
C                                                                       SPUN 010
C FUNCTION:  SPECUN COMPUTES THE LEAST SQUARES SPECTRUM OF              SPUN 011
C            AN UNEQUALLY SPACED TIME SERIES                            SPUN 012
C            AFTER SUPPRESSING KNOWN CONSTITUENTS                       SPUN 013
C                                                                       SPUN 014
C CALLED FROM: DRIVER                                                   SPUN 015
C                                                                       SPUN 016
C ARGUMENTS:                                                            SPUN 017
C   SPECIFYING THE INPUT TIME SERIES                                    SPUN 018
C          T(NF) = INPUT TIME SERIES TIMES                              SPUN 019
C          F(NF) = INPUT TIME SERIES VALUES                             SPUN 020
C                = OUTPUT RESIDUAL TIME SERIES VALUES                   SPUN 021
C          FNORM = OUTPUT QUADRATIC NORM OF RESIDUAL F                  SPUN 022
C                                                                       SPUN 023
C   SPECIFYING THE KNOWN CONSTITUENTS                                   SPUN 024
C          NK              = INPUT TOTAL NUMBER OF KNOWN CONSTITUENTS   SPUN 025
C          DAT(NDAT)       = INPUT TIMES NEW DATUM BEGINS               SPUN 026
C          LT              = INPUT LINEAR TREND SWITCH (1 = USE TREND)  SPUN 027
C                                                      (0 = DO NOT USE) SPUN 028
C          PER(NPER)       = INPUT FORCED PERIODS                       SPUN 029
C          NBASE           = INPUT NUMBER OF USER-DEFINED CONSTITUENTS   SPUN 030
C          A(NKDIM,NKDIM) = OUTPUT NORMAL EQUATION MATRIX RESULTING     SPUN 031
C                            FROM SUPPRESSION OF KNOWN CONSTITUENTS     SPUN 032
C          C(NK)           = OUTPUT PRELIMINARY AMPLITUDES OF KNOWN     SPUN 033
C                            CONSTITUENTS                               SPUN 034
C                                                                       SPUN 035
C   SPECIFYING THE OUTPUT SPECTRUM                                      SPUN 036
C          P(NW) = INPUT SPECTRAL PERIODS                               SPUN 037
C                  DSIN,DSQRT                                           SPUN 038
C          S(NW) = OUTPUT SPECTRAL VALUES                               SPUN 039
C          IB    = INPUT SPECTRAL BAND LABEL                            SPUN 040
C          ICRIT = OUTPUT ROUNDOFF FLAG                                 SPUN 041
C                  (1 = OK. CONTINUE ANALYSIS)                          SPUN 042
C                  (0 = RESIDUAL TIME SERIES CONSISTS ONLY OF ROUNDOFF) SPUN 043
C                                                                       SPUN 044
C EXTERNALS: DABS, DMAX1, BASE, DCOS, EPS, ERROR, DFLOAT, RESID,        SPUN 045
C                                                                       SPUN 046
C ERROR CONDITIONS:                                                     SPUN 047
C     1 = WARNING. ARGUMENT NDAT .LT. 0.   (SET TO 0.)                  SPUN 048
C     2 = WARNING. ARGUMENT LT NOT 0 OR 1. (SET TO 0.)                  SPUN 049
C     3 = WARNING. ARGUMENT NPER .LT. 0.   (SET TO 0.)                  SPUN 050
C     4 = WARNING. ARGUMENT NBASE .LT. 0.  (SET TO 0.)                  SPUN 051
C     5 = WARNING. ARGUMENT NK .NE. NDAT+LT+2*NPER+NBASE.               SPUN 052
C                  (SET TO NDAT + LT + 2 * NPER + NBASE.)               SPUN 053
C   104 = FATAL. LESS THAN 3 TIME SERIES VALUES INPUT.                  SPUN 054
C   105 = FATAL. T ELEMENT VALUES NOT MONOTONIC INCREASING              SPUN 055
C   106 = FATAL. NK TOO LARGE FOR DIMENSIONS OF A,B,U,V                 SPUN 056
C                (LIMITATION NO. 2 BELOW)                               SPUN 057
C   107 = FATAL. DAT(1) .NE. T(1). (REQUIREMENT NO. 2 BELOW)            SPUN 058
C   108 = FATAL. RESIDUAL TIME SERIES CONSISTS OF ROUNDOFF              SPUN 059
C                (NOW CALLED IN DRIVER)                                 SPUN 060
C                                                                       SPUN 061
```

```
C CALLING ROUTINE REQUIREMENTS:                                          SPUN 062
C   1. WHEN NO KNOWN CONSTITUENTS ARE TO BE SUPPRESSED, THE              SPUN 063
C         CALLING ROUTINE MUST PASS ZERO VALUES FOR NK,NDAT,             SPUN 064
C         LT,NPER AND NBASE.                                             SPUN 065
C   2. WHEN NDAT .GT. 0, THE CALLING ROUTINE MUST SET                    SPUN 066
C         DAT(1) = T(1)                                                  SPUN 067
C   3. THE CALLING ROUTINE MUST SET                                      SPUN 068
C         NK = NDAT + LT + 2 * NPER + NBASE.                             SPUN 069
C   4. WHEN NBASE .GT. 0, THE USER MUST SUPPLY CODING IN                 SPUN 070
C         FUNCTION BASE TO COMPUTE EACH USER-DEFINED                     SPUN 071
C         CONSTITUENT.                                                   SPUN 072
C   5. ON INITIAL CALL, CALLING ROUTINE MUST SET IB = 1 TO               SPUN 073
C         COMPUTE RESIDUAL TIME SERIES. MANY SPECTRAL BANDS              SPUN 074
C         FOR THE SAME SPECTRUM CAN THEN BE COMPUTED BY                  SPUN 075
C         SETTING IB .NE. 1, AND CALLING REPEATEDLY,                     SPUN 076
C         CHANGING ONLY P(NW).                                           SPUN 077
C   6. CALLING ROUTINE MUST DIMENSION ARGUMENT ARRAYS .GE.               SPUN 078
C         T(NF),F(NF),DAT(NDAT),PER(NPER),C(NK),P(NW),S(NW).             SPUN 079
C   7. T ELEMENT VALUES ARE UNRESTRICTED AS TO SPACING, BUT              SPUN 080
C         MUST MONOTONICALLY INCREASE. P,DAT AND PER ELEMENT             SPUN 081
C         VALUES MUST BE IN THE SAME UNITS AS T.                         SPUN 082
C                                                                        SPUN 083
C LIMITATIONS:                                                           SPUN 084
C   1. WHEN CALLED WITH IB = 1, AND NK .GT. 0, THE CONTENTS              SPUN 085
C         OF THE TIME SERIES F IS REPLACED BY THE RESIDUAL               SPUN 086
C         TIME SERIES VALUES.                                            SPUN 087
C   2. WHEN NK .GT. NKDIM, A,B,U AND V MUST BE REDIMENSIONED             SPUN 088
C         .GE. NK, AND NKDIM CHANGED TO THE NEW DIMENSION.               SPUN 089
C                                                                        SPUN 090
      IF(IB .NE. 1) GO TO 65                                             SPUN 091
C                                                                        SPUN 092
C PROCESS INPUT ARGUMENTS                                                SPUN 093
C   CHECK NF .GE. 3                                                      SPUN 094
C   CHECK T INCREASES MONOTONICALLY                                      SPUN 095
C   COMPUTE FMAX = MAXIMUM ABSOLUTE VALUE IN F                           SPUN 096
C   CHECK VALUES OF NDAT,LT,NPER,NBASE,AND NK                            SPUN 097
C   CHECK DAT(1) .EQ. T(1)                                               SPUN 098
      IF(NF .LT. 3)                                 CALL ERROR(104)      SPUN 099
      DO 5 I = 2, NF                                                     SPUN 100
        IF(T(I) .LE. T(I-1))                        CALL ERROR(105)      SPUN 101
    5   CONTINUE                                                         SPUN 102
      FMAX = DABS(F(1))                                                  SPUN 103
      DO 10 I = 2, NF                                                    SPUN 104
   10   FMAX = DMAX1(FMAX, DABS(F(I)))                                   SPUN 105
      IF(NDAT .GE. 0) GO TO 15                                           SPUN 106
      CALL ERROR(1)                                                      SPUN 107
      NDAT = 0                                                           SPUN 108
   15 IF(LT .EQ. 0 .OR. LT .EQ. 1) GO TO 20                              SPUN 109
      CALL ERROR(2)                                                      SPUN 110
      LT = 0                                                             SPUN 111
   20 IF(NPER .GE. 0) GO TO 25                                           SPUN 112
      CALL ERROR(3)                                                      SPUN 113
      NPER = 0                                                           SPUN 114
   25 IF(NBASE .GE. 0) GO TO 30                                          SPUN 115
      CALL ERROR(4)                                                      SPUN 116
      NBASE = 0                                                          SPUN 117
   30 IF(NK .EQ. NDAT + LT + 2 * NPER + NBASE) GO TO 35                  SPUN 118
      CALL ERROR(5)                                                      SPUN 119
      NK = NDAT + LT + 2 * NPER + NBASE                                  SPUN 120
   35 IF(NK .GT. NKDIM)                             CALL ERROR(106)     SPUN 121
      IF(NDAT .GE. 1 .AND. DAT(1) .NE. T(1)) CALL ERROR(107)             SPUN 122
```

```
C                                                                       SPUN 123
C SUPPRESS KNOWN CONSTITUENTS                                           SPUN 124
C   REPLACE F WITH RESIDUAL TIME SERIES                                 SPUN 125
C   COMPUTE QUADRATIC NORM OF F                                         SPUN 126
C   CHECK IF RMS VALUE OF RESIDUAL F IS LESS                            SPUN 127
C      THAN EPS * FMAX * ROUND, WHERE                                   SPUN 128
C      EPS = EPSARG = SMALLEST NUMBER SO 1 + EPS .GT. 1                 SPUN 129
C      FMAX = MAXIMUM ABSOLUTE VALUE OF ORIGINAL F                      SPUN 130
C      ROUND  ACCOUNTS FOR ACCUMULATED ROUNDOFF IN                      SPUN 131
C      COMPUTING RESIDUAL F                                             SPUN 132
      IF(NK .GT. 0) CALL RESID(T, F, NF,                                SPUN 133
     $                         NK, DAT, NDAT, LT, PER, NPER, NBASE,     SPUN 134
     $                         A, B, C, NKDIM)                          SPUN 135
       FNORM = 0.0D0                                                    SPUN 136
       EPSARG = EPS(ARG)                                                SPUN 137
       DO 60 I = 1, NF                                                  SPUN 138
   60    FNORM = FNORM + F(I) * F(I)                                    SPUN 139
C                                                                       SPUN 140
C CHECK IF RESIDUAL F CONSISTS OF ROUNDOFF                              SPUN 141
       ICRIT = 1                                                        SPUN 142
       IF(DSQRT(FNORM/DFLOAT(NF)) .LT.                                  SPUN 143
     $    EPSARG*FMAX*ROUND) ICRIT = 0                                  SPUN 144
C                                                                       SPUN 145
C FOR EACH SPECTRAL PERIOD P(I),COMPUTE SPECTRAL VALUE S(I)             SPUN 146
C   COMPUTE SCALAR PRODUCTS FCOS,FSIN,CC,CS,SS,U,V                      SPUN 147
C   COMPUTE BILINEAR FORMS UAU,UAV,VAV                                  SPUN 148
C   COMPUTE PERCENTAGE VARIANCE S                                       SPUN 149
   65 DO 130 I = 1, NW                                                  SPUN 150
        OMEGA = 2.0D0 * PI / P(I)                                       SPUN 151
        FCOS = 0.0D0                                                    SPUN 152
        FSIN = 0.0D0                                                    SPUN 153
        CC = 0.0D0                                                      SPUN 154
        CS = 0.0D0                                                      SPUN 155
        SS = 0.0D0                                                      SPUN 156
        IF(NK .EQ. 0) GO TO 75                                          SPUN 157
        DO 70 J = 1, NK                                                 SPUN 158
          U(J) = 0.0D0                                                  SPUN 159
   70     V(J) = 0.0D0                                                  SPUN 160
   75   DO 85 J = 1, NF                                                 SPUN 161
          WT = OMEGA * T(J)                                             SPUN 162
          COSWT = DCOS(WT)                                              SPUN 163
          SINWT = DSIN(WT)                                              SPUN 164
          FCOS = FCOS + F(J) * COSWT                                    SPUN 165
          FSIN = FSIN + F(J) * SINWT                                    SPUN 166
          CC = CC + COSWT * COSWT                                       SPUN 167
          CS = CS + COSWT * SINWT                                       SPUN 168
          SS = SS + SINWT * SINWT                                       SPUN 169
          IF(NK .EQ. 0) GO TO 85                                        SPUN 170
          DO 80 K = 1, NK                                               SPUN 171
            FUNC = BASE(K, T(J), DAT, NDAT, LT, PER, NPER)              SPUN 172
            U(K) = U(K) + FUNC * COSWT                                  SPUN 173
   80       V(K) = V(K) + FUNC * SINWT                                  SPUN 174
   85     CONTINUE                                                      SPUN 175
        UAU = 0.0D0                                                     SPUN 176
        UAV = 0.0D0                                                     SPUN 177
        VAV = 0.0D0                                                     SPUN 178
        IF(NK .EQ. 0) GO TO 125                                         SPUN 179
        DO 120 J = 1, NK                                                SPUN 180
          DO 120 K = 1, NK                                              SPUN 181
            UAU = UAU + U(J) * A(J,K) * U(K)                            SPUN 182
            UAV = UAV + U(J) * A(J,K) * V(K)                            SPUN 183
```

```
120        VAV = VAV + V(J) * A(J,K) * V(K)                              SPUN 184
125     S(I) = 0.0D0                                                     SPUN 185
        DET = (CC-UAU) * (SS-VAV) - (CS-UAV) * (CS-UAV)                  SPUN 186
        IF(DABS(DET) .LT. EPSARG) GO TO 130                              SPUN 187
        S(I) = 100.0D0 * ((SS - VAV) * FCOS * FCOS  -                    SPUN 188
     $             2.0D0 *  (CS - UAV) * FCOS * FSIN  +                 SPUN 189
     $                      (CC - UAU) * FSIN * FSIN) /                 SPUN 190
     $                      (DET * FNORM)                               SPUN 191
130     CONTINUE                                                         SPUN 192
      RETURN                                                             SPUN 193
      END                                                                SPUN 194
```