Analysis of Laser Induced Fluorescence Spectra

By D.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

I-Introduction ................................................................. 1
  1.1 Presentation ......................................................... 1
  1.2 The need for an on-line sensor ................................. 1
  1.3 Optical techniques ................................................ 2
  1.4 Laser induced fluorescence ........................................ 3
  1.5 The need for data processing ..................................... 4
  1.6 Purpose of the study ............................................... 5

II-Fluorescence ............................................................... 7
  2.1 Theory ................................................................. 7
    a) Absorption ....................................................... 7
    b) Scattering ........................................................ 9
    c) Vibrational Relaxation ......................................... 10
    d) Fluorescence .................................................... 10
    e) Phosphorescence ............................................... 11
    f) Internal Conversion ............................................ 11
  2.2 Relation with the molecular structure ......................... 12
  2.3 Environmental influences ....................................... 13
    2.3.1 Solvent Effects ............................................. 13
    2.3.2 Effects of pH ............................................... 15
2.3.3 Fluorescence Quenching ........................................... 16

2.3.4 Effects of Temperature ............................................. 17

2.4 Use in biotechnology ................................................. 18

2.4.1 Presentation ......................................................... 18

2.4.2 NADH probe ......................................................... 20

III-Measurements ........................................................... 24

3.1 The experimental setup ............................................... 24

3.1.1 Description ........................................................... 24

3.1.1.1 The light source .................................................. 26

   a) The Nd : YAG laser .............................................. 26

   b) The dye laser ..................................................... 28

   c) Frequency doubling .............................................. 29

3.1.1.2 The sample illumination ....................................... 30

3.1.1.3 The measurement devices ..................................... 31

   a) The monochromator .............................................. 31

   b) The photomultiplier tube ..................................... 33

   c) The photodiode .................................................. 34

3.1.1.4 The electronic processing ..................................... 34

3.1.2 The measurements presently done ................................ 35

   a) The amino acid measurements .................................. 36

   b) The fermentation measurements ............................... 36

3.1.3 The other measurements available ............................. 37
a) Time-resolved fluorescence ...................... 37
b) Excitation-emission matrix .......................... 37
c) Two-photon fluorescence ............................. 40

3.2 The pure amino acids measurements ............... 42

3.2.1 Fluorescence of amino acids ..................... 42

3.2.1.1 Absorption .................................. 43
3.2.1.2 Fluorescence .................................. 43

a) Tryptophan ........................................... 43
b) Tyrosine ............................................... 45
c) Phenylalanine ....................................... 45

3.2.2 Amino acids measurements ........................ 46

3.3 Noise and Reproducibility ......................... 47

3.3.1 Modifications of the experimental setup ........... 48
3.3.2 Degradation of the signal ........................ 49
3.3.3 Noise in fluorescence measurements ............... 52

a) Dark-current noise .................................. 53
b) Flicker noise ......................................... 53
c) Photon noise ......................................... 53

3.3.4 Smoothing ......................................... 55
3.3.5 Analysis of the experimental noise ............... 58

a) Correlation with the intensity fluctuations .......... 60
b) Noise autocorrelation ............................... 61
c) Signal to noise ratio improvement .....................63

3.3.6 The measurement procedure ..........................64

IV-SpecProc: a program for spectra analysis ..............65

4.1 Purpose ..................................................66

4.2 Main features .............................................67

4.2.1 Load a spectrum ........................................67

4.2.2 Linear combination .....................................69

4.2.3 Smoothing ...............................................70

4.2.4 Statistical analysis ....................................71

4.2.5 List save and delete ....................................72

4.2.6 Quit .....................................................72

4.2.7 Graphic capabilities ....................................73

4.2.8 Advanced operation ....................................74

4.3 Internal structure ........................................74

4.3.1 The variable structure ................................74

4.3.2 The structure ...........................................76

V-Deconvolution Techniques ................................78

5.1 Modelization of fluorescence ...........................78

5.1.1 Beer-Lambert law .....................................78

5.1.2 A more sophisticated model ..........................82

5.2 Multivariate analysis techniques .........................89
5.2.1 Multiple linear regression ........................................ 90
5.2.2 Principal component regression ............................... 92
5.2.3 Partial least-squares regression ............................... 96

VI-Applications to amino acids mixtures .......................... 100
  6.1 Multiple linear regression ..................................... 100
  6.2 Correction for absorption ..................................... 103
  6.3 Application of partial-least squares regression .......... 108

VII-Conclusion and future research ................................. 112

Appendix A: Fermentation measurements .......................... 116

Appendix B: Amino acids mixtures measurements ............... 131

Appendix C: Format of the data file .............................. 143

Appendix D: SpecProc listings ..................................... 145

Appendix E: Partial least squares program listing .......... 172

Bibliography .................................................................. 180
List of Tables

Table 2.1 : Substituent effects on fluorescence .........................13
Table 2.2 : Important biological fluorophores ..........................19
Table 3.1 : Tryptophan degradation .....................................51
Table 3.2 : Reproducibility of tryptophan .............................52
Table 3.3 : Correlation of smoothed spectra .........................58
Table 3.4 : Characteristics of the laser noise ......................61
Table 3.5 : Autocorrelation of the spectrum noise ................62
Table 3.6 : Autocorrelation of corrected spectrum noise ..........63
Table 6.1 : Multilinear estimation of the composition ..............100
Table 6.2 : Estimation of the absorbance .............................105
Table 6.3 : Multilinear estimation corrected for absorption ..........105
List of Figures

Figure 2.1: Absorption spectrum of tryptophan .......................... 8
Figure 2.2: Energy levels in photoluminescence ......................... 11
Figure 2.3: Spectrum of distilled water ................................. 14
Figure 2.4: Variation of fluorescence intensity with temperature ...... 18
Figure 2.5: Biomass concentration and fluorescence signal ............ 21
Figure 2.6: NADH fluorescence and partial pressure of $O_2$ .......... 22
Figure 2.7: Influence of stirrer speed on fluorescence .................. 22
Figure 3.1: Experimental setup ....................................... 25
Figure 3.2: Four-level laser ............................................ 27
Figure 3.3: Dye laser-cavity ............................................ 29
Figure 3.4: GCA/McPherson EU-700 monochromator .................... 32
Figure 3.5: Time resolved fluorescence ................................ 38
Figure 3.6: Excitation-Emission matrix ................................ 39
Figure 3.7: Two-photon fluorescence measurement ....................... 41
Figure 3.8: Tryptophan, tyrosine and phenylalanine chemical structures 42
Figure 3.9: Absorption spectra of tryptophan, tyrosine and phenylalanine 44
Figure 3.10: Fluorescence spectra of tryptophan, tyrosine and phenylalanine 44
Figure 3.11: Measured spectra of tryptophan and tyrosine ............ 47
Figure 3.12: Spectra of tryptophan fresh and after 10 and 20 minutes of irradiation

Figure 3.13: Another example of tryptophan degradation

Figure 3.14: Intensity of laser during an experiment

Figure 3.15: Half of the real part of a spectrum DFT

Figure 3.16: First terms of the Fourier transform

Figure 3.17: Raw and smoothed spectra of tryptophan

Figure 3.18: Two smoothed spectra of tryptophan

Figure 3.19: Noise from two different measurements

Figure 3.20: Laser intensity signal and noise

Figure 3.21: Laser intensity noise and spectrum noise

Figure 5.1: Repartition of fluorescence emission

Figure 5.2: Effect of the detector position

Figure 5.3: Mixture and pure components spectra

Figure 5.4: Reference axis

Figure 5.5: Mean centering and variance scaling

Figure 5.6: Prediction capabilities

Figure 5.7: Principal component analysis model

Figure 5.8: Outer-relation of Partial Least Squares

Figure 5.9: Partial Least Squares algorithm

Figure 6.1: Multilinear regression on amino acid mixtures
Figure 6.2: Pure tryptophan and tyrosine spectra and mixture spectrum
102

Figure 6.3: Corrected estimates .............................................106
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 \((10^{-11} - 10^{-7} \text{ s})\) and this is called fluorescence, or much longer \((10^{-5} - 10 \text{ s})\) 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 $h\nu/\lambda$, where $\lambda$ is the wavelength and $h\nu = 1.9865 \cdot 10^{-25} \text{ J.m}$ 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.

![Absorption spectrum of tryptophan 10^{-4} M.](image)

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 lose 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 π*

* Substituents can dramatically influence the fluorescence quantum yield
Aromatic substituent effects on fluorescence

<table>
<thead>
<tr>
<th></th>
<th>Effect on wavelength</th>
<th>Effect on Intensity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alkyl</td>
<td>None</td>
<td>Slight</td>
</tr>
<tr>
<td>OH, OAlkyl</td>
<td>Red shift</td>
<td>Increase, but be aware of pH effects</td>
</tr>
<tr>
<td>COOH</td>
<td>Red shift</td>
<td>Significant decrease</td>
</tr>
<tr>
<td>NH2, NAlkyl</td>
<td>Red shift</td>
<td>Significant increase, be aware of pH effects</td>
</tr>
<tr>
<td>NO2, NO</td>
<td>Red shift</td>
<td>Significant decrease</td>
</tr>
</tbody>
</table>

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.

**Figure 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 (–NH₂). 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° C and 30° 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). C — C, tryptophan or indoleacetic acid; • — •, indoleacetic acid in buffer saturated with benzene; ▲ — ▲, tyrosine; ■ — ■, quinine in 0.1 N sulfuric acid.

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 F₄₂₀, NADH or FADH. Table 2.2 shows some of the important biological fluo-
<table>
<thead>
<tr>
<th>Fluorophores</th>
<th>Maximum Wavelength</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Excitation</td>
<td>Emission</td>
<td></td>
</tr>
<tr>
<td>NADH, NADPH</td>
<td>350</td>
<td>450</td>
<td></td>
</tr>
<tr>
<td>FAD</td>
<td>450</td>
<td>520</td>
<td></td>
</tr>
<tr>
<td>Aromatic Amino Acids</td>
<td>275</td>
<td>340</td>
<td></td>
</tr>
<tr>
<td>Nucleotides</td>
<td>280</td>
<td>375</td>
<td></td>
</tr>
<tr>
<td>Riboflavins</td>
<td>445</td>
<td>520</td>
<td></td>
</tr>
<tr>
<td>Antibiotics</td>
<td>var.</td>
<td>var.</td>
<td></td>
</tr>
<tr>
<td>$F_{420}$</td>
<td>425</td>
<td>472</td>
<td></td>
</tr>
</tbody>
</table>

*Table 2.2. Luminescence peaks of the important biological fluorophores.*

Fluorophores 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 antigens or antibodies

* Time resolved techniques using fluorescent markers with very long lifetime like europium chelates for labeling antigens 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 Schepers [1986]).

in figure 2.7.

The tremendous variation of the fluorescence signal with the stirrer speed is explained by Schepers 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 fluores-
cence 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
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 violet light 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.
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 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 $100 \text{MWcm}^{-2}$.

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

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 $1 \times 1 \text{cm}$ quartz cuvette which contains the sample. The incidence is approximately $10^\circ$. A high quality quartz cuvette is indispensable because ordinary glass absorbs the light at $280 \text{nm}$ 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 lens 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 $14 \text{mm}$ 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 \(\alpha_l\) 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
an angle depending on the wavelength. The grating equation is:

\[ m\lambda = d(\sin i + \sin \theta) \]

where \( i \) is the angle of incidence, \( \theta \) the angle of diffraction, \( d \) the distance between two rules, \( m \) the order of the interference and \( \lambda \) 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 scanned.

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 $48 \times 48 \text{mm}$ ruled area at 1180 $\text{lines/mm}$. 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 coming 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 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 $-Nv/20\ n m$ 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 $1 \times 1 \text{cm}$ 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 305 nm, etc.). The scanning speed was usually $10 \AA/s$ 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 $7 \times 10 \text{ mm}$ 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 10Å/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

MEAN LIFETIME (ns) = 12.7245
S.D. OF LIFETIME (ns) = 0.383E-01
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 describe the fluorescence. The fluorescence matrix obtained by the videofluorometer is composed as follows:

* Each line is an emission spectrum at the excitation wavelength $\lambda_{ex}^i$.
* Each column is an excitation spectrum at the emission wavelength $\lambda_{em}^j$.

This means that the term $m_{i,j}$ is the intensity of the fluorescence emitted at the wavelength $\lambda_{em}^j$ when the sample is illuminated at the wavelength $\lambda_{ex}^i$. 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 from 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
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.

![Chemical structures of Phenylalanine, Tyrosine, and Tryptophan]

*Fig.3.8. Tryptophan, tyrosine 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 Wetlauffer [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-
The strong influence of temperature on tryptophan fluorescence has already been mentioned in Chapter 2. The intensity of the fluorescence decreases of 5% per degree Celsius between $20^\circ C$ and $30^\circ C$. The pH of the medium induces slight changes in the shape of the tryptophan fluorescence spectrum. $\text{R-C-COOH}^{\text{NH}_3^+}$ peaks at 347 nm, $\text{R-C-COO}^-^{\text{NH}_4^+}$ at 353 nm and $\text{R-C-COO}^-^{\text{NH}_2}$ 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. $\text{R-C-COOH}^{\text{NH}_3^+}$ has a high yield of 0.51, $\text{R-C-COO}^-^{\text{NH}_4^+}$ a yield of 0.20 and $\text{R-C-COO}^-^{\text{NH}_2}$ 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 $10^{-6} \text{ mol/l}$. 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 $s_{1\lambda}$ and $s_{2\lambda}$ 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.

<table>
<thead>
<tr>
<th>Sample</th>
<th>Ratio</th>
<th>Correlation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fresh sample</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>After 10 min</td>
<td>0.65</td>
<td>0.999</td>
</tr>
<tr>
<td>After 20 min</td>
<td>0.50</td>
<td>0.997</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Sample</th>
<th>Ratio</th>
<th>Correlation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fresh sample</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>After 10 min</td>
<td>0.73</td>
<td>0.998</td>
</tr>
<tr>
<td>After 20 min</td>
<td>0.68</td>
<td>0.998</td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>Sample</th>
<th>Signal Mean</th>
<th>Signal Variance</th>
<th>Intensity Mean</th>
<th>Correlation</th>
<th>Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>Trp3a</td>
<td>1.043</td>
<td>0.58</td>
<td>3.43</td>
<td>0.998</td>
<td>1.029</td>
</tr>
<tr>
<td>Trp4a</td>
<td>1.053</td>
<td>0.59</td>
<td>3.24</td>
<td>0.998</td>
<td>1.020</td>
</tr>
<tr>
<td>Trp5a</td>
<td>1.035</td>
<td>0.63</td>
<td>3.10</td>
<td>0.998</td>
<td>1.019</td>
</tr>
<tr>
<td>Trp6a</td>
<td>1.070</td>
<td>0.62</td>
<td>2.81</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

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 kinds 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 cannot 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
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:

\[ I_\phi = \sqrt{eBM/t} \sqrt{I} \]  \hspace{1cm} (1)
\[ B = 1 + 1/g + 1/g^2 + \cdots \]

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

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

The intensity used in the measurements is in the order of \( 2 \text{ mA} \) therefore the resulting noise is \( 0.36 \text{ mA} \) 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
transform is symmetrical. The lower frequencies are represented by the extremes 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 different 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
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.

<table>
<thead>
<tr>
<th>Sample</th>
<th>Ratio</th>
<th>Correlation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Original spectra</td>
<td>0.969</td>
<td>0.989</td>
</tr>
<tr>
<td>Smoothed spectra</td>
<td>0.980</td>
<td>0.999</td>
</tr>
</tbody>
</table>

*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
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

<table>
<thead>
<tr>
<th>Shift (nm)</th>
<th>0.1</th>
<th>0.2</th>
<th>0.3</th>
<th>0.4</th>
<th>0.5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Correlation</td>
<td>0.328</td>
<td>0.157</td>
<td>0.048</td>
<td>0.016</td>
<td>-0.047</td>
</tr>
<tr>
<td>Shift (nm)</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>Correlation</td>
<td>0.113</td>
<td>-0.026</td>
<td>0.102</td>
<td>-0.097</td>
<td>-0.124</td>
</tr>
</tbody>
</table>

Table 3.4. Characteristics of the laser intensity noise.
This signal is very close of a white noise. The values of autocorrelation do not show any significant correlation. 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-
<table>
<thead>
<tr>
<th>Shift (nm)</th>
<th>0.1</th>
<th>0.2</th>
<th>0.3</th>
<th>0.4</th>
<th>0.5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Correlation</td>
<td>0.973</td>
<td>0.919</td>
<td>0.857</td>
<td>0.795</td>
<td>0.734</td>
</tr>
<tr>
<td>Shift (nm)</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>Correlation</td>
<td>0.371</td>
<td>-0.259</td>
<td>-0.467</td>
<td>-0.244</td>
<td>-0.057</td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>Shift (nm)</th>
<th>0.1</th>
<th>0.2</th>
<th>0.3</th>
<th>0.4</th>
<th>0.5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Correlation</td>
<td>0.809</td>
<td>0.717</td>
<td>0.634</td>
<td>0.565</td>
<td>0.519</td>
</tr>
<tr>
<td>Shift (nm)</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>Correlation</td>
<td>0.341</td>
<td>-0.173</td>
<td>-0.329</td>
<td>-0.187</td>
<td>-0.009</td>
</tr>
</tbody>
</table>

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 Ås⁻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 \texttt{.tex} rather than \texttt{.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.

\subsection{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 $\texttt{comb}_n$ 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 l 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 curveffect 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 $\epsilon$, 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:

\[ \Phi = \frac{\text{number of photons emitted}}{\text{number of photons absorbed}} \]

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.

![Spectra of a Mixture](image)

**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 $\lambda_{ex}$, 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 $\lambda_{em}$, $I_0$ is the intensity of the excitation light, $a_i^{\lambda_{ex}}$ and $a_i^{\lambda_{em}}$ are the absorption coefficients at $\lambda_{ex}$ and $\lambda_{em}$ 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 $\lambda_{ex}$ 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 $I = I_0 \exp(-x A^{\lambda_{ex}})$. 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 $\lambda_{em}$ 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 \( \Gamma \) 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 \( \Gamma \) 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

\[ \frac{F_{\lambda_{em}}}{\Gamma(A_{\lambda_{ex}}, A_{\lambda_{em}})} = I_0 \sum_{i=1}^{n} \frac{c_{i}}{c_{i_0}} \frac{S_{i}^{\lambda_{em}}}{\Gamma(A_{i_0}^{\lambda_{ex}}, A_{i_0}^{\lambda_{em}})} \]

A linear dependence between the spectra divided by \( \Gamma \) 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 $\Gamma$. 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 $\lambda_{ez}$ and fluoresce at $\lambda_{em}$

* the second absorbs at $\lambda_{ez}$ but do not fluoresce
the third absorbs at $\lambda_{em}$ but not at $\lambda_{ex}$

One can measure $\Gamma$ 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 $\Gamma$. In the preceding equation (2), all the terms are divided by $\Gamma$ therefore any function proportional to $\Gamma$ can be used as well in this equation. In particular the fluorescence signal measured can be used as $\Gamma$ in this equation. The calibration of $\Gamma$ for any setup is therefore be very easy. The application of this model to the National Bureau of Standards setup is now considered.

![Figure 5.4. Reference axis.](image)

Using the convention defined in Figure 5.4, the absorption law gives:
\[ I = I_0 \exp \left( -\frac{x}{\cos \theta} A^{\lambda_{zz}} \right) \]

Therefore the function \( g \) defined in the preceding section is:

\[ g(x, y, z, A^{\lambda_{zz}}) = \exp \left( -\frac{x}{\cos \theta} A^{\lambda_{zz}} \right) \]

The amount of fluorescence that reaches the detector depends on 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 \( \exp(-xA^{\lambda_{zm}}) \). Therefore the function \( h \) is:

\[ h(x, y, z, A^{\lambda_{zm}}) = \psi(x, r) \exp(-xA^{\lambda_{zm}}) \]

And \( \Gamma \) is given by:

\[ \Gamma(A^{\lambda_{zz}}, A^{\lambda_{zm}}) = \int_V \psi(x, r) \exp \left( -x \left( \frac{A^{\lambda_{zz}}}{\cos \theta} + A^{\lambda_{zm}} \right) \right) \]

Therefore one can see that the function \( \Gamma \) is in fact a function of one variable only \( \frac{A^{\lambda_{zz}}}{\cos \theta} + A^{\lambda_{zm}} \). 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 \( \Gamma \) 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:

\[
\Gamma(A_{\lambda_{ex}}, A_{\lambda_{em}}) = k \int_{x=0}^{1} \exp(-\pi \left( \frac{A_{\lambda_{ex}}}{\cos \theta} + A_{\lambda_{em}} \right))
\]

Where \( k \) is a constant. The integration gives:

\[
\Gamma(A_{\lambda_{ex}}, A_{\lambda_{em}}) = \frac{k}{A_{\lambda_{ex}}/\cos \theta + A_{\lambda_{em}}} \left( 1 - \exp\left( \frac{A_{\lambda_{ex}}}{\cos \theta} + A_{\lambda_{em}} \right) \right)
\]

This value of \( \Gamma \) 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:

$$y = b_1 x_1 + b_2 x_2 + \cdots + b_m x_m + e$$

$e$ 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.

![Figure 5.5](image)

**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-
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 explained 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 $p'_h : p'_h = t'_h X/t'_ht_h$
3. normalize $p'_h$ to length 1 : $p'_{h_{new}} = p'_{h_{old}} / \| p'_{h_{old}} \|$  
4. calculate $t_h : t_h = Xp_h/p'_hp_h$
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 $E_h : E_h = X - t_hp'_h$
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
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 compo-
nents 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:

\[
Y = TB + E
\]

\[\begin{array}{c}
Y \\
T \\
B \\
E
\end{array}\]

**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.

95
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.

![Diagram of matrix relations](image)

*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:

* \( p'_h \) and \( q'_h \) have unit length
* \( t_h \) and \( u_h \) are centered around zero for each \( h \)
* \( w_h \) 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 al., 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 $u_{\text{start}} = \text{some } y_i$.

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

(3) $w_{\text{new}} = w_{\text{old}}/\|w_{\text{old}}\|$ (normalization)

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

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

(6) $q_{\text{new}} = q_{\text{old}}/\|q_{\text{old}}\|$ (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) $p_{\text{new}} = p_{\text{old}}/\|p_{\text{old}}\|$ (normalization)

(11) $t_{\text{new}} = t_{\text{old}}/\|t_{\text{old}}\|$

(12) $w_{\text{new}} = w_{\text{old}}/\|w_{\text{old}}\|$

($p'$, $q'$ 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

$E_h = E_{h-1} - t_h p_h; X = E_1$

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

$F_h = F_{h-1} - b_h t_h q_h; Y = F_1$

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 $X'Y Y'X$ 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 $a$

   $t_h = E_{h-1}w_h$
   
   $E_h = E_{h-1} - t_hp'_h$
   
   $Y = Y + b_h t_h q'_h$

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.

<table>
<thead>
<tr>
<th>Sample</th>
<th>tryptophan Real</th>
<th>tyrosine Real</th>
<th>tryptophan Estimated</th>
<th>tyrosine Estimated</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mix1a</td>
<td>0.5</td>
<td>0.5</td>
<td>0.778</td>
<td>0.239</td>
</tr>
<tr>
<td>Mix2a</td>
<td>0.5</td>
<td>0.5</td>
<td>0.747</td>
<td>0.217</td>
</tr>
<tr>
<td>Mix3a</td>
<td>0.75</td>
<td>0.25</td>
<td>0.798</td>
<td>0.071</td>
</tr>
<tr>
<td>Mix4a</td>
<td>0.75</td>
<td>0.25</td>
<td>0.851</td>
<td>0.097</td>
</tr>
<tr>
<td>Mix5a</td>
<td>0.25</td>
<td>0.75</td>
<td>0.517</td>
<td>0.459</td>
</tr>
<tr>
<td>Mix6a</td>
<td>0.25</td>
<td>0.75</td>
<td>0.502</td>
<td>0.461</td>
</tr>
<tr>
<td>Mix7a</td>
<td>0.1</td>
<td>0.9</td>
<td>0.297</td>
<td>0.698</td>
</tr>
<tr>
<td>Mix8a</td>
<td>0.1</td>
<td>0.9</td>
<td>0.273</td>
<td>0.836</td>
</tr>
</tbody>
</table>

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 \(\alpha\) and \(\beta\) 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 \( \Gamma \) is needed.

The estimation of \( \Gamma \) 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:

\[ \Gamma(A) = \frac{k \cos \theta}{A} (1 - \exp\left(-\frac{A}{\cos \theta}\right)) \]

Since \( \theta \) 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 \( \Gamma \) 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.
<table>
<thead>
<tr>
<th>tryptophan (10^{-3} \text{ mol/l})</th>
<th>tyrosine (10^{-3} \text{ mol/l})</th>
<th>Absorbance</th>
<th>(\Gamma(A))</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>11.5</td>
<td>0.087</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>3.45</td>
<td>0.28</td>
</tr>
<tr>
<td>0.75</td>
<td>0.25</td>
<td>9.49</td>
<td>0.105</td>
</tr>
<tr>
<td>0.5</td>
<td>0.5</td>
<td>7.47</td>
<td>0.134</td>
</tr>
<tr>
<td>0.25</td>
<td>0.75</td>
<td>5.46</td>
<td>0.182</td>
</tr>
<tr>
<td>0.1</td>
<td>0.9</td>
<td>4.26</td>
<td>0.231</td>
</tr>
</tbody>
</table>

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

Using these values of \(\Gamma\), 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.

<table>
<thead>
<tr>
<th>Sample</th>
<th>tryptophan Real</th>
<th>tyrosine Real</th>
<th>tryptophan Estimated</th>
<th>tyrosine Estimated</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mix1a</td>
<td>0.5</td>
<td>0.5</td>
<td>0.51</td>
<td>0.50</td>
</tr>
<tr>
<td>Mix2a</td>
<td>0.5</td>
<td>0.5</td>
<td>0.49</td>
<td>0.45</td>
</tr>
<tr>
<td>Mix3a</td>
<td>0.75</td>
<td>0.25</td>
<td>0.66</td>
<td>0.19</td>
</tr>
<tr>
<td>Mix4a</td>
<td>0.75</td>
<td>0.25</td>
<td>0.71</td>
<td>0.26</td>
</tr>
<tr>
<td>Mix5a</td>
<td>0.25</td>
<td>0.75</td>
<td>0.25</td>
<td>0.71</td>
</tr>
<tr>
<td>Mix6a</td>
<td>0.25</td>
<td>0.75</td>
<td>0.24</td>
<td>0.71</td>
</tr>
<tr>
<td>Mix7a</td>
<td>0.1</td>
<td>0.9</td>
<td>0.11</td>
<td>0.85</td>
</tr>
<tr>
<td>Mix8a</td>
<td>0.1</td>
<td>0.9</td>
<td>0.10</td>
<td>1.01</td>
</tr>
</tbody>
</table>

*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 \(10^{-3} \text{ M}\).*

The quality of the results is surprising when one considers the numerous approximations made in the computation of \(\Gamma\) 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
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 $\frac{hc}{\lambda} = 7 \times 10^{-19}$ J. Therefore a pulse contains $1.7 \times 10^{16}$ photons. In the cuvette, the intensity to be absorbed in a slice $dx$ is given by: $2.36cI_0dx$. The number of molecules in this slice $dx$ is: $6 \times 10^{23}csdx$ 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 $s = 1.96 \times 10^{-4}$. The number of molecules in the slice is therefore: $1.18 \times 10^{20}cdx$. Considering the first slice hit by the beam, the intensity to be absorbed is $2.36cI_0dx$ or expressing this in term of photons, $2.36c1.7 \times 10^{16}dx$ photons should be absorbed. Since the molar absorptivity of tryptophan is 5000, $1.96 \times 10^{20}cdx$ 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: $I = I_0 - 1.18 \times 10^{-20}cdx$. Therefore in a
$10^{-3}$ M solution the number of photons to be absorbed becomes equivalent to the number of molecules after $6 \times 10^{-2} cm$. 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 tyrosine and a mixture 0.25 mM tryptophan 0.75 mM tyrosine. The five spectra are stored in a $5 \times 30$ matrix and the corresponding concentrations in a $5 \times 2$ 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
Table 6.4. Partial Least Squares estimation of the composition of a millimolar tryptophan solution. The concentrations are given in mM.

<table>
<thead>
<tr>
<th></th>
<th>Real</th>
<th>Order 1</th>
<th>Order 2</th>
<th>Order 3</th>
<th>Order 4</th>
<th>Order 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tryptophan</td>
<td>1.0</td>
<td>0.868</td>
<td>0.957</td>
<td>0.993</td>
<td>1.0</td>
<td>1.0</td>
</tr>
<tr>
<td>Tyrosine</td>
<td>0.0</td>
<td>0.132</td>
<td>0.043</td>
<td>0.007</td>
<td>$10^{-16}$</td>
<td>$10^{-16}$</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th></th>
<th>Real</th>
<th>Order 1</th>
<th>Order 2</th>
<th>Order 3</th>
<th>Order 4</th>
<th>Order 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tryptophan</td>
<td>0.0</td>
<td>-0.090</td>
<td>-0.006</td>
<td>0.004</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>Tyrosine</td>
<td>1.0</td>
<td>1.09</td>
<td>1.006</td>
<td>0.996</td>
<td>1.0</td>
<td>1.0</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th></th>
<th>Real</th>
<th>Order 1</th>
<th>Order 2</th>
<th>Order 3</th>
<th>Order 4</th>
<th>Order 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tryptophan</td>
<td>0.5</td>
<td>0.630</td>
<td>0.501</td>
<td>0.515</td>
<td>0.5</td>
<td>0.5</td>
</tr>
<tr>
<td>Tyrosine</td>
<td>0.5</td>
<td>0.370</td>
<td>0.499</td>
<td>0.485</td>
<td>0.5</td>
<td>0.5</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th></th>
<th>Real</th>
<th>Order 1</th>
<th>Order 2</th>
<th>Order 3</th>
<th>Order 4</th>
<th>Order 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tryptophan</td>
<td>0.75</td>
<td>0.723</td>
<td>0.802</td>
<td>0.754</td>
<td>0.75</td>
<td>0.75</td>
</tr>
<tr>
<td>Tyrosine</td>
<td>0.25</td>
<td>0.277</td>
<td>0.198</td>
<td>0.246</td>
<td>0.25</td>
<td>0.25</td>
</tr>
</tbody>
</table>


<table>
<thead>
<tr>
<th></th>
<th>Real</th>
<th>Order 1</th>
<th>Order 2</th>
<th>Order 3</th>
<th>Order 4</th>
<th>Order 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tryptophan</td>
<td>0.25</td>
<td>0.368</td>
<td>0.246</td>
<td>0.234</td>
<td>0.25</td>
<td>0.25</td>
</tr>
<tr>
<td>Tyrosine</td>
<td>0.75</td>
<td>0.631</td>
<td>0.754</td>
<td>0.766</td>
<td>0.75</td>
<td>0.75</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th></th>
<th>Order 1</th>
<th>Order 2</th>
<th>Order 3</th>
<th>Order 4</th>
<th>Order 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Error</td>
<td>0.114</td>
<td>0.0092</td>
<td>0.0011</td>
<td>0.0</td>
<td>0.0</td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th></th>
<th>Real</th>
<th>Order 1</th>
<th>Order 2</th>
<th>Order 3</th>
<th>Order 4</th>
<th>Order 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tryptophan</td>
<td>0.5</td>
<td>0.621</td>
<td>0.536</td>
<td>0.509</td>
<td>0.496</td>
<td>0.502</td>
</tr>
<tr>
<td>Tyrosine</td>
<td>0.5</td>
<td>0.379</td>
<td>0.464</td>
<td>0.491</td>
<td>0.504</td>
<td>0.498</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th></th>
<th>Real</th>
<th>Order 1</th>
<th>Order 2</th>
<th>Order 3</th>
<th>Order 4</th>
<th>Order 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tryptophan</td>
<td>0.75</td>
<td>0.720</td>
<td>0.727</td>
<td>0.732</td>
<td>0.743</td>
<td>0.742</td>
</tr>
<tr>
<td>Tyrosine</td>
<td>0.25</td>
<td>0.280</td>
<td>0.273</td>
<td>0.268</td>
<td>0.257</td>
<td>0.258</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th></th>
<th>Real</th>
<th>Order 1</th>
<th>Order 2</th>
<th>Order 3</th>
<th>Order 4</th>
<th>Order 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tryptophan</td>
<td>0.25</td>
<td>0.351</td>
<td>0.292</td>
<td>0.294</td>
<td>0.297</td>
<td>0.313</td>
</tr>
<tr>
<td>Tyrosine</td>
<td>0.75</td>
<td>0.649</td>
<td>0.708</td>
<td>0.706</td>
<td>0.703</td>
<td>0.687</td>
</tr>
</tbody>
</table>


<table>
<thead>
<tr>
<th></th>
<th>Real</th>
<th>Order 1</th>
<th>Order 2</th>
<th>Order 3</th>
<th>Order 4</th>
<th>Order 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tryptophan</td>
<td>0.1</td>
<td>0.152</td>
<td>0.128</td>
<td>0.149</td>
<td>0.151</td>
<td>0.148</td>
</tr>
<tr>
<td>Tyrosine</td>
<td>0.9</td>
<td>0.848</td>
<td>0.872</td>
<td>0.851</td>
<td>0.849</td>
<td>0.852</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th></th>
<th>Real</th>
<th>Order 1</th>
<th>Order 2</th>
<th>Order 3</th>
<th>Order 4</th>
<th>Order 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tryptophan</td>
<td>0.1</td>
<td>0.116</td>
<td>0.039</td>
<td>0.094</td>
<td>0.089</td>
<td>0.076</td>
</tr>
<tr>
<td>Tyrosine</td>
<td>0.9</td>
<td>0.884</td>
<td>0.961</td>
<td>0.906</td>
<td>0.911</td>
<td>0.923</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Order 1</th>
<th>Order 2</th>
<th>Order 3</th>
<th>Order 4</th>
<th>Order 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Error</td>
<td>0.057</td>
<td>0.0162</td>
<td>0.0096</td>
<td>0.001</td>
<td>0.0137</td>
</tr>
</tbody>
</table>

*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 $pO_2$ 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 *Saccharomyces 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 $t=5$ h.
Fig. A.2. Fluorescence spectrum of a fermentation broth at $t=71\text{h}$. 
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 0.1 - Tryptophan 10^-4 M/1
Fig 8-2. Tyrosine $10^{-3}$ $M/L$
Fig. 8-3. Mixture 5x10^{-4}M tryptophan, 5x10^{-4}M tyrosine.
Fig. 6-4. Mixture 2.5 \times 10^{-4}$ tryptophan, $2.5 \times 10^{-4}$ tyrosine.
Figure 6-5. Mixture $7.5 \times 10^{-4}$ tryptophan, $2.5 \times 10^{-4}$ tyrosine.
Fig 8-6. Mixture 2.5 \times 10^{-4} M tryptophan + 2.5 \times 10^{-4} M tyrosine.
Fig 6-7. Titration 2.5 x 10^{-4} M tryptophan 7.5 x 10^{-4} M Tris buffer.
Fig 8.8. Mixture of $10^{-4}$ M Tyrosine and $10^{-4}$ M Tryptophan.
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.
program SpecProcessing;

(constant declaration)

const MaxMaxNumber = 3000;  (* maximum size of the spectrum file *)
DataArraySize = 400;      (* size of the linked arrays used to store the data *)
NarraySize = 204;         (* size of the arrays used in the FFT *)

(type value for the data file)
Fluorescence = 1;
NormedFluo = 2;
LaserPower = 3;
Continue = 5;
FourierTr = 11;

(file for the default value)
Default = 'SPECPROC.DAT';

(type definition)

(type LabelString = string[5]; (* used for the line numbers, label *)
    Name = string[50];         (* used for the following *)
    Line = string[100];        (* used for a screen line *)
    BigString = string[255];   (* used by some string procedure *)
    MenuLine = record          (* used for the menu procedure *)
        Title: Name;
        Type: integer;
        Real: boolean;
        Str: Name;
    end;

MenuTab = array [1..5] of MenuLine; (* input/output of the menu procedure *)

DataTab = array [1..DataArraySize] of real; (* used to store the data *)

(type spectrum used to store the spectra)

Grdata = record (* information from the *)
    OnOff,Power = boolean;
    PowScale = real;
    line = true;
    MinX,MaxX,Shift,Xscale = real;
    MinY,MaxY,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 (of a spectrum)
  Source,LocalName : Name;
  MeasureNumber : integer;
  Xmin,Increment : real;
  Measure1 : Dpont;
  Measure2 : Dpont;
  Graph : Grdata;
  Link : Pointer;
end;

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

AsDef = record // used to draw the a:
  Title,Label,:Labely : Name;
  Xmin,Xmax,DeltaX : real;
  Ymin,Ymax,DeltaY : real;
end;

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

Gansw = record // used for input
  Int : integer;
  Re : real;
  Chr : char;
  Str : Name;
  Typ : UT;type;
end;

*****************************************************************************
(* VARIABLES DECLARATION *)
*****************************************************************************
var
  Current_Dir : Name;
  Head : Pointer; // head of the spectra list
linked list
  CurveNum : array[1..9] of Pointer; // axes generally used
  Axis : Asdef;
  MainCh : char;
  lord : integer;
  AscreenScale,Anig,AscreenScale,Anig : real; // scale used for plotting
  Default_Type : boolean;
  Gansw : Gansw;
  Stortect,ScreenWave,LastShift : real;
  Fxstart,Fxend,y0,y1,yn : real;
  Textcol,Anstcl,Background : integer;
*****************************************************************************
(* INCLUDED FILES *)
*****************************************************************************
(*lib:)
(*glgraphlib)
(*glclib)
(*glinit)
(*glstiman)
(*glmenus)
(*glvis)
(*glview)
(*glplot)
(*glpengl)
(*glcharsize)
(*gltconvert)
(*fftinc)
(*inffft.inc)
(*inteff.fft.inc)
(*lprintf)
(*gle2f)
(*plfplot)
(*gplotlib)
(*glview)
(*glimage)
(*glimageplus)
BEGINNING OF THE CODE

begin
head := nil;
ini;
MainCh := Chr(120);
repeat
  lord := Ord(MainCh);
  if lord = 27 then SpecPlot(MainCh)
  else
    if lord = 31 then SpecTest(MainCh)
    else
      if (lord < 50) and (lord < 68) then Characterize(lord-50, MainCh)
      else
        if lord = 68 then AxisDef(MainCh, MainCh)
        else
          if lord = 121 then Define(MainCh)
          else
            if lord = 122 then LinComb(MainCh)
            else
              if lord = 123 then SmoothMainCh(MainCh)
              else
                if lord = 124 then Stat(MainCh)
                else
                  MainMenu(MainCh);
      until (Ord(MainCh) = 16) ;
quit;
end.
function Min(x, y : integer) : integer;
    begin
        min := x;
        if y < min then min := y;
        end;

function Max(x, y : integer) : integer;
    begin
        Max := x;
        if y > Max then Max := y;
    end;

function Power10(x : real) : real;
    begin
        Power10 := exp(x * log10(10));
    end;

function Log(x : real) : real;
    begin
        Log := ln(x) / ln(10);
    end;

function divide(i, j : integer) : boolean;
    begin
        divide := ((i * j) div j) = (i * (j div j));
    end;

function gcd(i, j : integer) : integer;
    begin
        var h, k, l : integer;
        exit := boolean;
        begin
            h := min(i, j);
            k := max(i, j);
            l := h;
            repeat
                if divide(k, h) then
                    if divide(l, h) then begin
                        gcd := h;
                        exit := true;
                    end;
            until exit;
        end;
    end;
function ppcm(i1,i2 : integer) : integer;
{
    THIS FUNCTION COMPUTES THE SMALLEST COMMON MULTIPLE OF i1 AND i2
}
begin
    ppcm := (i1*i2) div gcd(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 SpecIndir (SpecName : Name; Point : 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
            point := point;
            valid2 := true;
        end;
        point := Point^.Link;
    end;
end;
procedure string(s1 : real; s2 : labelString); forward; (defined later)

function Upstring(Source : BigString) : BigString; forward; (defined later)

(----------------------------------------------------------------------------------------------------)

(*  GRAPHICAL PROCEDURES *)

procedure tex;
begin
   textcolor(textcolor); 
end;

procedure Anz;
begin
   textcolor(anzahl);  
end;

procedure WriteTitle(StrTitle : Name); integer;
begin
   ' writeme[3];
   ' textbackground[background];
   ' direct;
   ' textcolor[yellow];
   gotoxy(1,1);
   write(Upstring(StrTitle));
   tex;
end;

procedure writemem;
begin
   textcolor(lightred);
   gotoxy(66,1):write('MemAvail = ', MemAvail);
   gotoxy(66,2):write('MaxAvail = ', MaxAvail);
   tex;
end;

procedure PlotAxis(axis : anzahl);
{  THIS PROCEDURE DRAW THE AXIS } 
var 
   'x,y1 : real;
   'x2,y2,xstart : integer;
   '1 : string[7];

procedure Arrow(x,y1,y2,xstart:integer);
{  THIS PROCEDURE DRAW THE ARROWS ON THE AXIS }
{  X,Y ARE THE COORDINATE ON THE SCREEN}
{  THAT IS THE ANGLE WITH THE X-AXIS } 
var 
   'x,y1,x2,y2 : integer;

begin
   angle := tord(100*70/4);
   x1 := int(2-4*int(cos(angle)));
   y1 := int(2-4*int(sin(angle)));
   draw(x1,y1,x1,y2);
   angle := tord(180-100*70/4);
   x1 := int(2-4*int(cos(angle)));
   y1 := int(2-4*int(sin(angle)));
   draw(x1,y1,x1,y2);
end;
procedure Bar(x1, y1, x2, y2: integer);

    THIS PROCEDURE DRAWS A BAR AT THE COORDINATES X,Y WITH AN ANGLE ETA.

var x1, y1, x2, y2: integer;
    angle, r1: real;

begin
    angle := theta*pi/180.;
    r1 := (2*r1*abs(cos(angle)));
    x1 := trunc(x1+11*abs(cos(angle)));
    y1 := trunc(y1+11*abs(sin(angle)));
    x2 := trunc(x2+11*abs(cos(angle)));
    y2 := trunc(y2+11*abs(sin(angle)));
    draw(x1, y1, x2, y2);
end:

{ procedure PlotAxis : beginning of the code }

begin
    XScreenScale := 540./(Axis.xmax-Axis.xmin); YScreenScale := 160./(Axis.ymax-Axis.ymin);
    Xorig := Axis.xmin-40*XScreenScale; Yorig := Axis.ymax-10/YScreenScale;
    draw(40, 170, 640, 170); arrow(35, 170, 0);
    i := (trunc(Axis.xmin/Axis.delta)x+1)*Axis.delta;
    repeat
        x := trunc((x1-xorig)*XScreenScale);
        bar(x1, y1, 1, 50);
        string(x, 'x1', 11);
        gotoxy(i, 170);
        v1 := x1 + Axis.delta;
        until (x > Axis.Xmax-1, 001);
        gotoxy(trunc((160-length(Axis.Label))/2), 20); write(Axis.Label);
        draw(40, 170, 400, 170); arrow(40, 170, 0);
        v2 := trunc(y1-YScreenScale);
        bar(40, v2, 1, 50);
        i := i + 5*Axis.delta;
        until (i > Axis.Xmax-1, 001);
        for i := 1 to length(Axis.Label) do
            begin
                gotoxy(i, trunc(160-length(Axis.Label)/2+1)); write(Axis.Label[i], 1, 11);
            end;
        gotoxy(30, 5); write(UpperTitle or Title);
end:

{ procedure PlotAxis : end of the code }
procedure init;
{
    THIS PROCEDURE INITIALIZE THE VARIABLES USED IN SPECPROC
}

var inputfile : text;
    error,1  : integer;
    Dtype    : integer;

begin
    assign(inputfile,'MyFile');
    ($1$)
    reset(inputfile);
    error := ioResult;
    ($1$)
    if Error = 0 then
    begin
        readln(inputfile);
        readln(inputfile,Current_Dir);
        readln(inputfile,Dtype);
        Default_type := true;
        if Dtype = 0 then Default_type := false;
        readln(inputfile);
        readln(inputfile,Axis.Title);
        readln(inputfile,Axis.Labels);
        readln(inputfile,Axis.Min,Axis.Max,Axis.Deltas);
        close(inputfile);
    end
else begin
    Current_Dir := 'Bi';
    Default_type := true;
    Axis.Title := 'Laser Induced Spectrum';
    Axis.Label := 'Wavelength (nm)'
    Axis.Labels := 'Intensity';
    Axis.Min := 250;Axis.Max := 460;Axis.Deltas := 20;
    Axis.Min := 0;Axis.Max := 10;Axis.Deltas := 2;
    xstart := 200;xstart := 300;
    xend := 460;xend := 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,'MyFile');
    rewrite(outputfile);
    writeln(outputfile,'SPECPFERENCE 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 Spectrum Analysis!!!
    writeln(outputfile,Current_Dir,Current_Dir);
    writeln(outputfile,'For Spectrum Analysis!!!
    writeln(outputfile,xstart,xend);
    writeln(outputfile,'For Spectrum Analysis!!!
    writeln(outputfile,Axis.Title);
    writeln(outputfile,Axis.Labels);
    writeln(outputfile,Axis.Min,Axis.Max,Axis.Deltas);
    writeln(outputfile,'SPECPFERENCE DEFAULT VALUES!!!
    writeln(outputfile,Current_Dir,Current_Dir,
    writeln(outputfile,xstart,xend);
    writeln(outputfile,Axis.Title);
    writeln(outputfile,Axis.Labels);
    writeln(outputfile,Axis.Min,Axis.Max,Axis.Delta
    writeln(outputfile,'SPECPFERENCE DEFAULT VALUES!!!
    writeln(outputfile,Current_Dir,Current_Dir,
    writeln(outputfile,xstart,xend);
    writeln(outputfile,Axis.Title);
    writeln(outputfile,Axis.Labels);
    writeln(outputfile,Axis.Min,Axis.Max,Axis.Deltas);
```plaintext
procedure SpecDel(Spec : Spectrum);
{
THIS FUNCTION CREATES A NEW PLACE IN THE LINKED LIST AND INSERTS SPECTRUM
}
var Point : Pointer;
y : integer;

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

-------------

procedure Datadel (point : Pointer);
{
ERASE A LINKED LIST OF DATA
}

begin
if point <> nil then
begin
Datadel (point^.data^.link);
dispose(point);
end;
end;
-------------

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

var lastPoint : Pointer;
Found : boolean;

-------------

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

begin
Datadel(specPoint^.measure);
Datadel(specPoint^.measure2);
dispose(specPoint);
end;
-------------

{BEGINNING OF SPECDEL CODE}
begin
if Head^.measure = 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;
next^.Link := next^.Link^.Link;
next := next^.Link;
end
end;
end;
end;
end;
```
procedure curve_effect(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;
                spec.indic(spec.localname.point);
                curvenum[i] := Point;
            end;
            i := i+1;
        until ((i > 0) or exit);
    end;

------------------------------------------------------------------

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

var quit,test,errno : boolean;
    x,y,z : real;
    i,j : integer;
    answer,defaultName : name;
    Name,Namefile : name;
    count : integer;
    cutfile : text;
    point1,point2 : point;
    Spoint : pointer;
    spec : spectrum;

colors:
    INITIALIZATION OF THE COLORS
    }
    textcol := white;
    Arccol := lightmagenta;
    Background := blue;
    textmode(3);
    textBackgroundColor(background);
    cirsrc;
    Quit := false;
    textcolor(yellow);
    gotoxy(30,5);write("SAVE A SPECTRUM");
    test := false;
    spec.indic(localName,spoint);
    spec := Spoint;
procedure Menu(lineNumber : integer; title : string; quit : boolean; var M : Menutable; var answer : integer; var i, j : integer; var error : boolean); 
begin
  var firstLet : integer; 
  quit : boolean;
  first, ch, ch2 : char;

  begin
  background := blue;
  TextCol := White;
  Ascol := White;
  FirstLet := yellow;
  i1 := 0:12 := 0;
  error := false;
  quit := false;
  for i := 1 to lineNumber do
    M[i].Title := UpStyle(M[i].Title);
    repeat
      Writeln(Title, ':');
      ( WRITE THE 7 FIRST ITEMS )
      n := min(7, lineNumber);
      for i := 1 to n do
        begin
          gtxtxy(5, 4+i1); 
          first := chr(64+i); 
          Textcolor(FirstLet); 
          write(first, ' '); 
          ch := write(M[i].Title); 
          if M[i].Type1 = '0' then
            begin
              gtxtxy(20, 6+i2);write(': ');
              if M[i].Type2 = 1 then
                begin
                  if M[i].bool then writeln('On') else writeln('Off');
                  end
                else begin
                  if M[i].Type3 = 2 then writeln(M[i].color:5:2)
                    else writeln(M[i].str);
                  end;
                end;
            end;
            ( WRITE THE OTHER ITEMS )
            for i := (LineNumber-7) to (LineNumber-1) do
              begin
                gtxtxy(5, 4+i1); 
                first := chr(64+i); 
                Textcolor(FirstLet); 
                write(first, ' '); 
                ch := write(M[i].Title); 
                if M[i].Type1 = '0' then
                  begin
                    gtxtxy(20, 6+i2);write(': ');
                    if M[i].Type2 = 1 then
                      begin
                        if M[i].bool then writeln('On') else writeln('Off');
                        end
                      else begin
                        if M[i].Type3 = 2 then writeln(M[i].color:5:2)
                          else writeln(M[i].str);
                        end;
                      end;
                end;
              end;
        end;
      end;
    end;
  end.
USER INPUT

```
goto(y(5,25));
write('Hit the letter of your choice : '); read(kbd.ch);
if ord(ch) = 77 then begin
  error := true;
i1 := 27;
if not (keyPressed) then i2 := 0
else begin
  read(kbd.ch);
i2 := ord(ch);
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;
goto(y(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].TypeL := not M[i].TypeL;
goto(y(29,40#1,6*2(i-1))).write('';goto(y(29,40#1,6*2(i-1))).write('');
    if M[i].bo = then write(On else write( Off)
; end
  end
  else begin
    goto(y(29,40#1,6*2(i-1))).write('');
    if M[i].TypeL = 2 then read(M[i].re);
    if M[i].TypeL = 3 then read(M[i].str);
  end
end;
```

...
procedure AxisDef(var MainCh : char; var axis : A.Def);
{
    THIS PROCEDURE ALLOWS THE USER TO CHANGE
    THE AXIS USED IN THE PLOTTING PROCEDURE
}
var answer,kl,k2 : integer;
    tab : menu.tab;
    title : name;
    error : boolean;
    Ch : char;
beg
MainCh := Chr(27);
title := 'PARAMETERS OF THE AXIS :';
tab[1].title := 'minimum of x';
tab[1].type := 2;
tab[1].re := axis.min;
tab[2].title := 'maximum of x';
tab[2].type := 2;
tab[2].re := axis.max;
tab[3].title := 'x-axis step';
tab[3].type := 2;
tab[3].re := axis.delta;
tab[4].title := 'x-axis label';
tab[4].type := 2;
tab[4].str := axis.labelx;
tab[5].title := 'Title';
tab[5].type := 5;
tab[5].str := axis.title;
tab[6].title := 'y-axis label';
tab[6].type := 2;
tab[6].str := axis.labely;
tab[7].title := 'y-axis step';
tab[7].type := 2;
tab[7].re := axis.deltaY;
tab[8].title := 'minimum of y';
tab[8].type := 2;
tab[8].re := axis.ymin;
tab[9].title := 'maximum of y';
tab[9].type := 2;
tab[9].re := axisymax;
menul5.tab,title,teile,tab,answer,kl,k2,error:
Axis.xmin :=tab[1].re;
Axis.xmax :=tab[2].re;
Axis.delta :=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 kl = 27 then:
    Ch := Chr(128)
else if kl = 2 then:
    Ch := Chr(101)
end;
procedure plotpoint(var point : Point);  
{ THIS PROCEDURE PLOTTES THE SPECTRUM AT WHICH IPONINT POINTS 
 AND EVENTUALLY THE LAGER INTENSITY }  
var i,j,x1,x2,y1,y2,z1,z2 : integer;  
xScale,xShift,yScale,yShift,PowerScale : real;  
point1,point2 : Point;  
begin  
  xScale := point$.Graph.XScale;  
  xShift := point$.Graph.XShift;  
  yScale := point$.Graph.YScale;  
  yShift := point$.Graph.YShift;  
  PowerScale := point$.Graph.PowerScale;  
  point1 := point$.measure1;  
  point2 := point$.measure2;  
  if (not point$.Graph.Power) or (point$.measure2 = nil) then  
    begin  
      j := 1;  
      x1 := trunc((i point$.Xmin*xShift)*xScale*Xorig)*xScreenScale;  
      y1 := trunc((i point$.data[j]*yShift)*yScale*Yorig)*yScreenScale;  
      i := 1;  
      repeat  
        i := i + 1;  
        j := j + 1;  
        x := point$.xMin + point$.increment*(i-1);  
        y := trunc((i point$.data[j]*yShift)*yScale*Yorig)*yScreenScale;  
        if (x < Xorig) then draw((i,1,y1,x2,y2,1));  
        y2 := y1;  
        if j = ExitRegion then begin  
          y1 := 0;  
          point1 := point$.data[j];  
          end;  
        until (j = point$.MeasNumber) or (j = 600);  
    end;  
  else begin  
    j := 1;  
    point1 := nil then write('point1 = nil');  
    if point1 = nil then write('point2 = nil');  
    x1 := trunc((i point$.Xmin*xShift)*xScale*Xorig)*xScreenScale;  
    y1 := trunc((i point$.data[j]*yShift)*yScale*Yorig)*yScreenScale;  
    i := 1;  
    repeat  
      i := i + 1;  
      j := j + 1;  
      x2 := trunc((i point$.increment*(i-1));  
      y2 := trunc((i point$.data[j]*yShift)*yScale*Yorig)*yScreenScale;  
      if (x < Xorig) then begin  
        draw(x1,y1,x2,y2,1);  
        draw(x1,y1,x2,y2,1);  
        end;  
    until (j = point$.MeasNumber) or (j = 600);  
  end;  
end;
procedure SpecPlot(var MainCh : char);
{
   THIS PROCEDURE PLOTS THE SPECTRA
   DEFINED BY CURVENUM
   }
const PlotColor = LightGreen;

var j, i : integer;
   d : pointer;
   ch, quot,ch,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;
       end;
   repeat
       ex := true;
       read(ddi,Chl);
       if number(ch) then begin
           j := curvenum[value(ch)];
           if i <> nil then
               begin
                   i'.Graph.OnOff := not i'.Graph.OnOff;
                   if i'.graph.OnOff then
                       ex := false;
                   Plotcurve(i);
               end;
       end;
   until(ex);
   if letter(ch) then begin
       for ; := 1 to 4 do
           begin
               if curvenum[] nil then curvenum[]]; graph.ch := ch;
           end;
   end;
   if ord.ch = 27 then
       begin
           if true then
               begin
                   end();
                   MainCh := ch;
           end;
   end:
   until(Quit);
end:
procedure SpecLister(MainCh : char);

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

var i,j,flag : integer;
ch,chl,ch2 : char;
point : pointer;
exit : boolean;
localname : name;
Str : longstring;
text : string;

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

procedure ListSave(point : pointer;Str : Bigstringi;var j : integer);

SAVE THE SPECTRA SELECTED BY THE USER

begin
  j := j + 1;
  if point^.lin[i] nil then ListSave(point^.lin[i].str,j);
  j := j + 1;
  if Str[j] = ' ' then specsave(point^.localname);
end;

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

procedure ListDel(point : pointer;Str : Bigstringi;var j : integer);

DELETE THE SPECTRA SELECTED BY THE USER

begin
  j := j + 1;
  if point^.lin[i] nil then ListDel(point^.lin[i].str,j);
  j := j + 1;
  if Str[j] = ' ' then specdel(point^.localname);
end;

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

procedure ListDetail(point : pointer;Str : Bigstringi;var j : integer);

WRITE THE .TIEX FILE ASSOCIATED WITH SPECTRA SELECTED
BY THE USER IF IT EXISTS

begin
end;

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

procedure procChoice(i : integer; bln : flag ; boolean);

USED FOR THE GRAPHIC

begin
  i := bln ? 1 : 0;
  begin
    text:='background=1';
    text:='background=2';
  end;
  else begin
    text:= 'background=1';
    text:= 'background=2';
  end;
end;

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

BEGINNING OF SPECLIST CODE
BEGINNING OF SCRISIZ CODE

begin
    terminal := white;
    background := blue;
    MainCh := chr(120);
    Textmode(3);
    textbackground(background);
    Color:
        textcolor(yellow);
        gotoxy(30,5); write('LISTING OF THE SPECTRA');
        gotoxy(27,7); write('LOCAL NAME');
        gotoxy(15,7); write('MEAS. NAME');
        gotoxy(27,7); write('TYPE');
        gotoxy(40,7); write('TITLE');
        j := 5;
        Point := hword;
        while point .- nil do
            begin
                i := i+1;
                gotoxy(5..4); write('point, localname');
                gotoxy(10,5); write('point, measnumber');
                gotoxy(15,5); write('point, datatype');
                gotoxy(20,5); write('Graph, title');
                point := point'.link';
                end;
                procnode(1, false);
                procnode(2, false);
                procnode(3, false);
                procnode(4, true);
                j := 1;
                exit := false;
                while (Ord(ch) = 27) do
                    begin
                        if not keypressed then exit := true
                        else begin
                            read(kbd, ch);
                            procnode(j, false);
                            if Ord(ch) = 75 then j := j-1;
                            if Ord(ch) = 77 then j := j+1;
                        end;
                    end;
                    if ord(ch) = 13 then exit := true;
                    end;
                    if j = 5 then j := 1;
                    if j > 0 then j := 4;
                    procnode(j, true);
                    until (exit);
        textbackground(background);
        if j > 4 then
            begin
                flag := 1;
                if flag = 1 then write('save');
                if flag = 2 then write('delete');
                if 'flag' = 3 then write('detail');
                chl := chr(24); ch2 := chr(25);
                str :=
                gotoxy(5,24);
                write('Move The Cursor With ', chl,' and ', ch2,'. Hit Return To Continue');
            end;
            j := 0;
            if ('exit' then write;
            gotoxy(4, j);
            read(*bd, ch);
            if (Ord(ch) = 27) then
                begin
                    if not keypressed then exit := true
                    else begin
                        read(kbd, ch);
                        if Ord(ch) = 75 then j := j-1;
                        if Ord(ch) = 77 then j := j+1;
                    end;
                    end;
                    if ord(ch) = 13 then exit := true
                    begin
                        output(4, j);
                        if str(j) = 1 then
                            begin
                                write(1);   str(j) := '1';
                            end;
else begin
  if ord(ch)=13 then
    begin
      gotoy(4,j);
      if str[j]="#" then
        begin
          write("#");
          str[j] := ";
        end;
      end;
      gotoy(#);
      if j = 9 then j := 10
      if j =10 then j:=9;
      gotoy(4,j);
    until [exit];
    j := 9;
    if flag = 1 then Listsave(head,str,j);
    if flag = 0 then ListDel(head,str,j);
    if flag = 2 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 FLOATING OF A SPECTRUM )

    var
    answer, n1, n2: integer;
    indel: pointer;
    tab: menu.tab;
    Ch: char;
    Title: name;
    Error: boolean;

    begin
    n1 := 123;
    n2 := Ch(17);
    indel := 123.Ch;
    begin
      indel := urvelic1[curve];
      if indel nil then
      begin
        Title := "Name of the float";
        tab(1).title := name;
        tab(1).value := "
        tab(1).text := indel;" of name;
      end;
    end;

```
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 . Xscale;

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;

enu[9].title := true . tab . answer . 11, 12, 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[1curve]);
end;

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

procedure convert(spec1, spec2 : Spectrum);

  THIS PROCEDURE CHECKS IF THE TWO SPECTRA
  ARE IN 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 : Point;
spec1, spec2, spec1, spec2 : Spectrum;
<spec1>, sum1, sum2 : real;
imin, imax, i, i, j, n1 : integer;

begin
  if spec1.increment = spec2.increment then
    begin
      ( First Case : SAME INCREMENT )
      if spec1.min = spec2.min then
        begin
          spec := spec2;
          spec := spec1;
          spec := spec1;
          end;
          spec := spec2;
          spec := spec2.measure1;
          spec := spec2.measure2;
          spec := spec.min;
          j := 0;
          i := 0;
          i := spec.min;
          n1 := spec.max;
          r-1 := spec.increment;
          point01 := spec.measure1;
          point02 := spec.measure2;
          point21 := spec.measure1;
          point22 := spec.measure2;
          repeat
            j := j + 1;
            i := i + 1 * spec1.increment;
            if i = spec.min then
              begin
                i := i + 1;
                i := i + 1;
                if i = max then
                  begin
                    point21.data[i] := point01.data[i];
                    point22.data[i] := point02.data[i];
                    end;
                else begin
                  point21.data[i] := 0;
                  point22.data[i] := 0;
                end;
                if j = DataArraySize then
                  begin
                    k := 0;
                    new(point21.data);
                    point21 := point21.data1;
                    point21.datalink := nil;
                    new(point22.data1);
                    point22 := point22.data1;
                    point22.data1 := nil;
                    end;
                  end;
                end;
            else begin
              point01 := point01.data1;
              point02 := point02.data1;
            end;
          until i = spec1.measure.number;
        end;
    else begin
      ( Second Case : DIFFERENT INCREMENT )
      if spec1.increment = spec2.increment then
        begin
          spec := spec1;
          spec := spec2;
          spec := spec1;
          end;
        end;
      ( INITIALIZATION OF SPEC2 )
    end;
(INITIALIZATION OF SPEC2)

new(spec2.measure1);
new(spec2.measure2);
inc := spec1.increment;
min := spec1.min;
max := spec1.min + (spec1.MeasNumber - 1) * inc;
j := 0;
point21 := spec2.measure1.point21 := 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.data);
        new(point22.data);
        point21 := point21.data;
        point22 := point22.data;
      end;
    end;
  end;
spec2.min := spec1.min;
spec2.increment := inc;
spec2.localname := '2';

(COMPUTATION OF SPEC2)

i := 0;
x1 := spec1.min; x2 := spec1.min + (i-1) * inc;
point01 := spec1.measure1.point01 := spec1.measure2;
point11 := spec2.measure1.point11 := spec2.measure2;
while (i <= DataArraySize) do
  begin
    k := i + 1;
x2 := x2 + (i-1) * inc;
    end;
  for i := 1 to spec1.MeasNumber do
    begin
      j := j + 1;
x1 := spec1.min + (i - 1) * inc;
      if j = DataArraySize then
        begin
          if (x1 = x2 + (i-1) * inc) then
            begin
              sum1 := sum1 + point01.data[j];
              sum2 := sum2 + point02.data[j];
              n1 := n1 + 1;
            end;
        end;
      else begin
        if n1 = 0 then
          begin
            point01.data[j] := sum1/n1;
            point02.data[j] := sum2/n1;
            sum1 := point01.data[j];
            sum2 := point02.data[j];
            n1 := 1;
            x2 := x2 + inc;
            i := k + 1;
          end;
        if j = DataArraySize then
          begin
            point21 := point21.data;
            point22 := point22.data;
          end;
        end;
      end;
    end;
end;
procedure TestInput(Numpoints : integer;
  var NumberOfBits : byte;
  var Error : byte);

{ - Input: Numpoints
  - Output: 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).}

begin

{ -begin-}
  Error := 2; { Assume Numpoints not a power of two }
  if Numpoints = 2 then
    begin
      Error := 1; { Numpoints < 2 }
      Term := 1;
      while (Term <= 12) 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;
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 them and store them. }

begin

{ -begin-}
  RealFactor, ImagFactor : real;
  Term : integer;
  TermMinus1 : integer;
  UpperLimit : integer;

  begin
    RealFactor := Cos(2 * pi / Numpoints);
    ImagFactor := -Sin(1 - Sqr(RealFactor));
    CosTable[0] := 1;
    SinTable[0] := 0;
    CosTable[1] := RealFactor;
    SinTable[1] := ImagFactor;
    UpperLimit := Numpoints shr 1 - 1;
    for Term := 0 to UpperLimit do
      begin
        TermMinus1 := Term + 1;
        SinTable[Term] := CosTable[TermMinus1] * RealFactor -
                         SinTable[TermMinus1] * ImagFactor;
        CosTable[Term] := CosTable[TermMinus1] * RealFactor +
                         SinTable[TermMinus1] * ImagFactor;
      end;
end;
end; { procedure MakeSinCosTable }
procedure FFTInvertBits : byte;
  var Term : integer;
  Invert : boolean;
  var xReal : TvectorPtr;
  var xImag : TvectorPtr;
  var xPairable : TvectorPtr;
  var xCostable : TvectorPtr;

  (- Input: NumberOfBits, NumPoints, Inversion, xReal, xImag, -)
  (- xPairable, xCostable -)
  (- 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 = 0.707106781186547;

var
  Term : byte;

var
  NumberOfBits : integer;
  NumPoints : integer;
  xReal : TvectorPtr;
  xImag : TvectorPtr;

procedure FFTInvert(NumberofBits : byte;
  var NumPoints : integer;
  var xReal : TvectorPtr;
  var xImag : TvectorPtr;

  (- 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; this 2 indices -)
  (- will be switched. For example, if there are 16 points, -)
  (- Index 7 (binary 111) 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;

var
  Invert : boolean;
  NumPointsDiv2, F : 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;
    end;
\begin{verbatim}
1 := NumPoints div 2;
while k := Invert do
begin
   Invert := Invert + k;
   k := k shr 1;
end;
Invert := Invert + k;
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 := CellSeparation;
   \{ CellSeparation doubles; equals 2^Term \}
   NumElementsInCell := NumElementsInCell + 1;
   NumElementsInCellLH := NumElementsInCell shr 1;
   NumElementsInCellRH := NumElementsInCellLH shr 1;

   \{ Special case: RootOfUnity = EXP(-i 0) \}
   Element := 0;
   while Element := NumPoints do
      \{ 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;
      Element := Element + CellSeparation;
end;
\end{verbatim}
for CellElements := 1 to NumELinCellSH # 1 do
begin
  Index := CellElements * NumberOfCells;
  Costerm := Costable[Index];
  Sumterm := LinTerm + SinTable[Index];
  Difterm := SinTable[Index] - Costerm;
  Element := CellElements;
  while Element < NumPoints do
begin
  Compute the X[Element] with the element in 
  the identical location in the next cell 
  ElementInNextCell := Element + NumElementsInCell;
  XReal'[ElementInNextCell] := XReal[ElementInNextCell] + XImag[ElementInNextCell] # ConTerm;
  Dummy1 := XReal[ElementInNextCell] # Difterm;
  RealDummy := Dummy1 - XImag[ElementInNextCell] # SumTerm;
  ImagDummy := Dummy1 + Dummy2;
  XReal[ElementInNextCell] := XReal[Element] - RealDummy;
  XImag[ElementInNextCell] := XImag[Element] + ImagDummy;
  Element := Element + CellSeparation;
end;
end; (for)

{ Special case: RootOfUnity = Exp(-1 Pi/4) }
if Term = 2 then begin
  Element := NumELinCellSH;
  while Element < NumPoints do begin
    Compute the X[Element] with the element in 
    the identical location in the next cell 
    ElementInNextCell := Element + NumElementsInCell;
    RealDummy := RootTwoOverTwo * (XReal[ElementInNextCell] + 
    Imag[ElementInNextCell] - XReal[ElementInNextCell] - 
    Imag[ElementInNextCell]);
    XReal[ElementInNextCell] := XReal[Element] + RealDummy;
    XImag[ElementInNextCell] := XImag[Element] + ImagDummy;
    Element := Element + CellSeparation;
  end;
end;

for CellElements := NumELinCellSH # 1 to NumELinCellSH # 1 do
begin
  Index := (CellElements * NumberOfCells);
  Costerm := Costable[Index];
  Sumterm := SinTable[Index] + Costerm;
  Difterm := SinTable[Index] - Costerm;
  Element := CellElements;
  while Element < NumPoints do begin
    Compute the X[Element] with the element in 
    the identical location in the next cell 
    ElementInNextCell := Element + NumElementsInCell;
    Dummy1 := XReal[ElementInNextCell] # Difterm;
    RealDummy := Dummy1 - XImag[ElementInNextCell] # SumTerm;
    ImagDummy := Dummy1 + Dummy2;
    XReal[ElementInNextCell] := XReal[Element] + RealDummy;
    XImag[ElementInNextCell] := XImag[Element] + ImagDummy;
    Element := Element + CellSeparation;
  end;
end; (for)
( Special case: RootOfUnity = EXP(-i Pi/2) )

if SomeCondition then
begin
  \text{Element} := \text{NumElInCellSHK1};
  \text{while} \ Element \ := \text{NumPoints} \ \text{do begin}
  \text{begin}
    \text{Combine the } X[\text{Element}] \text{ with the element in }
    \text{the identical location in the next cell }
  \end
  \text{ElementInNextCell} := \text{Element} + \text{NumElementsInCell};
  \text{RealDummy} := \text{XImag}[\text{ElementInNextCell}];
  \text{ImagDummy} := -\text{XReal}[\text{ElementInNextCell}];
  \text{XReal}[\text{ElementInNextCell}] := \text{XReal}[\text{Element}] - \text{RealDummy};
  \text{XImag}[\text{ElementInNextCell}] := \text{XImag}[\text{Element}] - \text{ImagDummy};
  \text{XReal}[\text{Element}] := \text{XReal}[\text{Element}] + \text{RealDummy};
  \text{XImag}[\text{Element}] := \text{XImag}[\text{Element}] + \text{ImagDummy};
  \text{Element} := \text{Element} + \text{CellSeparation};
end;
end;

\text{for} \ \text{CellElements} := \text{NumElInCellSHK1} + 1 \ \text{to} \ \text{NumElInCellSHK2} - 1 \ \text{do begin}
\text{Index} := \text{CellElements} - \text{NumberOfCells};
\text{CosTerm} := \text{CosTable}[\text{Index}];
\text{SinTerm} := \text{SinTable}[\text{Index}] * \text{CosTerm};
\text{DiffTerm} := \text{DiffTable}[\text{Index}] - \text{CosTerm};
\text{Element} := \text{CellElements};
\text{while} \ Element \ := \text{NumPoints} \ \text{do begin}
  \text{begin}
    \text{Combine the } Y[\text{Element}] \text{ with the element in }
    \text{the identical location in the next cell }
  \end
  \text{ElementInNextCell} := \text{Element} + \text{NumElementsInCell};
  \text{Dummy1} := \text{XReal}[\text{ElementInNextCell}] * \text{CosTerm};
  \text{Dummy2} := \text{XReal}[\text{ElementInNextCell}] * \text{DiffTerm};
  \text{RealDummy} := \text{Dummy1} - \text{XImag}[\text{ElementInNextCell}] * \text{SumTerm};
  \text{ImagDummy} := \text{Dummy1} - \text{Dummy2};
  \text{XReal}[\text{ElementInNextCell}] := \text{XReal}[\text{Element}] - \text{RealDummy};
  \text{XImag}[\text{ElementInNextCell}] := \text{XImag}[\text{Element}] - \text{ImagDummy};
  \text{XReal}[\text{Element}] := \text{XReal}[\text{Element}] + \text{RealDummy};
  \text{XImag}[\text{Element}] := \text{XImag}[\text{Element}] + \text{ImagDummy};
  \text{Element} := \text{Element} + \text{CellSeparation};
end; \ \text{for} \)
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;
end; { procedure FFT }
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: 25 January 1987

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

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

begin

{ Input: NumPoints, XReal };
{ Output: XReal, XImag };

{ This procedure shuffles the real data. There are 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. }

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

begin
New(DummyReal);
New(DummyImag);
for Index := 0 to NumPoints - 1 do
begin
NewIndex := Index shr 1;
DummyReal[Index] := XReal[NewIndex];
DummyImag[Index] := XReal[NewIndex + 1];
end;
XReal := DummyReal;
XImag := DummyImag;
Dispose(DummyReal);
Dispose(DummyImag);
end; { Procedure MakeRealDataComplex }
procedure UnshuffleComplex(Input NumPoints : integer;
      var SinTable : TVectorFtr;
      var CosTable : TVectorFtr;
      var XReal : TVectorFtr;
      var XImg : TVectorFtr);

{-----------------------------------------------------------------------}
{ Input: NumPoints, SinTable, CosTable, XReal, XImg   }
{ Output: XReal, XImg   }
{-----------------------------------------------------------------------}
{ This procedure unshuffles the complex transform.  }
{ The transform has NumPoints elements. This procedure unshuffles the transform so that it is 2xNumPoints 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
  F1OverNumPoints : real;
  Index : integer;
  IndexSHR : integer;
  NumPointMinusIndex : integer;
  SymmetricIndex : integer;
  Multiplier : real;
  CosFactor, SinFactor : real;
  RealSum, ImagSum, RealDiff, ImagDiff : real;
  RealDummy, Imag Dummy : TVectorFtr;
  NumPointsSHR : integer;

begin

  New(RealDummy);
  New(ImagDummy);
  RealDummy^ := Xreal;
  ImagDummy^ := Ximag;
  F1OverNumPoints := 1.0 / NumPoints;
  NumPointsSHR := NumPoints SHR 1;
  RealDummy[Index] := (Real[real] + Imag[imag]) / Sqrt(2);
  ImagDummy[Index] := (Real[real] - Imag[imag]) / Sqrt(2);
  for Index := 1 to NumPoints - 1 do
    begin
      Multiplier := 0.5 / Sqrt(2);
      Factor := F1OverNumPoints * Index;
      NumPointMinusIndex := NumPoints - Index;
      SymmetricIndex := NumPointsSHR - Index;
      if Odd(Index) then
        begin
          CosFactor := CosTable[IndexSHR];
          SinFactor := -SinTable[IndexSHR];
        end
      else
        begin
          IndexSHR := Index SHR 1;
          CosFactor := CosTable[IndexSHR];
          SinFactor := SinTable[IndexSHR];
        end;
      RealSum := Xreal[Index] + Real[NumPointMinusIndex];
      ImagSum := Ximag[Index] + Imag[NumPointMinusIndex];
      RealDiff := Xreal[Index] - Real[NumPointMinusIndex];
      ImagDiff := Ximag[Index] - Imag[NumPointMinusIndex];
      RealDummy[Index] := Multiplier * (RealSum * CosFactor + ImagSum * SinFactor);
      ImagDummy[Index] := Multiplier * (RealDiff * SinFactor + ImagDiff * CosFactor);
      RealDummy[SymmetricIndex] := RealDummy[Index];
      ImagDummy[SymmetricIndex] := -ImagDummy[Index];
    end;
  for Index := 1 to NumPoints - 1 do
    begin
      Real := RealDummy;
      Imag := ImagDummy;
      Dispose(RealDummy);
      Dispose(ImagDummy);
    end;
  procedure UnshuffleComplex Output ;
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 shr 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;
{ procedure RealFFT }

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

{- Turbo Pascal 7.0 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'
  - TRUE = 'Inverse'
  - XReal, XImag : TNvectorPtr Data points
  - Error : byte Indicates an error
  - Errors: 0: No Errors
  - 1: NumPoints <= 0
  - 2: NumPoints not a power of two
-}
{- Version Date: 26 January 1987
-}

var
  SinTable, CosTable : TNvectorPtr;
  NumberOfBits : byte;
begin (procedure ComplexFFT )
  TestInput(NumPoints, NumberOfBits, Error);
  if Error = 0 then
    begin
      New(SinTable);
      New(CosTable);
      MakeSinCosTable(NumPoints, SinTable, CosTable);
      FFT(NumberOfBits, SinTable, CosTable);
      Dispose(SinTable);
      Dispose(CosTable);
    end;
end;  // procedure ComplexFFT )
procedure shift (RealPtr : UPtr; Istart, Iend : integer; Var NumPoints : integer; Xreal, Ximag : InVectorPtr; Xr, Xi : real);
{
  PROCESSES THE SPECTRUM IN ORDER TO TRANSFORM IT
}

const
  averagenum = 5;

var
  i,j,ist : integer;
  error : byte;
  Spec : Spectrum;
  P1r, P1i, P2r, P2i : DPoint;
  Sum1, Sum2 : real;
begin

  COMPUTE THE NUMBER OF POINTS OF THE FFT
  i := trunc((Iend-Istart)/ln(2) + 1);
  NumPoints := round(exp(i*ln(2)));

  INITIALIZATION

  For i := 1 to NumPoints do
    xreal'[i-1] := 0;
    ist := Istart;
    P1r := RealPtr;
    while(ist < DataArraySize) do
      ist := ist + DataArraySize;
      P1r := P1r'.data[ist];
    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] := P1r'.data[ist];
        if ist = DataArraySize then
          begin
            ist := 0;
            P1r := P1r'.data[ist + 1];
          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;
    X0 := sum1/averagenum;
    Y0 := sum2/averagenum;
    for i := 0 to Iend-Istart-1 do
      begin
        real[i] := xreal'[i] - X0 - Y0*exp(i*ln(2)*((Iend-Istart-1)/2));
      end;
  goto(x, y): writeln('Number Of Points : ', NumPoints);
  goto(x, y): writeln('X0 : ', X0, ', Y0 : ', Y0);

  CALL OF THE FFT

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

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

var ch : char;
spec  : spectrum;
quit,test,error,reset : boolean;
answer,save,filename,defaultname,Name : string;
Count,low,x1 : integer;
y : real;
infile : text;
str   : bigstring;

procedure Make(filename : Name);
{
    THIS PROCEDURE READS A FILE AND MAKE A SPECTRUM
}

var inputFile
    TextFileName             : text;
    Title                   : string;
    y,x,l,inc,increment,sum1,sum2   : real;
    l,err,Error,xi,yi,Name : integer;
    ch : char;
L,store,Point1,Point2  : Point:

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

var tttfile : text;
str    : bigstring;

begin
  Assign(tttfile,TextFileName);
  Rewrite(tttfile);
  writeln('Write all the Information Available About ',Filename);
  writeln('Hit RETURN twice when you have finished');
  repeat
    readln(str);
    writeln(str);
  until length(str)=0;
  close(tttfile);
end;

procedure Make;
begin
{ CREATE A TEXT FILE IF IT DIDN'T EXIST }
  if (filename,1 амер (filename,1,trim(filename,1)) = nil)
  then begin
    writeln('Writing the Information Available About ',Filename);
    Assign(inputFile,TextFileName);
    Readln(inputFile,y,x,l,inc);
    Readln(inputFile,Name,Error);
    Readln(inputFile,Graph,title);
    Readln(inputFile,start);
    writeln(y,x,inc);
    writeln('Write all the Information Available About ',Filename);
  end;
  writeln('Hit RETURN twice when you have finished');
  repeat
    readln(str);
    writeln(str);
  until length(str)=0;
  close(inputFile);
end;

begin
{
    Definition of the record characteristics
}
; Definition of the record characteristics ;

Spec.filename := Filename;
Spec.MeanNumber in U;
If U > MaxMeanNumber then
Begin
   writeln('Unable To Process These Data');
   gotoxy(5,22);
   writeln('Only MaxMeanNumber Measurements considered');
   Spec.MeanNumber := MaxMeanNumber;
End;
increment := 1/4;
Spec.Increment := increment;
[ SPECTRUM DEFAULT CHARACTERISTICS ]
Spec.graph.xshift := 0;
Spec.graph.xscale := 1;
Spec.graph.yshift := 0;
Spec.graphyscale := 1;
Spec.graph.OnOff := true;
Spec.Graph.Rower := false;
Spec.Graph.FewScale := 1;
Spec.Graph.Xmin := 2;

( Read the spectrum )

(*1*)
readln(inputFile, x1, x2, x3);
If loresult <> 0 then writeln('Error Code = ', l0result);
(*2*)
Spec.xmin := 'x1';
If Default_Type then c2 := x2 / x3;
Spec.Measure1.Data[i] := c2;
Spec.Measure2.Data[i] := x3;
(*3*)
if i <= 1 then begin Spec.Measure1.point2 := Spec.Measure2;
if Default_Type then begin
while (i < Spec.MeasureNumber) do begin
   i := i + 1;
end
end;
(*4*)
readln(inputFile, x1, x2, x3);
If loresult <> 0 then writeln('Error Code = ', l0result);
(*4*)
If \((x1 - \text{SpecMin}) \text{ increment}\) then begin
  \textbf{Point1}.\textbf{Data}[j] := x2;
  \textbf{Point2}.\textbf{Data}[j] := x3;
end
else
begin
  \textbf{Point1}.\textbf{Data}[j] := \text{Round}(x1);
  \textcolor{LightRed}{\textbf{writeln}('Inconsistency In The Data At '}, \textbf{Point1}.\textbf{Data}[j], \textbf{Point2}.\textbf{Data}[j]
end
\textbf{end};

\{ CREATE NEW RECORDS IF NECESSARY \}
\textbf{if} \(j = \text{DataArraySize}\) then begin
  \textbf{new}([\textbf{Point1}.\textbf{Data}], \textbf{point2}.\textbf{Data});
  \textbf{new}([\textbf{Point1}.\textbf{Data}], \textbf{point2}.\textbf{Data});
  \textbf{point1} := \textbf{point1}.\textbf{Data} link;
  \textbf{point2} := \textbf{point2}.\textbf{Data} link;
  \textbf{point1}.\textbf{Data} link := \textbf{nil};
  \textbf{point2}.\textbf{Data} link := \textbf{nil};
  \textbf{Point1}.\textbf{DataType} := \text{Continue};
  \textbf{Point2}.\textbf{DataType} := \text{Continue};
  \textbf{j} := 0;
\textbf{end};
\textbf{end};
\textbf{end};

\{ CREATE NEW RECORDS IF NECESSARY \}
\textbf{if} \(j = \text{DataArraySize}\) then begin
  \textbf{new}([\textbf{Point1}.\textbf{Data}], \textbf{point2}.\textbf{Data});
  \textbf{new}([\textbf{Point1}.\textbf{Data}], \textbf{point2}.\textbf{Data});
  \textbf{Point1}.\textbf{Data} := \textbf{point1}.\textbf{Data} link;
  \textbf{Point2}.\textbf{Data} := \textbf{point2}.\textbf{Data} link;
  \textbf{Point1}.\textbf{DataType} := \text{Continue};
  \textbf{Point2}.\textbf{DataType} := \text{Continue};
\textbf{end};
\textbf{end};
\textbf{end};
{ procedure define }
begin
  background := blue;
  textcol := white;
  anscol := lightcyan;
  MainCh := Chr(120);
  quit := false;
  repeat
    WriteTitle('load a spectrum',27);
    gototxy(5,10);textwrite('Current Directory : ');
    answrite(Current_dir);
    gototxy(5,10);textwrite('Default Type : ');
    gototxy(26,10);ans:
    if Default_Type then write('With Laser Power')
    else write('Without Laser Power');Tex:
    test := false[count=1];
    gototxy(5,25);write('Alt D to change the directory,Alt T to change the type');
    repeat
      gototxy(5,12);write('DOS-Filename : ');
      Gread(26,12,errmsg);Tex:
      if Answe Chr then begin
        if Ganswer.chr = Chr(22) then Change_dir
        else begin
          if Ganswer.chr = Chr(20) then begin
            Default_Type := not Default_Type;Tex:
            gototxy(5,10);write('Default Type : ');
          end;
          gototxy(26,10);disp:
          if Default_Type then write('With Laser Power')
          else write('Without Laser Power');Tex:
          end;
        end;
      end;
      else begin
        MainCh := chr(27);
        Quit := true;
      end;
    end;
    else begin
      ans := Ganswer.str;
      if (CountChr(ans,\'\') = 0) and (length(Current_dir) > 0) then begin
        ans := Current_dir + \'\' + ans;
      end;
      Name := ans + '.dat';
      if exist(Name) then begin
        Namefile := Name;
        Test := true;
      end;
      else begin
        if exist(ans) then begin
          Namefile := ans;
          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);
readin(infile.u,v);readin(infile.str);
close(infile);
gotoxy(4,14);text('Title

');ans:=write(str);Text;
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;
Mspec(NameFile,spec);
Specre(spc);
curseffect(spc);
repeat
    gotoxy(1,25);text('Do You Want To Load An Other Spectrum (Y/N) ?
');read(bd,ch);
    until((ord(ch)=27)or(ord(Uppercase(ch))=78)or(ord(Uppercase(ch))=99));
if ord(Uppercase(ch)) = 78 then quit := true;
if ord(ch)=27 then
begin
    quit := true;
    MainCh:= ch;
    if keypressed then read(bd,MainCh);
end;
end;
until(quit);
end:

{---------------------------------------------------------------}
procedure Multiply(spectrum1:spectrum; spec2:spectrum; out1:spectrum)
{
    MULTIPLIES ALL THE DATA OF A SPECTRUM IN COEI
    AND RETURNS THE RESULT IN SPECOUT
}

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

begin
    specout := spec1;
    new(specout.measure1);
    new(specout.measure2);
    specout.measure1.datamink := nil;
    specout.measure2.datamink := nil;
    i := 0; i <= point1->spec.measure1->pointout1 := specout.measure1->point2 := spec.measure2->pointout2 := spec.measure2->
    repeat
        j := i+j; j++; j += 1;
        pointout1->data[j] := coeff*point1->data[j];
        pointout2->data[j] := coeff*point2->data[j];
        if i = DataArraySize then
            begin
                i := 0;
                new(pointout1->datamink);
                pointout1 := pointout1->datamink;
                point1->pointout1 := point1->datamink;
                pointout1->datamink := nil;
                new(pointout2->datamink);
                pointout2 := pointout2->datamink;
                point2->pointout2 := point2->datamink;
                pointout2->datamink := nil;
            end;
        until i = SpecMeasureNumber;
    end;

-----------------------------------------------------------------------------------------------------------------------------------

procedure Add(spectrum1:spectrum; spec2:spectrum; out1:spectrum)
{
    ADDS TWO SPECTRA AND RETURNS THE SUM IN SPECOUT
}

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

begin
    convert(spectrum1,spec2);
    specout := spec1;
    specout->measure1 := min(spectrum1->measure1,spec2->measure1);
    if (spectrum1->measure2 increment = spec2->measure2 increment) or (spectrum1->measure2 min = spec2->measure2 min)
        then
            writeln('Error in the addition procedure!');
    new(specout.measure1);
    new(specout.measure2);
    specout.measure1.datamink := nil;
    specout.measure2.datamink := nil;
    i := 0; i <= point1 := spec1.measure1->pointout1 := spec1.measure1->point2 := spec1.measure2->
    repeat
        j := i+j; j++; j += 1;
        pointout1->data[j] := point1->data[j] + point2->data[j];
        pointout2->data[j] := point2->data[j] + point2->data[j];
        if i = DataArraySize then
            begin
                i := 0;
                new(pointout1->datamink);
                pointout1 := pointout1->datamink;
                point1->pointout1 := point1->datamink;
                pointout1->datamink := nil;
                new(pointout2->datamink);
                pointout2 := pointout2->datamink;
                point2->pointout2 := point2->datamink;
                pointout2->datamink := nil;
            end;
        until i = SpecMeasureNumber;
    end;

-----------------------------------------------------------------------------------------------------------------------------------
procedure Lincomb(var MainCh : char);
{
    MAKES LINEAR COMBINATION OF SPECTRA
}
var spec:spec;spec2:spec;point:point;cnf:real;ind:integer
quit,exit,error:char;begin
begin
    Backround := 'blue';
textcol := 'lightred';
errorcol := 'lightgreen';
exit := 'false';
repeat
    MainCh := chr(120);
    WriteTitle('linear combination',30);
    writeln;
    gotoxy(5,7);write('Name Of The Spectrum 1: ');
    Read(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 1: ');
            Read(55,7,error);
            cnf := Ganswer.net;
        multispec,specout,coefs:spec := specout;
        i := 1;
    repeat
        i := i+1;
        quit := false;
    repeat
        writeln;
        gotoxy(5,5+2*i);write('');
        gotoxy(5,5-2*i);write('Name Of The Spectrum ',i,'. ');
        Read(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);writeln('');
            end;
        until(!valid(answer) or quit);
if not quit then
    begin
    gotoxy(40,5+2*i);
    write('Coefficient : ');
    Gread(35,5+2*i, error); coef := Ganswer.re;
    specindic(answer.point);
    mult(point',spec2, coef);
    add(spec,spec2,spec0ut);
    datadel(spec.measure1);
    datadel(spec.measure2);
    datadel(spec2.measure1);
    datadel(spec2.measure2);
    spec := spec0ut;
    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 := 'lin';
    spec.graph.title := 'LINEAR COMBINATION : ';
    specarc(spec);
    curveffect(spec);
    gotoxy(5,25); write(' Do You Want To Make Another Linear Combination : ');
    read(kbd.ch);
    if ord(Uppcase(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;
    until (exit);
    end;
procedure smoothing(var main: chart);
  ALLOWS THE USER TO SMOOTH SPECTRA
  
  const plotcolor = lightgreen;

  var spec : spectrum;
  xreal, yreal, ximag, yimag : TVectorPtr;
  specPtr, specoutPtr, specPtr2 : pointer;
  byte i, j, k, l;
  int n, m, npx, npy, npuntos, npunto, nchart, npoint, npoints, npoint1, npoint2, npoints, nchart;

  procedure ftsave(xreal, yimag : TVectorPtr; type : integer);

  var speci : spectrum;
  i, j : integer;
  FPtr1, FPtr2 : Dpoint;
  answer : name;

  begin
    Speci.source := Spec.source;
    Speci.localname := `sm`*spec.localname;
    Speci.measNumber := npoints;
    Speci.nmin := nstart;
    Speci.nincrement := spec.increment;
    Speci.graph.xshift := 0;
    Speci.graph.xscale := 1;
    Speci.graph.yshift := 0;
    Speci.graph.yscale := 1;
    Speci.graph.graphOnOff := true;
    Speci.graph.GnOff := true;
    Speci.graph.Power := false;
    Speci.graph.PowerScale := 1;
    Speci.graph.nmin := nstart;
    Speci.graph.title := `Smoothed Spectrum` + spec.localname;
    if type = FourierTr then begin
      Speci.localname := `fft` + spec.localname;
      Speci.graph.title := `Fourier Transform` + spec.localname;
      end;
    repeat
      gotore('1,25': write('' ));
    gotore('3,2': write(' Local Name `. `speci.localname,''
      ));
    gotore('1,2'; answer := read(answer));
    if length(answer) = 0 then begin
      answer := speci.localname;
      write(answer);
      end;
    until (not delete(answer));
    speci.localname := answer;

  { INITIALIZATION }

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

  { CONVERSION ARRAY TO LINKED LIST }

  for i := 1 to nmeas-1 do
    begin
      i := j + 1;
      ptr1.data[i] := xreal[i];
      ptr2.data[i] := ximag[i];
      if j < data.nsize then
        begin
          j := 0;
          result.data[j] := xreal[i];
          result.data[j] := ximag[i];
          ptr1 := ptr1.data[j];
          ptr2 := ptr2.data[j];
          end;
        end;
        exit;
  speci.terminal;
  CurrentPtr := 1;
procedure plotftt (ArrayPt : TVectorPtr; Increment, Yshift : real); // THIS PROCEDURE PLOTS THE CURVES DEFINED IN SPECTRUM 
var i1, x1, x2, y1, y2 : integer;

begin
  x1 := truncate(Xshift * XOrig * XScreenScale);
  y1 := truncate(((ArrayPt[0] * Yshift - 1) * YOrig * YScreenScale));
  for i := 1 to Npoints do begin
    x2 := truncate((Increment + Yshift) * XOrig * XScreenScale);
    y2 := truncate(((ArrayPt[i] * Yshift - 1) * YOrig * YScreenScale));
    if (x2 < 40) then draw(x1, y1, x2, y2);
    x1 := x2;
    y1 := y2;
  end;
end;

(SEMICOLON : BEGINNING OF THE CODE)

begin
  Background := blue;
  textcol := lightred;
  anscol := lightgreen;
  exit := false;
  repeat
    MainCh := chr(100);
    USER'S INPUT
    Write('Let smoothing', 3);
    Write('Enter type of smoothing:
      1: box, 2: uniform, 3: gaussian
      4: error');
    Begin
      Error := 'error';
      if error = 'box' then MainCh := '1'
      else MainCh := '2';
    end;
    answer := answer_str;
    until (answer = 'error') or exit;
    if not (answer = 'error') then begin
      if answer = '1' then
        specInd := newSpecr.SpecPtr;
        spec := SpecPtr;
        gotoxy(5, 10); write('If you want to transform the data:
          1: yes, 2: no');
        if answer = '1' then
          if answer = 'y' then MasPtr := spec measure;
          gotoxy(20, 25); write('Please wait...');
          x1 := truncate((x1 - spec.min) / spec.increment);
          y1 := truncate((y1 - spec.min) / spec.increment);
          DEFINE THE AXIS FOR PLOTTING;
        end;
      end;
      end;
end;
DEFINE THE AXIS FOR PLOTTING:

```c
#define title "FOURIER TRANSFORM"
define_Labels = "";
defines = "x: ";
define_xlabel = "";
define_xmin = 0;
define_xmax = npoints;
define_xlabel = npoints/8;
define_ymin = -1;
define_ymax = 1;
define_delta = 0.2;
repeat
  hirsi:
  hires_color (plotcolor);
  plotaxis (defines);
  plotfft (kreal, 1, 0, 0);
  plotfft (kimag, 1, 0, 1);
gotoxy (5, 25): write ('Frequency " "'); print (20, 25, error);
if ganswrtyp.esc then begin
  exit = true;
  MainCh = Ganswrtyp.chr;
  if Ord (MainCh) = 68 then begin
    a1def (MainCh, defines);
    next = ch (129);
    exit = false;
  end;
  until (exit or Ganswrtyp.int)
if Ganswrtyp.int then icut = Ganswrtyp.int;
if not exit then
  INVERSE TRANSFORM:
begin
  rew (real[newimag]);
  for i = 0 to npoints - 1 do
    begin
      vreal[i] = real[i];
      vimag[i] = imag[i];
    end;
    if (icut > 0) and (icut = npoints/2) then begin
      for i = > icut to npoints - icut - 1 do
        begin
          vreal[i] = 0;
          vimag[i] = 0;
        end;
      gotoxy (20, 25): write ('Please Wait');
      Complex FFT (npoints, true, vreal, vimag, byteer);
      for i = 0 to end - start - 1 do
        begin
          yreal[i] = yreal[i] + y + (yn - yo) * (end - start - 1);
        end;
    repeat
  hirsi:
  hires_color (plotcolor);
  plotaxis (defines);
  plotfft (vreal, spec.increment, start, 0);
  plotfft (vimag, spec.increment, start, 1);
  repeat
    gotoxy (1, 29): write ('');
  gotoxy (5, 25): write (Are you satisfied? (Y/N) " ");`
unless (ord(n)<>27) then record(n:=n+1) end;
begin
  exit := false;
  MainCh := chr(120);
  while exit = false do
    if keypressed then read(hbd,MainCh);
    if Ord(MainCh) = 68 then
      begin
        n := n+1;
        MainCh := chr(120);
      end;
      exit := true;
    end;
  until(exit or (upcase(Ch)="N") or (upcase(Ch)="Y");
end;
until (exit or (upcase(Ch)="N") or (upcase(Ch)="Y") :)
begin
  clear;
  gotoxy(5,15); write('Do You Want To Save The Fourier Transform " Y/N 
');
  read(hbd,Ch):
  if upcase(Ch)="Y" then fftsave(real,imag,FourierTr):
    dispose(real); dispose(imag);
  gotoxy(5,14); write('Do You Want To Save The Smoothed Spectrum " Y/N 
');
  read(hbd,Ch):
  if upcase(Ch)="Y" then fftsave(real,imag,SmoothFt):
    dispose(real); dispose(imag);
  gotoxy(5,13); write('Do You Want To Make Another Transform " Y/N 
');
  read(hbd,Ch):
  if ord(Ch)=27 or (ord(upcase(Ch))=76 or ord(upcase(Ch))=91) or ord(upcase(Ch)) = 'B' then exit := true;
  if ord(Ch)=77 then
    begin
      exit := true;
      MainCh := chr(120);
      if keypressed then read(hbd,MainCh);
    end;
  end;
until (exit);
end.
procedure stat(var n: integer; var answer: char); {This function calculates the mean value and variance of the two signals}

var
  answer : char;
  spec1, spec2 : Spectrum;
  mean1, variance1 : real;
  mean2, variance2 : real;
  coeff1, coeff2, cprop : real;
  samp : integer;
  specname : name;
  quit, m, error : boolean;
  ch : char;
  point : pointer;

begin
  answer := 'N';
  spec1.spec2 := Spectrum;
  mean1, variance1 := real;
  mean2, variance2 := real;
  coeff1, coeff2, cprop := real;
  samp := integer;
  specname := name;
  quit, m, error := boolean;
  ch := char;
  point := pointer;

------------------------------------------------------------------------

procedure mv(spec : Spectrum; var m, var1, var2 : real); {This procedure calculates the mean value and variance of the two signals}

var
  i, n1, n2 : integer;
  sum1, sum2, var1, var2 : real;
  point1, point2 : Pointer;

begin
  i := spec.min;
  n1 := i + sum1 / n1 i := i + sum2 / n2
  var1 := var1 + Sum(point1.datat[j]);
  var2 := var2 + Sum(point2.datat[j]);
end;

if i = DataArraySize then begin
  m1 := var1 / n1 - Sqr(m1);
  m2 := var2 / n2 - Sqr(m2);
end;

------------------------------------------------------------------------

procedure corre(spec1, spec2 : Spectrum; var dcorr: integer; var coeff1, coeff2, cprop : real); {This procedure computes the correlation coefficients in the two spectra}

var
  i, n1, n2 : integer;
  var1, var2, var1, var2 : real;
  dcorr : integer;
  coeff1, coeff2, cprop : real;
  point1, point2, point1, point2 : Pointer;

begin
  spec1.spec2 := Spectrum;
  coeff1, coeff2, cprop := real;
  point1, point2, point1, point2 := Pointer;

  for i := spec1.min do
    if i = dataarraysize then end;

  dcorr := var1 / n1 - Sqr(dcorr);
  dcorr := var2 / n2 - Sqr(dcorr);
end;
if not quit then
begin
repeat
getkey[1,256]:writer(): you want to work with another spectrum (Y/N)?
read(1,ch);
until(ord(ch) = 27 or (ord(Ucase(ch)) = 76 then quit = true;
if ord(ch) = 27 then
begin
quit := true;
MainCh := ch;
if keyp keyed then read bd, MainCh;
end;
end;
until(quit);
end;

Main Menu: SPECTRUM PROCESSING

procedure Main_Menu( MainCh: char);

var
Exit: boolean;
Array: Menutable;
Answer: 11, 12;
error: boolean;

begin
MainCh := chr(120);
Array[1].title := 'Load a spectrum from a file';
Array[1].type := 0;
Array[2].title := 'Linear combination';
Array[2].type := 0;
Array[3].title := 'Smoothing';
Array[3].type := 0;
Array[4].title := 'Statistical analysis';
Array[4].type := 0;
Array[5].title := 'List, save or delete';
Array[5].type := 0;
Array[6].title := 'Quit';
Array[6].type := 0;
Menu[16].Main MENU. false, Array, Answer, 11, 12, error);
case answer of
1: MainCh := chr(121);
2: MainCh := chr(122);
3: MainCh := chr(123);
4: MainCh := chr(124);
5: MainCh := chr(125);
6: MainCh := chr(16);
end;
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 psi;
{Psi}

const
  ns = 10;
  n = '0';
  ny = '2';

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

var
  w, p, q, u, s : array[1..ns] of vectorns;
  x, y, z, std, sdx : vectornx;
  x, y, z, std, sdx : vectornx;
  u, s, p, q, v, t, e, y, x, a, b, j : integer;
  infile, infile2 : file;
  filename, filelist : string[15];
  quit : boolean;
  norm1, norm2, norm3, r, rtp : real;
  ch : char;
  count : integer;
  model : integer;
  epsilon : real;

begin

  (DATA INPUT)

  quit := false;
  h := 1;
  repeat
    clrscr;
    gotoxy(5,10):write('filename:');
    read:=filelist;
    if filelist='.' then quit := true
    else begin
      assign(infile, filelist);
      reset(infile);
      readln(infile, filename);
      writeln(filename);
      while(filename) do
        begin
          readln(infile, x, y); end;
      begin
        assign(infile2, filelist);
        reset(infile2);
        for i := 1 to n do
          begin
            read(infile2, h, i, y);
            assign(infile2, filename);
            read(infile2, h, i, y);
            for i := 1 to n do
              begin
                read(infile2, r, rtp);
              end;
            close(infile);
            writeln(y,h,i);
            close(infile2);
          end;
        until quit=true;
      end;
  until quit=true;
  h := h+1;
  writeln;
  writeln('h=', h);
begin
  for i in 1 to n do
    begin
      for j in 1 to n do
        begin
          x[i][j] := x[i][j] + 0.5;
        end;
      end;
      for j in 1 to n do
        begin
          y[j] := y[j] + 0.5;
        end;
    end;
  end;
end.

begin
  for i in 1 to n do
    begin
      x[i] := x[i] + 0.5;
    end;
  end;
end.

begin
  for j in 1 to n do
    begin
      y[j] := y[j] + 0.5;
    end;
  end;
end.

begin
  for i in 1 to n do
    begin
      x[i] := x[i] + 0.5;
    end;
  end;
end.

begin
  for j in 1 to n do
    begin
      y[j] := y[j] + 0.5;
    end;
  end;
end.

begin
  for i in 1 to n do
    begin
      x[i] := x[i] + 0.5;
    end;
  end;
end.

begin
  for j in 1 to n do
    begin
      y[j] := y[j] + 0.5;
    end;
  end;
end.

begin
  for i in 1 to n do
    begin
      x[i] := x[i] + 0.5;
    end;
  end;
end.

begin
  for j in 1 to n do
    begin
      y[j] := y[j] + 0.5;
    end;
  end;
end.

begin
  for i in 1 to n do
    begin
      x[i] := x[i] + 0.5;
    end;
  end;
end.

begin
  for j in 1 to n do
    begin
      y[j] := y[j] + 0.5;
    end;
  end;
end.

begin
  for i in 1 to n do
    begin
      x[i] := x[i] + 0.5;
    end;
  end;
end.

begin
  for j in 1 to n do
    begin
      y[j] := y[j] + 0.5;
    end;
  end;
end.

begin
  for i in 1 to n do
    begin
      x[i] := x[i] + 0.5;
    end;
  end;
end.

begin
  for j in 1 to n do
    begin
      y[j] := y[j] + 0.5;
    end;
  end;
end.

begin
  for i in 1 to n do
    begin
      x[i] := x[i] + 0.5;
    end;
  end;
end.

begin
  for j in 1 to n do
    begin
      y[j] := y[j] + 0.5;
    end;
  end;
end.

begin
  for i in 1 to n do
    begin
      x[i] := x[i] + 0.5;
    end;
  end;
end.

begin
  for j in 1 to n do
    begin
      y[j] := y[j] + 0.5;
    end;
  end;
end.

begin
  for i in 1 to n do
    begin
      x[i] := x[i] + 0.5;
    end;
  end;
end.

begin
  for j in 1 to n do
    begin
      y[j] := y[j] + 0.5;
    end;
  end;
end.

begin
  for i in 1 to n do
    begin
      x[i] := x[i] + 0.5;
    end;
  end;
end.

begin
  for j in 1 to n do
    begin
      y[j] := y[j] + 0.5;
    end;
  end;
end.

begin
  for i in 1 to n do
    begin
      x[i] := x[i] + 0.5;
    end;
  end;
end.

begin
  for j in 1 to n do
    begin
      y[j] := y[j] + 0.5;
    end;
  end;
end.

begin
  for i in 1 to n do
    begin
      x[i] := x[i] + 0.5;
    end;
  end;
end.

begin
  for j in 1 to n do
    begin
      y[j] := y[j] + 0.5;
    end;
  end;
end.

begin
  for i in 1 to n do
    begin
      x[i] := x[i] + 0.5;
    end;
  end;
end.

begin
  for j in 1 to n do
    begin
      y[j] := y[j] + 0.5;
    end;
  end;
end.

begin
  for i in 1 to n do
    begin
      x[i] := x[i] + 0.5;
    end;
  end;
end.

begin
  for j in 1 to n do
    begin
      y[j] := y[j] + 0.5;
    end;
  end;
end.

begin
  for i in 1 to n do
    begin
      x[i] := x[i] + 0.5;
    end;
  end;
end.

begin
  for j in 1 to n do
    begin
      y[j] := y[j] + 0.5;
    end;
  end;
end.

begin
  for i in 1 to n do
    begin
      x[i] := x[i] + 0.5;
    end;
  end;
end.

begin
  for j in 1 to n do
    begin
      y[j] := y[j] + 0.5;
    end;
  end;
end.

begin
  for i in 1 to n do
    begin
      x[i] := x[i] + 0.5;
    end;
  end;
end.

begin
  for j in 1 to n do
    begin
      y[j] := y[j] + 0.5;
    end;
  end;
end.

begin
  for i in 1 to n do
    begin
      x[i] := x[i] + 0.5;
    end;
  end;
end.

begin
  for j in 1 to n do
    begin
      y[j] := y[j] + 0.5;
    end;
  end;
end.

begin
  for i in 1 to n do
    begin
      x[i] := x[i] + 0.5;
    end;
  end;
end.

begin
  for j in 1 to n do
    begin
      y[j] := y[j] + 0.5;
    end;
  end;
end.

begin
  for i in 1 to n do
    begin
      x[i] := x[i] + 0.5;
    end;
  end;
end.

begin
  for j in 1 to n do
    begin
      y[j] := y[j] + 0.5;
    end;
  end;
end.
for ia := 1 to ny do
begin
q[ia]:=0;
for ja := 1 to nx do q[ia]:=q[ia]+q[ja]*g[ja]*f[ja,ia];
end;

for ia := 1 to ny do norm:=0;
norm := sqrt(norm);
for ja := 1 to nx do q[ia]:=q[ia]/norm;

for ia := 1 to ny do u[ia]:=0;
for ja := 1 to nx do u[ia]:=u[ia]+q[ja]*g[ja]*f[ja,ia];
end;
norm:=0;
for ia := 1 to ny do norm := norm + q[ia]*g[ia];
norm := sqrt(norm);
for ia := 1 to ny do u[ia]:=u[ia]/norm;

for ia := 1 to h do norm := norm + (t[ia]-told[ia])*(t[ia]-told[ia]);
norm:=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;
writeint('r=',r);
until(interactive or (count=100));
writeint(' : vecotr t found in ',count,' iterations');
for ia := 1 to h do writeint(t[ia]);
read(told,chi);

for ia := 1 to nx do
begin
f[ia]:=0;
for ja := 1 to nx do f[ia]:=g[ia]*f[ja,ja]+q[ja]*f[ja,ia];
end;
norm:=0;
for ia := 1 to nx do norm := norm + f[ia]*f[ia];
for ia := 1 to nx do f[ia]:=f[ia]/norm;

for ia := 1 to nx do norm := norm + f[ia]*f[ia];
norm := sqrt(norm);
for ia := 1 to nx do f[ia]:=f[ia]/norm;
for ia := 1 to nx do f[ia]:=f[ia]*f[ia]*f[ia];

for ia := 1 to nx do norm := norm + f[ia]*f[ia];
ex:=ex+f[ia]*f[ia];
e[ia]:=e[ia]-ex;
end;
writeint('e[ia]=');
for ia := 1 to nx do writeint(e[ia],',');
ext[ia]:=e[ia];
m[ia]:=-1;
t[ia]:=1;
writeint('Calculation of the residuals');
for ia := 1 to h do
begin
for ja := 1 to nx do e[ia,ja]:=0;
end;
for ia := 1 to h do
begin
for ja := 1 to nx do f[ia,ja]:=f[ia,ja]-t[ia]*t[ia]*t[ia];
end;
writeint('f[ia,ja]=');
for ia := 1 to h do writeint(f[ia,ja],',');

[This text is not visible due to the image not being readable.]
BIBLIOGRAPHY


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


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


Froelich P. [1985], Fluorescence of organic compounds, Instrumentation-Research,
March 1985, p.98-103.


Guilbault[1973], Practical Fluorescence,


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

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.


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