THESIS REPORT

Master's Degree

Analysis of Laser Induced Fluorescence Spectra

by P.J.B. Rinaudo

ANALYSIS OF LASER INDUCED FLUORESCENCE SPECTRA

by

Philippe Jean Bernard Rinaudo

Thesis submitted to the Faculty of the Graduate School
of The University of Maryland in partial fulfillment
of the requirements for the degree of
Master of Science
1987

Table Of Contents

List of Tables

page

List of Figures

page

Chapter 1

Introduction

1.1 Presentation

At present, biotechnology is a very fast developing activity, and the continuous development of new laboratory techniques promises many innovations in the future of this industry . Besides these technological changes, an increasingly competitive environment will question the economic viability of old processes. A superior quality will be required to face the competition from abroad. To face these two challenges, innovation and quality, the biotechnological industry needs good on-line sensing. The measurement of important parameters, such as biomass concentration or metabolic activity, is essential to optimize, control, and scale up biotechnological processes.

1.2 The need for an on-line sensor

In spite of some early computer control applications developed in the 1960's, computers are still used less in biotechnology than in other compa-

rable industries. In many fermenters, the control is often limited to pH, pressure and temperature regulation loops. This relative underdevelopment is due to the lack of on-line measurements. In biotechnology, one deals with very complex molecules and living organisms and there is no way to access the important characteristics of a process with simple measurements like temperature, pH, viscosity, etc.

Armiger and Humphrey wrote in 1979 :" One of the major problems in developing mathematical models and control strategies, lies in the inability to measure on-line many of the important process parameters. Significant improvement in existing sensors and the development of new sensors is needed." No solution has been found yet and the challenge of on-line measurements is now a crucial one for industry. To face this challenge, optical techniques seem to be a powerful tool.

1.3 Optical techniques

The expression "optical techniques" covers a wide variety of techniques that chemists and biologists have been using for concentration or identification measurements for years. Absorption, scattering and fluorescence are some examples of them.

With respect to other measurement techniques like viscometry, pH, calorimetry, etc., optical techniques have some particular advantages which seem to make them well-suited for on-line fermentation measurements. In-

deed light measurements are :

- ⋆ Fast : many optical measurements can be carried out in a time frame of a second and often much less.
- ⋆ Sensitive : the sensitivity of light measurement is impressive, sometimes even a single photon can be detected.
- ⋆ Non-invasive : since the interaction with the system is limited to light, no changes are induced by the measurement.

The recent progresses in fiber optics, lasers and photodetectors have furthermore increased their domain of use as well as their convenience. Many optical measurements have the following advantages :

- ⋆ Implantation : a sterilizable fiber optic probe can be placed in a fermenter very easily.
- ⋆ Multiplexing : measurements in different positions of the fermenter can be obtained with a single light source and a single detector by multiplexing.

Among all the optical techniques available, fluorescence is the most sensitive and selective for molecular measurements. In biology, many compounds like proteins, nucleic acids, etc., fluoresce, and therefore fluorescence appears to be a most promising measurement technique for on-line fermentation monitoring.

1.4 Laser induced fluorescence

Since fluorescence was described first by G.G. Stokes in 1852, it has been widely used in chemistry as well as in biology. Although fluorescence requires more sensitive and more expensive setups than absorption, it is now a popular technique due to its sensitivity and selectivity. The power of fluorescence also lies in the wide variety of measurements one can make with a fluorescence setup. Changing the excitation and detection wavelengths, measuring the fluorescence decay in time or the phase, give many data from which the interesting parameters can be estimated. The use of a laser, as a light source, enhances the properties of fluorescence. The high intensity as well as the narrow bandwidth of the laser light increases the quality of the measurements. Despite all those advantages, fluorescence is not widely used in industry. Why ?

The main reason is surely the great complexity of the information given by molecular fluorescence techniques. Indeed, interactions between components and sensitivity to environmental conditions, has raised some doubts about the applicability of fluorescence to the analysis of complex mixtures. But the lack of linearity and selectivity should not eliminate the hope to get interesting information from fluorescence measurements.

1.5 The need for data processing

Today, with the use of computers in data processing, it is conceivable that valid information is extracted from very non-linear information. The

requirements of linearity and specificity that a sensor traditionally had to meet are not needed any more. Sensor linearity should be replaced by monotonicity. Selectivity is not required but just selective sensitivity which means that the parameter to be estimated should influence the set of sensors in a specific way. Therefore, one should be able to find a parameter through the processing of data coming from different sensors.

This is the present evolution of sensing systems where the sensor is not any more a device giving a signal proportional to a physical or chemical parameter. It is rather a system using a great deal of data and some knowledge about the sensing process or about the system behavior in order to give information about complex systems. This approach, called smart sensing, is the one that will give the capability to fully use fluorescence measurements. The realization of a smart sensing system requires interdisciplinary knowledge. Knowledge in spectroscopy, biotechnology as well as in data processing should be combined in the study of a fluorescence sensor for fermentation monitoring. For this reason, the project described in this thesis was developed in collaboration with the National Bureau of Standards and the Biotechnology Research Center of Lehigh University.

1.6 Purpose of the study

This study is a first step towards the development of smart biosensors. However, this goal is very far from being achieved. In this first step, in-

formation is collected about the fluorescence measurement in order to try to resolve the simple mixture analysis and describe the interaction between components.

After a brief presentation of fluorescence theory in the second chapter, the experimental setup is described in the third chapter. The reproducibility of the measurements is studied in Chapter 3 as well as the mathematical techniques for smoothing data. Chapter 4 presents the program developed during this study for the spectrum analysis. Models of fluorescence emission are presented in Chapter 5 as well as deconvolution techniques. Applications of these techniques to multiple components mixtures are then given in Chapter 6.

Chapter 2

Fluorescence

2.1 Theory

Fluorescence is a reemission of light after a luminous excitation. It is usually described within the wider theoretical frame of photoluminescence. Photoluminescence is the emission of light subsequent to the absorption of luminous radiation by molecules. Depending on the kind of excited states involved, the time frame of emission can be very short ($10^{-11} - 10^{-7}$ s) and this is called fluorescence, or much longer ($10^{-5} - 10$ s) and it is called phosphorescence.

Fluorescence is directly related to the transfer between electronic states of the molecule. The main phenomena involved in these electronic transitions are described below and represented in Figure 2.2 (on page 11).

a) Absorption

In the absorption process, a molecule is hit by electromagnetic radiation, and can access an excited electronic state by absorbing a photon. This

transition is possible only if the difference of energy between the ground state and the excited state is equal to the energy of the photon hc/λ, where λ is the wavelength and $hc = 1.9865 \cdot 10^{-25}$ $J.m$ is the product of the speed of light and Planck's constant.

Often, many vibrational levels of the excited state can be reached, and since the vibrational functions strongly overlap (specially in liquids), the absorption spectrum looks more like a band than sharp lines. A typical molecular absorption spectrum at room temperature shows peaks of 10 to 50 nm width as shown on Figure 2.1.

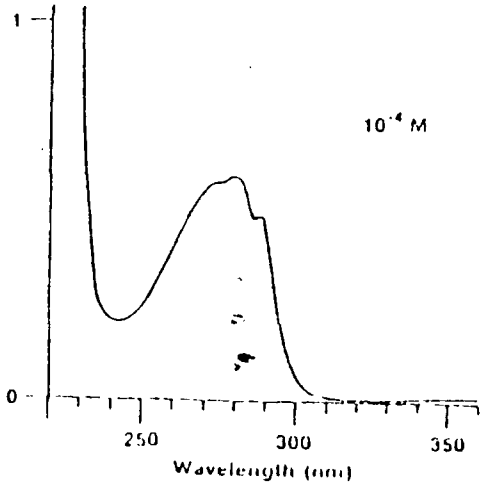


Fig.2.1. Absorption spectrum of tryptophan $10^{-4}M$.

From an excited state, a molecule can evolve in different ways, one of which results in fluorescence. To fully understand the processes involved in luminescence, one should distinguish two kinds of electronic states : the singlet and the triplet. The difference between the two comes from the repartition of spins within the molecule, which is measured by the multi-

plicity. Multiplicity is defined as $2S + 1$ where S is the total spin of the molecule. Usually all the electrons are paired with opposed spins. Therefore the multiplicity is 1, and the molecule is in a singlet state. Sometimes the molecule can transfer to a state where two electrons have the same spin and the multiplicity is 3, giving a triplet state.

One can assume that all molecules are in the lowest vibrational level of the ground state in a solution at room temperature, and that the ground state is a singlet state. The actual time required for photon absorption, i.e., the time required for a molecule to go from one electronic state to another, is 10^{-15} second which is short relative to the time required for all other electronic processes and for nuclear motion. This means that immediately after excitation a molecule has the same geometry and is in the same environment as it was in the ground state. In this situation it can either emit a photon from the same vibrational level to which it was excited initially, or undergo changes in vibrational level prior to emission radiation. Which of these two processes is dominant, depends upon the environment of the molecule. The direct reemission gives Rayleigh and Raman scattering.

b) Scattering

In Rayleigh scattering, a photon is reemitted within 10^{-15} second at the same wavelength as excitation. The intensity of this effect varies inversely with the fourth power of the wavelength. The Raman effect is another form of scattering emission, but in this case vibrational energy may be added or

subtracted from this photon depending upon the characteristic frequency of the electronic state. When reemission occurs (within 10^{-15} second) the photon can contain more or less energy. The energy difference is characteristic of a given molecular structure and is independent of the wavelength of the exciting light. As discussed later, the Rayleigh and Raman emissions of the solvent can be a source of a problem in the spectra analysis.

c) Vibrational relaxation

The excited molecule can also change its vibrational level by thermal relaxation. This relaxation happens in solution, where the solute transfers all its excess of vibrational energy to the solvent in 10^{-13} to 10^{-11} second. Once it arrives at the lowest vibrational state, the molecule can either emit a photon or loose its energy into heat by a process called internal conversion. The photon emission occurs then at the lowest vibrational level of the singlet excited state.

d) Fluorescence

The lifetime of a singlet excited state is 10^{-9} to 10^{-7} second and therefore this is the lifetime of fluorescence. The quantum efficiency of fluorescence is defined as the fraction of excited molecules that will fluoresce. For some chemicals such as fluorescein it reaches nearly 1. As shown on Figure 2.2, the reemitted photon has an energy less than that of the exciting photon because of the loss of vibrational energy. This causes a shift

of the fluorescence spectrum towards longer wavelengths compared to the absorption spectrum, called a Stokes shift.

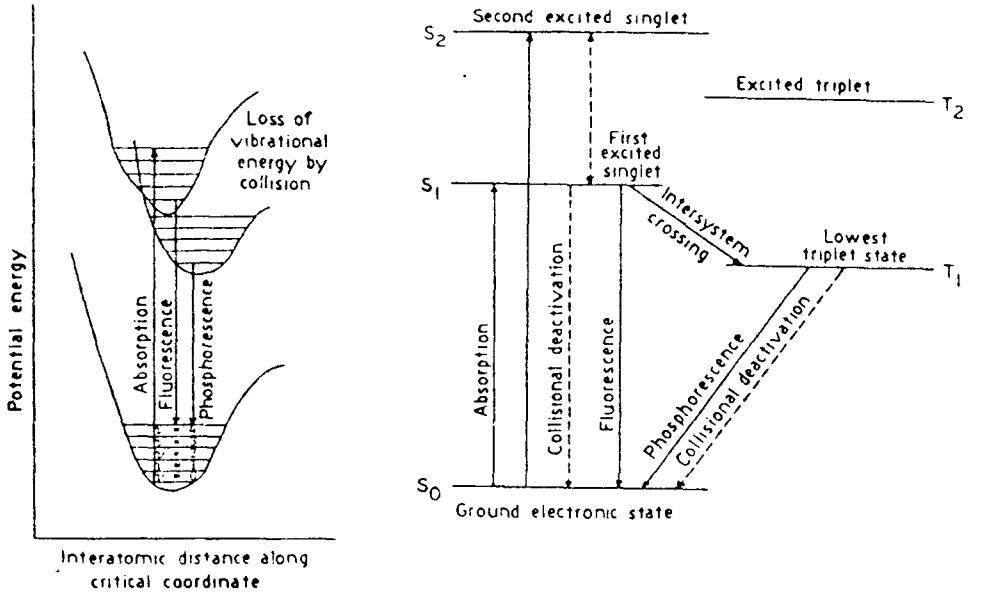


Fig.2.2. Energy levels in photoluminescence (from Guilbault [1973]).

e) Phosphorescence

Another path by which a molecule can return to its ground state, is phosphorescence. The molecule undergoes a transition to a triplet state. In phosphorescence, the light is emitted from the triplet state to the singlet ground state. Due to the low probability of this transition, the time constant of phosphorescence emission is much bigger than that for fluorescence.

f) Internal conversion

An other way a molecule can loose energy, is by internal conversion, also called collisional deactivation in Figure 2.2. In this process the molecule

can convert its absorbed energy into heat. The efficiency of this process is variable but it often allows the molecule to go from any excited state to its lowest excited singlet state in a time shorter than photon emission.

2.2 Relation with the molecular structure

From the chemical structure, one is able to predict if a molecule fluoresces or not. Theoretically any absorbing molecule should be able to fluoresce, but often the non-radiative processes of disexcitation overcome the photon emission, reducing the fluorescence to a level below the detection limit. Some specific features of the electronic structure appear to be needed in order to get a fluorescence signal, they will be presented below. More generally, the complete fluorescence spectrum could theoretically be obtained from the molecular structure since the light emission directly results from the transfer between electronic states. But no accurate theory is yet available to predict the whole spectrum. A few rules exist which can more or less indicate some properties of the fluorescence spectrum. The main rules that can allow one to predict some fluorescence characteristics are presented very briefly below. They represent an oversimplification of reality, but give an idea of the basic mechanisms involved in fluorescence :

- ⋆ The molecule should be aromatic
- ⋆ The lowest excited singlet state should be a $\pi^\star$
- ⋆ Substituents can dramatically influence the fluorescence quantum yield

Aromatic substituent effects on fluorescence		
	Effect on wavelength	Effect on intensity
Alkyl	None	Slight
OH, OAlkyl	Red shift	Increase, but be aware of pH effects
COOH	Red shift	Significant decrease
NH_2, NAlkyl	Red shift	Significant increase, be aware of pH effects
NO_2, NO	Red shift	Significant decrease

Table 2.1. Substituents effects on fluorescence (from Froehlich [1985]).

as shown in Table 2.1.

* The presence of atoms of high atomic weight reduces the fluorescence
* The aromatic ring should be coplanar
* The size of the ring system shifts the fluorescence

2.3 Environmental influences

Due to all the processes competing with photon emission, described earlier, the fluorescence is highly sensitive to environmental effects and this is its major problem. Changes in pH, temperature, or the presence of other components can influence the processes of excitation or disexcitation, and therefore degrade the original characteristics of the fluorescence signal tremendously. Let us consider some of the main effects which can alter the measurements made during this study.

2.3.1 Solvent effects

The solvent influences the measurements in many ways. First the sol-

vent itself produces a signal by scattering. The Rayleigh scattering at the excitation wavelength is not important since this wavelength is ignored in the data treatment. The Raman scattering produces a signal at wavelengths usually used in processing the data. But since the wavelength of this emission is known and it is a characteristic of the solvent, one can easily correct the spectra by subtracting the spectrum of the pure solvent or just not use the affected wavelengths in the data analysis. Figure 2.3 shows the relative importance of Rayleigh and Raman scatterings.

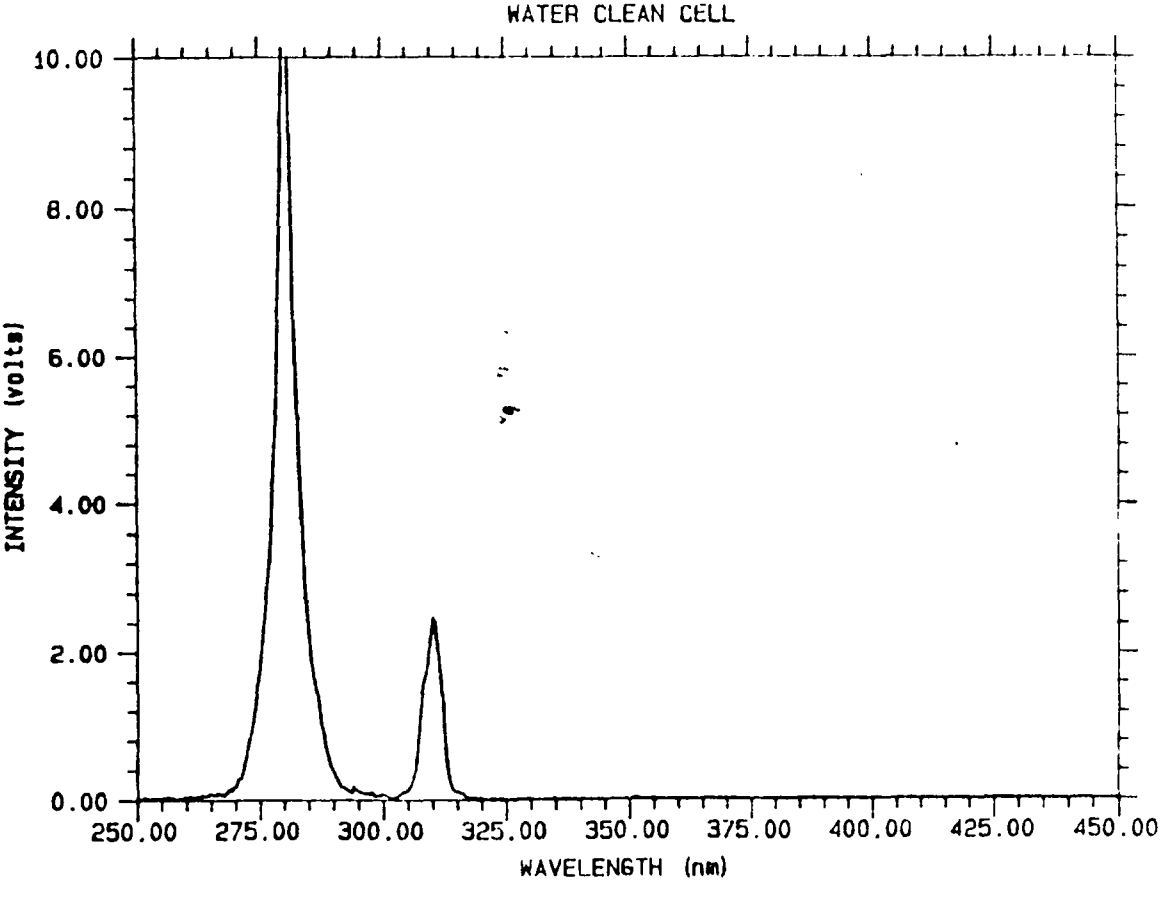


Fig.2.3. Spectrum of distilled water showing the Raleigh scattering (at 280 nm) and the Raman peak (at 310 nm).

Raman scattering is usually very weak and can serve as a test of the instrumentation sensitivity. For example, in water the Raman emission is known to appear at a wave number shift of 3380 cm^{-1} which results in a peak at 310 nm in our experiments where the excitation wavelength is 280 nm.

The solvent can influence fluorescence in many other ways :

- ⋆ Viscosity, since the fluorescence usually increases with it
- ⋆ Polarity, since often in polar molecules the excited state is more polar than the ground state. An increase in the dielectric constant of the solvent increases the stability of the excited state and results in a red shift of both the absorption and fluorescence spectra
- ⋆ Presence of heavy atoms in the solvent molecule which induces a decrease in the fluorescence efficiency as well as an increase in the phosphorescence efficiency.

All the measurements presented in this study are made in water and no consideration of the improvement possible by the use of other solvents has been made since the use of water seems compulsory in any biotechnological device.

2.3.2 Effects of pH

The pH has a very strong effect on the fluorescence. The ionization of the molecule can increase, decrease and even completely eliminate fluorescence. A difference in the pKa of the ground state and excited state will

result in significant changes of the shape of the fluorescence spectra with pH (an example with 2-naphthol is given in Guilbault[1973]). This change can be very important in our experiments since a good part of the work is done on components showing a complex acid-base behavior. Indeed the amino acids have an acidic group ($-COOH$) and a basic group ($-NH_2$). They can even have additional pH sensible groups like tyrosine which has a hydroxyl group. Therefore, good control of pH should be provided. Fermentation measurements are usually made under pH control of the fermenter and therefore no unexpected variations of fluorescence should occur from pH changes.

2.3.3 Fluorescence quenching

Fluorescence is also sensitive to some components which can decrease its efficiency significantly. Dissolved O_2 molecules are known to have such a quenching effect. This quenching can have a dramatic result on the analysis of fermentation data since the quantity of dissolved oxygen is likely to vary tremendously during fermentation. Therefore, an accurate analysis of the sensitivity to dissolved oxygen of the measured signal should be made and an eventual connection of the fluorescence probe with an oxygen probe should be considered. Other known quenching processes can occur with heavy-atom components or when some hydrogen bonding appears between the fluorophore and the solvent or some other solutes.

2.3.4 Effects of temperature

As some general considerations of fluorescence theory show, temperature is a very significant parameter in the fluorescence emission. By inducing a substantial change in the rate of collision between the fluorescent molecules and the solvent molecules, any increase of temperature can reduce the fluorescence efficiency by promoting non radiative processes. Figure 2.4 shows the fluorescence variations as a function of temperature for some commonly used fluorescent compounds. The temperature sensitivity depends strongly on the fluorophore. It is recognized as very high for tryptophan (as high as a 30% decrease between 20^0 C and 30^0 C) but relatively less for tyrosine (20% in the same range of temperature).

The fermentation measurements are probably not affected by this effect since the temperature is regulated in a fermenter.

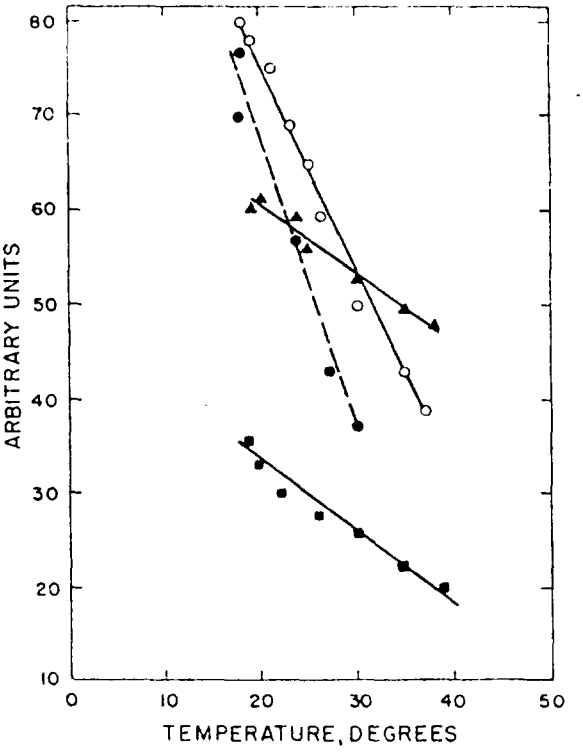


FIG. 2. Variations in fluorescence intensity of several compounds as a function of temperature. All compounds were dissolved in 0.1 *M* phosphate buffer, pH 7.0, except quinine (J. Green, unpublished observations). ○ — ○, tryptophan or indoleacetic acid; ● — ●, indoleacetic acid in buffer saturated with benzene; ▲ — ▲, tyrosine; ■ — ■, quinine in 0.1 *N* sulfuric acid.

Fig.2.4. (from Undenfriend [1962]).

2.4 Use in biotechnology

2.4.1 Presentation

Fluorescence is a very promising technique in biotechnology because many biological compounds fluoresce. Nucleic acids, amino acids as well as carbohydrates can fluoresce. Of high practical importance are the fluorescent coenzymes associated with specific metabolic pathways such as F_{420}, NADH or FADH. Table 2.2 shows some of the important biological fluo-

	Maximum Wavelength	
Fluorophores	Excitation	Emission
NADH, NADPH	350	450
FAD	450	520
Aromatic Amino Acids	275	340
Nucleotides	280	375
Riboflavins	445	520
Antibiotics	var.	var.
F_{420}	425	472

Table 2.2. Luminescence peaks of the important biological fluorophores.

rophores and their fluorescence characteristics.

The fluorescence of NADH is the most commonly used for fermentation monitoring. The functioning of an NADH probe is described below. The fluorescence of some plant pigments such as chlorophyll can also be used to measure the production of biomass or the photosynthetic activity. Some portable fluorescence probes have been developed for measuring the biomass of algae in rivers and lakes from the chlorophyll fluorescence.

Another area of biotechnology where fluorescence is commonly used, is immunoassays. Several fluorometric methods of detecting antigens or antibodies have been developed for example by using :

- ⋆ Antigens or antibodies directly labeled with fluorescent compounds
- ⋆ Enzymes producing fluorescent molecules linked to antigenes or antibodies
- ⋆ Time resolved techniques using fluorescent markers with very long lifetime like europium chelates for labeling antigenes or antibodies

This fast description gives an idea of the variety of fluorescence techniques used in biotechnology. For fermentation processes, the NADH probe is the only technique widely used in on-line monitoring.

2.4.2 The NADH probe

NADH is a coenzyme which has a very important function in energy metabolism. The couple NAD/NADH serves as an electron carrier in many biological reactions, in particular in the respiratory chain and in the production of alcohol. The maximum intensity of NADH fluorescence is obtained at 450 nm with a 350 nm excitation light. A commercial probe, like the Ingold fluorosensor, illuminates the sample with a mercury lamp at 360 nm and measures the intensity of the back-scattered fluorescence at 450 nm. The signal obtained is a measurement of the NADH-concentration within the cell influenced by the number of viable counts, the reducing state of the cells and the environmental effects (pH, temperature, etc.). Despite the complexity of the interactions, some significant correlations between the fluorescence signal, the biomass concentration and the metabolic activity have been shown (cf. Figures 2.5 and 2.6).

A few problems can arise from the direct implantation of the probe in the fermenter, mainly due to bubbles. The sensing side of the probe can be covered by bubbles or they can reduce the fluorescence by their scattering. Since the dissolved oxygen can also influence the measurements, stirrer speed and aeration are critical in fluorescence measurement as shown

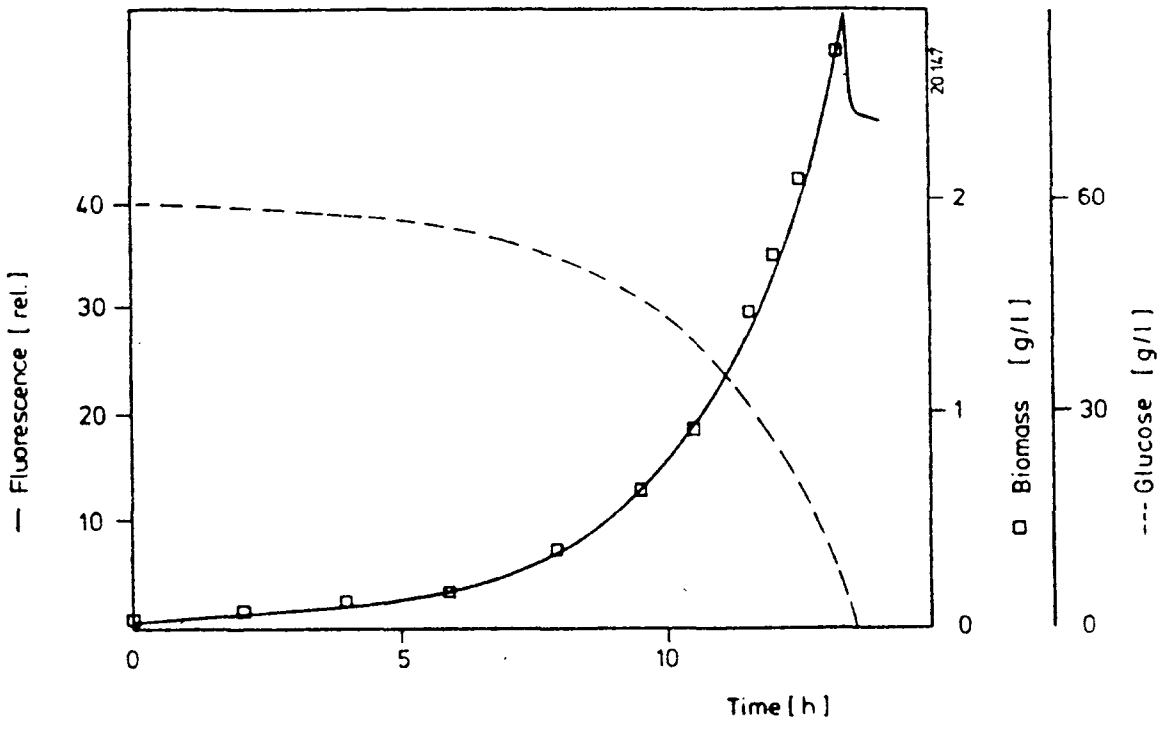


Fig.2.5. Biomass concentration, substrate concentration and fluorescence signal during a batch cultivation of Z. mobilis (from Scheper [1986]).

in figure 2.7.

The tremendous variation of the fluorescence signal with the stirrer speed is explained by Scheper and Schügerl as follows : " Without any aeration the fluorescence signal increases with a stirrer speed up to 370 rpm, since possibly produced bubbles will be washed away from the monitoring part of the sensor. When the revolution rate becomes higher than 370 rpm, the gaseous phase is mixed into the medium and the signal decreases rapidly. When the bubble diameter becomes smaller at higher speeds, the signal increases again. The course of the fluorescence plots is similar for

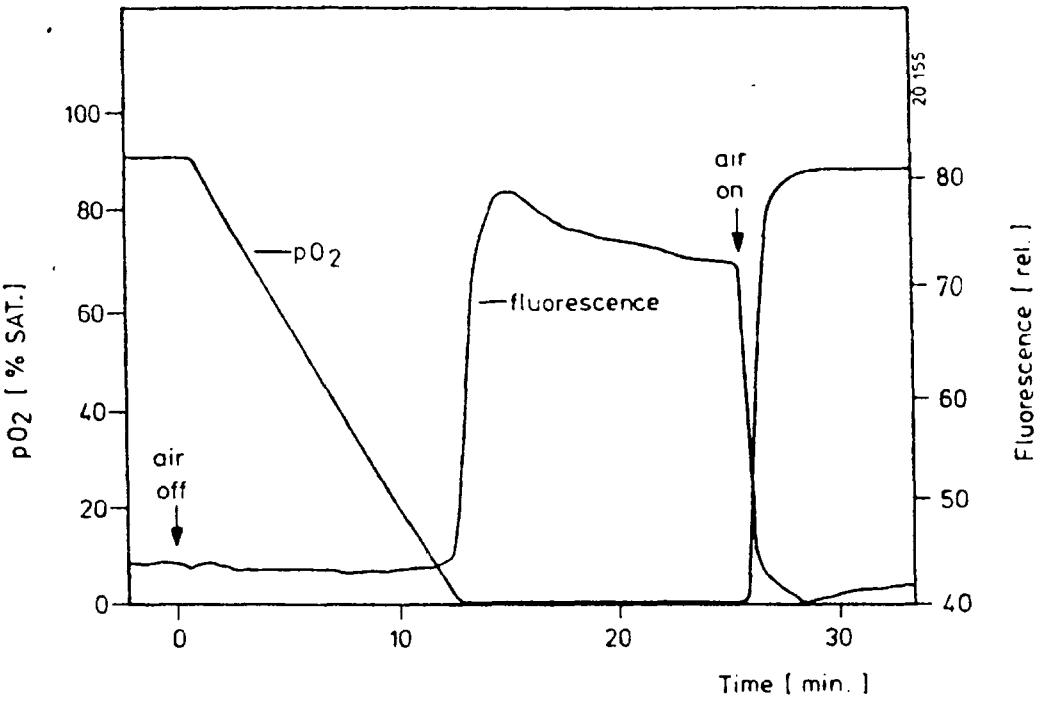


Fig.2.6. NADH-dependent fluorescence and partial pressure of oxygen of a continuous culture of Candida tropicalis during aerobic-anaerobic transitions (from Beyeler [1981]).

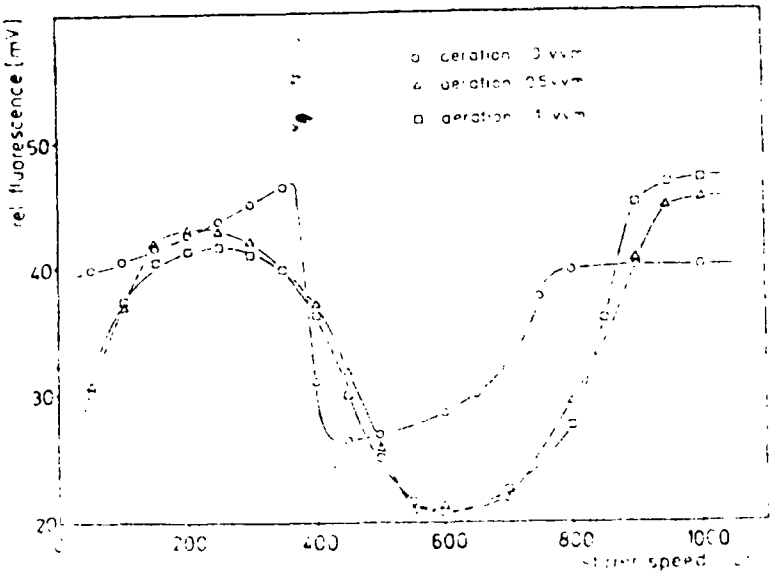


Fig.2.7. Influence of stirrer speed and aeration rate in a 1.5 l bioreactor (from Scheper [1986]).

different aeration rates. The signal decrease is not as abrupt as before and starts at lower revolution rates. At high impeller speeds and aeration of the reactor, the bubble diameter seems to be smaller, for the signals are higher." Even if this explanation is not very convincing, it points out a strong effect of stirrer speed and aeration on fluorescence. This fluorescence variation is of prime importance for any application of fluorescence measurement in fermenter, and it surely requires some consideration in the data analysis process.

Chapter 3

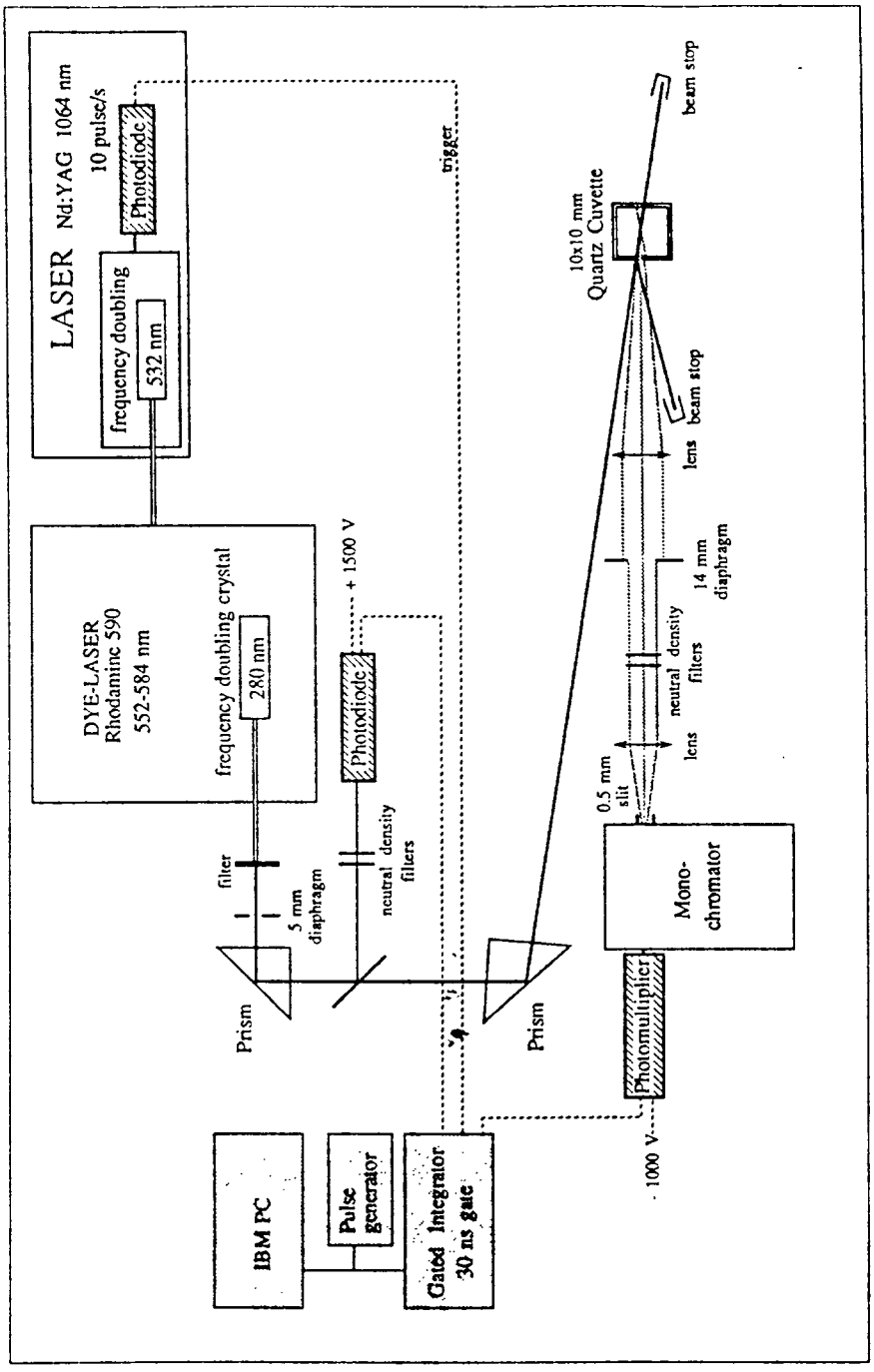
MEASUREMENTS

3.1 The experimental setup

All the experiments presented in this study were made at the National Bureau of Standards facilities in Gaithesburg, under the direction of Hratch G. Semerjian and John J. Horvath.

3.1.1 Description

The experimental setup, used in this research, is designed to measure fluorescence spectra in the ultra-violet and visible region. The choice of the setup configuration has been made keeping in mind a possible application of the sensor. This idea led us to choose a front surface detection geometry. In this configuration, the fluorescence is measured on the illuminated surface, in the opposite direction of the excitation beam as shown on Figure 3.1. This configuration allows measurement in very opaque or turbid solutions. Furthermore, the design of a probe with fiber optics is very simple, so this geometry is likely to be the one selected in any industrial device.



EXPERIMENTAL SETUP

Fig.3.1. Laser induced fluorescence experimental setup.

The ultra violet light beam, generated by a laser, hits the surface of the cuvette with less than 10^0 of incidence. The fluorescent light emitted by the sample is then collected perpendicularly to the illuminated surface by a set of lenses, and directed into the slit of a monochromator. The monochromatic light is measured by a photomultiplier tube. The excitation light intensity is also measured by a photodiode.

Each part of the setup is described more precisely in order to allow a complete understanding of the specific features of the measurements presented below.

3.1.1.1 The light source

The light source is composed of two lasers. The first one, an $Nd:YAG$ laser, generates pulsed light at 1064 nm. After frequency doubling, the light at 532 nm excites a dye laser which produces light at a tunable wavelength between 552−584 nm. Another frequency doubling gives the ultra violetlight usually tuned at 280 nm.

a) The $Nd:YAG$ laser

This laser is a Quantel YG581C laser. The active part in this kind of laser is the Neodymium ion Nd^{+3}. This ion is incorporated in a crystal known as YAG (for yttrium aluminum garnet). The Neodymium ion possesses the interesting property of having four levels usable to create the laser oscillations. This property greatly facilitates the pumping.

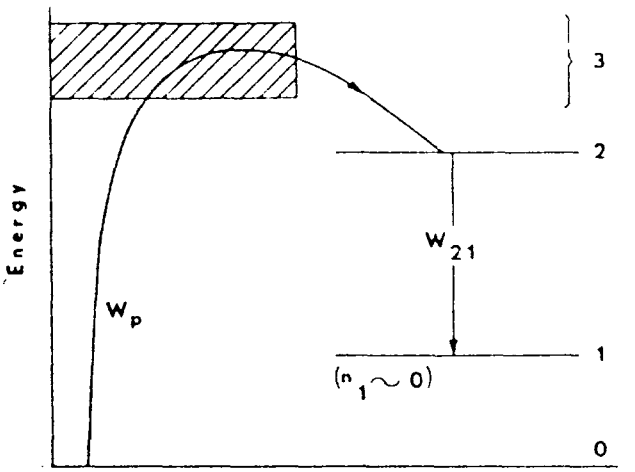


Fig.3.2. Four-level laser (from Young[1986])

As in any optically pumped laser, first a strong light irradiates the laser medium which absorbs this light in band 3, shown in Fig.3.2. However, this band is never populated, because some very fast non-radiative processes allow the transition to the level 2. Then light emission occurs in the transition from level 2 to level 1. This level is also non-populated because of the fast non radiative decay to the ground level 0. Therefore, an inversion of population between level 1 and 2 appears as soon as level 2 is populated. This four-level electronic structure is much more efficient than the three-level one where more than the half of the population should be pumped to the level 2 to produce inversion.

The $Nd:YAG$ laser works in pulsed mode. It generates 10 pulses per second, each one being nearly 10 ns long. The energy of each pulse is 1.2 J. This means that the average light power is 12 W, but the peak power is 120 MW. Such a high peak power is obtained by Q-switching.

gence like KDP (potassium dihydrogen phosphate), the efficiency of conversion to second-harmonic waves can be increased by phase matching the incident and second-harmonic waves, up to 15-20% for an input power of 100 $MWcm^{-2}$.

The light coming from the dye laser is frequency doubled in order to get the 280 nm ultra violetlight that is used in the experiments. The output power after this operation is reduced to 12 mJ per pulse. The diameter of the beam is approximately 5 mm.

3.1.1.2 The sample illumination

The beam which leaves the laser, crosses a quartz plate, which reflects 4% of the light to a photodiode used to record the laser power. The light is then directed to a $1 \times 1cm$ quartz cuvette which contains the sample. The incidence is approximately 10^0. A high quality quartz cuvette is indispensable because ordinary glass absorbs the light at 280 nm and even quartz may absorb a little and fluoresce at the wavelengths used in the experiments if it contains any impurity. The fluorescence is collected by a set of lenses. The first lense collimates the light coming from the cuvette in a parallel beam. The focal point of the first lens is the middle of the cuvette. A 14 mm diaphragm is placed in the beam path before a set of neutral density filters used to reduce the intensity of the light to a value acceptable to the electronics. Another lens focuses the beam on the slit of the monochromator. It is easy to compute the efficiency of the fluorescence

collection. The focal length of the first lens is 6 inches. The solid angle outside the cuvette can be computed as $2\pi(1-\cos\alpha)$ where $\tan\alpha = 7/(6\times 25.4)$ or $\alpha = 2.63^0$. But due to refraction, it is the angle inside the liquid which should be considered. So α_l is computed by $n_l \sin\alpha_l = n\sin\alpha$ which gives $\alpha_l = 1.98^0$. Therefore the solid angle is 3.7410^{-3} *str* which represents an efficiency of the light collection of 0.03%.

3.1.1.3 The measurement devices

The measurement of the fluorescence is made by a monochromator and a photomultiplier, and the laser intensity is recorded through a photodiode.

a) The monochromator

A monochromator GCA/McPherson EU-700 was used. Its optical system is composed of two parabolic mirrors, one for collimating and the other for focusing.

The wavelength dispersing element is a plane diffraction grating, which has three main characteristics :

- ⋆ Fixed-focus scanning
- ⋆ Linear relationship between mechanical movement and wavelength selection through the use of a sine bar
- ⋆ Wavelength dispersion is constant over the entire scanning range

The grating is a piece of glass very finely ruled. The interferences between the different diffracted light beams result in a dispersion of the light with

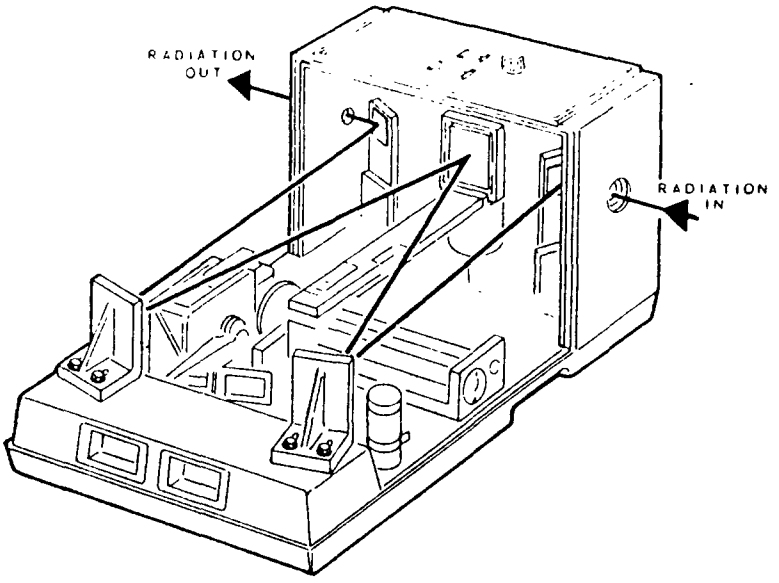


Fig.3.4. GCA/McPherson EU-700 monochromator (from GCA/McPherson instrument manual)

an angle depending on the wavelength. The grating equation is :

$$m\lambda = d(\sin i + \sin\theta)$$

where i is the angle of incidence, θ the angle of diffraction, d the distance between two rules, m the order of the interference and λ the wavelength. One can see in this equation that the different orders of interference are superposed. In the measurements where the excitation wavelength is 280 nm, the second order harmonic will begin to interfere at 560 nm. This superposition is not a problem since such large wavelengths are usually not scan. The chromatic resolving power of such an instrument is given by :

$$\lambda/\Delta\lambda = mN$$

where N is the total number of lines. The grating of this instrument is

a $48 \times 48mm$ ruled area at 1180 $lines/mm$. Since the first order of interference is used, the theoretical resolution is 0.1 Å. But considering the slit width used in the measurements (0.5 mm), the resolution is limited by the dispersion to 1 nm, which is not very good, but sufficient since no fine structure appears in the spectra of the biological components studied. The scanning speed of the monochromator can be varied between 0.05 $Å/s$ and 20 $Å/s$.

b) The photomultiplier tube

The monochromatic light comming out of the monochromator is measured by a Hamamatsu R955 photomultiplier tube. The photomultiplier is a photosensitive device consisting of a photoemissive cathode followed by focusing electrodes, an electron multiplier, and an electron collector. When light enters the photocathode, the photocathode emits photoelectrons into the vacuum. These photoelectrons are then directed by the focusing electrode voltage towards the electron multiplier, where electrons are multiplied by the process of secondary emission. The multiplied electrons are collected by the anode as an output signal. The photomultiplier tube used has multialkali (Na-K-Sb-Cs) photocathode which, combined with a fused-silica window, gives a very wide spectral response from the ultraviolet to the near infrared region with a quantum efficiency varying between 15-25% and a sensitivity of 60-70 mA/W in our range of measurement. The electron multiplier is a 9 stage circular-cage, giving an amplification of 10^7 for

a rise time of 2.2 ns.

c) The photodiode

The laser output intensity is recorded by an ITT F4018 photodiode which has a sensitivity of 30 $\mu A/lumen$ and a quantum efficiency of 10 % at 280 nm. This is a very fast device with a rise time less than to 0.5 ns.

3.1.1.4 The electronic processing

The photomultiplier tube and the photodiode described earlier, are connected to a boxcar integrator. This electronic system performs an integration of the two signals over the time period of a pulse, approximately 30 ns. A photodiode, installed in the $Nd:YAG$ laser, triggers this integration. The boxcar also allows to make an averaging of 3, 5, 10 or 30 measurements. The averaging reduces the noise on the measurements but has some drawbacks. Indeed, since the monochromator keeps scanning, the averaging is made with measurements taken at different wavelengths all shorter than the one the average value is affected at. This method of averaging results in a shift of the measurements towards longer wavelengths. If v is the scanning speed of the monochromator in nm/s, the step between two flashes of the laser is $v/10$ nm . Therefore if the boxcar averages N measurements, the computer can take a sample every $Nv/10$ nanometer. This rate of measurement allows that every pulse of the laser is used for one and only one measurement, preserving the independence of the mea-

surements and taking advantage of all the information available. When this averaging procedure of the boxcar is used, one should take care to shift all the measurements by $-Nv/20\ nm$ in order to get an unbiased spectrum. The amount of noise is then reduced by $\sqrt{N}$.

This reduction of noise is only obtain by a reduction of information (the number of measurements is divided by N) and is with this respect similar to smoothing techniques described below, even if it is directly accomplish by the electronics.

An IBM AT controls all the measurement process. The user can choose the scanning speed of the monochromator and the rate of data acquisition through a program written by researchers in the National Bureau of Standards. The data acquisition rate is limited to 10 readings per second since the measurements are integrated over one pulse and there are 10 pulses/s.

3.1.2. The measurements presently done

The setup described above is used for fluorescence measurement. Since one purpose is to determine concentration measurements in a fermenter, a wavelength of 280 nm was chosen for excitation. At this wavelength, many organic molecules absorb. This is true for the aromatic amino acids, tyrosine, tryptophan and phenylalanine. In particular tryptophan has a peak in absorption at this wavelength and since it is the most fluorescent protein component, a 280 nm excitation wavelength seems to be well suited for measurements in a fermenter.

Two kinds of measurements have been performed. The first ones were made on pure and mixed amino acids. For these the $1 \times 1cm$ quartz cuvette described earlier was filled with the sample each time. The second measurements were made on a fermenter using a flow cell. The fermenter solution was continuously pumped from the fermenter into the cell. Measurements were taken at regular intervals on the flowing liquid. The operating conditions were slightly different in the two cases.

a) The amino acid measurements

The fluorescence was recorded between 260 and 460 nm. This range allows having all the intesting peaks (tryptophan at 340 nm, tyrosine at $305nm$, etc.). The scanning speed was usually $10\mathring{A}/s$ which made our measurements length 3 min 20 sec. Sometimes the internal averaging of the boxcar was used. In this configuration 30 measurements were averaged and 2 readings per second were taken. More often the maximum rate of data acquisition of 10 readings per second was selected. Naturally no averaging was made at this rate.

b) The fermentation measurements

For the fermentation measurements, the fermentation broth was pumped from the fermenter through a flow cell where the measurements were done. The flow cell was a 7×10 mm quartz cuvette and the broth was pumped at approximately 10 ml/min. Some bubbles were circulating in the cell but it

has been checked that they did not influence the measurements. Indeed the probability of interaction with a bubble is small because the laser pulses are very short. The fluorescence was recorded between 250 and 550 nm at $10\mathring{A}/s$ with a data acquisition rate of 10 readings per second.

3.1.3 The other measurements available

This setup can be used to measure some other very interesting fluorescence characteristics, which have not been studied in this project but should be considered in evaluating the possibilities of laser induced fluorescence.

a) Time-resolved fluorescence

The fluorescence signal duration is, as it was said before, nearly 20 ns. In fact, it varies between 1 ns and 100 ns. The exact duration and the way the signal is decaying, is directly related to the probability of transition inside the fluorescing molecules and therefore this characteristics will change depending of the fluorophores present in the solution. Two components which fluorescence lifetimes differ by 3 nanoseconds can be resolved in a mixture(Cline Love and Sherer[1980]). A fast photomultiplier allows this kind of measurements. The figure 3.5 shows a time-resolved measurement performed at the National Bureau of Standards.

b) Excitation-emission matrix

This kind of measurements developed in the department of Chemistry of the University of Washington in 1975 using a new measurement device, the

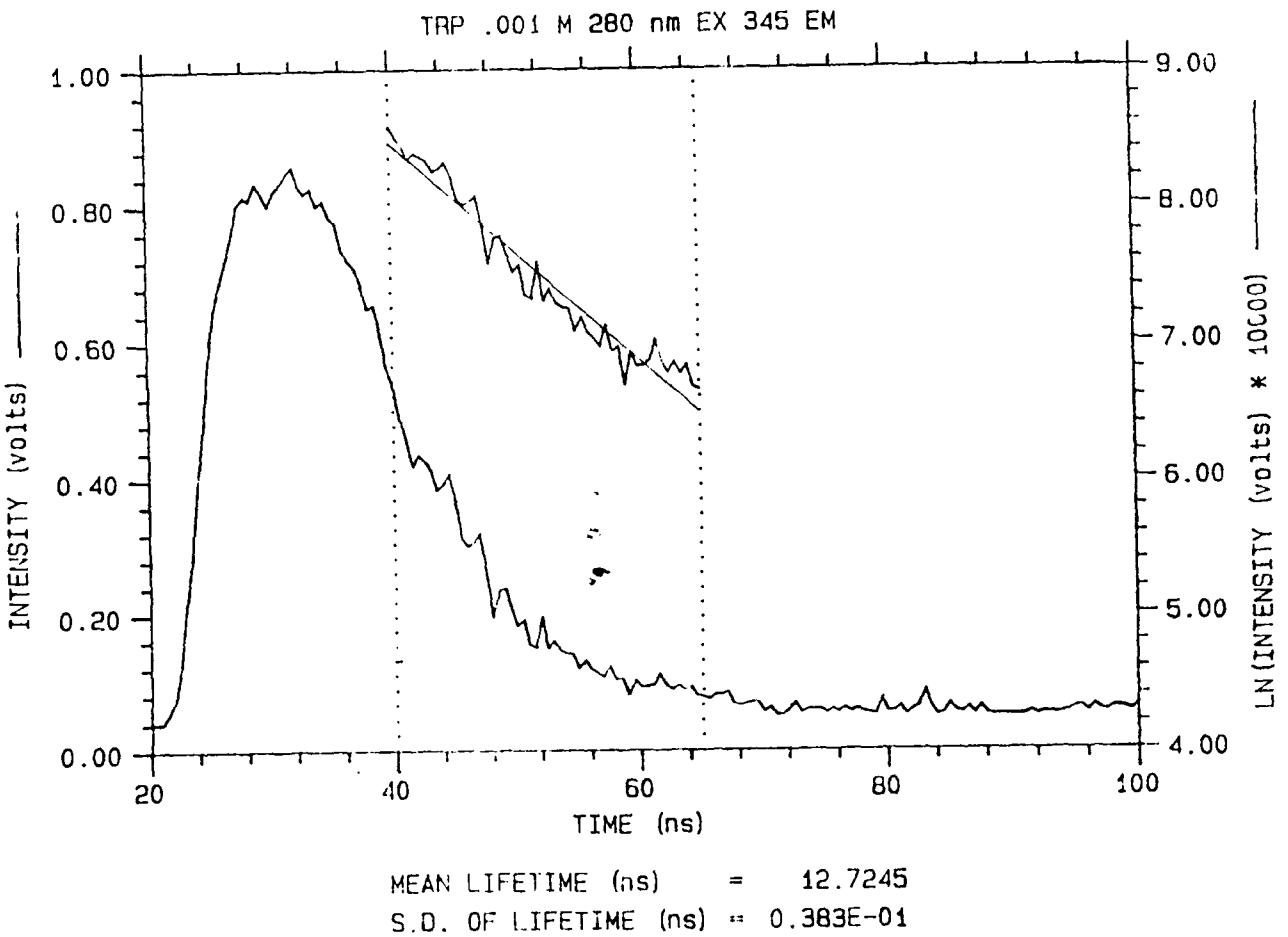


Fig.3.5. Time-resolved fluorescence

videofluorometer, which allows one to measure 241 spectra at 241 different wavelengths in less than 20 ms (Warner[1975]). It presents the great advantage to fully described the fluorescence. The fluorescence matrix obtained by the videofluorometer is composed as follows :

- ⋆ Each line is an emission spectrum at the excitation wavelength λ_{ex}^{i}
- ⋆ Each column is an excitation spectrum at the emission wavelength λ_{em}^{j}

  This means that the term $m_{i,j}$ is the intensity of the fluorescence emitted at the wavelength λ_{em}^{i} when the sample is illuminated at the wavelength λ_{ex}^{j}. The measurement of this matrix is performed by diffracting the excitation light before it hits the sample and then diffracting the fluorescence signal in a direction perpendicular to the direction of the first diffraction. Therefore each diode of a SIT (Silicon Intensified Target) vidicon is hit by a beam which is a single wavelength of the fluorescence emitted by the sample excited at a single wavelength. An excitation-emission matrix resulting of this measurement technique is shown in figure 3.6.

This measurement is naturally impossible to perform with a laser since the emission is monochromatic. But it is possible by tuning the dye-laser to obtain measurements at a few different wavelengths in order to get a matrix of measurements.

c) Two-photon excitation

The high power of the laser allows one to consider some measurements

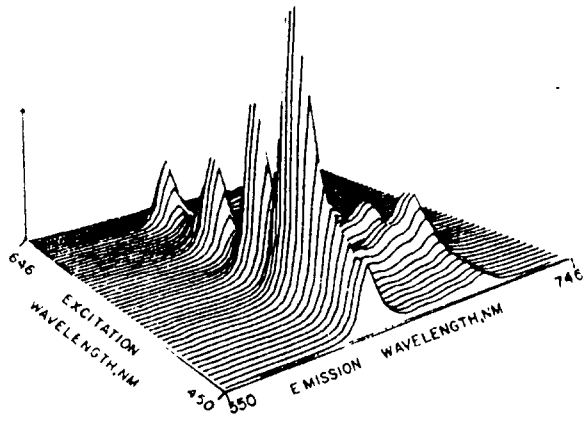


Fig.3.6. Excitation-emission matrix (from Warner [1977])

which are impossible with classical light sources. Two-photon fluorescence is one of these techniques which require a very high intensity source. The idea consists in illuminating the sample with a light at a frequency half than that at which it absorbs which produces an excitation of the molecule by absorption of two photons. This technique increases the selectivity of fluorescence. Figure 3.7 shows measurements of two-photon excitation fluorescence realized at the National Bureau of Standards on tryptophan and tyrosine samples.

Tryptophan clearly exhibits fluorescence in the 340 nm range as it does under a 280nm excitation. Tyrosine does not show any fluorescence at the wavelengths at which it fluoresces under a 280 nm excitation.

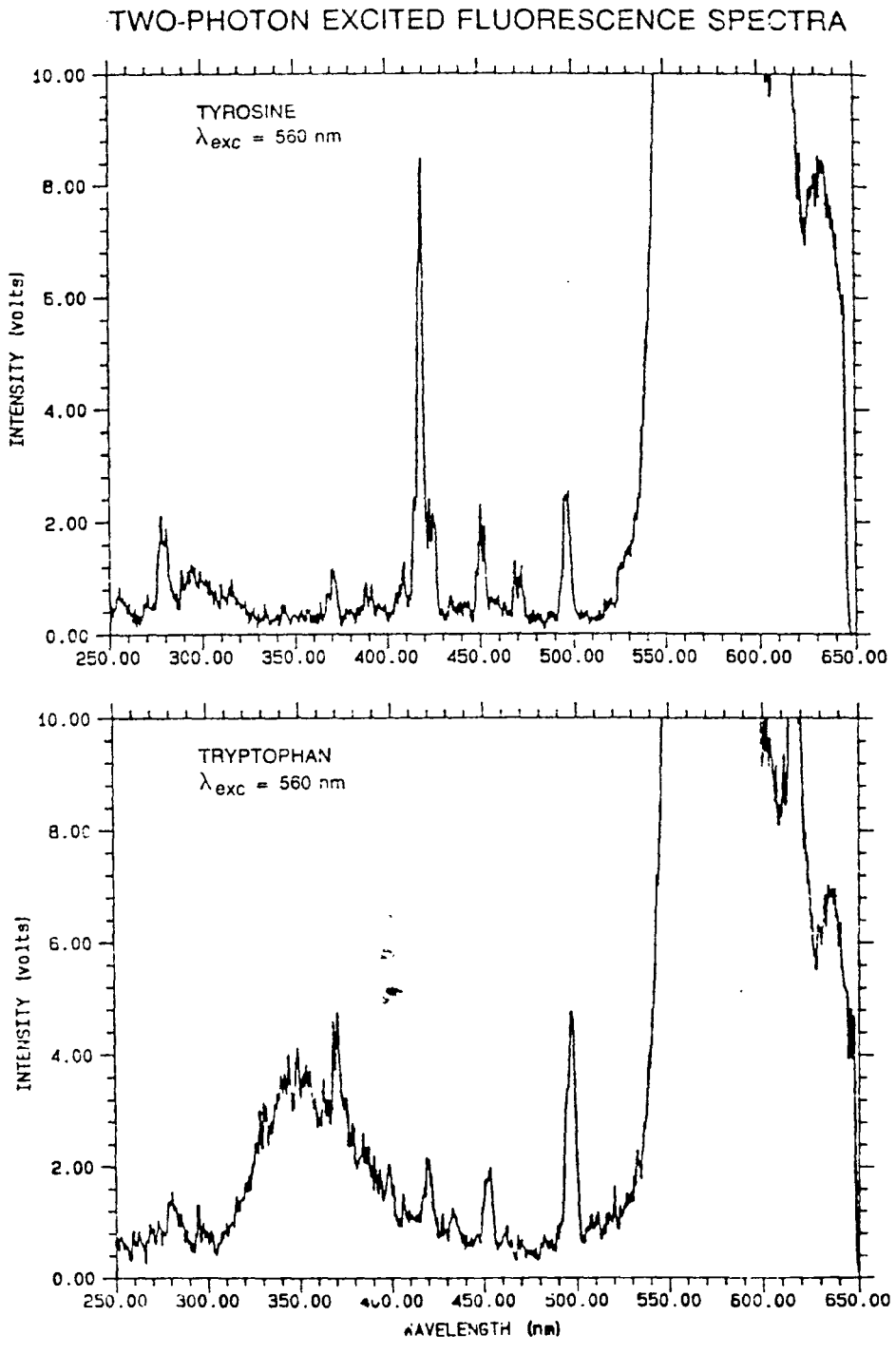


Fig.3.7. Two-photon fluorescence measurements on tryptophan and tyrosine samples.

3.2. The pure amino acids measurements

In order to test the performance of this setup, some experiments have been made with pure amino acids. The fluorescence of the amino acids have been studied by many researchers (Teale and Weber[1957], Konev[1967]) because of its importance in proteins fluorescence. The main results are presented below.

3.2.1. Fluorescence of the amino acids

Only the three aromatic amino acids exhibit a significant fluorescence. The fluorescence of cysteine has also been measured but is considerably less intense than the fluorescence of tryptophan, tyrosine and phenylalanine. As shown in Figure 3.8, these three compounds have an aromatic ring which is believed to be responsible for the intense fluorescence. The structural differences give nevertheless very different luminescent properties to the three molecules.

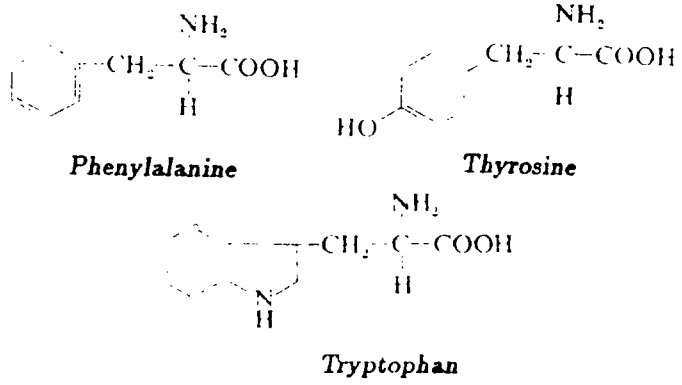


Fig.3.8. Tryptophan, thyrosine and phenylalanine structures.

3.2.1.1 Absorption

The absorption spectra of the three aromatic amino acids is shown in Figure 3.9. Tryptophan has a much larger absorptivity than tyrosine or phenylalanine. Its absorption spectrum is very wide extending up to 300 nm. The main peak is at 280 nm and a small side peak is visible at 288 nm. Tyrosine spectrum has no distinguishable structure, it peaks at 275 with a molar absorptivity approximately three times smaller than the peak absorptivity of tryptophan. The tyrosine absorption decreases quickly after 280 nm and become almost null at 290 nm. Phenylalanine absorbs even less, its peak at 257 nm is 15 times smaller than the one of tryptophan. The phenylalanine spectrum shows three different peaks at 251, 257 and 263 nm and decreases abruptly after the third peak until 275 nm where its absorptivity is negligible.

3.2.1.2 Fluorescence

Although the aromatic ring is the origin of the fluorescence in the three compounds, tryptophan, tyrosine and phenylalanine, they have very distinct fluorescence spectra as shown in Figure 3.10.

a) Tryptophan

The fluorescence spectrum of tryptophan in aqueous solution is a broad, structureless band with a maximum at 348 *nm* and a half width of 60 *nm*. The Stockes' shift (wavelength difference between the maxima of the exci-

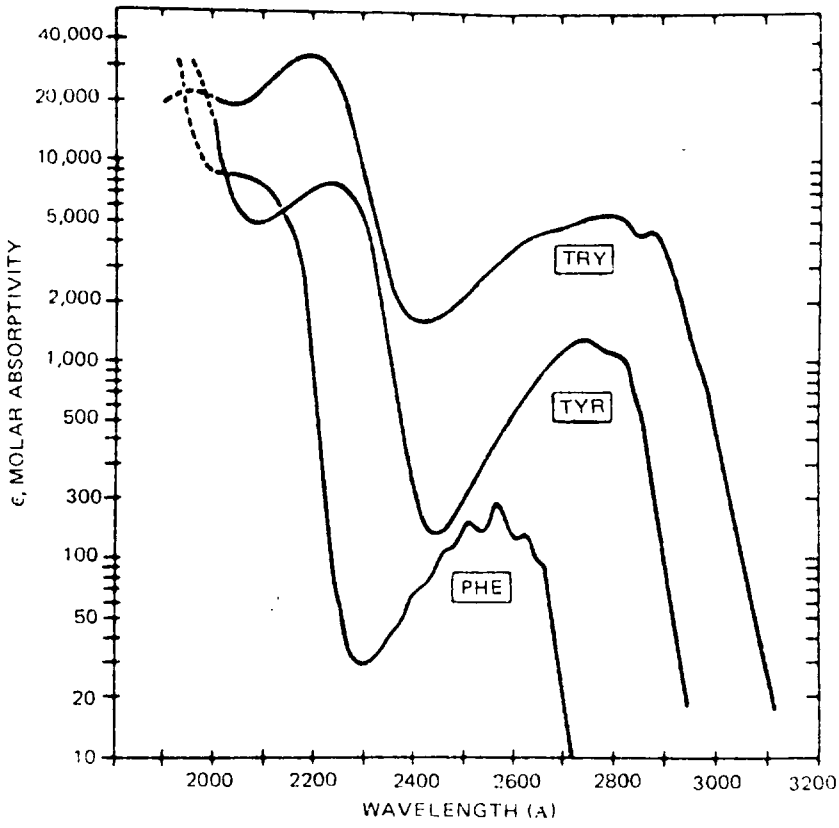


Fig.3.9. Absorption spectra of tryptophan, tyrosine and phenylalanine (from Wetlaufer [1962]).

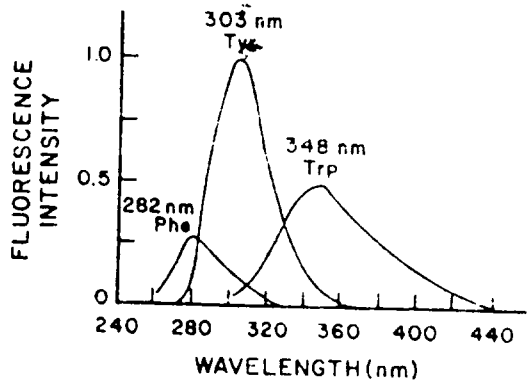


Fig.3.10. Fluorescence spectra of tryptophan, tyrosine and phenylalanine(from Teale and Weber [1957]).

tation and emission peak) is strong, approximately 70 nm. This strong shift results in a clear separation of the absorption and emission spec-

tra. The strong influence of temperature on tryptophan fluorescence has already be mentioned in Chapter 2. The intensity of the fluorescence decreases of 5% per degree Celsius between 20^0C and 30^0C. The pH of the medium induces slight changes in the shape of the tryptophan fluorescence spectrum. $R-\underset{\underset{NH_3^+}{|}}{C}-COOH$ peaks at 347 *nm*, $R-\underset{\underset{NH_3^+}{|}}{C}-COO^-$ at 353 *nm* and $R-\underset{\underset{NH_2}{|}}{C}-COO^-$ at 360 *nm*. (Konev [1967]). The quantum yield of fluorescence, defined as the ratio of the number of photons emitted over the number of photons absorbed, changes also with pH. $R-\underset{\underset{NH_3^+}{|}}{C}-COOH$ has a high yield of 0.51, $R-\underset{\underset{NH_3^+}{|}}{C}-COO^-$ a yield of 0.20 and $R-\underset{\underset{NH_2}{|}}{C}-COO^-$ only 0.085.

b) Tyrosine

The peak of tyrosine fluorescence occurs at 303 nm, a wavelength much shorter than tryptophan. The half width of the peak is also smaller, only 38 nm. The sensitivity of tyrosine to temperature is not as important as the tryptophan one but still reaches 3% per degree Celsius. Tyrosine has a quantum yield of 0.21 at neutral pH but at pH lower than 4 the quantum yield is reduced to 0.056 due to the conversion of the carboxyl group to the un-ionized state. At higher pH, the dissociation of the phenol hydroxyl (pK=9.7) occurs and creates a dissociated form which is believed to be non fluorescent (White[1959], Cowgill[1963]).

c) Phenylalanine

The spectrum of phenylalanine presents a main peak at 282 nm and a half width of 28 nm. Vladimirov[1959] and Vladimirov and Burshtein[1960] have resolved the structure of the fluorescence spectrum, recording maxima of the vibrational structure at 282, 285 and 289 nm, and a shoulder at 303 to 305 nm. The quantum yield of the fluorescence of phenylalanine is low, about 0.04 according to Teale and Weber[1957] and is constant over the whole excitation spectrum. The fluorescence is slightly quenched in highly basic or highly acid media.

3.2.2 Amino acid measurements

Measurements of amino acid fluorescence have been made for this study in order to estimate the performance of the setup and to try to resolve simple amino acids mixtures. The excitation wavelength was set at 280 nm, near the excitation peaks of tryptophan and tyrosine. However, the absorption of phenylalanine is almost null at this wavelength, and the fluorescence peak is hidden by the scattering. Therefore phenylalanine was not detectable. The rest of the study focuses on tryptophan and tyrosine. Many measurements have been made on tyrosine and tryptophan at room temperature, at concentrations varying between 10^{-3} and 10^{-6} mol/l. The signal of the photomultiplier tube is usually divided by the measurement of the photodiode in order to get a spectrum corrected for the laser intensity variation (this procedure is justified below). Figure 3.11 presents the two spectra of tryptophan and tyrosine in aqueous solution without any

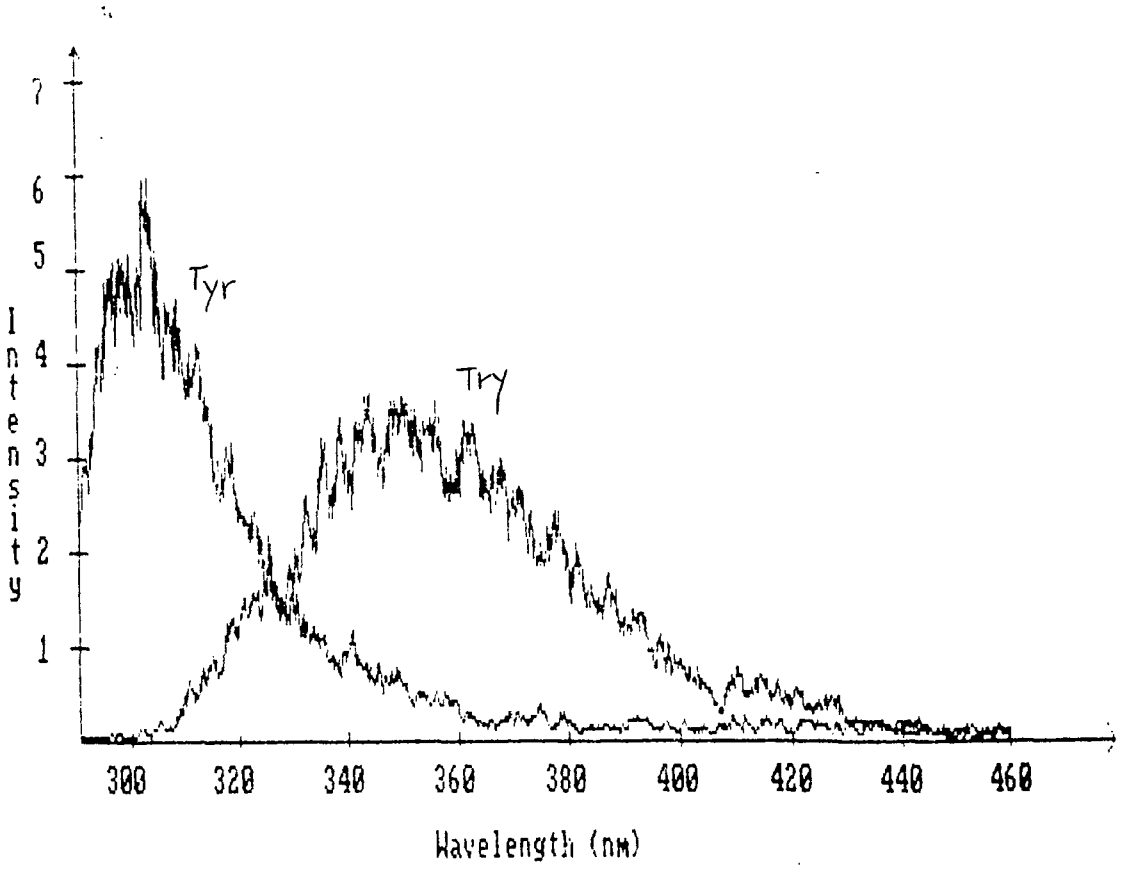


Fig.3.11. Measured spectra of tryptophan $10^{-3}M$ *and tyrosine* $10^{-3}M$.

smoothing.

3.3 Noise and Reproducibility

The first step in the fluorescence spectra analysis has been to study the reproducibility of the measurements which is a major concern in any sensor. Repeated measurements of tyrosine and tryptophan have been made in order to quantify the reproducibility obtainable on the the National Bureau of Standards setup. It quickly appeared that the shape of the spectra was very well reproducible but that the intensity was hardly comparable from a day to another. Indeed fluorescence measurements are always given in

arbitrary units because of the lack of reference. This uncertainty on the exact value of the intensity results in spectra hard to compare from one day to another. Some changes in the setup cause this lack of reproducibility observed during the first experiments.

3.3.1 Modifications in the experimental setup

One of the greatest changes which can occur is a variation of laser intensity. The output power of the laser can vary for reasons internal to the $Nd:YAG$ laser and this is not very well understood but also because of the aging of the dye or of a poor alignment of the frequency doubling crystals. The laser intensity is recorded by the photodiode and therefore a correction for the variation of laser intensity should be possible. But the photodiode measurement is not reliable enough. Slight movements of the plate which reflects the laser light towards the photodiode could change the ratio between the photodiode indication and the laser intensity. The sensitivity setting of the photodiode as well as the voltage of the power supply were not always recorded. Any comparison of the laser intensity between experimental sessions was therefore forbidden. The fluorescence measurement itself could be influenced by moves of the cuvette, changes in the position and diameter of the laser beam. The sensitivity setting and power supply voltage of the photomultiplier tube were sometimes changed without proper recording. Since the setup was used for other measurements, all the changes cited before are very likely to occur. After some poor results in

the first experiments, reference settings for the main pieces of equipment have been defined in order to improve the reproducibility. However the reproducibility from a day to another is weak and, therefore a set of standard measurements should be done at the beginning of each measurement session or only the measurements of a single session should be used in the analysis.

3.3.2 Degradation of the sample

The first experiments of reproducibility gave poor results. A degradation of the fluorescence signal was occurring during the measurements. As example Figure 3.12 shows a set of measurements done every 10 minutes on a sample which stayed irradiated by the laser during all the length of the experiment.

A significant decrease of the fluorescence signal is observed. Two reasons can explain this variation : temperature variation and photodecomposition. Indeed under the very intense laser light, the sample can warm up as well as undergo chemical transformations. Such a transformation seems to happen on the tyrosine samples which exhibit a yellowish color after a long irradiation. These experiments of degradation were not very reproducible. On Figure 3.13, a second degradation experiment on tryptophan sample is presented and compared with the first one in Table 3.1.

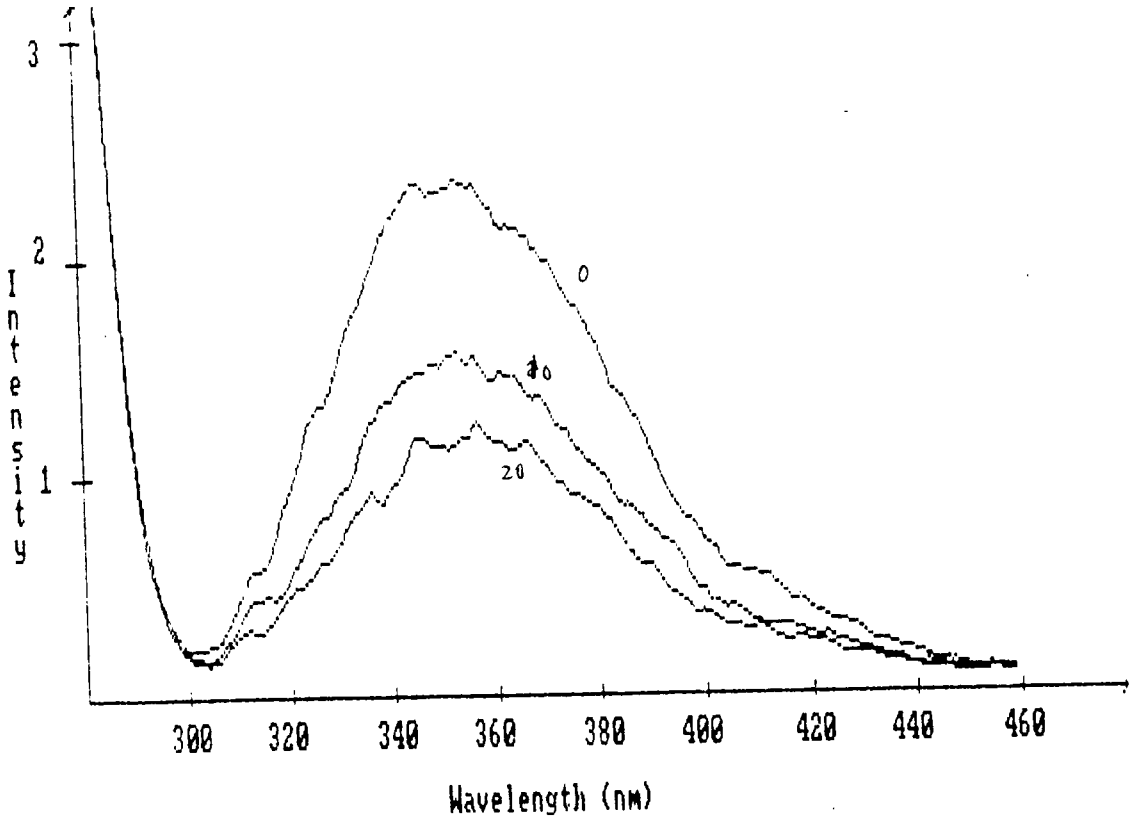


Fig.3.12. Spectra of tryptophan fresh and after 10 and 20 minutes of irradiation. Data recorded with a 30 measurements averaging.

The correlation coefficient used in Table 3.1 is computed by :

$$\frac{\sum_{\lambda} s_1^{\lambda} \times s_2^{\lambda}}{\sqrt{\sum_{\lambda}(s_1^{\lambda})^2 \times \sum_{\lambda}(s_2^{\lambda})^2}}$$

where s_1^{λ} and s_2^{λ} are the two spectra. It is a measure of the linear dependence between two signals which takes the value 1 when the two are proportional and 0 when they are independent. The proportionality coefficient is computed by :

$$\frac{\sum_{\lambda} s_1^{\lambda} \times s_2^{\lambda}}{\sum_{\lambda}(s_1^{\lambda})^2}$$

It gives the ratio of the two signals when they are proportional and is the least squares estimation of this ratio in any case.

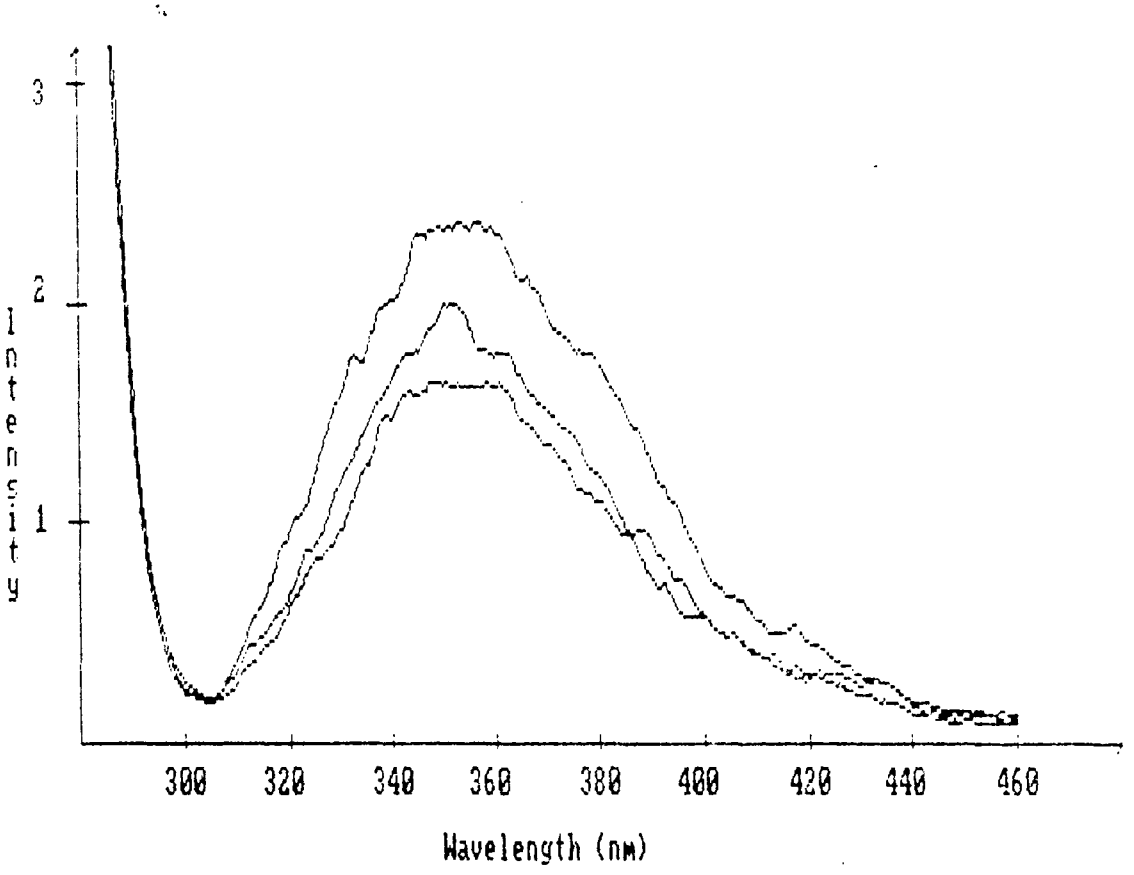


Fig.3.13. Second example of tryptophan degradation.

1^{st} tryptophan sample		
Sample	Ratio	Correlation
Fresh sample	1	1
After 10 min	0.65	0.999
After 20 min	0.50	0.997
2^{nd} tryptophan sample		
Sample	Ratio	Correlation
Fresh sample	1	1
After 10 min	0.73	0.998
After 20 min	0.68	0.998

Table 3.1. Degradation of tryptophan under irradiation. The two columns give the correlation and proportionality coefficients with the fresh sample.

Table 3.1 shows that the degradation occurs in the two cases at different rates. Some free convection circulation of the liquid inside the cuvette

is probably responsible for these changes. Since only a part of the liquid, approximately a tenth, is illuminated by the laser beam, differences of temperature as well as composition (in case of photodecomposition), are susceptible to appear inside the cuvette. It can result in convection currents which can replaced the liquid in the fluorescence collection volume by non irradiated liquid. The results of the experiments would greatly change in such case.

To avoid the effects of the degradation, fresh samples are used for every experiment. With this precaution the reproducibility of the measurements is much better as shown in Table 3.2.

Sample	Signal Mean	Signal Variance	Intensity Mean	Correlation	Ratio
Trp3a	1.043	0.58	3.43	0.998	1.029
Trp4a	1.053	0.59	3.24	0.998	1.020
Trp5a	1.035	0.63	3.10	0.998	1.019
Trp6a	1.070	0.62	2.81	1	1

Table 3.2. Study of reproducibility on 4 samples of tryptophan 10^{-5} M. Correlation and proportionality coefficients are computed with respect to Trp6a.

Table 3.2 shows that the measurements can be reproduced with a correlation higher than 0.99 which is satisfactory. The intensity seems to be reproducible only within 3%. This means that even the concentration of a pure compound can not be estimated better than 3%. Since a mixture is much harder to resolve, one can expect a poor precision of any multicomponent mixture analysis due to this poor reproducibility.

3.3.3 Noise in fluorescence measurement

As it appears at the first look, the spectra obtained with this setup are very noisy. Indeed, fluorescence measurements are subject to three kind of noise : dark-current noise, flicker noise and photon noise.

a) Dark-current noise

Dark-current noise originates in the photomultiplier tube. Due to the high potential difference between anode and cathode, some electrons are emitted from the cathode even when no photon is received. These electrons produce a current, called dark-current because it exists even in the complete darkness. The electronics can correct the signal by subtracting the average of the dark current but its noise can not be corrected. In the National Bureau of Standards setup the dark current noise was typically of 0.02 volts for a signal of 5 volts which means 0.4% of noise.

b) Flicker noise

Flicker noise is due to the variation of the intensity of the light source. The laser light monitored by the photodiode gives a signal similar to the one shown in Figure 3.14. The signal to noise ratio is close to 25 (4% of noise).

But since the intensity of the laser light is recorded and each measurement is normed to a constant light power, one could expect a correction of maybe 90% of the noise and therefore the resulting noise should have one

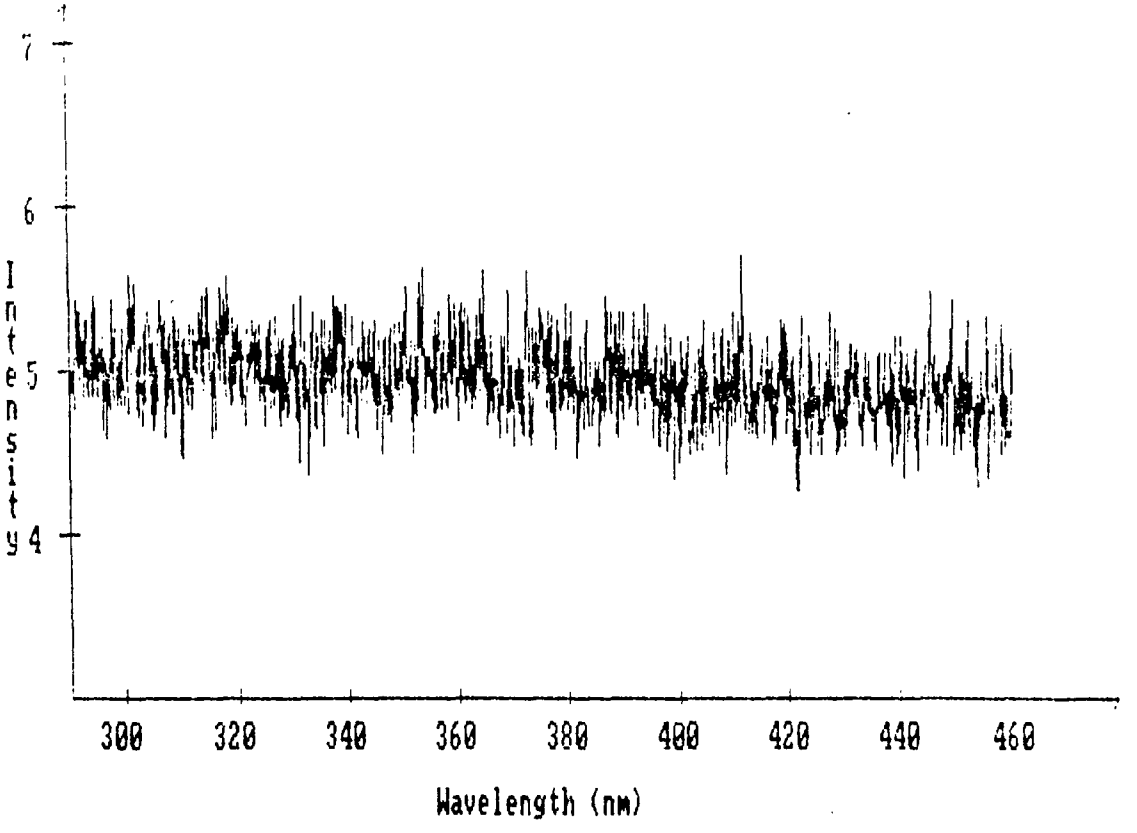


Fig.3.14. Intensity of the laser during a measurement.

order of magnitude less, 0.4 % approximately.

c) Photon noise

Photon noise is another noise resulting from the functioning of the light detector. The interaction between the photons and the detector is probabilistic, due to the nature of light itself. Therefore the signal coming from the photomultiplier tube depends of the number of photons which reach the cathode in a statistical way. Only the integration over a long period of time allows to reduce this white noise inherent to any light measurement. One can estimate the photon noise as follows :

$$I_\phi = \sqrt{eBM/t}\ \sqrt{I} \qquad (1)$$

$$B = 1 + 1/g + 1/g^2 + \cdots$$

where I is the intensity of the signal, I_ϕ the intensity of the photon noise, e the charge of an electron $1.6 \times 10^{-19} C$, M the amplification of the photomultiplier tube 10^7, g the amplification per stage 6, t the integration time 3×10^{-8} s, therefore :

$$I_\phi = 8 \times 10^{-3} \sqrt{I}$$

The intensity used in the measurements is in the order of 2 mA therefore the resulting noise is 0.36 mA which represents 18% of the signal. This level of noise is very high for fluorescence measurements. It is due to the short integration time.

3.3.4 Smoothing

In order to get a signal easier to analyze, one should try to remove the noise as much as possible. The signal is smoothed using a low pass filter. The filtering is done using a Fourier transform. A fast Fourier transform is performed on the spectrum and the high frequencies are set to zero. An inverse transform gives the smoothed spectrum. Many experiments were recorded with 2000 points. The scattering peak is removed from the spectrum and then a 2048-points fast Fourier transform is used. Figure 3.15 shows half of the real part of a transform.

Since the original signal is real, the real part of the discrete Fourier

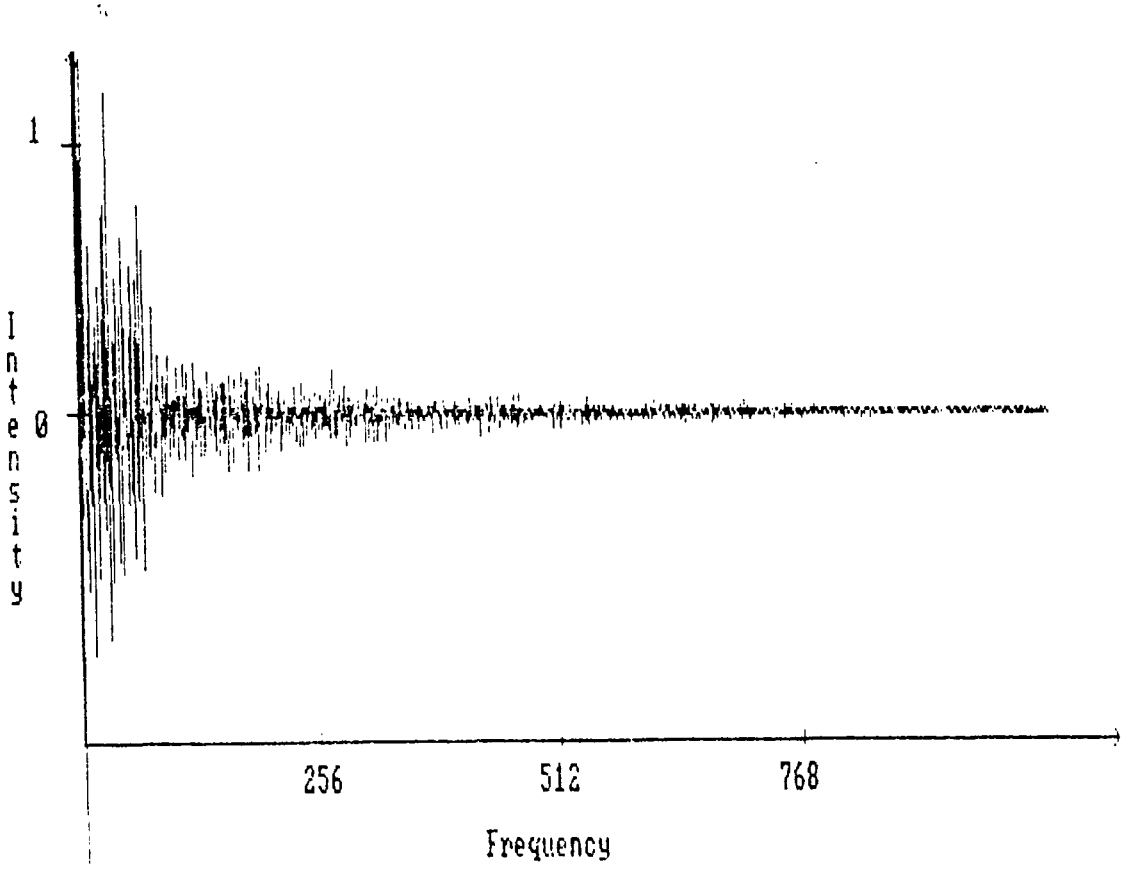


Fig.3.15. Real-part of the Fourier transform of a tryptophan spectrum. Only half of the real-part is shown since it is symmetrical.

transform is symmetrical. The lower frequencies are represented by the extremities of the transform and the higher ones are in the center (the right side on Figure 3.15 since only half of the transform is plotted). On Figure 3.16, the thirty first values of the transforms of two differents tryptophan measurements are plotted.

One can see on Figure 3.16 that only the first terms are reproducible. After the eighth point, the two transforms are completely different and do not seem related in any way. Therefore all the data after this point are assumed to be due to noise only and are discarded. A reverse transformation gives then a smoothed spectrum. Figure 3.17 shows the result of this

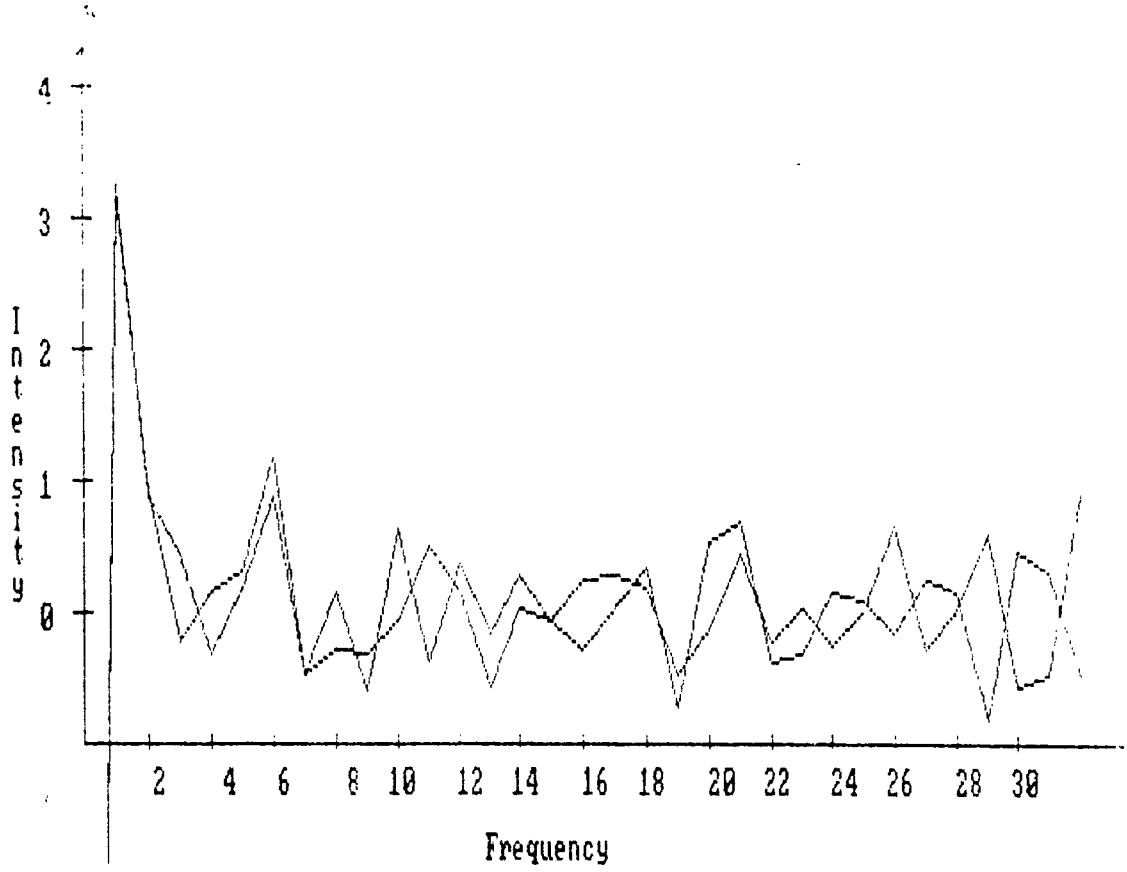


Fig.3.16. First terms of the Fourier transform of two different tryptophan measurements.

smoothing.

Let us consider how the smoothing process influences the reproducibility of the measurements. Table 3.3 gives the correlation and proportionality coefficients of two tryptophan spectra before and after smoothing.

Sample	Ratio	Correlation
Original spectra	0.969	0.989
Smoothed spectra	0.980	0.999

Table 3.3. Correlation coefficients and proportionality of the smoothed and original spectra (computed between 340 and 360 nm).

The noise removal does not change the correlation and proportionality factors, just a slight improvement can be seen. Figure 3.18 shows that the

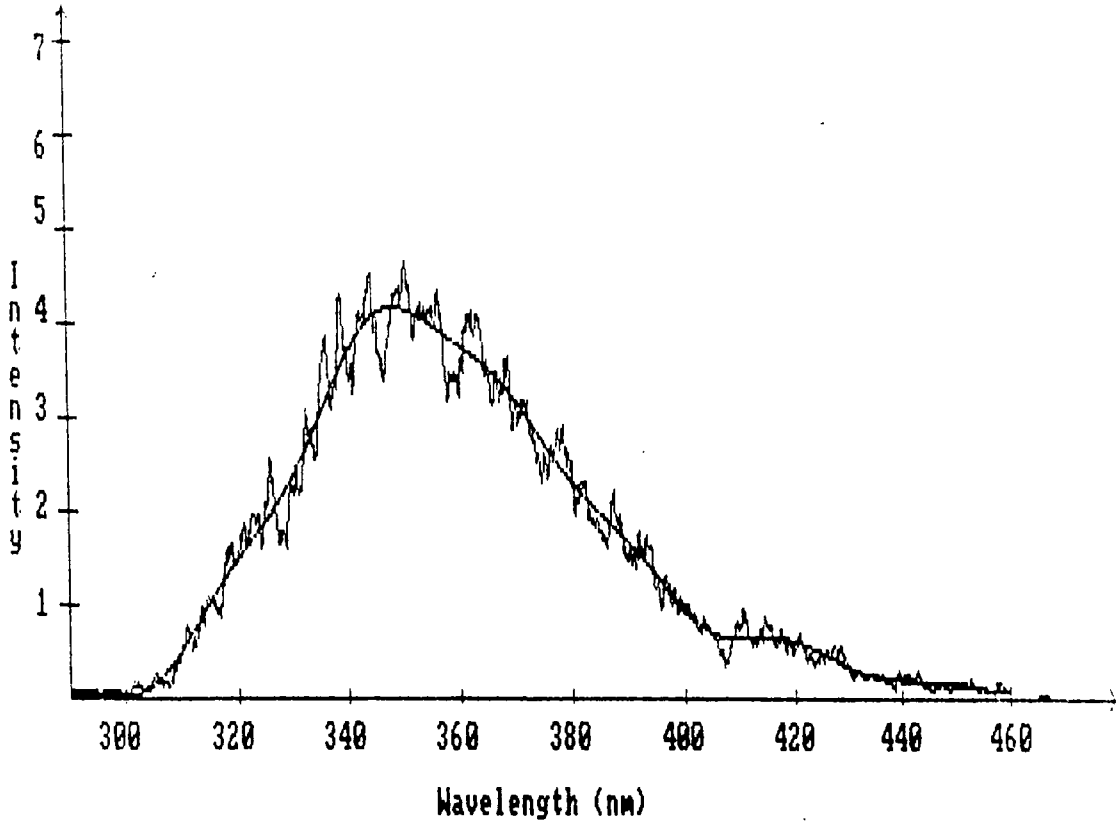


Fig.3.17. Raw and smoothed spectra of tryptophan.

two smoothed spectra are still different. The noise of the measurement can be derived from the smoothed spectrum by subtracting it from the original spectrum. Figure 3.19 shows the noise computed by this method on two different tryptophan measurements.

The correlation coefficient between the two signals is -0.058. This very low value shows that the two signals are independent. Therefore one can conclude that they do not contain any information and that the smoothing process keeps all the significant information and removes only noise.

3.3.5 Analysis of the experimental noise

The noise of the experimental spectra can be separated from the signal

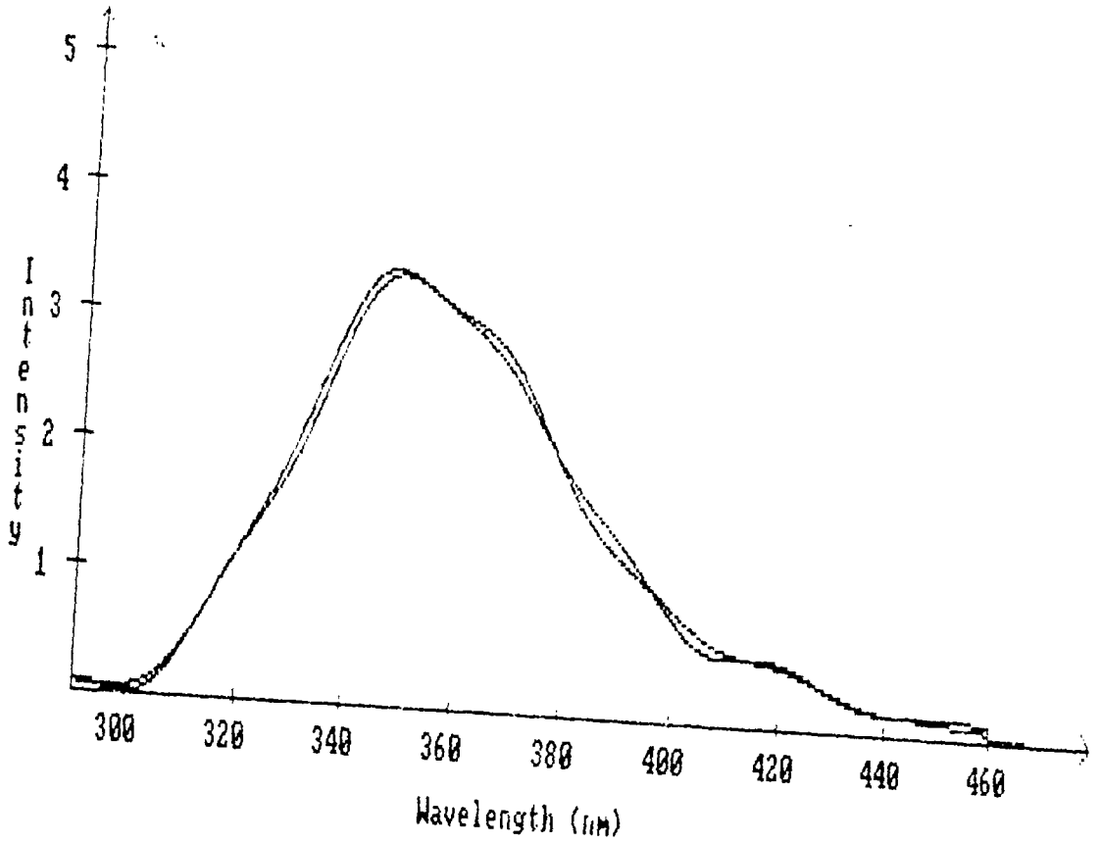


Fig.3.18. Two smoothed spectra of tryptophan.

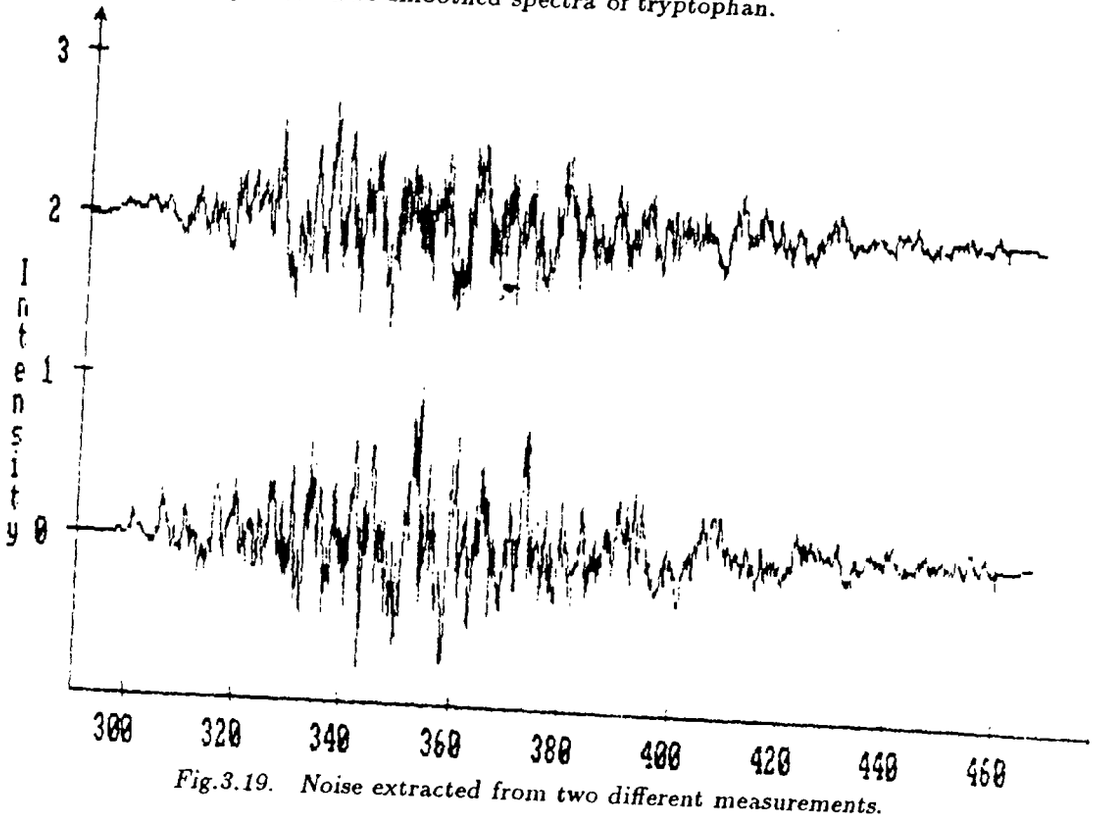


Fig.3.19. Noise extracted from two different measurements.

as explained in the preceding section. One can then study its amplitude and correlation.

The standard deviation of the noise computed between 340 and 360 nm on a tryptophan spectrum is 0.32 for a signal of 4 which means a signal to noise ratio of 12.

a) Correlation with the intensity fluctuations

The intensity of the excitation light was recorded during every measurement. The variations of the laser intensity during a typical run 3 *min* and 20 *sec* long, have been represented on Figure 3.14.

The laser intensity can be smoothed in the same way than a spectrum. This procedure gives a fairly constant signal. The noise of the laser intensity signal can be obtain by difference between the raw signal and the smoothed one. Figure 3.20 shows the result of this process and Table 3.4 gives the characteristics of this signal.

Laser intensity mean value = 1.25

Intensity noise variance = 0.0024

Standard deviation = 0.049

Signal to noise ratio = 25.5

Shift (nm)	0.1	0.2	0.3	0.4	0.5
Correlation	0.328	0.157	0.048	0.016	-0.047
Shift (nm)	1	2	3	4	5
Correlation	0.113	-0.026	0.102	-0.097	-0.124

Table 3.4. Characteristics of the laser intensity noise.

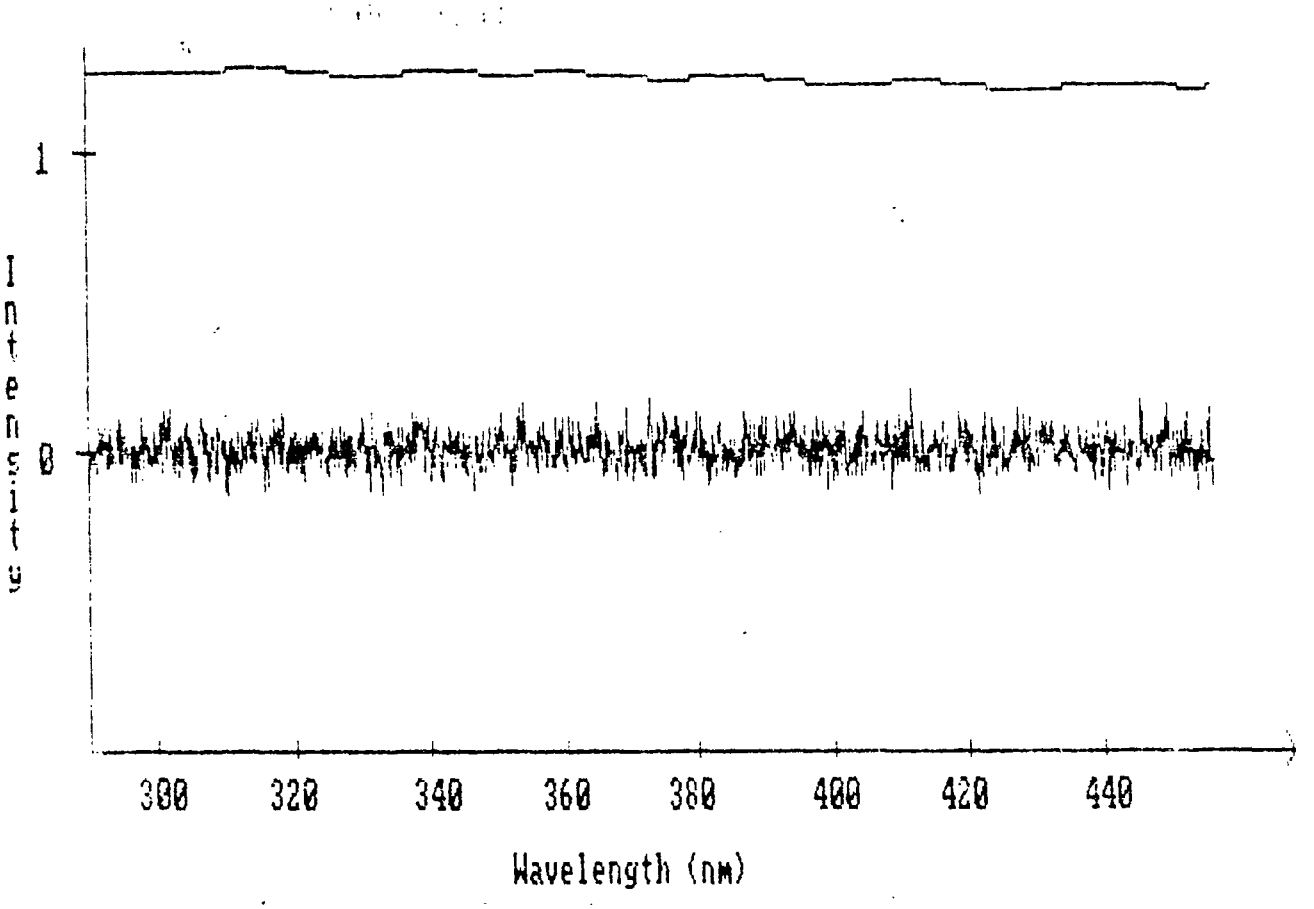


Fig.3.20. Laser intensity : smoothed signal and noise.

This signal is very close of a white noise. The values of autocorrelation do not show any significant corrélation. This result is in agreement with the theory. The correlation between the laser intensity noise and the spectrum noise can then be computed. Figure 3.21 shows these two noises extracted from a tryptophan measurement.

The correlation coefficient between the two is 0.257. This small number means a weak correlation. One could have expected this result since the noise on the fluorescence measurements is more than two times higher than the laser intensity noise. Therefore there is another source of noise which as shows the small correlation coefficient is not correlate to the laser intensity.

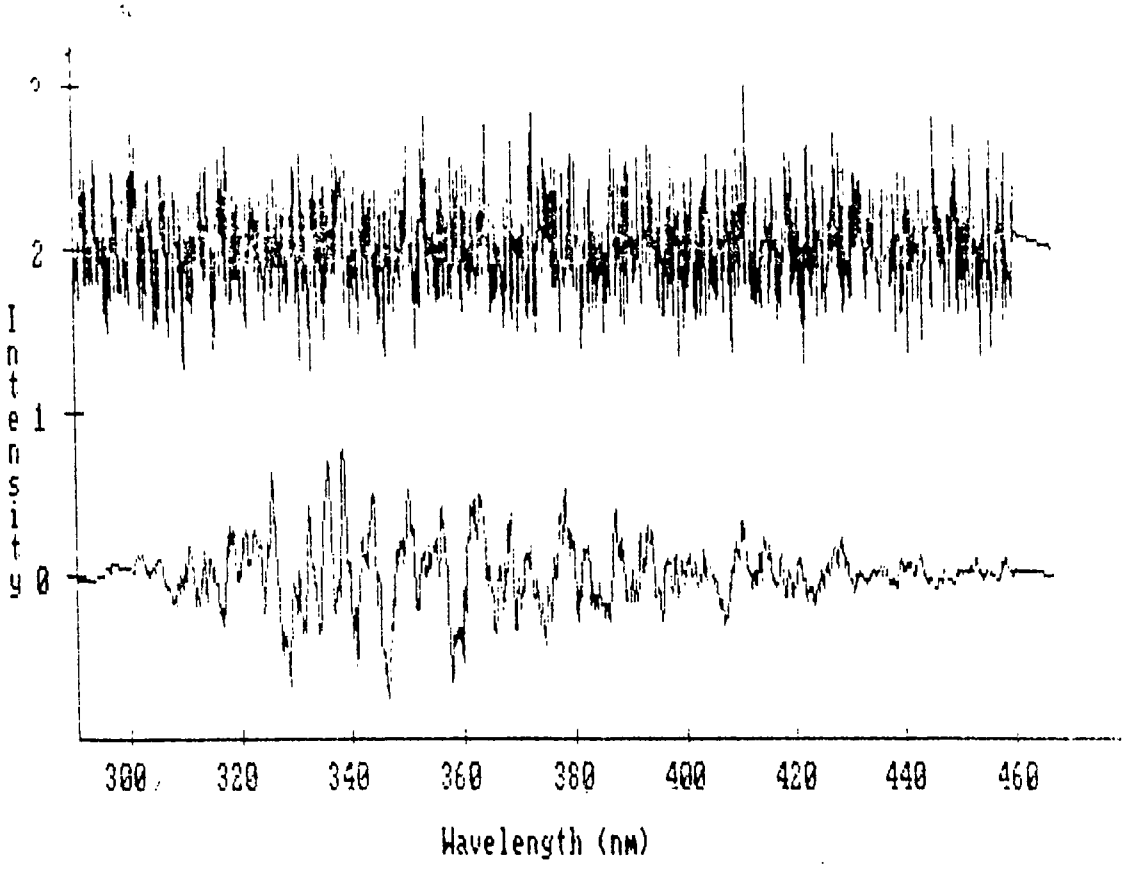


Fig.3.21. Laser intensity noise and spectrum noise.

If the spectrum is corrected for intensity fluctuations and then smoothed, the correlation coefficient decreases to 0.169.

b) Autocorrelation of the noise

As seen on Figure 3.21, the noise on the measurements looks very different than the laser intensity noise. The laser noise effectively seems to take values at random, every value is independent from the preceding one. The noise which appears on the fluorescence measurements, varies much more smoothly, often taking many positive values before it changes sign. Table 3.5 effectively shows that this noise is highly correlated.

A very high autocorrelation exists for wavelength shift less than a nano-

Shift (nm)	0.1	0.2	0.3	0.4	0.5
Correlation	0.973	0.919	0.857	0.795	0.734
Shift (nm)	1	2	3	4	5
Correlation	0.371	-0.259	-0.467	-0.244	-0.057

Table 3.5. Autocorrelation of the spectrum noise (computed between 340 and 360 nm).

meter and the correlation stays strong up to 3 or 3.5 nanometers. The same computation can be made with a spectrum corrected for the intensity variation. Table 3.6 shows the autocorrelation of the noise in this case.

Shift (nm)	0.1	0.2	0.3	0.4	0.5
Correlation	0.809	0.717	0.634	0.565	0.519
Shift (nm)	1	2	3	4	5
Correlation	0.341	-0.173	-0.329	-0.187	-0.009

Table 3.6. Autocorrelation of a corrected spectrum noise (computed between 340 and 360 nm).

A significant decrease is observed. However one can wonder why the noise is highly correlated. Dark-current noise, flicker noise and photon noise are theoretically white noises and the resulting noise on the fluorescent measurements is supposed to be uncorrelated. The autocorrelation is due to an electronic low-pass filter which decreases the noise. From the preceding calculations, one would expect 18% of photon noise plus 4% of flicker noise which results in 22% of noise. The filter reduces it to 8%. But this smoothing correlates the noise and decreases the effect of the laser intensity correction.

c) Signal to noise ratio improvement

One can try to reduce the intensity of the different noise. The photon

noise relative intensity can be reduced by acting on the parameters of the formula 1 (page 54). Intensity and integration time are the two influent parameters which allow to decrease the photon noise. Since the photon noise is proportional to the square root of the intensity, the signal to noise ratio is inversely proportional to the square root of the intensity. Therefore the intensity should be increased. It can hardly be increased by increasing the excitation light intensity since it is already very high. An increase of the volume of fluorescence collection can also increase the intensity but is not easy to realize. An easy way to increase the intensity consists in removing the filters often placed on the fluorescent beam. If the photomultiplier can stand the increase of intensity, the removal of these filters would decrease the photon noise by 3 to 10 folds on many measurements. The integration time is hard to increase on this setup. An interesting way would be to use a photodiode array for the light detection. In this type of detector an array of a thousand of diodes is used as detector. The light is diffracted by a grating and each diode measures the light at a single wavelength. At each pulse every wavelength of the fluorescent light is measured instead of a single one in the monochromator-photomultiplier system. For a same observation time, the signal to noise ratio is multiplied by the square root of the number of detectors.

3.3.6 The measurement procedure

Due to the multiple reasons explained before in the paragraph 3.3.1 the

measurements are hardly reproducible from one day to another. Nevertheless an attempt has been made to define a set of reference parameters which were used during every experiments if possible. All the measurements made when these reference settings were used are more or less comparable. Many parameters of the experiments could not be tuned (intensity of the laser, etc.) but the use of the same main settings for all the experiments allows to improve the reproducibility. These settings are :

- ⋆ Monochromator slit width : 500 μm
- ⋆ Photodiode sensitivity : 100 mV
- ⋆ Photomultiplier tube : 10 mV
- ⋆ Boxcar integration gate : 30 ns
- ⋆ No Boxcar averaging : position 'last'
- ⋆ Scanning speed : 10 $\mathring{A}s^{-1}$

The laser power that was usually available gave a 3 V reading on the photodiode when a 4.0 and a 1.5 neutral density filters were placed in front of it. As said before the sample in the cuvette was changed before each measurement so that no degradation could interfere with the measurements.

Chapter 4

SpecProc : A Program For Spectrum Analysis

4.1 Purpose

The laser induced fluorescence measurements made at the National Bureau of Standards produce a lot of data. Sometimes 3000 different wavelengths are used to record a spectrum which means that a spectrum is represented by 9000 real numbers plus some numbers to characterize the measurement. The format of the files used to store the data is explained in appendix C. SpecProc was written to analyze the spectral data recorded at the National Bureau of Standards. It is a menu-driven program with good graphic capabilities which allows the user to see the results of his analysis at each step. The program consists of 3500 lines of TurboPascal and is listed in appendix D. TurboPascal has been chosen for this application because it is one of the most convenient, most powerful and fastest languages presently available on the IBM PC. In the next section, the main functions of the program as well as the interaction with the user will be

described and, in the subsequent section, the internal structure and the variables will be presented.

4.2 Main features

When the program is started, it reads the file `specproc.dat` and displays the main menu which offers :

- Load a spectrum from a disk
- Linear combination of spectra
- Smoothing
- Statistical analysis
- List, save and delete
- Quit

Each one of these possibilities can be selected by typing a single letter. The six procedures defined in the main menu are described below.

4.2.1 Load a spectrum

The first thing to do is to load some spectra. The program works only with spectra loaded in main memory. This mode of work gives very fast access time to the data after they are loaded. Depending on the main memory of the computer and the size of the spectra, a variable number of spectra can be loaded. Computers with 256K of memory are too small to work with big spectra. Only five to ten spectra can be maintain in memory

and then the program may crash in some procedures. On computers with 500K of main memory, the user has enough space for all applications. Ten to fifteen spectra can be kept in memory without any limitations on the work of program.

When one enters the spectra loading menu, two lines are displayed showing the default directory and the type of loading. Type of loading means with or without a normalization of the fluorescence signal by the laser intensity. `Alt T` changes this type and `Alt D` changes the default directory. Then the user is asked for a file name. The computer looks for this file with the following assumptions :

* If no directory is given by the user in the file name then the default directory is assumed.
* If no ending of the file name is given, '`.dat`' is assumed.

If the computer does not find the file, it asks for another name. If it does, then it asks for a local name. Indeed each spectrum is given a name when it is loaded. This name is its only identification inside the program, and it will always be accessed by this name. If no name is given by the user, the file name is taken as default name. If the user gives a name already existing, the computer asks another name.

If the file to be loaded has never been loaded, the program will ask the user to input all the information available about this measurement. The program creates then a file with the same name as the data file but which

ends with '.tex' rather than '.dat'. This file contains all of the details about the data that the user wants to give. The purpose of this file is to store all the information that are not included in the data file itself but which are important for the analysis. The user can later consult this file and modify it. When the user is finished with this step, the program loads the spectrum. It should be noticed that spectra of any length can be loaded as long as they respect the format given in appendix C and that there is enough memory available to store them. The program asks the user if he wants to load another spectrum. If so the program goes back to the beginning of the loading procedure; if not it goes back to the main menu.

4.2.2 Linear combination

This procedure allows the user to create new spectra by linear combination of existing ones. The user is asked spectrum names and coefficients until he answers by a blank at the spectrum name question. The computer performs the linear combination and asks the user a name to give to the new spectrum. If the user does not give a name, a default value combn where n is a number large enough, is taken. The particular interest of this linear combination procedure is that it allows one to add spectra which do not have the same format. Indeed spectra which do not start at the same wavelength and which are not recorded with the same wavelength step, can be added. In these two cases, the new spectrum starts at the higher of the

two beginning wavelengths and has the larger increment. In case of different increments, values of the spectrum of smaller increment are averaged over the larger increment in the addition process.

After having given a name to the new spectrum, the program asks if the user wants to make another linear combination. If so the program returns to the beginning of the linear combination procedure; otherwise it goes back to the main menu.

4.2.3 Smoothing

The smoothing is done using a Fourier transform. First the user is asked for the name of the spectrum to smooth and for the wavelength interval on which the smoothing should be performed. Next the program asks if the user wants to smooth the fluorescence signal or the laser intensity. Only one at the time can be smoothed. Then the program computes the Fourier transform. The program chooses the number of points needed and transforms the signal using a fast Fourier transform of the TurboPascal Numerical Methods Toolbox. The real part and the imaginary part of the transform are displayed on the screen. By typing F10, the user enters a menu where he can redefine the axis of the plot. The program asks then the frequency at which the user wants to cut the transform for the low pass filter. The reverse transformation is performed on the transform with all the frequencies higher than the one given by the user set to zero. The result is displayed, and the computer asks if the user is satisfied. If not, the

program goes back to the Fourier transform display and asks for another frequency. If yes, the program asks if the user wants to save the Fourier transform and the smoothed spectrum. Each time the user answers yes, he is asked a name for the saved spectrum. The last question of the computer asks the user if he wants to smooth another spectrum. Depending on the answer, the program goes back to the beginning of the smoothing procedure or to the main menu.

4.2.4 Statistical analysis

The statistical analysis procedure starts by asking the user the name of the spectrum to analyze. After the user sets the wavelength interval of analysis, the program gives the mean value and variance of both the fluorescent signal and the laser intensity over the preset interval. The program asks then the name of another spectrum with which correlations will be studied. If the user answers by a blank, the computer skips this step. If the user answers by the name of a spectrum, the program asks a wavelength shift which defaults to zero. The correlation coefficients between the two fluorescent signals and the two laser intensities are computed and displayed as well as the proportionality constant between the two fluorescent signals. The program directly returns to the correlation step by asking the name of a spectrum with which to compute the correlation. A blank answer takes the user to the last question of the statistical analysis procedure. Depending on the answer, the program goes back to the beginning of the

procedure or to the main menu.

4.2.5 List, save and delete

This part of the program takes care of the management of the spectra list. When the user enters this procedure, the list of all the spectra present in memory is displayed on the screen. The name, size and title of the spectra is listed. Using the arrow keys the user can choose one of the four possibilities offered : List, Save, Delete, Exit. The first function allows one to type the text file associated with a selected spectrum when it exists (cf. 4.2.1). Save and Delete work on the same principle. The user can select some spectra in the list by using the arrow keys and the return key. When he has selected all the spectra he wants (the selected spectra are preceded by a star), he presses the escape key and all the selected spectra are deleted or saved on disk. If some spectra are to be saved, the computer asks for a file name. In the two cases, Save or Delete, the program directly goes back to the main menu at the end of the procedure. The Exit function just returns to the main menu.

4.2.6 Quit

The Quit function first asks if the user is sure he does not wants to save anything else, and then, if the user answers no, saves some settings of the program like default directory, default values for the wavelengths used in the statistical analysis as well as in smoothing.

4.2.7 Graphic capabilities

The most interesting feature of the SpecProc program is its graphic mode. In any part of the program, the user can access the graphic mode by simply pressing the escape key. In the graphic mode, up to nine curves can be plotted on the screen. Each of these curves is in fact a set of attributes associated with a key 1 to 9. Any loaded spectrum can be associated with one of the nine curves. The function keys F1,..,F9 give access to nine menus in which the user can specify all the parameters of the corresponding curve. In these menus the user can :

* ⋆ change the spectrum associated with the curve
* ⋆ turn the curve on or off
* ⋆ turn the laser intensity curve on or off
* ⋆ define a title
* ⋆ shift the curve along the X or Y axis
* ⋆ scale the curve along the X or Y axis
* ⋆ scale the laser intensity curve

As soon as one enters the graphic mode, all the curves which are "on" are plotted. Then the user can access the parameters menus through the keys F1..F9 , change the axis through the key F10, or return to the main menu through the escape key. The F10 key gives access to a menu where the user can define the maximum values of X and Y, the label of each axis as well as the title of the plot. When the curves are displayed on the screen,

the user can switch any curve on and off by using the corresponding key 1 to 9. Any letter key turn all the curves off.

4.2.8 Advanced operation

In order to prevent an obliged return to the main menu each time the user finishes a function, some direct moves from any procedure to any other are allowed. The main questions asked by the computer in every procedure accept other answers than the ones offered. They are :

* `Esc` to go to the graphic mode
* `F1..F9` to go to the curve parameter definition
* `F10` to go to the axis definition menu
* `Alt s` or `Alt 1` to go to the save-list procedure
* `Alt q` to quit
* `Alt 1` to go to the main menu
* `Alt 2` to go to the loading procedure
* `Alt 3` to go to the linear combination procedure
* `Alt 4` to go to the smoothing procedure
* `Alt 5` to go to the statistical analysis

Thus an advanced user can quickly access a desired function.

4.3 Internal structure

4.3.1 The variables

Many new types of variables have been defined in order to manipulate easily the spectra. The main data structure used is a linked chain of spectra. Each spectrum is defined as a record possessing a pointer to another spectrum. This structure allows one to load as many spectra as there is space in the memory. For any new spectrum to be loaded, a new record is dynamically created and added to the list in first position. The spectrum type is defined as a record containing the name of the source file from which it comes, the number of measurements, the minimum wavelength, the wavelength increment, two pointers to data arrays, a record containing information used for the plots and a pointer to another spectrum. The fluorescence spectrum and the laser signal are both represented in a linked chain of record containing one dimensional real array. The choice of a linked chain was directed by the need to work with data of unknown size.

Some other special types of variables are also defined for the menu system and for the axis definition. All the menus appearing in the program are done by a procedure called Menu1 which input is a variable of type menutab. This data type is a one dimensional array of records. Each of these record contains a line of text to be typed in the menu, a code of the type of answer expected, a boolean, a real and a string. The answer expected by the computer can be of four types : real, boolean, string of characters or nothing. The function Menu1 displays the menu given as input and stores the answers of any of the four types in the record and

returns to the calling program.

4.3.2 The structure

The program consists of a main program and nine main procedures. The only function of the main program is to go from one of the nine procedures to another. A character variable called MainCh, which is an output parameter of every main subprogram is used by the main program to select a procedure or to stop the program. This structure allows to access directly any one of the nine procedures. The hierarchical structure between the main menu and the other subprograms is done by giving to MainCh the value corresponding to the main menu at the end of each subprogram. The nine procedures are the following :

- ⋆ Specplot : opens a graphic screen and plots the axis and the curves. The procedure plotcurve is called for each plot.
- ⋆ Save : lists, saves and deletes the spectra. It calls the procedures specsave, specdel, listdel and listsave.
- ⋆ Characterize : displays the menu of the settings for a plot and allows to change it. The procedure menu1 is called for the menu display.
- ⋆ Axisdef : displays the menu of the axis settings and allows to change it. It calls menu1.
- ⋆ Define : loads a spectrum. Mkspec is called to load a spectrum, specre to insert the new spectrum in the list and curveaffect to assign it to one of the nine curve.

* Lincomb : performs the linear combinations. It calls the procedures mult and add. The procedure "Add" calls "convert" to make spectra with the same format from spectra with different ones.
* Smoothing : performs the low-pass filter. Smfft is called to filter the spectrum. A Borland software RealFFT is used to actually compute the Fourier transform.
* Stat : performs the statistical analysis. mv and correl are called to compute the mean, variance and correlation coefficient. Correl calls convert when two spectra have different format.
* Mainmenu : displays the main menu and allows the selection of a function.

Chapter 5

Deconvolution Techniques

5.1 Modelization of fluorescence

The relationship between concentration and fluorescence intensity has been longly studied. The simplest and most commonly used relation is the Beer-Lambert law.

5.1.1 The Beer-Lambert law

When a monochromatic light passes through a solution of concentration c and molar absorptivity ϵ, it is absorbed proportionally to its intensity. If I is the intensity of the light beam, then the light absorbed in a slice of thickness dx is given by :

$$dI = -\epsilon c I dx$$

The integration gives :

$$I = I_0 \exp(-\epsilon c l)$$

where l is the path length, I_0 the initial intensity and, I the transmitted

intensity. The light absorbed by the solution is therefore :

$$I_0 - I = I_0(1 - \exp(-\epsilon cl))$$

The fluorescence emitted is related to the absorbed light by a factor called the fluorescence quantum yield or efficiency :

$$\Phi = \frac{number\ of\ photons\ emitted}{number\ of\ photons\ absorbed}$$

Therefore the total fluorescence is :

$$F = I_0(1 - \exp(-\epsilon cl))\Phi$$

For low value of the absorbance, a first order development is used :

$$F = I_0 \epsilon cl \Phi$$

The linear equation obtained is called the Beer-Lambert law. It is used very often in the estimation techniques because it is linear and easy to compute. The validity of this equation is limited to absorbance of the order of 5%. For higher absorbance the behavior is not linear any more. The first order approximation is not valid and the model itself is not very useful because of the spatial variations of the fluorescence intensity.

As shown on Figure 5.1, the fluorescence intensity varies tremendously inside the cuvette for high absorbances. The Beer-Lambert law estimates the total fluorescence emitted which is not the quantity measured. In fact, the measurements depend greatly of the position of the detector at high concentration as shown in Figure 5.2.

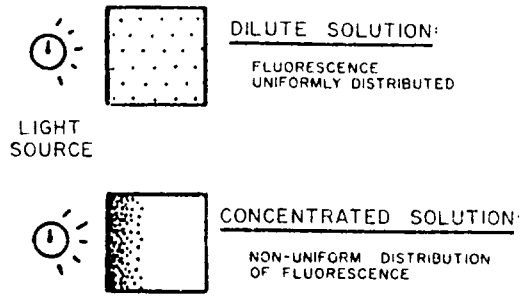


Fig.5.1. Repartition of the fluorescence emission (from Undenfriend [1962]).

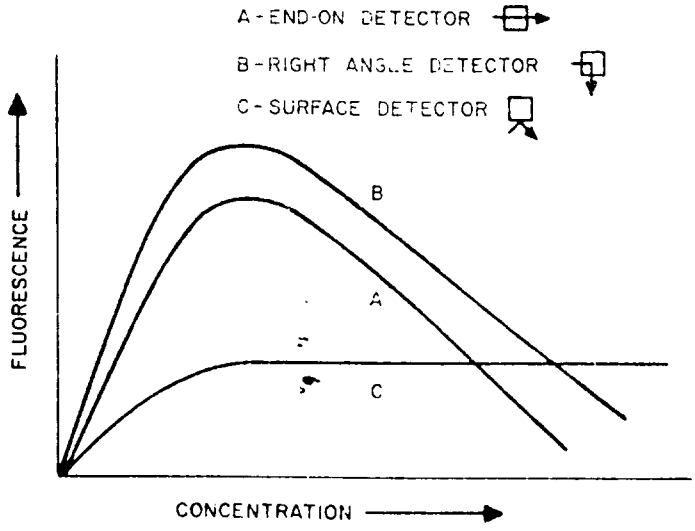


Fig.5.2. Effect of the detector position (from Undenfriend [1962]).

When multiple components absorb and fluoresce, one usually assumes that the components do not interact. Therefore, each component should absorb and reemit light as if it was alone in solution. The spectrum of the mixture is then the sum of the spectra of the components. This approximation is sometimes true but often interactions occur between the components.

Figure 5.3 shows the spectrum of a mixture of tryptophan and tyrosine and the sum of the pure components spectra. The assumption of linearity indicates that both are identical.

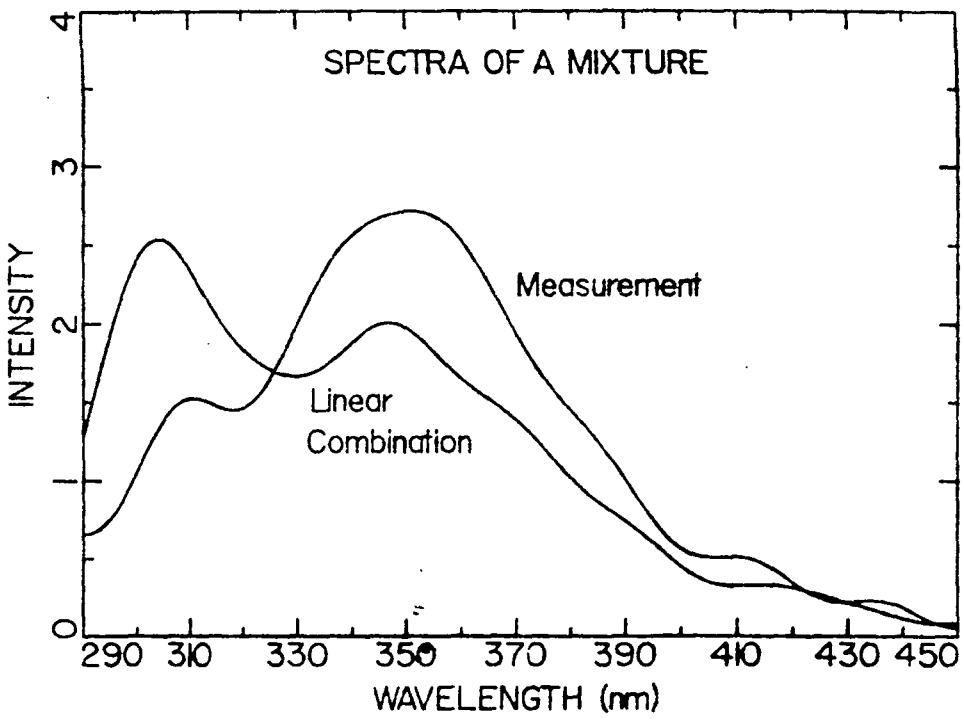


Fig.5.3. Mixture and sum of the pure components spectra.

The difference of shape between the two spectra is striking. The interaction can be due to simple optical effects or to more complex energy transfer. The optical interaction between the components can be due to reabsorption of the fluorescent light by others components or difference of absorptivity between components. This kind of effects can be modelized.

5.1.2 A more sophisticated model

To take into consideration possible reabsorption of the fluorescence signal, Simmons and Wang [87] have developed a model of fluorescence in multicomponent mixture. This model requires the knowledge of the entire absorption and fluorescence spectra of the components in order to estimate the fluorescence spectrum of the mixture. The detection system they consider, is a NADH probe which measures the back-scattering fluorescence light using a circular detector. For n components excited by a monochromatic light at λ_{ex}, they found that the fluorescence signal received by their detector can be estimated by the formula :

$$F^{\lambda_{em}} = \frac{I_0 \sum_{i=1}^{n} (f_i a_i^{\lambda_{ex}} c_i)}{\sum_{i=1}^{n} (a_i^{\lambda_{ex}} + a_i^{\lambda_{em}}) c_i + S}$$

where $f^{\lambda_{em}}$ is the fluorescence at the wavelength λ_{em},

I_0 is the intensity of the excitation light,

$a_i^{\lambda_{ex}}$ and $a_i^{\lambda_{em}}$ are the absorption coefficients at λ_{ex} and λ_{em} respectively,

c_i are the concentrations,

and S is a factor that theoretically varies but is assumed constant due to its small variations.

The fact that S is constant and that the fermenter is deep enough to neglect a term exponential in the path length, are the two main assumptions of the model. The result is strongly non linear, but for low concentrations a first order development gives the linear relation commonly used.

The particular configuration of the National Bureau of Standards setup

does not allow to use exactly the same model. The approximations used by Simmons and Wang are not valid any more.

Let us consider a more general approach. If a monochromatic beam at wavelength λ_{ex} and intensity I_0 enters a cuvette or any volume containing a fluorescent mixture, then the intensity of the light can be written in any point (x, y, z) of the volume as follows :

$$I = I_0 g(x, y, z, A^{\lambda_{ex}})$$

Where $A^{\lambda_{ex}}$ is the absorption of the mixture and g a function of the geometry of the equipment. This formula is just an extension of the classical $I = I_0 \exp(-xA^{\lambda_{ex}})$. A component i present in a volume dv at the concentration c_i will absorb some light. Assuming that this absorption is proportional to the intensity, the light absorbed is dI given by :

$$dI = a_i^{\lambda_{ex}} c_i I(x, y, z, A^{\lambda_{ex}}) dv = a_i^{\lambda_{ex}} c_i I_0 g(x, y, z, A^{\lambda_{ex}}) dv$$

The fluorescence emitted as the wavelength λ_{em} is proportional to the absorbed light in a ratio $f_i^{\lambda_{em}}$:

$$dF_i^{\lambda_{em}} = f_i^{\lambda_{em}} a_i^{\lambda_{ex}} c_i I_0 g(x, y, z, A^{\lambda_{ex}}) dv$$

This fluorescent light is emitted at the point (x, y, z) in every direction. Only a small part of the fluorescence will reach the detector. The amount of fluorescent light reaching the detector depends of the geometry of the

setup as well as the absorbance of the solution at the emission wavelength. If h represents the ratio of fluorescent light reaching the detector, one can write :

$$dF_i^{\lambda_{em}} = f_i^{\lambda_{em}} a_i^{\lambda_{ex}} c_i I_0 g(x,y,z,A^{\lambda_{ex}}) h(x,y,z,A^{\lambda_{em}}) dv$$

The integration over the volume of fluorescence gives :

$$F_i^{\lambda_{em}} = \int_V f_i^{\lambda_{em}} a_i^{\lambda_{ex}} c_i I_0 g(x,y,z,A^{\lambda_{ex}}) h(x,y,z,A^{\lambda_{em}}) dv$$

Or :

$$F_i^{\lambda_{em}} = f_i^{\lambda_{em}} a_i^{\lambda_{ex}} c_i I_0 \int_V g(x,y,z,A^{\lambda_{ex}}) h(x,y,z,A^{\lambda_{em}}) dv$$

A new function Γ can be defined as :

$$\Gamma(A^{\lambda_{ex}}, A^{\lambda_{em}}) = \int_V g(x,y,z,A^{\lambda_{ex}}) h(x,y,z,A^{\lambda_{em}}) dv$$

This function Γ is a function depending only of the geometry of the measurement system. It is independent of the wavelengths, components, etc.. It is a characteristic of the setup. It can be measured or calculated in simple cases as it is explained below. It is supposed known in the analysis presented now. The fluorescence of the component i gives a signal :

$$F_i^{\lambda_{em}} = I_0 f_i^{\lambda_{em}} a_i^{\lambda_{ex}} c_i \Gamma(A^{\lambda_{ex}}, A^{\lambda_{em}})$$

The total fluorescence signal is obtained by adding the fluorescence of the n components :

$$F^{\lambda_{em}} = I_0 \Gamma(A^{\lambda_{ex}}, A^{\lambda_{em}}) \sum_{i=1}^{n} f_i^{\lambda_{em}} a_i^{\lambda_{ex}} c_i$$

This can be rewritten :

$$\frac{F^{\lambda_{em}}}{\Gamma(A^{\lambda_{ex}}, A^{\lambda_{em}})} = I_0 \sum_{i=1}^{n} f_i^{\lambda_{em}} a_i^{\lambda_{ex}} c_i$$

On this equation, the non linear dependence of the fluorescence signal from the concentration is clear since the absorbances are function of the concentrations. However this formula proves that the measurement of the absorbance of the solution would allow to get a corrected spectrum linearly dependent of the concentrations. In particular, if the pure component fluorescence and absorbance spectra at concentration c_{i_0} are known, one can write :

$$\frac{S_i^{\lambda_{em}}}{\Gamma(A_{i_0}^{\lambda_{ex}}, A_{i_0}^{\lambda_{em}})} = I_0 f_i^{\lambda_{em}} a_i^{\lambda_{ex}} c_{i_0}$$

And

$$\frac{F^{\lambda_{em}}}{\Gamma(A^{\lambda_{ex}}, A^{\lambda_{em}})} = I_0 \sum_{i=1}^{n} \frac{c_i}{c_{i_0}} \frac{S_i^{\lambda_{em}}}{\Gamma(A_{i_0}^{\lambda_{ex}}, A_{i_0}^{\lambda_{em}})} \qquad 2$$

A linear dependence between the spectra divided by Γ is shown by this model. This model is very general and is applicable to any setup. The only limitations are :

- ⋆ Saturation of the absorption if the light is too intense. The proportionality between the excitation light and absorbed is not valid any more.
- ⋆ Reemission of the absorbed fluorescence.
- ⋆ Changes in the optical properties of the components due to chemical or physical effects.

This model shows the dependence of the fluorescence signal from the absorption of the mixture at the excitation and emission wavelengths. One way to solve the preceding equation would be to express the absorption as a function of the concentrations of the fluorescent components and resolve it by a non linear technique. This method would be complex and also will fail in analyzing mixture containing absorbing species which do not fluoresce. A direct measurement of the absorbance would give a much more accurate and simpler system. The measurement of the absorbance at the excitation wavelength is very easy since it can be measured directly with the excitation light. The simple addition of a photodiode to the setup can give this measurement. The correction of the spectra for absorbance can greatly improve the linearity of measurements as it will be shown in Chapter 6. The measurement of the complete absorbance spectrum is much more complex but would be a great improvement for the fluorescence signal analysis.

The last difficulty of the model is the evaluation of the function Γ. It is generally hard to calculate. The volume of fluorescence, the ratio of fluorescence collected by the detector are usually difficult to estimate accurately. However an experimental estimation is very easy. If three components have the following properties :

* the first one absorbs at λ_{ex} and fluoresce at λ_{em}

* the second absorbs at λ_{ex} but do not fluoresce

$\star$ the third absorbs at λ_{em} but not at λ_{ex}

One can measure Γ by measuring the fluorescence of three component mixtures. Indeed if some mixtures are prepared with a constant concentration of component 1 and different concentrations of components 2 and 3, one can write :

$$F^{\lambda_{em}} = I_0 f_1^{\lambda_{em}} a_1^{\lambda_{ex}} c_1 \Gamma(A^{\lambda_{ex}}, A^{\lambda_{em}})$$

The variation of the concentrations of components 2 and 3 allows to vary the absorbances. Since the concentration of the components 1 is constant, the fluorescence signal is proportional to Γ. In the preceding equation (2), all the terms are divided by Γ therefore any function proportional to Γ can be used as well in this equation. In particular the fluorescence signal measured can be used as Γ in this equation. The calibration of Γ for any setup is therefore be very easy. The application of this model to the National Bureau of Standards setup is now considered.

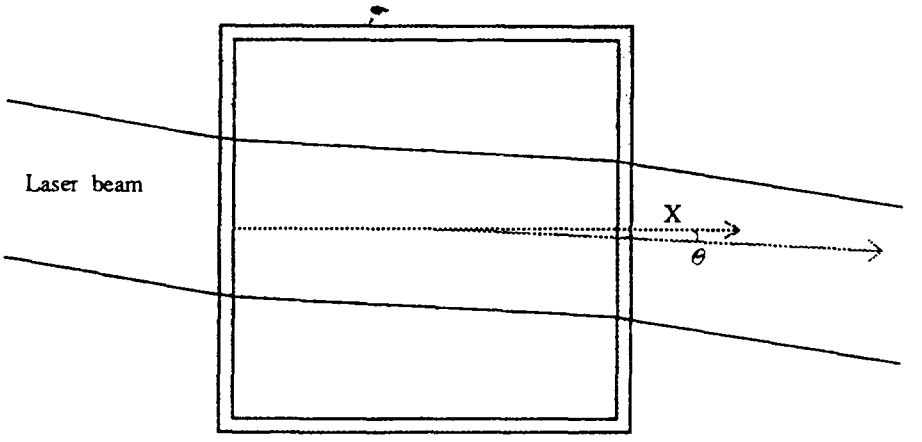


Fig.5.4. Reference axis.

Using the convention defined in Figure 5.4, the absorption law gives :

$$I = I_0 \exp(-\frac{x}{\cos\theta} A^{\lambda_{ex}})$$

Therefore the function g defined in the preceding section is :

$$g(x, y, z, A^{\lambda_{ex}}) = \exp(-\frac{x}{\cos\theta} A^{\lambda_{ex}})$$

The amount of fluorescence that reaches the detector depends of the position of the fluorescing point with respect to the geometry of the detection system. A general way to define this proportion of fluorescence is to use a function $\psi(x, r)$ where x is the coordinate already defined and r the distance to the optic axis of the system. The fluorescence signal can be reabsorbed by the solution itself. Since the solid angle of light collection is very small, one can assume that the path length to exit the cuvette is x. The absorption of the signal is proportional to $\exp(-xA^{\lambda_{em}})$. Therefore the function h is :

$$h(x, y, z, A^{\lambda_{em}}) = \psi(x, r) \exp(-xA^{\lambda_{em}})$$

And Γ is given by :

$$\Gamma(A^{\lambda_{ex}}, A^{\lambda_{em}}) = \int_V \psi(x, r) \exp(-x(\frac{A^{\lambda_{ex}}}{\cos\theta} + A^{\lambda_{em}}))$$

Therefore one can see that the function Γ is in fact a function of one variable only $\frac{A^{\lambda_{ex}}}{\cos\theta} + A^{\lambda_{em}}$. The calibration of the function is in this case very easy.

In order to check the model on an application in Chapter 6, an analytical formula of Γ is needed. $\psi(x, r)$ and V, the volume of fluorescence, are

very difficult to estimate. The simplest model is chosen. $\psi(x,r)$ is assumed constant and V is assumed to be a cylinder in the direction of the axis x. The equation is much simpler with these two assumptions :

$$\Gamma(A^{\lambda_{ex}}, A^{\lambda_{em}}) = k \int_{x=0}^{1} \exp(-x(\frac{A^{\lambda_{ex}}}{\cos\theta} + A^{\lambda_{em}}))$$

Where k is a constant. The integration gives :

$$\Gamma(A^{\lambda_{ex}}, A^{\lambda_{em}}) = \frac{k}{\frac{A^{\lambda_{ex}}}{\cos\theta} + A^{\lambda_{em}}}(1 - \exp(\frac{A^{\lambda_{ex}}}{\cos\theta} + A^{\lambda_{em}}))$$

This value of Γ can be used in the model equation 2 which is then completely determined. An estimation of the validity of this model is given in Chapter 6.

5.2 Multivariate analysis techniques

In order to resolve a mixture, many multivariate analysis techniques have been considered. Three of them are presented below. The problem addressed by these methods can be stated as follows :

- ⋆ Estimate the concentration of the main fluorescent compounds in a mixture.
- ⋆ All the fluorescent compounds are known.
- ⋆ The spectra of the pure components are known as well as the spectra of some reference mixtures if needed.

These assumptions are reasonable for the analysis of fluorescence in fermentation processes. The important fluorophores are known and often well

studied. It is not a problem to get the spectra of the pure components as well as some mixtures spectra.

5.2.1 Multiple linear regression

Multiple linear regression is the most classical way to analyze experiments when one variable y is related to a number of x variables. The model is written as follows :

$$y = Xb + e$$

In the present study, y is a column vector of length n representing the spectrum of the mixture, X is a $n \times m$ matrix which columns represent the pure components spectra. b is a column vector of m coefficients. These coefficients are the ones of the linear combination of pure components spectra :

$$y_i = b_1 x_{i1} + b_2 x_{i2} + \cdots + b_m x_{im} + e$$

e is a residual vector. Using the hypothesis of linearity for the mixture spectrum, the b_i should be equal to the concentration of the component i in the mixture divided by its concentration in the pure sample x_i. Since the spectra are defined with up to 3000 points and only 2 or 3 components are considered, $m < n$ and there is no exact solution. But one can get a solution by minimizing the norm of the residual vector e. The problem is then an optimization problem :

$$\min_b \sum_{i=1}^{n} e_i^2$$

$$e = y - Xb$$

The solution of this least squares problem is well known :

$$b = (X'X)^{-1}X'y$$

The limitations of this method are the assumption that the matrix X is exact and the inversion of $X'X$. Indeed as soon as m gets large some collinearity problems appear and the matrix inversion is impossible. To overcome the collinearity problem, some rank reducing methods such as principal component regression or partial least squares regression are needed. The two methods presented below are slightly different from multiple linear regression in the way they use the data. They use a set of reference measurements rather than only the pure components spectra. These reference measurements are done with mixtures of known concentration of each component. The pure components spectra were the reference measurements of multiple linear regression but no mixture spectra could be used in this method due to collinearity problems which would appear. Since the other methods do not have this problem, the use of mixtures data to build the model is possible. Some non linear effects can be taken into account by these two methods due to the use of mixture data.

New notations are used for the two next methods. Now Y is a matrix $n \times p$, each line of which is a vector of fluorophore concentrations in a reference mixture. X is a $n \times m$ matrix which lines are the fluorescence spectra of reference mixtures. For the two methods, the variables are usu-

ally mean-centered and scaled to unit variance before processing. Figure 5.5 shows the effects of the two operations.

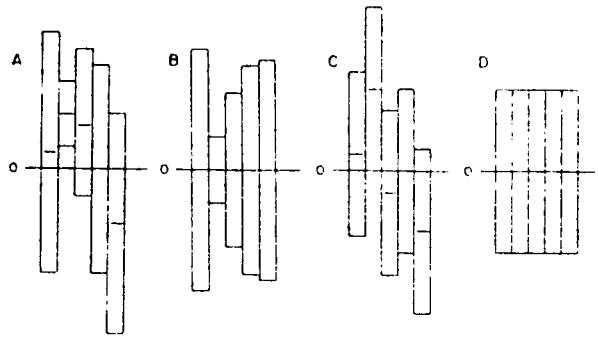


Fig.5.5. Effects of mean-centering and variance scaling. The data for each variable are represented by a variance bar and its center.(A)Most raw data look like this.(B) The result after mean centering only. (C)The result after variance scaling.(D) The result after mean centering and variance scaling (from Geladi and Kowalski [1986]).

Mean-centered variables are obtained by subtracting to every data point $x_{i,j}$ or $y_{i,j}$ the mean of its column. The unit variance scaling is obtained by dividing every data point by the standard deviation of its column. These two processes improve the conditioning of the data. If some variables are less important or significant than the others, one can reduce the variance of the corresponding column by multiplying it by a weighting factor.

5.2.2 Principal component regression

In principal component regression, a reduction of the rank of the system is done using principal component analysis. The idea consists in decompos-

ing the matrix X in a product of two matrices T and P of lower rank. The decomposition is done such that the maximum of the variance of X is explain by the decomposition of X in T and P. The model can be expressed by :

$$X = TP' + E$$

where T is a $n \times a$ matrix, P a $m \times a$ matrix and E a $n \times m$ matrix. a is the number of vectors (principal components) used in the model and can be determined by cross-validation as explained below. In order to explain the variance of X as well as possible, the column vectors of T are the eigenvectors found in the singular value decomposition of X. The algorithm to recursively compute the matrices T and P is the following :

(1) take a vector x_j from X and call it t_h : $t_h = x_j$

(2) calculate p'_h : $p'_h = t'_h X / t'_h t_h$

(3) normalize p'_h to length 1 : $p'_{h_{new}} = p'_{h_{old}} / \| p'_{h_{old}} \|$

(4) calculate t_h : $t_h = X p_h / p'_h p_h$

(5) compare the t_h used in step 2 with that obtained in step 4. If they are the same, stop (the iteration has converged). If they still differ, go to step 2.

(6) compute the residual E_h : $E_h = X - t_h p'_h$

(7) return to step 1 with $X = E_h$ and $h = h + 1$

The number of components one keeps in the model is very important. Figure 5.6 shows the influence of the number of principal components on

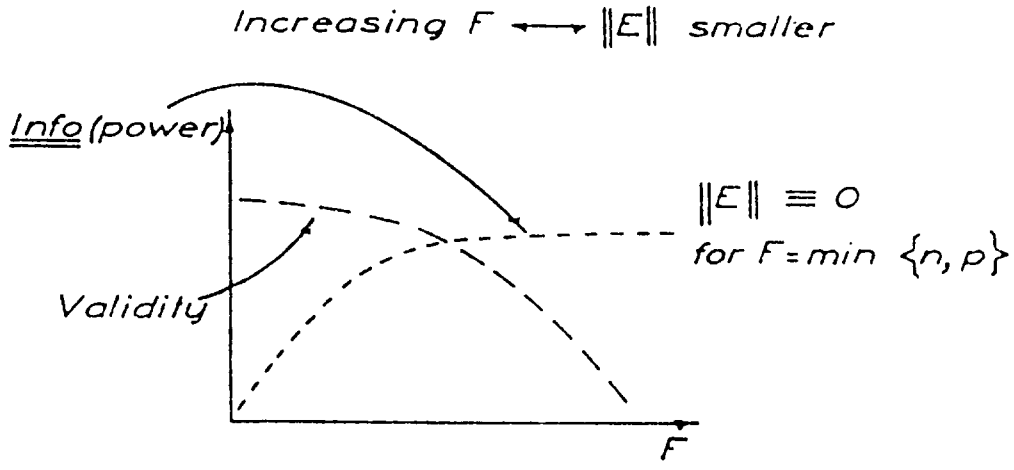


Fig.5.6. Power of the model and validity of the results in function of F the number of components (from NATO Proceedings [1983]).

the residual as well as the prediction capability of the model.

An accepted method to determine the number of principal components useful in the analysis is cross-validation. This method consists in quantifying the predictive power of the model with different number of principal components and therefore find the number of components optimal for prediction. It has been described by Wold[1978], Eastment and Krzanowski[1982] and is realized as follows :

A few data elements are kept out from the data matrix X each round, and the principal component model with different number of factors is fitted to the remaining data. So many rounds are made as needed to keep each data element out once and only once. The values of the kept out elements are calculated from the resulting models with different a and the deviations between calculated and predicted values are formed. The squares

of these deviations from the separate "rounds" are summed up and given an estimation of the predictive power for each a. The number of components giving the smaller sum of the squares of the deviations is chosen to be the one used in the model.

For the regression, the model obtained, $X = TP'$, is reported in the X-Y relation and gives :

$$Y = TB + E$$

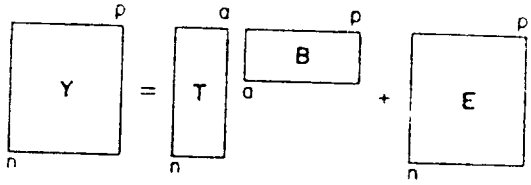


Fig.5.7. Principal component regression model.

Figure 5.7 shows graphically the matrix equation. The solution of this equation is obtained by multiple linear regression :

$$B = (T'T)^{-1}T'Y$$

The matrix inversion is easy since T has a small rank and its column vectors are orthogonal. The concentration estimation for a new spectrum S consists in getting the T values corresponding by multiplying S by P, then the result is multiplied by B to get the raw vector of concentrations.

It can be simply written :

$$y = SPB$$

Principal component regression is much more powerful than multiple linear regression but is very sensitive to outliers in the data. Also, as pointed out by Jolliffe[1982], in applying principal component regression there is a risk that numerically small structures in the X-data which explain Y disappear in the principal component modeling of X. In this case, principal component regression will give a bad prediction of Y. A new method, partial least squares regression tries to overcome these difficulties.

5.3 partial least squares regression

Partial least squares modeling, which has been developed by H. Wold et al.[1982], is a rank reducing technique like principal component analysis. The difference between partial least squares regression and principal component regression is that the Y data are used to determine a decomposition of X in lower rank matrices which is optimal for prediction. The complete model uses three equations, two for the decompositions of X and Y called outer relations, and a relation between the two preceding decompositions called inner relation. The three equations are the following :

$$X = TP' + E$$

$$Y = UQ' + F$$

$$U = TB$$

Where B is a diagonal $a \times a$ matrix and the others are represented in Figure 5.8.

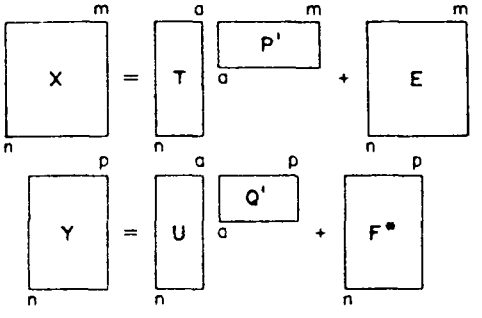


Fig.5.8. The two outer relations of partial least squares model.

In this model, the matrix T is a projection of X as in principal component regression but is calculated both to approximate X and to predict Y. The algorithm is given in Figure 5.9.

The properties of the partial least squares factors are the following :

- $\star$ p'_h and q'_h have unit length
- $\star$ t_h and u_h are centered around zero for each h
- $\star$ w_h and t_h are orthogonal

The algorithm was developed upon more or less intuitive arguments and its theoretical interpretation is not completely clear yet. Many papers try to clarify the theory of this algorithm (Lorber et alt.[1987]). The main part of the algorithm (steps 2 to 7) consists in a power method to find the

The PLS algorithm

It is assumed that X and Y are mean-centered and scaled:

For each component: (1) take u_{start} = some y_j.

In the X block: (2) $w' = u'X/u'u$

(3) $w'_{new} = w'_{old}/\|w'_{old}\|$ (normalization)

(4) $t = Xw/w'w$

In the Y block: (5) $q' = t'Y/t't$

(6) $q'_{new} = q'_{old}/\|q'_{old}\|$ (normalization)

(7) $u = Yq/q'q$

Check convergence: (8) compare the t in step 4 with the one from the preceding iteration. If they are equal (within a certain rounding error) go to step 9, else go to step 2. (If the Y block has only one variable, steps 5—8 can be omitted by putting $q = 1$, and no more iteration is necessary.)

Calculate the X loadings and rescale the scores and weights accordingly:

(9) $p' = t'X/t't$

(10) $p'_{new} = p'_{old}/\|p'_{old}\|$ (normalization)

(11) $t_{new} = t_{old}\|p'_{old}\|$

(12) $w'_{new} = w'_{old}\|p'_{old}\|$

(p', q' and w' should be saved for prediction; t and u can be saved for diagnostic and/or classification purposes).

Find the regression coefficient b for the inner relation:

(13) $b = u't/t't$

Calculation of the residuals. The general outer relation for the X block (for component h) is

$E_h = E_{h-1} - t_h p'_h$; $X = E_0$

The mixed relation for the Y block (for component h) is

$F_h = F_{h-1} - b_h t_h q'_h$; $Y = F_0$

From here, one goes to Step 1 to implement the procedure for the next component. (Note: After the first component, X in steps 2, 4 and 9 and Y in steps 5 and 7 are replaced by their corresponding residual matrices E_h and F_h.)

Fig.5.9. The partial least squares algorithm (from Geladi and Kowalski [1986]).

eigenvectors of $X'YY'X$ and therefore it proves a relation with the singular value decomposition of $X'Y$. The method used to compute the eigenvectors is not completely safe. Indeed the algorithm can diverge in case of very close eigenvalues. However it converges fast for almost all the matrices. The use of Y in the computation of the eigenvectors gives to the method a higher predictive power than principal component regression (Frank and

Kowalski[1984], Lindberg et al.[1983]). The number of components to take into account in the model is computed by using a cross-validation technique. The prediction step is very easy. p, q, w and b from the calibration step have been saved for this purpose. The algorithm for prediction is :

1. $E_0 = X \quad Y = 0$

2. $for \ \ h = 1 \ \ to \ \ a$

$t_h = E_{h-1} w_h$

$E_h = E_{h-1} - t_h p_h'$

$Y = Y + b_h t_h q_h'$

where a is the number of components to be included in the model. Applications have shown the power of partial least squares and its wide range of application. Lindberg and Persson[1983] used it to analyze spectrofluorometric data from mixtures of humic acid and ligninsulfonate, and Frank and Kowalski[1984] applied partial least squares modeling to the prediction of wine quality and geographic origin from chemical measurements.

Chapter 6

Application To Amino Acid Mixtures

Some of the ideas presented in the last chapter have been applied to the amino acids mixtures measurements given in Appendix B.

6.1 Multiple linear regression

The first try to resolve a multiple components mixture has been made using a simple multiple linear regression. Table 6.1 and Figure 6.1 show the results.

Sample	tryptophan Real	tyrosine Real	tryptophan Estimated	tyrosine Estimated
Mix1a	0.5	0.5	0.778	0.239
Mix2a	0.5	0.5	0.747	0.217
Mix3a	0.75	0.25	0.798	0.071
Mix4a	0.75	0.25	0.851	0.097
Mix5a	0.25	0.75	0.517	0.459
Mix6a	0.25	0.75	0.502	0.461
Mix7a	0.1	0.9	0.297	0.698
Mix8a	0.1	0.9	0.273	0.836

Table 6.1. Multilinear estimation of the composition of some tryptophan -tyrosine mixtures. All the concentrations are given in mM. The pure components reference spectra used in the regression are 10^{-3} *M.*

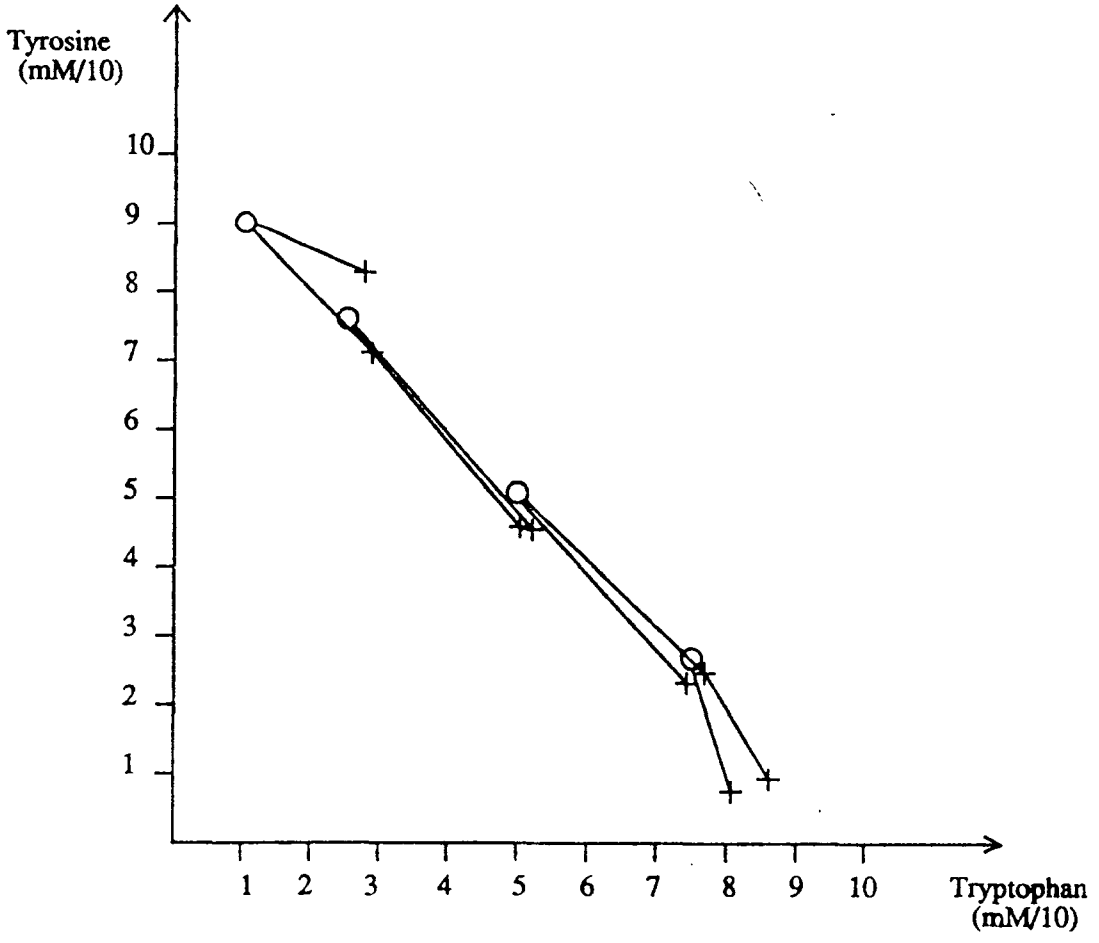


Fig.6.1. Results of a multiple linear regression on amino acids mixtures. The circles represent the true composition and the crosses represent the estimates of the composition.

The interesting part of the results is that they are almost all reproducible within 10 %. The negative part is that the estimates are all wrong, the errors varying between 10% and 300%. The direction of the errors is clearly the same for all the measurements. Therefore the errors are systematic, and it should be possible to figure out the cause of this very poor analysis. Figure 6.2 shows that tyrosine is really less represented in the mixture than it should from the pure component spectra.

The tyrosine peak is higher than the tryptophan one in pure solution but in a mixture the tyrosine peak is much smaller than the tryptophan peak. What can produce this effect ?

The reabsorption of tyrosine fluorescence by tryptophan is a possibility. But in fact the absorption by tryptophan in the 300 nm range is too low for this, only a tenth of its absorbance at 280 nm. Also if tryptophan would absorb the tyrosine fluorescence, then it should reemit approximately 20% of the absorbed light; this would give a tryptophan peak only slightly higher than the one predicted by the linear combination. Therefore another reason than reabsorption should be found. For this purpose, let us consider the model obtain in Chapter 5.

6.2 Correction for absorption

The mixture spectrum can be written as a function of the pure components spectra as follows :

$$\frac{F}{\Gamma(A)} = \frac{c_1}{c_{10}}\frac{F_{10}}{\Gamma(A_{10})} + \frac{c_2}{c_{20}}\frac{F_{20}}{\Gamma(A_{20})}$$

The concentrations c_1 and c_2 can then be deduced by a simple linear regression. As a matter of fact, this is the same regression that was done in the multiple linear regression. The multiple linear regression gives the two coefficients α and β such that :

$$F = \alpha F_{10} + \beta F_{20}$$

Therefore :

$$c_1 = \alpha c_{10} \frac{\Gamma(A_{10})}{\Gamma(A)}$$

$$c_2 = \beta c_{20} \frac{\Gamma(A_{20})}{\Gamma(A)}$$

In order to estimate the performance of this model on the mixture measurements presented before, an estimation of Γ is needed.

The estimation of Γ made in Chapter 5 can be simplified by neglecting the absorption at the emission wavelength. Indeed the absorptivity of the two amino acids is weak in the range of wavelengths considered. Therefore :

$$\Gamma(A) = \frac{k \cos\theta}{A}(1 - \exp(-\frac{A}{\cos\theta}))$$

Since θ is a few degrees, $\cos\theta$ is 1 for all practical purposes. Therefore :

$$\Gamma(A) = \frac{k}{A}(1 - \exp(-A))$$

For the pure components and mixtures measurements presented before, an estimation of the absorbance is possible from the known concentrations and therefore an estimation of Γ is possible. From Lakowicz[1967], one can estimate that the molar absorptivity of tryptophan is 5000 and the one of tyrosine is 1500. The concentrations of both are 10^{-3} M, therefore the value 5 and 1.5 are obtained for the absorptivities. But these values are given for absorptivities computed from decimal logarithm, therefore they should be multiply by 2.3. The absorptivities of the mixtures are calculated by linear combination. Table 6.2 presents these estimations.

tryptophan (10^{-3} mol/l)	tyrosine (10^{-3} mol/l)	Absorbance	$\Gamma(A)$
1	0	11.5	0.087
0	1	3.45	0.28
0.75	0.25	9.49	0.105
0.5	0.5	7.47	0.134
0.25	0.75	5.46	0.182
0.1	0.9	4.26	0.231

Table 6.2. Estimation of the absorbance of some mixtures of tryptophan and tyrosine.

Using these values of Γ, it is possible to compute the concentrations of the mixtures from the results of the multiple linear regression given in Table 6.1. Table 6.3 and Figure 6.2 present the new results.

Sample	tryptophan Real	tyrosine Real	tryptophan Estimated	tyrosine Estimated
Mix1a	0.5	0.5	0.51	0.50
Mix2a	0.5	0.5	0.49	0.45
Mix3a	0.75	0.25	0.66	0.19
Mix4a	0.75	0.25	0.71	0.26
Mix5a	0.25	0.75	0.25	0.71
Mix6a	0.25	0.75	0.24	0.71
Mix7a	0.1	0.9	0.11	0.85
Mix8a	0.1	0.9	0.10	1.01

Table 6.3. Multilinear estimation of the composition of some tryptophan -tyrosine mixtures corrected for absorption. The concentrations are given in mM. The pure components reference spectra used in the regression are 10^{-3} M.

The quality of the results is surprising when one considers the numerous approximations made in the computation of Γ and in the estimation of the absorbances. The estimates of the concentration are as good as they can be considering the precision of the measurements. The errors are almost all

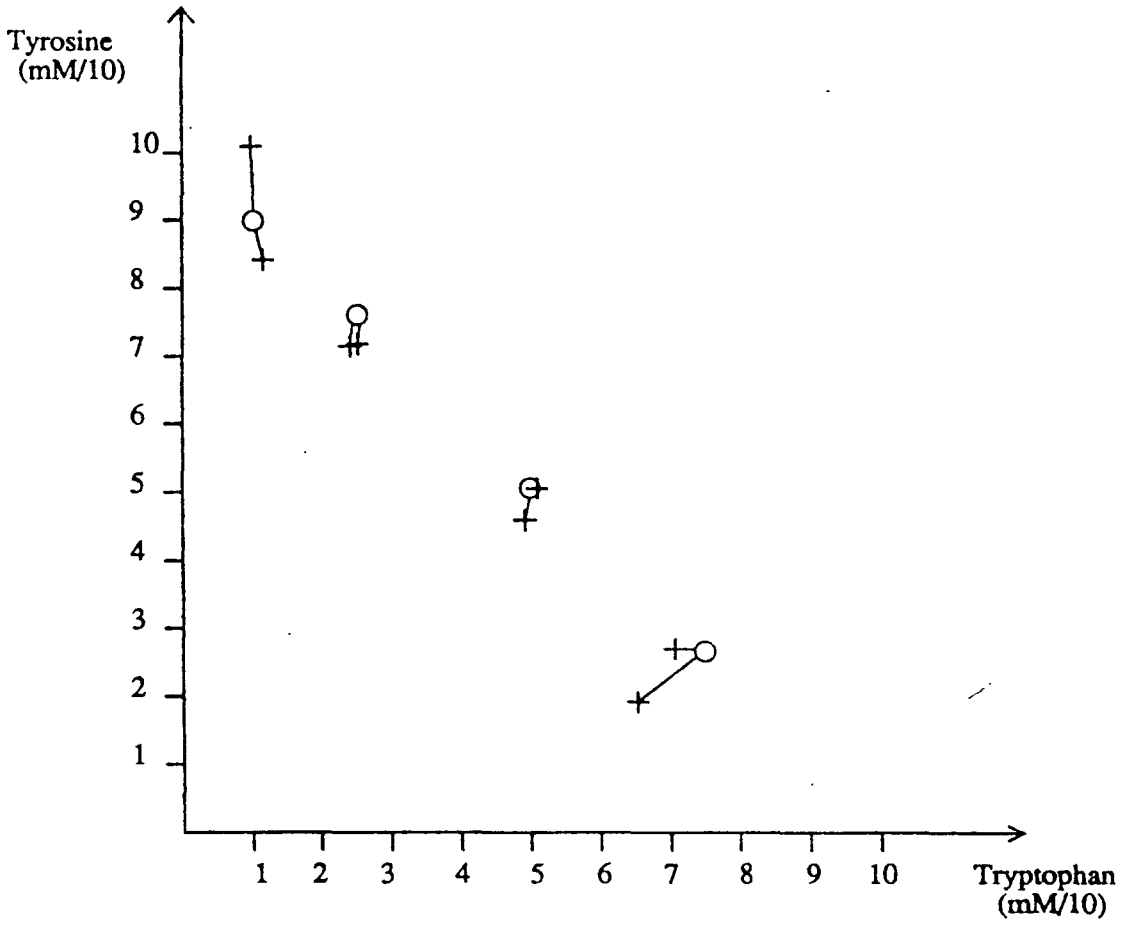


Fig.6.3. Results of a multiple linear regression corrected for absorption on amino acids mixtures. The circles represent the true composition and the crosses represent the estimates of the composition.

under 10% and seem random. It proves that, in these amino acids mixtures, the non linearity is mainly introduced by the absorbance at the excitation wavelength. The simple model developed can eliminate this non linearity if a measurement of the absorption at the excitation wavelength is provided.

A limitation of the model is the assumption on the law of absorption and fluorescence. In particular, the high intensity of the laser can produce a saturation of the absorption. For example, let us compute the number

of photons crossing the cuvette and the number of photons absorbed in a tryptophan solution during a laser pulse. The laser pulses obtained on the National Bureau of Standards setup have usually an energy of 12 mJ. Since the wavelength is 280 nm, each photon has an energy $\frac{hc}{\lambda} = 7\ 10^{-19}\ J$. Therefore a pulse contains $1.7\ 10^{16}$ photons. In the cuvette, the intensity to be absorbed in a slice dx is given by : $2.3\epsilon cIdx$. The number of molecules in this slide dx is : $6\ 10^{23}csdx$ where s is the section of the beam and c the concentration. Since the beam has a diameter of 5 mm, $s = 19.6\ mm^2$. In order to use the molar absorptivity, x should be expressed in centimeter and c in mole per liter, therefore s should be expressed in liter per centimeter or in tenth of meter square, so $s = 1.96\ 10^{-4}$. The number of molecules in the slice is therefore : $1.18\ 10^{20}cdx$. Considering the first slice hit by the beam, the intensity to be absorbed is $2.3\epsilon cI_0dx$ or expressing this in term of photons, $2.3\epsilon c1.7\ 10^{16}dx$ photons should be absorbed. Since the molar absorptivity of tryptophan is 5000, $1.96\ 10^{20}cdx$ photons should be absorbed. Since the photons are emitted in time frame of 10 nanoseconds which the same order as the fluorescence lifetime, a molecule can only absorb one photon and there are more photons to be absorbed than there are molecules. Therefore the linear relation for absorption does not apply, there is saturation. This saturation occurs mainly on the side hit by the beam. Assuming that every molecule absorbs a photon, the intensity in the cuvette will decrease as : $I = I_0 - 1.18\ 10^{-20}cdx$. Therefore in a

10^{-3} M solution the number of photons to be absorbed becomes equivalent to the number of molecules after 6 $10^{-2}cm$. This effect is very local but its consequences are hard to estimate.

6.3 Application of partial least squares regression

A partial least squares regression program has been written using the algorithm presented by Geladi and Kowalski[1986] and used to estimate the concentrations of the mixtures already studied in the preceding section. The spectra of the two pure components and eight mixtures shown in Appendix B are smoothed as described in section 3.3.4. From each smoothed spectrum, thirty values are taken, one every 5 nm from 285 up to 430 nm. The reference spectra set needed for partial least squares is composed of the two pure pure components spectra, a mixture 0.5 mM tryptophan 0.5 mM tyrosine, a mixture 0.75 mM tryptophan 0.25 mM tyrosineand a mixture 0.25 mM tryptophan 0.75 mM tyrosine. The five spectra are stored in a 5×30 matrix and the corresponding concentrations in a 5×2 matrix. Each variable is mean-centered and scaled to unit variance. The partial least squares algorithm is then used to determine a model at the maximum order 5. The p, q and w vectors are saved for the prediction. The tables 6.4 to 6.8 show the estimates of concentration of the five spectra included in the model as the order of the model increases.

Table 6.9 shows the sum of the squares of the errors of prediction as

	Real	Order 1	Order 2	Order 3	Order 4	Order 5
Tryptophan	1.0	0.868	0.957	0.993	1.0	1.0
Tyrosine	0.0	0.132	0.043	0.007	10^{-16}	10^{-16}

Table 6.4. Partial Least Squares estimation of the composition of a millimolar tryptophan solution. The concentrations are given in mM.

	Real	Order 1	Order 2	Order 3	Order 4	Order 5
Tryptophan	0.0	-0.090	-0.006	0.004	0.0	0.0
Tyrosine	1.0	1.09	1.006	0.996	1.0	1.0

Table 6.5. Partial Least Squares estimation of the composition of a millimolar tyrosine solution. The concentrations are given in mM.

	Real	Order 1	Order 2	Order 3	Order 4	Order 5
Tryptophan	0.5	0.630	0.501	0.515	0.5	0.5
Tyrosine	0.5	0.370	0.499	0.485	0.5	0.5

Table 6.6. Partial Least Squares estimation of the composition of a tryptophan-tyrosine mixture.

	Real	Order 1	Order 2	Order 3	Order 4	Order 5
Tryptophan	0.75	0.723	0.802	0.754	0.75	0.75
Tyrosine	0.25	0.277	0.198	0.246	0.25	0.25

Table 6.7. Partial Least Squares estimation of the composition of a tryptophan-tyrosine mixture.

	Real	Order 1	Order 2	Order 3	Order 4	Order 5
Tryptophan	0.25	0.368	0.246	0.234	0.25	0.25
Tyrosine	0.75	0.631	0.754	0.766	0.75	0.75

Table 6.8. Partial Least Squares estimation of the composition of a tryptophan-tyrosine mixture.

	Order 1	Order 2	Order 3	Order 4	Order 5
Error	0.114	0.0092	0.0011	0.0	0.0

Table 6.9. Sum of the squares of the estimation errors at different orders.

the number of components included in the model increases. As expected the accuracy of the estimates increases with the order of the model and the exact values are obtained when the order is equal to the size of the

reference set. The model can be used to estimate the concentration of mixtures non included in the reference set. The tables 6.10 to 6.15 show the results of this estimation.

	Real	Order 1	Order 2	Order 3	Order 4	Order 5
Tryptophan	0.5	0.621	0.536	0.509	0.496	0.502
Tyrosine	0.5	0.379	0.464	0.491	0.504	0.498

Table 6.10. Partial Least Squares estimation of the composition of a tryptophan-tyrosine mixture.

	Real	Order 1	Order 2	Order 3	Order 4	Order 5
Tryptophan	0.75	0.720	0.727	0.732	0.743	0.742
Tyrosine	0.25	0.280	0.273	0.268	0.257	0.258

Table 6.11. Partial Least Squares estimation of the composition of a tryptophan-tyrosine mixture.

	Real	Order 1	Order 2	Order 3	Order 4	Order 5
Tryptophan	0.25	0.351	0.292	0.294	0.297	0.313
Tyrosine	0.75	0.649	0.708	0.706	0.703	0.687

Table 6.12. Partial Least Squares estimation of the composition of a tryptophan-tyrosine mixture.

	Real	Order 1	Order 2	Order 3	Order 4	Order 5
Tryptophan	0.1	0.152	0.128	0.149	0.151	0.148
Tyrosine	0.9	0.848	0.872	0.851	0.849	0.852

Table 6.13. Partial Least Squares estimation of the composition of a tryptophan-tyrosine mixture.

	Real	Order 1	Order 2	Order 3	Order 4	Order 5
Tryptophan	0.1	0.116	0.039	0.094	0.089	0.076
Tyrosine	0.9	0.884	0.961	0.906	0.911	0.923

Table 6.14. Partial Least Squares estimation of the composition of a tryptophan-tyrosine mixture.

	Order 1	Order 2	Order 3	Order 4	Order 5
Error	0.057	0.0162	0.0096	0.001	0.0137

Table 6.15. Sum of the squares of the estimation errors at different orders.

The prediction with 3 components is the best one. The errors in prediction are in the order of 0.03 mM and are within the measurement errors. Therefore the result can be considered as very satisfactory.

The algorithm performed very well on this example. The computation of the model was very fast. The convergence in the loop (steps 2 to 8 of the algorithm) was generally obtained in 2 iterations. The prediction requires only vector and matrix multiplications which are almost instantaneous on a personal computer.

Chapter 7

Conclusion

The purpose of this study is an estimation of the capabilities of laser induced fluorescence in fermentation monitoring. The analysis of simple amino acids mixtures is a first step.

This study has shown the difficulties of fluorescence measurements and suggested some approaches to resolve multicomponents mixtures. More precisely, it has been shown that :

- ⋆ The measurements are not very well reproducible.
- ⋆ A degradation of the fluorescence signal appears after a prolonged illumination.
- ⋆ The photon noise is very intense on the National Bureau of Standards setup.
- ⋆ Strong non linearities appear even in simple two components mixtures due to purely optical reasons.
- ⋆ Absorbance measurements should allow to get ride of the optical non linearities.

⋆ Saturation of absorption appears in some part of the cuvette.

⋆ Partial Least Squares regression gives good concentration estimates despite the non linearities.

From this results, some axis of research seem particularly interesting to improve the experimental setup as well as to improve the data analysis.

7.1 The experimental setup

The noise reduction and the improvement of the measurement reproducibility are of major interest. One can not think to develop much further a sensor if these two problems are not solved in a satisfactory way. Even the most complex and sophisticated data processing technique will fail in analyzing data if the reproducibility is not sufficient. From the study, it appears that the noise is mainly a photon noise. Therefore an increase of the signal to noise ratio can be obtained by increasing the efficiency of the fluorescence collection or by using a multichannel detector. A photodiode array would probably permit to reduce the noise as well as the length of the measurements. Presently the measurement of a spectrum takes 3 to 5 *min.* It is long for an on-line measurement.

A much more accurate control of the environmental conditions like temperature, pH and p_{O_2} should allow to improve the reproducibility of the measurements. A correction for the instruments characteristics would be also interesting. Indeed such a correction would give to the measurement a constant sensitivity over the whole range of frequency and improve the

significance of most of the calculations. When working at low concentration such effects as Raman peak and background fluorescence appeared. It is very important if one want to work in this domain to make a background correction, by example by measuring a spectrum of water and subtracting it from every spectrum. The implementation of absorbance measurement simultaneously to the fluorescence measurement is also a very interesting idea. The improvement of the linearity should tremendously help the data deconvolution.

7.2 The data analysis

The data analysis is the key of the improvement of fluorescence measurement analysis. The high data acquisition rate allows to investigate sophisticated methods to resolve the mixture spectra. The sensitivity of fluorescence to environmental conditions and its lack of selectivity give a high complexity to the fluorescence measurements. Only computer aided data analysis can handle this large amount of complex data. Principal component analysis and partial least squares are good starting points to improve the fluorescence spectra analysis as shown in Chapter 6.

The great possibilities offered by fluorescence measurements have been recalled during this study. A good deconvolution of two components mixtures has been performed and as it is shown in Appendix A, fluorescence gives some information about the fermentation. However a lot of research

is still to be done in order to increase the understanding of the fluorescence processes, to improve the reliability and accuracy of the measurements and to analyze the data.

Appendix A

Fermentation Measurements

Fermentation measurements have been done at the National Bureau of Standards. The fermentation studied is a production of invertase by the yeast *Saccharamyces cerevisiae.* Fluorescence measurements and sampling were regularly performed during the fermentation. On the samples, the biomass concentration was measured. Figure A.1 shows a spectrum at the beginning of the fermentation and Figure A.2 one at the end.

One can clearly see the difference although no peak appears. The fluorescence in the 300-360 nm range is due to the tryptophan present in the proteins of the cell wall. Therefore it varies with the biomass concentration. The fluorescence in the 380-440 nm range is due to the pyridoxine present in the nutrient. To analyze these measurements a very simple procedure has been elaborated. An integration of the spectrum over the 300-360 nm interval and over the 380-440 nm interval is performed. Since the global intensity is not very reliable the two integrals are expressed as a ratio. Figure A.3 shows this ratio and the biomass concentration measured on the sample.

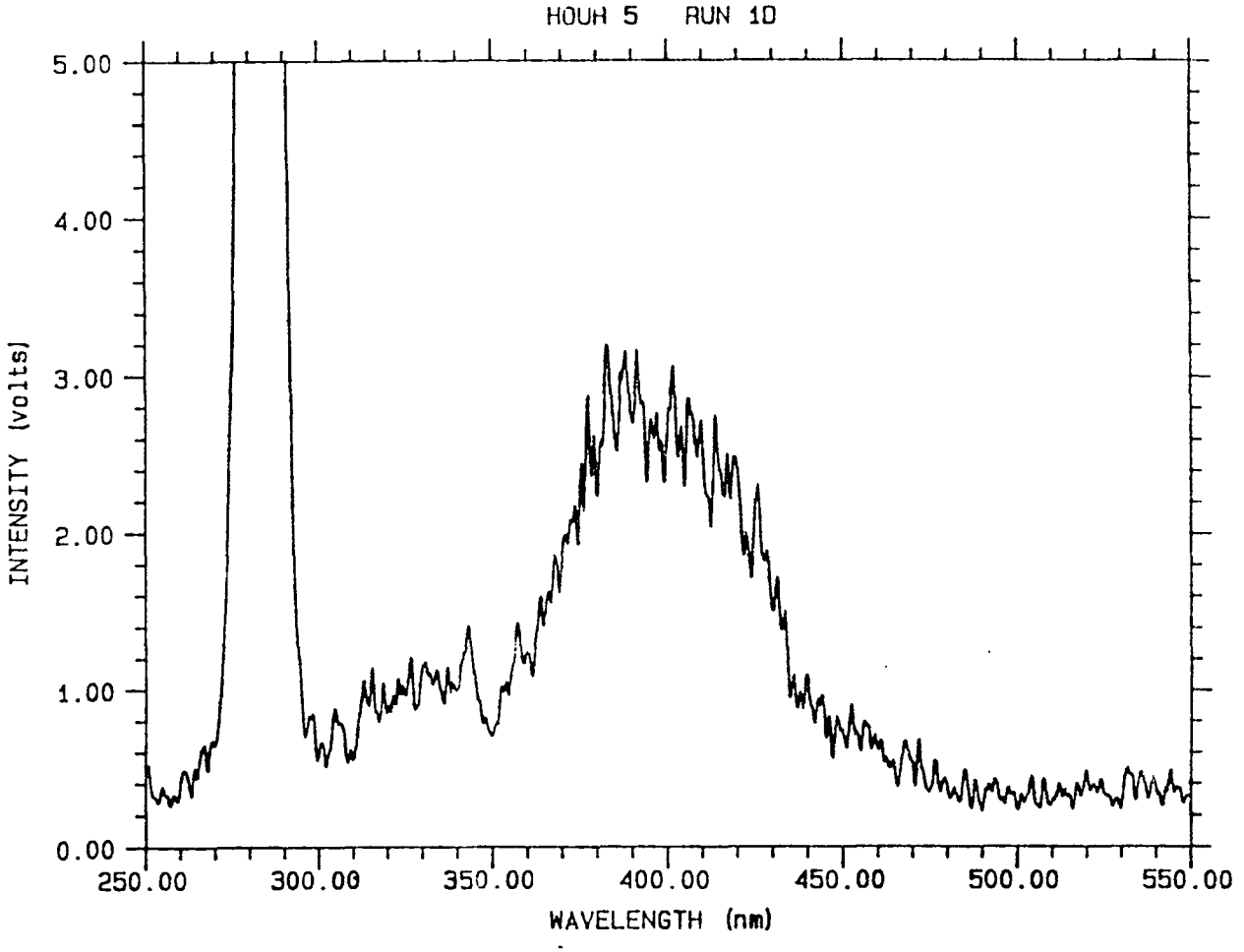


Fig.A.1. Fluorescence spectrum of a fermentation broth at t=5h.

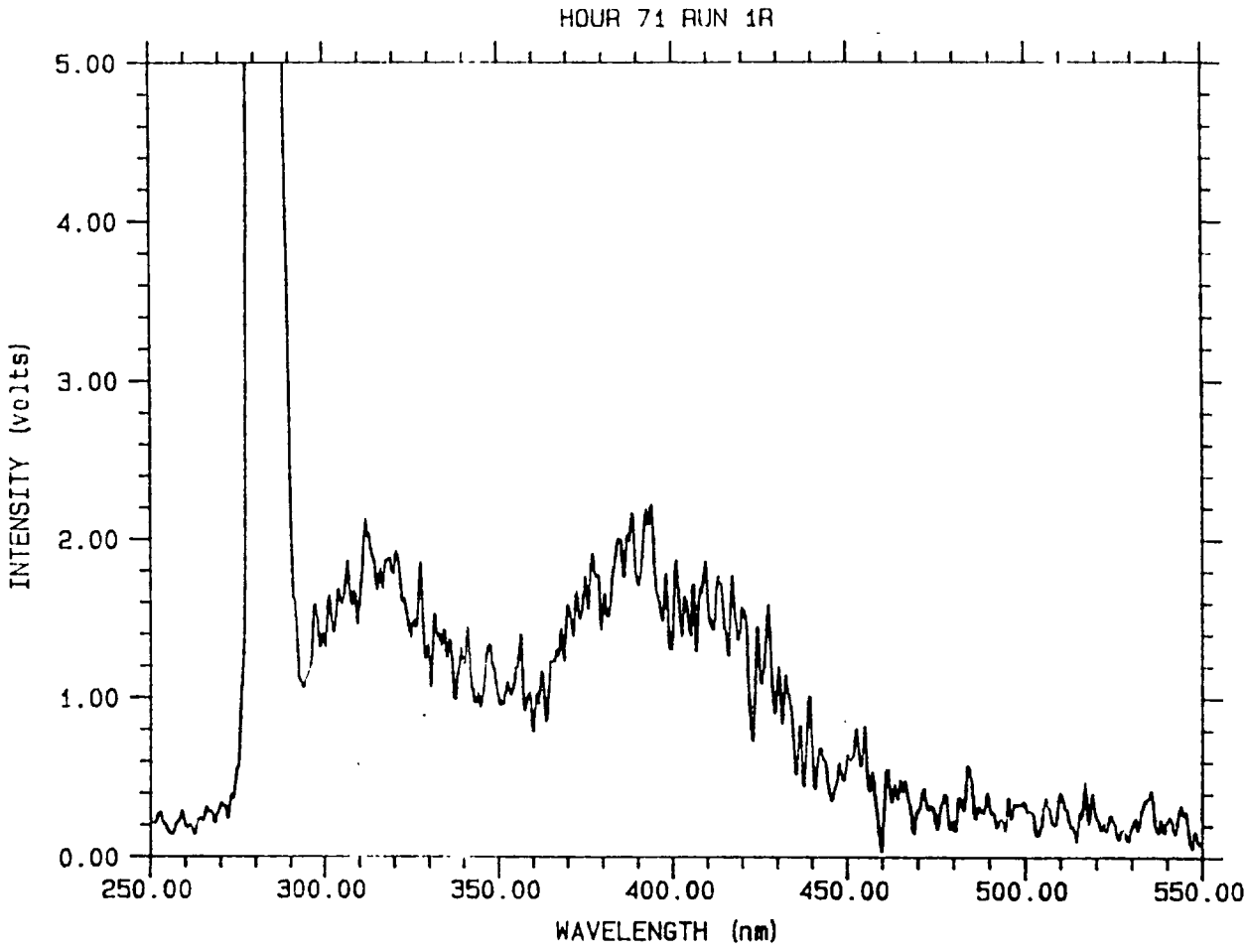


Fig.A.2. Fluorescence spectrum of a fermentation broth at t=71h.

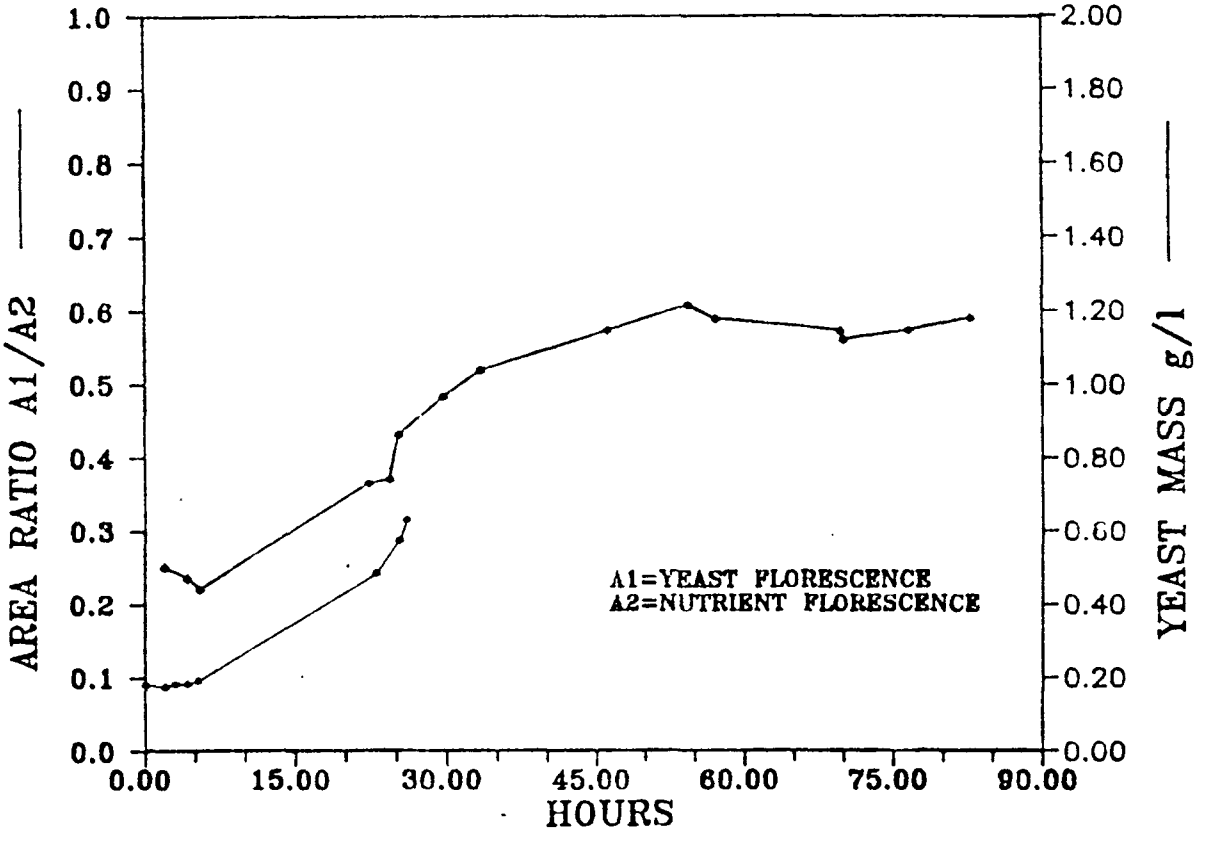


Fig. A.3 Ratio of the yeast fluorescence and nutrient fluorescence (upper curve) and yeast mass (lower curve).

Appendix B

Amino acids mixtures measurements

The next pages present the two pure components spectra and the eight mixture spectra used in Chapter 6. Figures B-1 to B-10 show the spectra corrected for the variation of the laser intensity.

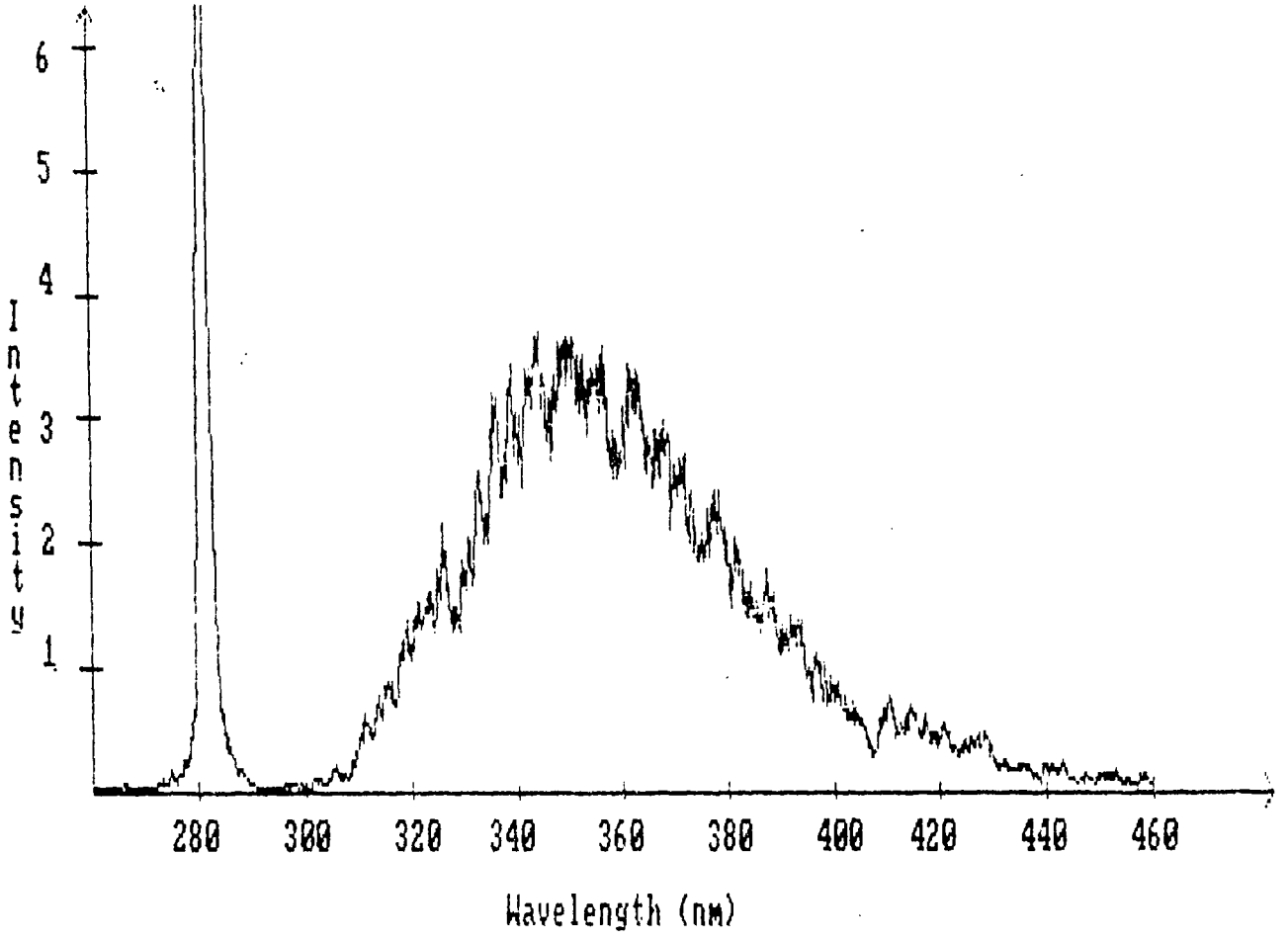


Fig B-1 - Tryptophan 10^{-3} mol/ℓ.

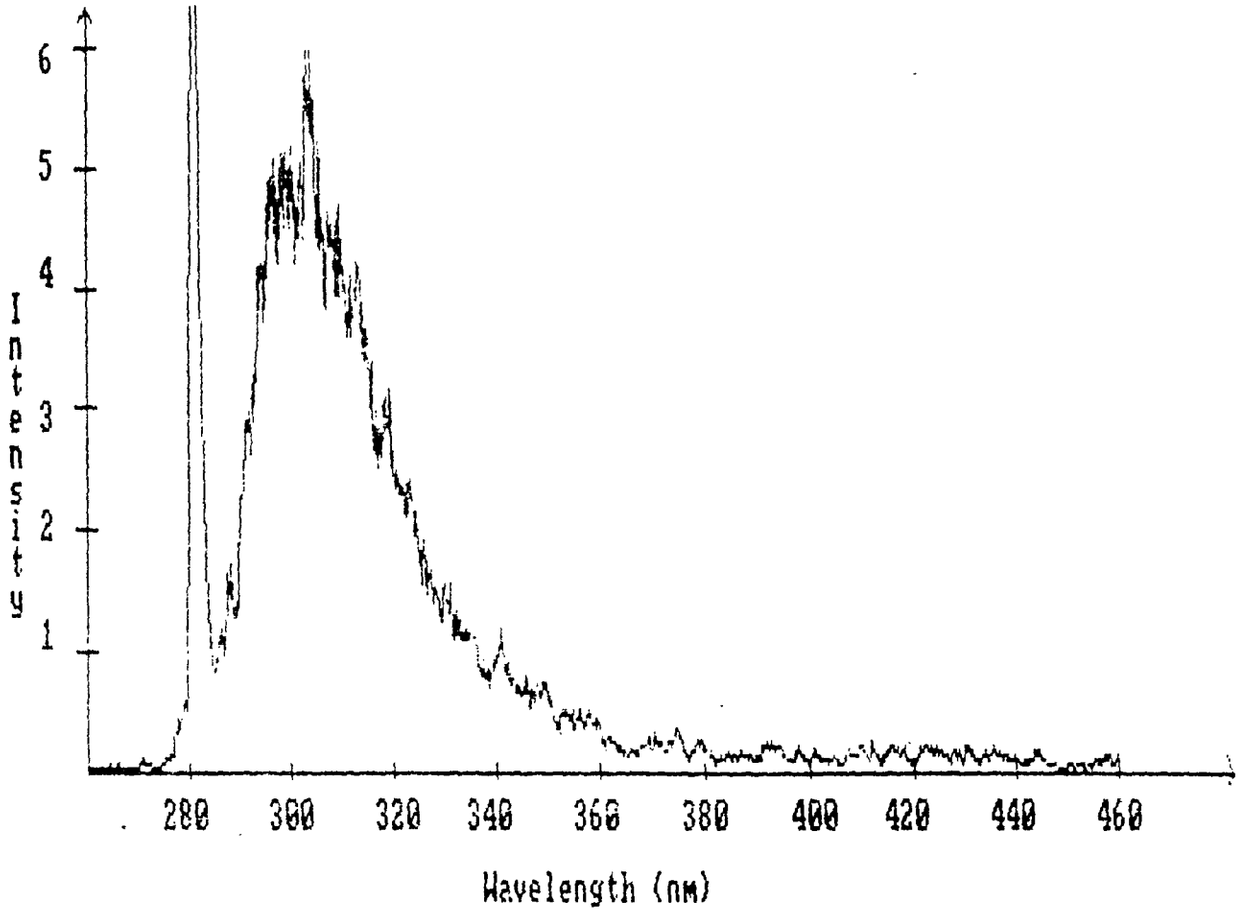


Fig B-2 - Tyrosine 10^{-3} mol/l

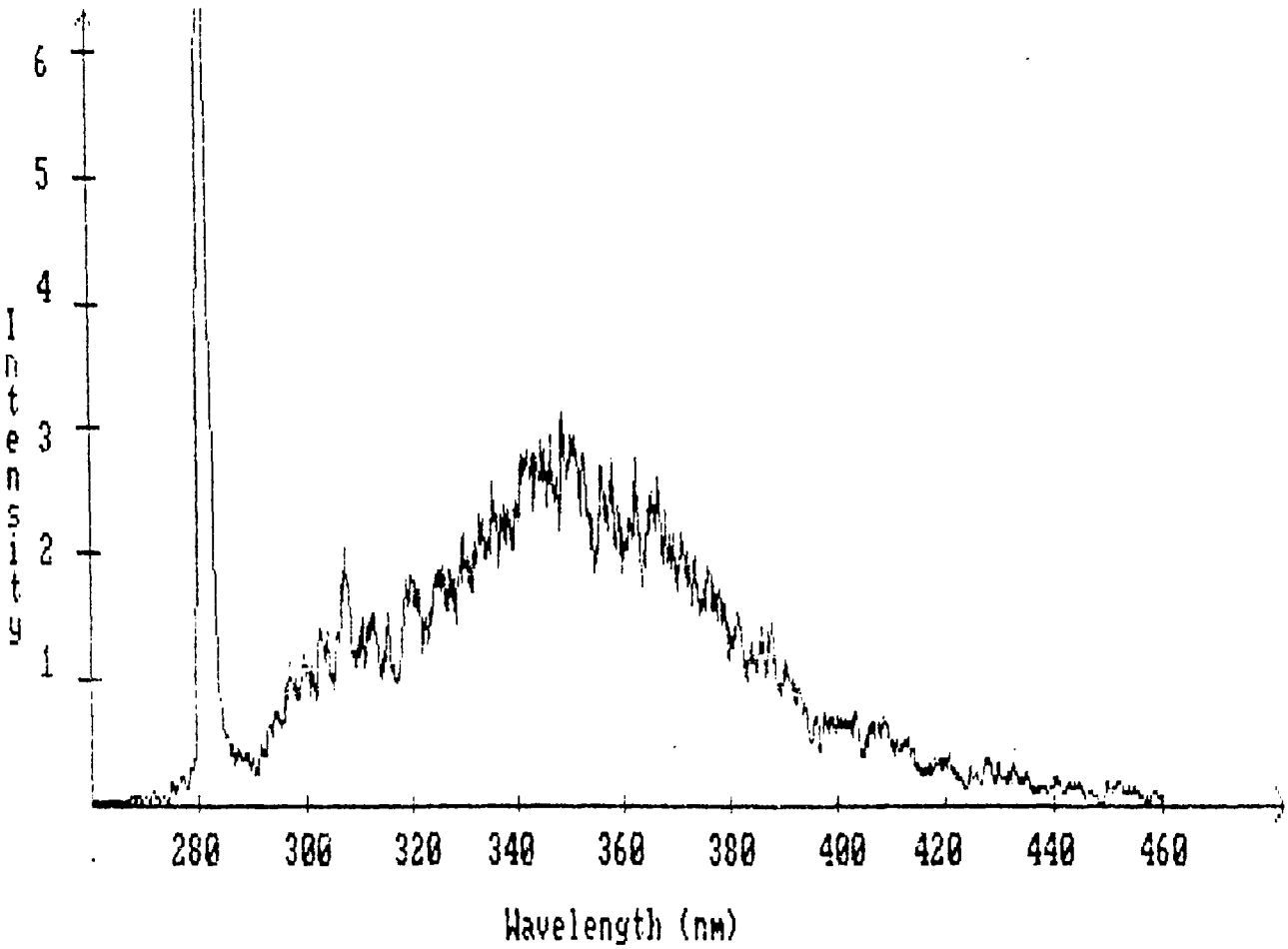


Fig. B-3. Mixture 5×10^{-4} M tryptophan 5×10^{-4} M tyrosine.

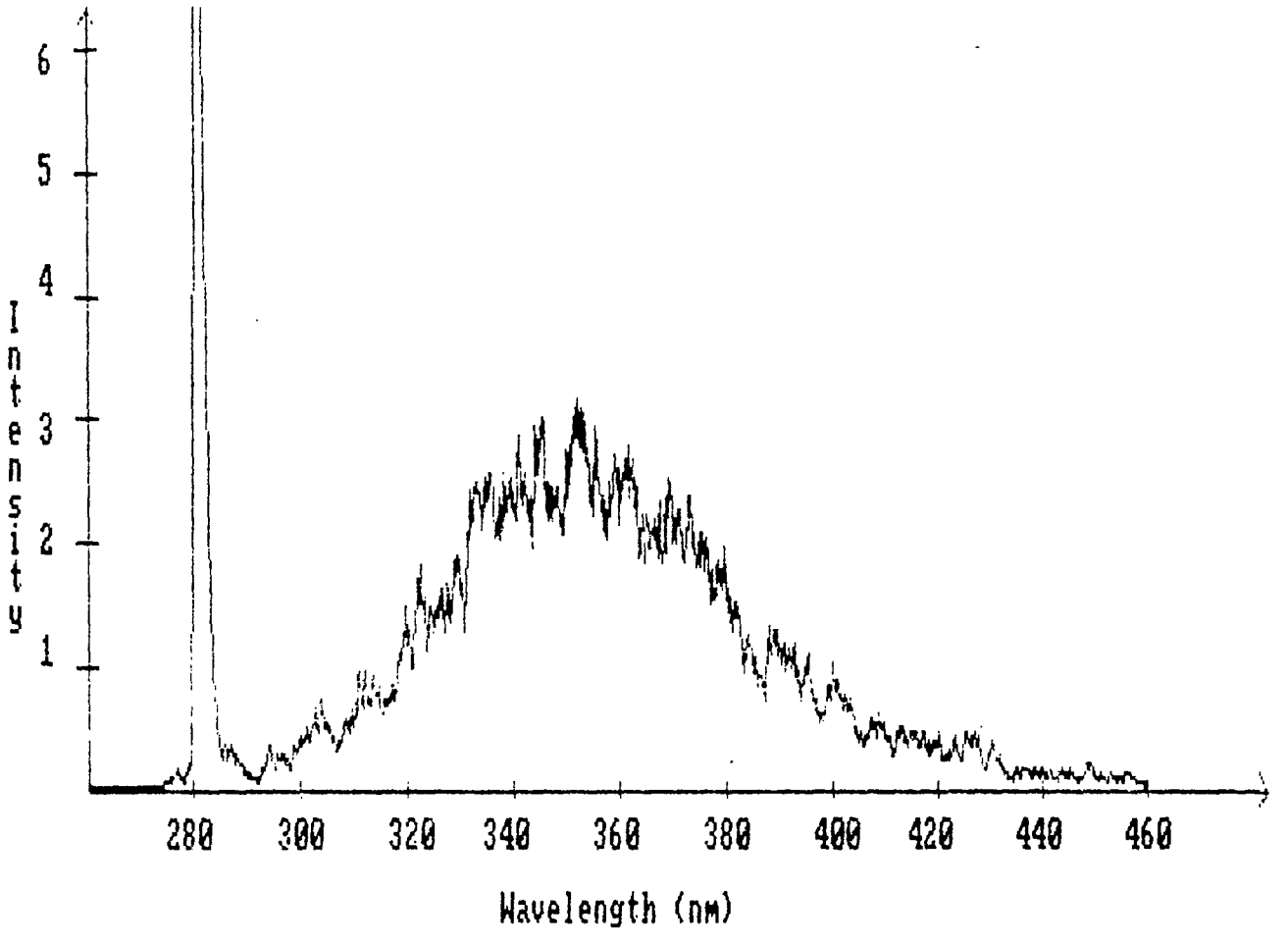


Fig. B-4 . Mixture $7.5\ 10^{-4}$ tryptophan $2.5\ 10^{-4}$ tyrosine -

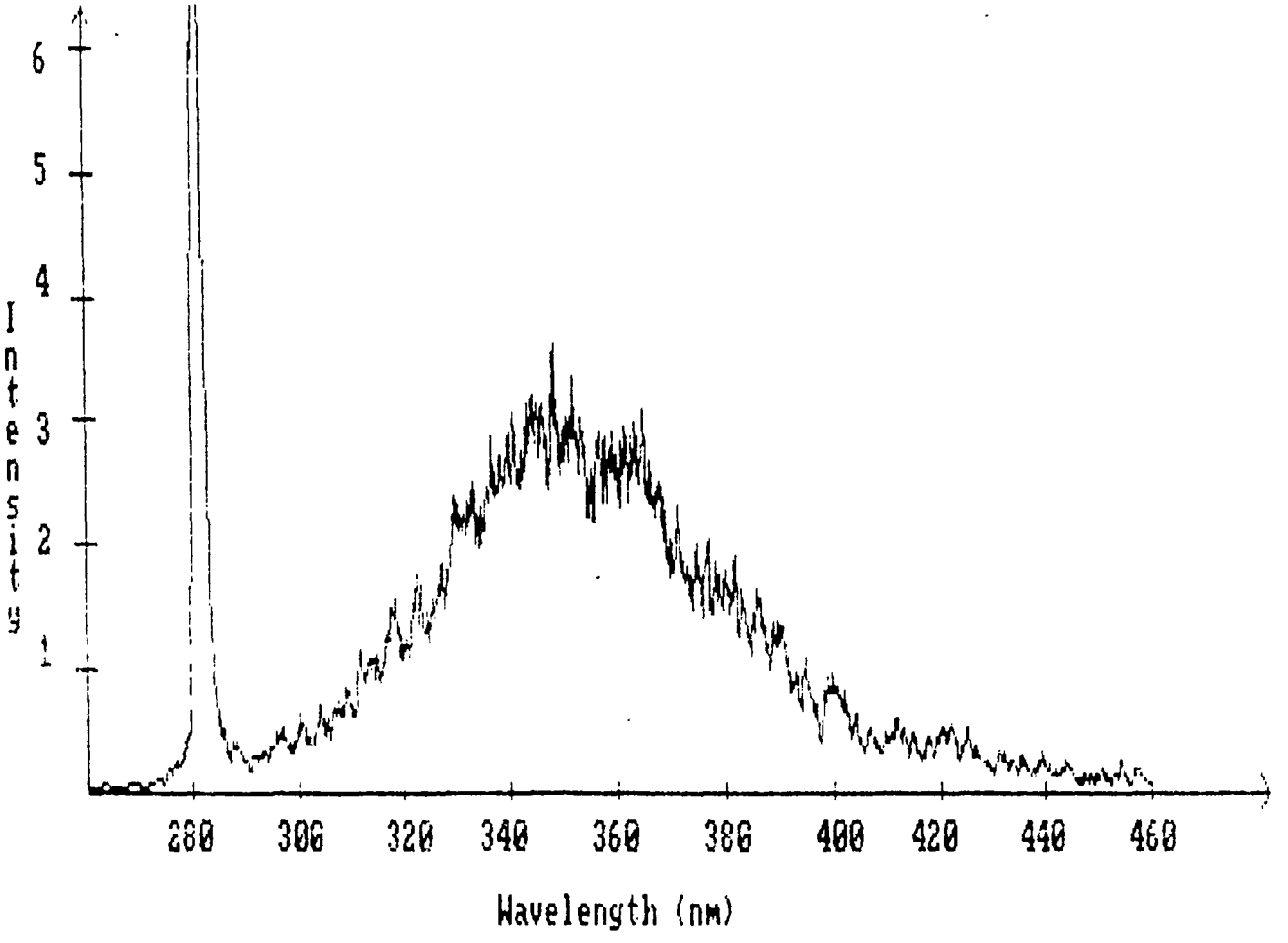


figure B-5. Mixture $7.5\ 10^{-4}$ M tryptophan $2.5\ 10^{-4}$ M tyrosine.

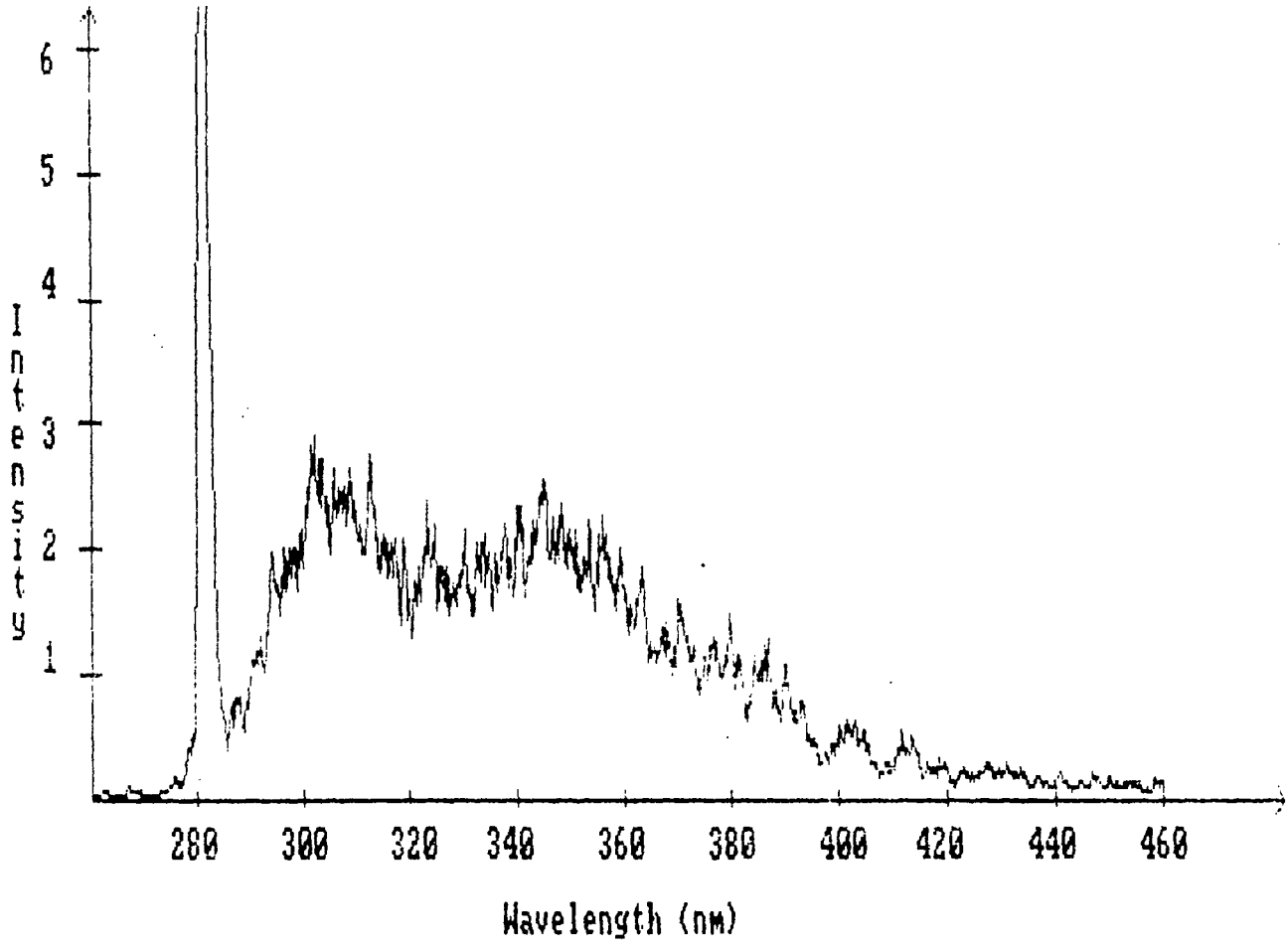


Fig B-6. Mixture $2.5\ 10^{-4}$M tryptophan $7.5\ 10^{-4}$M tyrosine.

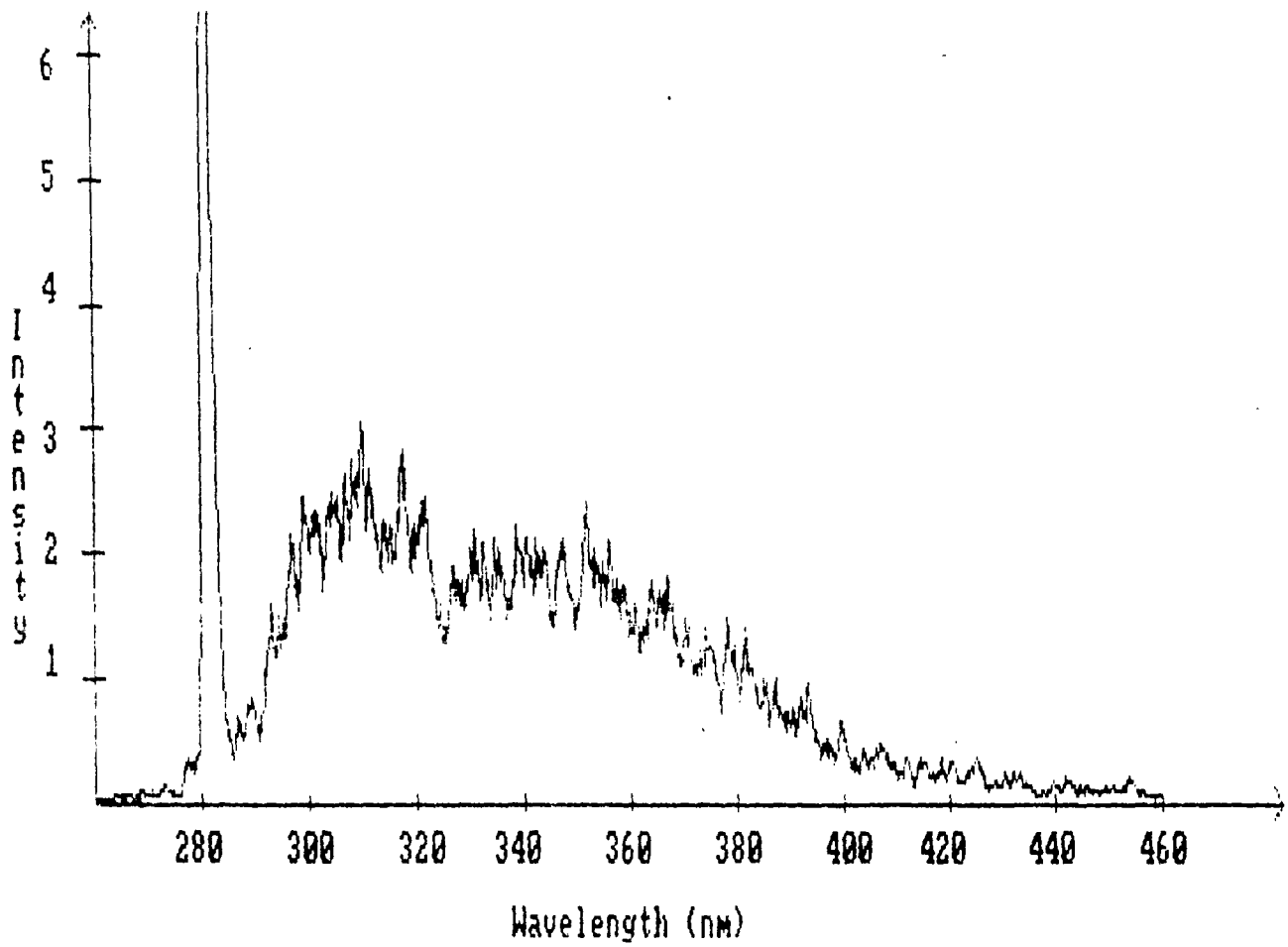


Fig 8-7. Mixture 2.5×10^{-4} M tryptophan 7.5×10^{-4} M tyrosine

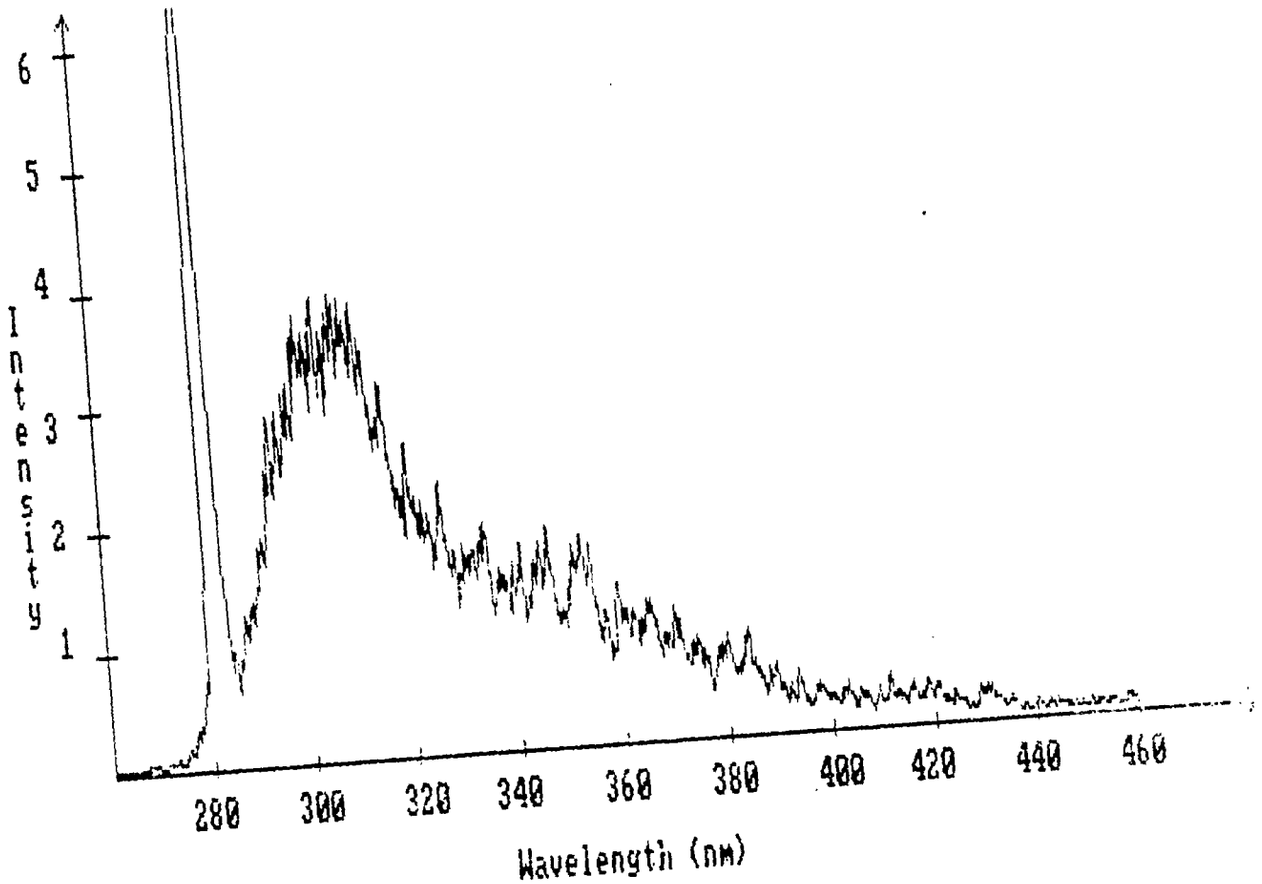


Fig 8-8. Mixture 9 10^{-4} m Tyrosine 10^{-4} m Tryptophan

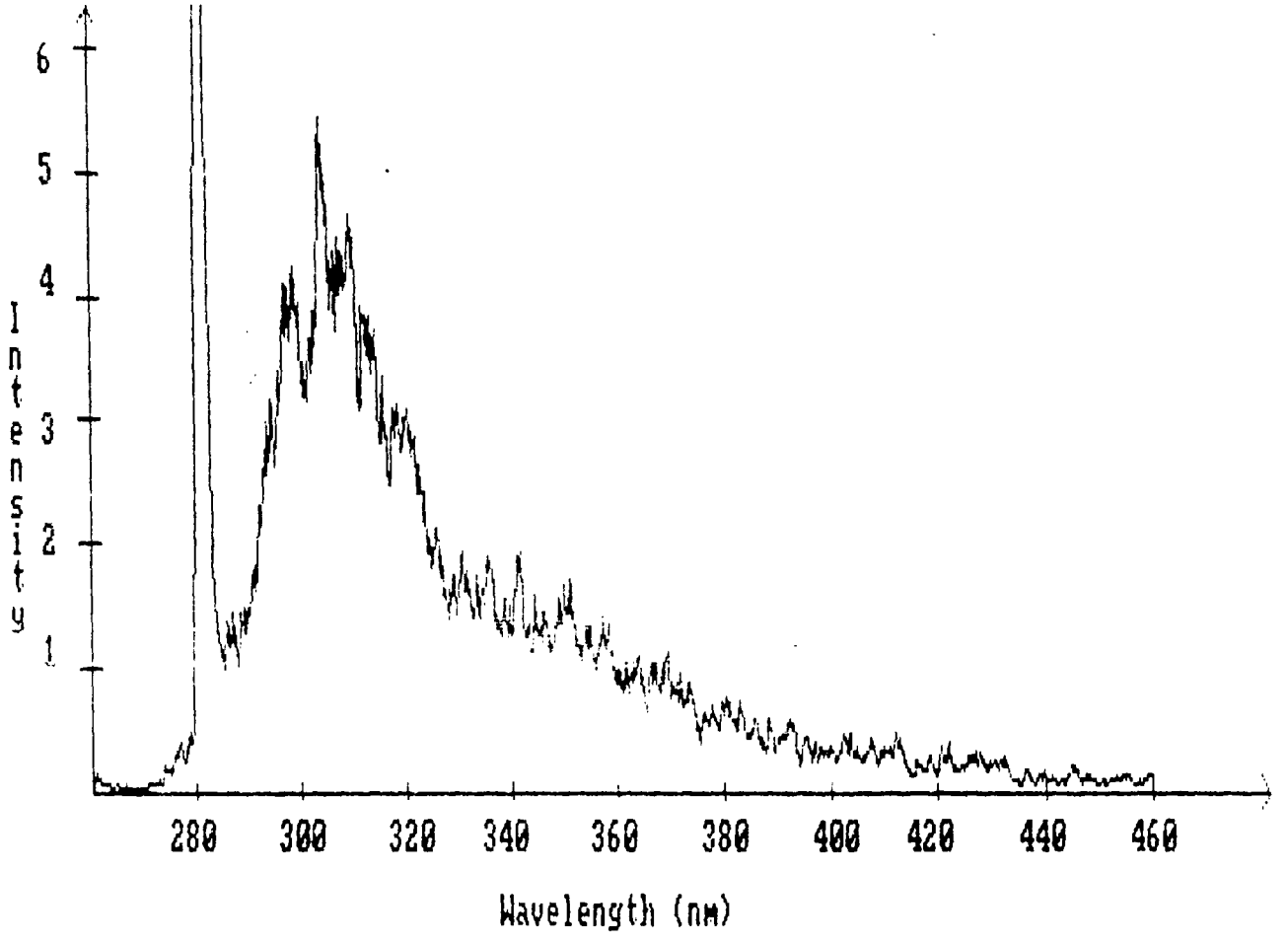


Fig 3-9 . Mixture 10^{-4}M tryptophan 4×10^{-4} M tyrosine

Appendix C

Format of the data file

The measurements are recorded on an IBM PC AT in an ASCII file with the following format :

- On the first line 3 numbers representing :
  - . The number of measurements (integer)
  - . The speed of the monochromator in Å/s (5,10 or 20,real)
  - . The number of measurements per nanometer (integer)
- On the second line a string representing :
  - . The title with 20 characters
  - . The time
  - . The date
- On the third line 2 reals :
  - . The higher wavelength
  - . The lower wavelength
- On the fourth line 2 reals :
  - . The range of the measurements (0 to 10 volts)
- The measurements given with 3 reals on each line :
  - . The wavelength in nanometer
  - . The intensity of the fluorescence
  - . The intensity of the laser output

Appendix D

Listing of SpecProc

The listing of SpecProc, the spectrum analysis program written for this study, is given in this appendix in the order of the files included in the main program.

```
{*****************************************************************}
{*                                                               *}
{*              MAIN PROGRAM FOR SPECTRUM PROCESSING             *}
{*                                                               *}
{*****************************************************************}

{$R+}                   { compiler option for arrays index checking }

          program SpecProcessing;

{*****************************************************************}

{*                  CONSTANT DECLARATION                         *}

     const MaxMeasNumber   = 3000;      { Maximum size of the spectrum file }
           DataArraySize   =  400;      { Size of the linked arrays used to store
 the data}
           TNarraysize     = 2047;      { Size of the arrays used in the FFT }

{                  TYPE VALUE FOR THE DATA FILE       }

           Fluorescence    =    1;
           NormedFluo      =    2;
           LaserPower      =    7;
           Continue        =    5;
           FourierTr       =   11;

{                  FILE FOR THE DEFAULT VALUE          }

           Defile          = 'SPECPROC.DAT';

{*****************************************************************}

{*                    TYPE   DEFINITION                          *}

     type  LabelString =  string[3];           { used for the axis numbers labe
ls}
           Name        =  string[30];          { used for the filename }
           Line        =  string[70];          { used for a screen line }
           BigString   =  string[255];         { used by some string procedure }

           MenuLine    =  record               { used for the menu procedure}
                            Title    :  Name;
                            TypeL    :  integer;
                            Bool     :  boolean;
                            Re       :  real;
                            Str      :  Name;
                          end;

           MenuTab     =  array [1..25] of MenuLine;   { input/output of the menu
procedure}

           DataTab     =  array [1..DataArraySize] of real; { used to store the da
ta}

{                    TYPE SPECTRUM USED TO STORE THE SPECTRA      }

           Grdata      =  record                       { information used to plo
t the spectrum}
                            OnOff,Power     :  boolean;
                            PowScale        :  real;
                            Title           :  Name;
                            Xmin,Xmax,Xshift,Xscale     :  real;
                            Ymin,Ymax,Yshift,Yscale     :  real;
                          end;

           Dpoint      =   Data;                  { pointer to data }

           Data        =  record                  { used to store the data }
                            DataType       : integer;
                            Data           : DataTab;
                            DataLink       : Dpoint;
                          end;
```

```
          Pointer    = ^Spectrum;                  { pointer to spectrum }

          Spectrum   =  record                     { source file, format and data
of a spectrum)
                          Source,LocalName :  Name;
                          MeasNumber       :  integer;
                          Xmin,Increment   :  real;
                          Measure1         :  Dpoint;
                          Measure2         :  Dpoint;
                          Graph            :  Grdata;
                          Link             :  Pointer;
                        end;


          TNvector    = array[0..TNarraysize] of real;    { used for FFT }
          TNvectorPtr = ^TNvector;


          AxDef      =  record                             { used to draw the ax
is }
                          Title,Labelx,Labely  :  Name;
                          Xmin,Xmax,DeltaX     :  real;
                          Ymin,Ymax,DeltaY     :  real;
                        end;

          Gtype      =  record                             { used for input }
                          Int,Re,Chr,Esc : boolean;
                        end;

          Gans       =  record                             { used for input }
                          Int    : integer;
                          Re     : real;
                          Chr    : char;
                          Str    : Name;
                          Typ    : Gtype;
                        end;

(*************************************************************************)

(*                  VARIABLES DECLARATION                                *)

   var    Current_Dir    : Name;
          Head           : Pointer;                    {head of the spectra l
inked list)
          CurveNum       : array[1..9] of Pointer;
          Axis           : Axdef;                      {axis generally used}
          MainCh         : char;
          Iord,i         : integer;
          Xscreenscale,Yorig,YscreenScale,Yorig : real;  {scale used for plotin
g)
          Default_Type   : boolean;
          Ganswer        : Gans;
          SXstart,SXend,WaveLengthShift  : real;
          Fxstart,Fxend,y0,yn            : real;
          Textcol,Anscol,Background      : integer;

(*************************************************************************)

(*                  INCLUDED FILES                                       *)

     {$Ilib}
     {$Igraphlib}
     {$Iiolib}
     {$Iinit}
     {$Ilistman}
     {$Imenu1}
     {$Iaxis}
     {$Ispecplot}
     {$Ispeclist}
     {$Icharacterize}
     {$Iconvert}
     {$Iff*b2.inc}
     {$IrealFFt.inc}
     {$IcompFFt.inc}
     {$Isnfft}
     {$Idefine}
     {$Ifunconly}
     {$Icomputing}
     {$Istat}
     {$Ianimenu}

 *************************************************************************
```

```
{*                    BEGINNING OF THE CODE                              *}

begin
  head := nil;
  init;
  MainCh := Chr(120);
  repeat
    Iord := Ord(MainCh);
    if Iord = 27 then SpecPlot(MainCh)
    else
      if Iord = 31 then Speclist(MainCh)
      else
        if (Iord > 58) and (Iord < 68) then Characterize(Iord-58,MainCh)
        else
          if Iord = 68 then AxisDef(MainCh,Axis)
          else
            if Iord = 121 then Define(MainCh)
            else
              if Iord = 122 then LinComb(MainCh)
              else
                if Iord = 123 then Smoothing(MainCh)
                else
                  if Iord = 124 then Stat(MainCh)
                  else MainMenu(MainCh);
  until( Ord(MainCh) = 16 ) ;
  quit;
end.
```

```
{-------------------------------------------------------------------------}

      function  Min(x,y : integer) : integer;
{
          THIS FUNCTION COMPUTES THE MINIMUM OF TWO INT
}

   begin
     min :=x;
     if y<x then min:=y;
   end;

{-------------------------------------------------------------------------}

                function  Max(x,y : integer) : integer;
{
          THIS FUNCTION COMPUTES THE MINIMUM OF TWO INTEGERS
}

   begin
     Max :=x;
     if y>x then Max:=y;
   end;

{-------------------------------------------------------------------------}

                 function PowerOfTen (x : real) : real;
 {
                THIS FUNCTION COMPUTES 10^X WHERE X IS REAL
 }
      begin
        powerOfTen := exp(x*ln(10));
      end;
{-------------------------------------------------------------------------}

                      function Log(x:real) : real;
 {
                THIS FUNCTION COMPUTES THE DECIMAL LOGARITHM OF A REAL  X
 }
      begin
        Log := ln(x)/ln(10);
      end;
{-------------------------------------------------------------------------}

                      function divide(i1,i2 : integer) : boolean;
{
                THIS FUNCTION TESTS IF i1 DIVIDES i2
}
       begin
         divide := ((i2/i1)= i2 div i1) { 1.e-3/i2;
       end;

{-------------------------------------------------------------------------}

                       function pgcd(i1,i2 : integer) : integer;
{
                THIS FUNCTION COMPUTES THE LARGEST COMMON DIVISOR OF i1 AND i2
}

      var  j1,j2,k :integer;
           exit    : boolean;

      begin
        j1 := min(i1,i2);
        j2 := max(i1,i2);
        k  := j1;
        repeat
          if divide(k,j1) then
            if divide(k,j2) then
            begin
              pgcd := k;
              exit := true;
            end;
          k := k-1;
        until exit;
      end;
```

```
{-------------------------------------------------------------------------}

                        function ppcm(i1,i2 : integer) : integer;

{
                THIS FUNCTION COMPUTES THE SMALLEST COMMON MULTIPLE OF i1 AND i2
}
       begin
         ppcm := (i1*i2) div pgcd(i1,i2);
       end;

{-------------------------------------------------------------------------}

             function Valid (SpecName : Name) : boolean;
{
        THIS FUNCTION TESTS IF A NAME IS ACTUALLY THE NAME OF A SPECTRUM
}

   var   point : pointer;
         valid2 : boolean;


  begin
    valid2 := false;
    point := head;
    while  (point    nil) and not valid2 do
    begin
      if point .LocalName = SpecName then valid2 := true;
      point := Point^.Link;
    end;
    valid := valid2;
  end;

{-------------------------------------------------------------------------}

             procedure  SpecIndic (SpecName : Name;var Po nti  : pointer) ;
{
            THIS PROCEDURE FINDS THE POINTER POINTING ON THE SPECTRUM
            WHOSE NAME IS SPECNAME
}

   var  point : pointer;
        valid2 : boolean;


  begin
    valid2 := false;
    point := head;
    while  (point <> nil) and not valid2 do
    begin
      if point^.LocalName = SpecName then
      begin
        point1 := point;
        valid2 := true;
      end;
      point := Point .Link;
    end;
  end;

{-------------------------------------------------------------------------}
```

```
    procedure string3(x : real;var s : labelstring); forward;  {defined later}
{-------------------------------------------------------------------------}
    function Upstring(Source : BigString) : BigString; forward; {defined later}
{-------------------------------------------------------------------------}

(*************************************************************************)

(*                        GRAPHICAL PROCEDURES                           *)

    procedure Tex;
    begin
      textcolor(textcol);
    end;

{-------------------------------------------------------------------------}

    procedure Ans;
    begin
      textcolor(anscol);
    end;

{-------------------------------------------------------------------------}

    procedure WriteTitle(StrTitle : Name:> : integer);
    begin
      textmode(3);
      textbackground(background);
      clrscr;
      textcolor(yellow);
      gotoxy(1,5);
      write(Upstring(strtitle));
      Tex;
    end;

{-------------------------------------------------------------------------}


    procedure writemem;

    begin
      textcolor(lightred);
      gotoxy(60,1);write('MemAvail = ', Memavail);
      gotoxy(60,2);write('MaxAvail = ',MaxAvail);
      Tex;
    end;

{-------------------------------------------------------------------------}

procedure PlotAxis(axis : axdef);

{   THIS PROCEDURE DRAWS THE AXIS   }

 var  x1,y1             : real;
      i,x2,y2,xistart   : integer;
      i1                : string[3];

{-------------------------------------------------------------------------
-)

procedure Arrow(x,y,teta:integer);

{   THIS PROCEDURE DRAWS THE ARROWS ON THE AXIS
    X,Y ARE THE COORDINATES ON THE SCREEN
    TETA IS THE ANGLE WITH THE X-AXIS                  }

 var x1,y1          : integer;
     angle,l1       : real;

    begin
     angle := teta*pi/180+3*pi/4;
     l1 := (2-4*abs(cos(angle)));
     x1 := trunc(x+l1*cos(angle));
     y1 := trunc(y+l1*sin(angle));
     draw(x1,y1,x,y,1);

     angle := teta*pi/180-3*pi/4;
     l1 := (2-4*abs(cos(angle)));
     x1 :=  trunc(x+l1*cos(angle));
     y1 := trunc(y+l1*sin(angle));
     draw(x1,y1,x,y,1);
    end;
```

```
{-------------------------------------------------------------------------}

 procedure Bar(x,y,teta:integer);

      THIS PROCEDURE DRAWS A BAR AT THE COORDINATES X,Y WITH AN ANGLE TETA    :

 var x1,y1,x2,y2  : integer;
     angle,l1     : real;

    begin
     angle := teta*pi/180;
     l1 := (2+4*abs(cos(angle)));
     x1 := trunc(x+l1*cos(angle));
     y1 := trunc(y+l1*sin(angle));
     x2 := trunc(x-l1*cos(angle));
     y2 := trunc(y-l1*sin(angle));
     draw(x1,y1,x2,y2,1);
    end;

{-------------------------------------------------------------------------}

 {  procedure PlotAxis : beginning of the code }

    begin
    XScreenScale := 540./(Axis.xmax-Axis.xmin);YScreenScale := 160./(Axis.Ymin-A
xis.Ymax);
    Xorig:= Axis.Xmin- 40/XScreenScale;Yorig := Axis.Ymax-10/yscreenscale;
    draw(40,170,640,170,1);arrow(639,170,0);
    x1:= (trunc(Axis.xmin/Axis.deltax)+1)*Axis.deltax;
    repeat
      x2 := trunc((x1-xorig)*XscreenScale);
      bar(x2,170,90);
      xistart := trunc(x2/8);
      string3(x1,i1);
      gotoxy(xistart,23);write(i1);
      x1 := x1 + Axis.deltax;
    until(x1 > Axis.Xmax*1.001);
    gotoxy(trunc((80-length(Axis.Labelx))/2),25);write(Axis.Labelx);

    draw(40,170,40,0,1);arrow(40,0,270);
    Y1:= (trunc(Axis.ymin/Axis.deltay)+1)*Axis.deltay;
    repeat
      y2 := trunc((y1-yorig)*YscreenScale);
      bar(40,y2,0);
      xistart := trunc(y2/8+1);
      gotoxy(2,xistart);write(y1:2:0);
      y1 := y1 + Axis.deltay;
    until(y1   A is.ymax*1.001);
    for  i := 1 to length(Axis.labely) do
    begin
      gotoxy(1,trunc((25-length(Axis.labely))/2+i));write(copy(Axis.labely,i,1))
;
    end;
    gotoxy(30,5);write(UpString(Axis.title));
    end;

{-------------------------------------------------------------------------}
```

```
{-----------------------------------------------------------------------------------------------}

  procedure Init;

{     THIS PROCEDURE INITIALIZE THE VARIABLES USED IN SPECPROC     }

  var  inputfile : text;
       error,i      : integer;
       Dtype        : integer;


  begin
    assign(inputfile,Defile);
    {$I-}
    reset(inputfile);
    error := IoResult;
    {$I+}
    If Error = 0 then
    begin
      readln(inputfile);
      readln(inputfile);
      readln(inputfile,Current_Dir);
      readln(inputfile,Dtype);
      Default_type := true;
      if Dtype = 0 then Default_type := false;
      readln(inputfile);
      readln(inputfile,SXstart,SXend,WaveLengthShift);
      readln(inputfile);
      readln(inputfile,fXstart,fXend);
      readln(inputfile);
      readln(inputfile,Axis.Title);
      readln(inputfile,Axis.Labelx);
      readln(inputfile,Axis.Labely);
      readln(inputfile,Axis.Xmin,Axis.Xmax,Axis.deltax);
      readln(inputfile,Axis.Ymin,Axis.Ymax,Axis.deltay);
      close(inputfile);
    end
    else begin
      Current_Dir := 'B:';
      Default_Type:= true;
      Axis.title := 'Laser Induced Spectrum';
      Axis.labelx := 'Wavelength (nm)';
      Axis.labely := 'Intensity';
      Axis.xmin  := 260;Axis.xmax := 460;Axis.deltax := 20;
      Axis.Ymin := 0;Axis.Ymax := 10;Axis.deltay := 2;
      sxstart := 300;fxstart := 300;
      sxend := 460;fxend := 460; WaveLengthShift :=0;
    end;
    for i:= 1 to 9 do curvenum[i] := nil;
  end;

{-----------------------------------------------------------------------------------------------}

  procedure quit;

{      THIS PROCEDURE SAVE THE DATA USED IN SPECPROC      }

  var    outputfile  : text;
         Dtype       : integer;

  begin
    assign (outputfile,Defile);
    rewrite (outputfile);
    writeln(outputfile,'                SPECPROCESSING DEFAULT VALUES');
    writeln(outputfile,'      For Spectrum Loading');
    writeln(outputfile,Current_Dir);
    Dtype := 1;
    if Default_type = false then Dtype := 0;
    writeln(outputfile,DType);

    writeln(outputfile,'      For Stat Analysis');
    writeln(outputfile,Sxstart,SXend,WaveLengthShift);

    writeln(outputfile,'      For Fft Analysis');
    writeln(outputfile,fxstart,fXend);

    writeln(outputfile,'      For Axis Plotting');
    writeln(outputfile,Axis.Title);
    writeln(outputfile,Axis.Label[illegible]);
    writeln(outputfile,Axis.Labely);
    writeln(outputfile,Axis.Xmin,Axis.Xmax,Axis.deltax);
    writeln(outputfile,Axis.[illegible],Axis.Ymax,Axis.d[illegible]lta[illegible]);

    close(outputfile);
  end;

{-----------------------------------------------------------------------------------------------}
```

```
{ . . . . . }

    procedure Spectre(Spec : Spectrum);
{
            THIS PROCEDURE CREATES A NEW PLACE IN THE LINKED
            LIST AND INSERTS SPECTRUM
}

    var Point  : Pointer;
        x,y    : integer;

    begin
      x := wherex;
      y := whereY;
      writemem;
      New(Point);
      Point^ := Spec;
      Point^.Link := Head;
      Head := Point;
    end;

{-------------------------------------------------------------------------}
    procedure Datadel ( point : Dpoint);
{
                ERASES A LINKED LIST OF DATA
}

    begin
      if point <> nil then
      begin
        datadel (point^.datalink);
        dispose(point);
      end;
    end;

{-------------------------------------------------------------------------}

    procedure SpecDel(SpecName : Name);
{
            FIND THE SPECTRUM CALLED SPECNAME AND ERASES IT
}

    var  Back,Next  : Pointer;
         Found      : boolean;

{-------------------------------------------------------------------------}

    procedure delete ( specpoint : pointer);
{
            DELETE A SPECTRUM
}

    begin
      datadel(specpoint^.measure1);
      datadel(specpoint^.measure2);
      dispose(specpoint);
    end;

{-------------------------------------------------------------------------}

    {BEGINNING OF SPECDEL CODE}
    begin
      If Head^.localname = SpecName then
      begin
        for i := 1 to 9 do
          if curvenum[i] = head then curvenum[i] := nil;
        Next := Head .Link;
        delete(Head);
        Head := Next;
      end
      else begin
        back := Head;
        next := Head .Link;
        found := false;
        while ( next <> nil ) and not found do
        begin
          if next^.LocalName = SpecName then
          begin
            for i := 1 to 9 do
              if curvenum[i] = next then curvenum[i] := nil;
            found := true;
            back^.Link := next .Link;
            dispose(next);
          end
          else begin
            back := next;
            next := next .Link;
          end;
        end;
      end;
    end;

{-------------------------------------------------------------------------}
```

```
{-------------------------------------------------------------------------}

    procedure curveaffect(spec : spectrum);
{
                LOOKS FOR AN AVAILABLE CURVE AND IF IT
                FINDS ONE AFFECTS SPEC TO THIS CURVE
}

    var i : integer;
        exit : boolean;
        Point : pointer;

    begin
        i := 1;exit := false;
        repeat
          if curvenum[i] = nil then
          begin
            exit := true;
            specindic(spec.localname,point);
            curvenum[i] := Point;
          end;
          i := i+1;
        until (( i > 9) or exit);
    end;

{-------------------------------------------------------------------------}
    procedure SpecSave(localName: name);
{
              SAVES THE SPECTRUM CALLED LOCALNAME TO DISK
}

    var   quit,test,error : boolean;
          x,y,z     : real;
          i,j       : integer;
          answer,defaultname : name;
          Nname,namefile : name;
          count : integer;
          outfile   : text;
          point1,point2 : Dpoint;
          Spoint : pointer;
          spec   : spectrum;


  begin
{       INITIALIZATION OF THE COLORS     }
    textcol := white;
    Arscol  := lightmagenta;
    BackGround := blue;
    textmode(3);
    textbackground(background);

    clrscr;
    Quit := false;
    textcolor(yellow);
    gotoxy(30,5);write('SAVE A SPECTRUM');
    test := false;
    tex;
    specindic(localname,Spoint);
    spec := Spoint^;
```

```
repeat
  gotoxy(5,6);write('Current Directory :                                    ');
  ans:gotoxy(27,8);write(Current_dir);
  gotoxy(5,25);write('  Alt D  to change the directory');
  gotoxy(5,10);write('Spectrum          : ');
  ans:write(localname);
  DefaultName := localname + '.dat';
  gotoxy(5,12);write( DOS-Filename >',defaultName,'       :
      ');
  Gread(40,12,error);text;
  if Ganswer.typ.esc then
  begin
    if Ganswer.typ.chr then
    begin
      if Ganswer.chr = Chr(32) then Change_dir
      else begin
        quit := true;
        MainCh := Ganswer.chr;
      end;
    end
    else begin
      MainCh := chr(27);
      Quit := true;
    end;
  end
  else begin
    answer := Ganswer.str;
    if length(Ganswer.str)=0 then answer := defaultName;
    if (CountChr(answer,'\') = 0) and (length(current_dir) > 0)
                    then Answer := Current_dir + '\' + Answer;
    if CountChr(Answer,'.') = 0 then
    begin
      Nname := Answer + '.dat';
      If not exist(Nname) then
      begin
        Namefile := Nname;
        Test := true;
      end;
    end
    else begin
      if not exist(answer) Then
      begin
        NameFile := Answer;
        Test := true;
      end;
    end;
    count := count + 1;
    if Count > 5 then quit := true;
  end;
until (test or quit);
if test then
begin
  assign(outfile,NameFile);rewrite(outfile);
  count := round(1 spec.increment);
  writeln(outfile,spec.measnumber,'  ',0.,'  ',count);
  writeln(outfile,spec.graph.title);
  writeln(outfile,spec.xmin + spec.increment*(spec.measnumber-1):7:1,'  ,sp
ec.xmin-spec.increment:7:1);
  writeln(outfile,10.0:6:2,'  ',0.0:6:2);
  x := round(10*spec.xmin)/10;
  point1 := spec.Measure1;
  point2 := spec.Measure2;
  j := 0;
  for i := 1 to spec.MeasNumber do
  begin
    j := j+1;
    y := point1 .data[j];
    z := point2 .data[j];
    writeln(outfile,x:5:1,y:7:3,z:7:3);
    if j = DataArraySize then
    begin
      j:=0;
      point1 := point1 .datalink;
      point2 := point2 .datalink;
    end;
    x := x + spec.increment;
  end;
  close(outfile);
end;
end;
```

```
{-------------------------------------------------------------------------}
procedure Menu1(LineNumber : integer;Title : name;Ch    ol : boolean;var M : Men
uTab;
                              var answer : integer;var i1,i2 : integer;var error
: boolean);

{
      THIS PROCEDURE PRINTS A MENU WITH A TITLE AND LineNumber LINES
       AND RETURNS THE INDEX OF THE MENU LINE CHOSEN BY THE USER

}


   var   i,j,n,firstlet  : integer;
         quit       : boolean;
         First,Ch,ch1   : char;

   begin
     Background := blue;
     TextCol := White;
     Anscol := White;
     FirstLet := yellow;
     i1:=0;i2 :=0;
     error := false;
     quit := false;
     for i := 1 to LineNumber do
       M[i].Title := UpStyle(M[i].Title);
     repeat
       WriteTitle(Title,30);

    {       WRITE THE 7 FIRST ITEMS     }

       n := min(7,LineNumber);
       for i:= 1 to n   do
       begin
         gotoxy(5,6+2*i);
         First := chr(64+i);
         Textcolor(FirstLet);
         write(First,'.');
         Tex;
         write(M[i].Title);
         if M[i].TypeL <> 0 then
         begin
           gotoxy(25,6+2*i);write(' :  ');
           if M[i].TypeL = 1 then
           begin
             if M[i].bool then writeln('On') else writeln('Off');
           end
           else begin
             if M[i].TypeL = 2 then writeln(M[i].Re:5:2)
                               else writeln(M[i].str);
           end;
         end;
       end;

{        WRITE THE OTHER ITEMS         }

       for i:= 1 to (LineNumber-7) do
       begin
         gotoxy(45,6+2*i);
         First := chr(64+2*i);
         Textcolor(FirstLet);
         write(First,'. ');
         Tex;
         write(M[i+7].Title);
         if M[i+7].TypeL <> 0 then
         begin
           gotoxy(65,6+2*i);write(' :  ');
           if M[i+7].TypeL = 1 then
           begin
             if M[i+7].bool then writeln('On') else writeln('Off');
           end
           else begin
             if M[i+7].TypeL = 2 then writeln(M[i+7].Re:5:2)
                                 else writeln(M[i+7].str);
           end;
         end;
       end;
```

```
{        USER INPUT   }

      gotoxy(5,25);
      write('Hit The Letter Of Your Choice : ');
      read(Kbd,Ch);
      if ord(ch) = 27 then
      begin
        error := true;
        i1 := 27;
        if (not KeyPressed) then i2 := 0
        else begin
          read(Kbd,ch1);
          i2 := ord(ch1);
        end;
      end
      else begin
        if CharBool  then
        begin
          if ord(upcase(ch))=65 then
          begin
            i1 :=13;
            error := true;
          end
          else i1 := 0;
        end
        else i1 := 0;
      end;

      gotoxy(5,25);
      write('                                                  ');

{     MODIFICATIONS OF THE VALUES IN THE MENU   }

      i := ord(Upcase(Ch))-64; j := trunc(i/7.5);
      if (i<1)or(i>LineNumber) then
      begin
        error := true;
      end
      else begin
        if M[i].TypeL <> 0 then
        begin
          if M[i].TypeL = 1 then
          begin
            M[i].bool := not M[i].bool;
            gotoxy(28+40*j,6+2*(i-7*j));write('                ');gotoxy(29+40*j,6
+2*(i-7*j));
            if M[i].bool then write('On ') else write('Off');
          end
          else begin
            gotoxy(28+40*j,6+2*(i-7*j));write('                    ');gotoxy(29+40
*j,5+2*(i-7*j));
            if M[i].TypeL = 2 then read(M[i].re);
            if M[i].TypeL = 3 then read(M[i].str);
          end;
        end
        else quit := true;
      end;
    until ( error or quit );
    answer := ' ';
  end;

{-------------------------------------------------------------------------}
```

```
{-------------------------------------------------------------------------}

   procedure AxisDef(var MainCh : char;var axis : AxDef);
{
              THIS PROCEDURE ALLOWS THE USER TO CHANGE
              THE AXIS USED IN THE PLOTTING PROCEDURE
}

   var   answer,i1,i2 : integer;
         Tab          : menutab;
         title        : name;
         error        : boolean;
         Ch           : char;

 begin
   MainCh := Chr(27);
   title := 'PARAMETERS OF THE AXIS ';

   tab[1].title := 'minimum of x';
   tab[1].typeL := 2;
   tab[1].re    := axis.xmin;

   tab[2].title := 'maximum of x';
   tab[2].typeL := 2;
   tab[2].Re    := axis.xmax;

   tab[3].title :=  x-axis step';
   tab[3].typeL := 2;
   tab[3].re    := Axis.deltax;

   tab[4].title := 'x-axis label';
   tab[4].typeL := 3;
   tab[4].str   := axis.labelx;

   tab[5].title :=  Title';
   tab[5].typeL := 3;
   tab[5].str   := Axis.title;

   tab[6].title := 'y-axis label';
   tab[6].typeL := 3;
   tab[6].str   := Axis.labely;

   tab[7].title := 'y-axis step';
   tab[7].typeL := 2;
   tab[7].re    := axis.deltay;

   tab[8].title := 'minimum of y';
   tab[8].typeL := 2;
   tab[8].re    := axis.ymin;

   tab[9].title := 'maximum of y';
   tab[9].typeL := 2;
   tab[9].re    := axis.ymax;

   menu1(9,title,false,tab,answer,I1,I2,error);

   Axis.xmin    :=tab[1].re;
   Axis.xmax    :=tab[2].re;
   Axis.deltax  :=tab[3].re;
   Axis.labelx  :=tab[4].str;
   Axis.title   :=tab[5].str;
   Axis.labely  :=tab[6].str;
   Axis.deltay  :=tab[7].re;
   Axis.ymin    :=tab[8].re;
   Axis.ymax    :=tab[9].re;
   if i1 = 27 then
     if i2 <> 0 then MainCh := Chr(i2);
 end;

{-------------------------------------------------------------------------}
```

```
{-------------------------------------------------------------------------}

procedure plotcurve(Ipoint : Pointer);

{      THIS PROCEDURE PLOTS THE SPECTRUM AT WHICH IPOINT POINTS
       AND EVENTUALLY THE LASER INTENSITY                              }

var    i,j,x1,x2,y1,y2,z1,z2   :  integer;
       x,Xscale,Xshift,Yscale,Yshift,PowScale :  real;
       point1,point2 : Dpoint;

   begin
     Xscale := Ipoint^.Graph.Xscale;
     Xshift := Ipoint^.Graph.Xshift;
     Yscale := Ipoint^.Graph.Yscale;
     Yshift := Ipoint^.Graph.Yshift;
     PowScale:= Ipoint^.Graph.PowScale;
     Point1 := Ipoint^.measure1;
     point2 := Ipoint^.measure2;
     if (not Ipoint^.Graph.Power) or (Ipoint^.measure2 = nil) then
     begin
       j :=1;
       x1 := trunc(((Ipoint^.Xmin+XShift)*XScale-XOrig)*XScreenScale);
       y1 := trunc(((Point1^.data[j]+YShift)*YScale-YOrig)*YScreenScale);
       i  := 1;
       repeat
         i  := i + 1;
         j  := j + 1;
         x  := Ipoint^.xmin + Ipoint^.increment*(i-1);
         x2 := trunc(((x+XShift)*XScale-XOrig)*XScreenScale);
         y2 := trunc(((Point1^.data[j]+YShift)*YScale-YOrig)*YScreenScale);
         if(x1 > 40 ) then  draw(x1,y1,x2,y2,1);
         x1 := x2;y1 := y2;
         if j = DataArraySize then
         begin
           j := 0;
           point1 := point1 .datalink;
         end;
       until (i = Ipoint^.MeasNumber ) or ( x1 > 600 );
     end
     else begin
       j :=1;
       if point1 = nil then write(' point1 = nil');
       if point2 = nil then write(' point2 = nil');
       x1 := trunc(((Ipoint^.Xmin+XShift)*XScale-XOrig)*XScreenScale);
       y1 := trunc(((Point1 .data[j]+YShift)*YScale-YOrig)*YScreenScale);
       z1 := trunc(((Point2^.data[j])*Yscale*Powscale-YOrig)*YScreenScale);
       i  :=1;
       repeat
         i  := i + 1;
         j  := j + 1;
         x  := Ipoint^.xmin + Ipoint .increment*(i-1);
         x2 := trunc(((x+XShift)*XScale-XOrig)*XScreenScale);
         y2 := trunc(((Point1^.data[j]+YShift)*YScale-YOrig)*YScreenScale);
         z2 := trunc(((Point2^.data[j])*Yscale*Powscale-YOrig)*YScreenScale);
         if(x1 > 40 ) then
         begin
           draw(x1,y1,x2,y2,1);
           draw(x1,z1,x2,z2,1);
         end;
         x1 := x2;y1 := y2;z1 := z2;
         if j = DataArraySize then
         begin
           j       := 0;
           point1 := point1 .datalink;
           point2 := point2 .datalink;
         end;
       until (i = Ipoint .MeasNumber ) or ( x1   600 );
     end;
   end;

{-------------------------------------------------------------------------}
```

```
{-----------------------------------------------------------------------}

procedure SpecPlot(var MainCh : char);

{       THIS PROCEDURE PLOTS THE SPECTRA
        DEFINED BY CURVENUM                             }

 const PlotColor = LightGreen;

 var    j : integer;
        i : pointer;
        ch    : char;
        quit,ex,cond : boolean;

      begin
        MainCh := Chr(27);
        Quit := false;
        repeat
          Hires;
          Hirescolor(plotcolor);
          PlotAxis(Axis);
          For j := 1 to 9  do
          begin
            i := curvenum[j];
            if i <> nil then
            begin
              If i^.Graph.OnOff then Plotcurve(i);
            end;
          end;
          repeat
            ex := true;
            read(Kbd,Ch);
            if number(ch) then
            begin
              i:=curvenum[value(ch)];
              if i <> nil then
              begin
                i^.Graph.OnOff := not i^.Graph.OnOff;
                if i^.graph.OnOff then
                begin
                  ex := false;
                  Plotcurve(i);
                end;
              end;
            end;
          until(ex);
          if letter(ch) then
          begin
            for j := 1 to 9 do
            begin
              if curvenum[j] <> nil then curvenum[j]^.graph.onoff := false;
            end;
          end;
          if ord(ch) = 27 then
          begin
             quit := true;
             if keypressed then
             begin
               read(Kbd,ch);
               MainCh := ch;
             end
             else mainch := chr(120);
          end;
        Until(quit);
      end;

 {-----------------------------------------------------------------------}
```

```
{-- ... ---------- ... ----------}
    procedure SpecList(var MainCh : char);

{
                LISTS THE SPECTRA AND ALLOWS THE USER TO SAVE OR DELETE
                A SPECTRUM OR TO READ THE .TEX FILE ASSOCIATED WITH THE
                SPECTRUM                                                }

    var  i,j,flag    : integer;
         ch,ch1,ch2 : char;
         point : pointer;
         exit  : boolean;
         localname : name;
         Str       : Bigstring;
         Ver$      : Name;
{-------------------------------------------------------------------------}

       procedure ListSave(point : pointer;str : Bigstring;var j : integer);
{
               SAVES THE SPECTRA SELECTED BY THE USER
}
         begin
           j := j+1;
           if point^.link <> nil then Listsave(point^.link,str,j);
           j := j - 1;
           if str[j] = '*' then specsave(point^.localname);
         end;

{-------------------------------------------------------------------------}
        procedure ListDel(point : pointer;str : Bigstring;var j : integer);
{
              DELETES THE SPECTRA SELECTED BY THE USER
}

        begin
           j := j+1;
           if point^.link <> nil then ListDel(point^.link,str,j);
           j := j - 1;
           if str[j] = '*' then specdel(point^.localname);
         end;

{-------------------------------------------------------------------------}
        procedure ListDetail(point : pointer;str : Bigstring;var j : integer);
{
            WRITES THE .TEX FILE ASSOCIATED WITH SPECTRA SELECTED
            BY THE USER IF IT EXISTS
}

        begin
        end;

{-------------------------------------------------------------------------}

        procedure prchoice(i : integer; blinkflag : boolean);

{         USED FOR THE GRAPHIC        }

       begin
          if blinkflag then
          begin
            textbackground(textcol);
            textcolor(background + blink);
          end
          else begin
            textbackground(textcol);
            textcolor(background);
          end;
          gotoxy( 70*(i-1) + 1,25);
          case i of
            1 : write('   SAVE    ');
            2 : write('  DELETE   ');
            3 : write('  DETAIL   ');
            4 : write('   EXIT    ');
          end;
        end;

{-------------------------------------------------------------------------}

{           BEGINNING OF SPECLIST CODE                  }
```

```
{            BEGINNING OF SPECLIST CODE

      begin
        Textcol := white;
        Background := blue;
        MainCh := Chr(120);
        Textmode(3);
        textbackground(background);
        Clrscr;
        textcolor(yellow);
        gotoxy(30,5);write('LISTING OF THE SPECTRA');
        gotoxy(2,7);write('LOCAL NAME');
        gotoxy(15,7);write('MEAS. NUM');
        gotoxy(27,7);write('TYPE');
        gotoxy(40,7);write('TITLE');
        i := 8;
        Point := head;
        while( point <> nil) do
        begin
          i:= i+1;
          gotoxy(5,i);write(point^.localname);
          gotoxy(18,i);write(point^.Measnumber);
          gotoxy(28,i);write(point^.Measure1 .Datatype);
          gotoxy(33,i);write(point^.Graph.Title);
          point := point^.link;
        end;
        prchoice(1,false);
        prchoice(2,false);
        prchoice(3,false);
        prchoice(4,true);
        j := 4;
        exit := false;
        repeat
          read(kbd,Ch);
          if (Ord(Ch) = 27) then
          begin
            if not keypressed then exit := true
            else begin
              read(kbd,ch);
              prchoice(j,false);
              if Ord(ch) = 75 then j := j-1;
              if Ord(ch) = 77 then j := j+1;
            end;
          end
          else begin
            if ord(ch)=13 then exit := true;
          end;
          if j = 5 then j := 1;
          if j = 0 then j := 4;
          prchoice(j,true);
        until (exit);
        textbackground(background);
        If j <> 4 then
        begin
          flag := j;
          if flag = 1 then verb := 'save';
          if flag = 2 then verb := 'delete';
          if flag = 3 then verb := 'detail';
          ch1 := Chr(24);ch2 := Chr(25);
          str :='                                                                    ';
          gotoxy(5,24);
          write( Move The Cursor With ',ch1,'and ',ch2,', Hit Return to  ,verb,
 , Esc to quit );
          j := 9;exit := false;
          gotoxy(4,j);
          repeat
            read(kbd,Ch);
            if (Ord(Ch) = 27) then
            begin
              if not keypressed then exit := true
              else begin
                read(kbd,ch);
                if Ord(ch) = 72 then j := j-1;
                if Ord(ch) = 80 then j := j+1;
              end;
            end
            else begin
              if ord(ch)=13 then
              begin
                gotoxy(4,j);
                if str[j]='*' then
                begin
                  write(  );
                  str[j] := ' ';
                end
```

```
          else begin
            if ord(ch)=13 then
            begin
              gotoxy(4,j);
              if str[j]='*' then
              begin
                write(' ');
                str[j] := ' ';
              end
              else begin
                write('*');
                str[j] := '*';
              end;
              j := j +1;
            end;
          end;
          if j = 8 then j := i;
          if j =i+1 then j:=9;
          gotoxy(4,j);
        until (exit);
        j := 9;
        if flag = 1  then Listsave(head,str,j);
        if flag = 2  then Listdel(head,str,j);
        if flag = 3  then Listdetail(head,str,j);
      end
      else begin
        if Ord(ch) = 27 then
        begin
          if keypressed then read(kbd,ch);
          Mainch:=ch;
        end;
      end;
    end;
```

```
{-------------------------------------------------------------------------}
```

```
{-------------------------------------------------------------------------}

procedure Characterize(Icurve : integer;var MainCh : char);

{    THIS PROCEDURE ALLOWS TO DEFINE THE CHARACTERISTICS
       OF THE PLOTTING OF A SPECTRUM                                      }

  var   answer,I1,I2 : integer;
        indic : pointer;
        Tab   : menutab;
        Ch    : char;
        Title : name;
        Error : boolean;

begin
  I1 := 13;
  MainCh := Chr(27);
  while (I1 = 13) do
  begin
    Indic := ArrayNum[Icurve];
    if indic <> nil then
    begin
      Title := 'PARAMETERS OF THE PLOT ';

      Tab[1].title :=  Name';
      Tab[1].TypeL := 'S';
      Tab[1].str   := indic^.localname;
```

```
      tab[2].title := 'Spectrum';
      tab[2].typeL := 1;
      tab[2].bool  := indic^.graph.onoff;

      tab[3].title := 'Power';
      tab[3].typeL := 1;
      tab[3].bool  := indic^.graph.power;

      tab[4].title := 'Title';
      tab[4].typeL := 3;
      tab[4].str   := indic^.graph.title;

      tab[5].title := 'Shift on X axis';
      tab[5].typeL := 2;
      tab[5].re  := indic^.graph.Xshift;

      tab[6].title := 'Scaling of X axis';
      tab[6].typeL := 2;
      tab[6].re  := indic^.graph.Kscale;

      tab[7].title := 'Shift on Y axis';
      tab[7].typeL := 2;
      tab[7].re  := indic^.graph.Yshift;

      tab[8].title := 'Scaling of Y axis';
      tab[8].typeL := 2;
      tab[8].re  := indic^.graph.Yscale;

      tab[9].title := 'Power Scale';
      tab[9].typeL := 2;
      tab[9].re  := indic^.graph.PowScale;

      menu1(9,title,true,tab,answer,I1,I2,error);

      if Valid(tab[1].str) then
      begin
        indic^.graph.OnOff   := tab[2].bool;
        indic^.graph.power   := tab[3].bool;
        indic^.graph.title   := tab[4].str;
        indic^.graph.xshift  := tab[5].re;
        indic^.graph.xscale  := tab[6].re;
        indic^.graph.yshift  := tab[7].re;
        indic^.graph.yscale  := tab[8].re;
        indic^.graph.Powscale:= tab[9].re;
        specindic(tab[1].str,curvenum[Icurve]);
      end;

      if i1 = 27 then
      if i2 <> 0 then  MainCh := Chr(i2);
    end
    else i1 := 0;
  end;
end;

{-------------------------------------------------------------------------------
```

```
{

    procedure convert(var spec1,spec2 : spectrum);

{           THIS PROCEDURE CHECKS IF THE TWO SPECTRA ...
            THE SAME FORMAT AND IF NOT CHANGES THE FORMAT
            OF ONE OF THE SPECTRA IN ORDER FOR THEM TO
            HAVE THE SAME FORMAT

}

    var point01,point02,point21,point22 : Dpoint;
        spec                : Spectrum;
        x1,x2,sum1,sum2 : real;
        inc,xmin,xmax       : real;
        i,j,k,n1            : integer;

    begin
      if spec1.increment = spec2.increment then
      begin

{ First Case : SAME INCREMENT     }

          if spec1.xmin < spec2.xmin then
          begin
            spec  := spec2;
            spec2 := spec1;
            spec1 := spec;
          end;
          spec := spec2;
          new(spec2.measure1);
          new(spec2.measure2);
          spec2.xmin := spec1.xmin;
          i:=0;j := 0;k := 0; x1 := spec.xmin; xmax := spec.xmin+(spec.measnumbe
r-1)*spec.increment;
          point01:= spec.measure1;point02 := spec.measure2;
          point21:= spec2.measure1;point22 := spec2.measure2;
          repeat
            j := j +1;
            x1:= x1 + spec1.increment;
            if x1 >= spec1.xmin then
            begin
              i := i+1;
              k:=k+1;
              if x1   xma  then
              begin
                point21 .data[k]:=point01^.data[j];
                point22 .data[k]:=point02^.data[j];
              end
              else begin
                point21^.data[k]:=0;
                point22 .data[k]:=0;
              end;
              if k = DataArraySize then
              begin
                k:=0;
                new(point21^.datalink);
                point21 := point21^.datalink;
                point21 .datalink := nil;
                new(point22^.datalink);
                point22 := point22^.datalink;
                point22^.datalink := nil;
              end;
            end;
            if j = DataArraySize then
            begin
              j:=0;
              point01 := point01^.datalink;
              point02 := point02^.datalink;
            end;
          until(i  = spec1.measnumber);
        end
        else begin

{ Second Case : DIFFERENT INCREMENT }

          if spec1.increment > spec2.increment then
          begin
            spec  := spec2;
            spec2 := spec1;
            spec1 := spec;
          end;
          spec := spec2;

  { INITIALIZATION OF SPEC2 }
```

```
{ INITIALIZATION OF SPEC2 }

        new(spec2.measure1);
        new(spec2.measure2);
        inc  := spec1.increment;
        xmin := spec1.xmin;
        xmax := spec1.xmin + (spec1.MeasNumber - 1)*inc;
        j:=0;point21 := spec2.measure1;point22 := spec2.measure2;
        for i:= 1 to spec1.MeasNumber do
        begin
          j := j+1;
          point21^.data[j] := 0;
          point22^.data[j] := 0;
          if j = DataArraySize then
          begin
            j := 0;
            new(point21^.datalink);
            new(point22^.datalink);
            point21 := point21^.datalink;
            point22 := point22^.datalink;
          end;
        end;
        spec2.xmin := spec1.xmin;
        spec2.increment := inc;
        spec2.localname := '';

  { COMPUTATION OF SPEC2  }

        j := 0;k := 1; x1 := spec.xmin; x2 := xmin;sum1 := 0;sum2 := 0;n1:= 0;
        point01:= spec.measure1;point02 := spec.measure2;
        point21:= spec2.measure1;point22 := spec2.measure2;
        while (x2 + inc/2 < x1) do                    { initialization}
        begin                                        { of k   }
          k  := k+1;
          x2 := xmin + (k-1)*inc;
        end;
        for i := 1 to spec.MeasNumber do
        begin
          j := j + 1;
          x1:= spec.xmin + (i-1)*spec.increment;
          if (x1 >= (xmin-inc/2))and (x1 <=(xmax + inc/2)) then
          begin
            if (x1 >= x2-inc/2) and (x1 <= x2+inc/2) then
            begin
              sum1 := sum1 + point01^.data[j];
              sum2 := sum2 + point02^.data[j];
              n1 := n1 +1;
            end
            else begin
              if n1  <>0 then
              begin
                point21^.data[k]:=sum1/n1;
                point22^.data[k]:=sum2/n1;
                sum1:=point01^.data[j];
                sum2:=point01^.data[j];
                n1 := 1;
                x2 := x2 + inc;
                k  := k + 1;
                if k = DataArraySize then
                begin
                  k:=0;
                  point21 := point21^.datalink;
                  point22 := point22 .datalink;
                end;
              end;
            end;
          end;
          if j = DataArraySize then
          begin
            j:=0;
            point01 := point01^.datalink;
            point02 := point02^.datalink;
          end;
        end;
      end;
    end;
```

```
{ ---------------------------------------------------------------------- }
{                                                                        }
{                              FFTB2.INC                                 }
{                                                                        }
{       Turbo Pascal Numerical Methods Toolbox                           }
{       (C) Copyright 1986 Borland International.                        }
{                                                                        }
{       Version Date: 26 January 1987                                    }
{                                                                        }
{ ---------------------------------------------------------------------- }

procedure TestInput(NumPoints     : integer;
                var NumberOfBits : byte;
                var Error        : byte);

{------------------------------------------------------------------------}
{- Input: NumPoints                                                     -}
{- Outputs: NumberOfBits, Error                                         -}
{-                                                                      -}
{- This procedure checks the input.  If the number of points            -}
{- (NumPoints) is less than two or is not a multiple of two             -}
{- then an error is returned.  NumberOfBits is the number of            -}
{- bits necessary to represent NumPoints in binary (e.g. if             -}
{- NumPoints = 16, NumberOfBits = 4).                                   -}
{------------------------------------------------------------------------}

type
  ShortArray = array[1..13] of integer;

var
  Term : integer;

const
  PowersOfTwo : ShortArray = (2, 4, 8, 16, 32, 64, 128, 256,
                              512, 1024, 2048, 4096, 8192);

begin
  Error := 2; { Assume NumPoints not a power of two  }
  if NumPoints < 2 then
    Error := 1;      { NumPoints < 2  }
  Term := 1;
  while (Term <= 13) and (Error = 2) do
  begin
    if NumPoints = PowersOfTwo[Term] then
    begin
      NumberOfBits := Term;
      Error := 0;  { NumPoints is a power of two }
    end;
    Term := Succ(Term);
  end;
end; { procedure TestInput }

procedure MakeSinCosTable(NumPoints : integer;
                      var SinTable  : TVectorPtr;
                      var CosTable  : TVectorPtr);

{------------------------------------------------------------------------}
{- Input: NumPoints                                                     -}
{- Output: SinTable, CosTable                                           -}
{-                                                                      -}
{- This procedure fills in a table with sin and cosine                  -}
{- values.  It is faster to pull data out of this                       -}
{- table than it is to calculate the sines and cosines.                 -}
{------------------------------------------------------------------------}

var
  RealFactor, ImagFactor : real;
  Term : integer;
  TermMinus1 : integer;
  UpperLimit : integer;

begin
  RealFactor :=  Cos(2 * Pi / NumPoints);
  ImagFactor := -Sqrt(1 - Sqr(RealFactor));
  CosTable [0] := 1;
  SinTable [0] := 0;
  CosTable [1] := RealFactor;
  SinTable [1] := ImagFactor;
  UpperLimit := NumPoints shr 1 - 1;
  for Term := 2 to UpperLimit do
  begin
    TermMinus1 := Term - 1;
    CosTable^[Term] := CosTable^[TermMinus1] * RealFactor -
                       SinTable [TermMinus1] * ImagFactor;
    SinTable^[Term] := CosTable [TermMinus1] * ImagFactor +
                       SinTable [TermMinus1] * RealFactor;
  end;
end; { procedure MakeSinCosTable }
```

```
procedure FFT(NumberOfBits : byte;
              NumPoints    : integer;
              Inverse      : boolean;
          var XReal        : TNvectorPtr;
          var XImag        : TNvectorPtr;
          var SinTable     : TNvectorPtr;
          var CosTable     : TNvectorPtr);

{-------------------------------------------------------}
{- Input: NumberOfBits, NumPoints, Inverse, XReal,     -}
{-        XImag, SinTable, CosTable                    -}
{- Output: XReal, XImag                                -}
{-                                                     -}
{- This procedure implements the actual fast Fourier  -}
{- transform routine.  The vector X, which must be     -}
{- entered in bit-inverted order, is transformed in    -}
{- place.  The transformation uses the Cooley-Tukey    -}
{- algorithm.                                          -}
{-------------------------------------------------------}

const
  RootTwoOverTwo = 0.707106781186548;

var
  Term : byte;
  CellSeparation : integer;
  NumberOfCells : integer;
  NumElementsInCell : integer;
  NumElInCellLess1 : integer;
  NumElInCellSHR1 : integer;
  NumElInCellSHR2 : integer;
  CosTerm, SumTerm, DifTerm : real;
  Element : integer;
  CellElements : integer;
  ElementInNextCell : integer;
  Index : integer;
  RealDummy, ImagDummy : real;
  Dummy1, Dummy2 : real;

procedure BitInvert(NumberOfBits : byte;
                    NumPoints    : integer;
                var XReal        : TNvectorPtr;
                var XImag        : TNvectorPtr);

{-------------------------------------------------------}
{- Input: NumberOfBits, NumPoints                      -}
{- Output: XReal, XImag                                -}
{-                                                     -}
{- This procedure bit inverts the order of data in the -}
{- vector X.  Bit inversion reverses the order of the  -}
{- binary representation of the indices; thus 2 indices-}
{- will be switched.  For example, if there are 16 points, -}
{- Index 7 (binary 0111) would be switched with Index 14   -}
{- (binary 1110).  It is necessary to bit invert the order -}
{- of the data so that the transformation comes out in the -}
{- correct order.                                      -}
{-------------------------------------------------------}

var
  Term : integer;
  Invert : integer;
  Hold : real;
  NumPointsDiv2, M : integer;

begin
  NumPointsDiv2 := NumPoints shr 1;
  Invert := 0;
  for Term := 0 to NumPoints - 2 do
  begin
    if Term < Invert then   { Switch these two indices }
    begin
      Hold := XReal^[Invert];
      XReal^[Invert] := XReal^[Term];
      XReal^[Term] := Hold;
      Hold := XImag^[Invert];
      XImag^[Invert] := XImag [Term];
      XImag^[Term] := Hold;
    end;
```

```
    K := NumPointsDiv2;
    while K <= Invert do
    begin
      Invert := Invert - K;
      K := K shr 1;
    end;
    Invert := Invert + K;
  end;
end; { procedure BitInvert }

begin { procedure FFT }
  { The data must be entered in bit inverted order }
  { for the transform to come out in proper order  }
  BitInvert(NumberOfBits, NumPoints, XReal, XImag);

  if Inverse then
    { Conjugate the input  }
    for Element := 0 to NumPoints - 1 do
      XImag^[Element] := -XImag^[Element];

  NumberOfCells := NumPoints;
  CellSeparation := 1;
  for Term := 1 to NumberOfBits do
  begin
    { NumberOfCells halves; equals 2^(NumberOfBits - Term)  }
    NumberOfCells := NumberOfCells shr 1;
    { NumElementsInCell doubles; equals 2^(Term-1)  }
    NumElementsInCell := CellSeparation;
--  { CellSeparation doubles; equals 2 Term  }
    CellSeparation := CellSeparation SHL 1;
    NumElInCellLess1 := NumElementsInCell - 1;
    NumElInCellSHR1 := NumElementsInCell shr 1;
    NumElInCellSHR2 := NumElInCellSHR1 shr 1;

    { Special case: RootOfUnity = EXP(-i 0)  }
    Element := 0;
    while Element < NumPoints do
    begin
      { Combine the X[Element] with the element in  }
      { the identical location in the next cell     }
      ElementInNextCell := Element + NumElementsInCell;
      RealDummy := XReal^[ElementInNextCell];
      ImagDummy := XImag^[ElementInNextCell];
      XReal^[ElementInNextCell] := XReal^[Element] - RealDummy;
      XImag^[ElementInNextCell] := XImag^[Element] - ImagDummy;
      XReal^[Element] := XReal^[Element] + RealDummy;
      XImag^[Element] := XImag^[Element] + ImagDummy;
      Element := Element + CellSeparation;
    end;
```

```
for CellElements := 1 to NumElInCellSHR2 - 1 do
begin
  Index := CellElements * NumberOfCells;
  CosTerm := CosTable^[Index];
  SumTerm := CosTerm + SinTable^[Index];
  DifTerm := SinTable^[Index] - CosTerm;
  Element := CellElements;

  while Element < NumPoints do
  begin
    { Combine the X[Element] with the element in  }
    { the identical location in the next cell     }
    ElementInNextCell := Element + NumElementsInCell;
    Dummy1 := (XReal^[ElementInNextCell] +
    XImag [ElementInNextCell]) * CosTerm;
    Dummy2 := XReal^[ElementInNextCell] * DifTerm;
    RealDummy := Dummy1 - XImag^[ElementInNextCell] * SumTerm;
    ImagDummy := Dummy1 + Dummy2;;
    XReal [ElementInNextCell] := XReal^[Element] - RealDummy;
    XImag [ElementInNextCell] := XImag^[Element] - ImagDummy;
    XReal [Element] := XReal^[Element] + RealDummy;
    XImag [Element] := XImag [Element] + ImagDummy;
    Element := Element + CellSeparation;
  end;
end; { for }

{ Special case: RootOfUnity = EXP(-i PI/4)  }
if Term = 2 then
begin
  Element := NumElInCellSHR2;
  while Element < NumPoints do
  begin
    { Combine the X[Element] with the element in  }
    { the identical location in the next cell     }
    ElementInNextCell := Element + NumElementsInCell;
    RealDummy := RootTwoOverTwo * (XReal [ElementInNextCell] +
                                   XImag^[ElementInNextCell]);
    ImagDummy := RootTwoOverTwo * (XImag^[ElementInNextCell] -
                                   XReal^[ElementInNextCell]);
    XReal [ElementInNextCell] := XReal^[Element] - RealDummy;
    XImag [ElementInNextCell] := XImag^[Element] - ImagDummy;
    XReal^[Element] := XReal^[Element] + RealDummy;
    XImag^[Element] := XImag^[Element] + ImagDummy;
    Element := Element + CellSeparation;
  end;
end;

for CellElements := NumElInCellSHR2 + 1 to NumElInCellSHR1 - 1 do
begin
  Index := CellElements * NumberOfCells;
  CosTerm := CosTable^[Index];
  SumTerm := SinTable^[Index] + CosTerm;
  DifTerm := SinTable [Index] - CosTerm;
  Element := CellElements;
  while Element < NumPoints do
  begin
    { Combine the X[Element] with the element in  }
    { the identical location in the next cell     }
    ElementInNextCell := Element + NumElementsInCell;
    Dummy1 := (XReal [ElementInNextCell] +
               XImag [ElementInNextCell]) * CosTerm;
    Dummy2 := XReal [ElementInNextCell] * DifTerm;
    RealDummy := Dummy1 - XImag^[ElementInNextCell] * SumTerm;
    ImagDummy := Dummy1 + Dummy2;;
    XReal [ElementInNextCell] := XReal [Element] - RealDummy;
    XImag [ElementInNextCell] := XImag^[Element] - ImagDummy;
    XReal [Element] := XReal [Element] + RealDummy;
    XImag [Element] := XImag [Element] + ImagDummy;
    Element := Element + CellSeparation;
  end;
end; { for }
```

```
{ Special case: RootOfUnity = EXP(-i PI/2)  }
if Term <> 1 then
begin
  Element := NumElInCellSHR1;
  while Element < NumPoints do
  begin
    { Combine the X[Element] with the element in  }
    { the identical location in the next cell     }
    ElementInNextCell := Element + NumElementsInCell;
    RealDummy :=  XImag^[ElementInNextCell];
    ImagDummy := -XReal^[ElementInNextCell];
    XReal^[ElementInNextCell] := XReal^[Element] - RealDummy;
    XImag^[ElementInNextCell] := YImag^[Element] - ImagDummy;
    XReal^[Element] := XReal^[Element] + RealDummy;
    XImag^[Element] := XImag^[Element] + ImagDummy;
    Element := Element + CellSeparation;
  end;
end;

for CellElements := NumElInCellSHR1 + 1 to
                    NumElementsInCell - NumElInCellSHR2 - 1 do
begin
  Index := CellElements * NumberOfCells;
  CosTerm := CosTable^[Index];
  SumTerm := SinTable^[Index] + CosTerm;
  DifTerm := SinTable^[Index] - CosTerm;
  Element := CellElements;
  while Element < NumPoints do
  begin
    { Combine the Y[Element] with the element in  }
    { the identical location in the next cell     }
    ElementInNextCell := Element + NumElementsInCell;
    Dummy1 := (XReal^[ElementInNextCell] +
               XImag^[ElementInNextCell]) * CosTerm;
    Dummy2 := XReal^[ElementInNextCell] * DifTerm;
    RealDummy := Dummy1 - XImag^[ElementInNextCell] * SumTerm;
    ImagDummy := Dummy1 + Dummy2;;
    XReal^[ElementInNextCell] := XReal^[Element] - RealDummy;
    XImag^[ElementInNextCell] := XImag^[Element] - ImagDummy;
    XReal^[Element] := XReal^[Element] + RealDummy;
    XImag^[Element] := XImag^[Element] + ImagDummy;
    Element := Element + CellSeparation;
  end;
end; { for }
```

```
    { Special case: RootOfUnity = EXP(-i*PI/4)  }
    if Term = 2 then
    begin
      Element := NumElementsInCell - NumElInCellSHR2;
      while Element < NumPoints do
      begin
        { Combine the X[Element] with the element in  }
        { the identical location in the next cell     }
        ElementInNextCell := Element + NumElementsInCell;
        RealDummy := -RootTwoOverTwo * (XReal^[ElementInNextCell] -
                                        XImag^[ElementInNextCell]);
        ImagDummy := -RootTwoOverTwo * (XReal^[ElementInNextCell] +
                                        XImag^[ElementInNextCell]);
        XReal^[ElementInNextCell] := XReal^[Element] - RealDummy;
        XImag^[ElementInNextCell] := XImag^[Element] - ImagDummy;
        XReal^[Element] := XReal^[Element] + RealDummy;
        XImag^[Element] := XImag^[Element] + ImagDummy;
        Element := Element + CellSeparation;
      end;
    end;

    for CellElements := NumElementsInCell - NumElInCellSHR2 + 1 to
                                              NumElInCellLess1 do
    begin
      Index := CellElements * NumberOfCells;
      CosTerm := CosTable^[Index];
      SumTerm := SinTable^[Index] + CosTerm;
      DifTerm := SinTable^[Index] - CosTerm;
      Element := CellElements;
      while Element < NumPoints do
      begin
        { Combine the X[Element] with the element in  }
        { the identical location in the next cell     }
        ElementInNextCell := Element + NumElementsInCell;
        Dummy1 := (XReal^[ElementInNextCell] +
                   XImag^[ElementInNextCell]) * CosTerm;
        Dummy2 := XReal^[ElementInNextCell] * DifTerm;
        RealDummy := Dummy1 - XImag^[ElementInNextCell] * SumTerm;
        ImagDummy := Dummy1 + Dummy2;
        XReal^[ElementInNextCell] := XReal^[Element] - RealDummy;
        XImag^[ElementInNextCell] := XImag^[Element] - ImagDummy;
        XReal^[Element] := XReal^[Element] + RealDummy;
        XImag^[Element] := XImag^[Element] + ImagDummy;
        Element := Element + CellSeparation;
      end;
    end; { for }
  end;

  {-------------------------------------------------}
  {-  Divide all the values of the transformation  -}
  {-  by the square root of NumPoints. If taking the  -}
  {-  inverse, conjugate the output.               -}
  {-------------------------------------------------}

  if Inverse then
    ImagDummy := -1 / Sqrt(NumPoints)
  else
    ImagDummy :=  1 / Sqrt(NumPoints);
  RealDummy := ABS(ImagDummy);
  for Element := 0 to NumPoints - 1 do
  begin
    XReal^[Element] := XReal^[Element] * RealDummy;
    XImag^[Element] := XImag^[Element] * ImagDummy;
  end;
end; { procedure FFT }

{-------------------------------------------------
```

```
{ ...
pr  ...ure RealFFT(NumPoints : integer;
                   Inverse   : boolean;
                   var XReal     : TNvectorPtr;
                   var XImag     : TNvectorPtr;
                   var Error     : byte);

{-  ...                                                                    }
{-                                                                        }
{-     Turbo Pascal Numerical Methods Toolbox                             }
{-     (C) Copyright 1986 Borland International.                          }
{-                                                                        }
{-    Input: NumPoints, Inverse, XReal, XImag,                            }
{-    Output: XReal, XImag, Error                                         }
{-                                                                        }
{-    Purpose:  This procedure uses the complex Fourier transform         }
{-              routine (FFT) to transform real data.  The real data      }
{-              is in the vector XReal.  Appropriate shuffling of indices }
{-              changes the real vector into two vectors (representing    }
{-              complex data) which are only half the size of the original }
{-              vector.  Appropriate unshuffling at the end produces the  }
{-              transform of the real data.                               }
{-                                                                        }
{-  User Defined Types:                                                   }
{-         TNvector = array[0..TNArraySize] of real                       }
{-      TNvectorPtr = ^TNvector                                           }
{-                                                                        }
{- Global Variables:  NumPoints   : integer     Number of data          }
{-                                              points in X               }
{-                    Inverse     : boolean     False => forward transform }
{-                                              True => inverse transform }
{-                    XReal,XImag : TNvectorPtr Data points               }
{-                    Error       : byte        Indicates an error        }
{-                                                                        }
{-             Errors:  0: No Errors                                      }
{-                      1: NumPoints < 2                                  }
{-                      2: NumPoints not a power of two                   }
{-                         (or 4 for radix-4 transforms)                  }
{-                                                                        }
{-    Version Date: 26 January 1987                                       }
{-                                                                        }
{-------------------------------------------------------------------------}

var
  SinTable, CosTable : TNvectorPtr;    { Tables of sine and cosine values  }
  NumberOfBits : byte;                 { Number of bits necessary to       }
                                       { represent the number of points    }

procedure MakeRealDataComplex(NumPoints : integer;
                              var XReal     : TNvectorPtr;
                              var XImag     : TNvectorPtr);

{-------------------------------------------------------------------------}
{- Input: NumPoints, XReal                                               -}
{- Output: XReal, XImag                                                  -}
{-                                                                       -}
{- This procedure shuffles the real data.  There are                     -}
{- 2*NumPoints real data points in the vector XReal.  The                -}
{- data is shuffled so that there are NumPoints complex                  -}
{- data points.  The real part of the complex data is                    -}
{- made up of those points whose original array Index was                -}
{- even; the imaginary part of the complex data is made                  -}
{- up of those points whose original array Index was odd.                -}
{-------------------------------------------------------------------------}

var
  Index, NewIndex : integer;
  DummyReal, DummyImag : TNvectorPtr;

begin
  New(DummyReal);
  New(DummyImag);
  for Index := 0 to NumPoints - 1 do
  begin
    NewIndex := Index shl 1;
    DummyReal^[Index] := XReal^[NewIndex];
    DummyImag^[Index] := XReal^[NewIndex + 1];
  end;
  XReal^ := DummyReal^;
  XImag^ := DummyImag^;
  Dispose(DummyReal);
  Dispose(DummyImag);
end; { procedure MakeRealDataComplex }
```

```
procedure UnscrambleComplexOutput(NumPoints : integer;
                                  var SinTable  : TNvectorPtr;
                                  var CosTable  : TNvectorPtr;
                                  var XReal     : TNvectorPtr;
                                  var XImag     : TNvectorPtr);

{-------------------------------------------------------------}
{- Input: NumPoints, SinTable, CosTable, XReal, XImag        -}
{- Output: XReal, XImag                                      -}
{-                                                           -}
{- This procedure unshuffles the complex transform.          -}
{- The transform has NumPoints elements.  This procedure     -}
{- unshuffles the transform so that it is 2*NumPoints        -}
{- elements long.  The resulting vector is symmetric         -}
{- about the element NumPoints.                              -}
{- Both the forward and inverse transforms are defined       -}
{- with a 1/Sqrt(NumPoints) factor.  Since the real FFT       -}
{- algorithm operates on vectors of length NumPoints/2,      -}
{- the unscrambled vectors must be divided by Sqrt(2).       -}
{-------------------------------------------------------------}

var
  PiOverNumPoints : real;
  Index : integer;
  indexSHR1 : integer;
  NumPointsMinusIndex : integer;
  SymmetricIndex : integer;
  Multiplier : real;
  Factor : real;
  CosFactor, SinFactor : real;
  RealSum, ImagSum, RealDif, ImagDif : real;
  RealDummy, ImagDummy : TNvectorPtr;
  NumPointsSHL1 : integer;

begin
  New(RealDummy);
  New(ImagDummy);
  RealDummy^ := XReal^;
  ImagDummy^ := XImag^;
  PiOverNumPoints := Pi / NumPoints;
  NumPointsSHL1 := NumPoints shl 1;
  RealDummy^[0] := (XReal^[0] + XImag^[0]) / Sqrt(2);
  ImagDummy^[0] := 0;
  RealDummy^[NumPoints] := (XReal^[0] - XImag^[0]) / Sqrt(2);
  ImagDummy^[NumPoints] := 0;
  for Index := 1 to NumPoints - 1 do
  begin
    Multiplier := 0.5 / Sqrt(2);
    Factor := PiOverNumPoints * Index;
    NumPointsMinusIndex := NumPoints - Index;
    SymmetricIndex := NumPointsSHL1 - Index;
    if Odd(Index) then
      begin
        CosFactor :=  Cos(Factor);
        SinFactor := -Sin(Factor);
      end
    else
      begin
        indexSHR1 := Index shr 1;
        CosFactor := CosTable^[indexSHR1];
        SinFactor := SinTable^[indexSHR1];
      end;

    RealSum := XReal^[Index] + XReal^[NumPointsMinusIndex];
    ImagSum := XImag^[Index] + XImag^[NumPointsMinusIndex];
    RealDif := XReal^[Index] - XReal^[NumPointsMinusIndex];
    ImagDif := XImag^[Index] - XImag^[NumPointsMinusIndex];

    RealDummy^[Index] := Multiplier * (RealSum + CosFactor * ImagSum
                         + SinFactor * RealDif);
    ImagDummy^[Index] := Multiplier * (ImagDif + SinFactor * ImagSum
                         - CosFactor * RealDif);
    RealDummy^[SymmetricIndex] :=  RealDummy^[Index];
    ImagDummy^[SymmetricIndex] := -ImagDummy^[Index];
  end;  { for }

  XReal^ := RealDummy^;
  XImag^ := ImagDummy^;
  Dispose(RealDummy);
  Dispose(ImagDummy);
end; { procedure UnscrambleComplexOutput }
```

```
begin { procedure RealFFT }

  { The number of complex data points will  }
  { be half the number of real data points  }
  NumPoints := NumPoints shr 1;
  TestInput(NumPoints, NumberOfBits, Error);
  if Error = 0 then
  begin
    New(SinTable);
    New(CosTable);
    MakeRealDataComplex(NumPoints, XReal, XImag);
    MakeSinCosTable(NumPoints, SinTable, CosTable);
    FFT(NumberOfBits, NumPoints, Inverse, XReal, XImag, SinTable, CosTable);
    UnscrambleComplexOutput(NumPoints, SinTable, CosTable, XReal, XImag);
    NumPoints := NumPoints shl 1;     { The number of complex points  }
                                      { in the transform will be the  }
                                      { same as the number of real    }
                                      { points in input data.         }
    Dispose(SinTable);
    Dispose(CosTable);
  end;
end; { procedure RealFFT }

{-------------------------------------------------------------------------}
```

```
procedure ComplexFFT(NumPoints : integer;
                     Inverse   : boolean;
                 var XReal     : TNvectorPtr;
                 var XImag     : TNvectorPtr;
                 var Error     : byte);

{-------------------------------------------------------------------------}
{-                                                                       -}
{-     Turbo Pascal Numerical Methods Toolbox                            -}
{-     (C) Copyright 1986 Borland International.                         -}
{-                                                                       -}
{-   Input: NumPoints, Inverse, XReal, XImag                             -}
{-   Output: XReal, XImag, Error                                         -}
{-                                                                       -}
{-   Purpose: This procedure performs a fast Fourier transform           -}
{-            of the complex data XReal, XImag. The vectors XReal        -}
{-            and XImag are transformed in place.                        -}
{-                                                                       -}
{  User Defined Types:                                                   -}
{-          TNvector = array[0..TNArraySize] of real                     }
{-          TNvectorPtr =  TNvector                                      -}
{-                                                                       -}
{- Global Variables:  NumPoints : integer      Number of data            }
{-                                             points in X               -}
{-                    Inverse   : BOOLEAN      FALSE => Forward          -}
{-                                                      Transform        -}
{-                                             TRUE => Inverse           -}
{-                                                     Transform         -}
{-                    XReal,                                             -}
{-                    XImag     : TNvectorPtr  Data points               -}
{-                    Error     : byte         Indicates an error        -}
{-                                                                       -}
{-             Errors:  0: No Errors                                     -}
{-                      1: NumPoints < 2                                 -}
{-                      2: NumPoints not a power of two                  -}
{-                                                                       -}
{-   Version Date:  26 January 1987                                      -}
{-                                                                       -}
{-------------------------------------------------------------------------}

var
  SinTable, CosTable : TNvectorPtr;       { Tables of sin and cosine values  }
  NumberOfBits : byte;                    { Number of bits to represent the  }
                                          { number of data points.           }

begin { procedure ComplexFFT }
  TestInput(NumPoints, NumberOfBits, Error);
  if Error = 0 then
  begin
    New(SinTable);
    New(CosTable);
    MakeSinCosTable(NumPoints, SinTable, CosTable);
    FFT(NumberOfBits, NumPoints, Inverse, XReal, XImag, SinTable, CosTable);
    Dispose(SinTable);
    Dispose(CosTable);
  end;
end; { procedure ComplexFFT }

{-------------------------------------------------------------------------}
```

```
{-----------------------------------------------------------------------

  procedure smfft (MeasPtr : Dpoint;istart,iend : integer;var NumPoints : intege
r;
                        Xreal,Ximag : InVectorPtr ;var y0,yn : real);

{
                      PROCESSES THE SPECTRUM IN ORDER TO TRANSFORM IT
}

    const   averagenum        = 5;

    var     i,j,ist           :  integer;
            error             :  byte;
            Spec              :  spectrum;
            Ptr,Ptr1,Ptr2     :  Dpoint;
            Sum1,Sum2         :  real;

    begin

{  COMPUTE THE NUMBER OF POINTS OF THE FFT  }

      i := trunc(ln(iend-istart)/ln(2) +1);
      NumPoints := round(exp(i*ln(2)));

{         INITIALIZATION          }

      For i:= 1 to Numpoints do
        xreal^[i-1] := 0;
      ist := istart;
      Ptr := MeasPtr;
      while(ist > DataArraySize) do
      begin
        ist := ist - DataArraySize;
        Ptr := Ptr^.datalink;
      end;
      ist := ist-1;

{       CONVERSION OF THE LINKED LIST TO AN ARRAY    }

      for i := 0 to iend-istart-1 do
      begin
        ist := ist+1;
        xreal^[i] := ptr^.data[ist];
        if ist = DataArraySize then
        begin
          ist := 0;
          Ptr := Ptr^.datalink;
        end;
      end;

{     SHIFTING TO MAKE THE FUNCTION PERIODIC     }

      sum1 := 0;
      sum2 := 0;
      for i := 0 to averagenum-1 do
      begin
        sum1 := sum1 + xreal^[i];
        sum2 := sum2 + xreal [i+iend-istart-averagenum];
      end;
      y0 := sum1/averagenum;
      yn := sum2/averagenum;
      for i := 0 to iend-istart-1 do
      begin
        xreal [i] := xreal [i]-y0-(yn-y0)*i/(iend-istart-1);
      end;
      gotoxy(5,16);
      writeln('Number Of Points : ',NumPoints);
      gotoxy(5,17);
      writeln('y0 : ',y0,' yN : ',yn);

{         CALL OF THE FFT        }

      RealFFT(NumPoints,False,Xreal,Ximag,error);
    end;

{-----------------------------------------------------------------------}
```

```
{-----------------------------------------------------------------------------}

  procedure Define(var MainMenu : char );
{
           USED TO LOAD A SPECTRUM FROM DISK
}

   var  ch                        : char;
        spec                      : spectrum;
        quit,test,error,exit      : boolean;
        answer,Nname,namefile,defaultname: Name;
        count,u,v,x,i             : integer;
        y                         : real;
        infile                    :text;
        str                       : bigstring;


{-----------------------------------------------------------------------------}

  procedure Mkspec(FileName : Name;var Spec : Spectrum);

{

      THIS PROCEDURE READS A FILE AND MAKE A SPECTRUM
}

    var InputFile                                   : text;
        TextFileName                                : Name;
        Y,Z,x1,x2,x3,Imax,Increment,sum2,sum3       : real;
        I,I1,j,Ierr,Error,X,U,Nmeas                 : integer;
        Answer                                      : char;
        LPstore,Point1,Point2                       : Dpoint;

{-----------------------------------------------------------------------------}

        procedure textfilemake(FileName : Name);
{
          THIS PROCEDURE MAKES A FILE WHERE ALL THE EXPLANATIONS
          ABOUT THE DATA OF THE FILE "FILENAME.DAT" SHOULD BE PUT
}

  var txtfile : text;
      str     : bigstring;

    begin
      Assign(txtfile,FileName);
      Rewrite(txtfile);
      gotoxy(5,18);writeln('Write All The Information Available About ',FileNam
e);
      writeln('    ( Hit RETURN twice when you have finished ) ');
      repeat
        readln(str);
        write(txtfile,str);
      until(length(str)=0);
      close(txtfile);
    end;

{-----------------------------------------------------------------------------

    {  procedure MkSpec  }

    begin
          { CREATE A TEXT FILE IF IT DIDN'T EXIST }
       TextfileName := copy (FileName,1,length(FileName)-3) + 'txt';
       If not exist(TextFileName) then TextFileMake (FileName);
                { READ THE CHARACTERISTICS OF THE DATA }
       assign(InputFile,FileName);
       reset(InputFile);
       readln(InputFile,U,Y,X);
       readln(InputFile,Spec.Graph.title);
       readln(InputFile,Y,Z);
       readln(InputFile);
       gotoxy(5,25);write('
                ');
       gotoxy(5,25);write('                                   (Please Wait )');
       new(spec.measure1);
       new(spec.measure2);
       spec.measure1^.datalink := nil;
       spec.measure2^.datalink := nil;
       Spec.measure1^.datatype := fluorescence;
       Spec.measure2^.datatype := laserpower;
       if Default_Type then  Spec.measure1^.datatype:=normedflu;

       {  Definition of the record characteristics  }
```

```
{  Definition of the record characteristics  }

Spec.source := Filename;
Spec.MeasNumber := U ;
If U > MaxMeasNumber then
begin
  textcolor(Lightred);
  gotoxy(5,22);
  writeln('Unable To Process These Data ');
  gotoxy(5,23);
  writeln('Only',MaxMeasNumber,' Measurements considered');
  Spec.MeasNumber := MaxMeasNumber;
end;
increment := 1/x;
Spec.Increment := increment;
{   SPECTRUM DEFAULT CHARACTERISTICS      }
Spec.graph.xshift:=0;
Spec.graph.xscale:=1;
Spec.graph.yshift:=0;
Spec.graph.yscale:=1;
Spec.graph.OnOff := true;
Spec.Graph.Power := false;
Spec.Graph.PowScale := 1;
Spec.Graph.Xmin := Z;


{  Read the spectrum }

{$I-}
readln(inputfile,x1,x2,x3);
If Ioresult <> 0 then writeln('Error Code = ',IOresult);
{$I+}
Spec.Xmin := x1;
if Default_Type then x2 := x2/x3;
Spec.Measure1^.Data[1] := x2;
Spec.Measure2^.Data[1] := x3;
i := 1; j := 1;point1 := Spec.Measure1;point2 := Spec.Measure2;
if Default_Type then
begin
  while (i < Spec.MeasNumber) do
  begin
    i:= i+1;j:=j+1;
    {$I-}
    readln(InputFile,x1,x2,x3);
    If Ioresult <> 0 then writeln('Error Code = ',IOresult);
    {$I+}
```

```
            If ((x1 - Spec.Xmi          *increme                  en
            begin
              Point1 .Data[j] := x2/x3;
              Point2 .Data[j] := x3;
            end
            else  begin
              i1 := round(x1);
              textcolor(Lightred);
              writeln;
              writeln('Inconcistency In The Data At ',i1,' nm');
              ierr := 1;
            end;
{     CREATE NEW RECORDS IF NECESSARY             }
            if j = DataArraySize then
            begin
              new(point1 .Datalink);
              new(point2^.Datalink);
              point1 := point1^.datalink;
              point2 := point2^.datalink;
              point1^.datalink := nil;
              point2^.datalink := nil;
              Point1^.DataType := Continue;
              Point2 .DataType := Continue;
              j := 0;
            end;
          end;
        end
        else begin
          while (i   Spec.MeasNumber) do
          begin
            i:= i+1;j:=j+1;
            {$I-}
            readln(InputFile,x1,x2,x3);
            If Ioresult <> 0 then writeln('Error Code = ',IOresult);
            {$I+}
            If ((x1 - Spec.Xmin -(i-1)*increment) < 0.001) then
            begin
              Point1 .Data[j] := x2;
              Point2^.Data[j] := x3;
            end
            else  begin
              i1 := round(x1);
              textcolor(Lightred);
              writeln;
              writeln('Inconcistency In The Data At ',i1,' nm');
              ierr := 1;
            end;
{     CREATE NEW RECORDS IF NECESSARY             }
            if j = DataArraySize then
            begin
              new(point1^.Datalink);
              new(point2^.Datalink);
              point1 := point1^.datalink;
              point2 := point2^.datalink;
              Point1^.DataType := Continue;
              Point2 .DataType := Continue;
              j := 0;
            end;
          end;
        end;
        close(InputFile);
      end;
{-------------------------------------------------------------------------}
```

```
{                                                                    }

    { procedure define }

  begin
    background := blue;
    textcol    := white;
    anscol     := lightcyan;
    MainCh := Chr(120);
    quit := false;
    repeat
      WriteTitle('load a spectrum',27);
      gotoxy(5,8):write('Current Directory  : ');ans;write(Current_dir);
      gotoy(5,10);tex;write ('Default Type        :
  ');
      gotoxy(26,10);ans;
      if Default_Type then write('With Laser Power')
      else write ('Without Laser Power');Tex;
      test := false;count:=0;
      gotoxy(5,25);write('  Alt D> to change the directory,^Alt T  to change the
 type');
      repeat
          gotoxy(5,12);write('DOS-Filename        :
);
          Gread(26,12,error);tex;
          if Ganswer.typ.esc then
          begin
            if Ganswer.typ.chr then
            begin
              if Ganswer.chr = Chr(32) then Change_dir
              else begin
                if Ganswer.chr = chr(20) then
                begin
                  Default_Type := not Default_Type;Tex;
                  goto y(5,10);write ('Default Type        :
          ');
                  goto y(26,10);Ans;
                  if Default_Type then write('With Laser Power')
                  else write ('Without Laser Power');Tex;
                end
                else begin
                  quit := true;
                  MainCh := Ganswer.chr;
                end;
              end;
            end
            else begin
              MainCh := chr(27);
              Quit := true;
            end;
          end
          else begin
            answer := Ganswer.str;
            if (CountChr(answer,'\') = 0) and (length(current_dir) > 0)
                        then Answer := Current_dir + '\' + Answer;
            if CountChr(Answer,'.') = 0 then
            begin
              Nname := Answer + '.dat';
              If exist(Nname) then
              begin
                Namefile := Nname;
                Test := true;
              end;
            end
            else begin
              if exist(answer) Then
              begin
                NameFile := Answer;
                Test := true;
              end;
            end;
            count := count + 1;
            if Count > 5 then quit := true;
          end;
      until (test or quit);
```

```
      if test then
      begin
        assign(infile,NameFile);reset(infile);
        readln(Infile,u,y,s);readln(infile,str);
        close(infile);
        gotoxy(5,14);tex;write('Title                : ');ans;write(str);Tex;
        Defaultname := copy (NameFile,1,Length(Namefile)-4);
        While (countchr(Defaultname,'\') > 0) do
        Defaultname := copy (defaultname,2,length(defaultname)-1);
        repeat
          gotoxy(5,16);write('Local Name <',defaultname,'> : ');ans;read(answer)
;
          if length(answer)= 0 then
          begin
            answer := defaultname;
            write(answer);
          end;
        until (not valid(answer));
        Spec.Localname := answer;
        Mkspec(Namefile,spec);
        Speccre(spec);
        curveaffect(spec);
        repeat
          gotoxy(5,25);tex;write('Do You Want To Load An Other Spectrum (Y/N) ?
');
          read(Kbd,ch);
        until((ord(ch)=27)or(ord(Upcase(Ch))=78)or(ord(Upcase(Ch))=89));
        if ord(Upcase(ch)) = 78 then quit := true;
        if(ord(ch)=27)then
        begin
          quit := true;
          MainCh:= ch;
          if keypressed then  read(Kbd,MainCh);
        end;
      end;
    until(quit);
  end;

{-------------------------------------------------------------------------}
```

```
{-----------------------------------------------------------------------}

procedure Mult(spec: spectrum;var specout : spectrum; coef: real) ;
{
            MULTIPLIES ALL THE DATA OF A SPECTRUM BY COEF
            AND RETURNS THE RESULT IN SPECOUT
}

    var i,j : integer;
        point1,pointout1 : Dpoint;
        point2,pointout2 : Dpoint;

    begin
      specout := spec;
      new(specout.measure1);
      new(specout.measure2);
      specout.measure1^.datalink := nil;
      specout.measure2^.datalink := nil;
      i := 0;j:=0;point1 := spec.Measure1;pointout1 := specout.measure1;
      point2 := spec.Measure2;pointout2 := specout.measure2;
      repeat
        i := i+1;j:=j+1;
        pointout1^.data[j] := coef*point1^.data[j];
        pointout2^.data[j] := coef*point2^.data[j];
        if j = DataArraySize then
        begin
          j:= 0;
          new(pointout1^.datalink);
          pointout1 := pointout1^.datalink;
          point1:= point1^.datalink;
          pointout1^.datalink := nil;
          new(pointout2^.datalink);
          pointout2 := pointout2^.datalink;
          point2:= point2^.datalink;
          pointout2^.datalink := nil;
        end;
      until( i  = Spec.MeasNumber);
    end;

{-----------------------------------------------------------------------}

  procedure Add(spec1,spec2 : spectrum;var specout : spectrum);
{
            ADDS TWO SPECTRA AND RETURNS THE SUM IN SPECOUT
}

    var i,j : integer;
        point11,point21,pointout1 : Dpoint;
        point12,point22,pointout2 : Dpoint;

    begin
      convert(spec1,spec2);
      specout := spec1;
      specout.MeasNumber := min(spec1.MeasNumber,spec2.MeasNumber);
      if (spec1.increment <> spec2.increment) or (spec1.xmin <> spec2.xmin) the
n
                writeln('Error in the addition procedure');
      new(specout.measure1);
      new(specout.measure2);
      specout.measure1^.datalink := nil;
      specout.measure2^.datalink := nil;
      i := 0;j:=0;point11 := spec1.Measure1;point12 := spec1.Measure2;
      point21 := spec2.Measure1;point22 := spec2.Measure2;
      pointout1 := specout.measure1;pointout2 := specout.measure2;
      repeat
        i := i+1;j:=j+1;
        pointout1^.data[j] := point11^.data[j] + point21^.data[j];
        pointout2^.data[j] := point12^.data[j] + point22^.data[j];
        if j = DataArraySize then
        begin
          j:= 0;
          new(pointout1^.datalink);
          pointout1 := pointout1^.datalink;
          point11:= point11^.datalink;
          point12:= point12^.datalink;
          pointout1^.datalink := nil;
          new(pointout2^.datalink);
          pointout2 := pointout2^.datalink;
          point21:= point21^.datalink;
          point22:= point22^.datalink;
          pointout2^.datalink := nil;
        end;
      until( i  = Specout.MeasNumber);
      dispose(spec2.measure1);
      dispose(spec2.measure2);
    end;

{-----------------------------------------------------------------------}
```

```
{ ..... }

  procedure Lincomb(var MainCh : char);

{
          MAKES LINEAR COMBINATION OF SPECTRA
}
    var spec,spec2,specout : spectrum;
        point              : pointer;
        coef               : real;
        i,j                : integer;
        answer,defans,stg  : name;
        quit,exit,error    : boolean;
        ch                 : char;


   begin
   Background := blue;
   textcol    := lightred;
   anscol     := lightgreen;
   exit := false;
   repeat
     MainCh := chr(120);
     WriteTitle( 'linear combination',30);
     writemem;
     repeat
       gotoxy(5,7);write('Name Of The Spectrum 1 :                              ');
       Gread(30,7,error);
       if Ganswer.typ.esc then
       begin
         exit := true;
         if Ganswer.typ.chr then MainCh := Ganswer.chr
                            else MainCh := chr(27);
       end;
       answer := Ganswer.str;
     until((valid(answer)) or exit);
     if not exit then
     begin
       specindic(answer,point);spec := point^;
       gotoxy(40,7);write('Coefficient : ');
       Gread(55,7,error);
       coef := Ganswer.re;
       mult(spec,specout,coef);spec := specout;
       i := 1;
       repeat
         i:= i+1;
         quit := false;
         repeat
           writemem;
           gotoxy(5,5+2*i);write('
');
           gotoxy(5,5+2*i);
           write('Name Of The Spectrum ',i,' : ');Gread(30,5+2*i,error);
           if Ganswer.typ.esc then
           begin
             exit := true;
             quit := true;
             if Ganswer.typ.chr then MainCh := Ganswer.chr
                                else MainCh := chr(27);
           end;
           answer := Ganswer.str;
           if length(answer) = 0 then
           begin
             quit := true;
             gotoxy(5,5+2*i);write('                                              ');
           end;
         until(valid(answer) or quit );
```

```
        if not quit then
        begin
          gotoxy(40,5+2*i);
          write('Coefficient : ');
          Gread(55,5+2*i,error);coef := Ganswer.re;
          specindic(answer.point);
          mult(point^,spec2, coef);
          add(spec,spec2,specout) ;
          datadel(spec.measure1);
          datadel(spec.measure2);
          datadel(spec2.measure1);
          datadel(spec2.measure2);
          spec := specout;
        end;
      until (quit);
      if not exit then
      begin
        repeat
          j := 0;
          repeat
            j:=j+1;
            defans := 'comb'+ chr( j + 48) ;
          until (not valid(defans));
          gotoxy(5,7+2*i);
          write('Name Of The Linear Combination ',defans,'  :
');
          gread(46,7+2*i,error);
          if length(Ganswer.str) = 0 then
          begin
            answer := defans;
            write(answer);
          end
          else answer := Ganswer.str;
        until ( not valid(answer));
        spec.LocalName := answer;
        spec.source :='';
        spec.graph.title :='LINEAR COMBINATION';
        speccre(spec);
        curveaffect(spec);
        gotoxy(5,25);write('Do You Want To Make Another Linear Combination (Y/N
) ? ');
        read(Kbd,ch);
        If ord(Upcase(ch))=78 then exit := true;
        if ord(ch) = 27 then
        begin
          exit := true;
          mainCh:= Chr(27);
          if keypressed then
          begin
            read(Kbd,ch);
            MainCh:= ch;
          end;
        end;
      end;
    end;
  until(exit);
  end;

{-------------------------------------------------------------------------
```

```
  procedure smoothing(var MainCh :char);

          ALLOWS THE USER TO SMOOTH SPECTRA
}

 const  plotcolor = lightgreen;

 var     spec            : spectrum;
         quit,error      : boolean;
         num,yo,yn       : real;
         fftaxis         : axdef;
         istart,iend     : integer;
         Npoints,icut    : integer;
         MeasPtr         : Dpoint;
         answer          : Name;
         ch              : char;
         Xreal,Ximag,Yreal,Yimag : TnVectorPtr;
         SpecPtr,SpecoutPtr,SmSpecPtr : pointer;
         byteer          : byte;
{-------------------------------------------------------------------------}

  procedure fftsave(xreal,ximag : TnVectorPtr; stype : integer);

{         SAVES A FOURIER TRANSFORM INTO A SPECTRUM       }

    var   spec1 : spectrum;
          i,j   : integer;
          Ptr1,Ptr2 : Dpoint;
          answer    : name;
     begin
        Spec1.source := Spec.source;
        Spec1.localname := 'sm'+spec.localname;
        Spec1.MeasNumber := Npoints ;
        Spec1.xmin := fxstart;
        Spec1.Increment := spec.increment;
        {   SPECTRUM DEFAULT CHARACTERISTICS      }
        Spec1.graph.xshift:=0;
        Spec1.graph.xscale:=1;
        Spec1.graph.yshift:=0;
        Spec1.graph.yscale:=1;
        Spec1.graph.OnOff := true;
        Spec1.Graph.Power := false;
        Spec1.Graph.PowScale := 1;
        Spec1.Graph.Xmin := fxstart;
        Spec1.Graph.title := 'Smoothed Spectrum '+ spec.localname;
        if stype = FourierTr then
        begin
          Spec1.localname := 'fft' + spec.localname;
          Spec1.graph.title := 'Fourier Transform ' + spec.localname;
        end;
        repeat
          gotoxy(1,25);write('
  ');
          gotoxy(5,25);write('Local Name <',spec1.localname,'> :
  ');
          gotoxy(1,25);ans;read(answer);
          if length(answer)= 0 then
          begin
            answer := spec1.localname;
            write(answer);
          end;
        until (not valid(answer));
        spec1.localname := answer;

{         INITIALIZATION  }

       new(spec1.measure1);
       new(spec1.measure2);
       ptr1 := spec1.measure1;
       ptr2 := spec1.measure2;
       j:=0;

{         CONVERSION  ARRAY TO LINKED LIST    }

       for i:= 0 to Npoints-1 do
       begin
         j := j+1;
         Ptr1^.data[j] := xreal [i];
         Ptr2^.data[j] := ximag [i];
         if j = DataarrayySize then
         begin
           j := 0;
           new(Ptr1^.datalink);
           new(Ptr2^.datalink);
           ptr1 := ptr1^.datalink;
           ptr2 := ptr2^.datalink;
         end;
       end;
       Spectra(spec1);
       Curveaffa(spec1);
     end;
```

```
{-------------------------------------------------------------------------------}

    procedure plotfft (ArrayPtr : InVectorPtr;Increment,Xshift,Yshift : real);

{     THIS PROCEDURE PLOTS THE CURVES DEFINED IN SPECTRUM     }

var   i,x1,x2,y1,y2   :  integer;

   begin
     x1 := trunc(Xshift - XOrig*XScreenScale);
     y1 := trunc(((ArrayPtr^[0]+Yshift)-YOrig)*YScreenScale);
     for i := 1 to Npoints-1  do
     begin
       x2 := trunc((i*increment+xshift-XOrig)*XScreenScale);
       y2 := trunc(((ArrayPtr^[i]+YShift)-YOrig)*YScreenScale);
       if(x1 > 40) then  draw(x1,y1,x2,y2,1);
       x1 := x2;y1 := y2;
     end;
   end;

{-------------------------------------------------------------------------------}
{    SMOOTHING : BEGINNING OF THE CODE }

    begin
      Background := blue;
      textcol    := lightred;
      anscol     := lightgreen;
      exit := false;
      repeat                              { LOOP UNTIL QUIT THE SMOOTHING }
        repeat                            { LOOP ON THE INPUT }
          MainCh := chr(120);

{               USER'S INPUT          }

          WriteTitle('smoothing',30);
          WriteMem;
           [illegible]
          Gread [illegible]);
          if Ganswer.typ esc then
          begin
            exit := true;
            if Ganswer.typ.chr then Mainch := Ganswer chr
                            else Mainch := chr(27);
          end;
          answer := Ganswer.str;
        until((valid(answer)) or exit);
        if not exit then
        begin
          specindic(answer,SpecPtr);
          spec := SpecPtr ;
          gotoxy(5,10);write('Starting Point  ',fxstart:3:1,'  : ');gread(30,10,
error);
          if Ganswer.typ.re then fxstart := ganswer.re
          else write(fxstart:3:1);
          gotoxy(5,12);write('Ending Point    ',fxend:3:1,'  :  ');gread(30,12,er
ror);
          if Ganswer.typ.re then fxend := ganswer.re
          else write(fxend:3:1);
          gotoxy(5,14);write('Do You Want The Transform Of The Laser Power (Y/N)
 ?');
          read(kbd,ch);
          MeasPtr := spec.measure1;
          if upcase(ch) ='Y' then MeasPtr := spec.measure2;
          gotoxy(30,25);write( Please Wait  );

          istart := trunc((fxstart-spec.xmin)/spec.increment);
          iend   := trunc((fxend-spec.xmin)/spec.increment);

 {      COMPUTE THE TRANSFORM   }

          new(Xreal);
          new(Ximag);
          smfft(MeasPtr,istart,iend,Npoints,Xreal,Ximag,yo,yn);

 {  DEFINE THE AXIS FOR PLOTTING }
```

```
{  DEFINE THE AXIS FOR PLOTTING }

        fftaxis.title := 'FOURIER TRANSFORM';
        fftaxis.Labelx := '';
        fftaxis.labely :='';
        fftaxis.xmin := 0;
        fftaxis.xmax := npoints;
        fftaxis.deltax := npoints/8;
        fftaxis.ymin :=-1;
        fftaxis.ymax := 1;
        fftaxis.deltay := 0.2;
        repeat
          repeat
            hires;
            hirescolor(plotcolor);
            plotaxis(fftaxis);
            plotfft(Xreal,1,0,0);
            plotfft(Ximag,1,0,1);
            gotoxy(5,25);write('Frequency ? ');Gread(20,25,error);
            if ganswer.typ.esc then
            begin
              exit := true;
              MainCh := Ganswer.chr;
              if Ord(mainch) = 58 then
              begin
                a:=def(Mainch,fftaxis);
                mainch := chr(120);
                exit := false;
              end;
            end;
          until ( exit or Ganswer.typ.int );
          if Ganswer.typ.int then icut := Ganswer.int;
          if not exit then                    { INVERSE TRANSFORM
          begin
            new(yreal);new(yimag);
            for i := 0 to Npoints-1 do
            begin
              yreal [i] := xreal^[i];
              yimag^[i] := ximag^[i];
            end;
            if (icut   0) and (icut  Npoints/2) then
            begin
              for i := icut to Npoints-icut-1 do
              begin
                yreal [i] := 0;
                yimag [i] := 0;
              end;
            end;
            gotoxy(30,25);write('Please Wait');
            ComplexFFT(Npoints,true,yreal,yimag,byteer);
            for i := 0 to iend-istart-1 do
            begin
              yreal [i] := yreal [i] + yo + ( yn - yo )*i/(iend-istart-1);
            end;
            repeat
              hires;
              hirescolor(plotcolor);
              plotaxis(axis);
              plotfft(yreal,spec.increment,fxstart,0);
              plotfft(yimag,spec.increment,fxstart,1);
              repeat
                gotoxy(1,25);write('
');
                gotoxy(5,25);write( Are You Satisfy (Y/N) ?
```

```
                gotoxy(5,25);write('Are You Satisfy (Y/N) ');
                read(Kbd,ch);
              until((ord(ch)=27)or(ord(Upcase(Ch))=78)or(ord(Upcase(Ch))=89));
              if(ord(ch)=27)then
              begin
                exit := true;
                MainCh:= ch;
                if keypressed then  read(Kbd,MainCh);
                if Ord(mainch) = 68 then
                begin
                  axisdef(Mainch,axis);
                  mainch := chr(120);
                  exit := false;
                end;
              end;
          {     until(exit or(Upcase(Ch)='N')or(Upcase(Ch)='Y'));
            end;
          until ( exit or (upcase(Ch) = 'Y') );
          if not exit then
          begin
            clrscr;
            gotoxy(5,12);write('Do You Want To Save The Fourier Transform ? (Y/N
)');
            read(Kbd,Ch);
            if upcase(Ch)='Y' then fftsave(xreal,ximag,FourierTr);
            dispose(xreal);dispose(ximag);
            gotoxy(5,14);write('Do You Want To Save The Smoothed Spectrum ? (Y/N
)');
            read(Kbd,Ch);
            if upcase(Ch)='Y' then fftsave(yreal,yimag,NormedFluo);
            dispose(yreal);dispose(yimag);
            repeat
              gotoxy(5,25);write( Do You Want To Make Another Transform (Y/N)
 );
              read(Kbd,ch);
            until((ord(ch)=27)or(ord(Upcase(Ch))=78)or(ord(Upcase(Ch))=89));
            if ord(Upcase(ch)) = 78 then exit := true;
            if(ord(ch)=27)then
            begin
              exit := true;
              MainCh:= ch;
              if keypressed then  read(Kbd,MainCh);
            end;
          end;
        end;
      until(exit);
    end;

{-------------------------------------------------------------------------}
```

```
{------------------------------------------------------------------------------}
procedure stat(var MainCh : char);
{
                 PROCEDURE OF STATISTICAL ANALY
}
   var    answer                 : Name;
          spec1,spec2            : Spectrum;
          meany,variancey        : real;
          meanz,variancez        : real;
          coeff1,coeff2,cprop    : real;
          scprod                 : real;
          i                      : integer;
          specname               : name;
          quit,exit,error        : boolean;
          ch                     : char;
          Spoint                 : pointer;

{------------------------------------------------------------------------------}
            procedure mv(spec : spectrum;var my,vy,mz,vz,xs,xe : real);

{           THIS PROCEDURE CALCULATES THE MEAN VALUE AND VARIANCE
                           OF THE TWO SIGNALS

}
    var i,n1,j          : integer;
        x1,sum1,sum2,var1,var2 : real;
        point1,point2 : Dpoint;

   begin
     x1 := spec.xmin ;n1 := 0;sum1:= 0;sum2 := 0;var1 := 0;var2 := 0;j :=0;
     point1 := spec.Measure1;
     point2 := spec.Measure2;
     for i := 1 to spec.MeasNumber do
     begin
       j := j+1;
       if(x1 >= xs)and(x1 <= xe)
       then begin
         n1    := n1 + 1;
         sum1 := sum1 + point1^.data[j];
         var1 := var1 + Sqr(point1^.data[j]);
         sum2 := sum2 + point2^.data[j];
         var2 := var2 + Sqr(point2^.data[j]);
       end;
       if j = DataArraySize then
       begin
         j:=0;
         point1 := point1^.datalink;
         point2 := point2^.datalink;
       end;
       x1 := x1 + spec.increment;
     end;
     my := sum1/n1;
     vy :=  var1 /(n1-1) - sqr(my);
     mz := sum2/n1;
     vz :=  var2 /(n1-1)- sqr(mz);
   end;
{------------------------------------------------------------------------------}
  procedure correl( spec1,spec2 : spectrum;xstart,xend,deltaw:real;var coeff1, c
oeff2,cprop,scalprod : real);

{          THIS PROCEDURE COMPUTES THE CORRELATION COEFFICIENTS BETWEEN
                  TWO SPECTRA
}
  var i,n1,j : integer;
      var11,var12,var21,var22 : real;
      x1,cor1,cor2 : real;
      point11,point12,point21,point22 : Dpoint;

 begin
    spec2.xmin :=  spec1.xmin + deltaw;
    convert(spec1,spec2);
    n1 := 0;var11:= 0;var12 := 0;var21 := 0;var22 := 0;
    cor1 := 0;cor2 := 0;j:=0;x1 := spec1.xmin;
    point11 := spec1.Measure1;
    point12 := spec1.Measure2;
    point21 := spec2.Measure1;
    point22 := spec2.Measure2;
    for i := 1 to spec1.MeasNumber do
    begin
      j := j+1;
      x1 := x1 + spec1.increment;
      if( x1 >= xstart)and( x1 <= xend)
      then begin
        n1    := n1 + 1;
        var11 := var11 + Sqr(point11^.data[j]);
        var21 := var21 + Sqr(point21^.data[j]);
        var12 := var12 + Sqr(point12^.data[j]);
        var22 := var22 + Sqr(point22^.data[j]);
        cor1  := cor1 + point11^.data[j]*point21^.data[j];
        cor2  := cor2 + point12^.data[j]*point22^.data[j];
      end;
```

```
      if i = DataArraySize then
      begin
        i := 0;
        point11 := point11^.datalink;
        point21 := point21^.datalink;
        point12 := point12^.datalink;
        point22 := point22^.datalink;
      end;
    end;
    coeff1 := cor1/sqrt(var11*var21);
    coeff2 := cor2/sqrt(var12*var22);
    prop  := cor1/var21;
    scalprod := cor1;
    datadel(spec2.measure1);
    datadel(spec2.measure2);
  end;
{-------------------------------------------------------------------------}
{STAT : BEGINNING OF THE CODE}

begin
quit := false;
repeat
  MainCh := chr(120);
  writeTitle('statistical analysis',30);
  repeat
    writemem;
    gotoxy(5,8);write('Name Of The Spectrum   :                              ');
    gread(30,8,error);
    if ganswer.typ.esc then
    begin
      quit := true;
      MainCh := ganswer.chr;
    end;
  until( valid(ganswer.str) or quit );
  if valid (ganswer.str) then
  begin
    specindic(ganswer.str,Spoint^);
    spec1 := Spoint ;
  end;
  if not quit then
  begin
    gotoxy(5,10);write('Starting Point   ',sxstart:3:1,'   : ');gread(30,10,error
);
    if Ganswer.typ.re then sxstart := ganswer.re
    else write(sxstart:3:1);
    gotoxy(5,12);write('Ending Point   ',sxend:3:1,'  : ');gread(30,12,error)
;
    if Ganswer.typ.re then sxend := ganswer.re
    else write(sxend:3:1);
    gotoxy(5,14);
    mv(spec1.meany,variancey,meanz,variancez,sxstart,sxend);
    writeln ('Mean Value of Y :  ',Meany:6:3);
    gotoxy(5,15);
    writeln ('Variance of Y   :  ',variancey:6:3);
    gotoxy(5,16);
    writeln ('Mean Value of Z :  ',Meanz:6:3);
    gotoxy(5,17);
    writeln ('Variance of Z   :  ',Variancez:6:3);
    exit := false;
    repeat
      repeat
        gotoxy(5,20);write('Correlation With :                              ');
        gread(25,20,error);
        if ganswer.typ.esc then
        begin
          quit := true;
          MainCh := Ganswer.chr;
        end;
        if length(ganswer.str)=0 then exit := true;
      until( valid(ganswer.str) or quit or exit);
      if valid(ganswer.str) then
      begin
        specindic(ganswer.str,Spoint);
        spec2 := Spoint^;
        gotoxy(5,21);
        write('Wavelength Shift  ',WaveLengthShift:3:1,' nm    :
 ');gread(37,21,error);
        if Ganswer.typ.re then WaveLengthShift := Ganswer.re
        else  write(WaveLengthShift:3:1);
        correl(spec1,spec2,sxstart,sxend,WaveLengthShift, coeff1,coeff2,prop
,scalprod);
        gotoxy(5,23);
        write('Correlation Coefficients    :  ',coeff1:6:3,'     ',coeff2:6:3,'  :
');
        gotoxy(5,22);
        write('Proportionality Coefficient :  ',prop:6:3,'     Scalar Product
 :',scalprod:6:3);
      end;
    until( quit or exit);
  end;
```

```
  if not quit then
  begin
    repeat
      gotoxy(5,25);write('Do You Want To Load Another Spectrum (Y/N) ');
      read(kbd,ch);
    until((ord(ch)=27)or(ord(Upcase(Ch))='N')or(ord(Upcase(Ch))='Y'));
    if ord(Upcase(ch)) = 78 then quit := true;
    if(ord(ch)=27)then
    begin
      quit := true;
      MainCh:= ch;
      if keypressed then  read(kbd,MainCh);
    end;
  end;
until(quit);
end;
```

```
{
        MAIN MENU FOR SPECTRUM PROCESSING

}
  procedure MainMenu(var MainCh : char);

    var   Exit            : boolean;
          Marray          : Menutab;
          answer,i1,i2    : integer;
          Error           : boolean;

    begin
      MainCh:=chr(120);
      marray[1].title := 'load a spectrum from a file';
      marray[1].typeL := 0;
      marray[2].title := 'Linear combination';
      marray[2].typeL := 0;
      marray[3].title := 'smoothing';
      marray[3].typeL := 0;
      marray[4].title := 'statistical analysis';
      marray[4].typeL := 0;
      marray[5].title := 'list,save or delete';
      marray[5].typeL := 0;
      marray[6].title := 'Quit';
      marray[6].typeL := 0;
      Menu1(6,'MAIN MENU',false,marray,answer,i1,i2,error);
      case answer of
            1       : MainCh :=chr(121);
            2       : MainCh :=chr(122);
            3       : MainCh :=chr(123);
            4       : MainCh :=chr(124);
            5       : MainCh :=chr(31);
            6       : MainCh :=chr(16);
      end;
      if i1 = 27 then
      begin
        MainCh:=chr(i1);
        if i2 > 0 then  MainCh := chr(i2);
      end;
    end;
```

Appendix E

Listing of the PLS regression program

The partial least squares regression program used in this study has been written from the algorithm given in Geladi and Kowalski[86].

```
program  pls;
{$N+}

{    IS PROGRAM WRITTEN FROM THE ALGORITHM GIVEN
     BY GELADI AND KOWALSKI IN ANALYTICA CHIMICA
     ACTA, 185(1986)1-17                          }


const  ns = 10;              {number of spectra}
       nx = 30;              {number of data per spectrum}
       ny = 2;               {number of y-data}

type   vectorny = array[1..ny] of real;
       vectorns = array[1..ns] of real;
       vectornx = array[1..nx] of real;

 var   ws,ps : array[1..ns] of vectornx;
       us,ts : array[1..ns] of vectorns;
       w,p,xp,meanx,sdx : vectornx;
       u,t,told : vectorns;
       q,yp,yest,meany,sdy : vectorny;
       qs    : array[1..ns] of vectorny;
       b     : array[1..ns] of real;
       x,e   : array[1..ns,1..nx] of real;
       y,f   : array[1..ns,1..ny] of real;
       a,i,ir,ia,ip,j,ja,h : integer;
       infile,infile2 : text;
       filename,filepls:string[15];
       quit : boolean;
       norm3,norm2,norm,r,tp:real;
       ch : char;
       count : integer;
       amodel:integer;
       epsilon:real;


 begin

          {DATA INPUT}

   quit := false;
   h := 1;
   repeat
     clrscr;
     gotoxy(5,10):write('file name :');
     read(filepls);
     if filepls='' then quit := true
     else begin
        assign(infile2,filepls);
        reset(infile2);
        readln(infile2,filename);
        writeln(filename);
        while(filename     ) do
        begin
          readln(infile2,x[h,1],y[h,2]);
          assign(infile,filename);
          reset(infile);
          for i := 1 to 3 do
          begin
            readln(infile,r,tp);
          end;
          for i := 1 to nx do
          begin
            readln(infile,r,x[h,i]);
          end;
          close(infile);
          writeln(y[h,1],y[h,2]);
          h := h+1;
          readln(infile2,filename);
        end;
        quit:=true;
        close(infile2);
     end;
   until(quit=true);
   h:=h-1;
   writeln;
   writeln('h=',h);
```

```
    {mean centering and variance scaling}

  for ia := 1 to nx do
  begin
    meanx[ia]:=0;
    for ja := 1 to h do meanx[ia]:=meanx[ia]+x[ja,ia];
    meanx[ia]:=meanx[ia]/h;
    for ja := 1 to h do x[ja,ia]:=x[ja,ia]-meanx[ia];
  end;

  for ia := 1 to ny do
  begin
    meany[ia]:=0;
    for ja := 1 to h do meany[ia]:=meany[ia]+y[ja,ia];
    meany[ia]:=meany[ia]/h;
    for ja := 1 to h do y[ja,ia]:=y[ja,ia]-meany[ia];
  end;

  for ia := 1 to nx do
  begin
    sdx[ia]:=0;
    for ja := 1 to h do sdx[ia]:=sdx[ia]+x[ja,ia]*x[ja,ia];
    sdx[ia]:=sqrt(sdx[ia]/(h-1));
    for ja := 1 to h do x[ja,ia]:=x[ja,ia]/sdx[ia];
  end;

  for ia := 1 to ny do
  begin
    sdy[ia]:=0;
    for ja := 1 to h do sdy[ia]:=sdy[ia]+y[ja,ia]*y[ja,ia];
    sdy[ia]:=sqrt(sdy[ia]/(h-1));
    for ja := 1 to h do y[ja,ia]:=y[ja,ia]/sdy[ia];
  end;

    {PLS algorithm}

  writeln;
  write('order of the model :');
  readln(amodel);
  write('Epsilon = ');readln(epsilon);

  while(amodel > 0) do
  begin
  for i:= 1 to h do told[i]:=0;

e:=x;
f:=y;
writeln(' MATRIX X ');
for ia :=1 to nx do
writeln(e[1,ia]:6:3,e[2,ia]:6:3,e[3,ia]:6:3,e[4,ia]:6:3,e[5,ia]:6:3);
writeln(' MATRIX Y ');
for ia := 1 to ny do
writeln(f[1,ia]:6:3,f[2,ia]:6:3,f[3,ia]:6:3,f[4,ia]:6:3,f[5,ia]:6:3);
for a := 1 to amodel do
begin
count := 0;
{step1}
for ia:=1 to h do u[ia] := f[ia,1];
repeat
  count := count +1; writeln('count=',count);
  {step2}
  for ia := 1 to nx do
  begin
    w[ia]:=0;
    for ja := 1 to h do w[ia]:=w[ia]+u[ja]*e[ja,ia];
  end;
  {step3}
  norm:=0;
  for ia := 1 to nx do norm := norm + w[ia]*w[ia];
  norm := sqrt(norm);
  for ia := 1 to nx do w[ia]:=w[ia]/norm;
  {step4}
  for ia := 1 to h do
  begin
    t[ia]:=0;
    for ja := 1 to nx do t[ia]:=t[ia]+w[ja]*e[ja,ia];
  end;
```

```
  {step5}
  for ia := 1 to ny do
  begin
    q[ia]:=0;
    for ja := 1 to h do q[ia]:=q[ia]+t[ja]*f[ja,ia];
  end;
  {step6}
  norm:=0;
  for ia := 1 to ny do norm := norm + q[ia]*q[ia];
  norm := sqrt(norm);
  for ia := 1 to ny do q[ia]:=q[ia]/norm;
  {step7}
  for ia := 1 to h do
  begin
    u[ia]:=0;
    for ja := 1 to ny do u[ia]:=u[ia]+q[ja]*f[ja,ia];
  end;
  norm:=0;
  for ia := 1 to ny do norm := norm + q[ia]*q[ia];
  norm := sqrt(norm);
  for ia := 1 to ny do u[ia]:=u[ia]/norm;
  {step8}
  norm:=0;
  for ia := 1 to h do norm := norm + (t[ia]-told[ia])*(t[ia]-told[ia]);
  norm2:=0;
  for ia := 1 to h do norm2 := norm2 + t[ia]*t[ia];
  r:=sqrt(norm/norm2);
  for ia := 1 to h do
  begin
    told[ia]:=t[ia];
  end;
  writeln('r=',r);
until((r<epsilon)or(count>100));
writeln(a,'vector t found in ',count,' iteration');
for i:= 1 to h do writeln(t[i]);
read(kbd,ch);

  {step9}
  for ia := 1 to nx do
  begin
    p[ia]:=0;
    for ja := 1 to h do p[ia]:=p[ia]+t[ja]*e[ja,ia];
  end;
  norm:=0;
  for ia := 1 to h do norm := norm + t[ia]*t[ia];
  for ia := 1 to nx do p[ia] := p[ia]/norm;
  {step10,11,12}
  norm:=0;
  for ia := 1 to nx do norm := norm + p[ia]*p[ia];
  norm := sqrt(norm);
  for ia := 1 to nx do p[ia]:=p[ia]/norm;
  for ia := 1 to nx do w[ia]:=w[ia]*norm;
  for ia := 1 to h do t[ia]:=t[ia]*norm;
  {step13}
  norm:=0;
  for ia := 1 to h do norm := norm + t[ia]*t[ia];
  norm2:=0;
  for ia := 1 to h do norm2 := norm2 + u[ia]*t[ia];
  b[a]:=norm2/norm;
  {save x,q,w,t,u}
  xs[a]:=x;
  qs[a]:=q;
  ws[a]:=w;
  ts[a]:=t;
  us[a]:=u;
  {calculation of the residuals}
  for ia:=1 to h do
  begin
    for ja := 1 to nx do e[ia,ja]:=e[ia,ja]-t[ia]*p[ia];
  end;
  for ia := 1 to h do
  begin
    for ja:= 1 to ny do f[ia,ja]:=f[ia,ja]-b[a]*t[ia]*q[ja];
  end;
  writeln(' MATRIX X  );
  for ia :=1 to nx do
  writeln(e[1,ia]:10:3,e[2,ia]:10:3,e[3,ia]:10:3,e[4,ia]:10:3,e[5,ia]:10:3 );
  writeln(' MATRIX Y  );
  for ia := 1 to ny do
  writeln(f[1,ia]:10:3,f[2,ia]:10:3,f[3,ia]:10:3,f[4,ia]:10:3,f[5,ia]:10:3
end;
```

```
    {PROJECT1(W)
  quit := false;
  repeat
    clrscr;
    gotoxy(5,10);write('complete dos file name :');
    read(filename);
    if filename='' then quit := true
    else begin
       assign(infile,filename);
       reset(infile);
       for i := 1 to 3 do
       begin
         readln(infile,r,tp);
       end;
       for i := 1 to nx do
       begin
         readln(infile,r,xp[i]);
       end;
       close(infile);
       for i:=1 to nx do xp[i]:=(xp[i]-meanx[i])/sdx[i];
       yp[1]:=0;
       yp[2]:=0;
       writeln;
       for ip := 1 to amodel do
       begin
         tp:=0;
         for i:=1 to nx do tp := tp+xp[i]*ws[ip][i];
         for i:=1 to nx do xp[i] := xp[i] -tp*ps[ip][i];
         for i:= 1 to ny do yp[i]:= yp[i]+b[ip]*tp*qs[ip][i];
         yest[1]:=yp[1]*sdy[1] +meany[1];
         yest[2]:=yp[2]*sdy[2] +meany[2];
         writeln(ip,yest[1],yest[2]);
       end;
       read(kbd,ch);
    end;
  until(quit=true);
     writeln;
    write('order of the model :');
    readln(amodel);
    write('Epsilon =  ');readln(epsilon);
    end;

end.
```

BIBLIOGRAPHY

Armiger and Humphrey [1979] in Peppler Microbial Technology,chapt. 15 vol. II, Academic Press,1979

Becker R. S.[1969], Theory and Interpretation of fluorescence and phosphorescence, Wiley, New York, 1969.

Beyeler[1981], On-line measurement of culture fluorescence, Method and application, European Journal of Applied Microbiology and Biotechnology 13:10-14.

Cline Love and Shover[1980], Critical evaluation of Lifetime Measurements via reiterative Convolution Using Simulated and Real Multiexponential Fluorescence Decay Curves.

Cowgill[1963], Biochim. Biophys. Acta, 75:272, 1963.

Eastment and Krzanowski[1982], Cross-validatory choice of the number of components from a principal components analysis. Technometrics 24,73-77.

Franck and Kowalski[1984], Prediction of Wine quality and geographic origin from chemical measurements by partial least squares modeling. Analytica Chimica Acta, 162(1984) 241-251.

Froelich P.[1985], Fluorescence of organic compounds, Instrumentation-Research,

March 1985, p.98-103.

Geladi and Kowalski[1986], Partial Least-Squares Regression : A Tutorial. Analytica Chimica Acta, 185(1986)1-17.

Guilbault[1973], Practical Fluorescence,

Jolliffe[1982], A note on the use of principal components in regression. Applied Statistics, 31, 300-303.

Konev[1967], Fluorescence and Phosphorescence of Proteins and Nucleic Acids. Plenum Press. New York 1967.

Lindberg[1983], Partial Least Squares method for spectrofluorimetric analysis of mixtures of Humic acid and Ligninsulfonate. Analytical Chemistry, 55(1983)643.

Lorber et alt.[1987], A theoritical foundation for the partial least squares algorithm. Journal of Chemometrics, vol. 1, 19-31(1987).

NATO proceedings[1983] Advanced Study Inst. on Chemometrics, Cozenza, Italy, september 1983. Reidel Publish Co, Dardrecht, Holland, 1984, pp.17-95.

Scheper et alt.[1986], Measurement of culture fluorescence during the cultivation of Penicillium Chrysogenum and Zymomonas mobilis. Journal of biotechnology, 3(1986)231-238.

Scheper and Schügerl[1986], Characterization of bioreactors by in-situ fluorometry. Journal of Biotechnology, 3(1986), 221-229.

Simmons and Wang[1987], Modeling of a fluorescence probe, to be submitted to bioengineering.

Teale and Weber[1957], Ultraviolet Fluorescence of the aromatic amino acids, Biochemistry Journal, 65:476-482.

Vladimirov[1959] Izv. Nauk SSSR, ser. fiz, 23:86,1959.

Vladimirov and Burshtein[1959] Biofizica, 5:385,160.

Warner I.M.[1975] I.M. Warner,J.B. Callis, E.R. Davidson, M.P. Gouterman, and G.D. Christian, "Fluorescence Analysis: A New Approach", Analytical Letters, 8,665-681(1975).

Warner I.M.[1977], I.M. Warner, E.R. Davidson and G.D. Christian, "Quanttitative Analysis of multicomponent fluorescence data by the methods of Least Squares and Non-negtive Least sum of Errors. Analytical Chemistry, 49,2155(1977).

Wetlaufer D.B., 1962 "Ultraviolet Absorption of Proteins and Amoni Acids" Advanced Protein Chemistry 7:33-39.

White A.[1959], Biochemistry Journal, 71:217(1959).

Wold H.[1982], Soft modeling. The basic design and some extensions In K.G.

Jöreskog and H.A. Wold, Systems Under Indirect observation, Part I and II, North Holland, Amsterdam,1982

Wold S.[1978] Cross validatory estimation of the number of components in factor and principal components models. Technometrics 20,397-406.