MASTER IN COMPUTER SCIENCE

Period determination methods for unequally spaced datas

Sprimont, Anne

Award date:
1987

Awarding institution:
University of Namur

Facultés Universitaires Notre-Dame de la Paix.
Institut d'Informatique.
Namur, Belgique.

European Southern Observatory.
Garching bei München, Deutschland.

PERIOD DETERMINATION METHODS

FOR UNEQUALLY SPACED DATAS.

Mémoire présenté par
Anne Sprimont
en vue de l'obtention du titre de
Licencié et Maître en Informatique.
Promoteur : M^me^ M. Noirhomme.
Année académique 1986-1987.

Contents

Acknowledgements

I would like to thank Mister Benvenuti, head of the Space Telescope European Coordinating Facilities (ST–ECF) for the studentship offered to me.
My highest gratitude goes to Mister Defert who supported this work for its whole duration, bringing helpfull aids as well as advices and encouragements.
In the same way, I would like to thank Mister Auriere, Baade, Courvoisier and all other astronomers wich have contributed at this work by enlightening discussions; and also Mister Boucher, Manfroid and Stellingwerf for the previous software send.
Atmost, my best gratitude to Mister Ponz for the help provided during the integration into MIDAS.
And lastly, special thanks to all ST–ECF 's and ESO 's staff for their kindness and constant good attentions.

On the University side, I should like to acknowledge the help of Miss Noirhomme in the preparation of this work.

Summary

This document presents and describes as clearly as possible the different steps leading to the elaboration of an usefull and efficient software for detecting the presence and significance of a period in unequally sampled time series data.

Knowing why this type of method is necessary for astronomers (chapter 1), the user 's requirements (chapter 2), and the implemented methods (chapter 3), the different commands introduced into MIDAS can be specified (chapter 4), and the different steps of this implementation described (chapter 5).
The sixth chapter deals with some tests realized with the package and an example of execution. In chapter 7, a study of package 's performance is done, leading to possible improvments. Brief conclusions containing other fields of application and future developments are made in the eighth chapter.

More details about MIDAS 's hardware configuration and the data structure (appendix A), the user 's manual contribution (appendix B), Toolpack examples and programs (appendix C), the package 's listings (appendix D) and a bibliography are presented at the end of this document.

Presentation of the ST–ECF and the ESO

The Space Telescope European Coordinating Facility (ST--ECF) was established in 1984 by agreement between the European Space Agency (ESA) and the European Southern Observatory (ESO), with the main goal of providing the european focal point of the space telescope related activities.

The ST–ECF is active in three main areas :

- the provision of informations about the Hubble Space Telescope (HST), its scientific instruments and their modes of operation,
- the coordination of the HST–related data analysis software development in Europe,
- the establishment of the european archive which will contain a copy of the observations made with the HST.

This european organization is located in the ESO 's building in Garching near Munich (Federal Republic of Germany).

Chapter 1

Introduction

The "opportunity analysis" of this project developed for ESO 's astronomers can be structured in five points :

- first the astronomers themselves had asked this package (section 1.1),
- a search in the already existing software was made (section 1.2),
- nothing was found, and a particular development adapted to the request was necessary, this had been preceded by a biliographical research (section 1.3),
- during this research, some contacts were taken with astronomers to help us, and particular with astronomers who had already developed this type of algorithm (section 1.4),
- and so only the integration into MIDAS had to be made (section 1.5).

1.1 Astronomers 's request

The astronomers actually working with MIDAS software in ESO wanted to have a specific package of period determination.
With the more sensitive astronomical mesure instrumentation and the more refined observationnal techniques, it occured more and more frequently that objects were found variable,which were previously believed to be constant.
A natural consequence of this observation is therefore research for possible periodicities in the signal curves.

If the matter of determination periods is not specific to the astronomy, it has to be noted an important point considering the classical way of collecting data in astronomy. "Nightly acquisition rate with missing sequences due to bad observing conditions, seasonal accessibility to most of the sources, etc."[1]

[1] see [Heck-Manfroid-Mersch]

A natural consequence of this is that contraryly to classical techniques, the techniques used for the determination of period, need measurements *not equally* spaced in time.

1.2 Software 's search

After this request, before beginning to develop ourselves the methods and knowing that they are based on mathematical formulas, it was necessary to look into the different mathematical software already existing like NAG, CERN, to see if no method was implemented. Nothing was found, only the *classical* methods, that is to say, methods for equally spaced data being developed.

1.3 Biliographical research

As nothing was already implemented, it has been necessary to do a bibliographical research.
A survey of the different time series analysis was made. The results of this research are presented in the bibliography, containing astronomical and mathematical references.

This research has been relatively fruitful.
It was also necessary to do a selection. This has been done by comparing the data with the required inputs of each method and by verifying the theoretical requirements of the algorithms.
As we are 'nt astronomers, many contacts with experimented people were needed to make this selection.

1.4 Astronomers 's package

During these contacts, we have met astronomers who have already developed their own package. These persons are Mister Manfroid, Boucher and Stellingwerf.
In the same way, we received a previous software which helped us in understanding better the different steps of the analysis 's procedure.

1.5 Integration into MIDAS

After having understood the methods as well as posssible, we were abble to integrate them into MIDAS.
The different steps of this integration were to take the received software and only change the interface with the data 's structure to be in conformation with MIDAS.

The received software had been written by astronomers. We were therefore obliged to rewrite the programs, not only to incoperate the existing data structure, but also to solve some problems :

- the software was 'nt well structured,
- no coherent use of single and double precision was made,
- and some approximations were not accurate.

Chapter 2

User 's requirements

During the development of this pacakge, we were obliged to pay particular attention to the specific astronomers 's requirements.
This chapter deals the imposed constraints.

- Methods in the astronomy 's world; this means that the implemented methods must be chosen among these already developed by astronomers.
  We have 'nt the possibility to impose specific mathematical methods.
- Fiability criterion; this means that the chosen methods must respect an astronomical fiability criterion.
  For us, it is impossible to impose our fiability criterions which are more mathematical.
- MIDAS 's environment : the implemented package must be integrated into MIDAS (see 4.1), using the MIDAS ' data structures and the MIDAS 's design, and respecting the MIDAS 's philosophy.
  Whith this, we have 'nt the possibility of using our data structures and are obliged to develop particular commands in adequation with the MIDAS 's philosophy.
- Better justified methods : after the implementation of this project, we would be abble to develop other methods, with better mathematically justification.

We will see in the next chapters, the contraints imposed by these requirements.

Chapter 3

The algorithms

This chapter deals with the different algorithms themselves.
It must be noted that a certain number of methods exist, algorithms that can solve the problem of the determination periods.
According to the dimension of the project, it is necessary to make a choice.

The reasons of this choice are explained in section 3.1 And section 3.2 discusses and describes the chosen algorithms.

3.1 Bases of the choice

We have decided to implement the more performant methods.

To justify our choice, we have based it on the recommendations made by Heck, Manfroid and Mersch in two very important articles[1], where they proceed at a comparative study of period determination methods, based on simulations.
"Since their statistical properties are difficult to establish theoretically, we undertook a comparative study, of the performances of some of them by numerical simulations. We have decided to simulate typical astronomical observations and more particularly photometric measurements obtained during a typical observing run at a mission observatory located on a good site. These represent the data that astronomers use to collect for periodicity search."

"A number of methods can be found in the litterature, but in this study we shall be distinguishing only two big families of algorithms : those based on the Fourier transform, on one hand, and on the other hand the non-parametric techniques derived from the criterion inspired by Lafler and Kinman."
The algorithms retained for the study were a *Fourier Technique* analysed by Deeming[2],

[1] see [Manfroid-Heck-Mersch], see [Heck-Manfroid-Mersch]

[2] see [Deeming]

and four *non parametric algorithms* : the autocorrelation of Burki[3], Lafler and Kinman[4], Renson[5] and Stellingwerf[6].
"Altough the two last techniques tend to perform slightly better in general, the tests applied to single periodic phenomena show that non of the methods are clearly superior to the others". But, "as far as the rapidity of carrying out the computations is concerned, most of the methods are roughly equivalent, except Stellingwerf 's which is slower, due to the particular way binning the data."

We have decided to implement *three* methods.
These are that described by :

1. *Stellingwerf,*
2. *Renson,*
3. *Deeming.*

3.2 Description of the algorithms

As this document is not a mathematical study, we will make a few general comments without entering again a complete description of the methods.
The detailed presentation can be found in the quoted references.
Before giving this description, the outlines of the availables methods are explained.

3.2.1 Outlines of the available methods

A time serie can be represented by a set of pairs (x_i, t_i) $i = 1, \ldots, N$ where x is the vector of the measured data and t the time when these measurements were taken Fig 3.1.

[3]see [Burki-Maeder-Rufener]

[4]see [Lafler-Kinman]

[5]see [Renson]

[6]see [Stellingwerf]

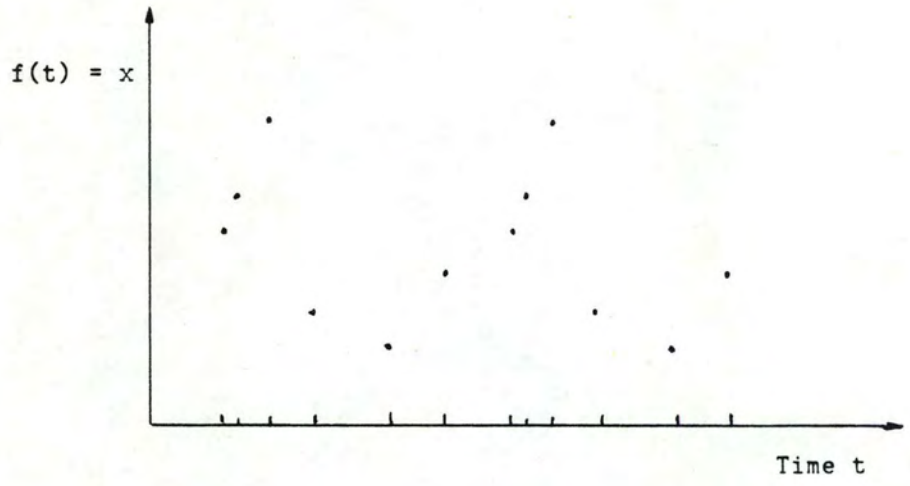


Fig 3.1 Vector of the time, vector of the measurements.

On the vector of the time, the user wants to determine a period according to the computation of a particular function of the period.
T having as unity $1/Time$, can be considered as a *period* of the function f if and only if, f applied to $(t+T)$ is equal to f applied to (t), where t is :

$$T \text{ period of } f \Leftrightarrow f(t+T) = f(t) \text{ (see Fig 3.2).}$$

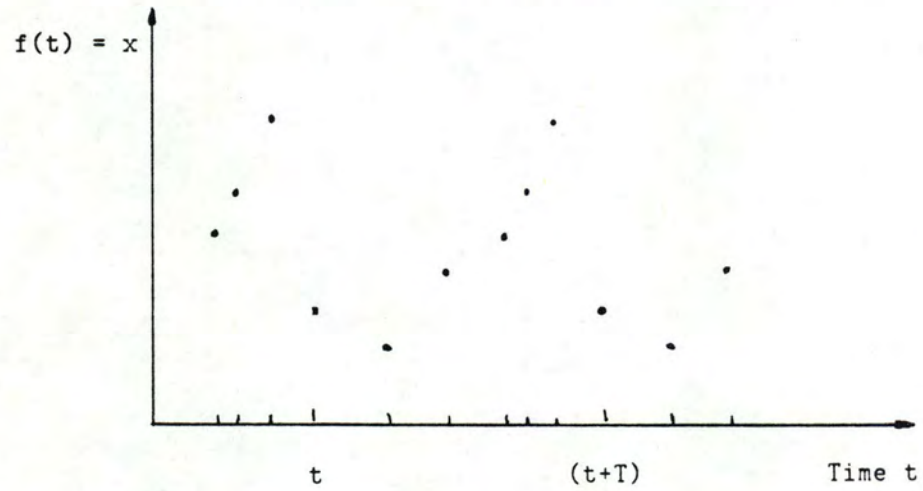


Fig 3.2 Period determinations.

As we use a discretization of the time, the definition of the period is quite different. T will be considered as the searched period if it is the minimal value containing in the interval of the periods tested that minimizes the distance between $f(t)$ and $f(t+T)$.

The average of the sample can be expressed as :

$$\overline{x} = \frac{\sum_{i=1}^{N} x_i}{N},$$

and the variance is

$$\sigma^2 = \frac{\sum_{i=1}^{N}(x_i - x)^2}{N-1}.$$

The successive steps of the analysis of time series are resumed as :

- the definition of a set of trial frequencies ν (with $\nu = 1/T$ if T is a period) based on some user knowledge of the data or on the Nyquist criteria.
- the definition of a criteria that will point out the actual candidates for the period. This can be the minimization of a statistic θ on the period range or the detection of peaks in a periodogram.
- from these candidates, the more probable period has to be extracted.

3.2.2 Stellingwerf 's method

Description

The development of this method is based on an article of R. F. Stellingwerf[7].

This method is a period determination technique well suited to the case of nonsinusoïdal time variation covered by only a few irregularly spaced observations.
This technique is simply an automated version of the classical method of distinguishing between possible periods, in wich the period producing the least observational scatter about the mean (derived) light curve is chosen.

For any subset of the measurements, we can define the sample variance s^2, with analogy to σ^2.

For any set of M distinct samples, having respectively the variances s_j^2 $j = 1, \ldots, M$ and n_j $j = 1, \ldots, M$ data points, the overall variance is given by:

$$s^2 = \frac{\sum_{j=1}^{M}(n_j - 1)\, s_j^2}{\sum_{j=1}^{M} n_j - M}.$$

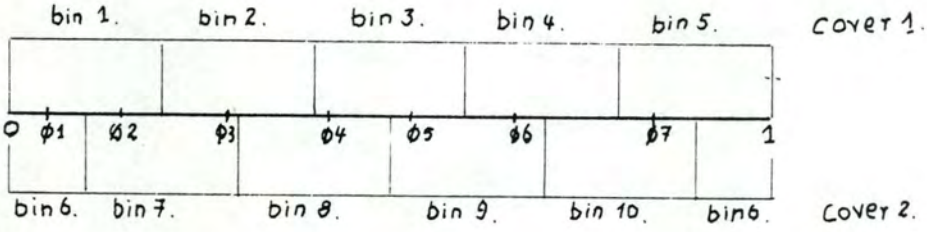
[7] see [Stellingwerf]

This technique is based on the Phase Dispersion Minimization (*PDM*).
Let T be a trial period, the phase vector ϕ is expressed as :

$$\phi_i = t_i/T - [t_i/T]$$

where $[t_i/T]$ represents the greatest integer in t_i/T.
Just remark that the ϕ_i $i = 1, \ldots, N$ lie in $[0, 1]$.

The method defines the set of samples as a structure of *bins* and *covers* in the phase interval. The unit interval is divided into N_b bins of length $1/N_b$, and N_c covers of N_b bins are taken, each cover offset in phase by $1/(N_b N_c)$ from the previous one.
Clearly, each data point will fall in exactly N_c bins.
Let us consider the following example of coverage (5,2) (in Stellingwerf 's notation) of the phase interval corresponding to a given frequency $1/T$:



giving the following composition :

- bin 1 : x_1, x_2
- bin 2 : x_3
- bin 3 : x_4, x_5
- bin 4 : x_6
- bin 5 : x_7
- bin 6 : x_1
- bin 7 : x_2, x_3
- bin 8 : x_4
- bin 9 : x_5, x_6
- bin 10 : x_7.

The variance of these samples is computed according to s^2.
This variance gives a measure of the scatter around the mean light curve defined by the means of the x_i in each sample, considered as a function of the ϕ.

The statistic θ_1 considered here is the overall variance s^2 of the bins and covers divided by the variance of the whole sample:

$$\theta_1 = \frac{s^2}{\sigma^2}.$$

The criterion is based on very simple assumptions : we wish to minimize the variance of the data to the mean light curve.
If the tested period T is not a good one, then $s^2 \approx \sigma^2$ and $\theta_1 \simeq 1$.
But, T can be considered as a correct period if θ_1 reaches a local minimum, compared with neighboring periods, hopefully with a near zero value.

F-test

The significance of the test performed on the found minima periods uses the *F-distribution*. Here we assume that the total "population" X of possible observations of an "object" is approximately normally distributed.

To compute θ_1, we take the ratio of the variances of two subsets of X, that of the actual observations x, and that of the bins and covers.
Therefore, θ_1 has a probability density distribution given by a F-distribution with respectively $\sum n_j - M$ and $N - 1$ degrees of freedom.
Normally F is defined as a number greater than the unity, so we are defining our F function as $F \equiv \frac{1}{\theta_1}$.

The probability P that a given value θ_1 is due to random fluctuation, also called the *significance* is twice the area of the F-distribution above θ_1^-1 (two sided test). This probability approaches the unity as $\theta_1 \rightsquigarrow 1$.
Thus for significance P, we compute

$$F(P/2, N_1 f, N_2 f) = 1/\theta_1$$

with $N_1 f = N - 1$ and $N_2 f = \sum n_j - M$.

P may then be obtained by reference to an F-table or by using an approximation to $P(F, N_1 f, N_2 f)$ given by M. Abramowitz and I. A. Segun[8], solution that we have chosen.

So the most probable periods have the smallest value in the F-test. Anyhow, to be affected with a correct significance, the F-test performed on the period candidates should not exceed 0.1.

[8] see [Abramowitz-Segun]

3.2.3 Renson 's method

Description

The *S*ignal *V*ariation *M*inimization method was introduced by P. Renson[9].
This described method is "intended to find the period of variable phenomenon from a given number of observations."

As the previous one, this method is based on the principle of the phase.
So for a given period T, the phase of the i-th observation is given by :

$$\phi_i = t_i/T - [t_i/T]$$

where $[t_i/T]$ is the greatest integer in t_i/T.

This method is more elaborated than Lafler 's and Kinman 's[10].
Lafler and Kinman introduce the following statistic :

$$\sum_i (x_i - x_{i-1})^2.$$

Indeed, if the tested period is good, the differences $x_i - x_{i-1}$ becomes small in mean; the points representing the measures corresponding to the ϕ_i gather around a mean curve. Nevertheless, it happens that the difference $x_i - x_{i-1}$ is important just because they correspond to very different phases.
A correction has to be introduced.

Therefore, the following statistic is defined:

$$\sum[(x_i - x_{i-1})^2/(\phi_i - \phi_{i-1})]$$

where $(i-1)$ is replaced by N if $i = 1$ and for this term 1 has to be added to the denominator.

This formula is also subject to criticism.
If different measurements x_i are corresponding to two proximal phases then even if they only differ by a quantity of order of the noise, the ratio $(x_i - x_{i-1})^2/(\phi_i - \phi_{i-1})$ will contribute too much to the sum as $(\phi_i - \phi_{i-1})$ is too small. The tested frequency could then be abnormally rejected.

To solve this problem, one can simply add to the denominator a term ε equal to the discard in phase corresponding to the noise.
If the error mean is e and the total variation of the function is r, for a more or less sinusoidal curve of period 1, ε can be estimated as :

$$\varepsilon = \frac{e}{2r}.$$

[9] see [Renson]

[10] see [Lafler-Kinman]

Finally the statistic θ_2 proposed by Renson is a weighted sum of the squares of the successive differences, i.e.

$$\theta_2 = \sum_{i=1}^{N} \frac{(x_i - x_{(i-1)})^2}{(\phi_i - \phi_{(i-1)} + v + \varepsilon)}$$

where

- $v = \begin{cases} 1 & \text{if i=1 (and(i-1)=N)} \\ 0 & \text{else} \end{cases}$
- $\varepsilon = e/2r$ where
  1. e is the precision in magnitude of the datas;
  2. $r = \max_i\ x_i - \min_i\ x_i$.

The periods minimizing the value of the statistic are good candidates.

No test

No accurate test is actually performed for this method.

3.2.4 Deeming 's method

Description

This is the Fourier Transorm based on the *D*iscrete *F*ourier *T*ransform investigated by T. J. Deeming[11].
Contrarily to classical Fast Fourier Transforms, the technique introduced here does not require measurements equally spaced in time.

The philosophy regarding to application of Fourier analysis techniques to real data corresponds the fact that one may adopt a totally phenomenological point of view in wich one simply defines the Complex Fourier Transform $F(\nu)$ of a function $f(t)$ as :

$$F(\nu) = \int_{-\infty}^{+\infty} f(t) e^{i2\pi\nu t} dt.$$

The question is then how to estimate $F(\nu)$, wich depends on an integral of $f(t)$ over $(-\infty, +\infty)$ from observations in a limited section at discrete times t_k.
It is also possible to consider a discrete version of the Fourier transform. In this case :

$$F_N(\nu) = \sum_{k=1}^{N} f(t_k) e^{i2\pi\nu t_k}.$$

[11] see [Deeming]

In the case of determinate processes, the Discrete Fourier Transform is the *convolution* of the continuous Fourier transform $F(\nu)$ with a function $\delta_N(\nu)$ called the spectral window which is obtainable as a function only of the ν and the times of the observations t_k :

$$F_N(\nu) = F(\nu) * \delta_N(\nu) \simeq \int_{-\infty}^{+\infty} F(\nu - \nu')\delta_N(\nu')d\nu'$$

with

$$\delta_N(\nu) = \sum_{k=1}^{N} e^{i2\pi\nu t_k}.$$

The symbol $\delta_N(\nu)$ is used for this quantity because in the limit of a completely filled time interval, the corresponding $\delta(\nu)$ (spectral window corresponding to the continuous Fourier transform) tends to the Dirac delta-functions as $T \to \infty$. The limit of $\delta_N(\nu)$ as $N \to \infty$ is similar to the delta function in its locating property.

In practical computations, it is most convenient to use the function $N^{-1}F_N(\nu)$ and the corresponding spectral window $\gamma_N(\nu) = N^{-1}\delta_N(\nu)$ because this yields a spectral window

$$\delta_N(0) = 1.$$

Thus

$$N^{-1}F_N(\nu) = \gamma_N(\nu) * F(\nu).$$

An important point is that "the pathology of the data distribution" is all contained in the spectral window, wich can be calculated from the data spaces alone, and dus not depend directly on the data themselves.
Thus there should be subsidiary peaks.
Furthermore by comparing the shape of any peak in the Discrete Fourier Transform with the shape of the spectral window, it is possible to judge whether a peak corresponds to a well defined delta function.

Clean

As the computed Discrete Fourier Transform is the convolution of the continuous Fourier transform and a spectral window, a non linear deconvolution according to the spectral window and using the *CLEAN* algorithm is performed.
This algorithm is explain in the article of U. J. Schwarz[12].

The Clean method is shown to be equivalent to solve a system of linear equations by an iterative method. "So far the method CLEAN has been described as a deconvolution procedure. Here, we will show that the method is especially well suited for the Fourier Transform data."

[12]see [Schwarz]

The method is used in the case that one has an obseved map, the "dirty" map d, which is the convolution of the brightness distribution t_0 with an instrumental response, called the "dirty" beam b.[13]. The dirty beam may have some unwanted secondary responses, as sidelobes, wing, etc. the aim of the method is to remove the effects of these secondary responses.
This is done in two steps.

- First a deconvolution step, in which the dirty map is decomposed in a set of scaled δ-functions, the components t, which, when convolved with the dirty beam would reproduce the original dirty map.
- Secondly the components are convolved with an hypothetical clean beam, h, which is free from the unwanted responses.
  This finaly gives the clean map c_0, which is the convolution of the true brightness distribution with the clean beam, h.

The CLEAN method is designed for the case that the brightness distribution contains only a few sources at well separated, small regions, i.e. the brightness distribution is essentielly empty. The method makes use of these characteristics in the following way : it searches for the maximum in the correlation between the dirty map and the dirty beam, which is in some applications identical to the absolute largest value in the dirty map. We can make the plausible assumption that this response is mainly due to a real signal and only a minor part comes from the filter response from other sources placed further away.

Some fraction g times the absolute largest value is accepted as a first approximation of the set of the δ-functions. A dirty beam pattern scaled by this value and centered at the corresponding position is substracted from the dirty map thereby removing a great deal of the unwanted secondary response. The procedure is repeated on the remaining map and by successive iterations one builds up the set of δ-functions.

In radio astronomy, the method CLEAN is often used in connection with interferometer and synthesis observations, where the map are derived from the Fourier Transform of the observed data.

Note : The measurements can be *segmented* in different sets. This segmentation is done in such a way that a large time interval without observations will not introduce bias.

[13]this nomenclature comes from the application of the method in radio astronomy.

For example, if we have the Fig 3.3,

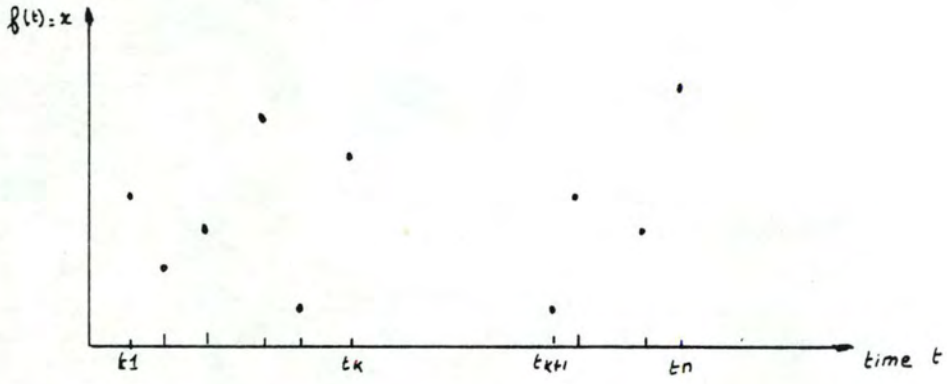


Fig 3.3 Time space.

The difference between the time t_{k-1} and t_k is to big, and can introduce errors in the computation of the statistic.

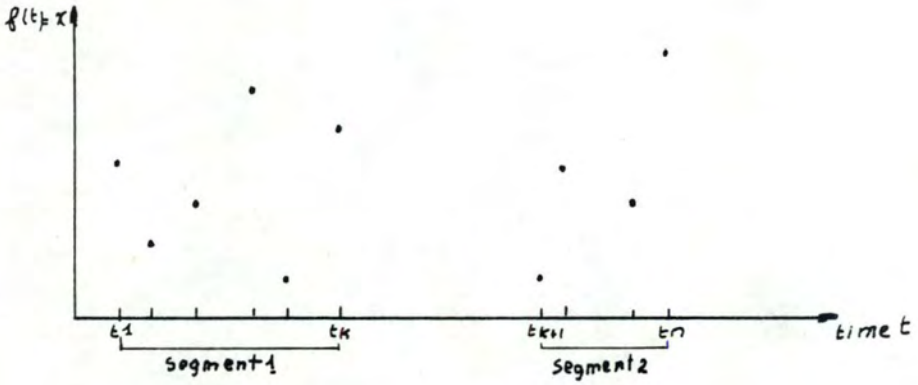


Fig 3.4 Division into segments.

Therefore, the time space is divided into *segments* Fig 3.4, and :

- $\theta(\text{whole}) = \theta(\text{segment1}) + \theta(\text{segment2})$
- Power spectrum(whole) = Power spectrum(segment1) + Power spectrum(segment2).

Chapter 4

The specifications.

As said before, the package has been developed fo the ESO 's astronomers, in the particular environment of MIDAS described in section 4.1.
In a second time, we can explain the chosen design of the user 's interface (section 4.2).

4.1 MIDAS 's environment

Only a few informations are given in this part, more details can be found in appendix A.

4.1.1 General introduction

" *MIDAS* is the acronym for *M*unich *I*mage *D*ata *A*nalysis *S*ystem"[1].

The present MIDAS 's environment is shown in Fig. 4.1.

[1] see [IPG-ug]

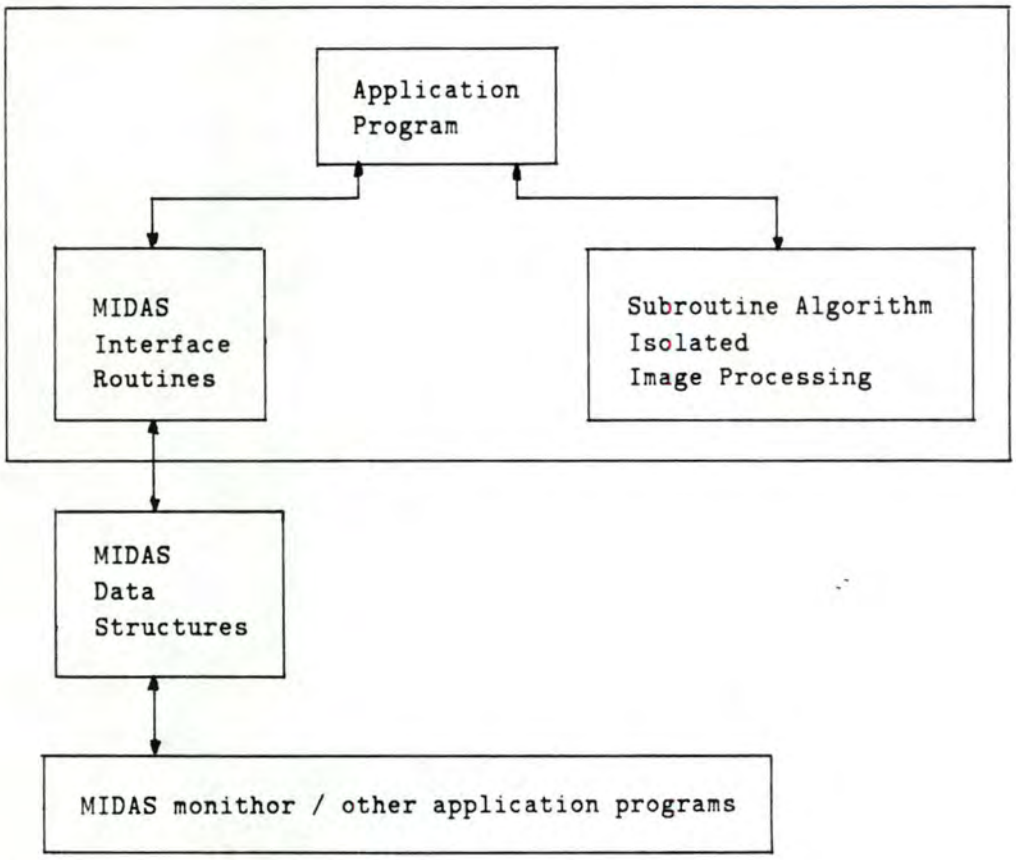


Fig. 4.1 The MIDAS 's environment.

Hardware configuration

The hardware configuration currently used is basically composed of two clustered Vaxes supermini computer supporting a given pool of terminals including image-processing workstations.

A common MIDAS 's workstation is typically composed of three different screens and of two keyboards.

Software configuration

MIDAS refers to the image processing system build on top of *VMS* operating system.

Vax Fortran is used as the programming language.
"Only some files accessing routines had to be written in MACRO in order to make use of

special RMS options not available in Fortran."[2]

The MIDAS 's system itself can be run in both *interactive and batch* modes. The interactive user can also create batch jobs running in parallel with the interactive user.

But MIDAS is a command driven system, especially tailored to the needs of interactive user. Thus an extensive online *Help* facility is provided for detailed descriptions of all commands and data structure.
A set of *Tutorial* commands guides the novice user through all major aspects of MIDAS.

4.1.2 The MIDAS 's monitor

The interactive MIDAS 's monitor interprets all command input, which may come from the terminal or procedure file.

The MIDAS 's commands fall into two categories :

1. the system commands which control the monitor and are executed directly,
2. the application commands which perform the different application functions. For these commands, the monitor "creates" a *spown* process in the context of which the relevant application program is run.
   So the monitor gives the program and the parameters to an executor, which "executes" the program. The control returns to the monitor as soon as the execution is ended.
   This imposes that each program :
   - begins with the *PROLOG* command which sets up the environment for the application program and must be called before any other reference to this one,
   - ends with the *EPILOG* command which disconnects the calling program and must be called as the last reference to it.

All user 's interaction and terminal output is logged by the monitor in a file. This *logfile* may then be printed for an history of a MIDAS 's session and may also be used to reply to the entire session later on.

4.1.3 Data structures

Image

Bulk data frames or images are the basic structure of MIDAS. An image consists of the *image data* (the pixels) and *descriptive information* stored in descriptors (see here under).

The descriptors contain the information necessary to interpret the structure of the image, the number of pixels in each axis, the start values, the stepsize and the unit in each axis, and an identification string.

[2] see [Banse-Crane-Ounnas-Ponz]

Descriptor

Descriptors are named variables or arrays of type real, integer, character or double precision. They may contain any auxiliary data values connected with an image and reside in the same file.

Beside the standard descriptors, new descriptors may be added and used for other purposes at will.

Keyword

Keywords from a global parameter serve for the monitor as well as for the application programs and provide a *channel* between them.
They are named variables or arrays of type real, integer, character or double precision.

Application programs get their parameters via these keywords only, so no Fortran input-output is used. This *standardizes* the input-output (see section 4.1.4).

Table

Tables represent tabular data, i.e. data arranged in rows and columns, but not necessarily of the same physical significance. For example, they are used to store positions of objects, their magnitudes and other informations derived from images.

Tabular data may be of type real, integer or character.

4.1.4 Standard interface

"MIDAS is based on the Standard Interface for application programs, a set of subroutines agreed between ESO and Space Telescope Science Institute (STSci), with the help of the ST-ECF."[3]
These program interfaces allow easy integration of application programs into MIDAS.

This set of interfaces provides the following functions :

- acces to keywords and descriptors,
  keys and descriptors may be read, written, created and deleted;

- access to bulk data frames and tables,
  working on a computer with a virtual memory system, it is possible to access data wholly in a program array instead of reading it in portions from a file. This is done by mapping file directly into the program virtual memory. So the file is not physically read from disk, only some *pointers* are changed. A pointer to the start of the data area is returned to the calling program and can be used as a normal Fortran array reference in a subroutine called from the main program;

[3] see [IPG-ug]

- communication with the user.

MIDAS 's application programs have a "clean" structure, separating parameter input-output and image data access from the actual algorithm.
As a matter of fact, the program gets all necessary input via keywords and-or descriptors, maps the input-output frames and passes the relevant pointers to the subroutine wich implements the actual algorithm.

4.1.5 Command structure

MIDAS is a command driven system in which the user enters commands followed by parameters.

A MIDAS 's command has the following structure :
COMMAND/QUALIFIER Par1 Par2 ... Par8;
where Par1 is the first parameter and so on.

The important points of this tructure are that :

- command and qualifier are separated by a slash (/),
- the command/qualifier and the parameters are separated by a space,
- most commands have qualifiers,
- in most cases, if the parameters are not specified, the system makes sensible defaults.

4.1.6 Graphic software

All the graphic software is offered by the MIDAS 's commands. To facilitate easy implementation of different graphic and display devices, MIDAS has now adopted a set of device independent interfaces for *plotting* and image display.

All plotting routines in MIDAS are based on the Astronet Graphic Library developed and maintained by the italian Astronet.

4.2 Design of the user 's interface

4.2.1 Introduction

This step aims at the defining the design of the new MIDAS 's commands that are necessary to use the package.
This will correspond more or less to the "functionnal analysis" described by F. Bodart and Y. Pigneur[4].

[4]see [Bodart-Pigneur]

The phase of the information structuration is replaced by the identification of the inputs and outputs of each method (section 4.2.2). From the process structuration (section 4.2.3) we can extract the process dynamic (section 4.2.4). The process static can be shown in section 4.2.5, and so the MIDAS 's commands can be designed (section 4.2.6).
It has to be noted that three commands must be added to respect MIDAS 's philosophy (section 4.2.7). All the "rules" to use this package are mentioned in section 4.2.8.

4.2.2 Inputs-Outputs

In this part, the inputs and the outputs of each method are described.
So we can see the different parameters in common, and make known the data 's structure chosen in input and output.

Phase Dispersion Minimization

This method needs as input :

- the vector of the time,
- the vector of the data corresponding to the time,
- the vector of the segments corresponding to the time and the data, if needed,
- the minimal frequency,
- the maximal frequency,
- the number of discretizations in this interval, or the frequency step,
- the number of bins,
- the number of covers.

In output, this method gives the best period with the F-test corresponding.

Signal Variation Minimization

The signal variation minimization method needs as input parameters :

- the vector of the time,
- the vector of the data corresponding to the time,
- the vector of the segments corresponding to the time and the data, if needed,
- the minimal frequency,
- the maximal frequency,
- the number of discretizations in this interval, or the frequency step,
- the noise to the signal ratio.

The result of this method corresponds to the best period found.

Discrete Fourier Transform.

The input parameters for this method are :

- the vector of the time,
- the vector of the data corresponding to the time,
- the vector of the segments corresponding to the time and the data, if needed,
- the minimal frequency,
- the maximal frequency,
- the number of discretizations in this interval, or the frequency step,
- the noise to the signal ratio.

In output, we have for this method the best period.

Conclusions and data structures

As we can see :

- the input parameters are rather the same, except the method 's parameters themselves, (number of bins and number of covers or the error),
- the output parameter is the same for each method.

We have decided on the following data 's structure :

- the measurements are stored in a table, with a particular column, for respectively, the vector of the time, the vector of the data and the one of the segments,
- to give as output, a table with an ordered list of the period candidates, the corresponding objective function, and the test performed if there is one,
- to give an image with all the frequencies and the objective function at each point of the discretizations,
- to give, for the Discrete Fourier Transform method only, two other images, one with the spectral window and the other with the deconvolved function, at each point of the discretization.

4.2.3 The process structuration

To structure our process, we will use the "Model of process structuration" explained by F. Bodart and Y. Pigneur[5].
The aim of this model is to give criterias to split a project into more and more elememtary treatments. So this structuration will give the ability to define a hierarchy of the operation that will have to be executed to realize the project.

This processes will be structured in four different groups of objects representing each a particular level of the hierarchy, see Fig. 4.2.

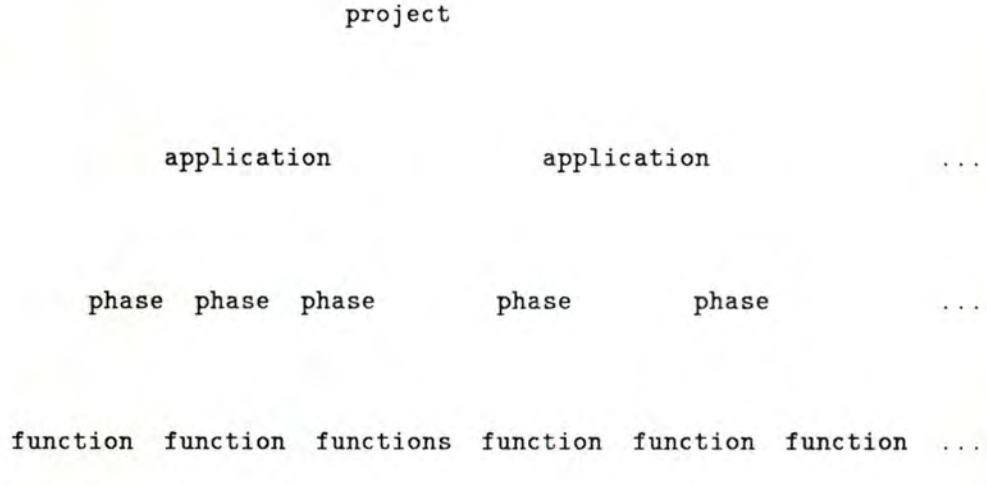


Fig. 4.2 The process 's structuration.

- The project is defined as being the *object* of the analysis.
  In this case, the project will be the determination of period for unequally spaced data.
- The application is a *quasi "self-living" process* with respect to the other applications of the project.
  In this project, we will say that there is only one application consisting in the determination of periods for unequally spaced data.
  The project and the application are considered in our particular case as the same object.

- By definition, a phase is a *manual or automatic process* executed in location and having an uninterrupted time duration.

[5]see [Bodart-Pigneur]

So, in this particular project, seven different main phases can be considered. These phases are :

1. it is necessary first to make the observations, the measurements,
2. the second phase consists in the formatting of the measurements, formatting corresponding to the one required by the package,
3. set the context of the time series analysis package,
4. in the fourth phase, the analysis 's method must be chosen and the parameters needed by this particular method have to be evaluated,
5. the fifth phase corresponds to the different computations required for the realization of the method,
6. the sixth phase displays parameter 's value and some results of the execution,
7. and the seventh phase, the last one, consists of a particular visualization of all the results of the execution, visualization adapted to the user 's need.

Notes :

* the phases considered in this system will be only the phases number 3, 4, 5, 6, and describe a *sub-system* of the project,
* the phases number 1, 2 and 7 are particular phases that must be realized by the user itself,
* to realize the seventh phase, the user needs a particular graphic software which already existing (see 4.1.6).

– Each function has to be associated with an *elementary* objective.

* The functions of the phase number 3 are :
  1. the giving of a default value to the keyword,
  2. the creation of the different commands,
* the functions of the phase number 4 could be :
  1. the reading of the chosen method,
  2. the validation of the chosen method,
  3. the preparation of the working area for this method,
  4. the reading of the value of each parameter required by this method,
  5. and the validation of these values,
* the functions of the fifth phase of the different computations could be :
  - if we use the Phase Dispersion Minimization method, the following functions shoul be :
    1. the initialization of the time to 0,
    2. the computation of the mean and the variance of the observations,
    3. the computation of the degrees of freedom for the F-test,
    4. the initialization of the output parameters, i.e. : the image with the objective function at each point of the discretization, and the table with the period candidates, the objective function and the F-test corresponding,

5. to compute the residual at each point of the discretization;
   this mean for each frequency : to compute the modulo, the phase for the given frequency and to compute the theta, the statistic for this frequency,
6. to check for minima,
7. to compute the F-test for each minimum,

· if we proceed according to the Signal Variation Minimization method, the function should be :

1. the initialization of the time to 0,
2. the computation of the mean and the variance of the observations,
3. the initialization of the output parameters, i.e. : the image with the objective function at each point of the discretization, and the table with the period candidate and the objective function corresponding,
4. to compute the residual at each point of the discretization;
   this mean for each frequency : to compute the modulo, the phase for the given frequency and to compute the theta, the statistic for this frequency,
5. to check for minima,

· if we use the Discrete Fourier Transform method we get something quite different that consists in :

1. the initialization of the time to 0,
2. the initialization of the vector of the datas,
3. the initialization of the output paramaters, i.e. : the image with the objective function at each point of the discretization, the image with the spectral window at each point of the discretization, the deconvolved image at each point of the discretization, and the table with the period candidates, the corresponding Discrete Fourier Transform and the deconvolved and normalized function,
4. to compute the Discrete Fourier Transform at each point of the discretization,
   this mean for each frequency, to compute the Discrete Fourier Transform for this frequency, and the corresponding spectral window,
5. to normalize the spectral window for the clean process,
6. to compute the clean process at each point of the discretization,
7. and finnaly to check for maxima,

* and lastly the functions of the phase number 6 could be :

1. to display on the screen the value of the different parameters of the execution,
2. to display on the screen the different period candidates with the objective function and the test corresponding if there is one.

4.2.4 Process dynamic.

The aim of this *process* dynamic is to show the conditions of triggering, of the execution and of the stopping of the processes in view to indicate the manner the system will react.

The process dynamic diagram is show in Fig 4.3.

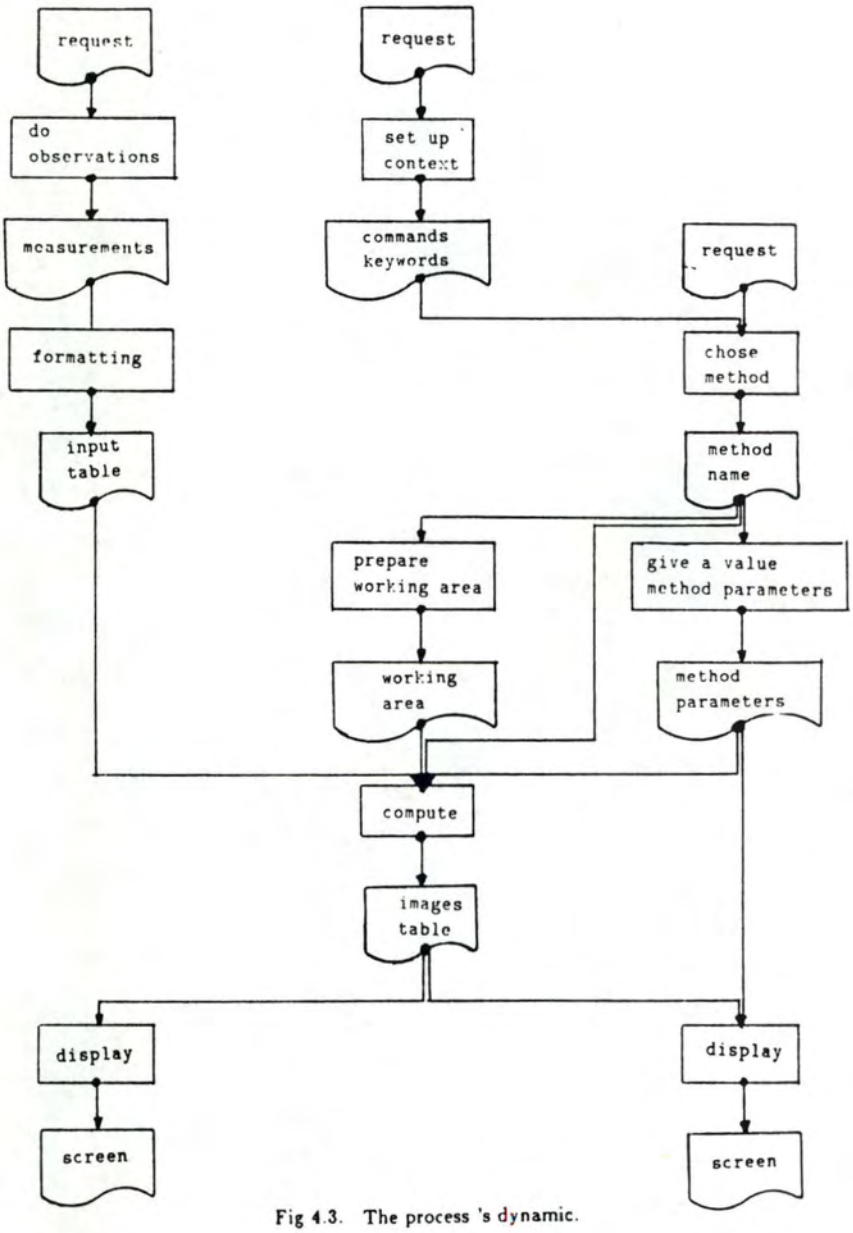


Fig 4.3. The process 's dynamic.

4.2.5 Process static

In this model each process will be seen as a *black box* of which we know only the specification and the parameters.

So each process will be better defined in term of :

- the objective to be realized, the specification;
- the information in input and output.

This will be done for each phase first and each function later.

Phase 3 :

- Name	set context
- Objective	set the *T*ime *S*eries *A*nalysis context
- Input	user request
- Output	default value to the keywords, installation of the different commands

Phase 4 :

- Name	set value
- Objective	set the value to the method 's name and the method 's parameters
- Input	user 's request
- Output	method 's name method 's parameters

Phase 5 :

- Name	computations
- Objective	compute the determinate period according to the chosen method and parameters
- Input	method 's name method 's parameters
- Output	table image(s)

Phase 6 :

- Name visualization
- Objective to show the value of the parameters and the different period candidates
- Input parameters
  table
- Output screens

Phase 3, function 1 :

- Name default values
- Objective set the default value to the keyword
- Input keyword
- Output keyword

Phase 3, function 2 :

- Name procedures
- Objective set the procedures
- Input user 's request
- Output

Phase 4, function 1 :

- Name initialize method
- Objective reading the name of the method
- Input screen
- Output variable with the name of the method

Phase 4, function 2 :

- Name validation method
- Objective to validate the name of the method read in phase 3, function 1
- Input variable with the name of the method
- Output if the method 's name is correct then there is no output, otherwise, the output is an error 's code

Phase 4, function 3 :

- Name initalize working area
- Objective to initialize the working area
- Input method 's name
- Output working area

Phase 4, function 4 :

- Name initialize method 's parameters
- Objective to read the value of each parameter required by the method
- Input screen
  method 's name
- Output parameters with the value

Phase 4, function 5 :

- Name validation parameters
- Objective to validate the value of the method 's parameters
- Input parameters
- Output if the method 's parameter is correct then there is no output, otherwise, the output is an error 's code

For the Phase Dispersion Minimization :

Phase 5, function 1 :

- Name time zero
- Objective set the time to zero
- Input vector of the time
- Output vector of the time

Phase 5, function 2 :

- Name mean-variance
- Objective to compute the mean and the variance of the measurements
- Input vector of the measurements
- Output mean of the measurements
  variance of the measurements

Phase 5, function 3 :

- Name degrees of freedom
- Objective to compute the degrees of freedom for the F-test
- Input number of measurements
  number of segments
  number of bins
  number of covers
- Output degrees of freedom for the F-test

Phase 5, function 4 :

- Name initialization output
- Objective to initalize the output parameters
- Input number of discretization
  frequency step
  minimal frequency
- Output an image with the objective function at each point of the discretization
  a table with the period candidates

Phase 5, function 5 :

- Name computations
- Objective to compute the residu at each point of the discretization
- Input vector of the time
  vector of the measurements
  number of discretization
  frequency step
  minimal frequency
- Output vector of the residus

Phase 5, function 6 :

- Name minima
- Objective to check for minima
- Input vector of the residus produced at phase 5, function 5
- Output table with the period candidates initialize at phase 5, function 4

Phase 5, function 7 :

- Name F-test
- Objective to compute the F-test for each minima
- Input table with the period candidates produced at phase 5, function 6
- Output table with the period candidates and the F-test coresponding

For the Signal Variation Minimization :

Phase 5, function 1 :

- Name time zero
- Objective set the time to zero
- Input vector of the time
- Output vector of the time

Phase 5, function 2 :

- Name mean-variance
- Objective to compute the mean and the variance
- Input vector of the measurements
- Output mean of the measurements
  variance of the measurements

Phase 5, function 3 :

- Name initialization output
- Objective to initalize the output parameters
- Input number of discretization
  frequency step
  minimal frequency
- Output an image with the objective function at each point of the discretization
  a table with the period candidates

Phase 5, function 4 :

- Name computations
- Objective to compute the residu at each point of the discretization
- Input vector of the time
  vector of the measurements
  number of discretization
  frequency step
  minimal frequency
- Output vector of the residus

Phase 5, function 5 :

- Name minima
- Objective to check for minima
- Input vector of the residus produced at phase 5, function 5
- Output table with the period candidates initialize at phase 4, function 3

For the Discrete Fourier Transform :

Phase 5, function 1 :

- Name time zero
- Objective set the time to zero
- Input vector of the time
- Output vector of the time

Phase 5, function 2 :

- Name initialize datas
- Objective to initialize the data 's vector
- Input vector of the data
- Output vector of the data

Phase 5, function 3 :

- Name initialize outputs
- Objective to initialize the output parameters
- Input number of discretizations
  frequency step
  minimal frequency
- Output image with the Discrete Fourier Transform at each point of the discretization
  image with the spectral window at each point of the discretization
  image with the deconvolved spectrum at each point of the discretization
  table with the period candidates

Phase 5, function 4 :

- Name computations
- Objective to compute the Discrete Fourier Transform and the spectal window at each point of the discretization
- Input vector of the time
  vector of the measurements
  number of discretizations
  frequency step
  minimal frequency
- Output vector of the Discrete Fourier Transform
  vector of the spectral window

Phase 5, function 5 :

- Name normalization
- Objective to normalize the spectral window
- Input vector of the spectral window produced at phase 5, function 4
- Output vector of the spectral window normalized

Phase 5, function 6 :

- Name clean
- Objective to compute the clean process
- Input vector with the normalized spectral window produced at phase 5, function 5
  number of discretization
  frequency step
  minimal frequency
- Output image with the deconvolved spectrum

Phase 5, function 7 :

- Name maxima
- Objective to check for the maxima
- Input vector with the deconvolved and normalized spectral window produced at the phase 5, function 5
- Output table with the period candidates initialize at phase 5, function 3

Phase 6, function 1 :

- Name display parameters
- Objective to display the value of the different parameters on the screen
- Input method 's name
  method 's parameters
- Output screen

Phase 6, function 2 :

- Name display results
- Objective to display on the screen the different period candidates
- Input table with the period candidates
- Output screen

4.2.6 MIDAS 's commands

Now that the problem is well defined, we may attach ourselves to the design of the new MIDAS 's commands introduced by the development of all the parts of the sub-system.

A remark must be done; the MIDAS 's commands proposed and agreed by the ESO 's staff are not exactly corresponding to the different phases and functions identified in the process structuration.
This means that different phases and functions are taking place in the same MIDAS 's command.

The different commands containing the process are :

- `SET/CONTEXT`,
- `SET/TSA`,
- `TSA/TABLE`.

The command SET/CONTEXT corresponds to the phase 3, and the two other commands contain the phases number 4, 5, 6.

The structure of these commands is in adequation with all the MIDAS 's package.

Each command being seen as a *black box*, we will give only a specification containing :

- the objective, the aim,
- the syntax ,
- the input parameter(s) with the significance,
- the output parameter(s) with the significance.

Their implementation will be described in chapter 4.

SET/CONTEXT

- Objective set the time series analysis context
- Syntax SET/CONTEXT TSA
- Input name of the context "TSA"
- Output the name of the keyword

SET/TSA

- Objective it sets up the different qualifiers for the next execution of the actual analysis
- Syntax **SET/TSA** [METHOD=method name] [PRINT=nr_prints] [WIDTH=width] [OUTPUT= output]
- Input *method_name* is the name of the method and is one of the following :

  1. *PDM* for the Phase Dispersion Minimization method, that is to say the default value
  2. *SVM* for the Signal Variation Minimization method
  3. *DFT* for the Discrete Fourier Transform method

  nr_print; the most probable nr_print periods are displayed at the end of the treatment, this is an integer value defaulted to 0
  width; width is half the minimal distance or the number of trial frequencies between two successive extrema, its default value is 1
  output is a prefix for the name of the default output; if no output is given in the **TSA/TABLE** command (see below), the results will be stored in images and tables name built with this prefix followed by the method 's name. Default value of the prefix is TSA.
- Output no output

TSA/TABLE

- Objective production of the actual analysis of the time series
- Syntax **TSA/TABLE** [method_par] INTAB [time [data[,segment]]] [OUTPUT]
- Input *method_par*
  the method parameters are *fmin,fmax,ndis[,nbin,ncov]* if the method is PDM and *fmin,fmax,ndis[,error]* otherwise.

  The minimal frequency *fmin* and and the maximal *fmax* fix the investigated frequency interval and *ndis* is the number of trial frequencies in this interval if it is a positive value and represents the frequency step if it is a negative one. If they are omitted or set to a negative value (except ndis), they are computed to satisfy the Nyquist criteria.

The number of bins and covers of the phase unit interval, for the PDM method are exprimed by *nbin* and *ncov*, if omitted or nul they are set to $(5, 2)$.

The *error* is the noise to signal ratio with $0 \leq error \leq 1$. The default value is 0.1.
INTAB is the name of the input table containing the time series
time is the reference to the column of *INTAB* containing the time vector. The default value is *:TIME*
data is the reference to the data column in *INTAB* , it is defaulted to *:DATA*
segment is the reference to the segment column in *INTAB*. If not given, the existence of only one segment is assumed.

This column contains the value of the segment corresponding to the time and the data. When the value change, the time and the data are consedering in an other segment

- Output for each method, an image named *OUTPUT*.BDF contains the value of the objective function (the statistic or periodogram) value at each point of the discretization and a table named *OUTPUT*.TBL is an ordered list of the period candidates. The first column of this table is :FREQ and contains the frequencies where the extrema occurred, the second one named :FUNC is the value of the objective function. The third one is either the F-test (:FTEST) for PDM either the amplitude of the peak (:AMPLI) for DFT, and is used to appreciate the validity of results.

  Moreover, for DFT, two other images are produced : the first named *OUTPUT'SW'*.BDF is the spectral window at each point of the discretization and the second *OUTPUT'DC'*.BDF is the deconvolved spectrum

4.2.7 Others commands

To be in agreement with the MIDAS 's philosophy, it is necessary to introduced three other commands. These commands are not existing in the process described.

These commands are :

- `SHOW/TSA`,
- `HELP/TSA`,
- `TUTORIAL/TSA`.

As we have done for the precious commands, we will specified these commands by giving :

- the objective, the aim,
- the syntax ,
- the input parameter(s) with the significance,
- the output parameter(s) with the significance.

SHOW/TSA

- Objective this causes the display of the actual value of the different qualifiers of the TSA package
- Syntax `SHOW/TSA`
- Input no input
- Output *method_name* name of the method actually performed
  print number of periods display at the end of the treatment
  width half the number of trial frequencies between two successive extrema
  output prefix of the default name of the outputs

HELP/TSA

- Objective this causes the display of all informations concerning the commands of the TSA package
- Syntax `HELP/TSA`
- Input no input
- Output all the informations concerning each command

TUTORIAL/TSA

- Objective try to help in the use of the TSA package
- Syntax `TUTORIAL/TSA`
- Input no input
- Output execution of the different commands of the TSA package

4.2.8 Documentation

With the definition of the design of the user 's interface, a preliminary document destined to the users can be written.
This document corresponds to the specification of the time series analysis package which must be respected during the implementation.
The document is presented in appendix 2.

Chapter 5

The implementation

Trying to respect the specification of the package gives in (chapter 4), we can look now at his implementation and integration into MIDAS.

The programs are made in *FORTRAN-77*. To develop these programs, we have used a suit of software tools for Fortran programmers, section 5.1.
In section 5.2, the Fortran program in relation wich each new MIDAS 's command is designed.

5.1 Toolpack

After a general introduction, describing the public release of Toolpack, the different tools used for developping these programs are defined and explained briefly.

5.1.1 General introduction

Toolpack is both the name of a project and a suite of software tools for Fortan programmers.

The original aims of the Toolpack project were :

- to provide a suite of tools to aid the Fortran-77 programmer,
- to investigate the development and the use of extensible programming support environments built around integrated tool suites.

In our particular case, only the first aim will interest us. In this context, a *software tool* is an utility program that may be used to assist the various phases of constructing, analysing, testing, adapting or maintaining a body of Fortran software.
Alternative terms used to refer to such programs are programming aids and mechanical aids.

The tool reads the software as input data, processes it and produces output that may have one both of the following forms :

- a report that gives an analysis of the input program.
  A tool that produces this form of report without requiring the user 's program to be executed is called *static analyser*,
- a modified version of the input program.
  In this case, the processing tool is a *transformer*.

In some cases, the input may not be a program as such, but could be an associated or derived body of informations. Tools that assist directly in the preparation of the documentation are usually called documentation general aids.

Here, we can find the integrated tool suite. The sequence of actions used in our case is shown Fig 5.1 :

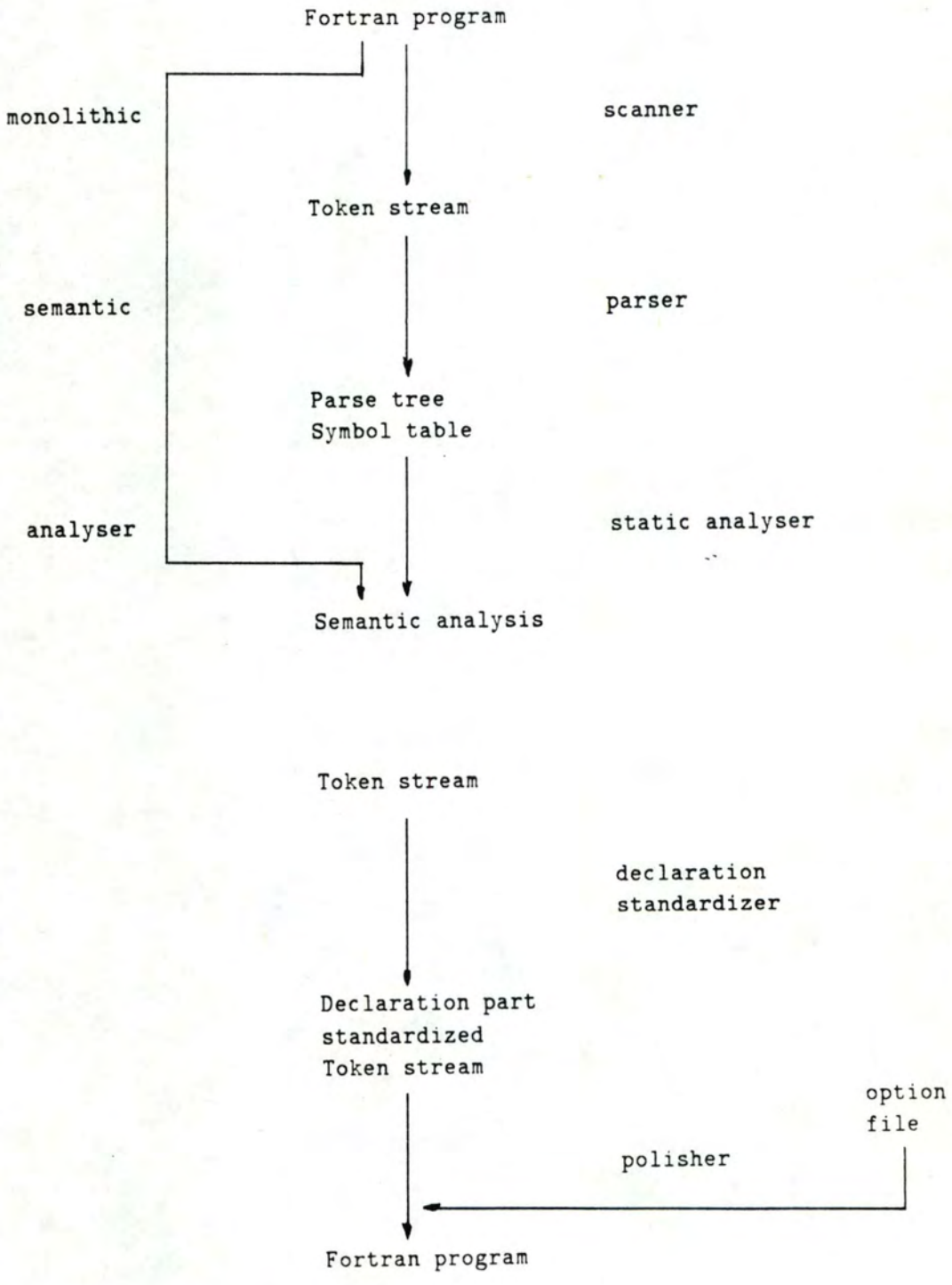


Fig. 5.1 The sequence of actions.

Other tools have been used as those that allow the user in *viewing* the intermediate objects created by tools, an *execution analyser* and a *version controller*.

Notes : only definitions about these different tools are found below, more details and examples can be found in appendix 3.

5.1.2 Monolithic semantic analyser

This tool which is a *static semantic analyser* will perform lexical, syntaxic and semantic analyses on a source code file.
This equates to :

- the lexer or scanner,
- the parser,
- the static analyser.

The scanner

This tool is Fortran-77 lexical analyser that :

- converts Fortran-77 source text to a token stream,
- detects and reports lexical errors.

The scanner has been mechanically generated from a specification of the Fortran-77 language.

The lexical analyser reads Fortran-77 source text from the source file.
The resulting token stream is placed in the tokenfile and the comments in the comment file. Any errors discovered are reported to the error file and an attempt is made to continue scanning by deleting or adding tokens.

During operation, the scanner produces a list file which contains the input source text preceded by the token number of the first token for each statement.
As the token number when the error occured is also reported, this can be related back to the source code using the scanner list file to correct the errors.

The parser

This is a tool which parses a Fortran-77 program to produce a parse tree and symbol table.
It takes as input a token stream produced by the scanner, and gives as results a parse tree, a symbol table and a comment index.

All error and warning messages produced by the tool are written both to the standard error channel, the symbol table file.
When a tool which uses the symbol table is executed, these warning and error messages are displayed again.

The static analyser

This tool is equivalent to a static semantic analyser.

This major tool provides a semantic analysis capability to check conformance to the Fortran-77 standard.
It analyses the parse tree and symbol table of a program, checking for conformance and storing information about the program in a modified parse tree and symbol table with an extended attribute area.

5.1.3 The standardizer

This is a set of tools that allows to standardize the declaration and the presentation of the program.
For this, we have :

- the declaration standardizer,
- the polisher.

The declaration standardiser

This tool standardizes, rebuilds the declarative parts of Fortran-77 program unit, according to a set of templates.
This results in the implicit declaration of all implicity typed items, the removal of all implicit statements and separate declarations for different types of item; i.e. local scalar variables and local array variables.
Array declarators will be output in the type statement, not in DIMENSION statements.

It takes as input the token stream produced by the scanner, the parse tree and the symbol table produced by the parser, and gives as result a new token stream that can be converted to Fortran source code.

Any error or warning messages produced by the parser will be displayed on the standard error channel by the declaration standardizer.

This standardizer will insert all necessary declarative sections, if there are none corresponding in the source code. The order in wich the declarative sections occur is preserved by the standardizer.

The polisher

This tool is a programmable Fortran-77 "pretty printing", "polisher" or unscanner.
This is a formatter for programs written in Fortran.

It takes a token stream as input and produces a text file containing the formatted program as output. It is controlled by option settings from an option file, together with

any command line options.
The Fortran program must be "lexically correct" in the sense that it must be analysed by the scanner without producing errors.

As the polisher make no use of the original source file, it may be used as an "unscan" program wich turns the token stream into text.

When the polisher detects an error, when possible, it corrects this error and continues processing the user 's program. In this case the error message will be inserted into the output file as comment.

Because of the large number of possible options for this tool, an option file editor is provided and is recommended for option file creation and maintenance.

5.1.4 Viewers

It exists a serie of tools that allow the user to view the intermediate objects created by tools and may also be used to detect some minor coding problems and to dispose of a *callgraph*.

Attribute viewer

This tool produces a listing of the information contained in the attribute table created by the semantic analyser.

View symbol table

This tool produces a brief report of the contents of the symbol table for a file .

This produces a formatted listing of the symbol table output from the parser of a Fortran-77 program. The symbols for each program unit are listed separately, and a section is output for each symbol type; i.e. label, common block, name, program unit identifier, variable, procedure, statement function and entry point, which exists in the program unit. Any attributes produced by the parser are listed for each symbol.

View parse tree

This tool interactively displays on the terminal screen sections of a parse tree produced by the parser. The user can move about the tree, and request parts of the tree to be written to a file.

View warnings

This tool provides a capability of displaying warnings.
It produces a formatted listing warnings produced from the symbol table obtained by the execution of the parser.

The warning summary contains information about some common features of a program unit that often signal that something has been overlooked.

Callgraph

This permits the user to show the routine dependencies of those routines and entry points detailed within the specified symbols table files.

5.1.5 Execution analyser

This is a Fortran-77 instrumentor for the dynamic analysis of software.
This tool allows execution and control flow statics to be gathered.

The execution analyser takes as input a Fortran program in token stream form and produces an instrumented Fortran source, a statement summary file for input, an annotated token sream and a summary report.

It is not necessary for the input to contain a complete Fortran program. If only a few routines are to be analysed, they may be input to the execution analyser and the instrumented output combined with the rest of the program.
The instrumented program produces as output a listing file, an optional history file, optional tracing informatiom and optional single run data file. These results are in addition to any output produced by the program.

5.1.6 Version controller

This tool is a version controler, maintaining a number of separate versions of a file in an archive file.

It can be used to maintain information about changes made to any formatted sequential file. The information about changes is kept in a version control file specific to the text file being monitored. This tool compares the current copy of the text file with the more recent version in the verion control file and stores the differencies.

Earlier versions of a file can be recovered from this version control file either by version number or date/time. When date/file option is used, the current version at the specified date and time is recovered.
It is also possible to recover information about the contents of the version control file or about any particular entry.

5.2 Fortran Programs and MIDAS 's procedures

5.2.1 Introduction

This part of the analysis aims to give a clear and complet description of each command 's implementation of the sub-system defined in chapter 3.

In this part, each command being seen as a *white box*; we will give :

- a short remembering, containing a brief description detailed in chapter 4.;
- the program design;
- the variables used in this program, with their signification;
- the pseudo code;
- the program code.

Emphasis will be done on the command *TSA/TABLE*, that is the one concerning the computations.

To explain the program design, we will use the method of A. Van Lamswerde [1]. The program will be considered as a hierarchy divided into *levels*.

5.2.2 SET/CONTEXT

Remembering

This command sets the time series analysis context; with the name of the context as input and gives as output the name of the context 's keyword. The syntax chosen is SET/CONTEXT TSA.

The program design

We have three levels in the hierarchy of this program.

The level 1 corresponds to the command.

In the level 2, we have :

- the writting of the commands,
- the creation of the commands,
- the display of the informations.

[1] see [Van Lamsweerde]

The level 3, corresponding to the creation of the command is decomposed in the creation of :

- SET/TSA,
- TSA/TABLE,
- SHOW/TSA,
- TUTORIAL/TSA.

This hierarchy can be visualized in Fig. 5.2.

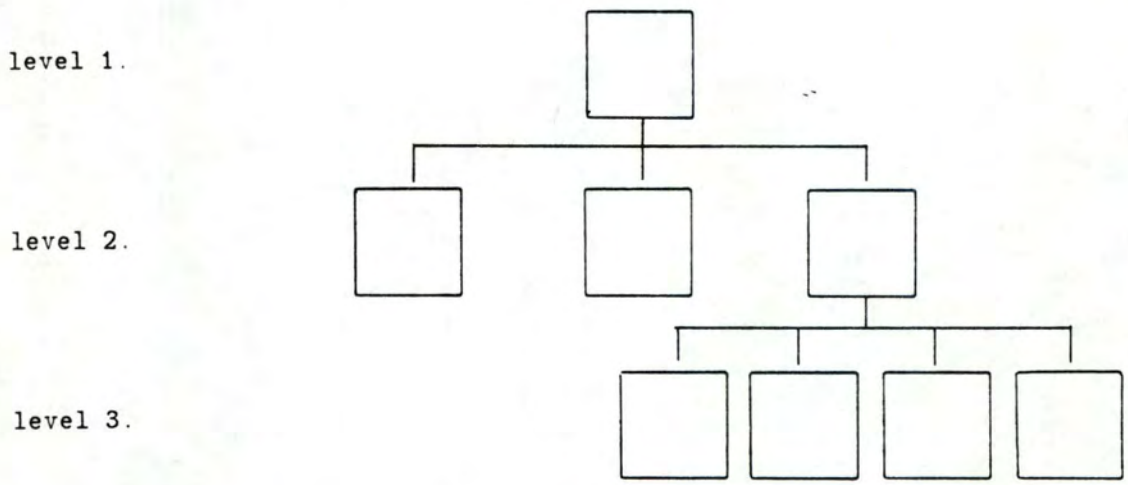


Fig. 5.2 Hierarchy of SET/CONTEXT.

Variables used

TSAKEY name of the keyword of the TSA context
array of 20 characters where :
1...5 method name
6...8 number of printed periods
9...11 half the number of trial fequencies between two successive extrema
12...14 prefix for the name of the default output
15...20 nothing yet.

Pseudo code

```
WRITE KEY  TSAKEY[1..20] = "PDM  0000001TSA      "

CREATE COMMAND  TSA/TABLE    TSATAB.PRG

CREATE COMMAND  SET/TSA      SETTSA.PRG

CREATE COMMAND  SHOW/TSA     SHOWTSA.PRG

CREATE COMMAND  TUTORIAL/TSA TUTTSA.PRG
```

Program code

See appendix 4.

5.2.3 SET/TSA

Remembering

This command sets up the different qualifiers for the next execution of the actual analysis. The input parameters are :

- method_name,
- nr_print periods,
- half the minimal distance between two successive extrema,
- prefix for the name of the default output.

There is no output parameter.

Program design

In this hierarchy 's program, we have three levels :

The level 1 corresponds to the command.

The level 2 corresponds :

- to the cross reference,
- to define the parameters,
- to write the parameters,

The level 3, corresponding to the definition of the parameters is divided in the definition of :

- the parameter with the method name,
- the parameter with the number of printed periods,
- the parameter with half the minimal distance between two successive extrema,
- the parameter with the prefix of the default output,

The level 3, corresponding to the writting of the parameters is divided in the writting of :

- the method name,
- the number of printed periods,
- half the minimal distance between two successive extrema,
- the prefix of the default output.

The hierarchy of this command is visualized in Fig. 5.3.

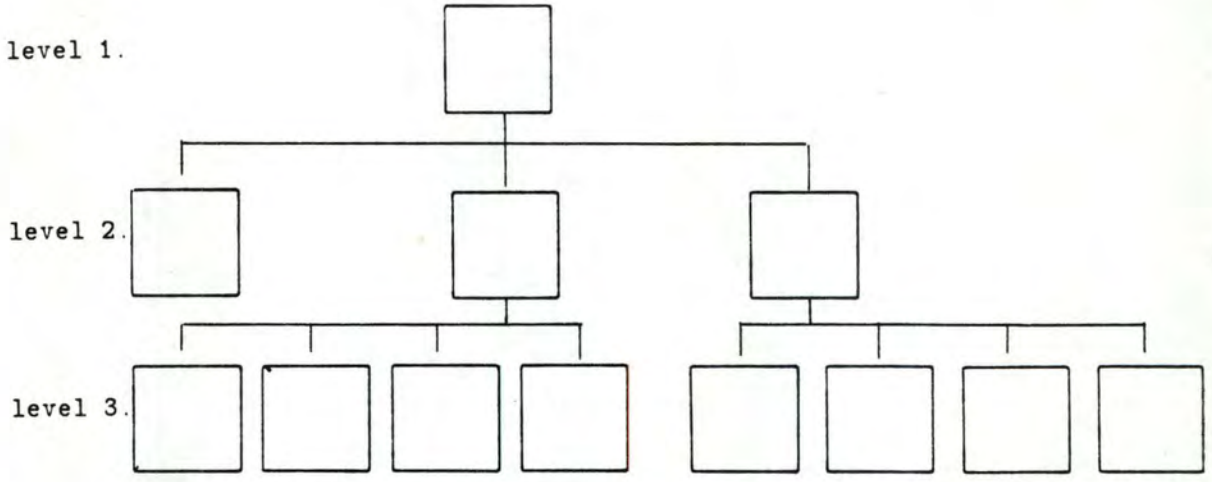


Fig. 5.3 Hierarchy of SET/TSA.

Variables used

METHOD	name of the method array of 5 characters
PRINT	number of printed periods integer scalar
WIDTH	half the minimal distance between two successive extrema integer scalar
OUTPUT	prefix of the output name array of 3 characters
*P*1	parameter with the method name array of 5 characters
*P*2	parameter with the number of printed period integer scalar
*P*3	parameter with half the minimal distance between two successive extrema integer scalar
*P*4	parameter with the prefix of the output name array of 3 characters

Pseudo code

```
CROSREF  METHOD  PRINT  WIDTH OUTPUT

DEFINE PARAMETER  P1 = 'TSAKEY[1..5]'?

DEFINE PARAMETER  P2 = 'TSAKEY[6..8]'?

DEFINE PARAMETER  P3 = 'TSAKEY[9..11]'?

DEFINE PARAMETER  P4 = 'TSAKEY[12..14]'?

WRITE KEY  'TSAKEY[1..5]' = P1

WRITE KEY  'TSAKEY[6..8]' = P2

WRITE KEY  'TSAKEY[9..11]' = P3

WRITE KEY  'TSAKEY[12..14]' = P4
```

Program code

See appendix 4.

5.2.4 TSA/TABLE

Remembering

This command performs the actual analysis of the time series.
With input parameters :

- method ' s parameters,
- name of the table,
- time column,
- data column,
- segment column,
- name of the outputs;

and with output parameters :

- table,
- image(s).

Program design

In the hierarchy of this command, we have 7 levels.

We have first the level corresponding to the command.

In the second level, we have :

- define parameters,
- write keywords,
- execute the computations, this corresponds to the program TSAALL.FOR detailed after,
- delete the scratch table.

The level 3, corresponding to the writing of the keyword, is divided in :

- the initialization of the keyword,
- the writting the keyword.

The level 3, corresponding to *TSAALL.FOR*, is divided in :

- the connection with the environment,
- the reading and validation of the method name,
- the reading of the output name,
- the initialization of the input table,
- the initialization of the time 's column,
- the initialization of the data 's column,
- the initialization of the segment 's column,
- the preparation of the working area, and the solving of the method, this corresponds to the subroutine *SOLVE*,
- the disconnection from the MIDAS 's environment.

The level 4, corresponding to *SOLVE*, is divided in :

- initialize the scratch table,
- copy the input table in the scratch table, column by column,

- make necessary initializations, take the parameters from the corresponding keywords and compute necessary values, this corresponds to the subroutine *INIPT*,
- display informations about the chosen method, this corresponds to the subroutines *DISPDM*, *DISSVM*, *DISDFT*,
- compute the chosen method, this corresponds to the subroutines *PDM*, *RENSON*, *DMING*,
- display the periods computed by the chosen method this corresponds to the subroutine *DREPDM*, *DRESVM*, *DREDFT*,

The level 5, corresponding to *INIPT*, is divided in :

- initialize the vector of the time to 0, this corresponds to the subroutine *INITIM*,
- get the parameters from the corresponding keyword.

The level 5, correponding to *DISPDM*, is divided in display :

- the name of the method,
- the name of the input table,
- the name of the output image,
- the name of the output table,
- the minimal frequency,
- the maximal frequency,
- the number of sample,
- the frequency step,
- the number of bins,
- the number of covers,
- half the minimal frequency between two successive extrema.

The level 5, correponding to *DISSVM*, is divided in display :

- the name of the method,
- the name of the input table,
- the name of the output image,
- the name of the output table,
- the minimal frequency,
- the maximal frequency,
- the number of sample,
- the frequency step,
- the error,
- half the minimal frequency between two successive extrema.

The level 5, correponding to *DISDFT*, is divided in display :

- the name of the method,
- the name of the input table,
- the name of the output images,
- the name of the output table,
- the minimal frequency,
- the maximal frequency,
- the number of sample,
- the frequency step,
- the error,
- half the minimal frequency between two successive extrema.

The level 5, corresponding to *PDM* is divided in :

- compute the mean and the variance, this corresponds to the suboutine *VARMV*,
- compute the degrees of freedom for the F-test, this corresponds to the subroutine *SETDEG*,

- initialize the output image with the thetafs,
- compute the residuals at each point of the discretization, this corresponds to the subroutine *RESIDU*,
- initialize the table with the frequency, the residu and the F-test corresponding,
- check for minima this corresponds to the subroutine*FINEXT*,
- compute the F-test for each minimum, this corresponds to the subroutine *MINFTE*.

The level 5 corresponding to *RENSON* is divided in :

- compute the mean and the variance, this corresponds to the suboutine *VARMV*,
- compute the discard in phase, this corresponds to the subroutine *DISCAR*,
- initialize the output image with the thetafs,
- compute the residuals at each point of the discretization, this corresponds to the subroutine *RESREN*,
- initialize the table with the frequency, the residu corresponding,
- check for minima, this correponds to the subroutine *FINEXT*,
- sort the output table.

The level 5, corresponding to *DMING*, is divided in :

- substract of each values the mean, this corresponds to the subroutine *INITX*,
- initialize the output image with the Discrete Fourier Transform,
- initialize the output image with the spectral window,
- compute the Discrete Fourier Transform and the spectral window at each point of the discretization, this corresponds to the subroutine *FTFREQ*,
- initialize the output table with the maxima,
- normalize the spectral window this corresponds to the subroutine *NORSPW*,
- initialize the output image with the clean,
- compute the clean this corresponds to the subroutine *CLEAN*,
- check for maxima this corresponds to the subroutine *FINEXT*,
- sort the table on the clean,

- normalize the column with the clean.

The level 5, corresponding to *DREPDM*, is divided in the displaying of :

- a tittle,
- the signification of the results,
- the frequencies,
- the thetas,
- the F-tests.

The level 5, corresponding to *DRESVM*, is divided in the displaying of :

- a tittle,
- the signification of the results,
- the frequencies,
- the thetas.

The level 5, corresponding to *DREDFT*, is divided in the displaying of :

- a tittle ,
- the signification of the results,
- the frequencies,
- the thetas,
- the Discrete Fourier Transform,
- the clean.

The level 6, corresponding to *INTIM* is divided in :

- sort the table on the segments and the time,
- initialize the first time of each segment to 0 and compute the other value.

The level 6, corresponding to *VARMV* is divided in :

- initialize the mean and the variance to 0,
- compute the sum of the elements of each segment,

- compute the sumsquare of the elements of each segment,
- compute the mean of each segment,
- compute the variance of each segment,
- compute the mean of the measurements,
- compute the variance of the measurements.

The level 6, corresponding to *SETDEG* is divided in :

- compute the degrees of freedom for the χ^2,
- compute the degrees of freedom for the F-test.

The level 6, corresponding to *RESIDU* is divided in :

- initialization of the frequency,
- for each discretization :
  - compute the modulo for the frequency, this corresponds to the subroutine *MODFRE*,
  - compute the theta for the given frequency, this corresponds to the subroutine *PHADIS*.

The level 6, corresponding to *FINEXT* is divided in :

- initialization of the columns,
- initialization of the last element,
- loop on the number of element in the input vector :
  - * reading of the element in the input table,
  - * check the value of the current and the last element,
  - * check the value of the increment,
  - * check if we can write the extremum found.

The level 6, corresponding to *MINFTE* is divided in :

- loop on the number of minima :
  - * read the minimum in the table,
  - * compute the F-test for it, this corresponds to the subroutine *PCHI*,
  - * compute the significance of the F-test,
  - * write the F-test corresponding to the minimum in the table,

- sort the table on the F-test.

The level 6, corresponding to *DISCAR* is divided in :

- find the maximal and the minimal value,
- compute the discard in phase.

The level 6, corresponding to *RESREN* is divided in :

- initialization of the frequency,
- for each discretization :
  - * compute the modulo for the frequency, this corresponds to the subroutine *MODFRE*,
  - * compute the theta for the given frequency, this corresponds to the subroutine *THEREN*,
- substract of each value the mean of the values.

The level 6, corresponding to *FTFREQ* is divided in :

- for each frequency between 0 and fmin :
  - * compute the exposant,
  - * compute the spectral window this corresponds to the subroutine *FOUSPW*,
  - * increment the frequency,
- for each frequency between fmin and (nech * incspw) * frestp :
  - * compute the exposant,
  - * compute the Discrete Fourier Transform and the spectral window this corresponds to the subroutine *FOUTRA*,
  - * increment the frequency,
- for each frequency between (nech * incspw) * frestp and fmax :
  - * compute the exposant,
  - * compute the DFT this corresponds to the subroutine *FOUDFT*,
  - * increment the frequency.

The level 6, corresponding to *NORSPW* is divided in :

- find the maximum,
- for each value compute the normalization.

The level 6, corresponding to *CLEAN* is divided in :

- initalize the scratch image,
- fill the scratch image this corresponds to the subroutine *FILL*,
- initialize the clean vector to 0,
- find the indice of the maximum and check the clean this corresponds to the subroutine *FINMAX*,
- compute the clean this corresponds to the subroutine *COMCLE*,
- find the indice of the maximum and check the clean this corresponds to the subroutine *FINMAX*.

The level 7 corresponding to *MODFRE* is divided in :

- for each value compute the modulo.

The level 7 corresponding to *PHADIS* is divided in :

- initializations,
- segment loop,
- cover loop,
- reading of the table for a bin,
- compute the theta for a bin,
- compute the theta for the cover,
- compute the theta for the segment,
- compute the number of samples and the divisor,
- compute the thetaf.

The level 7 corresponding to *THEREN* is divided in :

- compute the difference of the first and the last element of the segment,
- compute the difference of their modulo,
- compute the theta,
- for each element of the segment :
  - * compute of this element and the next one,
  - * compute the difference of their modulo,
  - * compute the theta,
  - * increment the frequency.

The level 7 corresponding to *FOUSPW* is divided in :

- initialize the real and imaginary part of the spectral window,
- for each observations :
  - * compute the exposant,
  - * compute the sinus of it,
  - * compute the cosinus of it,
  - * compute the real and imaginary part of the spectral window,
- compute the spectral window normalized.

The level 7 corresponding to *FOUTRA* is divided in :

- initialize the real and imaginary part of the Discrete Fourier Transform and the spectral window,
- for each observations :
  - * compute the exposant,
  - * compute the sinus of it,
  - * compute the cosinus of it,
  - * compute the real and imaginary part of the Discrete Fourier Transform and the spectral window,
- compute the Discrete Fourier Transform and the spectral window normalized.

The level 7 corresponding to *FOUDFT* is divided in :

- initialize the real and imaginary part of the Discrete Fourier Transform,
- for each observations :
  - * compute the exposant,
  - * compute the sinus of it,
  - * compute the cosinus of it,
  - * compute the real and imaginary part of the Discrete Fourier Transform,
- compute the Discrete Fourier Transform normalized.

The level 7 corresponding to *FILL* is divided in :

- for each value of the input vector copy it in the output one.

The level 7 corresponding to *FINMAX* is divided in :

- initializations,
- check for maxima and minima,
- check for the clean.

The level 7 corresponding to *COMCLE* is divided in :

- search the maximal value,
- substract to each value 17

Variables used

It is not possible to identify the variables used by each level, so we will do it for each subroutine.

Working on a computer with a virtual memory system, we have used the possibility described in section 4.1.4; some pointers were, dus, needed.

The design used for the variables 's description correponds to :

- the input variables of the subroutine,
- the output variables of the subroutine,
- the variables used by the subroutine.

For the command :

- There is only variables used.

P1	parameter representing the method 's parameters array of 5 characters
INPUTR	name of the keyword representing the method 's parameters and equal to *P1* array of 5 characters
TSASCR.TBL	name of the scratch table used

For TSAALL.FOR :

- There is only variables used.

TSAKEY	real name of the keyword array of 20 characters
METHOD	name of the chosen method array of 5 characters
INTAB	name of the input table array of 20 characters

COLSEG reference to the segment 's column in the input table *INTAB*
integer scalar

COLTIM reference to the time 's column in the input table *INTAB*
integer scalar

COLX reference to the data 's column in the input table *INTAB*
integer scalar

NCOL number of columns in the input table *INTAB*
integer scalar

NROW number of rows, datas in the input table *INTAB*
integer scalar

LABTIM transfer variable to read from the parameter the reference of the time 's column in the input table
integer scalar

LABX transfer variable to read from the parameter the reference of the data 's column
integer scalar

OUTPUT name of the generic output
array of 20 characters

For SOLVE :

SUBROUTINE SOLVE (METHOD, INTAB, INTIM, INX, INSEG, NROW, OUTPUT)

– Input variables :

METHOD name of the chosen method
array of characters

INTAB name of the input table
array of characters

INTIM reference to the column of the time in the input table *INTAB*
integer scalar

INX reference to the column of the data in the input table *INTAB*
integer scalar

INSEG reference to the column of the segment in the input table *INTAB*
integer scalar

NROW number of rows in the input table *INTAB*
integer scalar

OUTPUT name of the generic outputs
array of characters

– Variables used :

PTRIN pointer to the start of the column in the input table *INTAB*
integer scalar

SCRTAB name of the scrath table used
array of 20 characters

PTRMSK pointer to the start of the scratch table *SCRTAB*
integer scalar

COLTIM reference to the column of the time in the scratch table *SCRTAB*
integer scalar

PTRTIM pointer to the start of the start of *COLTIM*
integer scalar

COLX reference to the column of the data in the scratch table *SCRTAB*
integer scalar

PTRX pointer to the start of *COLX*
integer scalar

COLMOD reference to the column of the modulo in the scratch table *SCRTAB*
integer scalar

PTRMOD pointer to the start of *COLMOD*
integer scalar

COLSEG reference to the column of the modulo in *SCRTAB*
integer scalar

PTRSEG pointer to the start of *COLMOD*
integer scalar

FMAX maximal frequency
real scalar

FMIN	minimal frequency real scalar
FRESTP	frequency step real scalar
NECH	number of discretizations integer scalar
INCSPW	increment for the spectral window integer scalar
NBIN	number of bins integer scalar
NCOV	number of covers integer scalar
ERMEAN	error mean real scalar
SIGCLE	significance value for the clean real scalar
NREXT	number of extrema integer scalar
NSEG	number of segments integer scalar
PRINT	number of printed periods at the end of the treatment integer scalar
WIDTH	half the minimal distance between two successive extrema integer scalar

For INIPT :

SUBROUTINE INIPT (METHOD, SCRTAB, COLTIM, COLX, COLMOD, COLSEG, TIME, SEGM, NSEG, NROW, FMIN, FMAX, NECH, FRESTP, NBIN, NCOV, SIGCLE, ERMEAN, WIDTH, PRINT, INCSPW)

– The input parameters are :

METHOD	name of the chosen method array of characters

SCRTAB	name of the scratch table array of characters
COLTIM	reference to the column of the time in *SCRTAB* integer scalar
COLX	reference to the column of the data in *SCRTAB* integer scalar
COLMOD	reference to the column of the modulo in *SCRTAB* integer scalar
COLSEG	reference to the column of the segment in *SCRTAB* integer scalar
NROW	number of rows, of datas integer scalar
TIME	vector of the time array of *NROW* integers
SEGM	vector of the segment array of *NROW* integers
OUTPUT	generic name of the outputs array of characters

– The output variables are :

NSEG	number of segments integer scalar
FMAX	maximal frequency real scalar
FMIN	minimal frequency real scalar
NECH	number of discretizations integer scalar
FRESTP	frequency step real scalar
NBIN	number of bins integer scalar
NCOV	number of covers integer scalar
SIGCLE	significance value for the clean real scalar

ERMEAN error mean
real scalar

WIDTH half the number of points between two successive extrema
integer scalar

PRINT number of printed periods at the end of the treatment
integer scalar

INCSPW increment for the spectral window
real scalar

– The variables used are :

DEFTIM difference of the time
real scalar

DFTISG difference of the time between the first and the last element of a segment
real scalar

FNYMAX maximal frequency computed on the Nyquist criteria
real scalar

FNYMIN minimal frequency computed on the Nyquist criteria
real scalar

FNYMID middle frequency computed on the Nyquist criteria
real scalar

LGTISG longer of the time in one segment
real scalar

DFFRST number of discretizations corresponding to the Nyquist criteria
real scalar

IROW indice on *NROW*
integer scalar

INDSGB indice of the beginning of the segment
integer scalar

NPTSEG number of points in a segment
integer scalar

NSOR number of column to be sorted
integer scalar

BUFF intermediate buffer to read the value from the parameter
array of 3 characters

PARAM intermediate buffer to read the value from the parameter
array of 5 characters

COLSOR array with the references to the column to be sorted
array of 4 integers

For INITIM :

SUBROUTINE INITIM (INTAB, COLTIM, COLX, COLMOD, COLSEG, TIME, SEGM, NROW)

– The input variables are :

INTAB name of the input table
array of characters

COLTIM reference to the column of the time in *INTAB*
integer scalar

COLX reference to the column of the data in *INTAB*
integer scalar

COLMOD reference to the column of the modulo in *INTAB*
integer scalar

COLSEG reference to the column of the segment in *INTAB*
integer scalar

NROW number of rows, of datas
integer scalar

TIME vector of the time
array of *NROW* integers

SEGM vector of the segment
array of *NROW* integers

– The output variables are :

TIME vector of the time
array of *NROW* integers

– The variables used are :

$TIME1$	value of the time of the first element of a segment real scalar
I	indice in $NROW$ integer scalar
$IROW$	indice in the number of rows integer scalar
$ISEG$	indice in the number of segments integer scalar
$NSOR$	number of columns to be sorted in the table integer scalar
$COLSOR$	vector with the reference of the columns to be sorted array of 4 integers

For DISPDM :

SUBROUTINE DISPDM (INTAB, OUTPUT, FMIN, FMAX, NECH, FRESTP, NBIN, NCOV, WIDTH)

– The input variables are :

$INTAB$	name of the input table array of characters
$OUTPUT$	generic name of the outputs array of characters
$FMAX$	maximal frequency real scalar
$FMIN$	minimal frequency real scalar
$NECH$	number of discretizations integer scalar
$FRESTP$	frequency step real scalar
$NBIN$	number of bins integer scalar

NCOV number of covers
integer scalar

WIDTH half the number of points between two successive extrema
integer scalar

- There is no output variable.
- The variables used are :

OUTIMA name of the output image
array of 25 characters

OUTTAB name of the output table
array of 25 characters

OUTTEX text to be displayed
array of 120 characters

For DISSVM :

SUBROUTINE DISSVM (INTAB, OUTPUT, FMIN, FMAX, NECH, FRESTP, ERMEAN, WIDTH)

- The input variables are :

INTAB name of the input table
array of characters

OUTPUT generic name of the outputs
array of characters

FMAX maximal frequency
real scalar

FMIN minimal frequency
real scalar

NECH number of discretizations
integer scalar

FRESTP frequency step
real scalar

ERMEAN error mean
real scalar

WIDTH half the number of points between two successive extrema
integer scalar

- There is no output variable.
- The variables used are :

OUTIMA name of the output image
array of 25 characters
OUTTAB name of the output table
array of 25 characters
OUTTEX text to be displayed
array of 120 characters

For DISDFT :

SUBROUTINE DISDFT (INTAB, OUTPUT, FMIN, FMAX, NECH, FRESTP, ERMEAN, WIDTH)

- The input variables are :

INTAB name of the input table
array of characters
OUTPUT generic name of the outputs
array of characters
FMAX maximal frequency
real scalar
FMIN minimal frequency
real scalar
NECH number of discretizations
integer scalar
FRESTP frequency step
real scalar
ERMEAN error mean
real scalar
WIDTH half the number of points between two successive extrema
integer scalar

- There is no output variable.

– The variables used are :

OUTIMA	name of the output image array of 25 characters
OUTTAB	name of the output table array of 25 characters
OUTDCV	name of the deconvolved spectrum image array of 25 characters
OUTSPW	name of the image with the spectral window array of 25 characters
OUTTEX	text to be displayed array of 120 characters

For PDM :

SUBROUTINE PDM (INTAB, COLTIM, COLX, COLMOD, COLSEG, TIME, XVAL, MODULO, SEGM, NROW, NBIN, NCOV, NSEG, FMIN, NECH, FRESTP, WIDTH, OUTPUT, NRMIN)

– The input variables are :

INTAB	name of the input table array of characters
COLTIM	reference to the column of the time in *INTAB* integer scalar
COLX	reference to the column of the data in *INTAB* integer scalar
COLMOD	reference to the column of the modulo in *INTAB* integer scalar
COLSEG	reference to the column of the segment in *INTAB* integer scalar
NROW	number of rows, of datas integer scalar
TIME	vector of the time array of *NROW* integers
XVAL	vector of the measurements array of *NROW* reals

MODULO	vector of the modulo, the phase array of *NROW* reals
SEGM	vector of the segment array of *NROW* integers
NBIN	number of bins integer scalar
NCOV	number of covers integer scalar
NSEG	number of segments integer scalar
FMIN	minimal frequency real scalar
NECH	number of discretizations integer scalar
FRESTP	frequency step real scalar
WIDTH	half the number of points between two successive extrema integer scalar
OUTPUT	generic name of the outputs array of characters

– The output variables are :

NRMIN	number of minima found integer scalar

The image and table

– The variables used are :

SIG2	variance of the measurements real scalar
XMEAN	mean of the measurements real scalar
COLFRE	reference to the column of the frequency in the output table integer scalar

COLTHE	reference to the column of the theta in the output table integer scalar
COLFTE	reference to the column of the F-test in the output table integer scalar
FLAG	indicates if minima have to be found (value equal to -1) or if maxima have to be found (value equal to 1) integer scalar
NDEG	number of degree of freedom for the F-test integer scalar
NDEGF1	number of degree of freedom for the F-test integer scalar
NDEGF2	number of degree of freedom for the F-test integer scalar
PTRFTE	pointer to the start of the column of the F-test in the output table integer scalar
PTRTHE	pointer to the start of the column of the theta in the output table integer scalar
START	start coordinate for the output image array of 1 real
STEP	step for the output image array of 1 real
NPIX	number of pixels in the output image array of 1 integer

For VARMV :

SUBROUTINE VARMV (COLSEG, XVECT, SGVECT, NELE1, NBELEM, NSEG, SIG2, MEAN)

- The input variables are :

COLSEG	reference to the column of the segment in the input table integer scalar
NBELEM	number of elements integer scalar
XVECT	vector of the measurements array of *NBELEM* reals
SGVECT	vector of the segments array of *NBELEM* reals
*NELE*1	indice of the first element of the measurements integer scalar
NSEG	number of segment integer scalar

- The output variables are :

*SIG*2	variance of the measurements real scalar
MEAN	mean of the measurements real scalar

- The variables used are :

SUM	sum of the measurements real scalar
SMSQ	sum square of the measurements real scalar
I	indice in the number of measurements integer scalar
IBEG	indice of the first element of the segment integer scalar

ILST	indice of the last element of the segment integer scalar
ISEG	indice in the number of segment integer scalar
NPTSEG	number of points in a segment integer scalar

For SETDEG :

SUBROUTINE SETDEG (NDAT, NSEG, NBIN, NCOV, NDEG, NDEGF1, NDEGF2)

- The input variables are :

NDAT	number of rows, of datas integer scalar
NSEG	number of segment integer scalar
NBIN	number of bins integer scalar
NCOV	number of covers integer scalar

- The output variables are :

NDEG	number of degree of freedom for the F-test integer scalar
*NDEGF*1	number of degree of freedom for the F-test integer scalar
*NDEGF*2	number of degree of freedom for the F-test integer scalar

- There is no variable used.

For RESIDU :

SUBROUTINE RESIDU (INTAB, COLTIM, COLX, COLMOD, COLSEG, TIME, XVAL, MODULO, SEGM, NROW, NSEG, NBIN, NCOV, NECH, FMIN, FRESTP, SIG2X, THETA)

– The input variables are :

INTAB	name of the input table array of characters
COLTIM	reference to the column of the time in *INTAB* integer scalar
COLX	reference to the column of the data in *INTAB* integer scalar
COLMOD	reference to the column of the modulo in *INTAB* integer scalar
COLSEG	reference to the column of the segment in *INTAB* integer scalar
NROW	number of rows, of datas integer scalar
TIME	vector of the time array of *NROW* integers
XVAL	vector of the measurements array of *NROW* reals
MODULO	vector of the modulo, the phase array of *NROW* reals
SEGM	vector of the segment array of *NROW* integers
NBIN	number of bins integer scalar
NCOV	number of covers integer scalar
NSEG	number of segments integer scalar
FMIN	minimal frequency real scalar
NECH	number of discretizations integer scalar

FRESTP frequency step
real scalar
SIG2X variance of the measurements
real scalar

– The output variables are :

THETA vector with the residu
array of $NECH$ reals

– The variables used are :

FREQ frequency
real scalar
IFREQ indice in the number of frequencies
integer scalar
NSOR number of columns to be sorted in the table
integer scalar
COLSOR vector with the reference of the columns to be sorted
array of 4 integers

For MODFRE :

SUBROUTINE MODFRE (TIME, MODULO, NROW, FREQ)

– The input variables are :

NROW number of rows, of datas
integer scalar
TIME vector of the time
array of $NROW$ integers
FREQ frequency
real scalar

– The output variable is :

MODULO vector of the modulo, the phase
array of $NROW$ reals

– The variables used are :

TIMPH	transfer variable real scalar
IROW	indice in *NROW* integer scalar

For PHADIS :

SUBROUTINE PHADIS (COLSEG, XVAL, MODULO, SEGM, NROW, NSEG, NBIN, NCOV, SIG2X, THETA)

– The input variables are :

COLSEG	reference to the column of the segment in *INTAB* integer scalar
NROW	number of rows, of datas integer scalar
XVAL	vector of the measurements array of *NROW* reals
MODULO	vector of the modulo, the phase array of *NROW* reals
SEGM	vector of the segment array of *NROW* integers
NBIN	number of bins integer scalar
NCOV	number of covers integer scalar
NSEG	number of segments integer scalar
SIG2X	variance of the measurements real scalar

– The output variable is :

THETA	residu for a frequency real scalar

– The variables used are :

BI1SSQ	sum square of the elements of the first bin double precision scalar
BI1SUM	sum of the elements of the first bin double precision scalar
BINSSQ	sum square of the element of a bin double precision scalar
BINSUM	sum of the element of a bin double precision scalar
DIV	divisor double precision scalar
THETAF	intermediate theta for the frequency double precision scalar
BEGBIN	value of the beginning of a bin real scalar
BEGBNS	value of the first modulo in a bin real scalar
ENDBIN	value of the end of a bin real scalar
THIBIN	discard between two successive bins real scalar
THICOV	discard between two successive covers real scalar
ICOV	indice in the number of covers integer scalar
ILOW	indice of the first element of the bin integer scalar
IROW	indice in the number of elements *NROW* integer scalar
ISEG	indice in the number of segments *NSEG* integer scalar
NBELEM	number of elements integer scalar
NDIV	number of divisor integer scalar
NPTBI1	number of elements in the first bin integer scalar

NPTBIN	number of elements integer scalar
NSAMP	number of samples integer scalar

For FINEXT :

SUBROUTINE FINEXT (FLAG, EXTREM, VECAUX, NECH, FMIN, FRESTP, WIDTH, EXTTAB, COLFRE, COLEXT, COLAUX, NREXT)

– The input variables are :

FLAG	indicates if minima have to be found (value equal to -1) or if maxima have to be found (value equal to 1) integer scalar
NECH	number of discretizations integer scalar
EXTREM	vector with the objective function array of *NECH* reals
VECAUX	auxiliary vector array of *NECH* reals
FMIN	minimal frequency real scalar
FRESTP	frequency step real scalar
WIDTH	half the number of points between two successive extrema integer scalar
EXTTAB	name of the table with the extrema array of characters
COLAUX	reference to the auxiliary column in *EXTTAB* integer scalar
COLEXT	reference to the column of the extrema in *EXTTAB* integer scalar

COLFRE reference to the column of the frequency in *EXTTAB*
integer scalar

– The output variables are :

NREXT number of extrema in the output table
integer scalar

The output table *EXTTAB*

– The variables used are : (see next page)

CURAUX	auxiliary value of the current element real scalar
CURELE	value of the current element real scalar
CURFRE	value of the current frequency real scalar
LSTAUX	auxiliary value of the last element treated real scalar
LSTELE	value of the last element treated real scalar
LSTFRE	value of the last frequency treated real scalar
I	indice in the number of elements *NECH* integer scalar
LSTINC	indice of the last increase integer scalar
LSTMIN	indice of the last extrema integer scalar
NRCOL	number of columns integer scalar
INCREA	true if we increase, false otherwise logical scalar
LSTELT	vector with the different value corresponding to the last element treated array of 3 reals
COLMIN	vector with the references to the column of the table *EXTTAB* array of 3 integers

For MINFTE :

SUBROUTINE MINFTE (MINTAB, COLTHE, COLFRE, COLFTE, NRMIN, NDEGF1, NDEGF2)

– The input variables are :

MINTAB	name of the table with the minima array of characters
COLFRE	reference to the column of the frequency in the output table integer scalar
COLTHE	reference to the column of the theta in the output table integer scalar
COLFTE	reference to the column of the F-test in the output table integer scalar
*NDEGF*1	number of degree of freedom for the F-test integer scalar
*NDEGF*2	number of degree of freedom for the F-test integer scalar
NRMIN	number of minima found integer scalar

– The output variable is :

The table with the minima *MINTAB*

– The variables used are :

THETA	value of the theta real scalar
INVTHE	inverse of the value of *THETA* real scalar
SIGNIF	significance of the F-test real scalar
I	indice in the number of minima integer scalar

COLSOR vector with the reference of the columns to be sorted
array of 3 integers

For PCHI :

FUNCTION PCHI (CHI2, F, NDEG, ND2, NT)

– The input variables are :

CHI2 distribution of the X2
real scalar
F value to be checked
real scalar
ND2 degrees of freedom
integer scalar
NDEG degrees of freedom
integer scalar
NT indicates if it is one or two tailed area
integer scalar

– The output variable is :

PCHI significance of the F-test
real scalar

– The variables used are :

A1 intermediate value containing a constant
real scalar
A2 intermediate value containing a constant
real scalar
A3 intermediate value containing a constant
real scalar
F13 intermediate value containing a constant
real scalar
G1 intermediate value containing a constant
real scalar

$P1$	intermediate value containing a constant real scalar
T	intermediate value to compute the significance real scalar
$T29$	intermediate value containing a constant real scalar
$X2$	intermediate value to compute the significance real scalar
$Z2$	intermediate value to compute the significance real scalar
$IPOS$	indice integer scalar

For RENSON :

SUBROUTINE RENSON (INTAB,COLTIM,COLX,COLMOD,COLSEG, TIME, XVAL, MODULO, SEGM, NROW, NSEG, FMIN, NECH, FRESTP,ERMEAN, WIDTH, OUTPUT, NRMIN)

– The input variables are :

$INTAB$	name of the input table array of characters
$COLTIM$	reference to the column of the time in $INTAB$ integer scalar
$COLX$	reference to the column of the data in $INTAB$ integer scalar
$COLMOD$	reference to the column of the modulo in $INTAB$ integer scalar
$COLSEG$	reference to the column of the segment in $INTAB$ integer scalar
$NROW$	number of rows, of datas integer scalar
$TIME$	vector of the time array of $NROW$ integers
$XVAL$	vector of the measurements array of $NROW$ reals

MODULO	vector of the modulo, the phase array of *NROW* reals
SEGM	vector of the segment array of *NROW* integers
ERMEAN	error mean real scalar
NSEG	number of segments integer scalar
FMIN	minimal frequency real scalar
NECH	number of discretizations integer scalar
FRESTP	frequency step real scalar
WIDTH	half the number of points between two successive extrema integer scalar
OUTPUT	generic name of the outputs array of characters

– The output variable is :

NRMIN	number of minima found integer scalar

– The variables used are :

SIG2	variance of the measurements real scalar
XMEAN	mean of the measurements real scalar
COLFRE	reference to the column of the frequency in the output table integer scalar
COLTHE	reference to the column of the theta in the output table integer scalar

COLAMP reference to the column of the amplitude in the output table
integer scalar

FLAG indicates if minima have to be found (value equal to -1) or if maxima have to be found (value equal to 1)
integer scalar

PTRAMP pointer to the start of the column of the amplitude in the output table
integer scalar

PTRTHE pointer to the start of the column of the theta in the output table
integer scalar

START start coordinate for the output image
array of 1 real

STEP step for the output image
array of 1 real

NPIX number of pixels in the output image
array of 1 integer

DISPHA discard in phase
double precision scalar

COLSOR vector with the reference of the columns to be sorted
array of 3 integers

For DISCAR :

SUBROUTINE DISCAR (XVAL, NELEM, ERMEAN, DISPHA)

– The input variables are :

NROW number of rows, of datas
integer scalar

XVAL vector of the measurements
array of *NROW* reals

ERMEAN error mean
real scalar

– The output variable is :

DISPHA	discard in phase double precision scalar

– The variables used are :

VALMAX	maximal value of the measurements real scalar
VALMIN	minimal value of the measurements real scalar
IELEM	indice in the number of measurements integer scalar

For RESREN :

SUBROUTINE RESREN (INTAB, COLTIM, COLX, COLMOD, COLSEG, TIME, XVAL, MODULO, SEGM, NROW, NSEG, FMIN, FRESTP, NECH, DISPHA, SIG2, THETA)

– The input variables are :

INTAB	name of the input table array of characters
COLTIM	reference to the column of the time in *INTAB* integer scalar
COLX	reference to the column of the data in *INTAB* integer scalar
COLMOD	reference to the column of the modulo in *INTAB* integer scalar
COLSEG	reference to the column of the segment in *INTAB* integer scalar
NROW	number of rows, of datas integer scalar
TIME	vector of the time array of *NROW* integers
XVAL	vector of the measurements array of *NROW* reals

MODULO	vector of the modulo, the phase array of *NROW* reals
SEGM	vector of the segment array of *NROW* integers
NSEG	number of segments integer scalar
FMIN	minimal frequency real scalar
NECH	number of discretizations integer scalar
FRESTP	frequency step real scalar
SIG2	variance of the measurements real scalar
DISPHA	discard in phase double precision scalar

– The output variable is :

THETA	vector with the modulo array of *NECH* reals

– The variables used are :

FREQ	frequency real scalar
IFREQ	indice in the number of frequencies integer scalar
NSOR	number of columns to be sorted in the table integer scalar
COLSOR	vector with the reference of the columns to be sorted array of 4 integers

For THEREN :

SUBROUTINE THEREN (XVAL, MODULO, SEGM, NROW, NSEG, DISPHA, SIG2, THETA)

- The input variables are :

NROW	number of rows, of datas integer scalar
XVAL	vector of the measurements array of *NROW* reals
MODULO	vector of the modulo, the phase array of *NROW* reals
SEGM	vector of the segment array of *NROW* integers
NSEG	number of segments integer scalar
DISPHA	discard in phase double precision scalar
SIG2	variance of the measurements real scalar

- The output variable is :

THETA	value of the residu real scalar

- The variables used are :

DIFMOD	difference of the modulo of two successive elements double precision scalar
DIFVAL	difference of the value of two successive elements double precision scalar
DIV	divisor double precision scalar
THETAF	transfer value for the residu double precision scalar
INSEGB	indice of the begin of the segment integer scalar

IROW indice in the number of elements *NROW*
integer scalar

For DMING :

SUBROUTINE DMING (TIME, XVAL, NROW, FMIN, NECH, FRESTP, WIDTH, OUTPUT, SIGCLE, INCSPW, NRMAX)

– The input variables are :

NROW number of rows, of datas
integer scalar

TIME vector of the time
array of *NROW* integers

XVAL vector of the measurements
array of *NROW* reals

NSEG number of segments
integer scalar

FMIN minimal frequency
real scalar

NECH number of discretizations
integer scalar

FRESTP frequency step
real scalar

WIDTH half the number of points between two successive extrema
integer scalar

OUTPUT generic name of the outputs
array of characters

SIGCLE value for the clean
real scalar

INCSPW increment for the spectral window
integer scalar

– The output variable is :

NRMAX number of maxima found
integer scalar

– The variables used are :

VALCLE	value of the clean real scalar
VALNOR	value of the normalization real scalar
COLCLE	reference to the column of the clean in the output table integer scalar
COLDFT	reference to the column of the Discrete Fourier Transform in the output table integer scalar
COLFRE	reference to the column of the frequency in the output table integer scalar
FLAG	indicates if minima have to be found (value equal to -1) or if maxima have to be found (value equal to 1) integer scalar
PTRDCV	pointer to the start of the column of the deconvolved spectrum in the output table integer scalar
PTRDFT	pointer to the start of the column of the Discrete Fourier Transform in the output table integer scalar
PTRSPW	pointer to the start of the colum of the spectral window in the output table integer scalar
START	start coordinate for the output image array of 1 real
STEP	step for the output image array of 1 real
NPIX	number of pixels in the output image array of 1 integer
COLSOR	vector with the reference of the columns to be sorted array of 3 integers

For INITX :

SUBROUTINE INITX (XVAL, NROW)

- The input variables are :

NROW	number of rows, of datas integer scalar
XVAL	vector of the measurements array of *NROW* reals

- The output variable is :

XVAL	vector of the measurements array of *NROW* reals

- The variables used are :

XSUM	sum of the measurements real scalar
XMEAN	mean of the measurements real scalar
IROW	indice in the number of elements *NROW* integer scalar

For FTFREQ :

SUBROUTINE FTFREQ (TIME, XVAL, NROW, NECH, FRESTP, INCSPW, DIFOTR, SPEWIN)

- The input variables are :

NROW	number of rows, of datas integer scalar
TIME	vector of the time array of *NROW* integers
XVAL	vector of the measurements array of *NROW* reals

$NECH$	number of discretizations integer scalar
$FRESTP$	frequency step real scalar
$INCSPW$	incrementation for the spectral window integer scalar

– The output variables are :

$DIFOTR$	vector with the Discrete Fourier Transform at each point of the discretization array of $NECH$ reals
$SPEWIN$	vector with the spectral window at each point of the discretization array of $NECH$ reals

– The variables used are :

$PI2$	value of the $\Pi * 2$ real scalar
$VALEXP$	value of the exposant double precision scalar
$FREQ$	frequency real scalar
I	indice in the number of elements integer scalar
$IFREQ$	indice in the number of frequency integer scalar
$NROW2$	square number of the measurements integer scalar

For FOUSPW :

SUBROUTINE FOUSPW (TIME, NROW, NROW2, VALEXP,SPWNND)

- The input variables are :

NROW	number of rows, of datas integer scalar
TIME	vector of the time array of *NROW* integers
XVAL	vector of the measurements array of *NROW* reals
NROW2	square number of the measurements integer scalar
VALEXP	value of the exposant double precision scalar

- The output variable is :

SPWNND	spectral window normalized real scalar

- The variables used are :

EXPTIM	value of the exposant double precision scalar
COSEXP	cosinus of the exposant double precision scalar
SINEXP	sinus of the exposant double precision scalar
IPSWFQ	imaginary part of the spectral window double precision scalar
RPSWFQ	real part of the spectral window double precision scalar
IROW	indice in the number of elements integer scalar

For FOUTRA :

SUBROUTINE FOUTRA (TIME, XVAL, NROW, NROW2, VALEXP, FOTRND, SPWNND)

- The input variables are :

NROW	number of rows, of datas integer scalar
TIME	vector of the time array of *NROW* integers
XVAL	vector of the measurements array of *NROW* reals
NROW2	square number of the measurements integer scalar
VALEXP	value of the exposant double precision scalar

- The output variables are :

FOTRND	Discrete Fourier Transform normalized real scalar
SPWNND	spectral window normalized real scalar

- The variables used are :

EXPTIM	value of the exposant double precision scalar
COSEXP	cosinus of the exposant double precision scalar
SINEXP	sinus of the exposant double precision scalar
IPFTFQ	imaginary part of the Discrete Fourier Transform double precision scalar
RPFTFQ	real part of the Discrete Fourier Transform double precision scalar
IPSWFQ	imaginary part of the spectral window double precision scalar
RPSWFQ	real part of the spectral window double precision scalar
IROW	indice in the number of elements integer scalar

For FOUDFT :

SUBROUTINE FOUDFT (TIME, XVAL, NROW, NROW2, VALEXP, FOTRND)

– The input variables are :

NROW	number of rows, of datas integer scalar
TIME	vector of the time array of *NROW* integers
XVAL	vector of the measurements array of *NROW* reals
NROW2	square number of the measurements integer scalar
VALEXP	value of the exposant double precision scalar

– The output variable is :

FOTRND Discrete Fourier Transform normalized
real scalar

– The variables used are :

EXPTIM value of the exposant
double precision scalar
COSEXP cosinus of the exposant
double precision scalar
SINEXP sinus of the exposant
double precision scalar
IPFTFQ imaginary part of the Discrete Fourier Transform
double precision scalar
RPFTFQ real part of the Discrete Fourier Transform
double precision scalar

For NORSPW :

SUBROUTINE NORSPW (VECSPW, NPOINT)

– The input variables are :
NPOINT number of measurements
integer scalar
VECSPW vector with the spectral window
array of $NPOINT$ reals

– The output variable is :

VECSPW vector with the spectral window
array of $NPOINT$ reals

– The variables used are :

$VALMAX$	maximal value of the spectral window integer scalar
$IMAX$	indice of $VALMAX$ integer scalar
$IPOINT$	indice in the number of elements integer scalar

For CLEAN :

SUBROUTINE CLEAN (IMABDF,IMASPW,IMADCV, NECH, FMIN, FRESTP, SIGCLE)

- The input variables are :

FMIN	minimal frequency real scalar
NECH	number of discretizations integer scalar
FRESTP	frequency step real scalar
IMABDF	vector with the objective function to be cleaned array of *NECH* reals
IMASPW	vector with the spectral window array of *NECH* reals
SIGCLE	value for the clean real scalar

- The output variable is :

IMADCV	vector with the deconvolved spectra array of *NECH* reals

- The variables used are :

AMPMAX	value of the maximum for the clean real scalar
CLEMAX	value of the maximum for the clean real scalar
IMAX	indice of the maximum integer scalar
IPIX	indice in the number of pixels integer scalar
IMASCR	name of the scratch image array of 20 characters

PTRSCR pointer to the start of the scratch image
integer scalar

TRCLEA true if the clean has to be computed; false otherwise
logical scalar

START start coordinate for the output image
array of 1 real

STEP step for the output image
array of 1 real

NPIX number of pixels in the output image
array of 1 integer

For FILL :

SUBROUTINE FILL (INVEC, NPOINT, OUTVEC)

– The input variables are :

NPOINT number of elements
integer scalar

INVEC input vector with the value
array of *NPOINT* reals

– The output variable is :

OUTVEC output vector with the value
array of *NPOINT* reals

– The variable used is :

IVAL indice in the number of elements
integer scalar

For FINMAX :

SUBROUTINE FINMAX (VECT, NPOINT, SIGCLE, IMAX, CLEMAX, AMPMAX, TRCLEA)

- The input variables are :

NPOINT	number of elements integer scalar
VECT	vector with the input elements array of *NPOINT* reals
SIGCLE	significance for the clean real scalar
AMPMAX	value of the first maximum real scalar

- The output variables are :

IMAX	indice of the maximum of the elements integer scalar
CLEMAX	value of the maximum real scalar
TRCLEA	true if the computation of the clean is needed, false otherwise logical scalar

- The variables used are :

IMIN	indice of the minimum value integer scalar
IPOINT	indice in the number of elements integer scalar

For COMCLE :

SUBROUTINE COMCLE (INVEC1, INVEC2, NPOINT, IMAX, CLEMAX, OUTVEC)

- The input variables are :

$NPOINT$	number of elements integer scalar
$INVEC1$	vector with the spectrum array of $NPOINT$ reals
$INVEC2$	vector with the spectral window array of $NPOINT$ reals
$IMAX$	indice of the maximum integer scalar
$CLEMAX$	value of the maxumum real scalar

- The output variable is :

$OUTVEC$	vector with the deconcolved spectrum array of $NPOINT$ reals

- The variables used are :

$VALCLE$	value for the clean real scalar
$IVAL$	indice in the number of elements integer scalar
$CLECST$	indice to stop the computation of the clean integer scalar

For DREPDM :

SUBROUTINE DREPDM (INTAB, NELEM)

- The input variables are :

NELEM	number of elements integer scalar
INTAB	name of the input table containing the elements to be displayed array of characters

- There is no output variable.
- The variables used are :

IELEM	indice in the number of elements integer scalar
OUTTEX	text to be displayed array of 100 characters
VALOUT	vector of the frequency, the objective function and the F-test to be displayed array of 3 reals
COLOUT	vector with the reference to the column of the frequency, the column of the objective function and the one of the F-test in the table *INTAB* array of 3 integers

For DRESVM :

SUBROUTINE DRESVM (INTAB, NELEM)

- The input variables are :

NELEM	number of elements integer scalar

INTAB name of the input table containing the elements to be displayed
array of characters

- There is no output variable.
- The variables used are :

IELEM indice in the number of elements
integer scalar

OUTTEX text to be displayed
array of 100 characters

VALOUT vector of the frequency, the objective function to be displayed
array of 2 reals

COLOUT vector with the reference to the column of the frequency, the column of the objective function in the table *INTAB*
array of 2 integers

For DREDFT :

SUBROUTINE DREDFT (INTAB, NELEM)

- The input variables are :

NELEM number of elements
integer scalar

INTAB name of the input table containing the elements to be displayed
array of characters

- There is no output variables.
- The variables used are :

IELEM	indice in the number of elements integer scalar
OUTTEX	text to be displayed array of 100 characters
VALOUT	vector of the frequency, the objective function and the deconvolved spectrum to be displayed array of 3 reals
COLOUT	vector with the reference to the column of the frequency, the column of the objective function and the one of the deconvolved spectrum in the table *INTAB* array of 3 integers

Pseudo code

Before looking at the pseudo code corresponding to each subroutine, it is necessary to give the specification of the different commands of the MIDAS 's interface used.

- CALL PROLOG_ST
  This routine will sets up the environment for the application program.
- CALL EPILOG ST
  This routine will disconnect the calling program from the environment.
- CALL GETKEY ST (KEY, FELEM, MAXVALS, VALUES)
  This routine will read value(s) from the keyword area.

  * The input arguments are :

KEY	keyword name array of characters
FELEM	position of the first data item to be accessed integer scalar
MAXVALS	maximum number of values required integer scalar

* The output argument is :

  VALUES buffer to hold value
  if *MAXVALS* ¿ 1 *VALUES* must be dimensioned at least that size

– CALL PUTIMAG_ST (NAME, NAXIS, NPIX, START, STEP, PNTR)
This routine writes data frame and the standard descriptors.

* The input arguments are :

  NAME file name of data frame
  array of characters

  NAXIS number of dimensions of the image
  integer

  NPIX sizes dimensions
  array of *NAXIS* integers

  START start coordinates
  array of *NAXIS* reals

  STEP step sizes
  array of *NAXIS* reals

* The output argument is :

  PNTR pointer to mapped data area
  integer scalar

– CALL TBL_READ (NAME)
This routine reads the specified table file already existing on disk.

* The input argument is :

  NAME table name
  array of characters

- CALL TBL RDINFO (NAME, COLUMN, NROW)
  This routine returns general information on the specified table.
  * The input argument is :

    NAME table name
    array of characters

  * The output arguments are :

    COLUMN number of columns
    integer scalar
    ROW number of rows
    integer scalar

- CALL TBL_FNDLAB (NAME, LABEL, COLUMN)
  This routine returns the table column number corresponding to the specified column label.
  * The input arguments are :

    NAME table name
    array of characters
    LABEL column label
    array of characters

  * The output arguments are :

    COLUMN number of column associated with that label
    integer scalar

- CALL TBL INIT (NAME, ALLCOL, ALLROW)
  This routine initializes a non existing table file on disk and allocates initial memory space.
  * The input arguments are :

    NAME table name
    array of characters
    ALLCOL number of physical columns allocated
    integer scalar
    ALLROW number of physical rows allocated
    integer scalar

- CALL TBL MAPCOL (NAME, COLUMN, PNTR)
  This routine returns the address of a column in the specified table.
  * The input arguments are :

    NAME table name
    array of characters

    COLUMN column number
    integer scalar

  * The output argument is :

    PNTR pointer to the start of the column
    integer scalar

- CALL TBL_DEFCOL (NAME, LABEL, COLUMN)
  This routine define a new column in the specified table.
  * The input arguments are :

    NAME table name
    array of characters

    LABEL column label
    array of characters

  * The output argument is :

    COLUMN number of the column just created
    integer scalar

- CALL TBL_CPYCOL (NAME, FROM, TO, NELEM)
  This routine copies one column to an other.
  * The input arguments are :

NAME	pointer to the beginning of the input table integer scalar
FROM	pointer to the beginning of the input column integer scalar
TO	pointer to the beginning of the output column integer scalar
NELEM	number of elements to copy integer scalar

– CALL TBL SORT (NAME, NC, COLUMN, SRTFLG)
This routine sorts the rows of the specified table with the respect to the ascending/descending order.

* The input arguments are :

NAME	table name array of characters
NC	number of columns to be sorted integer scalar
COLUMN	array defining the columns to be dorted array of *NC* integers
SRTFLG	sort flag = 1 ascending order = -1 descending order integer scalar

– CALL TBL_RDELEM (NAME, ROW, COLUMN, VALUE)
This routine reads element from the specified table.

* The input arguments are :

NAME	table name array of characters
ROW	row number integer scalar
COLUMN	column number integer scalar

* The output argument is :

VALUE	table element

– CALL TBL_WRROW (NAME, ROW, NC, COLUMN, VALUE)
This routine writes a row into the specified table.

* The input arguments are :

NAME	table name array of characters
ROW	sequence number integer scalar

NC	number of columns to be written integer scalar
COLUMN	array with the column numbers array of *NC* integers

* The output argument is :

VALUE	array with the data array of *NC* elements

– CALL TBL_WRELEM (NAME, ROW, COLUMN, VALUE)
This routine write an element into the specified table.

* The input arguments are :

NAME	table name array of characters
ROW	sequence number integer scalar
COLUMN	column number integer scalar
VALUE	value to be written

– CALL TBL_DELEM (NAME, ROW, COLUMN)
This routine deletes the specified element in the table.

* The input arguments are :

NAME	table name array of characters
ROW	sequence number integer scalar
COLUMN	column number integer scalar

VTME(PTR) is used to obtain the virtual memory corresponding to the pointer PTR.
VTME(PTR) corresponds to the Fortran instruction %VAL(PTR).

We can now look at the different subroutines.

For the command :

```
DEFINE PARAMETER  P1
WRITE KEY  INPUTR
WRITE KEY  INPUTR = P1
RUN TSAALL
DELETE TSASCR.TBL
```

For TSAALL.FOR :

```
BEGIN
      CALL PROLOG_ST
      CALL GETKEY_ST('TSAKEY',1,5,METHOD)
      CALL GETKEY_ST('P5',1,20,OUTPUT)
      IF (OUTPUT(1:1).EQ.'?') THEN
          CALL GETKEY_ST('TSAKEY',12,14,OUTPUT)
          OUTPUT(4:8) = METHOD
       END IF
       CALL GETKEY_ST('P2',1,20,INTAB)
       CALL TBL_READ(INTAB)
       CALL TBL_RDINFO(INTAB,NCOL,NROW)
       CALL GETKEY_ST('P3',1,8,LABTIM)
       CALL TBL_FNDLAB(INTAB,LABTIM,COLTIM)
```

```
        CALL GETKEY_ST('P4',1,17,LABXS)
        CALL TBL_FNDLAB(INTAB, LABXS, COLX)
        IF (SEGMENT) THEN
            CALL TBL_FNDLAB(INTAB, LABXS, COLSEG)
         END IF
        CALL SOLVE(METHOD,INTAB,COLTIM,COLX,COLSEG,NROW,OUTPUT)
        CALL EPILOG_ST
END
```

For SOLVE :

SUBROUTINE SOLVE (METHOD, INTAB, INTIM, INX, INSEG, NROW, OUTPUT)

```
BEGIN
        SCRTAB = 'TSASCR.TBL'
        CALL TBL_INIT(SCRTAB,4,NROW)
        CALL TBL_MAPCOL(INTAB,0,PTRMSK)
        CALL TBL_DEFCOL(SCRTAB,'TIME',COLTIM)
        CALL TBL_MAPCOL(INTAB,INTIM,PTRIN)
        CALL TBL_MAPCOL(SCRTAB,COLTIM,PTRTIM)
        CALL TBL_CPYCOL(VTME(PTRMSK),VTME(PTRIN),VTME(PTRTIM),NROW)
        CALL TBL_DEFCOL(SCRTAB,'VALUE',COLX)
        CALL TBL_MAPCOL(INTAB,INX,PTRIN)
```

```
CALL TBL_MAPCOL(SCRTAB,COLX,PTRX)

CALL TBL_CPYCOL(VTME(PTRMSK),VTME(PTRIN),VTME(PTRX),NROW)

CALL TBL_DEFCOL(SCRTAB,'MODULO',COLMOD)

CALL TBL_MAPCOL(SCRTAB,COLMOD,PTRMOD)

IF (INSEG <> 0) THEN

    CALL TBL_DEFCOL(SCRTAB,'SEGM',COLSEG)

    CALL TBL_MAPCOL(INTAB,INSEG,PTRIN)

    CALL TBL_MAPCOL(SCRTAB,COLSEG,PTRSEG)

    CALL TBL_CPYCOL(VTME(PTRMSK),VTME(PTRIN),VTME(PTRSEG),
                    NROW)

ELSE

    PTRSEG = 0

    COLSEG = 0

END IF

CALL INIPT(METHOD,SCRTAB,COLTIM,COLX,COLMOD,COLSEG,VTME(PTRTIM),
           VTME(PTRSEG),NSEG,NSEL,FMIN,FMAX,NECH,FRESTP,NBIN,
           NCOV,SIGCLE,ERMEAN,WIDTH,PRINT,INCSPW)

IF (METHOD = 'PDM   ') THEN

    CALL DISPDM(INTAB,OUTPUT,FMIN,FMAX,NECH,FRESTP,NBIN,NCOV,
                WIDTH)

    CALL PDM(SCRTAB,COLTIM,COLX,COLMOD,COLSEG,VTME(PTRTIM),
             VTME(PTRX),VTME(PTRMOD),VTME(PTRSEG),NSEL,NBIN,
             NCOV,NSEG,FMIN,NECH,FRESTP,WIDTH,OUTPUT,NREXT)

    CALL DREPDM(OUTPUT,PRINT)
```

```
        ELSE

            IF (METHOD = 'DFT   ') THEN

                CALL DISDFT(INTAB,OUTPUT,FMIN,FMAX,NECH,FRESTP,SIGCLE,
                            WIDTH)

                CALL DMING(VTME(PTRTIM),VTME(PTRX),NSEL,FMIN,NECH,
                           FRESTP,WIDTH,OUTPUT,SIGCLE,INCSPW,NREXT)

                CALL DREDFT(OUTPUT,PRINT)

            ELSE
                IF (METHOD = 'SVM   ') THEN

                    CALL DISSVM(INTAB,OUTPUT,FMIN,FMAX,NECH,FRESTP,
                                ERMEAN,WIDTH)

                    CALL RENSON(SCRTAB,COLTIM,COLX,COLMOD,COLSEG,
                                VTME(PTRTIM),VTME(PTRX),VTME(PTRMOD),
                                VTME(PTRSEG),NSEL,NSEG,FMIN,NECH,
                                FRESTP,ERMEAN,WIDTH,OUTPUT,NREXT)

                    CALL DRESVM(OUTPUT,PRINT)

                END IF

            END IF

        END IF

END
```

For INIPT :

SUBROUTINE INIPT (METHOD, SCRTAB, COLTIM, COLX, COLMOD, COLSEG, TIME, SEGM, NSEG, NROW, FMIN, FMAX, NECH, FRESTP, NBIN, NCOV, SIGCLE, ERMEAN, WIDTH, PRINT, INCSPW)

```
BEGIN
```

```
CALL INITIM(SCRTAB,COLTIM,COLX,COLMOD,COLSEG,TIME,SEGM,NROW)
CALL GETKEY_ST('INPUTR',1,6,PARAM)
IF (METHOD = 'PDM  ') THEN
    IF (PARAM(4) < 1.E-04) THEN
        NBIN = 5
    ELSE
        NBIN = PARAM(4)
    END IF
    IF (PARAM(5) < 1.E-04) THEN
        NCOV = 2
    ELSE
        NCOV = PARAM(5)
    END IF
END IF
IF (COLSEG = 0) THEN
    NSEG = 1
ELSE
    NSEG = SEGM(NROW)
END IF
IF (METHOD = 'DFT') THEN
    IF (PARAM(4) < 1.E-04) THEN
```

```
            SIGCLE = 0.1
        ELSE
            SIGCLE = PARAM(4)
        END IF
    END IF
    IF (METHOD = 'SVM') THEN
        IF (PARAM(4) < 1.E-04) THEN
            ERMEAN = 0.1
        ELSE
            ERMEAN = PARAM(4)
        END IF
    END IF
    CALL GETKEY_ST('TSAKEY',9,3,BUFF)
    IF (BUFF(1:1) = '?') THEN
        WIDTH = 1
    ELSE
        WIDTH = BUFF
    END IF
    CALL GETKEY_ST('TSAKEY',6,3,BUFF)
    IF (BUFF(1:1) = '?') THEN
        PRINT = 0
```

```
ELSE
    PRINT = BUFF
END IF
FMIN = PARAM(1)
FMAX = PARAM(2)
IF (PARAM(3) > 0) THEN
    NECH = PARAM(3)
ELSE
    NECH = 0
END IF
LGTISG = 0
DEFTIM = 0
IF (COLSEG <> 0) THEN
    COLSOR(1) = COLSEG
    COLSOR(2) = COLTIM
    COLSOR(3) = COLX
    COLSOR(4) = COLMOD
    NSOR = 4
    CALL TBL_SORT(SCRTAB,NSOR,COLSOR,1)
    INCSPW = -1
    NBPTSE = 0
```

```
        INDSGB = 1
        DO 20 IROW = 1,NROW
            IF (SEGM(IROW) <> SEGM(INDSGB)) THEN
                DFTISG = TIME(IROW-1) - TIME(INDSGB)
                DEFTIM = DEFTIM + (DFTISG/ (IROW-INDSGB-2))
                LGTISG = MAX(LGTISG,DFTISG)
                INDSGB = IROW
            END IF
        END DO
        DFTISG = TIME(IROW-1) - TIME(INDSGB)
        DEFTIM = DEFTIM + (DFTISG/ (IROW-INDSGB-2))
        LGTISG = MAX(LGTISG,DFTISG)
    ELSE
        LGTISG = TIME(NROW)
        DEFTIM = LGTISG/ (NROW-1)
    END IF

    DEFTIM = DEFTIM/NSEG
    IF (PARAM(3) <> 0) THEN
        FRESTP = 1/ (5*LGTISG)
    ELSE
        FRESTP = PARAM(3)
```

```
END IF
FNYMIN = 1/LGTISG
FNYMAX = 1/ (2*DEFTIM)
FNYMID = (FNYMAX-FNYMIN)/2
DFFRST = (FNYMAX-FNYMIN)/FRESTP + 1
IF (PARAM(1) < 0) THEN
    IF (PARAM(2) < = 0) THEN
        FMIN   = FNYMIN
        FMAX   = FNYMAX
        IF (PARAM(3) = 0) THEN
            FRESTP = FNYSTP
            NECH   = DFFRST
            INCSPW = 5
        ELSE IF (PARAM(3) > 0) THEN
            NECH   = PARAM(3)
            FRESTP = (FMIN-FMAX)/ (NECH-1)
            ATMP   = FMIN/FRESTP
            NTMP   = INT(ATMP)
            IF (ATMP-NTMP < 0.5E-2) INCSPW = NTMP
        ELSE
            FRESTP = ABS(PARAM(3))
```

```
            NECH    = (FMAX-FMIN)/FRESTP + 1
            ATMP    = FMIN/FRESTP
            NTMP    = INT(ATMP)
            IF (ATMP-NTMP < 0.5E-2) INCSPW = NTMP
        END IF
    ELSE
        FMAX    = PARAM(2)
        FMIN    = FNYMIN
        IF (PARAM(3) = 0) THEN
            FRESTP = FNYSTP
            NECH    = (FMAX-FMIN)/FRESTP
            INCSPW = 5
        ELSE IF (PARAM(3) > 0) THEN
            NECH    = PARAM(3)
            FRESTP = (FMIN-FMAX)/ (NECH-1)
            ATMP    = FMIN/FRESTP
            NTMP    = ATMP
            IF (ATMP-NTMP < 0.5E-2) INCSPW = NTMP
        ELSE
            FRESTP = PARAM(3)
            NECH    = (FMAX-FMIN)/FRESTP + 1
```

```
            ATMP    = FMIN/FRESTP
            NTMP    = ATMP
            IF (ATMP-NTMP < 0.5E-2) INCSPW = NTMP
        END IF
    END IF
ELSE
    IF (PARAM(2) < = 0) THEN
        FMIN    = PARAM(1)
        FMAX    = FNYMAX
        IF (PARAM(3) = 0) THEN
            FRESTP = FNYSTP
            NECH    = (FMAX-FMIN)/FRESTP
        ELSE IF (PARAM(3) > 0.) THEN
            NECH    = PARAM(3)
            FRESTP = (FMAX-FMIN)/ (NECH-1)
        ELSE
            FRESTP = PARAM(3)
            NECH    = (FMAX-FMIN)/FRESTP
        END IF
        ATMP    = FMIN/FRESTP
        NTMP    = ATMP
```

```
            IF (ATMP-NTMP < 0.5E-2) INCSPW = NTMP

        ELSE

            FMIN    = PARAM(1)

            FMAX    = PARAM(2)

            IF (PARAM(3) = 0) THEN

                FRESTP = FNYSTP

                NECH   = (FMAX-FMIN)/FRESTP + 1

            ELSE IF (PARAM(3) > 0.) THEN

                NECH   = PARAM(3)

                FRESTP = (FMAX-FMIN)/ (NECH-1)

            ELSE
                FRESTP = PARAM(3)

                NECH   = (FMAX-FMIN)/FRESTP + 1

            END IF

            ATMP    = FMIN/FRESTP

            NTMP    = ATMP

            IF (ATMP-NTMP < 0.5E-2) INCSPW = NTMP

        END IF

    END IF
END
```

For INITIM :

SUBROUTINE INITIM (INTAB, COLTIM, COLX, COLMOD, COLSEG, TIME, SEGM, NROW)

```
BEGIN
      IF (COLSEG <> 0) THEN
          COLSOR(1) = COLSEG
          COLSOR(2) = COLTIM
          COLSOR(3) = COLX
          COLSOR(4) = COLMOD
          NSOR = 4
          CALL TBL_SORT(INTAB,NSOR,COLSOR,1)
          ISEG = SEGM(1)
          TIME1 = TIME(1)
          DO I = 1,NROW
              IF (SEGM(I) = ISEG) THEN
                  TIME(I) = TIME(I) - TIME1
              ELSE
                  TIME1 = TIME(I)
                  TIME(I) = 0
                  ISEG = SEGM(I)
              END IF
          END DO
      ELSE
```

```
            COLSOR(1) = COLTIM
            COLSOR(2) = COLX
            COLSOR(3) = COLMOD
            NSOR = 3
            CALL TBL_SORT(INTAB,NSOR,COLSOR,1)
            TIME1 = TIME(1)
            DO IROW = 1,NROW
                TIME(IROW) = TIME(IROW) - TIME1
            END DO
        END IF
END
```

For DISPDM :

SUBROUTINE DISPDM (INTAB, OUTPUT, FMIN, FMAX, NECH, FRESTP, NBIN, NCOV, WIDTH)

```
BEGIN
        OUTTEX = 'PHASE DISPERSION MINIMIZATION'
        WRITE (OUTTEX)
        OUTIMA = '.BDF'
        OUTTAB = '.TBL'
        OUTTEX =  'INPUT TABLE      OUTPUT TABLE AND IMAGES'
```

```
      WRITE (OUTTEX)

      OUTTEX = (INTAB,OUTPUT,OUTIMA,OUTTAB)

      WRITE (OUTTEX)

      OUTTEX =  'FREQ MIN  FREQ MAX  SAMPLE  FREQ  STEP
                 NR OF BINS  NR OF COVERS  WIDTH'

      WRITE (OUTTEX)

      OUTTEX  = (FMIN,FMAX,NECH,FRESTP,NBIN,NCOV,WIDTH)

      WRITE (OUTTEX)

END
```

For DISSVM :

SUBROUTINE DISSVM (INTAB, OUTPUT, FMIN, FMAX, NECH, FRESTP, ERMEAN, WIDTH)

```
BEGIN

      OUTTEX =  'SIGNAL VARIATION MINIMIZATION'

      WRITE (OUTTEX)

      OUTIMA = '.BDF'

      OUTTAB = '.TBL'

      OUTTEX = 'INPUT TABLE     OUTPUT TABLE AND IMAGES'

      WRITE (OUTTEX)

      OUTTEX = (INTAB,OUTPUT,OUTIMA,OUTTAB)

      WRITE (OUTTEX)
```

```
        OUTTEX = 'FREQ MIN   FREQ MAX   SAMPLE   FREQ   STEP   ERROR
                  WIDTH'

        WRITE (OUTTEX)

        OUTTEX = (FMIN,FMAX,NECH,FRESTP,ERMEAN,WIDTH)

        WRITE (OUTTEX)

END
```

For DISDFT :

SUBROUTINE DISSVM (INTAB, OUTPUT, FMIN, FMAX, NECH, FRESTP, ERMEAN, WIDTH)

```
BEGIN

        OUTTEX = 'SIGNAL VARIATION MINIMIZATION'

        WRITE (OUTTEX)

        OUTIMA = '.BDF'

        OUTSPW = 'SP.BDF'

        OUTDCV = 'DC.BDF'

        OUTTAB = '.TBL'

        OUTTEX = 'INPUT TABLE      OUTPUT TABLE AND IMAGES'

        WRITE (OUTTEX)

        OUTTEX =  (INTAB,OUTPUT,OUTIMA,OUTTAB)

        WRITE (OUTTEX)
```

```
      OUTTEX = 'FREQ MIN  FREQ MAX  SAMPLE  FREQ  STEP ERROR
                WIDTH'

      WRITE (OUTTEX)

      OUTTEX = (FMIN,FMAX,NECH,FRESTP,ERMEAN,WIDTH)

      WRITE (OUTTEX)

END
```

For PDM :

SUBROUTINE PDM (INTAB, COLTIM, COLS, COLMOD, COLSEG, TIME, XVAL, MODULO, SEGM, NROW, NBIN, NCOV, NSEG, FMIN, NECH, FRESTP, WIDTH, OUTPUT, NRMIN)

```
BEGIN

      CALL VARMV(COLSEG,XVAL,SEGM,1,NROW,NSEG,SIG2,XMEAN)

      CALL SETDEG(NROW,NSEG,NBIN,NCOV,NDEG,NDEGF1,NDEGF2)

      STEP(1) = FRESTP

      START(1) = FMIN

      NPIX(1) = NECH

      CALL PUTIMAG_ST(OUTPUT,1,NPIX,START,STEP,PTRTHE)

      CALL RESIDU(INTAB,COLTIM,COLX,COLMOD,COLSEG,TIME,XVAL,
                  MODULO,SEGM,NROW,NSEG,NBIN,NCOV,NECH,FMIN,
                  FRESTP,SIG2,VTME(PTRTHE))

      CALL TBL_INIT(OUTPUT,3,NECH)
```

```
        CALL TBL_DEFCOL(OUTPUT,'FREQ',COLFRE)

        CALL TBL_DEFCOL(OUTPUT,'FUNC',COLTHE)

        CALL TBL_DEFCOL(OUTPUT,'FTEST',COLFTE)

        CALL TBL_MAPCOL(OUTPUT,COLFTE,PTRFTE)

        FLAG = -1

        CALL FINEXT(FLAG,VTME(PTRTHE),VTME(PTRFTE),NECH,FMIN,
                    FRESTP,WIDTH,OUTPUT,COLFRE,COLTHE,COLFTE,
                    NRMIN)

        CALL MINFTE(OUTPUT,COLTHE,COLFRE,COLFTE,NRMIN,NDEGF1,
                    NDEGF2)

END
```

For VARMV :

SUBROUTINE VARMV (COLSEG, XVECT, SGVECT, NELE1, NBELEM, NSEG, SIG2, MEAN)

```
BEGIN

        SIG2 = 0

        MEAN = 0

        IF (NBELEM <> 0) THEN

            SUM = 0

            SUMSQ = 0

            ILST = NELE1 + NBELEM - 1
```

```
      IF (COLSEG = 0) THEN
          DO I = NELE1,ILST
              SUM = SUM + XVECT(I)
              SUMSQ = SUMSQ + XVECT(I)**2
          END DO
          MEAN = SUM/NBELEM
          IF (NBELEM > 1) SIG2 = (SUMSQ-SUM*SUM/NBELEM)/
                                 (NBELEM-NSEG)
      ELSE
          IBEG = NELE1
          DO ISEG = 1,NSEG
              DO I = IBEG,ILST
                  IF (SGVECT(I) <> ISEG) GO TO 20
                  SUM = SUM + XVECT(I)
                  SUMSQ = SUMSQ + XVECT(I)**2
                  NPTSEG = NPTSEG + 1
              END DO
20            IF (NPTSEG <> 0) THEN
                  MEAN = MEAN + (SUM/NPTSEG)
                  SIG2 = SIG2 + (SUMSQ-SUM*SUM/NPTSEG)
              END IF
              SUM = 0
```

```
                    SUMSQ = 0

                    NPTSEG = 0

                    IBEG = I

                END DO

                MEAN = MEAN/NSEG

                SIG2 = SIG2/ (NBELEM-NSEG)

            END IF

        END IF

END
```

For SETDEG :

SUBROUTINE SETDEG (NDAT, NSEG, NBIN, NCOV, NDEG, NDEGF1, NDEGF2)

```
BEGIN

    NDEG = (NDAT-NBIN*NSEG)*NCOV

    NDEGF1 = NDAT - NSEG

    NDEGF2 = NDEG

END
```

For RESIDU :

SUBROUTINE RESIDU (INTAB, COLTIM, COLX, COLMOD, COLSEG, TIME, XVAL, MODULO, SEGM, NROW, NSEG, NBIN, NCOV, NECH,

FMIN, FRESTP, SIG2X, THETA)

```
BEGIN
      IF (COLSEG = 0) THEN
          COLSOR(1) = COLMOD
          COLSOR(2) = COLX

              COLSOR(4) = COLMOD
              CALL TBL_SORT(INTAB,NSOR,COLSOR,1)
          END IF
          CALL MODFRE(TIME,MODULO,NROW,FREQ)
          IF (COLSEG <> 0) THEN
              COLSOR(2) = COLMOD
              COLSOR(4) = COLTIM
          END IF
          CALL TBL_SORT(INTAB,NSOR,COLSOR,1)
          CALL PHADIS(COLSEG,XVAL,MODULO,SEGM,NROW,NSEG,NBIN,NCOV,
                      SIG2X,THETA(IFREQ))
          FREQ = FREQ + FRESTP
      END DO
END
```

For MODFRE :

SUBROUTINE MODFRE (TIME, MODULO, NROW, FREQ)

```
BEGIN
      DO IROW = 1,NROW
          TIMPH = TIME(IROW)*FREQ
          MODULO(IROW) = TIMPH - IFIX(TIMPH)
      END DO
END
```

For PHADIS :

SUBROUTINE PHADIS (COLSEG, XVAL, MODULO, SEGM, NROW, NSEG, NBIN, NCOV, SIG2X, THETA)

```
BEGIN
      THETA = 0
      THETAF = 0
      ILOW = 1
      THIBIN = 1/NBIN
      THICOV = 1/NCOV*THIBIN
      NBELEM = 0
      DO ISEG = 1,NSEG
          BEGBNS = MODULO(ILOW)
```

```
      DO ICOV = 1,NCOV
          BEGBIN = BEGBNS
          ENDBIN = BEGBNS + THIBIN
          BINSUM = 0
          BINSSQ = 0
          BI1SUM = 0
          BI1SSQ = 0
          NPTBIN = 0
          NPTBI1 = 0
          DO IROW = ILOW,NROW
50            IF (COLSEG <> 0) THEN
                  IF (SEGM(IROW) > ISEG) GO TO 30
              END IF
              IF (MODULO(IROW) < = ENDBIN) THEN
                  XIROW = XVAL(IROW)
                  BINSUM = BINSUM + XIROW
                  BINSSQ = BINSSQ + XIROW*XIROW
                  NPTBIN = NPTBIN + 1
              ELSE
                  IF (BEGBIN < -1.E-8) THEN
                      BI1SUM = BINSUM
```

```
                    BI1SSQ = BINSSQ

                    NPTBI1 = NPTBIN

                ELSE

                    IF (NPTBIN <> 0) THEN

                        THETAF = THETAF + (BINSSQ-
                                 (BINSUM*BINSUM/NPTBIN))

                        NBELEM = NBELEM + NPTBIN

                    END IF

                END IF

                BEGBIN = ENDBIN

                ENDBIN = BEGBIN + THIBIN

                BINSUM = 0

                BINSSQ = 0

                NPTBIN = 0

                GO TO 50

            END IF

        END DO

30      NPTBIN = NPTBIN + NPTBI1

        IF (NPTBIN <> 0) THEN

            BINSUM = BINSUM + BI1SUM

            BINSSQ = BINSSQ + BI1SSQ

            THETAF = THETAF + (BINSSQ- (BINSUM*BINSUM/NPTBIN))
```

```
                    NBELEM = NBELEM + NPTBIN
                END IF
                BEGBNS = BEGBNS - THICOV
            END DO
            ILOW = IROW
        END DO
        NSAMP = NSEG*NCOV*NBIN
        NDIV = NBELEM - NSAMP
        DIV = NDIV*SIG2X
        IF (DIV > 1.E-4) THETA = THETAF/DIV
END
```

For FINEXT :

SUBROUTINE FINEXT (FLAG, EXTREM, VECAUX, NECH, FMIN, FRESTP, WIDTH, EXTTAB, COLFRE, COLEXT, COLAUX, NREXT)

```
BEGIN
        EQUIVALENCE (LSTELE,LSTELT(1))
        EQUIVALENCE (LSTFRE,LSTELT(2))
        EQUIVALENCE (LSTAUX,LSTELT(3))
        COLMIN(1) = COLEXT
        COLMIN(2) = COLFRE
```

```
IF (FLAG = 1) THEN
    NRCOL = 3
    COLMIN(3) = COLAUX
    LSTAUX = VECAUX(1)
ELSE
    NRCOL = 2
END IF
LSTINC = -WIDTH + 1
LSTMIN = 0
PREMIN = 0
INCREA = .FALSE.
NREXT = 0
LSTELE = (-FLAG)*EXTREM(1)
LSTFRE = FMIN
CURFRE = LSTFRE + FRESTP
DO I = 2,NECH
    CURELE = (-FLAG)*EXTREM(I)
    IF (FLAG = 1) CURAUX = VECAUX(I)
    IF (CURELE <> LSTELE) THEN
        IF (CURELE > LSTELE) THEN
            IF (NOT INCREA) THEN
```

```
            IF (LSTINC < = I-WIDTH) THEN
                NREXT = NREXT + 1
                PREMIN = LSTMIN
                LSTMIN = I - 1
                IF (NREXT < = 0) NREXT = 1
                LSTELE = (-FLAG)*LSTELE
                CALL TBL_WRROW(EXTTAB,NREXT,NRCOL,
                               COLMIN,LSTELT)
            END IF
            INCREA = .TRUE.
        END IF
        LSTINC = I
    ELSE
        IF (INCREA) THEN
            IF (I-1-LSTMIN < = WIDTH) THEN
                NREXT = NREXT - 1
                IF (NREXT > = 0) THEN
                    DO ICOL = 1,NRCOL
                        CALL TBL_DELEM(EXTTAB,NREXT+1,
                                       ICOL)
                    END DO
                END IF
```

```
                        LSTMIN = PREMIN
                    END IF
                    INCREA = .FALSE.
                END IF
            END IF
        ELSE
            IF (INCREA) LSTINC = I
        END IF
        LSTELE = CURELE
        LSTFRE = CURFRE
        IF (FLAG = 1) LSTAUX = CURAUX
        CURFRE = CURFRE + FRESTP
    END DO
END
```

For MINFTE :

SUBROUTINE MINFTE (MINTAB, COLTHE, COLFRE, COLFTE, NRMIN, NDEGF1, NDEGF2)

```
BEGIN
    DO I = 1,NRMIN
        CALL TBL_RDELEM(MINTAB,I,COLTHE,THETA)
```

```
            INVTHE = 1/THETA
            SIGNIF = PCHI(0,INVTHE,NDEGF1,NDEGF2,2)
            CALL TBL_WRELEM(MINTAB,I,COLFTE,SIGNIF)
        END DO
        COLSOR(1) = COLFTE
        COLSOR(2) = COLTHE
        COLSOR(3) = COLFRE
        CALL TBL_SORT(MINTAB,3,COLSOR,1)
END
```

For PCHI :

FUNCTION PCHI (CHI2, F, NDEG, ND2, NT)

```
BEGIN
        T29 = 0.22222222
        P1 = 0.33267
        A1 = 0.4361836
        A2 = -.1201676
        A3 = 0.9372980
        G1 = 0.3989423
        IF (CHI2 = 0) GO TO 10
```

```
      X2 = (CHI2/NDEG)** (1/3) - (1-T29/NDEG)
      X2 = X2/SQRT(T29/NDEG)
      GO TO 20
   10 CONTINUE
      F13 = F** (1/3)
      X2 = F13* (1-T29/ND2) - (1-T29/NDEG)
      X2 = X2/SQRT(T29/NDEG+F13**2*T29/ND2)
   20 CONTINUE
      IF (X2 > 0) GO TO 30
      X2 = -X2
      IPOS = 0
      IF (X2 > 10) GO TO 60
      GO TO 40
   30 CONTINUE
      IPOS = 1
      IF (X2 > 10) GO TO 70
   40 CONTINUE
      Z2 = G1*EXP(-X2**2/2)
      T = 1/ (1+P1*X2)
      PCHI = Z2* (T* (A1+T* (A2+T*A3)))
      IF (NT <> 2) GO TO 50
```

```
      PCHI = 2*PCHI
      RETURN
   50 CONTINUE
      IF (IPOS <> 0) PCHI = 1 - PCHI
      RETURN
   60 CONTINUE
      PCHI = 0
      RETURN
   70 CONTINUE
      PCHI = 1
      IF (NT = 2) PCHI = 0
END
```

For RENSON :

SUBROUTINE RENSON (INTAB, COLTIM, COLX, COLMOD, COLSEG, TIME, XVAL, MODULO, SEGM, NROW, NSEG, FMIN, NECH, FRESTP, ERMEAN, WIDTH, OUTPUT, NRMIN)

```
BEGIN
      CALL VARMV(COLSEG,XVAL,SEGM,1,NROW,NSEG,SIG2X,XMEAN)
      CALL DISCAR(XVAL,NROW,ERMEAN,DISPHA)
      STEP(1) = FRESTP
      START(1) = FMIN
```

```
        NPIX(1) = NECH

        CALL PUTIMAG_ST(OUTPUT,1,NPIX,START,STEP,PTRTHE)

        CALL RESREN(INTAB,COLTIM,COLX,COLMOD,COLSEG,TIME,XVAL,
                    MODULO,SEGM,NROW,NSEG,FMIN,FRESTP,NECH,DISPHA,
                    SIG2X,VTME(PTRTHE))

        CALL TBL_INIT(OUTPUT,2,NECH)

        CALL TBL_DEFCOL(OUTPUT,'FREQ',COLFRE)

        CALL TBL_DEFCOL(OUTPUT,'FUNC',COLTHE)

        CALL TBL_DEFCOL(OUTPUT,'AMPLI',COLAMP)

        CALL TBL_MAPCOL(OUTPUT,COLAMP,PTRAMP)

        FLAG = -1

        CALL FINEXT(FLAG,VTME(PTRTHE),VTME(PTRAMP),NECH,FMIN,FRESTP,
                    WIDTH,OUTPUT,COLFRE,COLTHE,COLAMP,NRMIN)

        IF (NRMIN > 0) THEN

            COLSOR(1) = COLTHE

            COLSOR(2) = COLFRE

            COLSOR(3) = COLAMP

            CALL TBL_SORT(OUTPUT,3,COLSOR,1)

        END IF

    END
```

For DISCAR :

SUBROUTINE DISCAR (XVAL, NELEM, ERMEAN, DISPHA)

```
BEGIN
      VALMAX = XVAL(1)
      VALMIN = XVAL(1)
      DO IELEM = 2,NELEM
          IF (XVAL(IELEM) > VALMAX) THEN
              VALMAX = XVAL(IELEM)
          ELSE
              IF (XVAL(IELEM) < VALMIN) VALMIN = XVAL(IELEM)
          END IF
       END DO
       DISPHA = (ERMEAN/4)* ((VALMAX+VALMIN)/ (VALMAX-VALMIN))
END
```

For RESREN :

SUBROUTINE RESREN (INTAB, COLTIM, COLX, COLMOD, COLSEG, TIME, XVAL, MODULO, SEGM, NROW, NSEG, FMIN, FRESTP, NECH, DISPHA, SIG2, THETA)

```
BEGIN
      IF (COLSEG <> 0) THEN
          COLSOR(1) = COLSEG
          COLSOR(2) = COLMOD
```

```
            COLSOR(3) = COLX

            COLSOR(4) = COLTIM

            NSOR = 4

        ELSE

            COLSOR(1) = COLMOD

            COLSOR(2) = COLX

            COLSOR(3) = COLTIM

            NSOR = 3

        END IF

        FREQ = FMIN

        DO IFREQ = 1,NECH

            CALL MODFRE(TIME,MODULO,NROW,FREQ)

            CALL TBL_SORT(INTAB,NSOR,COLSOR,1)

            CALL THEREN(XVAL,MODULO,SEGM,NROW,NSEG,DISPHA,SIG2,
                        THETA(IFREQ))

            FREQ = FREQ + FRESTP

        END DO

END
```

For THEREN :

SUBROUTINE THEREN (XVAL, MODULO, SEGM, NROW, NSEG, DISPHA, SIG2, THETA)

```
BEGIN
        THETA = 0
        THETAF = 0
        IF (NSEG = 1) THEN
            DIFVAL = XVAL(1) - XVAL(NROW)
            DIFMOD = MODULO(1) - MODULO(NROW)
            THETAF = (DIFVAL*DIFVAL)/ (DIFMOD+1+DISPHA)
            DO IROW = 2,NROW
                DIFVAL = XVAL(IROW) - XVAL(IROW-1)
                DIFMOD = MODULO(IROW) - MODULO(IROW-1)
                THETAF = THETAF + ((DIFVAL*DIFVAL)/ (DIFMOD+DISPHA))
            END DO
        ELSE
            INSEGB = 1
            DO IROW = 2,NROW
                IF (SEGM(IROW) = SEGM(INSEGB)) THEN
                    DIFVAL = XVAL(IROW) - XVAL(IROW-1)
                    DIFMOD = MODULO(IROW) - MODULO(IROW-1)
                    THETAF = THETAF + ((DIFVAL*DIFVAL)/ (DIFMOD+DISPHA))
                ELSE
                    DIFVAL = XVAL(INSEGB) - XVAL(IROW-1)
```

```
                    DIFMOD = MODULO(INSEGB) - MODULO(IROW-1)

                    THETAF = THETAF + ((DIFVAL*DIFVAL)/
                             (DIFMOD+1+DISPHA))

                    INSEGB = IROW

                END IF

            END DO

            DIFVAL = XVAL(INSEGB) - XVAL(IROW-1)

            DIFMOD = MODULO(INSEGB) - MODULO(IROW-1)

            THETAF = THETAF + ((DIFVAL*DIFVAL)/ (DIFMOD+1+DISPHA))

        END IF

        DIV = (NROW-1)*SIG2

        IF (DIV > 1E-04) THETA = THETAF/DIV

END
```

For DMING :

SUBROUTINE DMING (TIME, XVAL, NROW, FMIN, NECH, FRESTP, WIDTH, OUTPUT, SIGCLE, INCSPW, NRMAX)

```
BEGIN

        CALL INITX(XVAL,NROW)

        STEP(1) = FRESTP

        START(1) = FMIN

        NPIX(1) = NECH
```

```
CALL PUTIMAG_ST(OUTPUT,1,NPIX,START,STEP,PTRDFT)

START(1) = 0

CALL PUTIMAG_ST(OUTPUT//'SW',1,NPIX,START,STEP,PTRSPW)

CALL FTFREQ(TIME,XVAL,NROW,NECH,FRESTP,INCSPW,VTME(PTRDFT),
            VTME(PTRSPW))

CALL TBL_INIT(OUTPUT,3,NECH)

CALL TBL_DEFCOL(OUTPUT,'FREQ',COLFRE)

CALL TBL_DEFCOL(OUTPUT,'FUNC',COLDFT)

CALL TBL_DEFCOL(OUTPUT,'AMPLI',COLCLE)

CALL NORSPW(VTME(PTRSPW),NECH)

START(1) = FMIN

CALL PUTIMAG_ST(OUTPUT//'DC',1,NPIX,START,STEP,PTRDCV)

CALL CLEAN(VTME(PTRDFT),VTME(PTRSPW),VTME(PTRDCV),NECH,
           FMIN,FRESTP,SIGCLE)
FLAG = 1

CALL FINEXT(FLAG,VTME(PTRDCV),VTME(PTRDFT),NECH,FMIN,
            FRESTP,WIDTH,OUTPUT,COLFRE,COLCLE,COLDFT,
            NRMAX)

IF (NRMAX > 0) THEN

    COLSOR(1) = COLCLE

    COLSOR(2) = COLDFT

    COLSOR(3) = COLFRE

    CALL TBL_SORT(OUTPUT,3,COLSOR,-1)
```

```
            CALL TBL_RDELEM(OUTPUT,1,COLCLE,VALNOR)
            DO IROW = 1,NRMAX
                CALL TBL_RDELEM(OUTPUT,IROW,COLCLE,VALCLE)
                VALCLE = VALCLE/VALNOR
                CALL TBL_WRELEM(OUTPUT,IROW,COLCLE,VALCLE)
            END DO
        END IF
    END
```

For INITX :

SUBROUTINE INITX (XVAL, NROW)

```
    BEGIN
        XSUM = 0
        XMEAN = 0
        DO IROW = 1,NROW
            XSUM = XSUM + XVAL(IROW)
        END DO
        IF (NROW <> 0) XMEAN = XSUM/NROW
        DO IROW = 1,NROW
            XVAL(IROW) = XVAL(IROW) - XMEAN
        END DO
```

```
END
```

For FTFREQ :

SUBROUTINE FTFREQ (TIME, XVAL, NROW, NECH, FRESTP, INCSPW, DIFOTR, SPEWIN)

```
BEGIN
      PI2 = 6.283185005187988D0
      NROW2 = NROW*NROW
      FREQ = 0
      DO I = 1,NECH
          SPEWIN(I) = 0
          DIFOTR(I) = 0
      END DO
      DO IFREQ = 1,INCSPW - 1
          VALEXP = PI2*FREQ
          CALL FOUSPW(TIME,NROW,NROW2,VALEXP,SPEWIN(IFREQ))
          FREQ = FREQ + FRESTP
      END DO
      DO IFREQ = INCSPW,NECH
          VALEXP = PI2*FREQ
          CALL FOUTRA(TIME,XVAL,NROW,NROW2,VALEXP,
```

```
                        DIFOTR(IFREQ-INCSPW+1),SPEWIN(IFREQ))
            FREQ = FREQ + FRESTP
        END DO
        DO IFREQ = NECH - INCSPW + 2,NECH
            VALEXP = PI2*FREQ
            CALL FOUDFT(TIME,XVAL,NROW,NROW2,VALEXP,DIFOTR(IFREQ))
            FREQ = FREQ + FRESTP
        END DO
END
```

For FOUSPW :

SUBROUTINE FOUSPW (TIME, NROW, NROW2, VALEXP,SPWNND)

```
BEGIN
        RPSWFQ = 0
        IPSWFQ = 0
        DO IROW = 1,NROW
            EXPTIM = VALEXP*TIME(IROW)
            COSEXP = COS(EXPTIM)
            SINEXP = SIN(EXPTIM)
            RPSWFQ = RPSWFQ + COSEXP
            IPSWFQ = IPSWFQ + SINEXP
```

```
        END DO

        SPWNND = (RPSWFQ**2+IPSWFQ**2)/NROW2

    END
```

For FOUTRA :

SUBROUTINE FOUTRA (TIME, XVAL, NROW, NROW2, VALEXP, FOTRND, SPWNND)

```
    BEGIN

        RPFTFQ = 0

        IPFTFQ = 0

        RPSWFQ = 0

        IPSWFQ = 0

        DO IROW = 1,NROW

            EXPTIM = VALEXP*DBLE(TIME(IROW))

            COSEXP = COS(EXPTIM)

            SINEXP = SIN(EXPTIM)

            RPFTFQ = RPFTFQ + XVAL(IROW)*COSEXP

            IPFTFQ = IPFTFQ + XVAL(IROW)*SINEXP

            RPSWFQ = RPSWFQ + COSEXP

            IPSWFQ = IPSWFQ + SINEXP
        END DO
```

```
      FOTRND = (RPFTFQ**2+IPFTFQ**2)/NROW2
      SPWNND = (RPSWFQ**2+IPSWFQ**2)/NROW2
END
```

For FOUDFT :

SUBROUTINE FOUDFT (TIME, XVAL, NROW, NROW2, VALEXP, FOTRND)

```
BEGIN
      RPFTFQ = 0
      IPFTFQ = 0
      DO IROW = 1,NROW
          EXPTIM = VALEXP*TIME(IROW)
          COSEXP = COS(EXPTIM)
          SINEXP = SIN(EXPTIM)
          RPFTFQ = RPFTFQ + XVAL(IROW)*COSEXP
          IPFTFQ = IPFTFQ + XVAL(IROW)*SINEXP
      END DO
      FOTRND = (RPFTFQ**2+IPFTFQ**2)/NROW2
END
```

For NORSPW :

SUBROUTINE NORSPW (VECSPW, NPOINT)

```
BEGIN
      IMAX = 1
      DO IPOINT = 2,NPOINT
          IF (VECSPW(IPOINT) > VECSPW(IMAX)) IMAX = IPOINT
      END DO
      VALMAX = VECSPW(IMAX)
      DO IPOINT = 1,NPOINT
          VECSPW(IPOINT) = VECSPW(IPOINT)/VALMAX
      END DO
END
```

For CLEAN :

SUBROUTINE CLEAN (IMABDF,IMASPW,IMADCV, NECH, FMIN, FRESTP, SIGCLE)

```
BEGIN
      IMASCR = 'SCRIMA.BDF'
      START(1) = FMIN
      STEP(1) = FRESTP
      NPIX(1) = NECH
      CALL PUTIMAG_ST(IMASCR,1,NPIX,START,STEP,PTRSCR)
      CALL FILL(IMABDF,NECH,VTME(PTRSCR))
```

```
      DO IPIX = 1,NECH
          IMADCV(IPIX) = 0
      END DO
      CALL FINMAX(VTME(PTRSCR),NECH,SIGCLE,IMAX,CLEMAX,0,TRCLEA)
      AMPMAX = CLEMAX
   10 CALL COMCLE(VTME(PTRSCR),IMASPW,NECH,IMAX,CLEMAX,IMADCV)
      CALL FINMAX(VTME(PTRSCR),NECH,SIGCLE,IMAX,CLEMAX,AMPMAX,TRCLEA)
      IF (TRCLEA) GO TO 10
END
```

For FILL :

SUBROUTINE FILL (INVEC, NPOINT, OUTVEC)

```
BEGIN
      DO IVAL = 1,NPOINT
          OUTVEC(IVAL) = INVEC(IVAL)
      END DO
END
```

For FINMAX :

SUBROUTINE FINMAX (VECT, NPOINT, SIGCLE, IMAX, CLEMAX, AMPMAX, TRCLEA)

```
BEGIN
      IMAX = 1
      IMIN = 1
      CLEMAX = 0
      DO IPOINT = 1,NPOINT
          IF (VECT(IPOINT) > CLEMAX) THEN
              IMAX = IPOINT
              CLEMAX = VECT(IPOINT)
          ELSE
              IF (VECT(IPOINT) < = VECT(IMIN)) IMIN = IPOINT
          END IF
      END DO
      TRCLEA = (VECT(IMIN) > (-2*SIGCLE*AMPMAX)) AND
               ((CLEMAX-VECT(IMIN)) > (SIGCLE*AMPMAX))
END
```

For COMCLE :

SUBROUTINE COMCLE (INVEC1, INVEC2, NPOINT, IMAX, CLEMAX, OUTVEC)

```
BEGIN
      PARAMETER (CLECST=0.17)
```

```
        VALCLE = CLECST*CLEMAX
        DO IVAL = IMAX,NPOINT
            INVEC1(IVAL) = INVEC1(IVAL) - (VALCLE*INVEC2(IVAL-IMAX+1))
        END DO
        DO IVAL = IMAX - 1,1,-1
            INVEC1(IVAL) = INVEC1(IVAL) - (VALCLE*INVEC2(-IVAL+IMAX+1))
        END DO
        OUTVEC(IMAX) = OUTVEC(IMAX) + VALCLE
    END
```

For DREPDM :

SUBROUTINE DREPDM (INTAB, NELEM)

```
    BEGIN
        COLOUT(1) = 1
        COLOUT(2) = 2
        COLOUT(3) = 3
        OUTTEX = 'MOST PROBABLE PERIODS'
        WRITE (OUTTEX)
        OUTTEX = 'FREQ    THETA    F-TEST'
        WRITE (OUTTEX)
        DO IELEM = 1,NELEM
```

```
        CALL TBL_RDROW(INTAB,IELEM,3,COLOUT,VALOUT)
        OUTTEX =  (VALOUT(1), VALOUT(2), VALOUT(3))
        WRITE (OUTTEX)
    END DO
END
```

For DRESVM :

SUBROUTINE DRESVM (INTAB, NELEM)

```
BEGIN
    COLOUT(1) = 1
    COLOUT(2) = 2
    OUTTEX = 'MOST PROBABLE PERIODS'
    WRITE (OUTTEX)
    OUTTEX = 'FREQ    THETA'
    WRITE (OUTTEX)
    DO IELEM = 1,NELEM
        CALL TBL_RDROW(INTAB,IELEM,2,COLOUT,VALOUT)
        OUTTEX = (VALOUT(1), VALOUT(2))
        WRITE (OUTTEX)
    END DO
```

```
END
```

For DREDFT :

SUBROUTINE DREDFT (INTAB, NELEM)

```
BEGIN
      COLOUT(1) = 1
      COLOUT(2) = 2
      COLOUT(3) = 3
      OUTTEX = 'MOST PROBABLE PERIODS'
      WRITE (OUTTEX)
      OUTTEX = 'FREQ    THETA    CLEAN'
      WRITE (OUTTEX)
      DO IELEM = 1,NELEM
          CALL TBL_RDROW(INTAB,IELEM,3,COLOUT,VALOUT)
          OUTTEX = (VALOUT(1), VALOUT(2), VALOUT(3))
          WRITE (OUTTEX)
       END DO
END
```

Program code

See appendix 4.

5.2.5 SHOW/TSA

Remembering

The command causes the display of the actual value of the different qualifiers of the TSA package.
It has no input and gives as outputs :

- the method's name,
- the number of printed periods at the end of the treatment,
- half the number of trial frequencies between two successive extrema,
- the prefix of the name of the default output.

Program design

In this hierarchy, we have three levels :

- level 1 : command
- level 2 : display informations
- level 3 :
  - * display the method 's name
  - * display the number of printed periods at the end of the treatment
  - * display the number of trial frequencies between two successive extrema
  - * display the prefix of the default output.

The hierarchy of SHOW/TSA is visualized in Fig 5.4.

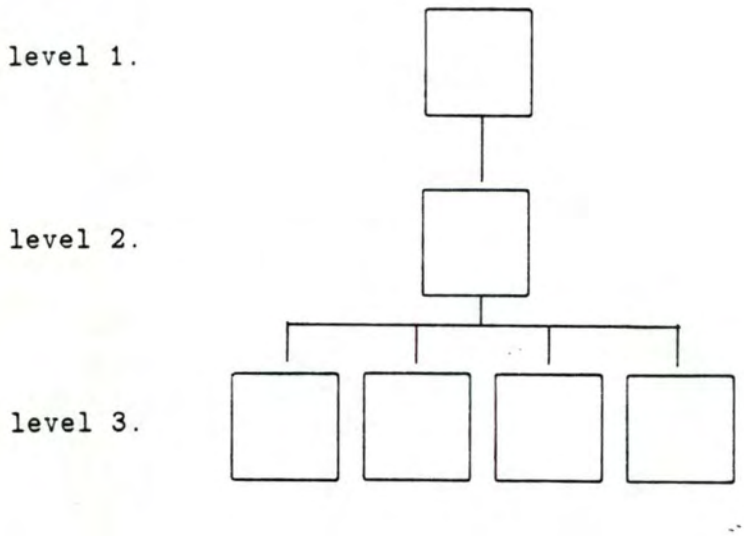


Fig. 5.4 Hierarchy of SHOW/TSA.

Variables used

TSAKEY name of the TSA key containing all the values
array of 20 characters

Pseudo code

```
WRITE OUT  METHOD = 'TSAKEY(1:5)'

WRITE OUT  PRINT  = 'TSAKEY(6:8)'

WRITE OUT  WIDTH  = 'TSAKEY(9:11)'

WRITE OUT  OUTPUT = 'TSAKEY(12:14)'
```

Program code

See appendix 4.

5.2.6 HELP/TSA

Remembering

This causes the display of all informations concerning the commands of the TSA package.

This command has no input, and gives as output the informations about each command.

Note : It was not possible to use the same partition for this command as for the other. Only the informations displayed can be seen.

Informations displayed

See appendix 4.

5.2.7 TUTORIAL/TSA

Remembering

This command tries to help the user of the TSA package.
The command has no input and produced the execution of the different commands of package.

Program design

The hierarchy of this command has only three levels.

The first level is the one of the command.
The second level is :

- to give explanations
- to proceed the execution

The third level corresponding to the execution, is divided in :

- copy the table with the datas in the user 's directory,
- select the Phase Dispersion Minimization,

- give a value to the different method 's parameters and execute the chosen method,
- show the value of the different parameters,
- select the Discrete Fourier Transform method,
- give a value to the different method 's parameters and execute the chosen method,
- select the Signal Variation Minimization method,
- give a value to the different method 's parameters and execute the chosen method,
- delete all the files created for the execution.

The hierarchy of this command can be shown if Fig 5.5.

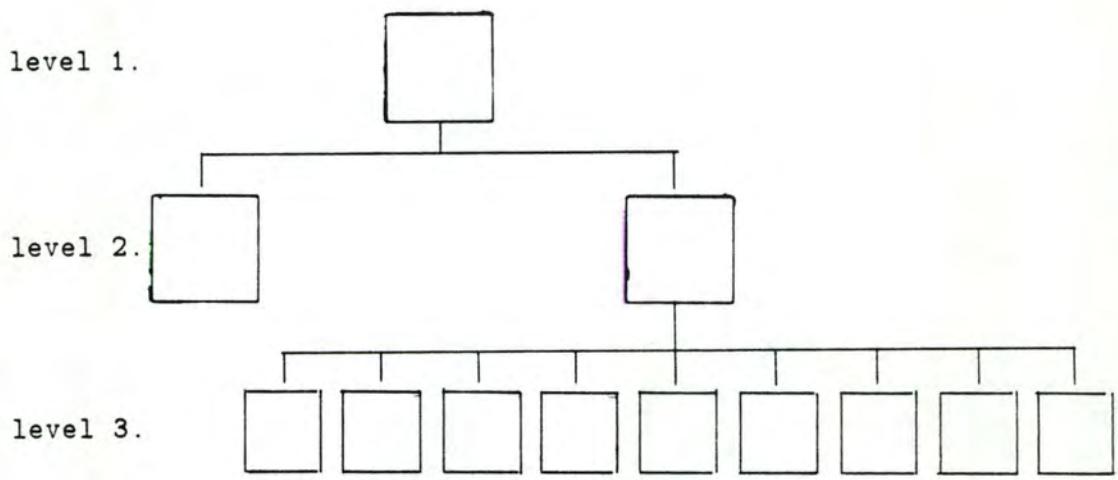


Fig. 5.5 Hierarchy of TUTORIAL/TSA.

Variables used

No particular variables are used.

Pseudo code

```
WRITE OUT  'EXPLANATIONS'

COPY  TSATUT.TBL TSATUT.TBL

SET/CONTEXT  TSA

SET/TSA  METHOD = PDM  PRINT = 4 WIDTH = 2

TSA/TABLE  0.1,0.3,20,5,2  TSATUT :T :X,:S TUTPDM

SHOW/TSA

SET/TSA  METHOD = DFT

TSA/TABLE  0,0,0 TSATUT :T :X,:S TUTDFT

SET/TSA  METHOD = SVM

TSA/TABLE  0.2,0.5,0,0 TSATUT :T :X,:S TUTSVM

DELETE TUT*.*;*, TSATUT.TBL;*
```

Program code

See appendix 4.

5.3 Toolpack program

All the tools from Toolpack presented in section 4.1 have been utilised on the Fortran-77 program designed in section 5.2.

And particular results from this tools are presented in appendix 3.

Chapter 6

Tests and results

Now that the implementation is ended, we would like to have a look at two important points. These are :

- the phase of testing the package (section 6.1 and 6.2),
- the results produced by the package (section 6.3).

Three different types of tests consisting of a static analysis, a fortran portability and a numerical tests are realized on the Fortran programs.
In a second step, particular tests of the MIDAS 's procedures are made.

6.1 Fortran programs

6.1.1 Static Analysis

In this phase of the static analysis of the Fortran programs, we have proceed at a lexical and a syntaxic analysis of the source code file. This has bee realized in using the scanner and the parser of Toolpack (see section 5.1.2).

6.1.2 Fortran portability

As the MIDAS software must become a portable version in January 1988, all the new written programs must be portable.
We have checked the portability of our TSA package by using the static analyser of Toolpack (see section 5.1.2).

Note : our package has no particularity except the %VAL function which allow the using of virtual memory.

6.1.3 Numerical tests

This numerical tests concern the subroutines of the command TSA/TABLE.

To test these subroutines, we have used the methodology described by A. Van Lamswerde in his course intituled : "Méthodologie de développement de logiciels".[1]

To see if a subroutine is in adequation with its specification, we have three steps to make :

- the first corresponds to the description of the *attending* results produced by the subroutine, using only the specification;
- the second step is to identify the results *obtained* by the execution
- the third step is the *comparaison* between the attending results and the obtained ones; if these results are equal, we can say the subroutine respects its specification.

As this phase is a little too long, we have not described it in this document.

6.1.4 MIDAS 's procedures

To test the MIDAS 's procedures, we have used, more or less, the same methodology as the one used for the numerical tests.

- For SET/TSA :
  the only way to test this command is to execute all the other of the package, when possible to determine that, this command has done its work.
- For SET/TSA :
  as this command has to join the value to the TSA keyword; while the execution we must visualize the value of the keyword and compare them with that put.
- For TSA/TABLE :
  to give a data' set in input, we must execute this command with all parameters. And on the other side, we execute the previous received package with the same value for each parameters; and after this, we compare the different results obtained by the two packages.
  It must be note that the results obtained by the two packages are not really the same. But it seems to be in advantage for us, as in the software received no accurate attention is made during the computations.
- For SHOW/TSA :
  knowing the value of the TSA keyword, and after execution of this command, we can compare if it gives the good results.

[1] see [Van Lamsweerde]

- For HELP/TSA :
  the only way to test this command is to execute it, read the given informations and compare then with the one requested.
- For TUTORIAL/TSA :
  to test this command, we have only to execute it and see if it does everything asked for.

6.2 Example of an execution

Now that we are sure that the package is giving good results, we can execute it, and look at the results obtained.

For this particular execution, we will use a set of observations about *TU CAS* from the years 1966 to the years 1977.
These observations are divided into four seasons, for us we will say in four segments.

The input table corresponds to the one described in Fig. 6.1.

```
 TABLE : TUCAS
SELECT  :T.GT.10

Seq.no.        T          X          S

      1      37936.969 8.010     1.000000
      2      37937.945 7.550     1.000000
      3      37938.949 8.040     1.000000
      4      37939.969 7.630     1.000000
      5      37940.664 7.910     1.000000
      6      37940.949 7.900     1.000000
      9      38219.852 7.252     2.000000
     10      38220.828 7.908     2.000000
     11      38220.895 7.926     2.000000
     12      38221.781 7.584     2.000000
     13      38221.887 7.407     2.000000
     14      38225.906 8.052     2.000000
     15      38228.910 7.518     2.000000
     16      38229.910 7.988     2.000000
     17      38231.824 8.011     2.000000
     18      38232.926 7.574     2.000000
     19      38233.832 7.846     2.000000
     20      38234.801 7.689     2.000000
     21      38234.898 7.367     2.000000
```

22	38235.902	7.996	2.000000
25	43015.902	8.050	3.000000
26	43017.891	8.024	3.000000
27	43052.578	7.198	3.000000
28	43052.715	7.147	3.000000
29	43052.824	7.292	3.000000
30	43052.941	7.457	3.000000
31	43057.574	7.694	3.000000
32	43057.703	7.787	3.000000
33	43057.832	7.892	3.000000
34	43061.559	7.574	3.000000
35	43061.652	7.601	3.000000
36	43061.754	7.633	3.000000
37	43061.867	7.681	3.000000
38	43066.723	7.936	3.000000
39	43066.801	7.960	3.000000
40	43066.895	7.977	3.000000
41	43073.605	8.076	3.000000
42	43073.820	7.707	3.000000
43	43092.605	7.973	3.000000
44	43092.637	7.981	3.000000
45	43094.734	8.070	3.000000
46	43094.758	8.085	3.000000
47	43098.625	7.866	3.000000
48	43104.680	7.719	3.000000
49	43104.832	7.831	3.000000
50	43126.707	8.047	3.000000
51	43403.762	7.489	4.000000
52	43403.883	7.620	4.000000
53	43403.973	7.705	4.000000
54	43404.625	8.021	4.000000
55	43404.742	8.044	4.000000
56	43405.629	7.539	4.000000
57	43405.699	7.574	4.000000
58	43405.762	7.606	4.000000
59	43405.832	7.646	4.000000
60	43405.887	7.658	4.000000
61	43405.965	7.686	4.000000
62	43406.008	7.706	4.000000
63	43406.637	7.859	4.000000
64	43406.730	7.886	4.000000
65	43406.816	7.903	4.000000
66	43406.891	7.937	4.000000
67	43407.000	7.962	4.000000

68	43407.625	7.744	4.000000
69	43407.723	7.427	4.000000
70	43407.770	7.262	4.000000
71	43407.871	7.192	4.000000
72	43407.949	7.248	4.000000
73	43408.012	7.328	4.000000
74	43408.621	7.889	4.000000
75	43408.684	7.928	4.000000
76	43408.738	7.953	4.000000
77	43408.813	7.980	4.000000
78	43411.621	7.882	4.000000
79	43411.707	7.836	4.000000
80	43411.836	7.754	4.000000
81	43411.910	7.707	4.000000
82	43412.012	7.625	4.000000
83	43492.652	8.087	4.000000
84	43500.598	7.817	4.000000

Fig. 6.1. Table with the value of TU CAS.

We can set the context of the time series analysis with the command `SET/CONTEXT TSA`.

Suppose that we want to execute the Phase Dispersion Minimization, with the default value for all the parameters except for :

- the name of the input table,
- the reference to the column of the time,
- the reference to the column of the values,

In correspondance, we have the two commands :

- `SET/TSA METHOD = PDM`,
- `TSA/TABLE ? TUCAS :T, :X,:S`.

The results of this execution are :

- a table described in Fig 6.2,
- an image plotted in Fig 6.3.

TABLE : TSAPDM

Seq.no.	FREQ	FUNC	FTEST
1	0.15703	0.59537	0.10955E-01
2	0.18050	0.62726	0.22135E-01
3	0.18591	0.64323	0.30407E-01
4	0.16425	0.79013	0.24695
5	0.12635	0.82926	0.35645
6	0.34295E-01	0.83825	0.38450
7	0.13176	0.85009	0.42290
8	0.63174E-01	0.85914	0.45328
9	0.90249E-01	0.90410	0.61540
10	0.10288	0.90689	0.62595
11	0.14620	0.90698	0.62630
12	0.11010	0.91997	0.67587
13	0.41514E-01	0.92437	0.69284
14	0.16245E-01	0.94872	0.78767
15	0.13718	0.96597	0.85529

Fig. 6.2 Table corresponding to the PDM method.

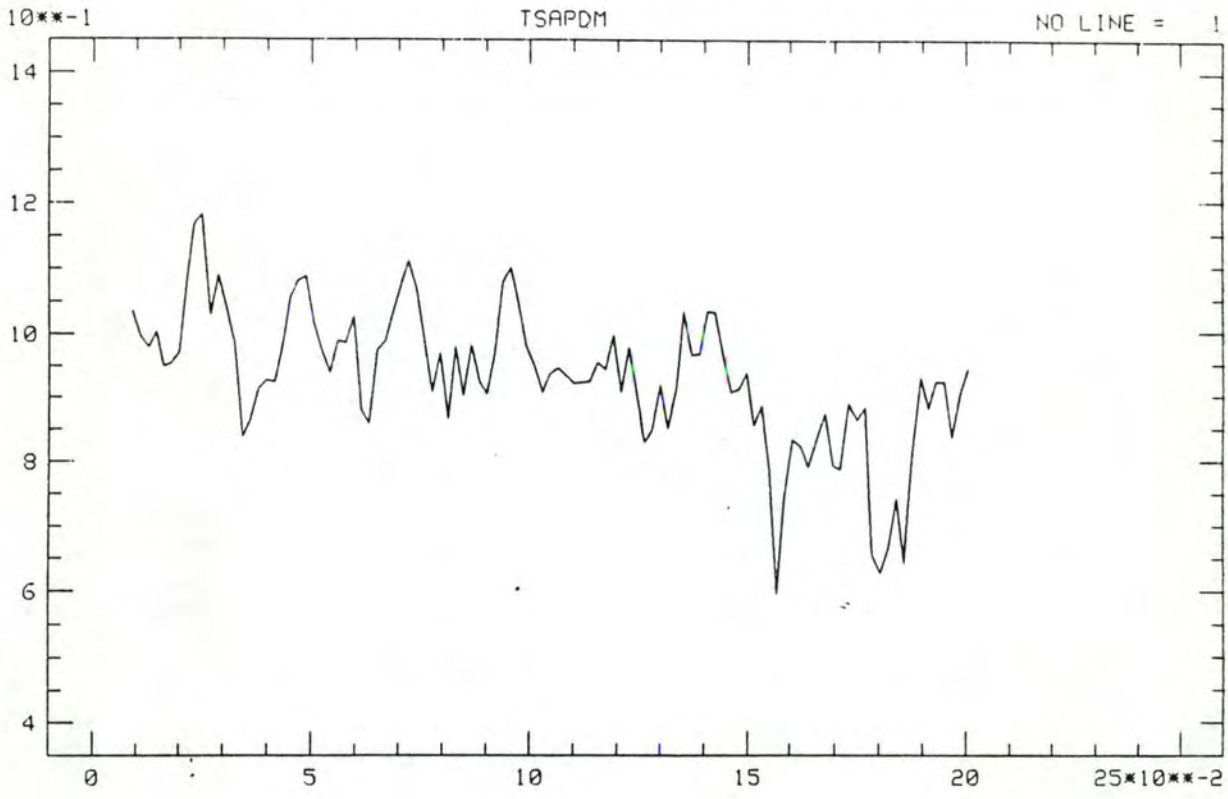


Fig. 6.3 Image corresponding to the PDM method.
Horizontal Axe = Tested frequencies.
Vertical Axe = Phase dispersion corresponding.

We want to execute the Signal Variation Minimization with the default value for all the parameters except for :

- the name of the input table,
- the reference to the column of the time,
- the reference to the column of the values.

The commands corresponded will be :

- `SET/TSA METHOD = SVM.`
- `TSA/TABLE ? TUCAS :T, :X,:S.`

The results of this execution are :

- a table described in Fig 6.4,
- an image plotted in Fig 6.5.

TABLE : TSASVM

Seq.no.	FREQ	FUNC	AMPLI
1	0.15703	1.7358	*
2	0.74004E-01	2.0233	*
3	0.10469	2.1319	*
4	0.63174E-01	2.1388	*
5	0.12635	2.1796	*
6	0.12093	2.2536	*
7	0.18050	2.2834	*
8	0.36100E-01	2.2924	*
9	0.45124E-01	2.3444	*
10	0.16967	2.4169	*
11	0.13898	2.4240	*
12	0.18050E-01	2.4343	*
13	0.84834E-01	2.4647	*
14	0.18772	2.4745	*
15	0.90249E-02	2.5209	*
16	0.50539E-01	2.5894	*
17	0.19494	3.1953	*

Fig. 6.4 Table corresponding to the SVM method.

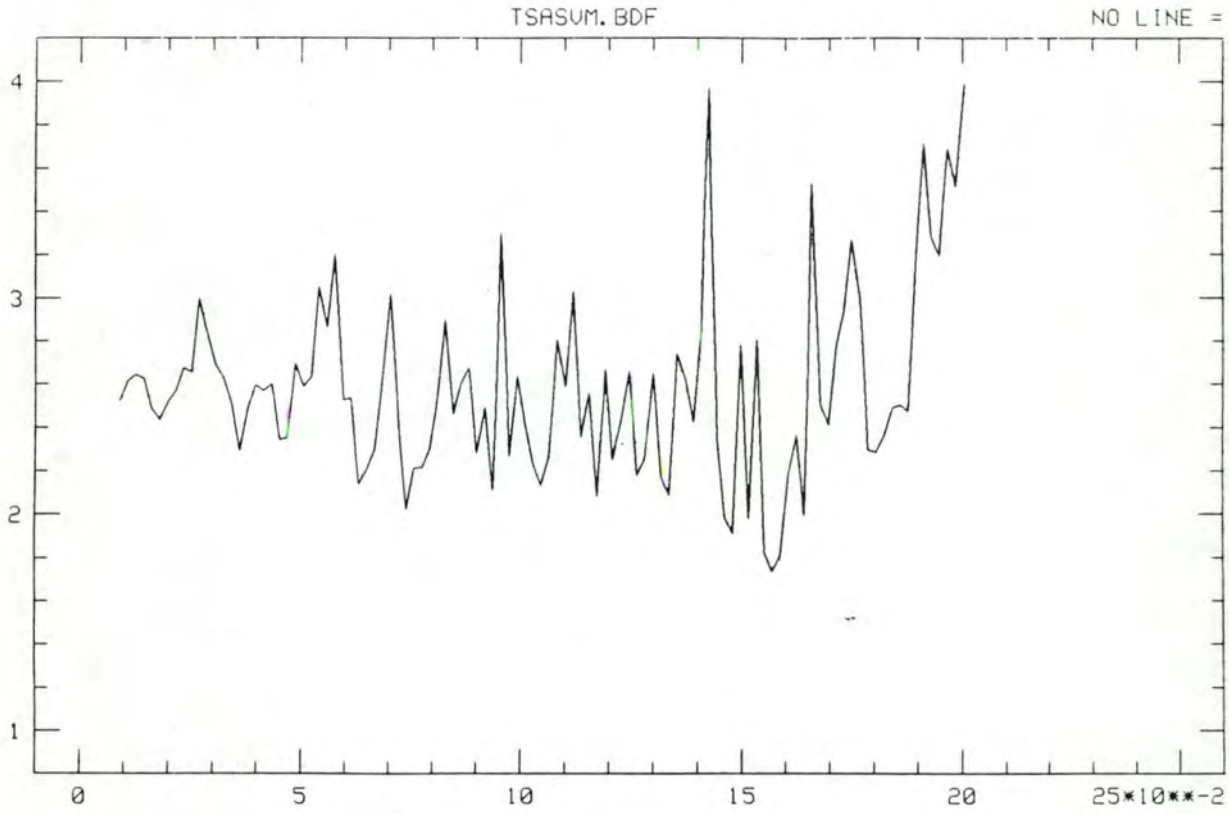


Fig. 6.5 Image corresponding to the SVM method.
Horizontal Axe = Tested frequencies.
Vertical Axe = Signal variation corresponding.

If it is the Discrete Fourier Transform method that has to be executed, with the default value for all parameters except :

- the name of the input table,
- the reference to the column of the time,
- the reference to the column of the values.

The commands corresponded will be :

- `SET/TSA METHOD = DFT.`
- `TSA/TABLE ? TUCAS :T, :X,:S.`

The results of this execution are :

- a table described in Fig 6.6,
- the image with the Fourier Transform at each point of the discretization plotted in Fig 6.7,
- the image with the spectral window at each point of the discretization plotted in Fig 6.8,
- and the image with the deconvolved spectrum plotted in Fig 6.9.

TABLE : TSADFT

Seq.no.	FREQ	FUNC	AMPLI
1	0.18230	1.0000	1.0000
2	0.86639E-01	0.64006	0.36816
3	0.32490E-01	0.55914	0.17745

Fig. 6.6 Table corresponding to the DFT method.

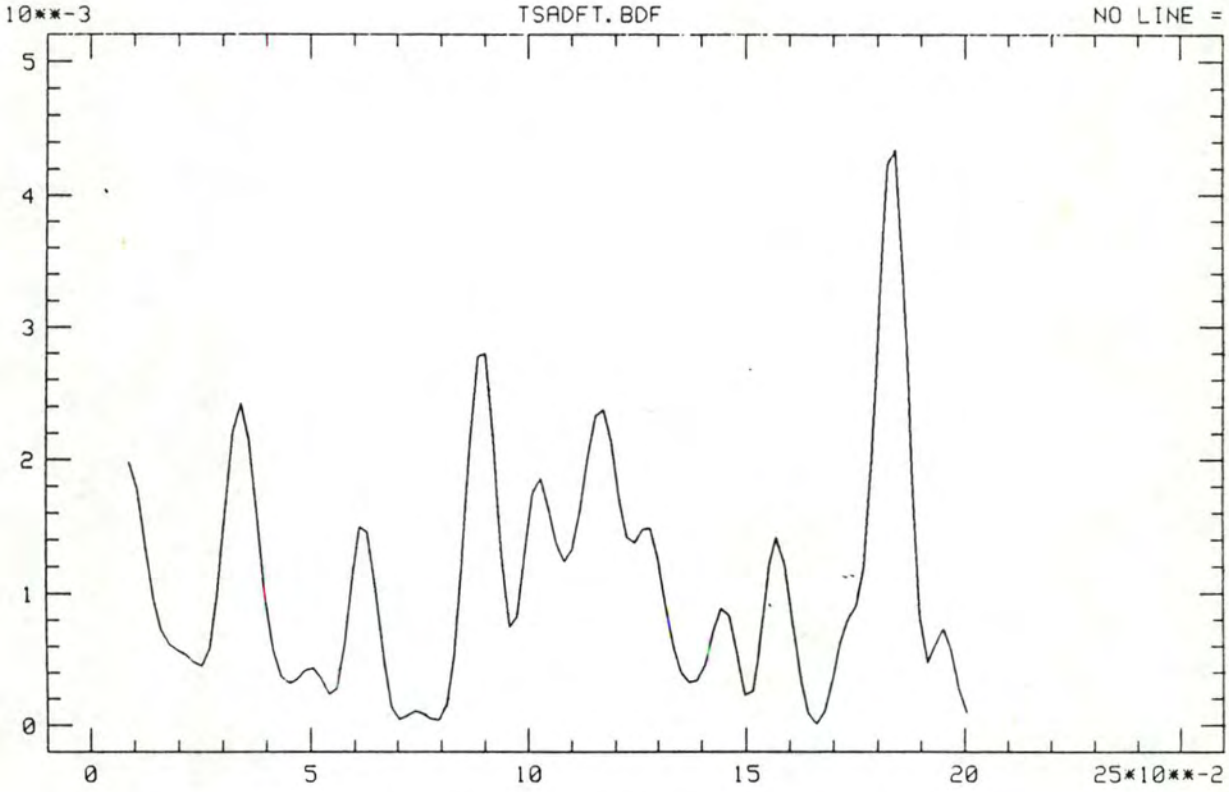


Fig. 6.7 Image corresponding to the DFT method.
Horizontal Axe = Tested frequencies.
Vertical Axe = Fourier transform corresponding.

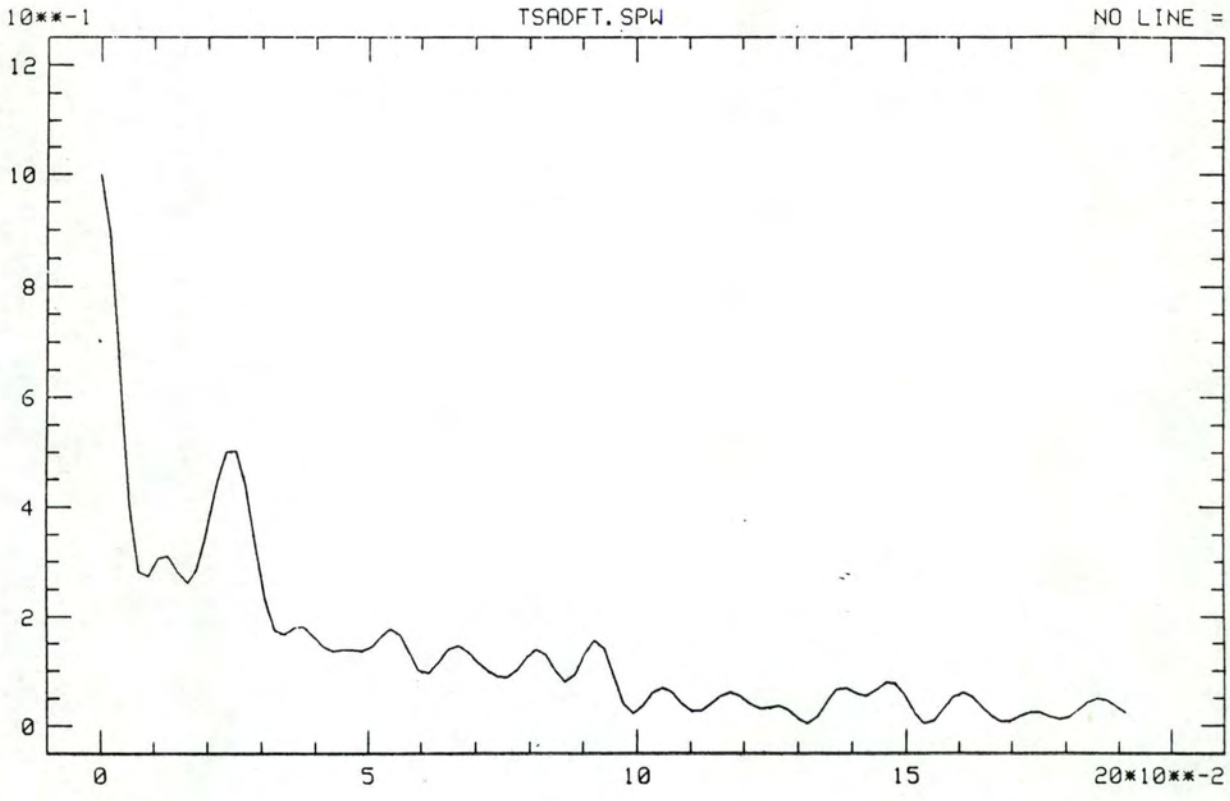


Fig. 6.8 Image corresponding to the spectral window.
Horizontal Axe = Tested frequencies.
Vertical Axe = Spectral window corresponding.

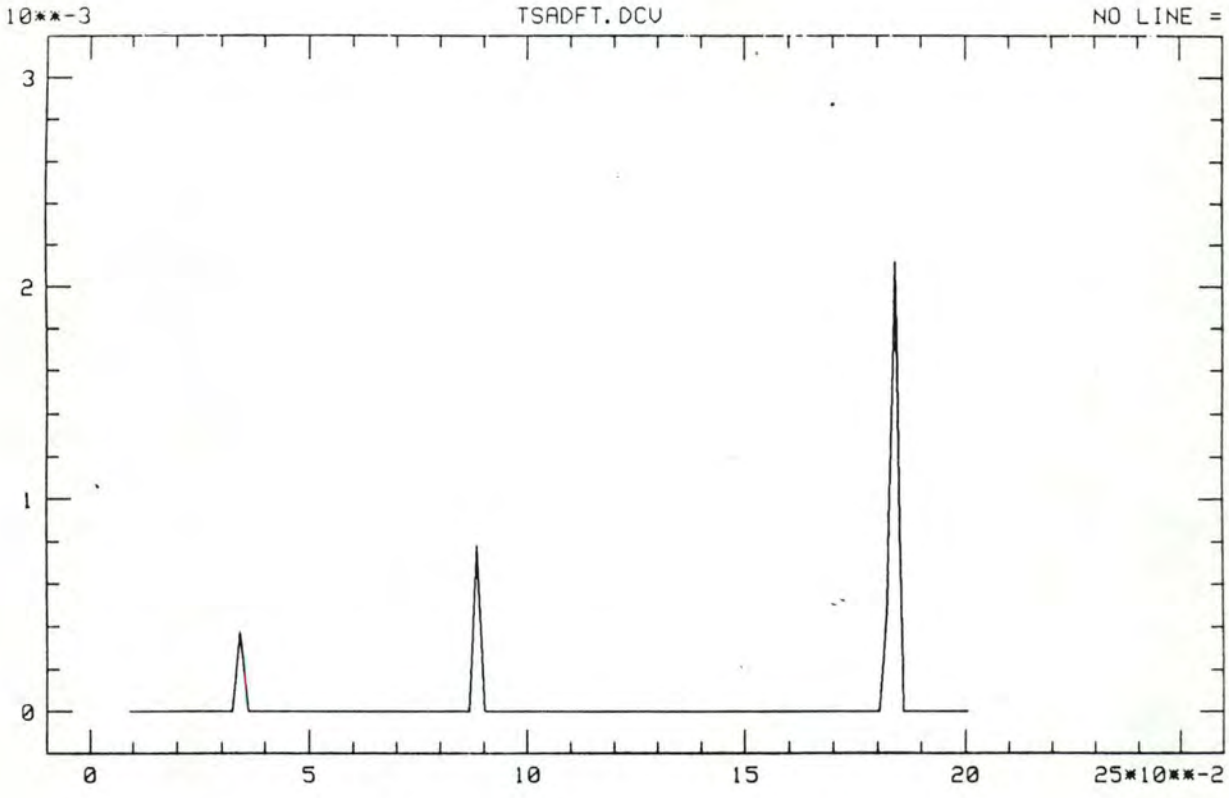


Fig. 6.9 Image corresponding to the deconvolved spectrum. Horizontal Axe = Tested frequencies.
Vertical Axe = Deconvolved spectrum corresponding.

The logfile corresponding to this particular execution can be found in appendix 1.
In conclusion of this execution, we can see :

- that the Phase Dispersion Minimization method and the Signal Variation Minimization method give the same results,
- nevertheless the Signal Variation Minimization gives more intermediated frequencies, periods,
- the results obtained by the Discrete Fourier Transform are more or less the same as the ones produced by the two other methods.

To have a more complete study of these three methods, other executions with particular sets of input data have to be realized.

But only with this execution can we have the same conclusions as the one obtained by Heck, Manfroid and Mersch[2].
None of the methods is clearly superior to the others. Nevertheless Fourier 's and non-parametric methods have relatively different scopes and they are, to some extend, complementary.

[2]see [Heck-Manfroid-Mersch]

Chapter 7

Performances, possible improvments

This chapter deal with the point concerning the performance tests of our package.

This point is consider to be important for different reasons.
The first one can be the user 's request, precisely if the computations of the program take more time that manual computations, this package is not used over long periods.
The second reason is that the performance analysis is an operation that can help in underlining those parts of the process taking too much time, in such a way that one can eventually find a way out , or at least, propose different remedies.

The study of performance is divided into two steps :

1. first the number of theoritical operations is count for each method,
2. in a second step, a dynamic analysis of the Fortran programs is made.

7.1 Theoritical operation counts

The method's performances will be done in terms of number of computations made.

7.1.1 Phase Dispersion Minimization

As explained in setion 3.2.2, the Phase Dispersion Minimization is divided into three parts of computations :

1. the computations of the variance of the all measurements (this correponds to the subroutine VARMV),
2. the computations of the phase vector (this corresponds to the subroutine MODFRE),

3. the computations of the overall variance of the bins and covers, the phase dispersion (this corresponds to the subroutine PHADIS).

- To compute the variance of all the segments, two variables are important. These are :
  - the number of segments NBS,
  - the number of measurements NBE.

  For one segment, if we have NBES elements in this segment, we will do (2 * NBES) additions and NBES multiplications (to compute the sum and the sumsquare), 1 multiplication, 2 divisions, 2 additions, 1 substraction.
  As the sum of the number of elements of each segment is equal to the number of elements NBE.
  So we have : $\sum_{i=1}^{NBS} NBES_i = NBE$

  To compute the variance of the all segments, we will do :
  - (NBE + NBS) multiplications
  - (2 * (NBS + 1)) divisions
  - (2 * (NBE + NBS)) additions
  - (NBS + 1) substractions.
- The computations of the phase vector depends of two variables :
  - the number of measurements NBE,
  - the number of trial frequencies NBF.

  For one frequency, we have to do :
  - NBE multiplications
  - NBE substractions.

  So, for NBF frequencies, we will do :
  - (NBF * NBE) multiplications
  - (NBF * NBE) substractions.
- For the computations of the phase dispersion, we have to know :
  - the number of trial frequencies NBF,
  - the number of segments NBS,
  - the number of measurements NBE,
  - the number of bins NBB,
  - the number of covers NBC.

For 1 frequency, 1 segment, 1 cover, 1 bin having NBEB elements; we do :

- (NBEB + 1) multiplications
- 1 division
- ((2 * NBEB) + 1) additions
- 1 substraction.

For 1 frequency, 1 segment, 1 cover and NBB bins having NBEB elements each, we do :

- (NBB * (NBEB + 1)) multiplications
- NBB divisions
- (NBB * ((2 * NBEB) + 1)) additions
- (NBB + 1) substractions.

For 1 frequency, 1 segment, NBC covers and NBB bins having NBEB elements each, we do :

- (NBC * (NBB * (NBEB + 1))) multiplications
- (NBC * NBB) divisions
- (NBC * (NBB * ((2 * NBEB) + 1))) additions
- (NBC * (NBB + 1)) substractions.

For 1 frequency, NBS segments, NBC covers and NBB bins having NBEB elements each, knowing that :
$\sum_{i=1}^{NBB} NBEB_i = NBES$ with NBES, the number of elements in each segment
$\sum_{i=1}^{NBS} NBES_i = NBE$
we do :

- (NBC * (NBB * (NBE + NBS))) + 3 multiplications
- (NBS * (NBC * NBB))) + 1 divisions
- (NBC * (NBB * ((2 * NBE) + NBS))) additions
- (NBS * (NBC * (NBB + 2))) substractions.

And for NBF frequencies, we do :

- (NBF * (NBC * (NBB * (NBE + NBS))) + 3) multiplications
- (NBF * (NBS * (NBC * NBB))) + 1) divisions
- (NBF * (NBC * (NBB * ((2 * NBE) + NBS)))) additions

- (NBF * (NBS * (NBC * (NBB + 2)))) substractions.

- And NBF additions to increment the frequency.

For the all method, if :

- NBF is the number of trial frequencies,
- NBS is the number of segments,
- NBE is the number of measurements,
- NBB is the number of bins,
- NBC is the number of covers,

the number of computations is resumed in Fig 7.1.

multiplications	divisions	additions	substractions
NBE+NBS NBF*NBE NBF*(NBC*NBB*(NBE+NBS)+3)	2*(NBS+1) NBF*(NBS*NBC*NBB+1)	2*(NBE+NBS) NBF*NBC*NBB*(NBE*2+NBS) NBF	NBS+1 NBF+NBE NBF*NBS*NBC*(NBB+2)
NBF*(((NBC*NBB)*(NBE+NBS))+NBE+3)	(2*(NBS+1)+(NBF*(NBS+NBC*NBB+1))	2*(NBE+NBS)+(NBF*(1+(NBC*NBB)*(NBE+NBF+NBS)))	NBF*(1+(NBS*NBC)*(NBB+2)))+NBS+NBE+1

Fig. 7.1 Number of computations required by the PDM method.

7.1.2 Signal Variation Minimization

Described in section 3.2.3, this method is divided in four phases of computations :

- the computations of the variance of the all measurements (this correponds to teh subroutine VARMV),

- the computations of the discard in phase (this corresponds to the subroutine DISCAR),
- the computations of the phase vector (this corresponds to the subroutine MODFRE),
- and the computations of the phase dispersion (this corresponds to the subroutine THEREN).
- To compute the variance, as we used the same pocedure as the one described for the Phase Dispersion Minimization, the number of computations, for NBES segments and NBE elements, is :
  - (NBE + NBS) multiplications
  - (2 * (NBS + 1)) divisions
  - (2 * (NBE + NBS)) additions
  - (NBS + 1) substractions.
- The computations of the discard in phase is done in :
  - 1 multiplication
  - 2 divisions
  - 1 addition
  - 1 substraction.
- The computations of the phase vector depends of two variables :
  - the number of measurements NBE,
  - the number of trial frequencies NBF.

  The number of computations is :
  - (NBF * NBE) multiplications
  - (NBF * NBE) substractions.

  as explained before.
- For the phase dispersion, the important variables are :
  - the number of trial frequencies,
  - the number of segments NBS,
  - the number of elements NBE.

  For one segment having NBES elements we have :
  - NBES multiplications
  - NBES divisions

- ((2 * NBES) + 1) additions
- (2 * NBES) substractions.

But, $\sum_{i=1}^{NBS} NBES_i = NBE$

For NBS segments, we have :

- NBE multiplications
- NBE divisions
- (2 * NBE) + NBS) additions
- (2 * NBE) substractions.

And for NBF frequencies, there are :

- (NBF * (NBE + 1)) multiplications
- (NBF * (NBE + 1)) divisions
- (NBF * ((2 * NBE) + NBS) additions
- (NBF * ((2 * NBE) + 1) substractions.

- And NBF additions for the incrementation of the frequency.

If :

- NBF is the number of trial frequencies,
- NBS is the number of segnents,
- NBE is the number of measurements,

the number of computations for the Signal Variation Minimization is resumed in Fig 7.2.

multiplications	divisions	additions	substractions
NBE+NBS 1 NBF*NBE NBF*(NBE+1)	(2*(NBS+1) 2 NBF*(NBE+1)	2*(NBE+NBS) 1 NBF*(2*NBE+NBS) NBF	NBS+1 1 NBF*NBE NBF*(2*NBE)
NBF*(2*NBE+1)+NBE+NBS+1	2*(NBS+2)+NBF*(NBE+1)	2*((NBE*(NBF+1))+NBS)+NBF*(NBS+1)+1	3*NBF*NBE+NBS+2

Fig. 7.2 Number of computations required by the SVM method.

7.1.3 Discrete Fourier Transform

As described in section 3.2.4. this method is divided into two phases of computations :

- the substraction of the mean of the value (this corresponds to the subroutine INITX),
- the computations of the Discrete Fourier Transform and the spectral window for each frequency (this corresponds to the subroutine FTFREQ).
- For the substraction of the mean of the measurements, if we have NBE measurements, we will carry out :
  - 1 division
  - NBE additions
  - NBE substractions.
- For the computations of the Discrete Fourier Transform and the spectral window, we have to know :
  - the number of trial frequencies NBF,
  - the number of observations NBE.

  For one frequency, we will do :

- (3 * (NBE + 2)) multiplications
- 2 divisions
- NBE cosinus
- NBE sinus
- (4 * NBE) additions.

For NBF frequencies, we will do :

- ((3 * NBF) * (NBE + 2)) multiplications
- (NBF * 2) divisions
- (NBF * NBE) cosinus
- (NBF * NBE) sinus
- (4 * (NBF * NBE)) additions.

- And NBF additions for the incrementation of the frequencies.

If :

- NBF is the number of trial frequencies,
- NBE is the number of measurements,

the number of computations is described in Fig 7.3.

mul	div	sin	cos	add	sub
(3*NBF)*(NBE+2)	1 2*NBF	NBF*NBE	NBF*NBE	NBE 4*NBF*NBE NBF	NBE
(3*NBF)*(NBE+2)	2*NBF+1	NBF*NBE	NBF*NBE	NBF*(4*NBE+1)+NBE	NBE

Fig. 7.3 Number of computations required by the DFT method.

7.2 Possible improvments

To have more informations about the execution and the performances, we have used the execution analyser of Toolpack described in section 4.1.5.
This tool divides the program into segments, and after the execution gives the number of passages by each segment. We can see where the improvments have to be done.

In our case, particular improvments would be made in the computation of the phase dispersion from Stellingwerf 's method, with the particular decomposition in bins and covers. Perhaps, accurate searchs would be done to improve this particular phase.
For the two other methods, no particular improvment has to be realized.

Chapter 8

Conclusions

Now that all the package has been described, some conclusions have to be made. We can divide these conclusions into three parts.

- The first concerns other fields of application for this package section 8.1.
- The second one deals with future developments section 8.2.
- And the third makes evident conclusions section 8.3.

8.1 Other fields of application

As this package has been developed for astronomers, and the great majority of the references given corresponds to astronomical papers, one can thing that these methods are good only for astronomers.
But other professions can allow this type of method. With the necessary changings dictated by the required data 's format, this package can be used by other people who have time series with unequal spaced datas.
We can also try to use the package in a way to prove that in a particular set of observations, no period can be found.

Not only the time is concerned in this part. If we have irregularly sampled data depending from other parameters that the time, these methdos can be also used.

8.2 Future developments

- The first development envisaged can be the implementation of the other methods described in the comparative study.
- In a second step, the implementation of accurate tests to help the user in the choice of the best period between the candidates can be realized.
- The computations of the test realized on the period candidates can be separated from the ones of the method in itself.

- An other development could be the research and the implementation of methods which have a beter mathematical justification.
- The last development envisaged could be the extension of the activity to the automatic classification of time series.

8.3 Conclusions

As a certain number of constraints concerning the mathematical contents of the package and the imposed data structure, we have attach ourselves particularly at the designing of an "user-friendly" interface, at the rigorously implementation and at the testing phase.

This package has been realized in collaboration with ESO 's astronomers, and it must be integrated in the MIDAS 's environment. This implies that particular decisions take in the process structuration and in the implementation can be different for other people who wants to implement the chosen method.

On an other side, this document presents only the different steps leading to the elaboration of the package.
It can be completed by different studies concerning :

- the different methods implemented,
- new methods,
- other fields of application,
- a comparaison between these methods on one hand and on the other hand methods for equally spaced data.

Appendix A

Midas

A.1 Hardware

The hardware configuration actually used at ESO and described in section 4.1.1 can be visualized in the Fig A.1 here-under.

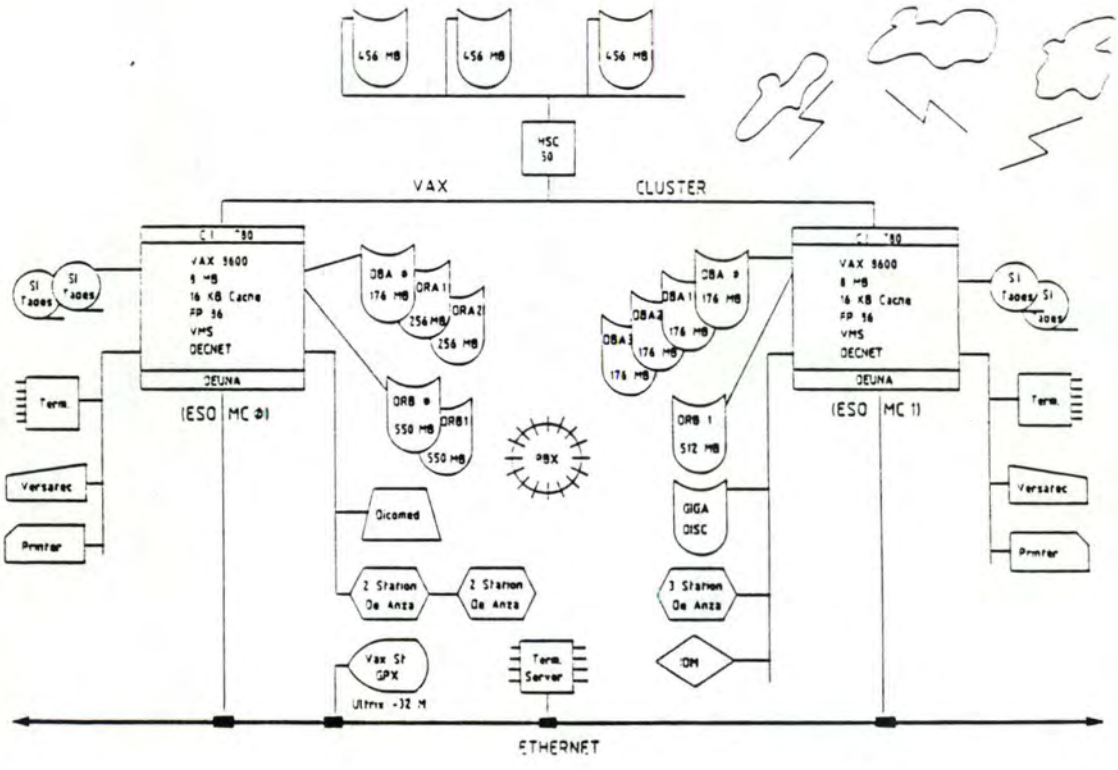


Fig. A.1 The computer configuration currently used at ESO.

The MIDAS 's worksation is typically composed of three different screens and two keyboards. The first keyboard/screen pair is used to enter a MIDAS 's session and to provide the computer the commands to execute. This terminal also received the results of the computations, the on-line help, the mail messages and the error messages .
The second terminal is a graphic display terminal used to plot the graphics, i.e. curves, histograms and so on.
The third terminal is composed of a high resolution color monitor and of a joystick. The display device is used to show the images while the joystick is helpful to interactively point and define zones to process the image.

The hardware also comprises other typical devices such disk- and tape drives, many different type printing devices, i.e. line printers, dot-matrix printer, laser printers, graphic printers and so on; and a device especiallly design to copy the digital images on photographic film.

The system is also equipped with modem to allow communication with the widespread world of astronomy.

A.2 Image

An image is an array of numbers representing data values.
An image can be considered as a set of *pixels*. These pixels are drawn up in lines, each line usually contains the same number of pixels. The number of pixels in the line determine the image size.

Such a structure implies that an image will be accessed as a *matrix*, giving the pixel coordinates of the point to access in the image.

In our particular case, it is not possible to access to the pixels of the output image. The structure of image used is only a manner of producing output. This structure has been chosen because it permits to give for each defined axe the start coordinate and the step.

A.3 Table

A table is a two dimensional array orginazed with *rows* and *columns*.
The item in one row may describe different properties of the same object or feature.

On the other hand, all elements in a given column must be of the same type and be associated by the same property.
An item in a table is accessed by giving its column and row in addition to the table 's name.

The row number can either be given as an absolute value, i.e. the sequence number, or indicated by the value in a previously reference column. Columns are normally referred to

by either user- defined label, but can be addressed by their absolute number. A character string containing the unit of the values is also associated with each column.

Three types of datas can be stored in a table : real value in single or double precision and character strings. If an element is not defined it will be a *NULL* entry.

A.4 Logfile

The logfile corresponding at our particular execution is presented in Fig. A.2

```
MIDAS 001> SET/CONTEXT TSA
TSA Package    Version 1.0 (JAN 1987)

Available Commands:

      SET/TSA
      SHOW/TSA
      TSA/TABLE
      PLOT/TSA
      TUTORIAL/TSA

MIDAS 002> SET/TSA METHOD=PDM
MIDAS 003> TSA/TABLE ? TUCAS :T :X,:S

 Nyquist frequencies:    Minimum     Maximum    Step
 -------------------     0.90E-02    0.20E+00    0.18E-02


 Phase Dispersion Minimisation Method.
 -------------------------------------

 Input table                  Output Images and tables
 TUCAS                        TSAPDM              : .BDF,.TBL

 Freq Min   Freq Max  Sample  Freq Step  Nr of bins Nr of covers Width
 0.902E-02  0.199      105    0.180E-02      5          2          1

MIDAS 004> SET/TSA METHOD=SVM
MIDAS 005> TSA/TABLE ? TUCAS :T :X,:S

 Nyquist frequencies:    Minimum     Maximum    Step
 -------------------     0.90E-02    0.20E+00    0.18E-02
```

```
 Signal Variation Minimisation Method.
 -------------------------------------

 Input table                    Output Images and tables
 TUCAS                          TSASVM              : .BDF,.TBL

 Freq Min   Freq Max  Sample  Freq Step  Rel Error  Width
 0.902E-02  0.199      105    0.180E-02    0.10       1

MIDAS 006> SET/TSA METHOD=DFT
MIDAS 007> TSA/TABLE ? TUCAS :T :X,:S

 Nyquist frequencies:    Minimum     Maximum     Step
 -------------------     0.90E-02    0.20E+00    0.18E-02


 Dicrete Fourier Transform Method.
 ---------------------------------

 Input table                    Output Images and tables
 TUCAS                          TSADFT              : .BDF,.TBL
                                TSADFTSW.BDF        TSADFTDC.BDF

 Freq Min   Freq Max  Sample  Freq Step  Rel Error Width


 0.902E-02  0.199      105    0.180E-02    0.10      1
```

Fig. A.2 Logfile.

Appendix B

User 's manual contribution

The user 's manual presented in this appendix correponds to the document given to the astronomers and containing the specification of the time series analysis project.
As it must be integrated into the MIDAS user 's guide, some details would be changed in the future.

Time Series Analysis.

This chapter deals with the Time Series of unequally spaced data. The period determination techniques are discussed and described in section B.1. Three methods have been implemented and can be applied to MIDAS tables. The testing phase was performed comparing the results of this package and of the previously written software that we received from Manfroid and Boucher and from Stellingwerf.

Section B.2 describes the syntax of the different MIDAS commands, and section B.3 illustrates a typical MIDAS session to analyse the period of some data.

The outputs of the program and their possible interpretation are given in section B.4. An example is presented in section B.5, it may be convenient for the first time users to run the command `TUTORIAL/TSA` while reading this section. Section B.6 contains a summary of the commands.

B.1 Outline of the available methods.

This section contains an introduction to time series analysis and gives a brief description of the implemented methods.

A time series can be represented by a set of pair (x_i, t_i) $i = 1, \ldots, N$ where x is the vector of the measured data and t the time when these measurements were taken. The mean of the sample is

$$\bar{x} = \frac{\sum_{i=1}^{N} x_i}{N},$$

and the variance

$$\sigma^2 = \frac{\sum_{i=1}^{N}(x_i - x)^2}{N-1}.$$

To analyse a time series, the Fourier transform can be thought of importance but some non–parametric statistical methods give good results. The successive steps of the analysis are:

- the definition of a set of trial frequencies ν based on some user knowledge of the data or on the Nyquist criteria.
- the definition of a criteria that will point out the actual candidates for the period. This can be the minimization of a statistic θ on the period range or the detection of peaks in a periodogram.
- from these candidates, the more probable period are to be extracted.

The TSA package is offering complete first and third steps. Referring to a comparative study of the different period determination methods performed by A. Heck, J. Manfroid, and G. Mersch , three methods or criteria have been chosen for the present package with regards to their efficiency and robustness:

1. the phase dispersion minimization method developed by *Stellingwerf*
2. the signal variation minimization method introduced by *Renson* minimazation of a quadratic form.
3. the Fourier analysis based on the discrete Fourier transform investigated by *Deeming*

To know more about the reasons of our choice, the reader is invited to consult the above mentioned papers.

B.1.1 The Phase Dispersion Minimization Method.

The development of this method is based on an article of R. F. Stellingwerf.

For any subset of the measurements, we can define the sample variance s^2, with analogy to σ^2. For any set of M distinct samples, having respectively the variances s_j^2 $j = 1, \ldots, M$ and n_j $j = 1, \ldots, M$ data points, the overall variance is given by:

$$s^2 = \frac{\sum_{j=1}^{M}(n_j - 1)\, s_j^2}{\sum_{j=1}^{M} n_j - M}.$$

Let T be a trial period, the phase vector ϕ is:

$$\phi_i = t_i/T - [t_i/T]$$

where [a] represents the greatest integer in a. Just remark that the ϕ_i $i = 1, \ldots, N$ lie in [0 1].

The method defines the set of samples as a structure of bins and covers in the phase interval. The unit interval is divided into N_b bins of length $1/N_b$, and N_c covers of N_b bins are taken, each cover offset in phase by $1/(N_b N_c)$ from the previous one.

The statistic θ_1 considered here is the overall variance s^2 of the bins and covers divided by the variance of the whole sample:

$$\theta_1 = \frac{s^2}{\sigma^2}.$$

The criterion is based on very simple assumptions.
If the tested period T is not a good one, then $s^2 \approx \sigma^2$ and $\theta_1 \simeq 1$. But, T can be considered as a correct period if θ_1 reaches a local minimum, hopefully with a near zero value.

A significance test is performed on the found minima using the F-distribution. The most probable periods have the smallest value in the F-test. Anyhow to be affected with a correct significance, the F-test performed on a period should not exceed 0.1.

B.1.2 The Signal Variation Minimization Method.

This method was developed in an article by P. Renson.

For a given period T, the phase of the k-th observation is:

$$\phi_i = t_i/T - [t_i/T]$$

where $[a]$ is the greatest integer in a.

Lafler and Kinman introduced the following statistic:

$$\sum_i (x_i - x_{i-1})^2$$

If the tested period is good, the differences $x_i - x_{i-1}$ become small in mean. Nevertheless, it happens that the difference $x_i - x_{i-1}$ is important just because they correspond to very different phases. A correction has to be introduced.

Therefore, a following statistic is defined:

$$\sum [(x_i - x_{i-1})^2/(\phi_i - \phi_{i-1})]$$

But this formula is also subject to criticism. If different measurements x_i are corresponding to two proximal phases then even if they only differ by a quantity of the order of the noise, the ratio $(x_i - x_{i-1})^2/(\phi_i - \phi_{i-1})$ will contribute too much to the sum as $(\phi_i - \phi_{i-1})$ is too small. The tested frequency could then be abnormally rejected. To solve this problem, one

can simply add to the denominator a term ε equal to the discard in phase corresponding to the noise. If the error mean is e and the total variation of the function is r, for a more or less sinusoidal curve of period 1, ε can be estimated:

$$\varepsilon = \frac{e}{2r}.$$

The statistic θ_2 proposed by Renson is a weighted sum of the squares of the successive differences, i.e.

$$\theta_2 = \sum_{i=1}^{N} \frac{(x_i - x_{(i-1)})^2}{(\phi_i - \phi_{(i-1)} + v + \varepsilon)}$$

where

- $v = \begin{cases} 1 & \text{if i=1(and(i-1)=N)} \\ 0 & \text{else} \end{cases}$
- $\varepsilon = e/2r$ where
  1. e is the precision in magnitude of the datas;
  2. $r = \max_i\ x_i - \min_i\ x_i$.

The periods minimising the value of the statistic are good candidates.

B.1.3 The Fourier Analysis.

This method is based on the paper of T. J. Deeming. and uses the discrete Fourier transform.

The complex Fourier transform $F(\nu)$ of a function $f(t)$ is:

$$F(\nu) = \int_{-\infty}^{+\infty} f(t) e^{i2\pi\nu t} dt.$$

and the discrete Fourier transform is the discrete version of it:

$$F_N(\nu) = \sum_{k=1}^{N} f(t_k) e^{i2\pi\nu t_k}.$$

In the case of determinate processes, the discrete Fourier transform is the *convolution* of the continuous Fourier transform $F(\nu)$ with a function $\delta_N(\nu)$ called the spectral window that is only a consequence of the discretization of the time space:

$$F_N(\nu) = F(\nu) * \delta_N(\nu) \simeq \int_{-\infty}^{+\infty} F(\nu - \nu')\delta_N(\nu')d\nu'$$

with

$$\delta_N(\nu) = \sum_{k=1}^{N} e^{i2\pi\nu t_k}.$$

In practical computations, it is most convenient to use the function $N^{-1}F_N(\nu)$ and the corresponding spectral window $\gamma_N(\nu) = N^{-1}\delta_N(\nu)$. Thus

$$N^{-1}F_N(\nu) = \gamma_N(\nu) * F(\nu).$$

On the computed discrete Fourier transform, a non–linear deconvolution is performed according to the spectral window, using the CLEAN algorithm. An approximation of the actual Fourier transform is then obtained. Localisation and amplitude of the peaks are detected.

B.2 Commands in the TSA.

This section gives a description of the different commands developed in the package i.e.

- `SET/TSA`
- `TSA/TABLE`
- `SHOW/TSA`

B.2.1 SET/TSA.

It sets up the different qualifiers for the next execution of the actual analysis. The syntax is:
`SET/TSA` [METHOD=method_name] [PRINT=nr_prints] [WIDTH=width] [OUTPUT= output].
Where:

1. *method name* is one of the following:
   (a) PDM: Phase Dispersion Minimization (defaulted),
   (b) SVM: Signal Variation Minimization,
   (c) DFT: Discrete Fourier Transform.
2. *nr_print* is an integer, the most probable nr_print periods are displayed at the end of the treatment, this is defaulted to 0.
3. width is half the minimal distance (number of trial frequencies) between two successive extrema, its default value is 1.
4. *output* is a prefix for the name of the default output; if no output is given in the `TSA/TABLE` command, the results will be stored in images and tables the name of which are built with this prefix followed by the name of the method. Default value of the prefix is TSA.

Examples:
`SET/TSA` METHOD=PDM PRINT=1 WIDTH=2 OUT
`SET/TSA` DFT 2

B.2.2 SHOW/TSA.

This causes the display of the actual value of the different qualifiers of the TSA package.
The calling sequence is:
`SHOW/TSA` .

B.2.3 TSA/TABLE.

It performs the actual analysis of the time series.
The syntax is:
`TSA/TABLE` [method_par] INTAB [time [data[,segment]]] [OUTPUT]
where

- the method parameters are *fmin,fmax,ndis[,nbin,ncov]* if the method is PDM and *fmin,fmax,ndis[,error]* otherwise. *fmin* and *fmax* fix the investigated frequency interval and *ndis* is the number of trial frequencies in this interval. If they are omitted or set to 0, they are computed to satisfy the Nyquist criteria.
  nbin and *ncov* are, for the PDM, the number of bins and covers of the phase unit interval, if omitted or null they are set to $(5, 2)$.
  error is the noise to signal ratio with $0 \leq error \leq 1$. The default value is 0.1
- *INTAB* is the name of the input table that contains the time series.
- *time* is the reference to the column of *INTAB* containing the time vector. The default value is *:TIME*.
- *data* is the reference to the data column in *INTAB*, it is defaulted to *:DATA*.
- *segment* is the reference to the segment column. Measurements can be separated in different segments or sets. If not given, the existence of only one segment is assumed. This column contains the value of the segment corresponding to the time and the data. When the value change, the time and the data are consedering in an other segment.
- OUTPUT is the name under which will be stored the results of the program. As different outputs are available, they will differ by their extension. For each method, an image named *OUTPUT*.BDF contains the value of the objective function (the statistic or periodogram) value at each point of the discretisation and a table named *OUTPUT*.TBL is an ordered list of the period candidates. The first column of the latter is :FREQ and contains the frequencies where the extrema occurred, the second one named :FUNC is the value of the objective function. The third one is either the F-test (:FTEST) either the amplitude of the peak (:AMPLI) and is used to appreciate the validity of results. Moreover, for DFT, two other images are produced: the first named *OUTPUT*.SWI is the spectral window and the second *OUTPUT*.DCV is the deconvolved spectrum. The default value of output is TSA followed by the method name, for instance TSAPDM.

PDM examples:
`TSA/TABLE` 1.05,2.05,65,5,2 MYTABLE :TIME :MEASURE,:SEGM OUTPDM
`TSA/TABLE` ? MYTABLE #1 #2
SVM examples:
`TSA/TABLE` 0.125,0,93,0.05 MYTABLE :TIME :MEASURE
`TSA/TABLE` 0,0,0,0 MYTABLE #1 #2 OUTSVM
DFT examples:
`TSA/TABLE` 0,0.067,0,0.05 MYTABLE :TIME :MESURE OUTDFT
`TSA/TABLE` ? MYTABLE .

B.3 The Process

This section illustrates the typical sequence of commands for the use of the TSA process.

The qualifiers must first be chosen: the method, the number of searched period and so on. The actual analysis can then be performed with the right parameters. Several execution can be asked if results are not satisfying. The results can be viewed on a graphics terminal.

This typical sequence of MIDAS command would be:

1. `SET/TSA` [METHOD=method_name] [PRINT=nr_prints] [WIDTH=width] [OUTPUT=output]
2. `TSA/TABLE` [method par] INTAB [time] [data[,segment]] [OUTPUT]

The command `SHOW/TSA` can be called at each moment of the sequence.

B.4 Interpretation of Outputs.

Interpreting the results obtained by the different period determination techniques is not an easy work and some help is given below.

B.4.1 The PDM method.

The outputs from the PDM method are:

- an image with the value of the statistic at each point of the discretization,
- a table with the frequencies where the statistic is minimal, the value of that minimum and the corresponding *F-TEST*.

As defined, the statistic θ_1 is the ratio of the overall variance of the bins and covers and the variance of all the observations. Therefore, θ_1 is the quotient of two sums of squares; its probability distribution is thus a F distribution with respectively $\sum n_j - M$ and $N - 1$ degrees of freedom.
As it is more convenient to make a test on the F distribution with a condition written with a $>$ operator, we are defining our F function as

$$F \equiv 1/\theta_1$$

The probability $p_1$that a given value of θ_1 is due to random fluctuations is twice the area of the F distribution above θ_1^{-1} (two sided test). This probability approaches unity as $\theta_1 \to 1$. Thus the significance is p_1 satisfying the following condition:

$$F_{p_1/2,N_{1f},N_{2f}} = 1/\theta_1$$

where $N_{1f} = N - 1$ and $N_{2f} = \sum n_j - M$. p_1 is obtained by reference to a F table or by using an approximation of the inverse function to $P(F, N_{1f}, N_{2f})$. In this particular implementation, the approximation of Abramowitz and Segun , was chosen.

B.4.2 The SVM method.

The outputs from the SVM method are:

- an image with the value of the statistic at each point of the discretization.
- a table with the frequencies where the statistic is minimal and the corresponding minimum value.

B.4.3 The DFT method.

The corresponding outputs are:

- an image with the discrete Fourier transform at each point of the discretization (periodogram)
- an image with the spectral window,
- a third image with the deconvolved spectrum,
- a table with the frequencies where the deconvolved spectrum is maximal and the corresponding normalised value of the maximum,

B.5 The Tutorial.

An example of the use of the package is provided in this section.

A tutorial procedure (`TUTORIAL/TSA`) shows how to use the TSA package. This example works on a table (TSATUT.TBL) of three columns:

1. the time, labeled by :T,
2. the data, labeled by :X,
3. the segment, labeled by :S.

The different steps of this execution are :

1. copying the given table into your directory,
2. selecting the PDM method, allowing to print the 4 best minima and setting the width for the minima to 2 by issuing:
   `SET/TSA` METHOD=PDM PRINT=4 WIDTH=2
3. executing the PDM algorithm with
   - fmin = 0.1,
   - fmax = 0.3,
   - ndis = 20,
   - nbin = 5,
   - ncov = 2,
   - INTAB = TSATUT,
   - coltim = :T,
   - colmes = :X,
   - colseg = :S,
   - OUTPUT = TUTPDM,

   by issuing the command:
   `TSA/TABLE` 0.1,0.3,20,5,2 TSATUT :T :X,:S TUTPDM
4. showing the value of the different parameters, MIDAS instruction : `SHOW/TSA`
5. modifying the selection of the method to use the DFT:
   `SET/TSA` METHOD=DFT
6. executing the DFT algorithm:
   `TSA/TABLE` 0,0,0 TSATUT :T :X,:S TUTDFT
7. selecting the SVM method
   `SET/TSA` METHOD=SVM

8. executing the analysis through SVM:
   `TSA/TABLE` 0.2,0.5,0,0 TSATUT :T :X,:S TUTSVM

9. deleting all the files created for this execution.

B.6 The Command Summary.

This section gives a summarize of the implemented commands.

The table B.1 summarises the commands which are implemented in the context of time series analysis.

```
SET/TSA  [METHOD=method_name] [PRINT=nr_prints] [WIDTH=width] [OUTPUT=output]
SHOW/TSA
TSA/TABLE [method_par] INTAB [time [data[,segment]]] [OUTPUT]
```

Table B.1: TSA Commands

Appendix C

Toolpack

C.1 Example

In this part more details of the using of the different tools of Toolpack described in section 4.1 will be given, and the using ' s sequence will be illustrated.

- First, the program has to be scanned with the lexical analyser : ISTLX.
  To use this tool, no pre-requisite is needed.
  The arguments are :
  - the name of the input source file,
  - the name of the output list file,
  - the name of the output token stream,
  - the name of the output comment stream.

  All errors are reported to the error file.
  Errors are as follow :
  - token too long,
  - error in token,
  - error in token to be screened,
  - unprocessed text remaining in token to be screened,
  - screen ended in error,
  - scan ended in error,
  - screened token ends unexpectadly,
  - buffer overflow.

  Fatal errors are reported separately.

- After the scan, the program can be parse using the parser : ISTLY.
  The pre-requisite for this tool is the execution of ISTLX described before.
  The arguments are :
  - the name of the input token stream,
  - the name of the input comment stream,
  - the name of the output parse tree,
  - the name of the output symbol table,
  - the name of the output comment index.
- The conformance of the program to the Fortran-77 standard can be checked, ISTSA.
  The pre-requisites are ISTLX and ISTYP already described.
  The arguments are :
  - the name of the input parse tree,
  - the name of the input symbol table,
  - the name of the output parse tree,
  - the name of the output symbol table,
  - the name of the output attribute file.

  The output parse tree and symbol table are compatible with the parse tree and symbol table produced by ISTYP, and may be processed by any tool wich takes an ISTYP-produced parse tree and/or symbol table as input.

- We can now standardize the declarative parts of the program unit, by using the declaration standardizer ISTDS.
  The pre-requisites are ISTLX and ISTYP.
  The arguments are :
  - the name of the input parse tree,
  - the name of the input symbol table,
  - the name of the input comment stream,
  - the name of the output token stream,
  - the name of the output comment stream,
  - option string (not required).
- Now that the program is complete, we can polish it with the pretty printing ISTPL.
  The pre-requisite is ISTLX.
  The arguments are :
  - the name of the input token stream,

- the name of the input comment stream,
- the name of the polished output file,
- the name of the option file.

This option file is created by the tool ISTPO which is the polish option file editor. It has no pre-requisite and as argument has only the name of the output option file.[1]

- We can use the different tools allowing to view the intermediate objects created by the tools.
  These are :
  - the attribute viewer ISTVA,
  - the view symbol table ISTVS,
  - the view parse tree ISTVT,
  - the view warnings ITSVW.

  To view the attribute, ISTLX, ISTYP and ISTSA must be executed.
  The arguments are :
  - the name of the input symbol table,
  - the name of the input attribute file,
  - the name of the output listing file,
  - the header text of the listing (not required).

  The view symbol table requires the execution of ISTLX and ISTYP.
  The arguments are :
  - the name of the input symbol table,
  - the name of the output listing file,
  - the header text of the listing (not required).

  To use the view parse tree ISTLX and ISTYP have to be first executed.
  The arguments are :
  - the name of the input parse tree,
  - the name of the input symbol table.

  And for the view warnings, the pre-requisites are ISTLX and ISTYP.
  The arguments are :

[1] more details about these options can be found in [Nag]

- the name of the input parse tree,
- the name of the input symbol table.

- We can now use the execution analyser ISTAN.
  This tool necessites the execution of ISTLX before its own execution.
  The arguments are :
  - the name of the input token stream,
  - the name of the input comment stream,
  - the name of the output instrumented file,
  - the name of the output statement summary file,
  - the name of the output annotated comment stream,
  - the name of the output summary report file,
  - the option string (not required).

  ISTAN produces another program from the user 's program, wich has the same effect but has some instrumentation code added to it.
  This instrumentation code may :
  - perform execution frequency couting of code segments,
  - count assertion failures (assertions being special comments containing Frotran logical expressions, which the programmer expects to be true at that point),
  - trace analysis, this may be :
    * execution of particular code segments,
    * control flow following execution of particular code segments, or
    * control flow prior to execution of particular code segments,
  - update history files containing cumulative frequency counts over a number of program execution.

  Having processed their program using ISTAN, the user must then compile the instrumental code produced by ISTAN to obtain an instrumented versioon of their executable program.

  This instrumented program may then be executed to produce the data required by the user.

- Now that the program is ended, we can used the version controller ISTVC to maintain a number of separate versions of the program.
  There is no pre-requisite.
  The arguments are :
  - option,
  - the name of the version file,
  - the name of the text file (not required).

Appendix D

Listings

Two different kinds of listings are available :

1. the Fortran codes of the different programs,
2. the subroutine 's programs associated with the programs listed before. These subroutines are written in the MIDAS 's command language.

The following listings will take place in the next pages :

- TSA.CTX associated with the new MIDAS 's command SET/CONTEX TSA;
- SETTSA.PRG associated with the new MIDAS 's command SET/TSA;
- TSAMAIN.FOR, TSAUTIL.FOR,
  PDMFUNCT.FOR, PDMFTEST.FOR, PDMIO.FOR,
  SVMFUNCT.FOR, SVMIO.FOR,
  DFTDMING.FOR, DFTTRANS.FOR, DFTUTIL.FOR,
  DFTCLEAN.FOR, DFTIO.FOR
  and TSATAB.PRG associated with the new MIDAS 's command TSA/TABLE;
- SHOW.PRG associated with the new MIDAS 's command SHOW/TSA;
- TSAHELP.HLP associated with the new MIDAS 's command HELP/TSA,
- TSATUT.PRG associated with the new MIDAS 's command TUTORIAL/TSA.

As this can take a certain place in this document, all the listings mentionned abouve won't be added, however they are available on simple request.

Bibliography

Astronomical and mathematical references.

[Abramowitz-Segun] M. Abramowitz and I.A. Segun.
Handbook of mathematical functions with formulas, graphs and mathematical tables.
Dover Publications, Inc, New York 1968.

[Ansley-Kohn-83] C.F. Ansley and R. Kohn.
On estimations of arima models with missing values.
In *Time Series Analysis of Irregularly Observed Data, pages 9-37, Texas A & M, 1983.*

[Ansley-Kohn-84] C.F. Ansley and R. Kohn.
New algorithmic procedure for estimation problems in time series.
In *COMPSTAT Proceedings in Computational Statistics, pages 23-24, Physica-Verlag. Wien, 1984.*

[Barbieri-Romano-di Serego-Zambon] C. Barbieri, G. Romano, S. di Serego A., and M. Zambon.
Survey of the optical variability of compact extragalactic objects.
Astron. and Astrophys., 59:419-426, 1977.

[Biraud] Y.G. Biraud.
Some simple comments on the method used in the short period determination of variable stars. In *Proceedings of the Workshop on Pulsating B Stars, pages 297-298, Observatoire de Nice, 1981.*

[Bloomfield] P.Bloomfield.
Spectral analysis with randomly missing observations.
J. Roy. Statis. Soc., 32:369-380, 1970.

[Box-Jenkins] G.E.P. Box and G.M. Jenkins.
Time Series Analysis, Forecasting and Control.
Holen-Day, 1976.

[Brigham] E. Oran Brigham.
The Fast Fourier Transform.
by prentice Hall, Inc. Englewood Clifs, New Jersey, 1974.

[Burki-Maeder-Rufener] G. Burki, A. Maeder, and F. Rufener.
Variable stars of small amplitude iii. semi-period of variation for seven b2 to g0 supergiant stars.
Astron. Astrophys., 65:363-367, 1978.

[Burki-Rufener] G. Burki and F. Rufener.
Variable stars of small amplitude vi. the case of hd 13970.
Astron. Astrophys., 70:105-110, 1978.

[Clinger-Van Ness] W. Clinger and J.W. Van Ness.
On unequally spaced time points in time series.
Ann. Statist., 4:736-745, 1976.

[Perez de la Blanca-Garrido] N. Perez de la Blanca and R. Garrido.
Period determination techniques.
In *Proceedings of the Workshop on Pulsating B Stars, pages 285-296, Observatoire de Nice, 1981.*

[Deeming] T.J. Deeming.
Fourier analysis with unequally spaced data. *Astron. Astrophys. Suppl. Ser., 36:137-158, 1975.*

[Ferraz-Mello] S. Ferraz-Mello.
Estimation of periods from unequally spaced observations.
The Astrophys. J., 86:619-624, 1981.

[Groth] E.J. Groth.
Probability distributions related to power spectra.
The Astrophys. J. Suppl. Ser., 29:285-302, 1975.

[Harvey-McKensie] A.C. Harvey and C.R. McKensie.
Missing observations in dynamic econometric models : a partial synthesis.
In *Time Series Analysis of Irregularly Observed Data, pages 134-175, Texas A & M, 1983*

[Heck-Manfroid-Mersch] A. Heck, J. Manfroid and G. Mersch.
On period determination methods.
Astron. Astrophys. Suppl. Ser., 59:63-72, 1985.

[Horne-Baliunas] J.H. Horne and S.L. Balinuas.
A prescription for period analysis of evenly sampled time series.
The Astrophys. J., 302:757-763, 1986.

[Jessner] A. Jessner.
Efficient period determination for irregularly sampled data.
Inauguraldissertation zur Erlangung der Doktorwürde der Mathematisch Naturwissenschaftlichen.
Rheinischen Friedrich Wlihelms Universität zu Bonn, 1985.

[Jones] R.H. Jones.
Fitting multivariate models to unequally spaced data.
In *Time Series Analysis of Irregularly Observed Data, pages 158-188, Texas A & M, 1983*

[Kay-Marple] S.M. Kay and S.L. Marple.
Spectrum analysis : a modern perspective. *Proc. IEEE, 69:1380-1419, 1981.*

[Kendall-Stuart] M.Kendall and A. Stuart.
The Advanced Theory of Statistics.
Volume 3, Ch. Griffin & Co Lim., 1976.

[Kitagawa] G. Kitagawa.
State space modelling of nonstationary time series and smoothing of unequally spaced data.
In *Time Series Analysis of Irregularly Observed Data, pages 189-210, Texas A & M, 1983*

[Koopmans] L.H. Koopmans.
The Spectral Analysis of Time Series.
Academic Press, 1974.

[Krzysko-Smoczynski] M. Krzysko anf Smoczynski.
Parameter estimation and order determination of multivariate autoregressive process.
In *COMPSTAT Proceedings in Computational Statistics, page 35-40, Physica- Verlag. Wien, 1984.*

[Lafler-Kinman] J. Lafler and T.D. Kinman.
An rr lyrae star survey with 20-inch astograph. ii. the calculation of rr lyrae periods by electronic computer.
The Astrophys. J. Suppl. Ser., 11:216-222, 1965.

[Manfroid-Heck-Mersch] J. Manfroid, A. Heck and G. Mersch.
Comparative study of period determination methods.
In *Proceedings of the Statistical Methods In Astronomy Symposium, pages 37-42, ESA SP-201, 1983.*

[Marquardt-Acuff] D.W. Marquardt and S.K. Acuff.
Direct quadratic spectrum estimation with irregularly spaced data.
In *Time Series Analysis of Irregularly Observed Data, pages 211-223, Texas A & M, 1983*

[Mélard] G. Mélard.
On fast algorithm for several problems in time series models.
In *COMPSTAT Proceedings in Computational Statistics, pages 41-45, Physica-Verlag. Wien, 1984.*

[Miller-Ferreiro] R.B. Miller and O.M. Ferreiro.
Multiple series analysis on irregularly spaced data.
In *Time Series Analysis of Irregularly Observed Data, pages 276-289, Texas A & M, 1983*

[Nobile-Roberto] A. Nobile and V. Roberto.
MFFT : A Package for Two- and Three-dimensional Vectorized Discrete Fourier Transforms.
Technical Report, ISAS-International School for Advanced Studies, 1986.

[Parzen] E. Parzen, editor.
Time Series Analysis for Irregularly Observed Data *Texas A & M, Springer-Verlag, College Station. Texas, 1983.*

[Pelt] J. Pelt.
Phase dispersion minimization methods for estimation of periods from unequally spaced sequences.
In *Proceedings of the Statistical Methods In Astronomy Symposium, pages 37-42, ESA SP-201, 1983.*

[Subba-Rao-Gabr] T. Subba Rao and M.M. Gabr.
An Introductioin to Bispectral Analysis and Bilinear Time Series Models.
Springer-Verlag, 1984.

[Renson] P. Renson.
Méthode de recherche des périodes des étoiles variables.
Astron. Astrophys., 63:125-129, 1978.

[Scargle-81] J.D. Scargle.
Studies in statistical time series analysis. i. random process in the time domain.
The Astrophys. J. Suppl. Ser., 45:1-71, 1981.

[Scargle-82] J.D. Scargle.
Studies in statistical time series analysis. ii. statistical aspects of sepctral analysis of unevenly spaced data. *The Astrophys. J., 263:835-853, 1982.*

[Schafer-Mersereau-Richards] R.W. Schafer, R.M. Mersereau, and M.A. Richards.
Constrained iterative restoration algorithms.
Proc. IEEE, 69:432-450, 1981.

[Schwarz] U.J. Schwarz.
Mathematical statistical description of the Iterative Beam Removing Technique

(Method CLEAN).
Astron. Astrophys., 65:345-356, 1978.

[Shunway] R.H. Shunway.
Some applications of the em algorithm to analysing incomplete time series. In *Time Series Analysis of Irregularly Observed Data, pages 290-334, Texas A & M, 1983*

[Stellingwerf] R.F. Stellingwerf.
Period determination using phase dispersion minimization.
The Astrophys. J., 224:953-960, 1978.

[Tong] H. Tong.
Threshold Models in Non-linear Time Series Analysis.
Springer-Verlag, 1983.

[Turlot] J.C. Turlot.
Some remarks on second stationary process (or weakly stationarity).
In *Proceedings of the Statistical Methods In Astronomy Symposium, pages 245-250, ESA SP-201, 1983.*

Methods in Computer Science references.

[Bodart-Pigneur] F. Bodart, Y. Pigneur.
Conception assistée des applications informatiques
1. études d'opportunité et analyse conceptuelle.
Masson Presses Universitaires de Namur, 1983.

[Van Lamsweerde] A. Van Lamsweerde.
Méthodologie de éveloppements de logiciels.
Facultés Universitaires de Namur, 1985.

User 's manuals.

[Banse-Crane-Ounnas-Ponz] K. Banse, P. Crane, C. Ounnas and D. Ponz.
MIDAS ESO 's interactive image processing system based on VAX/VMS.
Proccedibgs of the Digital Equipment Compuer Users Society, 1983.

[Cowel-Hague-Iles] W.R. Cowel, S.J. Hague, R.M.J. Iles.
Toolpack 1 Release 2.
Introductory Guide.
Nag and Argonne Nationnal Laboratory, 1986.

[DEC-for] Digital Equipment Corporation.
Vax-11 Fortran.
Language Reference Manual.
Digital Equipment Corporation, version 3.0., 1982.

[DEC-fug] Digital Equipment Corporation.
Vax-11 Fortran.
User 's guide.
Digital Equipment Corporation, 1982.

[DEC-vms] Digital Equipment Corporation.
Vax/VMS Primer.
Digital Equipment Corporation, version 3.0., 1982.

[IPG-me] Image Processing Group.
The MIDAS 's environment.
Version 3.7., May 1986.

[IPG-ug] Image Processing Group.
MIDAS 's users guide.
Version 4.2., July 1986.

[Lamport] L. Lamport.
A document preparation system LaTeX.
User 's guide & Reference Manual.
Digital Equipment Corporation Addison-Wesley Publishing Company, Inc., 1986.

[Nag] Nag.
Toolpack user 's guides. Version 2.1.
Nag, 1986.

[ST-ECF] ST-ECF.
Computer System Operations manual.
Vol. I.
Computer System User 's Guide.
ST-ECF, December 1985.