NEURAL NETWORK ON AUTOMATIC BRAIN TUMOUR DIAGNOSIS

Shuxian Jane Wang

B.Eng. (Beijing Computer Institute, Beijing, China), 1985

School of Computer Science

McGill University

Montreal, Quebec, Canada

A project submitted to the Faculty of Graduate Studies and Research

in partial fulfillment of the requirements for the degree of

Master of Science

August 1994

Abstract

For many decades the attempts have been made to apply computer technology to biomedical problems. One aspect of this effort is related to processing information that is presented in the form of an image or a spectrum. The neural network technology has been applied in many fields, One of the applications is data classification.

This project is on automatic brain tumour diagnosis which combines the computer technology for **MRSI** (Magnetic Resonance Spectroscopic Imaging) feature extraction and the neural network technology to build a generic multilayer feed-forward architecture with supervised, back propagation learning procedure for automatic brain tumour classification by the neural network simulator tool called **Xerion**. This project also provides the statistical analysis of the neural network performance for brain tumour diagnosis. The evaluation is base on the results of training and testing different clean and noisy data sets through the neural network.

Each data sample includes six feature values choline-containing phospholipids(**Cho**), phosphocreating + creatine(**Cr**), Nacetylaspartate(**NAA**), alanine(**Ala**), lactate(**LA**), and lipid(**Lip**) for discriminating six most common types of adult supratentorial brain tumours, astrocytoma I (**as1**); astrocytoma II (**as2**); glioblastoma (**gbm**); meniningioma (**men**); metastasis (**met**), as well as normal control subjects (**nml**). The Clean data set includes 105 samples extracted from 100 patients and the Noisy data set includes at most 151 tumour voxel(tissue) samples extracted from 12 patients.

Acknowledgements

There are many that have helped me to do this project in many different ways. With deep gratitude I would like to thank Professor Renato De Mori for his many ideas, help, and guidance. I would like to thank Dr. Doug Arnold for the use of the MRI lab and his guidance in biomedical field for this project.

I would like to thank Dr. Mark Preul, Mr. Z. Caramanos, Mr. Liqun Fu and the rest of the members of MRI lab in Montreal Neurological Institute and Hospital for their help in every aspect of this project.

I am very grateful to Mr. Marco Petroni for his help with some neural network tools, and to Mr. Ameen Maluf for his help to use the computer facilities in the lab of Center of Intelligence Machine.

As well, I would like to thank all my colleagues and friends who shared with me their time and made my stay a pleasant experience. Particularly, I would like to thank Ms. Zhi Yang and her family, Ms. Suk yin N.G and her family for their constant encouragement, which gave me confidence and determination in my study.

To my father Yao Wang, my mother Yibi Chen, and my brothers Zhaoming Wang and Zhaoxin Wang, for their unfailing love, support, patience and encouragement wherever I was home or I am thousands miles away.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Overview

This project is the automatic brain tumour diagnosis by neural network. The diagnosis is for six most common types of adult supratentorial brain tumours, astrocytoma I (**as1**); astrocytoma II (**as2**); glioblastoma (**gbm**); meniningioma (**men**); metastasis (**met**), as well as normal control subjects (**nml**).

The brain tumour information is portrayed in the **MRSI** (Magnetic Resonance Spectroscopic Imaging) picture which can be processed by the tool called **SUNspec1**. The image has 32 x 32 voxels(spectra), and each voxel contains 150 points for its spectrum image.

The six feature elements are extracted from each **ROI** spectrum based on the changes in **MR** signals from choline-containing phospholipids(**Cho**), phosphocreating + creatine(**Cr**), Nacetylaspartate(**NAA**), alanine(**Ala**), lactate(**LA**), and lipid(**Lip**). The **intensity Cr**(also called **contralateral Cr** or **contra Cr**) is the average value in contralateral homologous normal appearing voxels or, if that was not possible, by the voxels in remote normal-appearing brain tissue[Preu 94]. The feature values can be the ratios of **contra Cr** by both the peaks and areas of these six elements for each **ROI** spectrum, they also can be the ratios of **contra Cr** by

the averaged tumour **ROI** spectra and averaged normal **ROI** spectra of these six elements for each patient. A **sample** is composed of these extracted six ratio feature values.

The extracted samples become the inputs for the neural network's training or testing data set. The neural network is design as multilayer, feedforward architecture, input layer with six units which represent six extracted feature values; output layer with six units which represent actual diagnosis for six tumour types ; and one hidden layer with three units. The back propagation simulator and supervised learning procedure is employed. Through the neural network, the classification for actual brain tumour type is made, and the statistical analysis is given to demonstrate the performance of the neural network for experimenting different types of sample data inputs.

Summarized above, three steps have been performed in this project:

1. Feature extraction of tumour normal voxels(spectra) data from **MRSI**.

   - Developed **SUNspec1** tool for getting the selected spectra numbers and their tumour types of the Region Of Interest(**ROI**).
   - Created program to extract the six feature values for each selected spectrum by its peak values and area values and then made different sample data files. For each patient, get the feature values for both tumour voxels and normal voxels, calculate Creatine(**Contra-Cr**) from the normal voxels, and get the spectrum sample data using each of the six feature values divided by **contra-Cr**.

2. Diagnosis for brain tumour by neural network automatically.

   - Built neural network for brain tumour diagnosis by the neural network simulator tool **Xerion**.
   - Created training and testing data sets for neural network automatically.

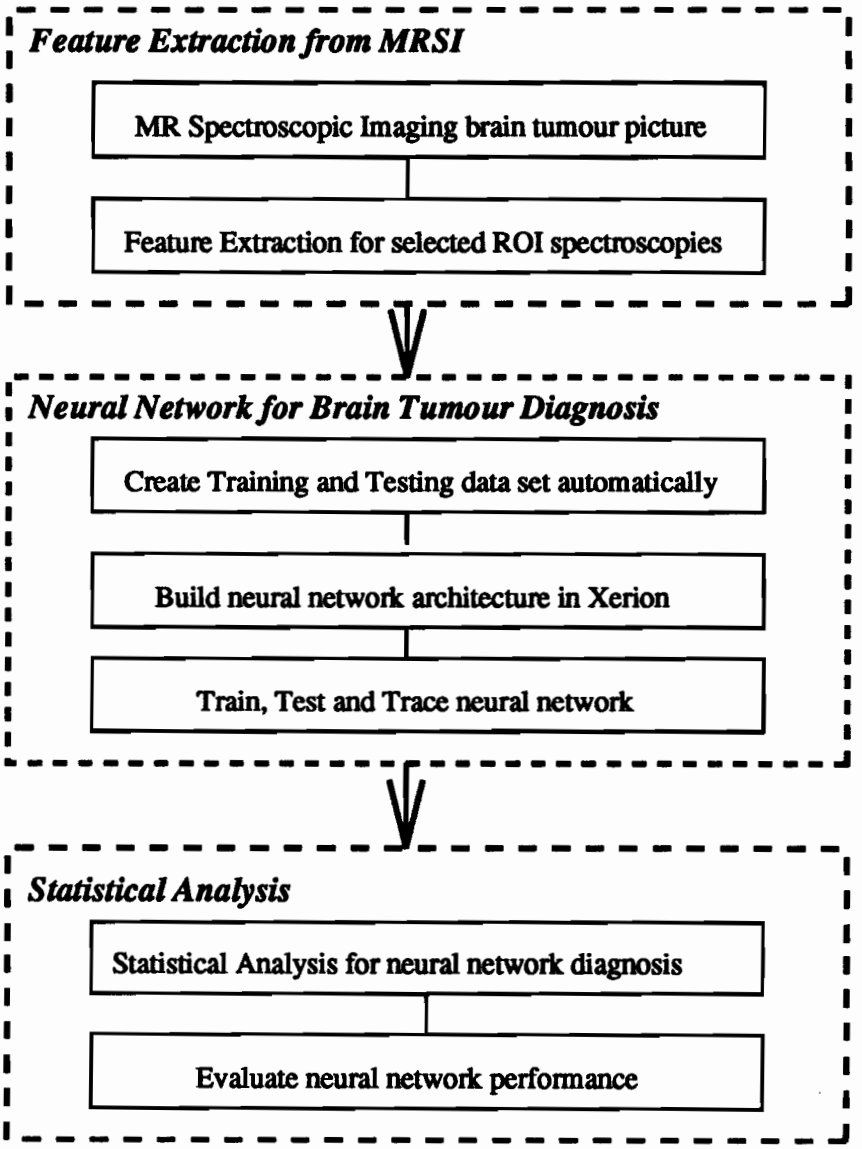


Figure 1.1: Automatic Brain Tumour Diagnosis System Architecture

- Trained and tested the neural network using the different training and testing data sets.
- Developed the tool **Xerion** for tracing the neural network's diagnosis information.

3. Made statistical analysis based on the neural network's diagnosis results to demonstrate the neural network's performance.
   - Set examine bound to pick out failure samples whose absolute difference of actual output and target output is more than the examine bound.
   - Calculated the failure samples percentages for each tumour type and give the actual diagnosis tumour type.
   - Accumulated the total success and failure diagnosis over the total testing samples for all test cases to demonstrate the performance of the neural network.

Figure 1.1 illustrates the system architecture for automatic brain tumour diagnosis.

1.2 MRI Image Processing

Knowledge and understanding of the brain tumour comes from the detailed observations carried out using a device called **MR**(Magnetic resonance) to produce an image. The image is a 32 x 32 matrix representing 1024 voxels(spectra). In each voxel, there are 150 data to represent the spectrum signal changes associated with the chemical density in the brain. So for each **MRSI**(MR Spectroscopic Image), there are 1024 spectrum data. The image processing tool **SUNspec1** has been added new functions which can save the select the region of interest(**ROI**) spectra and types from the **MRSI** for the feature extraction. Figure 1.2 illustrate the **MRSI** and its selected spectra image. On the left, two images are two slices of brain tumour for

patient *valul1*, and the numbers 1 to 4 in the image are the tumour voxels, corresponding to the spectrum number 658,659,660 and 628 and the spectrum image respectively. The number 5 to 10 are normal brain voxels, and their spectra numbers and spectra images are 556,557,654,687 and 492 respectively. In normal spectra, the three peaks are **Cho, Cr**, and **NAA** from the left to right, while **Ala, Lac** and **Lip** are not outstanding. In contrast, the tumour voxels have high peak of**Cho**, reasonable **Cr**, abnormal low **NAA**, **Ala** is just on the left shoulder of **Lac**, extream high **Lac**, and obvious **Lip**.

The various pointers and other objects in the network can be examined all

1.3 Feature Extraction

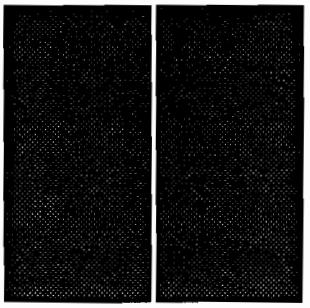
The feature values are obtained from the ROI spectrum by extracting the chemicals of **Cho, Cr, NAA, Ala, LA and Lip** to the ratio of **Contra-Cr** The extracted feature values can be one of four types, peaks, areas, averaged peaks and averaged areas for the six feature elements.

1.4 Neural Network

1.4.1 Neural Network Simulator Xerion

Xerion is a Neural Network simulator developed and used by the connectionist group at the University of Toronto. Xerion V3.1 contains libraries of routines for building networks, and graphically displaying them. As well it contains an optimization package which can train nets using several different methods including conjugate gradient. It is written in C and uses the X window system to do the graphics[Xeri 93].

This project also involves in how to install **Xerion** into our system; how to set up environments for running **Xerion**; how to build specific architecture for brain



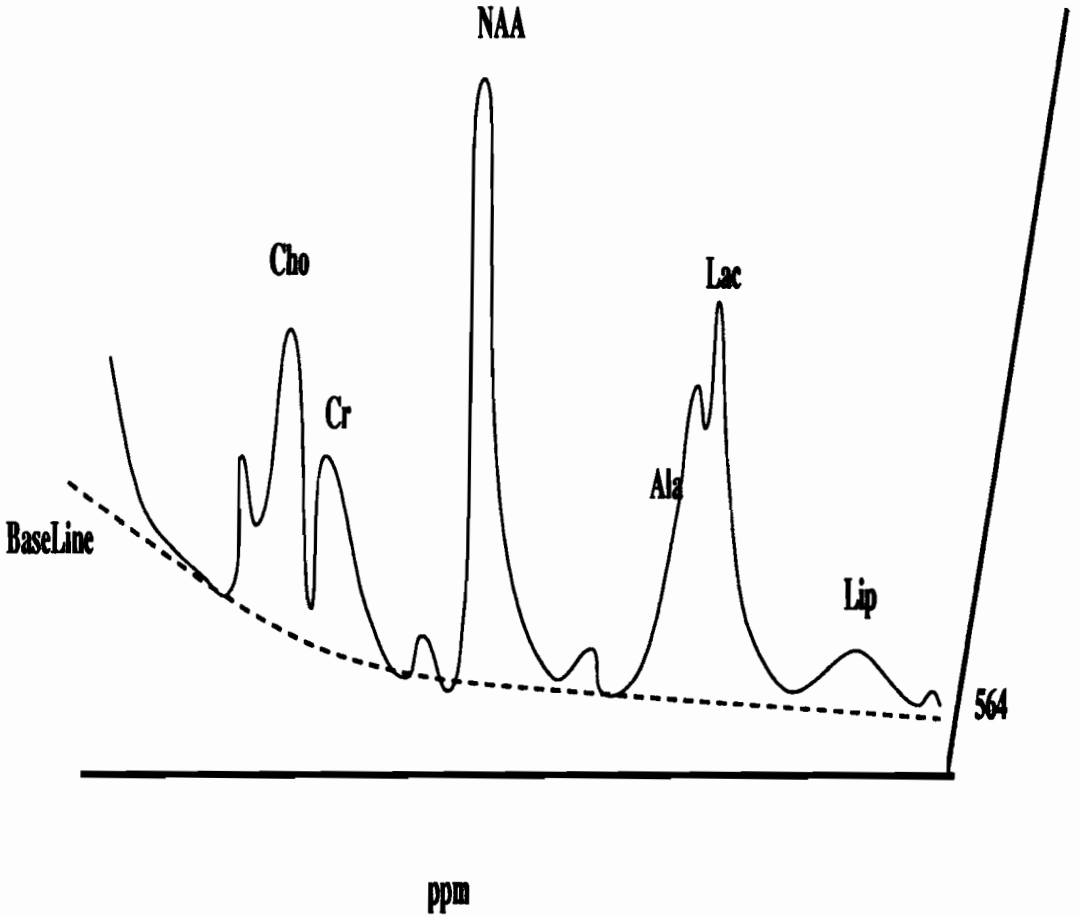
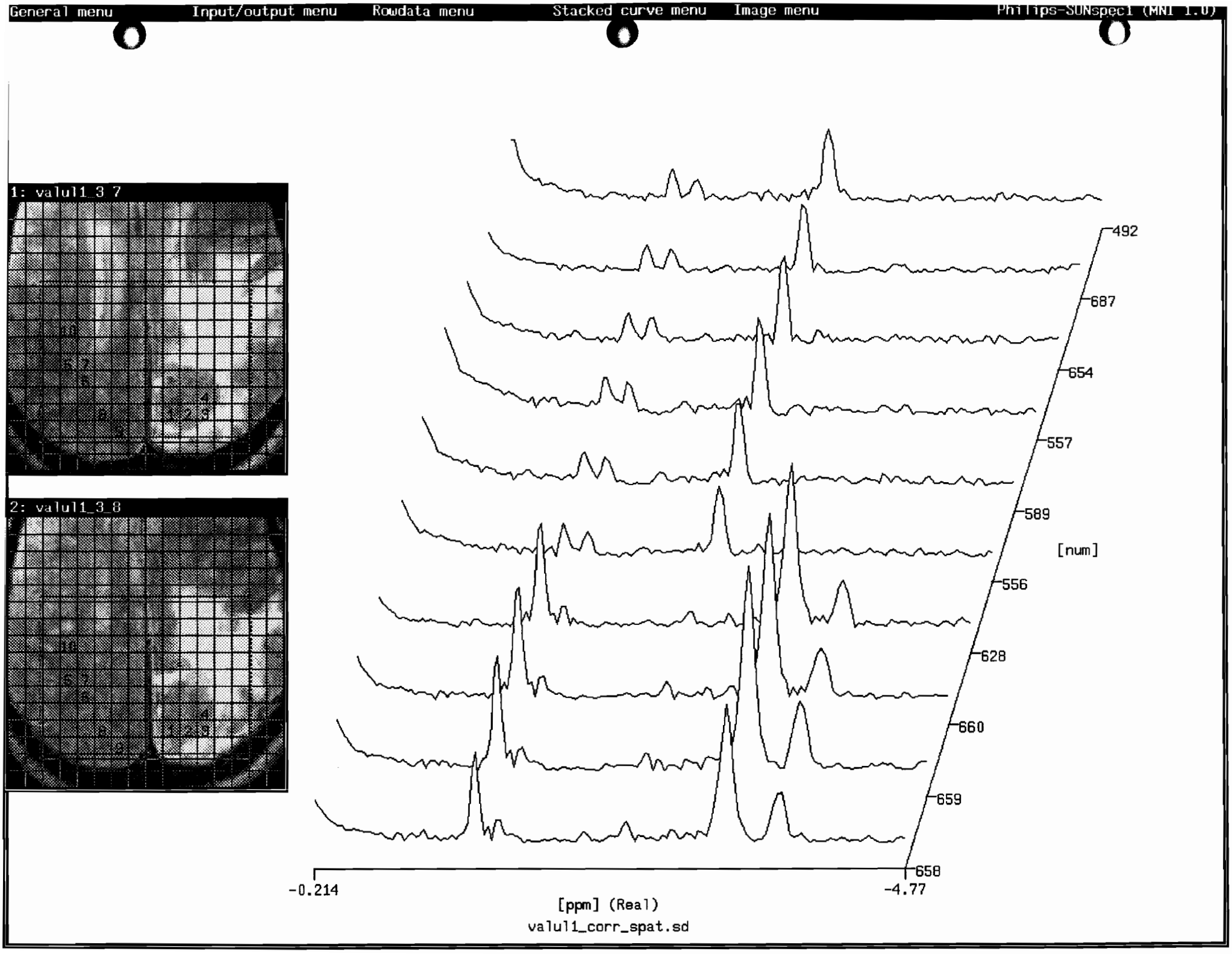


Figure 1.2: MRSI of Brain Tumour


General menu
Input/output menu
Rowdata menu
Stacked curve menu
Image menu
Philips-SUNspec1 (MNI 1.0)
1: valul1_3 7
2: valul1_3_8
492
687
654
557
589
[num]
556
628
660
659
658
-0.214
-4.77
[ppm] (Real)
valul1_corr_spat.sd

tumour diagnosis and trace the neural network diagnosis information using **Xerion**'s libraries routines.

1.4.2 Neural Network Data Format

Neural network requires the data in [0,1] intervals normally, and **Xerion** has its own format for the data set. So the data after the feature extraction has to be transformed to **Xerion** format.

1.4.3 Build Neural Network Architecture

Xerion is a Neural Network simulator. It can be used to implement many different Neural Network paradigms.

Back-propagation, multilayer feedforward with supervised learning neural network is designed for the brain tumour diagnosis. The input layer has 6 input units to represent the extracted six chemical feature values; the output layer has 6 output units to represent 6 different types of brain tumours; the hidden layer has 3 hidden units.

1.4.4 Create Neural Network Data Sets

The program **FeatureExtract.c** is created for producing different data sets automatically by selecting different options. After the extracted data file name is given, by choosing "all", all the data go to the testing data set; by choosing "none", all the data will be the training data set; by giving a specific sample name, this sample will go to the testing data set and the rest samples will go to the training data set. This is applied "leave-one-out" method. The "leave-one-out" can be leave one patient's sample data out which include many tumour voxels, and also can be leave one tumour voxel sample out if the specific sample name is the individual tumour voxel number. According to the input data file type, the created data sample

can have many types of peak or area for each voxel; peaks or areas for one patient, peaks or areas for all patients, averaged peaks or areas for all patients.

1.4.5 Train and Test Neural Network

Once the data samples have been added to the net, they can be trained. There are many commands in **Xerion** for training a network, but all are different versions of one principal command: *minimize*. After the neural network has been trained, it can be tested for the classification of the test data set.

1.4.6 Trace Neutral Network Information

The neural network information for testing data set are traced by the routine *runExamples*. For each sample data, the traced information includes the six input feature values, the six actual output values and the six error measurement values(absolute difference between the target output and actual output). According to the *examine bound* given by the user, this routine finds out the success samples and "failure" samples for later statistical analysis. The success sample is defined if all of six error measurement values are less than the *examine bound* and the "failure" sample is defined if any one or more of the six error measurement values are more than the *examine bound*. In the statistical analysis procedure, the "failure" will be decided if it is real failure or not.

1.5 Statistical Analysis

From the traced information, the statistical analysis evaluates the success rate for all the tested samples. For those over *examine bound* samples, this analysis gives the percentages for the six possible tumour types, the biggest percentage will be the actual tumour type by "winner takes all" assumption for diagnosis decision. So the over *examine bound* samples may or may not be failure depending on whether they

affect the diagnosis decision or not.

1.6 Scope and Organization of This Report

The basic theme of this project is the neural network technology for automatic brain tumour diagnosis, which associated to feature data extraction from the image data, and statistical analysis for the neural network's performance.

Chapter 2 gives the biomedical background, summarizes the image processing tool **SUNspec1** and the developed new functions, as well the technology of using **MRSI** for brain tumour feature data extraction.

The neural network theory is derived from many disciplines, including psychology, mathematics, neuroscience, physics, engineering, computer science, philosophy, biology, and linguistics. In Chapter 3, some neural network concepts and theory are introduced which involved in this project. The main theme of whole process for automatic brain tumour diagnosis in neural network is also provided.

The statistical analysis and evaluation neural network performance are presented in Chapter 4.

Finally, Chapter 5 summarizes the work accomplished in this project and suggests future research directions in the areas of image pattern recognization and neural network architecture design.

Chapter 2

Feature Extraction

2.1 Biomedical Background

Attaining high specificity in preoperative brain tumour diagnosis remains a challenge. The gains made by **MRI**(Magnetic resonance imaging) in sensitive detection of brain tumours have not been paralleled by improved specificity. Even pathological information from tumour biopsies may be indeterminate [Preu 94].

Proton **MRI**(Magnetic resonance imaging) and **MRSI**(Magnetic resonance spectroscopic imaging) of brains were obtained using a Philips Gyroscan S15 HP combined imaging and spectroscopy system operating at 1.5 tesla, and a standard 30 cm diameter "mirror" head coil. These proton images, based on the signal from water, were used for selection of a large volumn of interest(VOI) for proton spectroscopic imaging[Arno 91].

Including the normal control subject(**nml**), and other five most common types of adult supratentorial brain tumours, astrocytoma I (**as1**); astrocytoma II (**as2**); glioblastoma (**gbm**); meniningioma (**men**); metastasis (**met**) can be biochemically characterized by the Proton **1H MRSI**[Preu 94].

Now the problem is how to extract feature values from the **MRSI** data and automatically classify six different types of tumours using these features.

2.2 New Functions Added to Sunspec1

Metabolite-based images were reconstructed for Cho, Cr, lactate(Lac) and NAA, etc.(defined later) based on peak areas in magnitude spectra, and using interactively determined boundaries for the individual resonances[Arno 91]. The image can be processed by **SUNspec1**.

SUNspec1 is a predevelopment software package for processing of 1D and 2D SI imaging/spectroscopy data sets by **Philips Medical Systems** in The Netherlands. It can be used in both VAX and SUN systems. Many rich data transformation abilities , image and spectroscopy combined data processing functions, and friendly graphical user interface make it a powerful tool. For example, Input/Output functions deal with saving and reading rowdata, image data, curve dump; Rowdata processing functions apply different data transformation theories to the rowdata; Viewing functions display the spectrum in different ways by changing its properties; Image functions process images data and illustrate the images represented by 32 x 32 "matrix". A element of "image matrix" is called **voxel**. We also refer **voxel** to **spectrum** because each voxel consists of 150 data points which can be used to draw a spectrum image. **ROI** is a large region of interest, including the tumour and contralateral or remote MRI-normal-appearing brain tissue. One of the most important image functions is *ROI-select*, which selects voxels in the **ROI** from the image and draw the corresponding selected **ROI** spectrum at the same time. Actually ROI-select plays a key step to feature extraction.

Proton MRSI is a major advance over single voxel spectroscopy for non-invasive chemicopathological evaluation of demyelinating lesions, as it provides metabolite maps of the regional distribution of metabolite changes in what is fundamentally a regionally heterogeneous problem.

Even though **Sunspec1** is a wonderful tool for image/spectroscopy data processing, it still has its limitations. Some new **Output functions** stated below have been developed for **SUNspec1** in this project.

1. Saved arbitrary selected ROI spectra numbers and associated tumour types into **.sp** or **.spar** file instead of only saving the continuous spectra numbers.

2. Changed the graphical user interface to display the selected spectra numbers for user to decide whether keep them in the parameter file.

3. Expended **Sunspec1** data structure file for adding more functions later on.

These new functions make *ROI-select* more efficient because the users can save any spectra numbers and tumour types into a file for feature extraction and reviewing them later. The second advantage is that it saves a lot of disk space by storing particular, discrete and interested spectra data instead of storing continuous spectra data with some of uninterest data. For example, if the user selected spectra No.400 and No.500, by the new function, he only needs to save 400 and 500 two spectra data and associated tumour types; but by the unupdated **SUNspec1**, he has to save 101 spectra data from 400 through 500 without knowing their tumour types.

The complete programs for the developed new functions are in Appendix A.

2.3 Feature Extraction from Spectroscopic Image

Feature extraction is concerned with the problem of finding and identifying feature elements and their values which may be hidden amid a great deal of confusing and irrelevant data. It is impractical to attempt to base decisions on whole 1024 voxel's data since many of the differences between individual image or spectrum are not relevant. therefore the purpose of feature extraction is customary to work with feature elements' values rather than the whole image. Although the *ROI-select* has played a key step to extract the **ROI** voxels, 150 data points in each spectrum(voxel) are still too big and complex for catching the characteristics of a tumour. A particular process in this project is that of automatically extracting 6 feature elements'

values from these 150 data points, and these extracted feature values make a vector called a **sample.**

What are these six elements and why are these six elements taken rather than eight or four? These questions have been answered by the experts from many years' experience. The six feature elements are based on the changes in **MR** signals from choline-containing phospholipids(**Cho**), phosphocreating + creatine(**Cr**), Nacetylaspartate(**NAA**), alanine(**Ala**), lactate(**LA**), and lipid(**Lip**)[Preu 94]. Associated these six element values are called **feature values.**

After the selected ROI spectra numbers and their tumour types are saved into the parameter file **.sp** or **.spar**(the former for SUN system and the latter for VAX system) by the new developed Output functions, the feature extraction program **FeatureExtract.c** can be run for deciding the six feature values in four types. As shown from Figure 2.1 to Fifure 2.6 are the spectra for **as1, as2,gbm,men,met** and normal voxel **nml** six types of tumours. The several spectra on the top with higher peak **NAA** in the middle are the normal voxels for each **MRSI** which are used to calculate **contra-Cr.** During the feature catching process, the spectra *ppm* intervals and base line are the major factors. For more detail, refer to the source codes in Appendix A. The concept of **contra-Cr** has been defined in Chapter 1. The feature data type(*sample data type*) listed below:

- Peaks feature values.

  The six Peaks feature values to the ratio of **contra-Cr.**

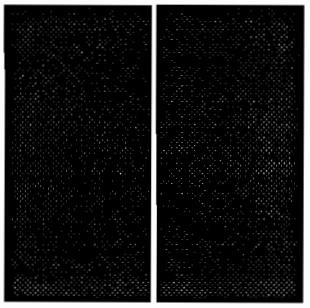
- Peak Areas feature values.

  The six Areas feature values divide by **contra-Cr.**

- Averaged Peak *tumour* and *normal* feature values.

  Calculate Average six *tumour* Peaks feature values and six *normal* Peaks values, and then calculate the ratios of **contra-Cr** for each of these two samples.

- Averaged Areas *tumour* and *normal* feature values.



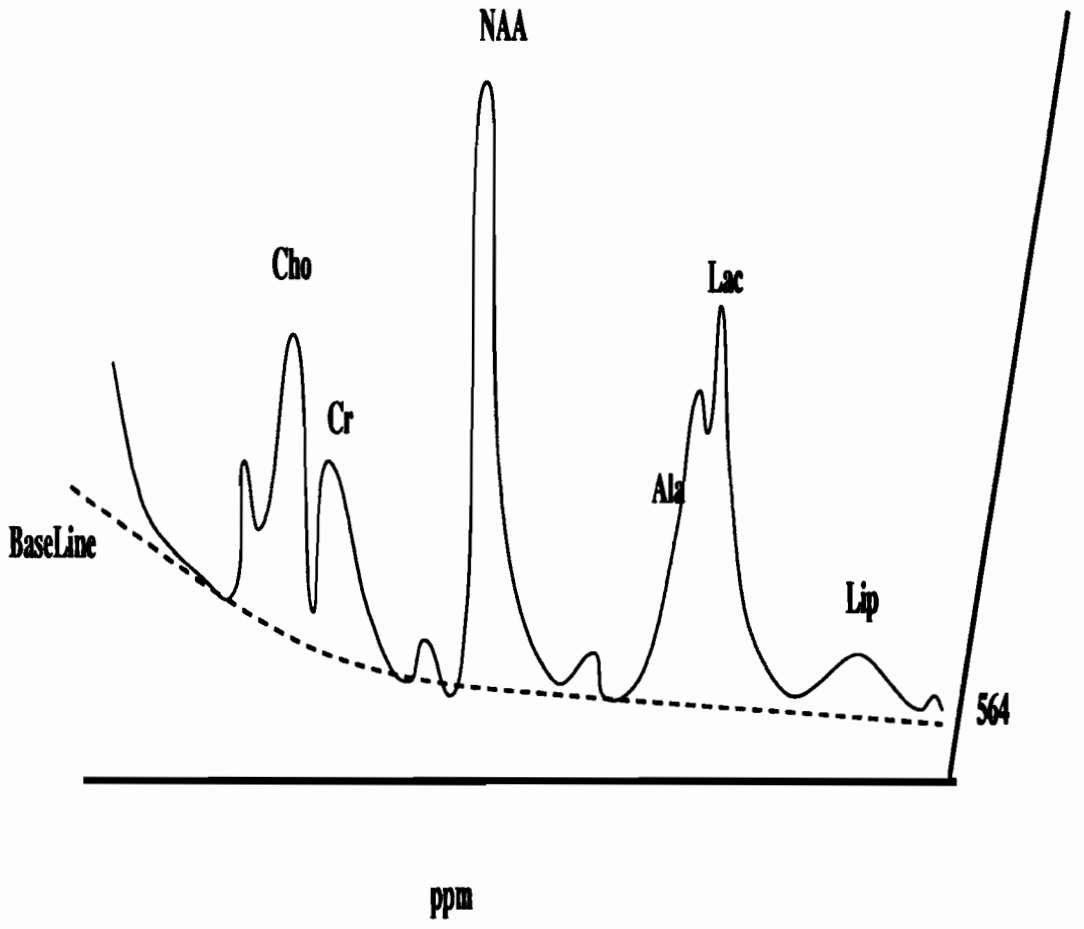
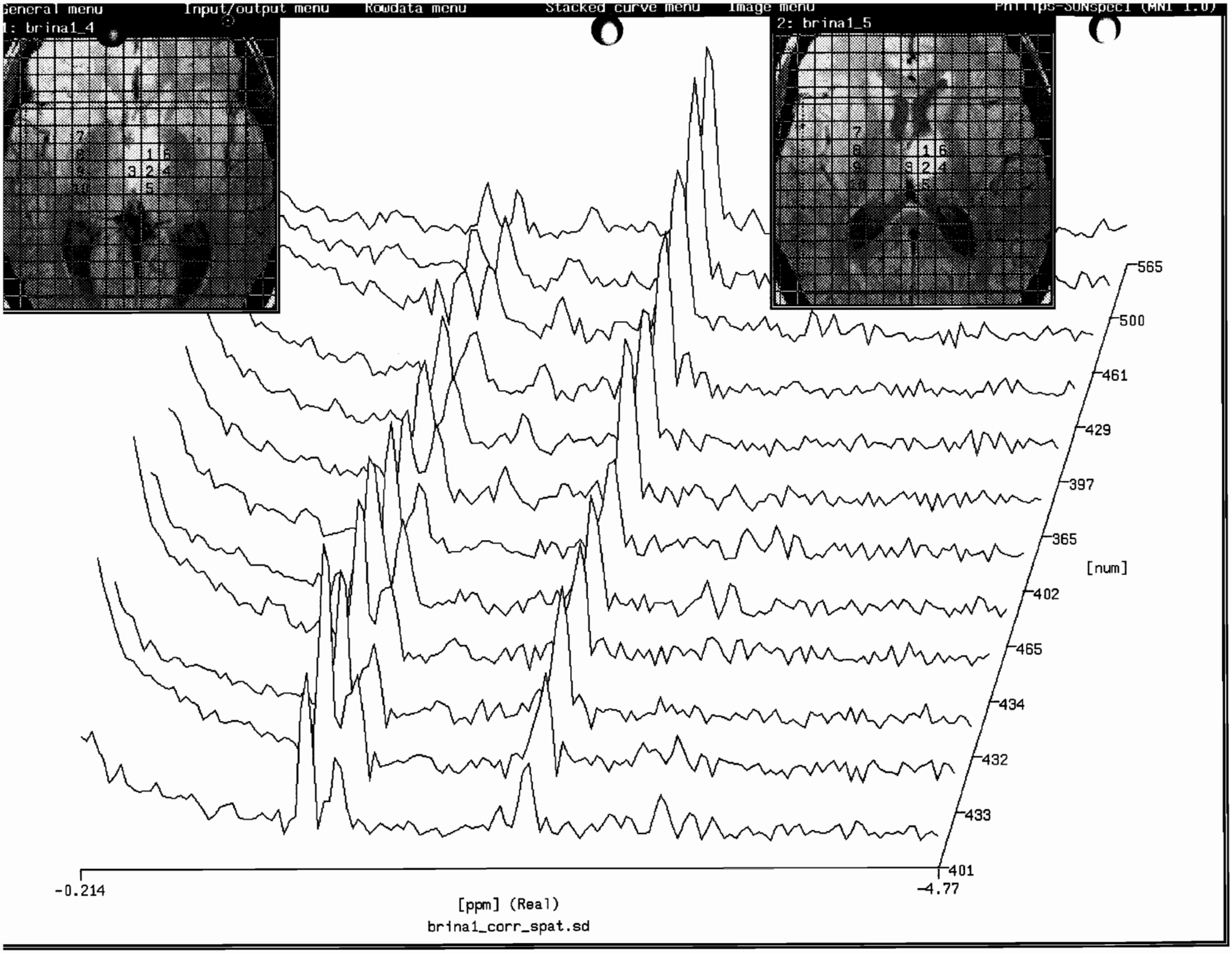
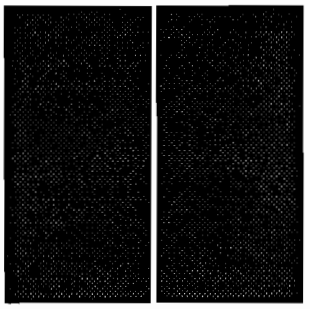


Figure 2.1: MRSI Tmour as1 Feature


General menu
Input/output menu
Rowdata menu
Stacked curve menu
Image menu
Philips-SUNspect (MNI 1.0)
1: brina1_4
2: brina1_5
565
500
461
429
397
365
[num]
402
465
434
432
433
401
-0.214
-4.77
[ppm] (Real)
brina1_corr_spat.sd



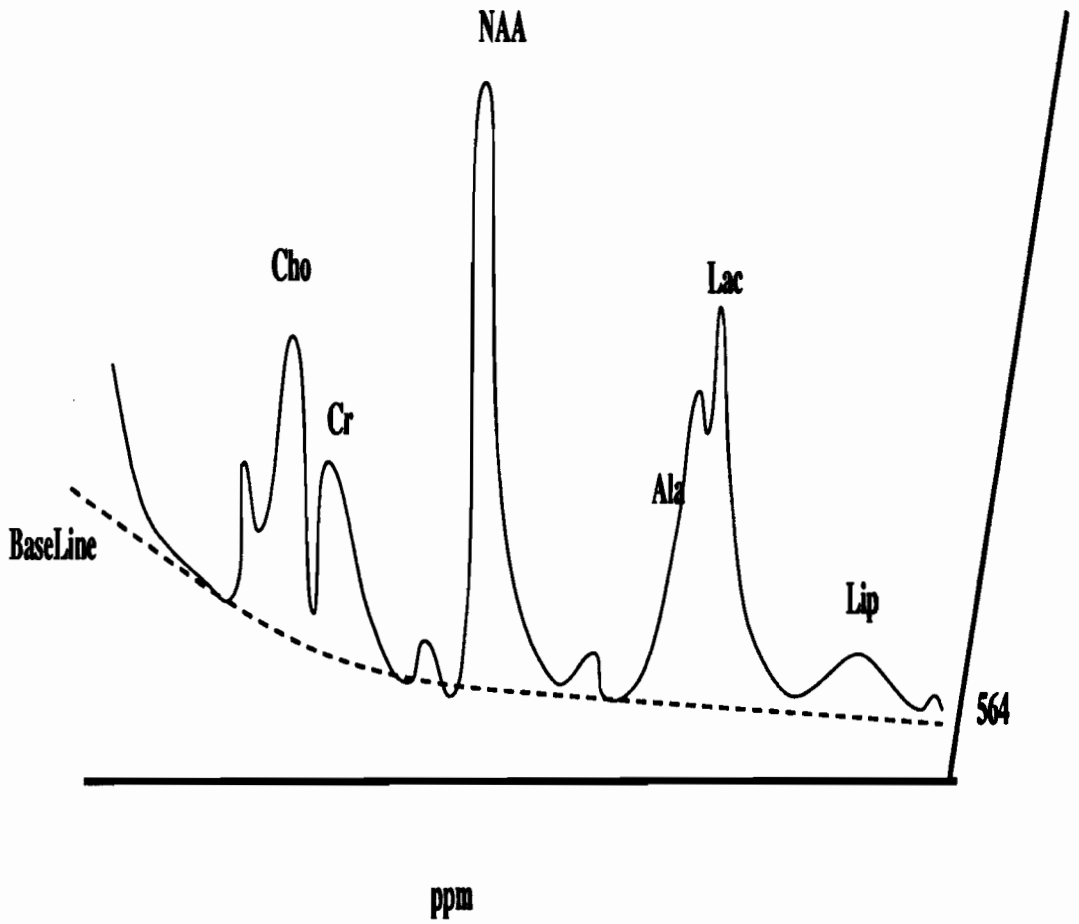
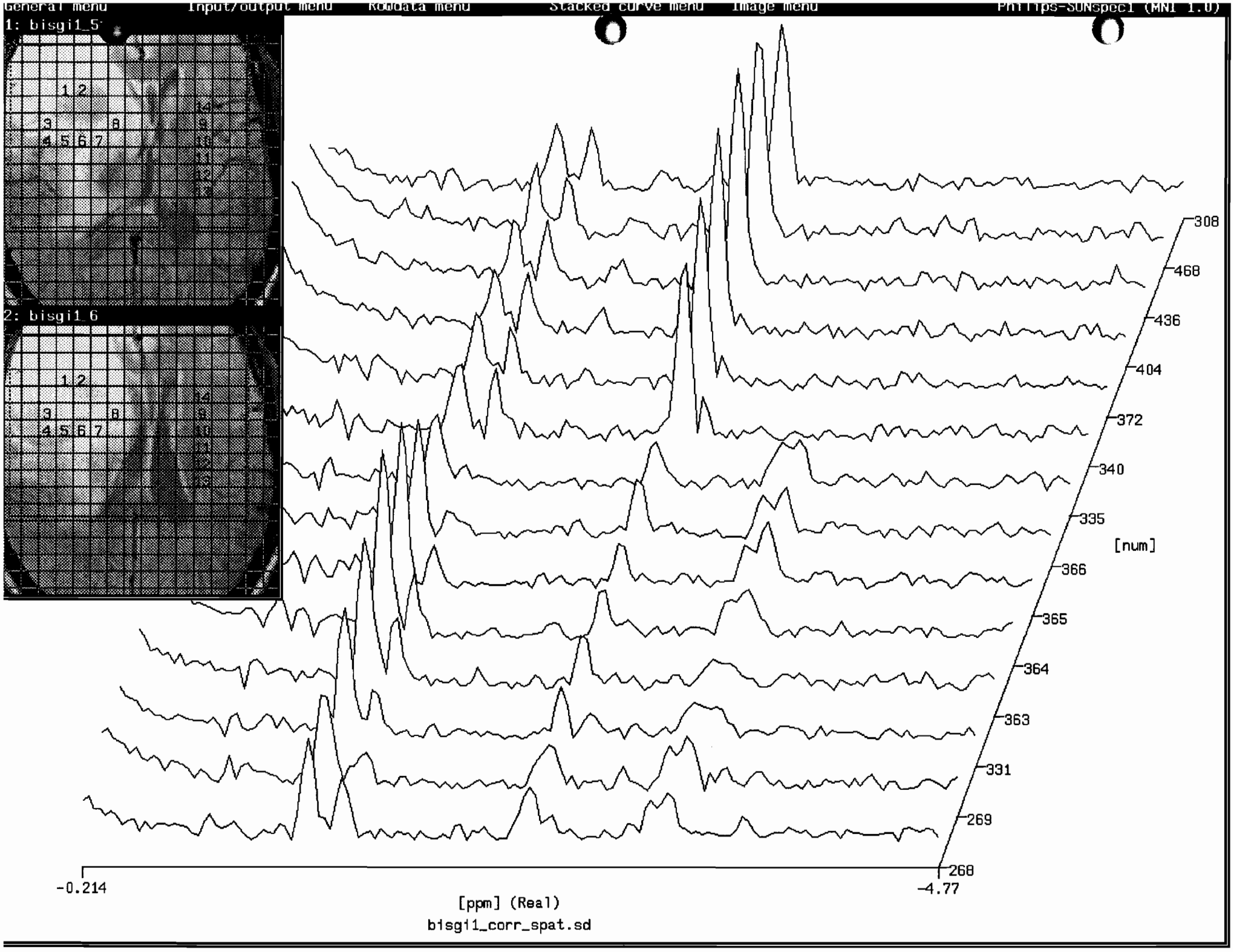
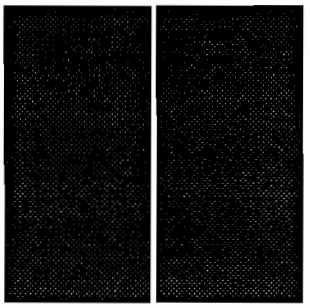


Figure 2.2: MRSI Tmour as2 Feature


General menu
Input/output menu
Rawdata menu
Stacked curve menu
Image menu
Philips-SUNspec1 (MNI 1.0)
1: bisgi1_5
2: bisgi1_6
308
468
436
404
372
340
335
[num]
366
365
364
363
331
269
268
-0.214
-4.77
[ppm] (Real)
bisgi1_corr_spat.sd



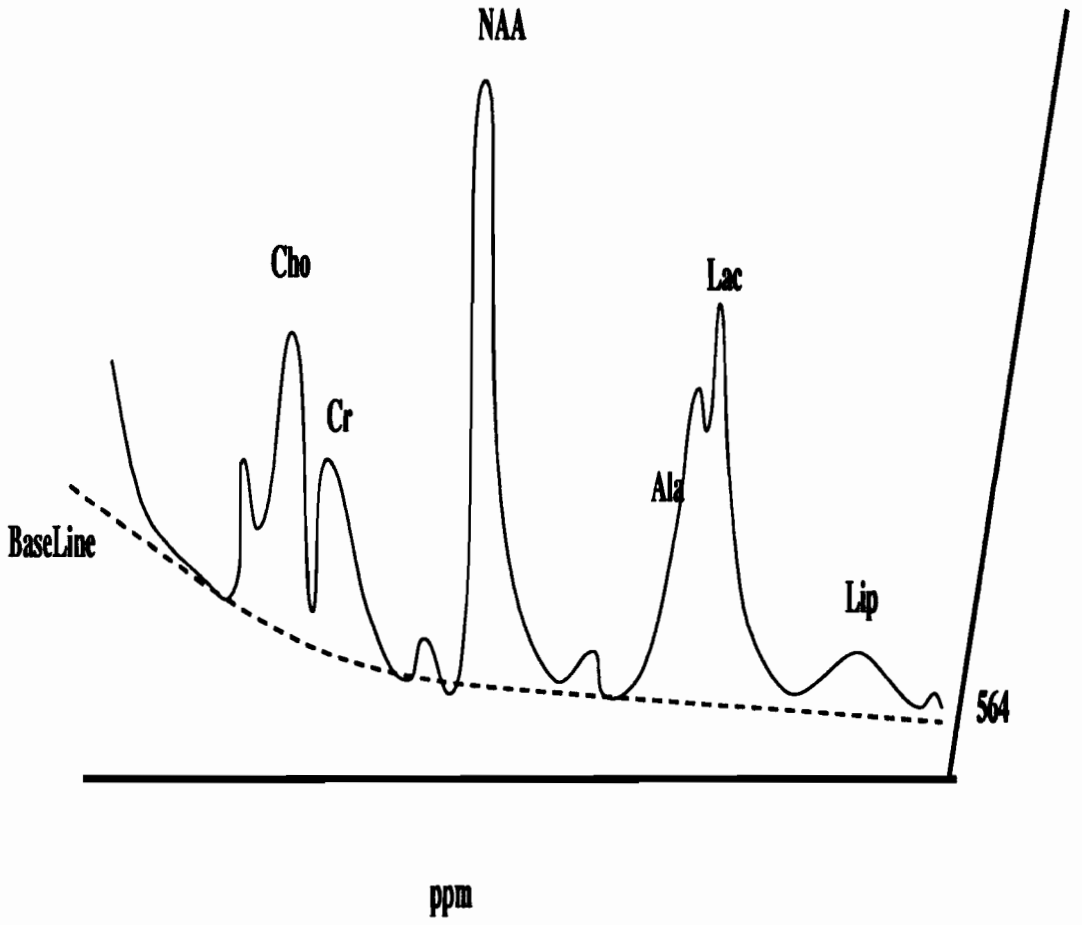
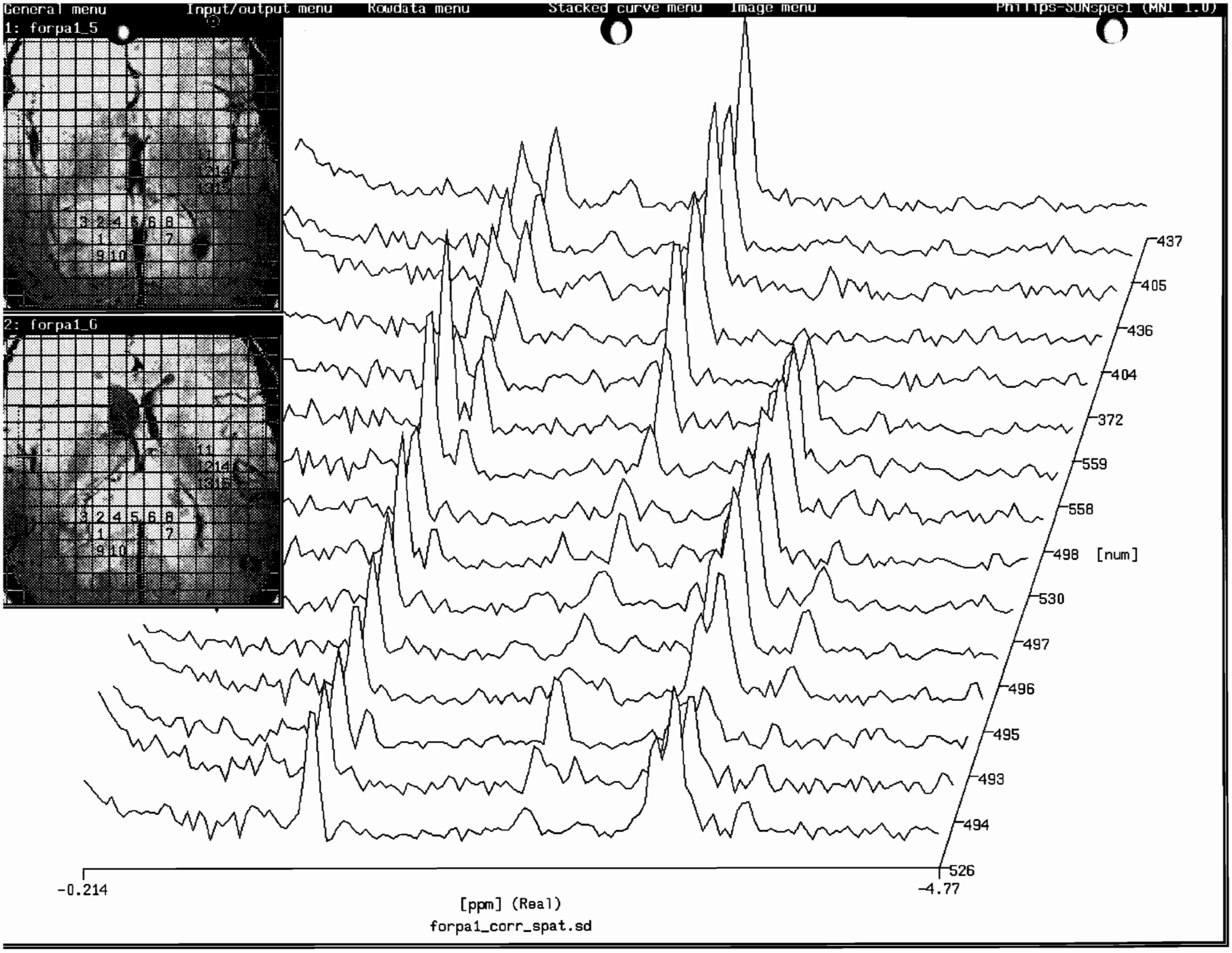
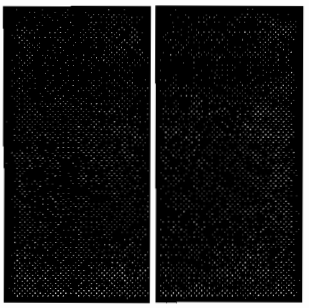


Figure 2.3: MRSI Tmour gbm Feature


General menu
Input/output menu
Rowdata menu
Stacked curve menu
Image menu
Philips-SUNspec1 (MNI 1.0)
1: forpa1_5
2: forpa1_6
437
405
436
404
372
559
558
498 [num]
530
497
496
495
493
494
526
-0.214
-4.77
[ppm] (Real)
forpa1_corr_spat.sd



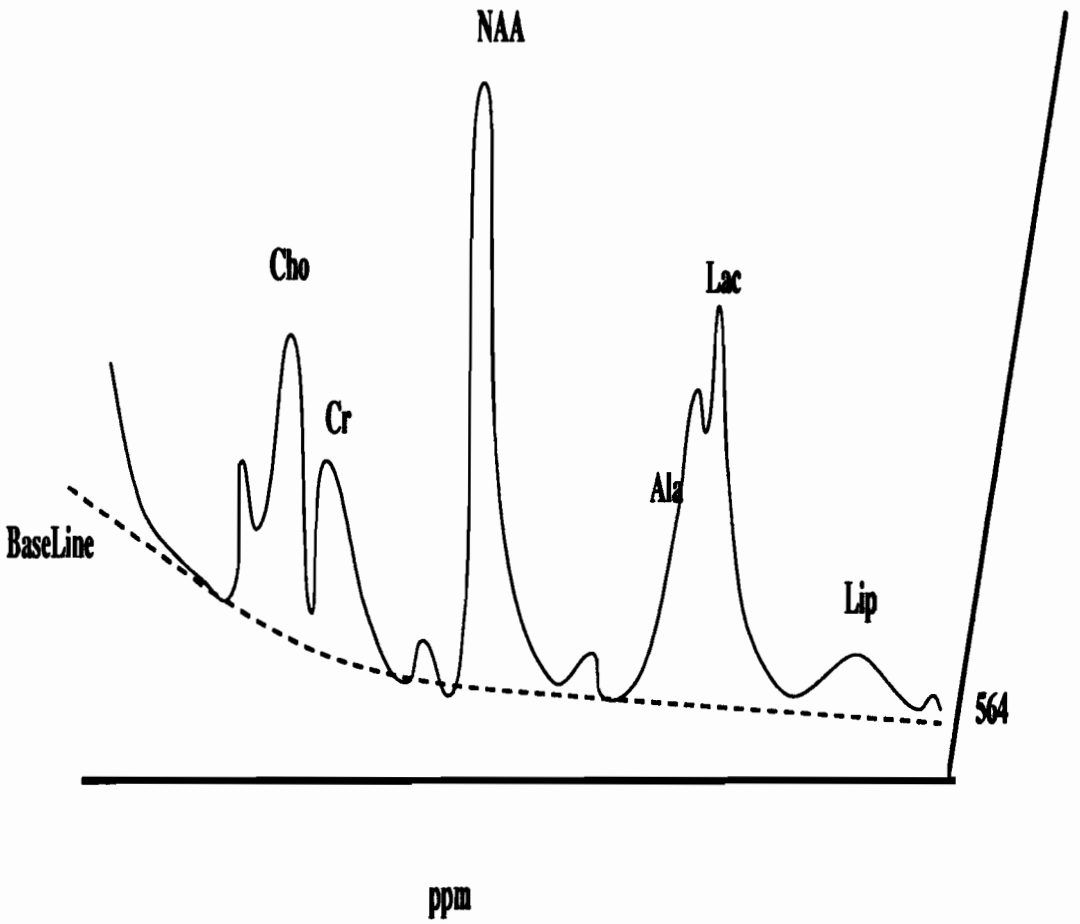
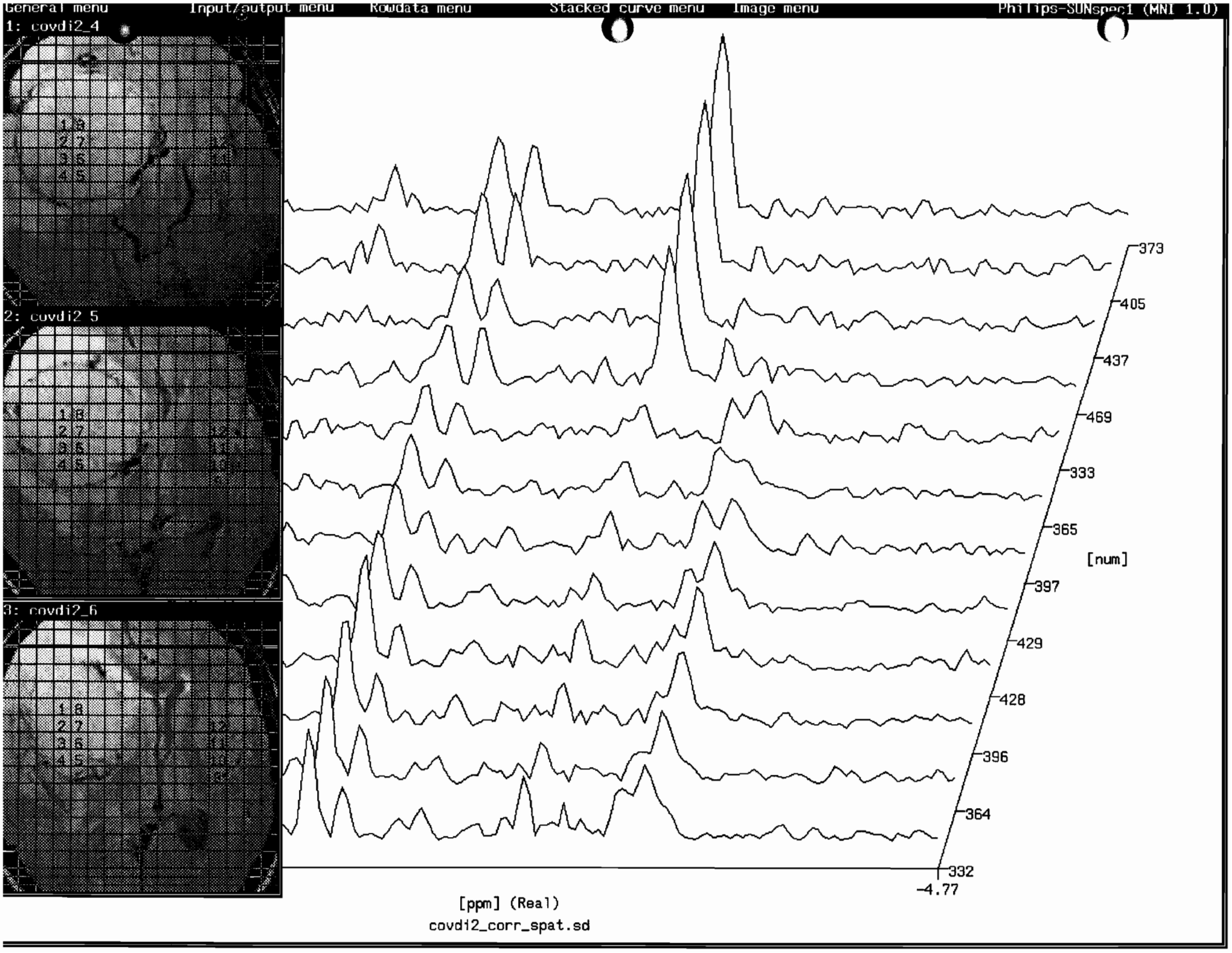
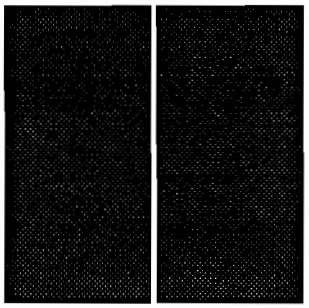


Figure 2.4: MRSI Tmour men Feature


General menu
Input/output menu
Rawdata menu
Stacked curve menu
Image menu
Philips-SUNspec1 (MNI 1.0)
1: covdi2_4
2: covdi2_5
3: covdi2_6
373
405
437
469
333
365
[num]
397
429
428
396
364
332
-4.77
[ppm] (Real)
covdi2_corr_spat.sd



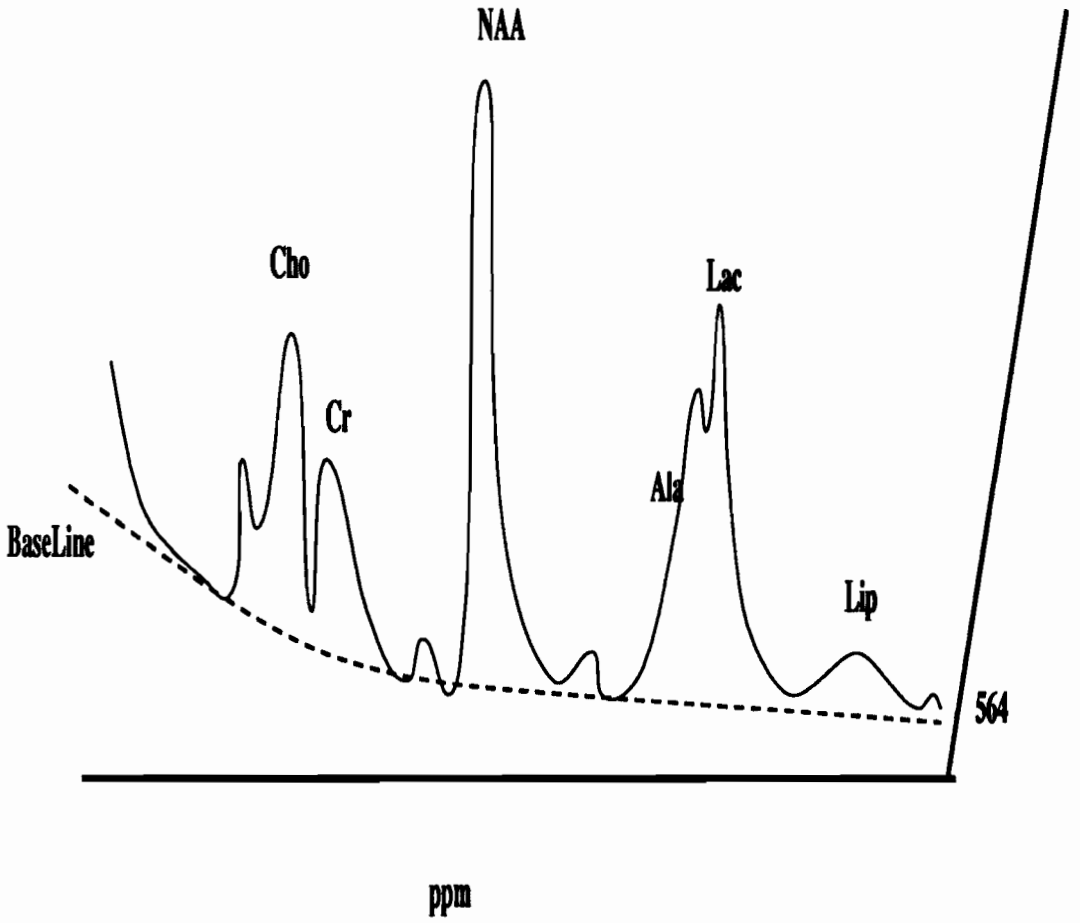
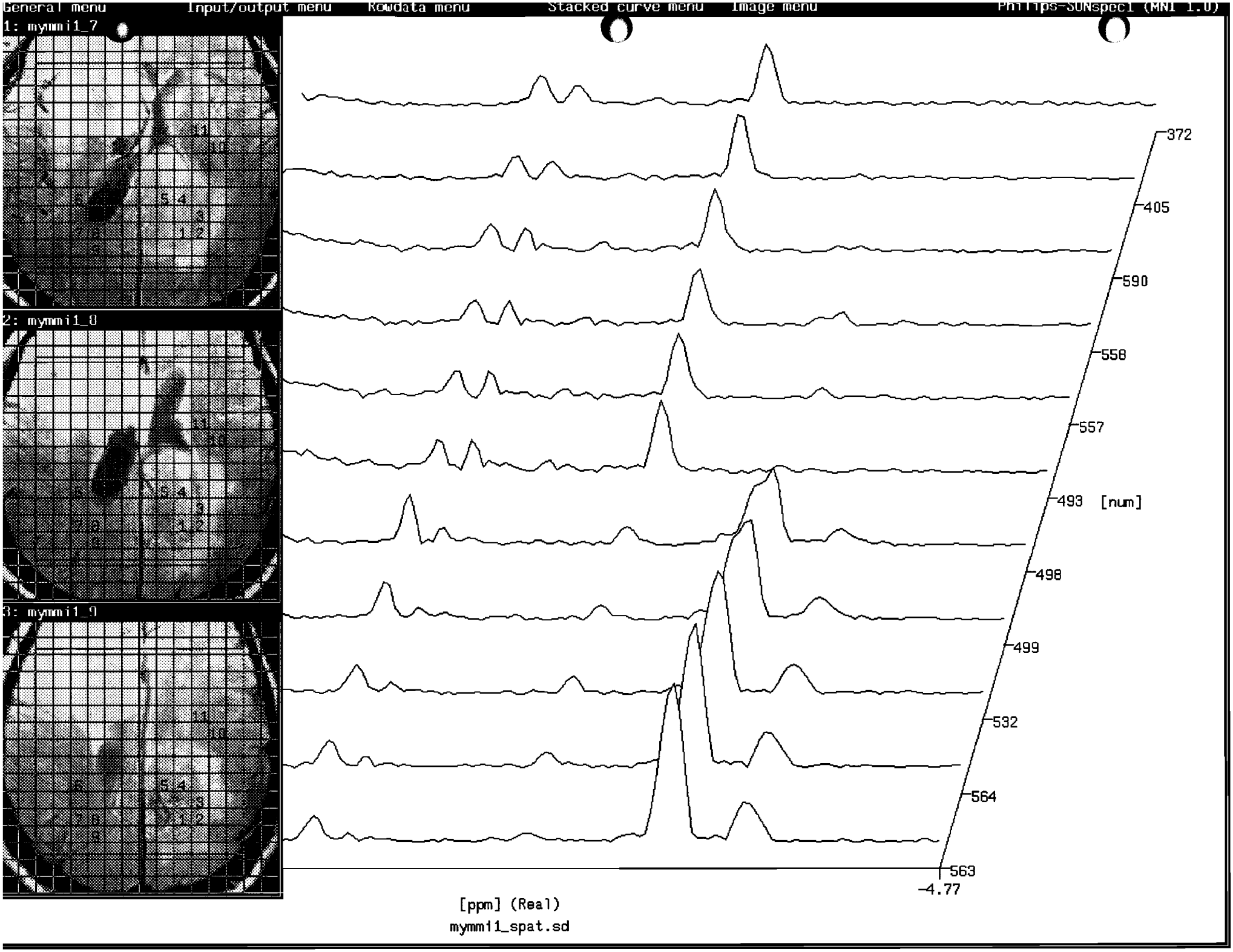
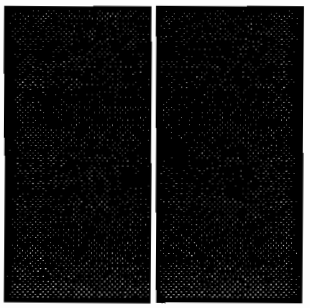


Figure 2.5: MRSI Tmour met Feature


General menu
Input/output menu
Rawdata menu
Stacked curve menu
Image menu
Philips-SUNspec1 (MNI 1.0)
1: mymmi1_7
2: mymmi1_8
3: mymmi1_9
372
405
590
558
557
493 [num]
498
499
532
564
563
-4.77
[ppm] (Real)
mymmi1_spat.sd



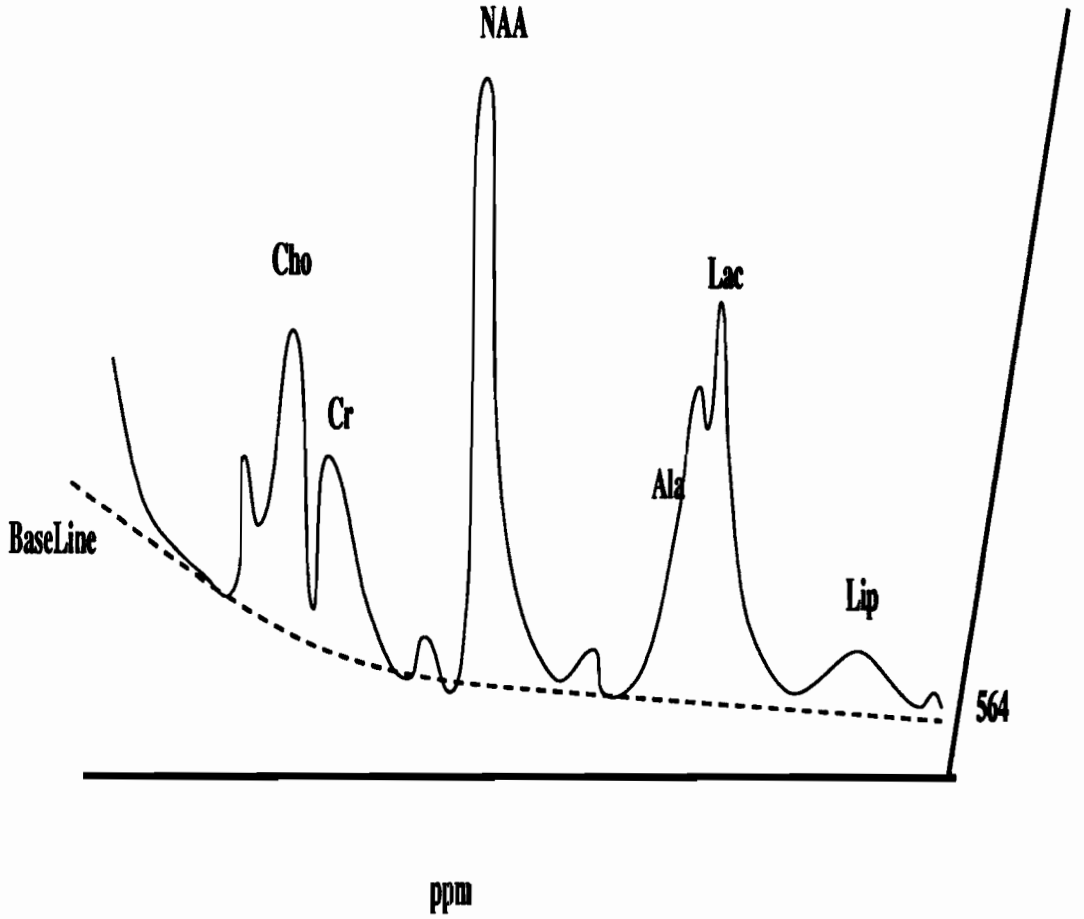
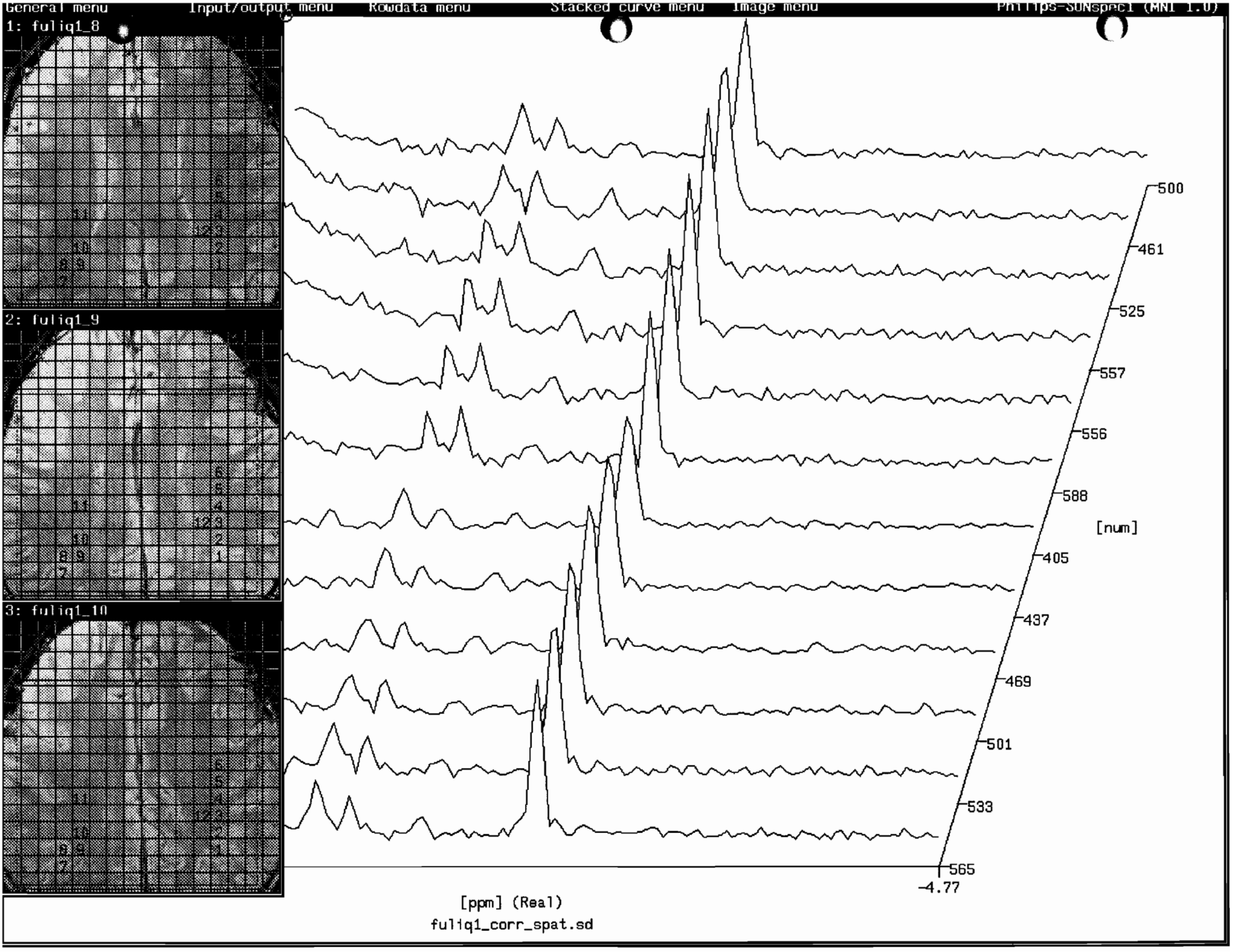


Figure 2.6: MRSI Tmour nml(normal) Feature


General menu
Input/output menu
Rowdata menu
Stacked curve menu
Image menu
Philips-SUNspect (MNI 1.0)
1: fuliq1_8
2: fuliq1_9
3: fuliq1_10
500
461
525
557
556
588
[num]
405
437
469
501
533
565
-4.77
[ppm] (Real)
fuliq1_corr_spat.sd

Calculate Average six *tumour* Areas feature values and six *normal* Areas values, and then calculate the ratios of **contra-Cr** for each of these two samples.

Here *tumour* indicates one of the tumour types **Cho, Cr, NAA, Ala, Lac** and **Lip**; and *normal* means tumour type **nml**.

Feature extraction is very important to the accuracy of classification in later step stated in Chapter 3. So far we don't know what kind of feature data type is best for successful classification, but we can experiment as many as possible cases by inputing different types of feature data sets through the neural network, and then evaluate the coming out results to find out which feature data type is the best. This is the purposes of extracting so many different types of features values.

Chapter 3

Neural Network

This chapter includes overviewing the neural network basic theories, introducing neural network simulator **Xerion**, and pay more attention to describing how the neural network on automatic brain tumour diagnosis is designed and implemented by employing the theories and **Xerion**.

3.1 Neural Network Theory

Artificial network systems (ANSs) are mathematical models of theorized mind and brain activity. ANSs are also referred to as **neural networks, connectionism, adaptive systems** and so on [Simp 90].

Neural networks can be used in many fields, such as speech, vision, touch, knowledge processing and motor-control, pattern matching, optimization problems, system modeling and function approximation.

The key issue in neural networks is learning algorithms which concern to produce networks that generalize correctly to new cases after training on a sufficiently large set of typical cases from some domain.

Neural network models typically consist of simple, neuron-like processing elements called "units" that interact using weighted connections. Each unit has a "state" or "activity level" that is determined by the input received from other units

in the network. One common, simplifying assumption is that the combined effects of the rest of network on the j^{th} unit are mediated by a single scalar quantity, x_j. This quantity, which is called the "total input" of unit j, is usually taken to be a *linear* function of the activity level of the units that provide input to j:

$$x_j = -\theta_j + \sum_i y_i w_{ji} \qquad (1)$$

where y_i is the state of the i^{th} unit, w_{ji} is the weight on the connection from the i^{th} to the j^{th} unit and θ_j is the threshold of the j^{th} unit.

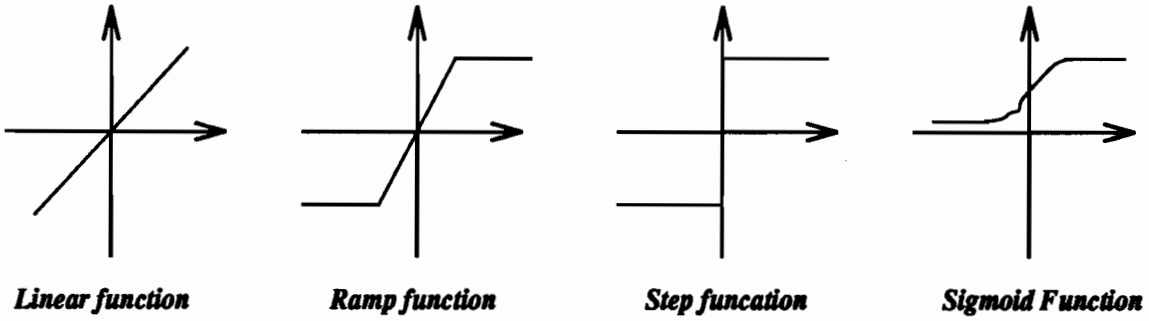


Figure 3.1: Neural Network's Threshold Functions

The **threshold functions** include *linear, ramp, step, and sigmoid* four common functions[Dayh 90]. Figure 3.1 shows typical of these functions and Eq. 2 gives the most common used *sigmoid function* which is for continuous states with non-linear input-output function.

$$y_j = \frac{1}{1 + e^{-x_j}} \qquad (2)$$

The states of the units or thresholds are used to encoded the network.

The neural network topology[Judd 90] contains *feedforward and feedback,* showed in Figure 3.2.

Learning procedures of neural networks can be divided into three broad classes:

1. **Supervised procedures:**

   Incorporates an external teacher to specify the desired output vector,

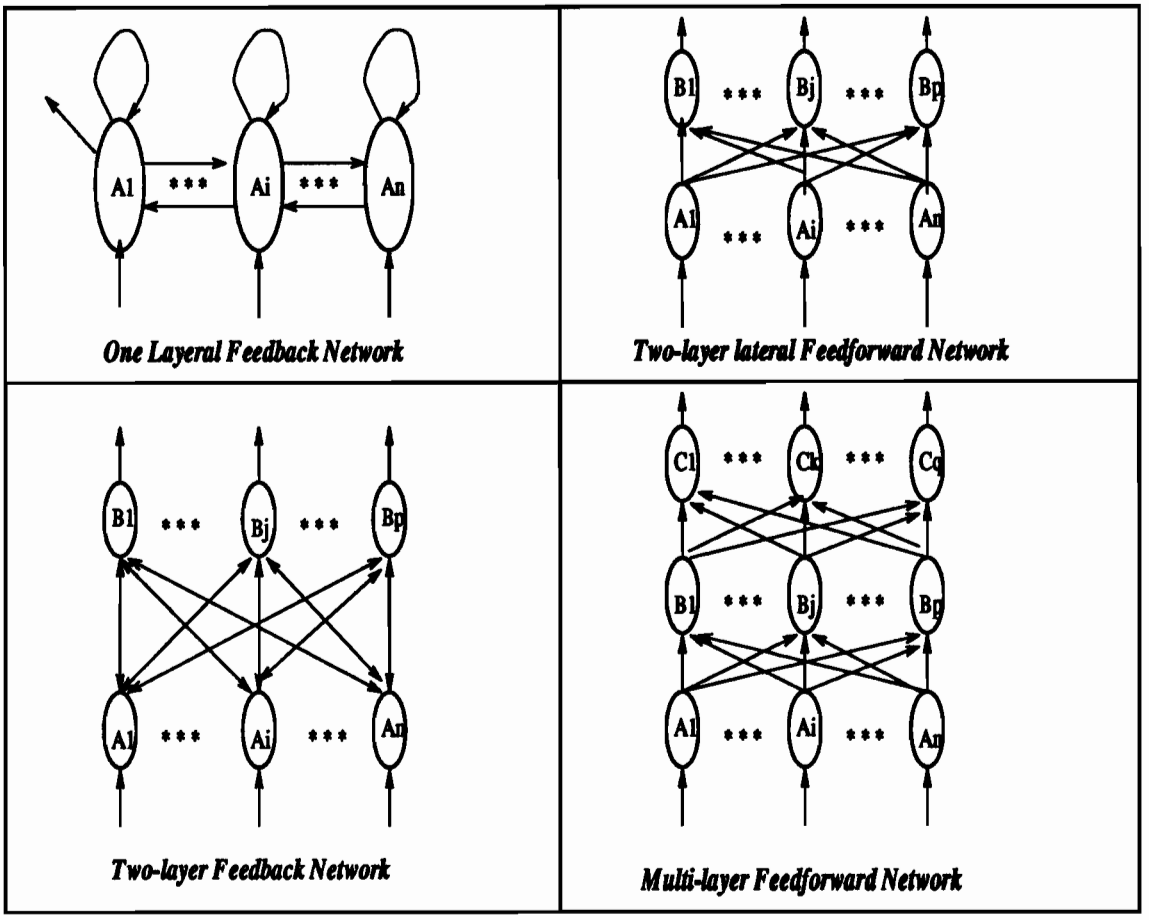


Figure 3.2: Neural Network's Topologies

2. **Reinforcement procedures:**

   Requires a single scalar evaluation of the output.

3. **Unsupervised procedures:**

   Incorporates no external teacher and construct internal models to capture regularities in their input vectors.

These three learning procedures are further divided as followings[Hint 87].

1. *Associative memories without hidden units*

   - Characteristics:

     The aim is to store a set of associations between input vectors and output vectors by modifying the weights on the connections.

   - Linear associators: **Hebbian procedure.**

     Eq. 1 defines **Herrian procedure** that the state of an output unit is a linear function of the total input that it receives from the input units.

   - Non-linear associative nets: **Hopfield nets.**

     Perform perfect recall for the set of associations which have non-orthogonal input vectors.

     Hopfield's greatest contribution is his "energy function". Hopfield nets store vectors whose components are all +1 or -1 using the simple storage procedure described in Eq. 3.

$$\Delta w_{ji} = y_i y_j \qquad (3)$$

     The behavior of the network is governed by the global energy function

$$E = -\sum_{i<j} s_i s_j w_{ij} + \sum_j s_j \theta_j \qquad (4)$$

     where s_i and s_j are the states of two units.

$$\Delta E_j = E(s_j = -1) - E(s_j = +1) = -\theta_j + \sum_i s_i w_{ij} \qquad (5)$$

The weights define an "energy landscape" over global states of the network and the stored vectors are local minima in this landscape. The retrieval process consists of moving downhill from a starting point to a nearby local minimum.

- Deficiencies:

  The deficiencies of associators without hidden units is that Linear independent does not hold for most tasks that can be characterized as mapping input vectors to output vectors.

2. Simple supervised learning procedures

- Characteristics:

  The input units are directly connected to output units whose states are a continuous smooth function of their total input. A measure of how poorly the network is performing with its current set of weights is:

$$E = \frac{1}{2}\sum_{j,c}(y_{j,c} - d_{j,c})^2 \qquad (6)$$

  where $y_{j,c}$ is the actual state of output unit j in input-output case c, and $d_{j,c}$ is its desired state.

- Error measurement method:

  The error measure given in Eq. 6 can be minimized by starting with any set of weights and repeatly changing each weight by an amount proportional to $\partial E/\partial w$.

$$\Delta w_{ji} = -\epsilon\frac{\partial E}{\partial w_{ji}} \qquad (7)$$

  In the limit, as ϵ tends to 0 and the number of updates tends to infinity, this learning procedure is guaranteed to find the set of weights that gives the least squared error.

$$\frac{\partial E}{\partial w_{ji}} = \sum_{cases} \frac{\partial E}{\partial y_j} \frac{\partial y_j}{\partial x_j} \frac{\partial x_j}{\partial w_{ji}} = \sum_{cases} (y_j - d_j) \frac{dy_j}{dx_j} y_i \qquad (8)$$

- deficiencies:

  The major deficiency is that the mappings between input and output vectors can not be captured by any combination of weights in such simple networks, so the guarantee that the learning procedure will find the best possible combination of weights is of little value.

  Another deficiency is that gradient descent may be very slow.

3. Back-propagation: A multilayer least squares procedure

- Characteristics:

  The "back-propagation" learning procedure [Rume 86] is a generalization of the least squares procedure that works for networks which have layers of hidden units between the input and output units.

  In a multilayer network it is possible, using Eq. 8. to compute $\partial E/\partial w_{ji}$ for all the weights in the network provided we can compute $\partial E/\partial y_j$ for all the units that have modifiable incoming weights.

  The main idea of back-propagation is that these derivatives can be computed efficiently by starting with the output layer and working backwards through the layers. For each input-output case, c, first a forward pass is used, starting at the input units, to compute the activity levels of all the units in the network. Then a backward pass is followed, starting at the output units, to compute $\partial E/\partial y_j$ for all the hidden units. For a hidden unit, j, in layer J the only way it can affect the error is via its effects on the units, k, in the next layer, K (assuming units is connected in feedforward mode).

$$\frac{\partial E}{\partial y_j} = \sum_k \frac{\partial E}{\partial y_k} \frac{\partial dy_k}{\partial dx_k} \frac{\partial dx_k}{\partial dy_j} = \sum_k \frac{\partial E}{\partial y_k} \frac{\partial dy_k}{\partial dx_k} w_{kj} \qquad (9)$$

- Applications:

  The back-propagation net can be applied in many areas, such as discovering semantic features, mapping text to speech, phoneme recognition, and iterative networks. This project is employed back-propagation.

  The self-supervised back-propagation used to construct compact codes resembles the use of principal components analysis to perform dimensionality reduction, but is has the advantage that it allows the code to be a non-linear transform of the input vector.

- Deficiencies:

  The major deficiencies are the slow convergent speed and non global minima. Back-propagation is inadequate for large tasks because the learning time scales poorly. Empirically, the learning time on a serial machine is very approximately $O(N^3)$ where N is the number of weights in the network. On a parallel machine that used a separate processor for each connection, the time would be reduced to approximately $O(N^2)$.

4. **Boltzmann machines**

   A **Boltzmann machine** is a generalization of a **Hopfield net** in which the units update their states according to a *stochastic* decision rule. The units have states of 1 or 0, and the probability that unit j adopts the state 1 is given by

$$P_j = \frac{1}{1 + e^{-\Delta E_j / T}} \qquad (10)$$

   where $\Delta E_j = x_j$ is the total input received by the j^{th} unit and T is the "temperature". It can be shown that if this rule is applied repeatly to the units, the network will reach "thermal equilibrium".

   The Boltzmann machine learning procedure is based on the simplicity of the expression for the derivative of the asymmetric divergence between the conditional probability distribution exhibited by the output units of a Boltzmann machine and a desired conditional probability distribution.

5. **Unsupervised Hebbian learning**

   A unit can develop selectivity to certain kinds of features in its ensemble of input vectors by using a simple weight modification procedure that depends on the correlation between the activity of the unit and the activity on each of its input lines. This is called "Hebbian" learning rule because the weight modification depends on both pre-synaptic and post-synaptic activity.

6. **Competitive learning**

   - Characteristics:

     Competitive learning is an unsupervised procedure that divides a set of input vectors into a number of disjoint clusters in such a way that the input vectors within each cluster are all similar to one another. It is called competitive learning because hidden units compete with one another to become active.

   - Application:

     One major theme has been to show that competitive learning can produce topographic maps[Koho 82]. The hidden units are laid out in a spatial structure and instead of just updating the weight vector of the hidden unit that receives the greatest total input, the procedure also updates the weight vectors of adjacent hidden units.

   - Competitive learning and back-propagation:

     The relationship between competitive learning and back-propagation is that: back-propagation is needed in order to compute the error derivatives of the weights from the input units to the hidden units, but the winner-take-all non-linearity makes back-propagation unnecessary in this network because all these derivatives are equal to zero. Since the error derivatives are so simple, the learning can be done without output units and output weights. Therefore no constraints for the input and output weights of a

hidden unit to be identical. Thus the simplified version of competitive learning is a degenerate case of self-supervised back-propagation.

7. **Reinforcement learning procedures**

   A main idea in many reinforcement learning procedures is that assigning a credit to a local decision to measure how it correlates with the global reinforcement signal.

   Reforcement learning's main advantage is easy to implement. The main disadvantage is very inefficient when there are more than a few local variables, the other disadvantage is that gradient ascent may get stuck in local optima.

   "Associative Reward-Penalty" or A_{R-P} uses stochastic units like those in Boltzmann machine(Eq. 10). They prove that if the input vectors are linearly independent and the network only contains one unit, A_{R-P} finds the optimal values of the weights.

The common problem for various learning procedures is the slow speed of learning, particularly procedures that construct complicated internal representations. Using optimization methods such as recursive least squares or using dedicated hardware for each connection may speed them up,

3.2 Neural Network Simulator Xerion

3.2.1 What Is Xerion

Xerion is a neural network simulator. It is developed by the connectionist group at the University of Toronto. The current version 3.1 contains libraries of routines for building networks. A simulator written using **Xerion** libraries has access to a command line interface with built in commands for creating and training networks, examining and modifying data structures, redirecting output using streams, as well

as miscellaneous utilities. **Xerion** also has a graphical interface for displaying network activations on specified examples, and for checking connection weights between units. As well it contains an optimization package which can train nets using several different methods including conjugate gradient. It is written in C and uses the X window system to do the graphics.

Since there are many different kinds of neural network simulators applied in different paradigms. **Xerion** has been used to build several simulators, such as *Back Propagation, Recurrent Back Propagation, Boltzmann Machine, Mean Field Theory, Free Energy Manipulation, Kohonnen Net, Hard and Soft Competitive Learning.*

The following platforms can run **Xerion**:

- SGI Personal Iris and SGI 4d running IRIX 4.0.1.
- Sun 4 running SunOS 4.1.2.
- Sun 3 running SunOS 4.0.3.
- DEC 5000 running ULTRIX.
- DEC Alpha running OSF/1.
- HP 730 running HP-UX 8.07.
- MIT X11R4 and X11R5(libraries: Xaw, Xmu, Xt, X11).

3.2.2 Install Xerion for Free

1. Get free source files of **Xerion**

   The **Xerion** source files can be obtained from internet networks by *ftp.*

   ftp ftp@cs.uotoron.edu

   cd /pub/xerion

   Ftp all the necessary source files below:

   One more file man-3.1.tar.Z contains pre-formatted man pages for SYSV systems that do not ship nroff.

SIMULATOR		ASSOCIATED TAR FILE
bm-3.1.tar.Z	-	Boltzmann Machine Simulator
bp-3.1.tar.Z	-	Backprop Simulator
cascor-3.1.tar.Z	-	Cascade Correlation Simulator
fem-3.1.tar.Z	-	Free Energy Manipulation Simulator
hcl-3.1.tar.Z	-	Hard Competitive Learning Simulator
kcl-3.1.tar.Z	-	Kohonen Learning Simulator
mft-3.1.tar.Z	-	Mean Field Theory Simulator
rbp-3.1.tar.Z	-	Recurrent Backprop Simulator
scl-3.1.tar.Z	-	Soft Competitive Backprop Simulator

2. Building **Xerion**

For those simulators you want to build, untar their associated files, then check whether the environment variable *$BINTYPE* is set up properly, if not, check with the person who installed it to see if there are any site-specific setups you must do, and then run *make* procedure; otherwise, type *make* after make sure that *imake* and *xmkmf* have been already installed on your system because **Xerion** uses *Imakefiles.*

For example: Build *Boltzmann Machine* simulator and *Backprop* simulator using **double precision** in your application instead of **single precision.**

First of all, untar the required simulator files:

unix> zcat xerion-3.1.tar.Z | tar xvf -

unix> zcat bp-3.1.tar.Z | tar xvf -

unix> zcat bm-3.1.tar.Z | tar xvf -

Second, in original *xerion/config/xerion.cf* file, you may make the following changes when you install Xerion on your system:

change #define BinDest $(XERIONDIR)/bin/$(BINTYPE)
to #define BinDest $(XERIONDIR)/bin
change #define LibDest $(XERIONDIR)/lib/$(BINTYPE)
to #define LibDest $(XERIONDIR)/lib
change #define Defines -D_BSD_COMPAT $(PW_DEFINES)
to #define Defines -D_BSD_COMPAT -DDOUBLE $(PW_DEFINES)

and then, do the compile:

unix> make

3. Running **Xerion**

First: Add directories and environment variables to your search path.

- In your .login (csh) insert the following lines:
  setenv XERIONDIR <directory>
  source $XERIONDIR/config/setup.csh
  where <directory> is the place that the xerion config directory (as well as the man, doc, templates and nets directories) resides.
- If you use sh or ksh, then put the following in your .profile:
  XERIONDIR=<directory> ; export XERIONDIR
  . $XERIONDIR/config/setup.sh

And then: Run and exit the simulator:

Once your environment is configured, the *Back Propagation* module can be started from a UNIX prompt by typing the command:

unix> bp

or

unix> run bp # run in its own window
bp→ quit # quit or exit the simulator

4. Online help

Xerion provides two ways for getting online help. *sman* for **Unix** man pages and *help help* for inside the simulator.

unix> sman sman

or

bp→ help help

5. Requests and Questions

   For any further questions or suggestions, send mail to xerion-request@ai.toronto.edu or xerion@ai.toronto.edu.

3.2.3 Build Neural Network Architecture

Building a neural network in **Xerion** involves in creating *objects* and connecting them together in certain structure.

There are three levels of objects:

- The lowest level objects are **Units** and **Links**.

  A **Unit** is an object that has an input, an output and a function for transferring the input to the output. A **Link** can connect two Units together. Links have modifiable connection weights, derivatives for the weights, and a function for calculating these derivatives. i.e., input Units, hidden Units and output Units. Links from the input Units to hidden Unit, or from the hidden Units to output Units.

- The middle level object is the **Group**.

  A **Group** is a set of similar Units. All Units in a Group have the same method of activition. i.e., INPUT Group and OUTPUT Group.

- The highest level object is a **Net**.

  Nets contain a set of Groups and a list of all the Links connecting the Units in the Groups. i.e., **ExampleSets** are lists of examples(input and target vectors) used to train, or test the network.

There are many commands to build, examine, and set the variables of the current net, such as *addNet, useNet, show currentNet, randomize, set currentNet.weightCost = 0.001* etc.. There are also some commands for adding examples to the net, training the net by the command *minimize* with different parameters, freezing weights, and saving or restoring the Nets. For more detail, refer to [Xeri 93].

The neural network architecture for classifying brain tumour can be an example stated in *Tumour_net_run.in* in Appendix B.

3.2.4 Graphical User Interface for Xerion

Xerion has a very good graphical user interface with seven display windows for users to view or modify the network, learning methods, or simulator variables.

Figure 3.3 shows the **Xerion** main window.

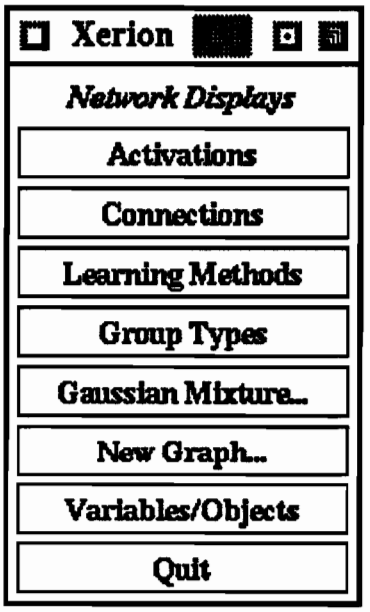


Figure 3.3: **Xerion** Main window

The seven displays and Quit option explained here :

1. **Activation Display:**

   View the activitions of the network and process individual train, test or validation examples. Figure 3.4 shows the **Xerion** Activition display which is

designed for brain tumour diagnosis.

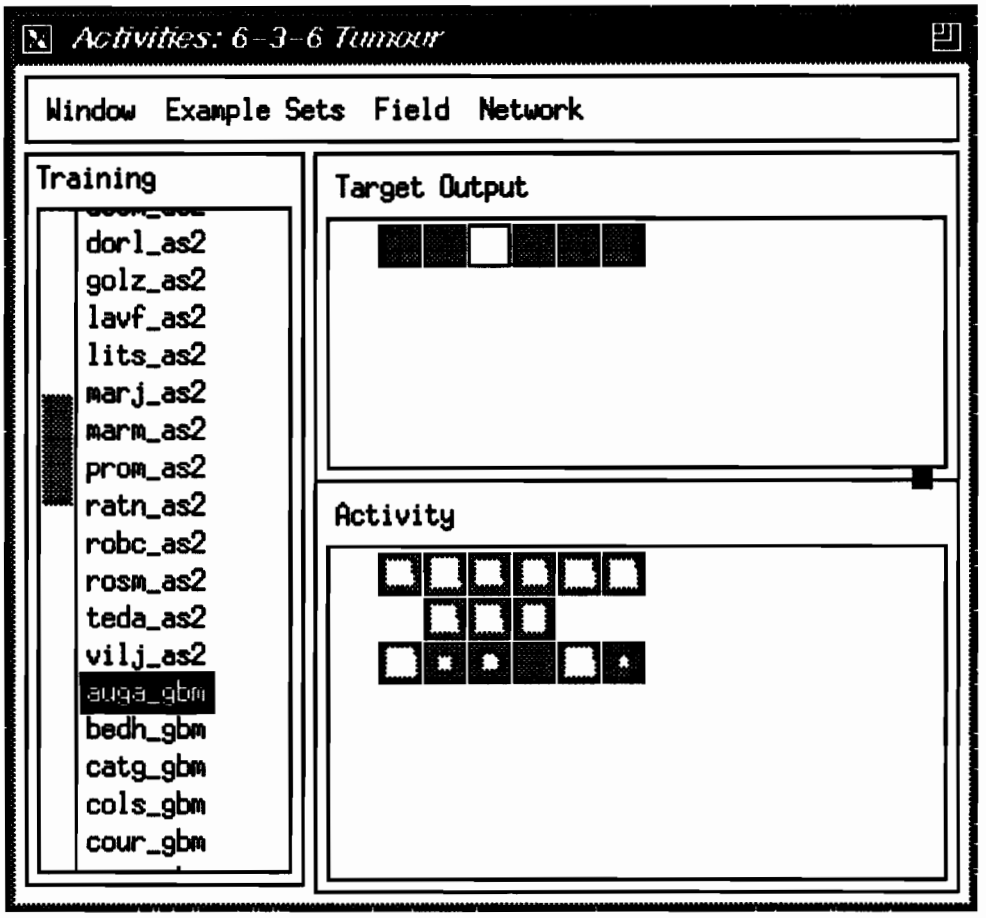


Figure 3.4: **Xerion** Activition Display

2. **Connection Display:**

   View the connection weights of the network. Figure 3.5 shows the **Xerion** Connection display designed for this project.

3. **Learning Methods:**

   Select the learning method and set parameters that affect it. Figure 3.6 shows the **Xerion** Learning Methods display.

4. **Group Types:**

   Change the activition functions, error measures, and cost models of groups of units. Figure 3.7 shows the **Xerion** Group Types display.

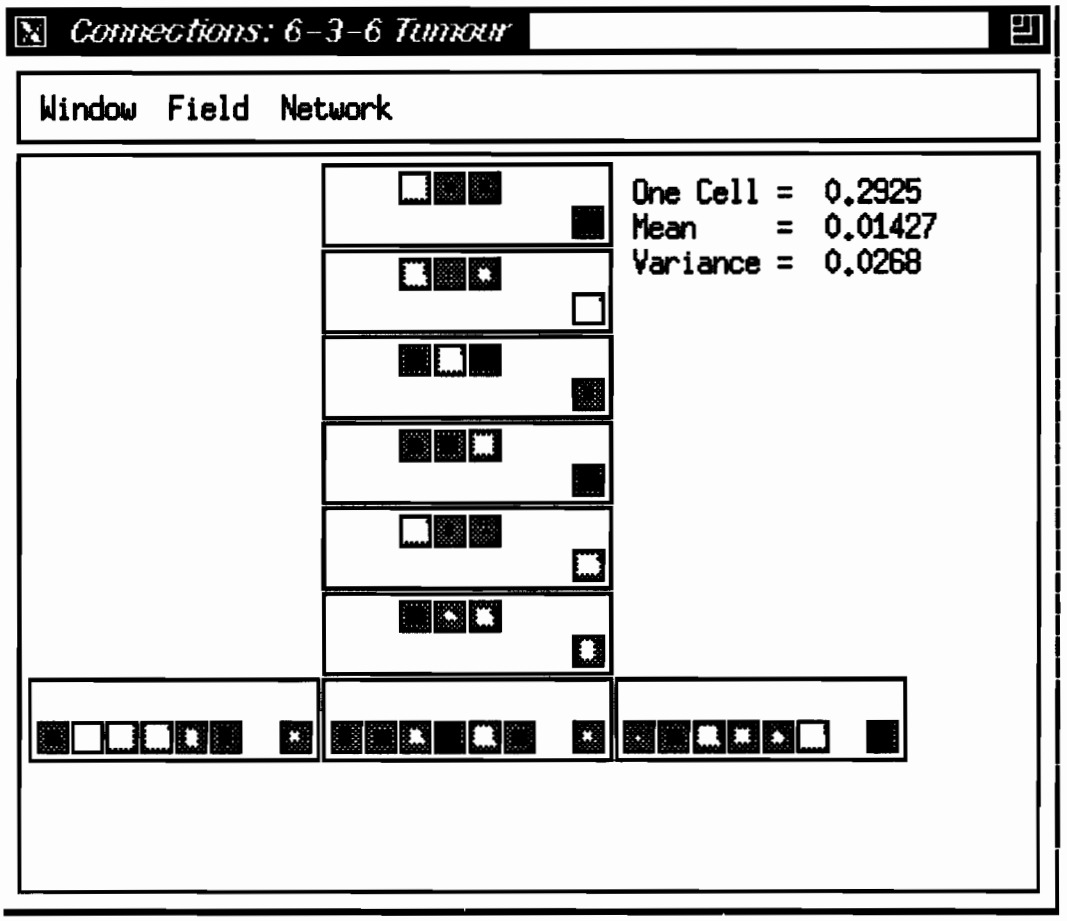


Figure 3.5: **Xerion** Connection Display

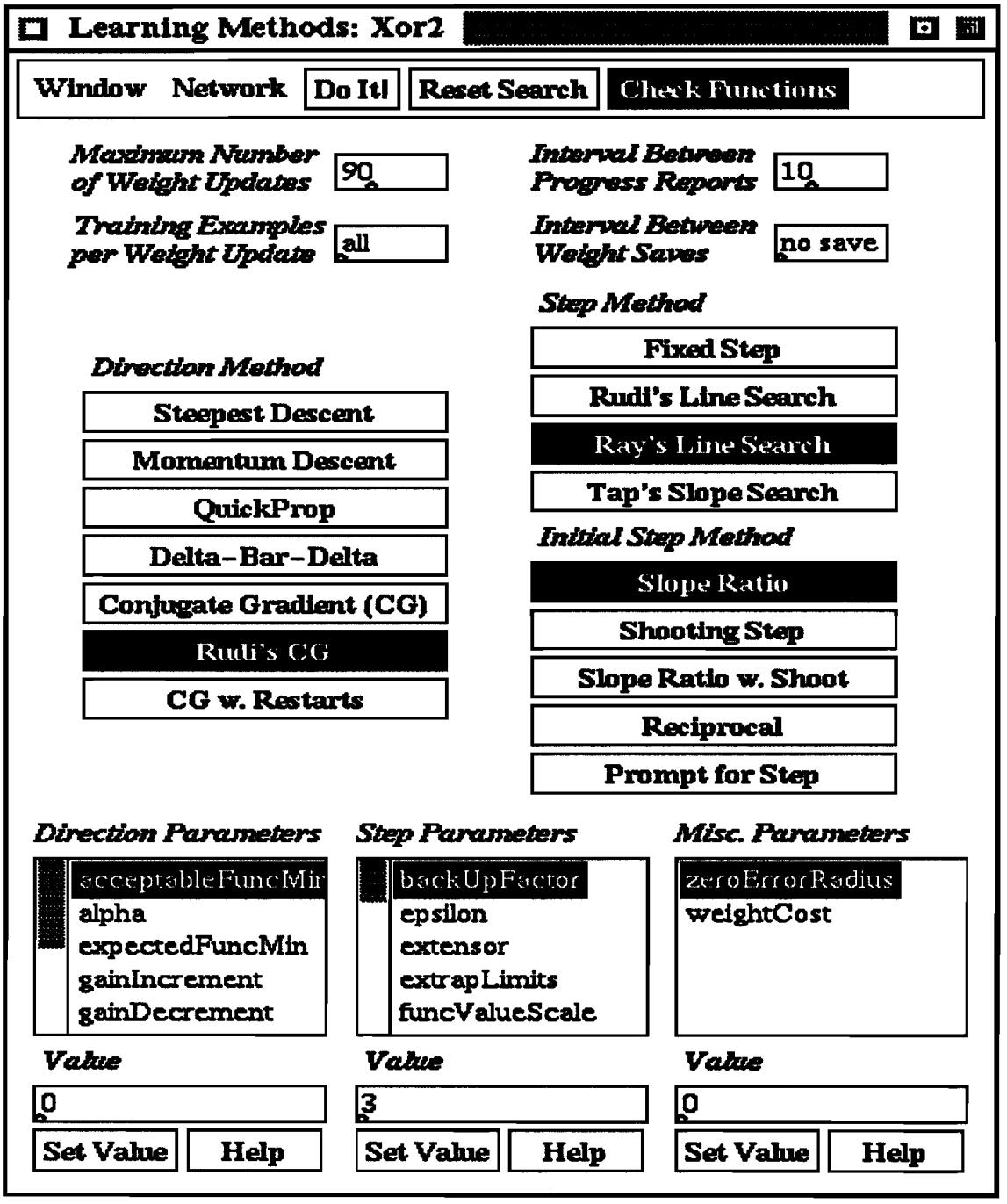


Figure 3.6: **Xerion** Learning Methods Display

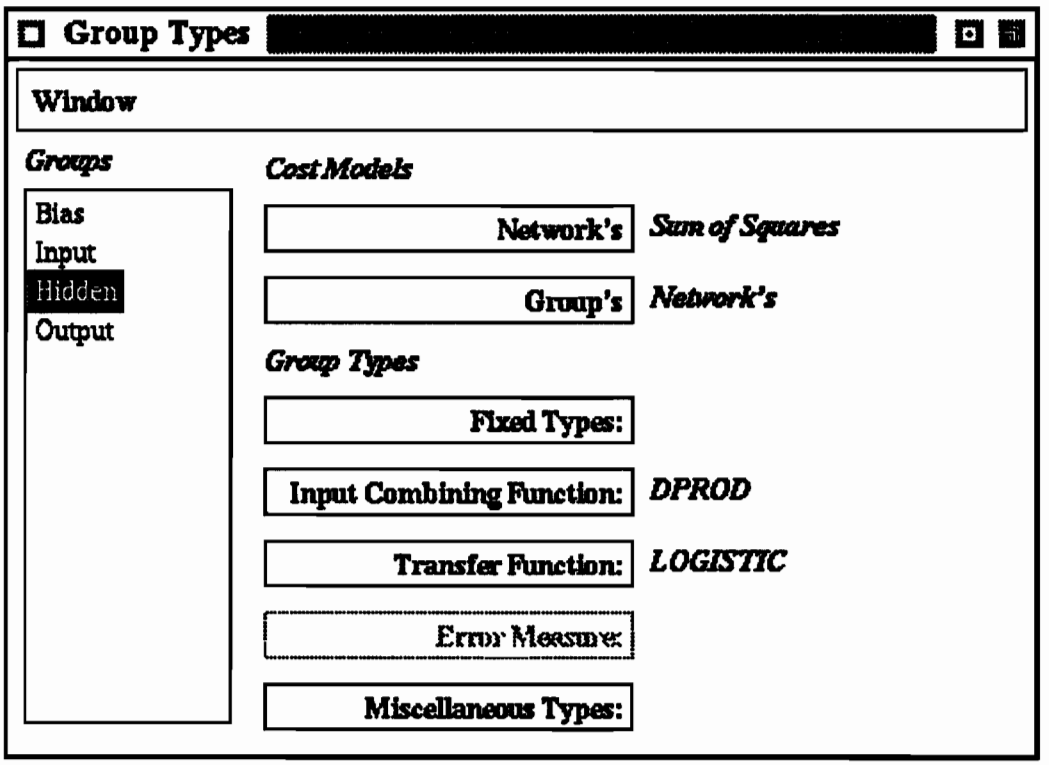


Figure 3.7: **Xerion** Group Types Display

5. **Gaussian Mixture:**

   View a mixture of Gaussians cost model(used in soft weight sharing) as it changes during training. Figure 3.8 shows the **Xerion** Gaussian Mixture display.

6. **Graphs:**

   Plot values as they change through time. (usually during training to view the currentNet.error). See Figure 3.9.

7. **Variables/Objects:**

   View and modify a list of system variables and objects used by the simulator. see Figure 3.10.

8. **Quit:**

   Exit the simulator.

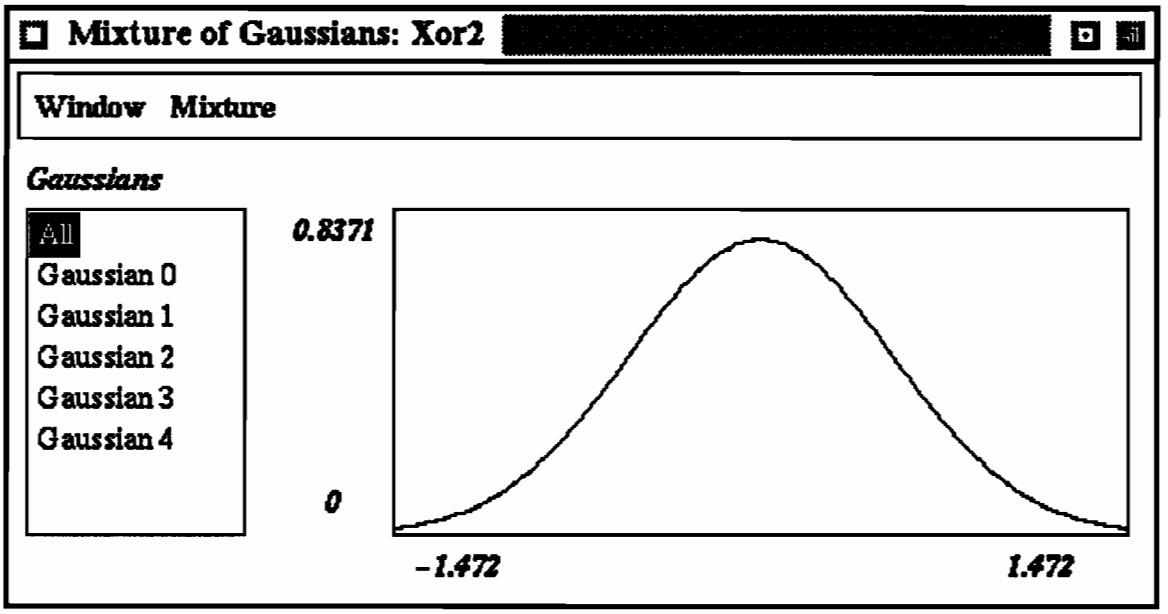


Figure 3.8: **Xerion** Gaussian Mixture Display

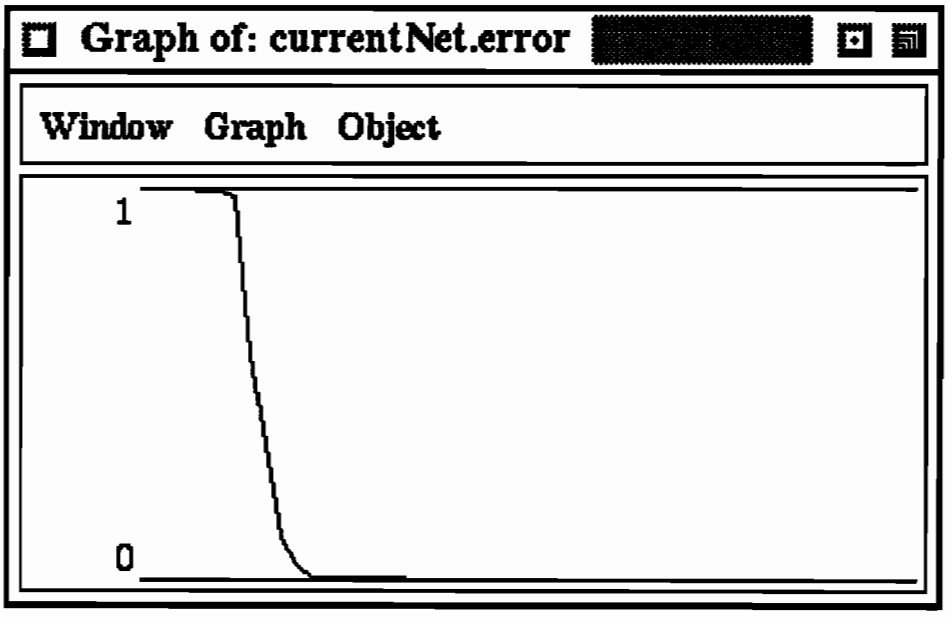


Figure 3.9: **Xerion** Graphs Display

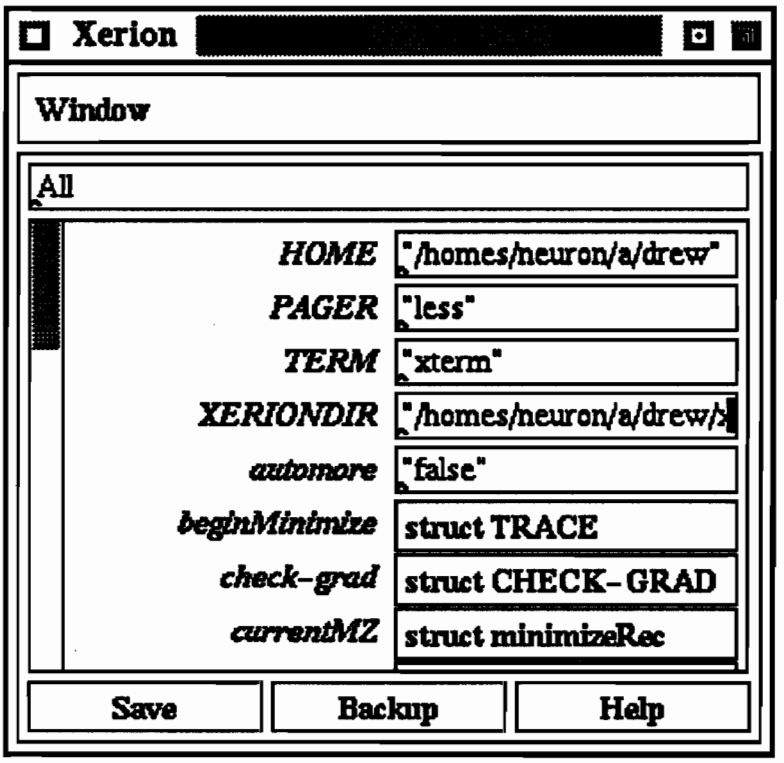


Figure 3.10: **Xerion** Variables/Objects Display

Generally **Activition, Connection, Learning Methods and Graphs displays** are used more often than the rest of displays, So more detailed information are described below:

- *Activation Display*:

  In Figure **Activition**, There are pull-down menu bar area, ExampleSets list area, Targets output area, and Activity area.

  In the pull-down menu area, The **Window** menu contains the buttons for refreshing the display, selecting updates, hiding Target output area, and closing the display. The **Example Sets** menu is for selecting one of the train, test and validation example sets to be displayed in the ExampleSets area. The **Field** menu is for deciding which field of the units is displayed. The **Network** menu can be used to change the properties of the network dispay.

  In the ExampleSets area, by clicking the mouse on one of the examples in

the list or using the up and down arrow key(↑ and ↓) to select the individual example of training, testing or validating.

The Activity and Target output areas use Hinton diagrams to show both the network activitions and the target values for the selected example. Since each unit with positive or negative values is represented by a small square with white and black respectively, so by comparing the example's activition output in Activition area with its target output in Target output area, if they match, the network's performance is very good, otherwise, it's poor.

The left mouse button is used to check the unit's value; the middle or right mouse are used for slightly decreasing or increasing the input units' value respectively. These functions are use to introduce some noisy into the input data and observe how the network performance changes.

- *Connection Display*:

  The neural network structure layout and connections for both **Activition display** and **Connection display**is set up in script file *Tumour_net.layout* in Appendix B.

  In **Connection** display, the weights and linkage can be viewed by clicking any of the small square.

- *Learning Methods*:

  Many kinds of learning methods and associated step methods, initial step methods have been set up in this display. Such as *steepest descent, momentum descent, quickprop, delta-bar-delta, conjugate gradient(CG), rudi's CG and CG with restarts* learning methods; *fixed step, rudi's line search, ray's line search and tap's slope search* step methods; and *slope ratio, shooting step, slope ratio with shoot, reciprocal and prompt for step* initial step methods. For brain tumour classification problem, the *Rudi's CG* learning method which comes with *Ray's line search* step method and *Slope ratio* initial step method is employed.

In this display, the parameters that affect the chosen method also can be modified. It is the graphical equivalent of the *minimize* command.

To start training the network, click the **Do It!** button. If the minimize routine gets stuck, restart it using the **Reset Search** option. If any changes of the minimization or network parameters are made, *must* reset the search before continuing the minimization. Otherwise, the search will terminate prematurely.

- *Graphs*:

  Graphs display is a very useful tool for viewing particular variable especially the *currentNet.error*'s changes through time. When the curve goes down to zero, that indicates the perfect training has been reached; otherwise the training will be continued until certain constrains being reached or **interrupt** button is clicked.

3.2.5 Xerion's Data Structure

In **Xerion**, the data structure mainly includes *NetRec, ExampleSetRec, ExampleRec, GroupRec, UnitRec, LinkRec, minimizeRec* and some extension records. as shown in Figure 3.11.

The various pointers and other objects in the network can be examined all the way down to individual links, and arbitrary variable's value can be traced through the pointers.

3.3 Neural Network Data Format

In general, the neural network's input units with values between 0 and 1 inclusively, therefore the sample data extracted from the spectra image have to be constrained into [0,1] interval.

The algorithm is very simple, find out the biggest value in whole data file, and then each sample value divided by it. The detailed source codes *Tumour_nn_data.c*

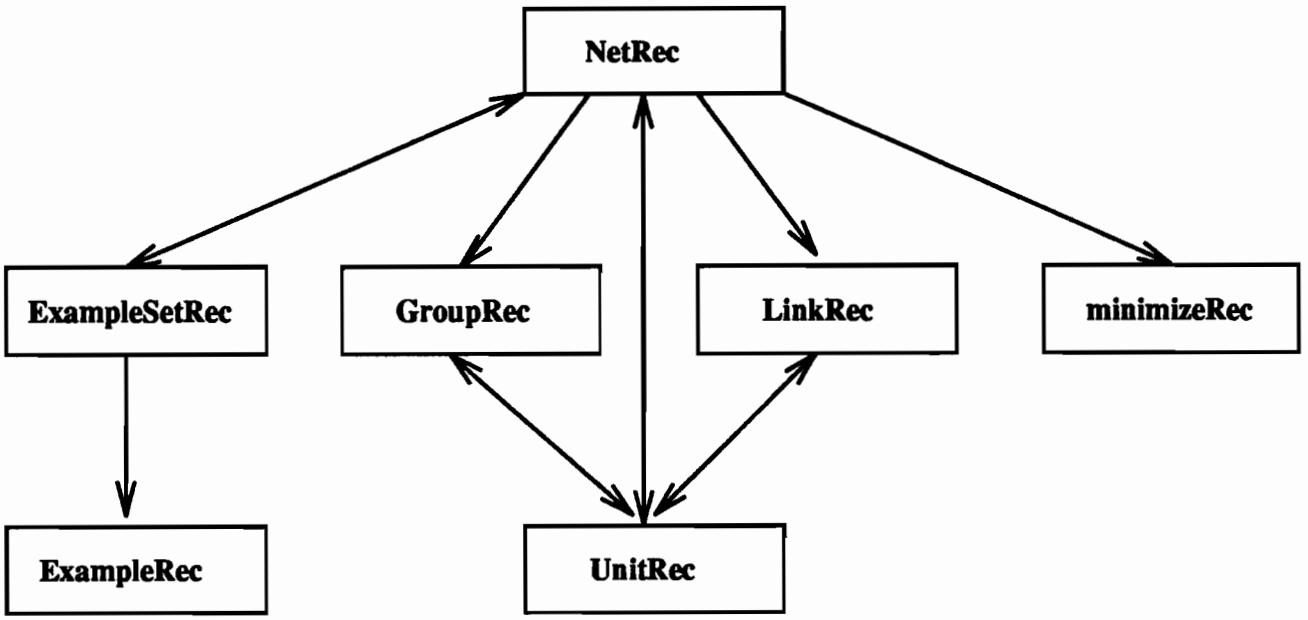


Figure 3.11: Data Structures and Linkages in **Xerion**

are in Appendix C.

3.4 Neural Network Architecture

The neural network for brain tumour diagnosis is built with backpropagation feedforward paradigm by three layers. Input layer consists of six Units represented six extracted feature values; output layer has also six Units for six classified tumour types; and hidden layer in between contains three Units. The detailed script source codes *Tumour_net_run.in* for building this neural net is stated in Appendix B.

Why do we design the network structure like this? Here is the theoretical foundations for approximation properties of multilayer feedforward networks:

The theory has proven that multilayer feedforward networks with fixed nonlinear node functions and linear combinations of inputs to nodes can approximate many input-output mappings with very small error [Anto 91].

Four increasingly complex classes of functions are listed here:

1. Boolean mappings ($\{0,1\}^N \rightarrow \{0,1\}$):

A network of linear threshold units with one hidder layer of 2^N can represent exactly this boolean mappings.

2. Classification of real vectors ($R^N \rightarrow \{0,1\}$):
   The networks having two hidden layers of threshold elements can perform this classification. The first hidden layer is used to separate the input space, and the second hidden layer is used to compute intersections of the first layer units.

3. Continuous mappings from real vectors to real values ($R \rightarrow R$):
   For the case of networks with sigmoid nonlinearities(defines as $f(x) = \frac{1}{1+e^{-x}}$), here is the theorem:
   Networks with one layer of hidden units can uniformly approximate (to within any $\epsilon > 0$) any real continuous function on a finite real internal.

4. Continuous mappings from real vectors to real values ($R^N \rightarrow R$):
   Again for sigmoid nonlinearities, and the theorem is:
   Networks with two layers of hidden units can uniformly approximate (to within any $\epsilon > 0$) any real continuous function of multiple real inputs.

Since brain tumour diagnosis problem matches the third case, so its neural network architecture is set up based on the theorem.

3.5 Create Neural Network Data Sets

In addition to the Clean data file, From the feature extraction program, several data files for Noisy data are produced, *.peak* file, *.area* file, *LSTpeak, LSTarea* file, and *AVEpeak, AVEarea* file. All these data files are formatted to [0,1] intervals for the neural network.

The program *Tumour_set_test.c* in Appendix C is developed for creating the training and testing data set automatically. The algorithm is:

- Enter the data file name. The training and testing data set types are based on the data file name with associated type being *.peak, area, LSTpeak, LSTarea, AVEpeak,* or *AVEarea.*

- Enter the testing data set identifier which you want to get.
  If "all" is entered, then all the samples go to the testing data set.
  If "none" is entered, then all the samples go to the training data set.
  If patient identifier(i.e. "mym") is entered, then the samples with "mym" identifiers go to the testing data set and the rest become training data set.
  If voxel number(i.e. "564" in data file "mym") is entered, then the voxel sample "564" of "mym" goes to the testing data set and the rest voxel samples of "mym"become training data set.

Associated to Figure 3.12, the definitions for different types of data sets are as followings:

1. The data sets include *Clean data set* and *Noisy data set.* If it is not specified to Clean data set, it indicates Noisy data set.

   Clean data : a group of patients' averaged peaks data calculated manually.

   Noisy data : a group of patients' feature data calculated by computer.

2. The noisy data sets have two groups, the *peaks data set* and *areas data set.*

   Peak data : The feature data are represented by the peaks' values.

   Area data : The feature data are represented by the areas' values.

3. For each group of the noisy data set, there are four types :

   List : a group of patients' individual voxel feature data.

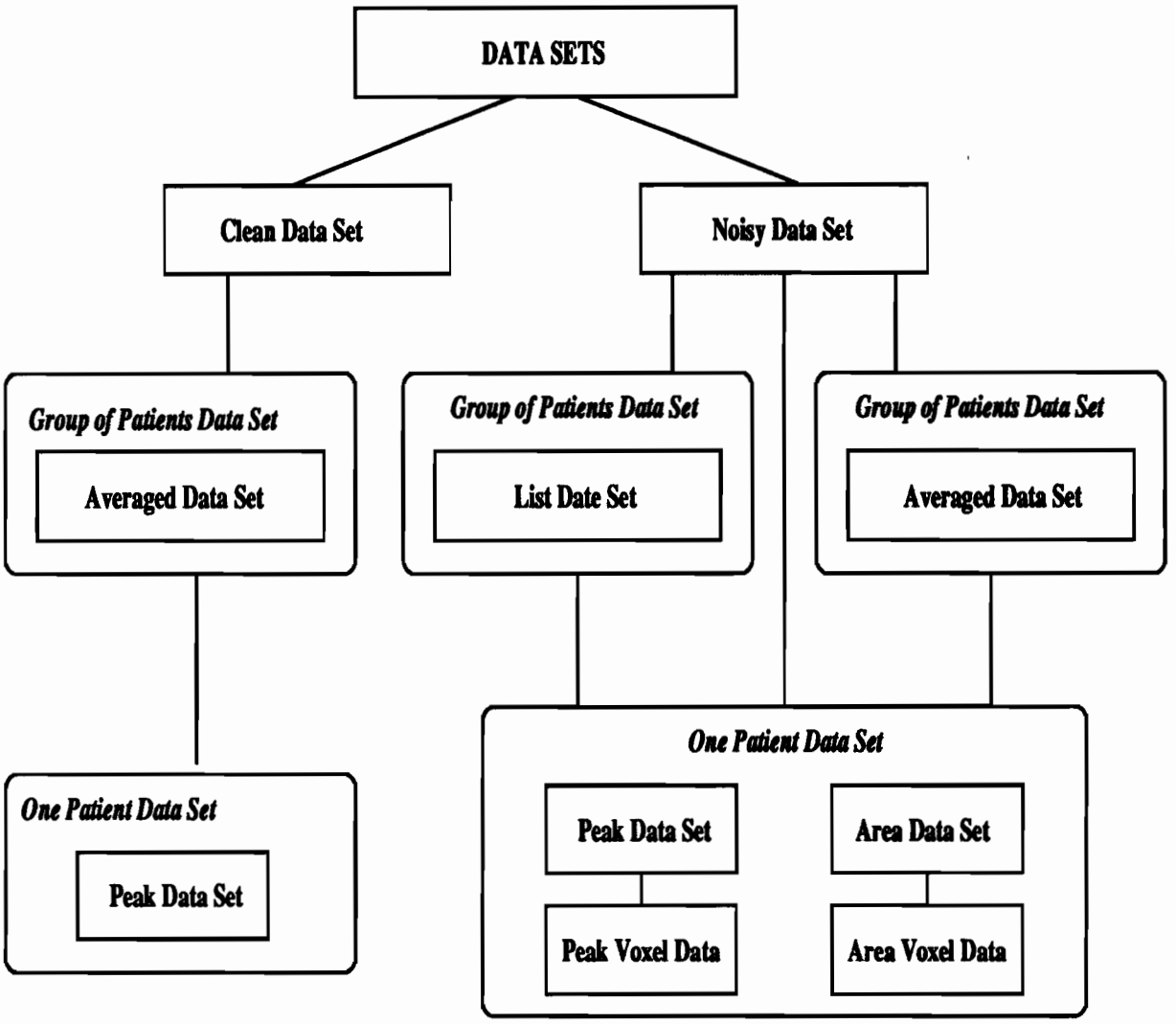


Figure 3.12: Data Set Types for Brain Tumour Diagnosis

Average : a group of patients' feature data which are average tumour and normal feature values for each patient.

Patient data : either List data or Average data of a patient.

Voxel data : the individual voxel data of a patient.

By the definitions above, if we say *Average Peak data*, that means the noisy data set which contains the averaged peaks feature values.Some examples of data sets are in Appendix D.

3.6 Train and Test Neural Network

There are two ways can be applied in training and testing neural network. One is using command *minimize* combining different parameters which include initializing nets variables, setting learning method and step method and so on. The other most recommended way is using **Xerion**'s seven display windows described in Section 2.

The neural network on automatic brain tumour diagnosis can be performed by using **Learning methods** display for training network, using **Graphs** display for controlling the error measurement (*currentNet.error*) to the lowest point; and using **Activation** display for checking whether the training data set well trained and the diagnosis results for the testing data set by comparing the activition output with its target output. Figure 3.13 illustrates the successful diagnosis and Figure 3.14 shows the misdiagnosis.

More detailed steps are performed for training and testing described below:

1. Loading the neural network architecture and its layout files.

   After these are read, In **Activition** display, activition area shows three layers with number of Units in each layer. From bottom to the top are input layer, hidden layer and output layer. The linkages and weights are shown in **Connection** display.

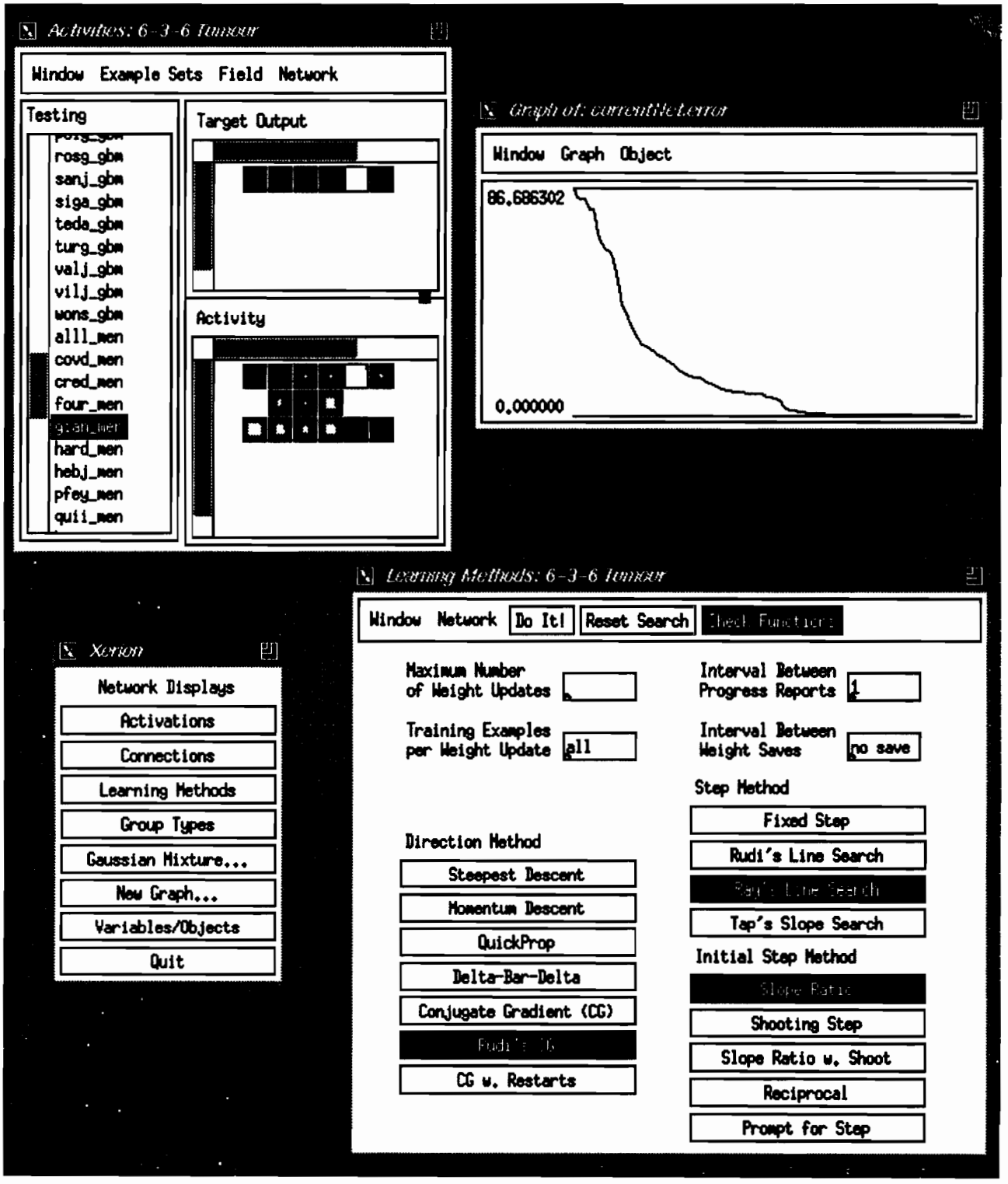


Figure 3.13: Successful Diagnosis

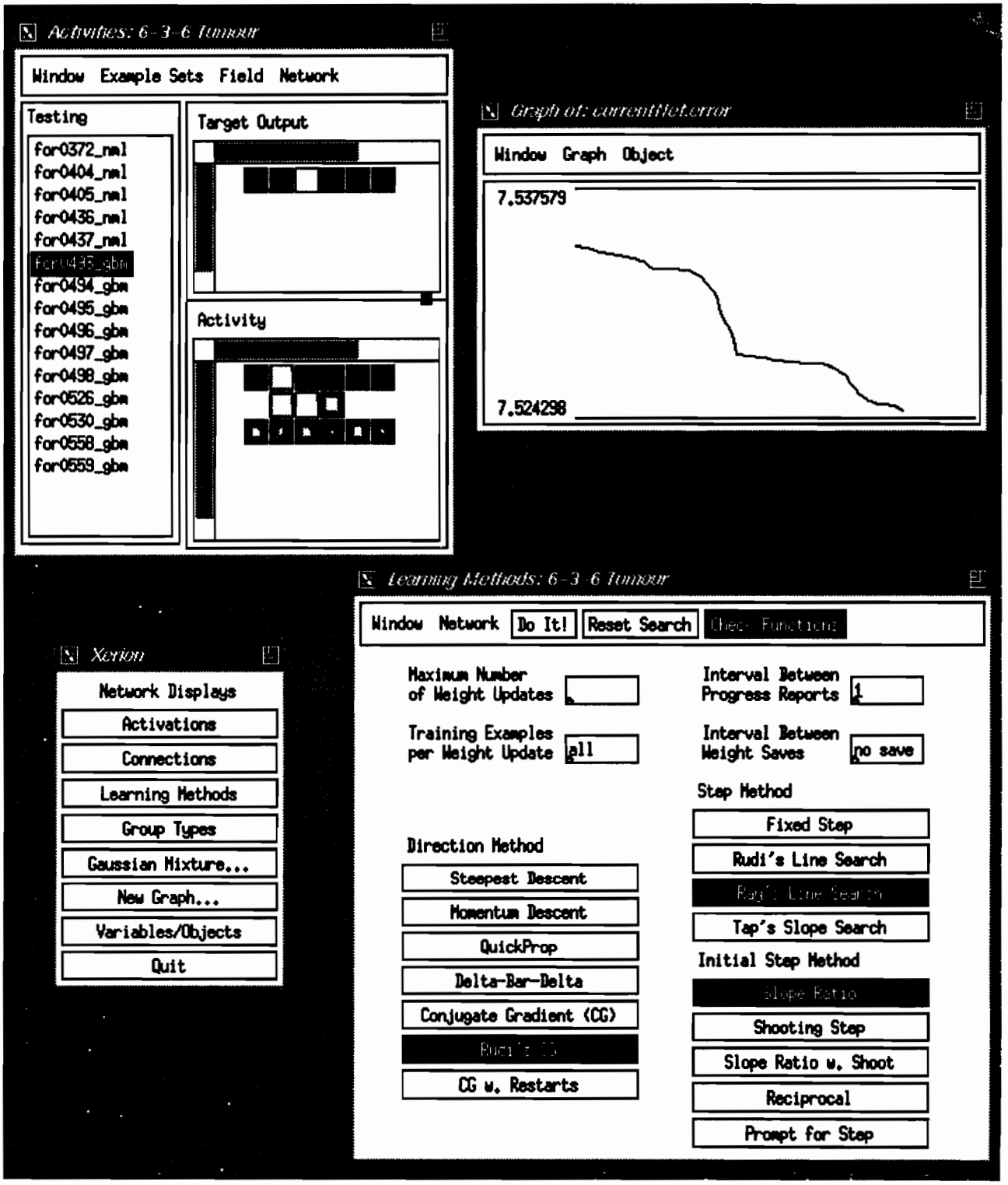


Figure 3.14: Misdiagnosis

2. Add examples to training and testing data sets.

   The examples of training or testing data set are listed in ExampleSet area in **Activition** display.

3. Open the **Graphs** display with being viewed *CurrentNet.error* variable.

4. Open **Learning Method** display and set up learning method.

   The *Rudi's CG* learning method which comes with *Ray's line search* step method and *Slope ratio* initial step method is employed in this project.

5. Click **"Do It!"** in **Learning Method** display.

6. Stop training.

   Click **"Interrupt"** button when the curve in **Graphs** display goes down to zero or lowest point through the training process.

7. Switch to test set

   Selecting **test** from **ExampleSet** menu in **Activition** display.

8. Examine the diagnosis results intuitively

   Pressing ↓ or ↑ key while comparing the match between the actual output in activition area and target output from target output area in **Activition** dispay.

9. Train another set.

   Click **"Reset Search"** button before the next training is started.

3.7 Trace Neural Network Information

Since **Xerion** has provided the commands for examining and modifying the variables, so the script files *runExamples, printAexample* and *printTraceInfo* in Appendix B are created to trace the accurate diagnosis information for the neural network. By given a value called *examine bound*, these routines will check all the diagnosis results and find out those "failure" samples if any output Unit whose absolute difference

between actual output and target output is more than *examine bound*. Obviously, the smaller the examine bound, the more the "failure" samples. Since the "failure" output here has not considered the whole probabilities for each tumour type, so it will be further analyzed in statistical analysis part to decide if it is real "failure" or not.

Chapter 4

Statistical Analysis

4.1 Introduction

The statistical analysis is based on the traced information obtained from the neural network according to the different training and testing cases. An example of statistical report is shown as Figure 4.1, which lists the tested tumour total, successful total, and "questionable failure" total under the examine bound titled "*EXBD*". For those "Fail" samples, the actual outputs probabilities for six types of tumour are calculated by percentage in field *Failure Output.* Applying "winner takes all" methodology, for the "Fail" sample, if the final diagnosis is as the same as its target tumour type, then this sample is still correct diagnosed; i.e. Three samples for tumour type **gbm** are correct classified. Otherwise the sample is really misdiagnosed. i.e. the rest of "Fail_Sample" in Figure 4.1.

There are any kinds of ways to category the test cases for statistical analysis. For example, the "leave-one-out" method is categoried by testing the left one sample after training the rest of samples within the same data set; and "data-sets" method is categoried by the Clean or Noisy or both data sets for training and testing many samples in one experiment. No matter which category is applied, the following test cases can be experimented:

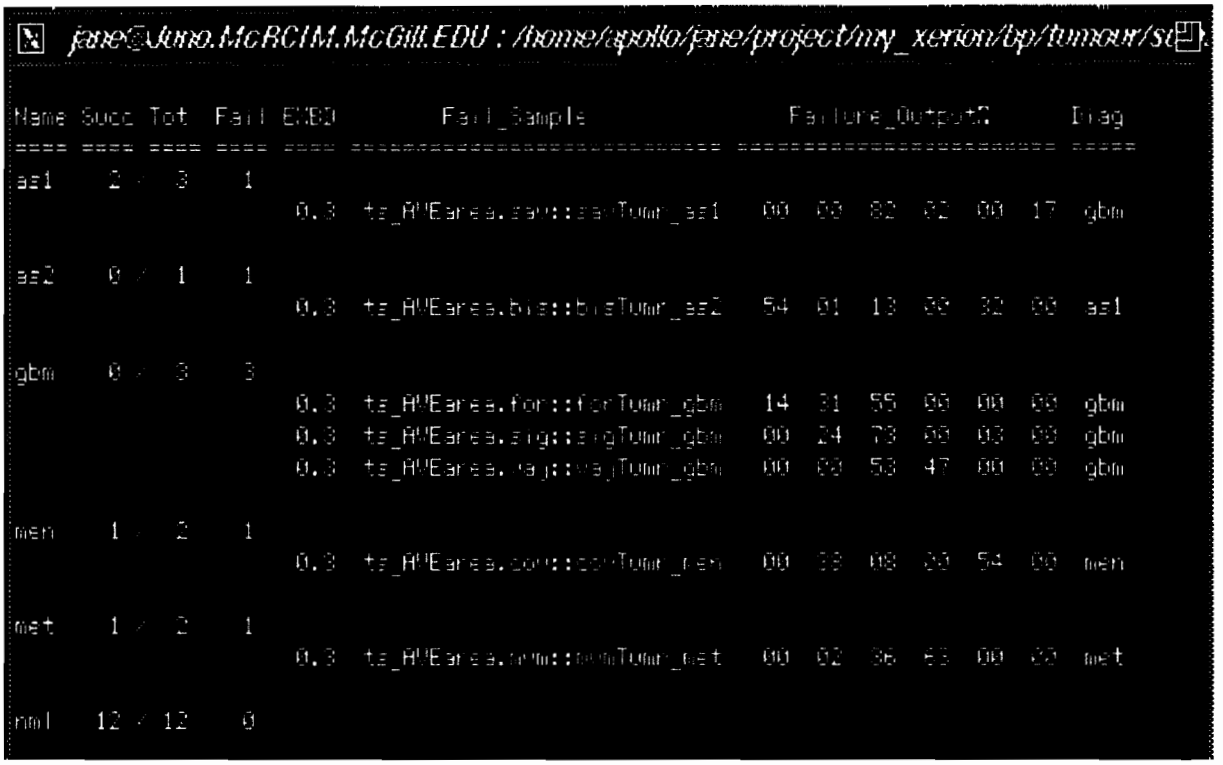


Figure 4.1: Sample of Statistical Report

1. "Leave-one-out" category method :

   "Many-to-one" - training many samples and test one:

   - Case 1: Train Clean data and "leave-one-out" for test.
   - Case 2. Train Average data and "leave-one-out" for test.
   - Case 3. Train List data and "leave-one-patient-out" for test.
   - Case 4. Train one Patient List data and "leave-one-voxel-out" for test.

   "Many-to-many" - training many and testing many samples:

   - Case 5. Train Clean data and test List data.
   - Case 6. Train List data and test Clean data.
   - Case 7. Train Clean data and test Average data.
   - Case 8. Train Average data and test Clean data.

- Case 9. Train List data and test Average data.
- Case 10. Train Average data and test List data.

2. "Data-sets" category method :

   - Clean data via Clean data.

     Case 1 falls into this category.

   - Clean data via Noisy data.

     Case 5, Case 6, Case 7 and Case 8 fall into this category.

   - Noisy data via Noisy data.

     Case 2, Case 3, Case 4, Case 9 and Case 10 fall into this category.

All of above ten cases can be applied by Peak data sets, and Case 2 to Case 10 can be applied to Area data sets because Clean data set is the Average Peak data rather than the Area data.

There are many kinds of category methods, In the following sections, we use the "leave-one-out" as the category method and give the statistical analysis of final diagnosis.

4.2 Case 1: Train Clean data and "leave-one-out" for Test

There are 105 samples in Clean data. Every time the neural network trains 104 samples and tests the left one sample, by changing the "leave-one-out" sample iteratively, until all the individual sample can be tested by the neural network.

Figure 4.2 shows the process of training and testing Clean data by "leave-one-out" method.

Table 4.1 summarizes the recursive test results for training Clean data set and testing the left one sample each time. The results show that 100% correct diagnosis rate for this experiment.

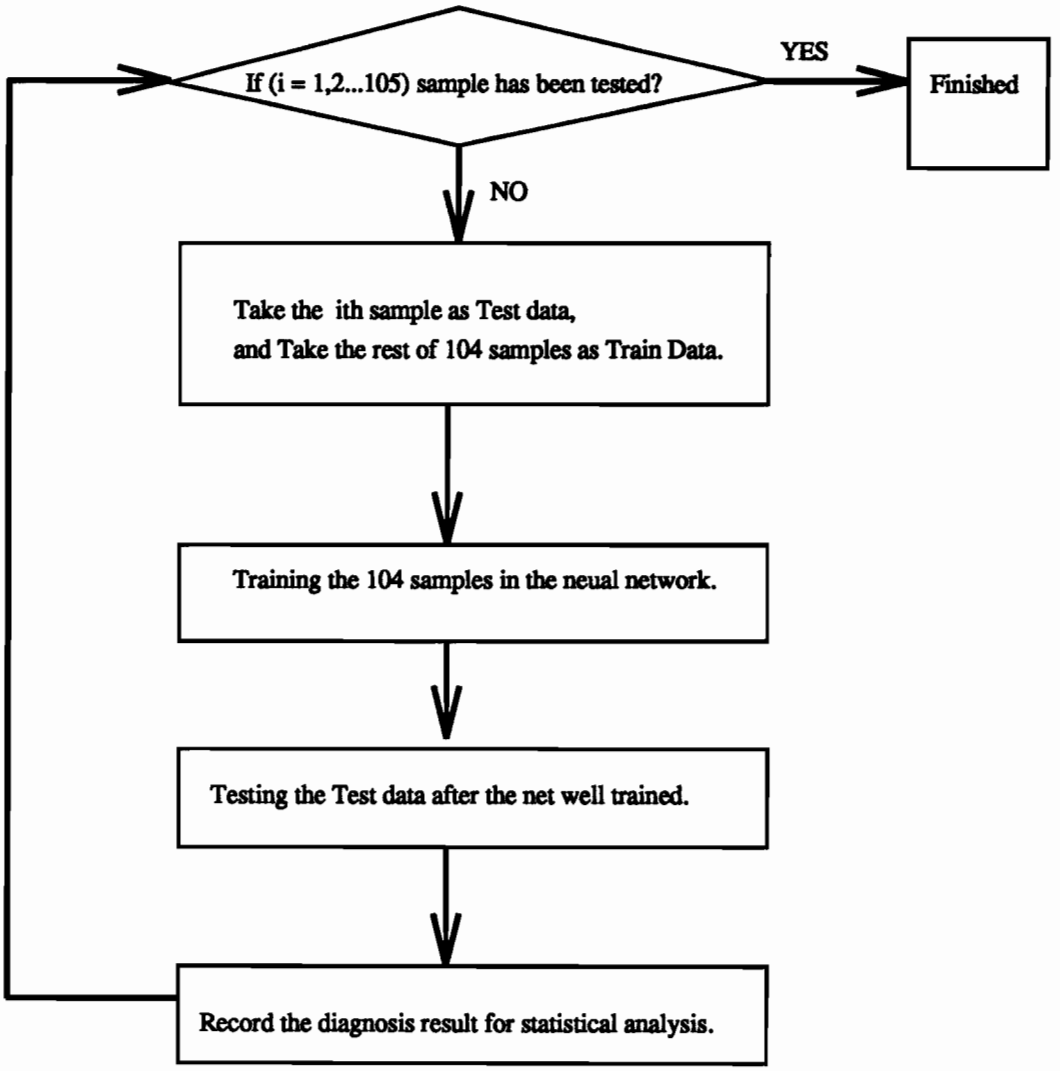


Figure 4.2: "Leave-one-out" Process for Diagnosis of Clean Data

Tumor Type	*Train Sample*	*Test Sample*	*Diagnosis Records*						*Diag Type*	*Succ Rate*	*Succ/ Fail*
			as1	*as2*	*gbm*	*men*	*met*	*nml*			
as1	20	20	20	0	0	0	0	0	as1	20/20	S
as2	22	22	0	22	0	0	0	0	as2	22/22	S
gbm	24	24	0	0	24	0	0	0	gbm	24/24	S
men	9	9	0	0	0	9	0	0	men	9/9	S
met	16	16	0	0	0	0	16	0	met	16/16	S
nml	14	14	0	0	0	0	0	14	nml	14/14	S

Table 4.1: Summary of "leave-one-out" for Clean Data

Since the Clean data set is the averaged peaks feature values, so no data set of areas feature values is applied in this case.

4.3 Case 2. Train Average data and "leave-one-out" for Test

Table 4.2 and Table 4.3 summarize the diagnosis results for both Peak and Area of Average data by "leave-one-patient-out" method.

Although the training data set is too small, the classification for Area data shows the total of 5/6 successful rate. Comparing with Peak data's 2/6 correct diagnosis, the Area feature values for classification are much better.

Tumor Type	*Train Sample*	*Test Sample*	*Diagnosis Records*						*Diag Type*	*Succ Rate*	*Succ/ Fail*
			as1	*as2*	*gbm*	*men*	*met*	*nml*			
as1	3	3	2	0	0	0	0	1	as1	2/3	S
as2	1	1	0	0	0	1	0	0	men	0/1	F
gbm	3	3	0	0	1	1	1	0	-	1/3	F
men	2	2	0	0	1	1	0	0	-	1/2	-
met	2	2	0	0	1	0	1	0	-	1/2	-
nml	12	12	0	0	0	0	0	12	nml	12/12	S

Table 4.2: Summary of "leave-one-out" for Average Peak Data

4.4 Case 3. Train List data and "leave-one-patient-out" for Test

Table 4.4 and Table 4.5 show the neural network diagnosis for training the noisy individual peaks feature samples and testing the left one patient's sample.

Table 4.6 and Table 4.7 show the same experiment above but using *Area data set* instead of *Peak data set.*

Tumor Type	*Train Sample*	*Test Sample*	*Diagnosis Records*						*Diag Type*	*Succ Rate*	*Succ/ Fail*
			as1	*as2*	*gbm*	*men*	*met*	*nml*			
as1	3	3	2	0	1	0	0	0	as1	2/3	S
as2	1	1	0	0	0	1	0	0	men	0/1	F
gbm	3	3	0	0	3	0	0	0	gbm	3/3	S
men	2	2	0	0	0	2	0	0	men	2/2	S
met	2	2	0	0	0	0	2	0	met	2/2	S
nml	12	12	0	0	0	0	0	12	nml	12/12	S

Table 4.3: Summary of "leave-one-out" for Average Area Data

Test Sample	*Name Type*		bisgi1 as2	brina1 as1	covdi2 men	forpa1 gbm	fuliq1 nml	mymmi1 met
Total	*Tumour*		8	6	8	10	0	5
	Normal		5	6	4	5	12	6
Test Samples Diagnosis Results	*Tumour*	*as1*	3	6	0	1	0	0
		as2	0	0	4	5	0	0
		gbm	1	0	0	0	0	3
		men	4	0	4	0	0	0
		met	0	0	0	4	0	2
		nml	0	0	0	0	0	0
	Normal	*as1*	0	0	0	0	0	0
		as2	0	0	0	0	0	0
		gbm	0	0	0	0	0	0
		men	0	0	0	0	0	0
		met	0	0	0	0	0	0
		nml	5	6	4	5	12	6
Success Rate	*Tumour*		0/8	6/6	4/8	0/10	-	2/5
	Total		5/13	12/12	8/12	5/15	12/12	8/11
ActualType			men	as1	-	as2	nml	gbm
Succ/Fail			F	S	-	F	S	F

Table 4.4: Summary of "leave-one-out" for Peak List data

Test Sample	*Name*		pfeye1	ravri1	savyv1	sigam2	valje1	valul1
	Type		men	as1	as1	gbm	gbm	met
Total	*Tumour*		8	9	4	6	9	4
	Normal		6	6	5	5	5	6
Test Samples Diagnosis Results	*Tumour*	*as1*	0	6	0	0	0	0
		as2	0	0	1	0	0	0
		gbm	0	1	0	5	4	4
		men	8	0	1	1	0	0
		met	0	0	0	0	5	0
		nml	0	2	2	0	0	0
	Normal	*as1*	0	0	0	0	0	0
		as2	0	0	0	0	0	0
		gbm	0	0	1	0	0	0
		men	0	0	0	0	0	0
		met	0	0	0	0	0	0
		nml	6	6	4	5	5	6
Success Rate	*Tumour*		8/8	6/9	0/4	5/6	4/9	0/4
	Total		14/14	12/15	4/9	10/11	9/14	6/10
ActualType			men	as1	nml	gbm	met	gbm
Succ/Fail			S	S	F	S	F	F

Table 4.5: Continue of Summary of "leave-one-out" for Peak List Data

Test Sample	*Name*		bisgi1	brina1	covdi2	forpa1	fuliq1	mymmi1
	Type		as2	as1	men	gbm	nml	met
Total	*Tumour*		8	6	8	10	0	5
	Normal		5	6	4	5	12	6
Test Samples Diagnosis Results	*Tumour*	*as1*	0	3	0	1	0	0
		as2	0	1	5	1	0	0
		gbm	3	1	0	8	0	5
		men	5	0	3	0	0	0
		met	0	0	0	0	0	0
		nml	0	1	0	0	0	0
	Normal	*as1*	0	0	0	0	0	0
		as2	0	0	0	0	0	0
		gbm	0	0	0	0	0	0
		men	0	0	0	0	0	0
		met	0	0	0	0	0	0
		nml	5	6	4	5	12	6
Success Rate	*Tumour*		0/8	3/6	3/8	8/10	-	0/5
	Total		5/13	9/12	7/12	13/15	12/12	6/11
ActualType			men	as1	as2	gbm	nml	gbm
Succ/Fail			F	S	F	S	S	F

Table 4.6: Summary of "leave-one-out" for Area List data

Test Sample	*Name*		pfeye1	ravri1	savyv1	sigam2	valje1	valul1
	Type		men	as1	as1	gbm	gbm	met
Total	*Tumour*		8	9	4	6	9	4
	Normal		6	6	5	5	5	6
Test Samples Diagnosis Results	*Tumour*	*as1*	0	6	0	0	0	0
		as2	0	3	0	1	0	0
		gbm	0	0	2	5	3	0
		men	8	0	0	0	0	0
		met	0	0	0	0	6	4
		nml	0	0	2	0	0	0
	Normal	*as1*	0	0	0	0	0	0
		as2	0	0	0	0	0	0
		gbm	0	0	0	0	0	0
		men	0	0	0	0	0	0
		met	0	0	0	0	0	0
		nml	6	6	5	5	5	6
Success Rate	*Tumour*		8/8	6/9	0/4	5/6	3/9	4/4
	Total		14/14	12/15	5/9	10/11	8/14	10/10
ActualType			men	as1	-	gbm	met	met
Succ/Fail			S	S	F	S	F	S

Table 4.7: Continue of Summary of "leave-one-out" for Area List data

Comparing with results produced by the peaks feature values and areas feature values in the neural network, the success diagnosis rate is 5/12 for the peaks feature values as input and 6/12 for the areas feature values as input. So the conclusion is made that extracting the areas values as feature values is better than extracting the peaks values.

4.5 Case 4. Train one Patient List data and "leave-one-voxel-out" for Test

The patient *mymmi1* has 11 voxel sample data, 5 voxels are tumour type *met* and 6 voxels are normal type *nml*.

Training *mymmi1*'s 10 voxel sample data, and test each left voxel sample iteratively. The result shows the perfect diagnosis. Table 4.8 gives the 100% correct diagnosis results.

Another similar experiment is given in *summ_for30.p_vaj_sig*. Since these three patients (*forpa1, valje1* and *sigam2*) have the same type of *gbm* tumour, so this experiment trains *forpa1*'s sample data and tests the rest of two patients' samples. The result shows: for tumour samples, all the diagnosis are correct, but for normal samples, some of samples are misdiagnosed as *gbm*.

These results tell us that the training set for specific tumour is increasing the diagnosis direction to this type of tumour.

4.6 Case 5. Train Clean data and Test List data

In this experiment, for testing both peaks feature samples and areas feature samples, the results are not good. The diagnoses for men, met, nml are slightly correct, but for as1, as2 and gbm are wrong.

Voxel Number	*Train Sample*	*Test Sample*	*Diagnosis Records*						*Diag Type*	*Succ Rate*	*Succ/ Fail*
			as1	*as2*	*gbm*	*men*	*met*	*nml*			
563	10	1	0	0	0	0	1	0	met	1/1	S
564	10	1	0	0	0	0	1	0	met	1/1	S
532	10	1	0	0	0	0	1	0	met	1/1	S
499	10	1	0	0	0	0	1	0	met	1/1	S
498	10	1	0	0	0	0	1	0	met	1/1	S
493	10	1	0	0	0	0	0	1	nml	1/1	S
557	10	1	0	0	0	0	0	1	nml	1/1	S
558	10	1	0	0	0	0	0	1	nml	1/1	S
590	10	1	0	0	0	0	0	1	nml	1/1	S
405	10	1	0	0	0	0	0	1	nml	1/1	S
372	10	1	0	0	0	0	0	1	nml	1/1	S

Table 4.8: Summary of "leave-one-voxel-out" for Patient mymmi1

4.7 Case 6. Train List data and Test Clean data

The result of this testing case is the same as Case 5. Since the feature extraction methods are different for Clean data set and Noisy data set, the Clean data set are the averaged peak feature values and noisy data set are the individual voxel peaks feature values. This is the main reason.

4.8 Case 7. Train Clean data and Test Average data

The results of training Clean data set and testing both noisy averaged peaks data set and noisy averaged areas data set are stated in Table 4.9 and Table 4.10.

From this testing case, we know that the areas feature samples are better than the peaks feature samples.

Tumor Type	*Train Sample*	*Test Sample*	*Diagnosis Records*						*Diag Type*	*Succ Rate*	*Succ/ Fail*
			as1	*as2*	*gbm*	*men*	*met*	*nml*			
as1	20	3	1	0	0	2	0	0	men	1/3	F
as2	22	1	0	0	0	1	0	0	men	0/1	F
gbm	24	3	0	0	0	0	3	0	met	0/3	F
men	9	2	0	0	0	2	0	0	men	2/2	S
met	16	2	0	0	1	0	1	0	-	1/2	F
nml	14	12	0	0	0	0	0	12	nml	12/12	S

Table 4.9: Train Clean Data and Test Average Peak Data

Tumor Type	*Train Sample*	*Test Sample*	*Diagnosis Records*						*Diag Type*	*Succ Rate*	*Succ/ Fail*
			as1	*as2*	*gbm*	*men*	*met*	*nml*			
as1	20	3	0	0	0	2	1	0	men	0/3	F
as2	22	1	0	0	0	1	0	0	men	0/1	F
gbm	24	3	0	0	0	0	3	0	met	0/3	F
men	9	2	0	0	0	2	0	0	men	2/2	S
met	16	2	0	0	0	0	2	0	met	2/2	S
nml	14	12	0	0	0	0	0	12	nml	12/12	S

Table 4.10: Train Clean Data and Test Average Area Data

4.9 Case 8. Train Average data and Test Clean data

Table 4.11 and Table 4.12 illustrate the results for training noisy average peaks data set and areas data set respectively, and testing Clean data set.

Tumor Type	*Train Sample*	*Test Sample*	*Diagnosis Records*						*Diag Type*	*Succ Rate*	*Succ/ Fail*
			as1	*as2*	*gbm*	*men*	*met*	*nml*			
as1	3	20	20	0	0	0	0	0	as1	20/20	S
as2	1	22	0	0	0	0	22	0	met	0/22	F
gbm	3	24	0	0	0	0	24	0	met	0/24	F
men	2	9	0	8	1	0	0	0	as2	0/9	F
met	2	16	0	0	1	0	14	1	met	14/16	S
nml	12	14	0	0	0	0	0	14	nml	14/14	S

Table 4.11: Train Average Peak Data and Test Clean Data

Tumor Type	*Train Sample*	*Test Sample*	*Diagnosis Records*						*Diag Type*	*Succ Rate*	*Succ/ Fail*
			as1	*as2*	*gbm*	*men*	*met*	*nml*			
as1	3	20	20	0	0	0	0	0	as1	20/20	S
as2	1	22	22	0	0	0	0	0	as1	0/22	F
gbm	3	24	23	0	0	0	1	0	met	0/24	F
men	2	9	0	9	0	0	0	0	as2	0/9	F
met	2	16	0	0	6	0	10	0	met	10/16	S
nml	12	14	14	0	0	0	0	0	as1	0/14	F

Table 4.12: Train Average Area Data and Test Clean Data

Since the training data set is too small, it can not cover most of feature characteristics in the testing data set. Comparing the peaks feature samples' testing with the areas feature samples' testing, it makes sense that the first test is better than the latter because the Clean data set is made of peaks feature.

4.10 Case 9. Train List data and Test Average data

In Case 9 and Case 10, the training and testing data set are both Noisy data. Case 9 trains List data and tests Average data, while Case 10 is another way round. The results for Case 9 are in Table 4.13 and Table 4.14.

The experiments are done in the same way as before by both Peak data and Area data. The experiment for Peak data group gets 100% correct diagnosis. For Area data group, all the classifications are correct but the tumour type *met*.

Tumor Type	*Train Sample*	*Test Sample*	*Diagnosis Records*						*Diag Type*	*Succ Rate*	*Succ/ Fail*
			as1	*as2*	*gbm*	*men*	*met*	*nml*			
as1	19	3	3	0	0	0	0	0	as1	3/3	S
as2	8	1	0	1	0	0	0	0	as2	1/1	S
gbm	25	3	0	0	3	0	0	0	gbm	3/3	S
men	16	2	0	0	0	2	0	0	men	2/2	S
met	9	2	0	0	0	0	2	0	met	2/2	S
nml	72	12	0	0	0	0	0	12	nml	12/12	S

Table 4.13: Train List Peak Data and Test Average Peak Data

Tumor Type	*Train Sample*	*Test Sample*	*Diagnosis Records*						*Diag Type*	*Succ Rate*	*Succ/ Fail*
			as1	*as2*	*gbm*	*men*	*met*	*nml*			
as1	19	3	3	0	0	0	0	0	as1	3/3	S
as2	8	1	0	1	0	0	0	0	as2	1/1	S
gbm	25	3	0	0	3	0	0	0	gbm	3/3	S
men	16	2	0	0	0	2	0	0	men	2/2	S
met	9	2	0	0	2	0	0	0	gbm	0/2	F
nml	72	12	0	0	0	0	0	12	nml	12/12	S

Table 4.14: Train List Area Data and Test Average Area Data

4.11 Case 10. Train Average data and Test List data

This experiment reverses the training data set and testing data set in Case 9. Table 4.15 and Table 4.14 show training Average data and testing List data with Peak and Area data respectively.

Tumor Type	*Train Sample*	*Test Sample*	*Diagnosis Records*						*Diag Type*	*Succ Rate*	*Succ/ Fail*
			as1	*as2*	*gbm*	*men*	*met*	*nml*			
as1	3	19	16	1	0	2	0	0	as1	16/19	S
as2	1	8	0	4	1	3	0	0	as2	4/8	S
gbm	3	25	0	3	18	3	1	0	gbm	18/25	S
men	2	16	0	0	0	16	0	0	men	16/16	S
met	2	9	0	0	2	0	7	0	met	7/9	S
nml	12	72	0	0	0	0	0	72	nml	72/72	S

Table 4.15: Train Average Peak Data and Test List Peak Data

Tumor Type	*Train Sample*	*Test Sample*	*Diagnosis Records*						*Diag Type*	*Succ Rate*	*Succ/ Fail*
			as1	*as2*	*gbm*	*men*	*met*	*nml*			
as1	3	19	9	0	1	2	0	7	nml	9/19	F
as2	1	8	0	4	0	4	0	0	-	4/8	-
gbm	3	25	0	1	22	2	0	0	gbm	22/25	S
men	2	16	0	0	0	16	0	0	men	16/16	S
met	2	9	0	0	2	0	7	0	met	7/9	S
nml	12	72	0	0	0	0	0	72	nml	72/72	S

Table 4.16: Train Average Area Data and Test List Area Data

4.12 Conclusions

Overview the classifications made by the neural network for both Clean data and Noisy data, the evaluations of the performance of neural network are as followings:

- Excellent(100%):

  Train Clean data and "leave-one-out" for test.

  Train one Patient data and "leave-one-voxel-out" for test.

  Train List Peak data and test Average Peak data.

  Train Average Peak data and test List Peak data.

- Good(99-51%):

  Train Average Area data and "leave-one-patient-out" for test.

  Train Area List data and "leave-one-out" for test.

  Train List Area data and test Average Area data.

  Train Average Area data and test List Area data.

  Train Clean data and test Average Area data.

  Train Average Peak data and test Clean data.

- poor(50-33%):

  Train Average Peak data and "leave-one-patient-out" for test.

  Train Peak List data and "leave-one-out" for test.

  Train Clean data and test Average Peak data.

  Train Average Area data and test Clean data.

The neural network on automatic brain tumour diagnosis performs very good for Clean data or Noisy data set separately, it is about average for the combinations of Clean and Noisy data set as training and testing data sets. Since the Noisy data set came from 12 patients for 6 types of tumours(3 patients for *as1*; 1 patient for *as2*; 3 patients for *gbm*; 2 patients for *men*, 2 patients for *met* and 1 patient for *nml*), so it is too small to cover all different tumour characteristics. For examples, the patient

bisgi1 is the only one to have tumour type of *as2*, when "leave-one-patient-out" for testing *bisgi1*, in the corresponding training data set, there is no tumour type *as2* samples at all. So it is hard to classify *bisgi1*'s tumour type at this case.

There are three major factors to achieve the successful classification: The first factor is that the size of the training data set should big enough to cover all kinds of diagnosis tumour types. The second factor is that the extracted features should reveal the different kinds of tumour's characteristics. So the feature extraction plays a key role for the classification. The last factor is the neural network's architecture and learning method.

Chapter 5

Summary And Future Work

Neural network application are widely applied in various fileds, such as knowledge processing, pattern recognition, speech, classification and so on.

This project is a data classification problem used for automatic brain tumour diagnosis. It employs the **Back propagation** paradigm, feedforward multilayer network architecture with supervised learning procedure.

From feature extraction to get training and testing neural network data sets; from the traing and testing neural network to final diagnosis statistical analysis, this project makes these processes performed automatically. Concluded in Chapter 4, some diagnosis results get 100% successful rate, some tests are above 50% correct classification. Since the Noisy data set is extracted from 12 patients for 6 tumour types, it is too small for training and testing neural network comparing with the Clean data set extracted from 100 patients for 6 tumour types. So 50% successful diagnosis rate may not be a general solution.

Although this project has shown that neural network is a good approach for data processing, it still has some parts needed to improve or to try in the future.

Some future works might be considered as followings:

1. Image segmentation by computer.

   So far the **MR** Image is recognized by the brain tumour experts. Since the

spectra of region of interest(**ROI**) are picked out manually for the feature extraction, a lot of time are spent and this procedure can be done only by experts. In the future, if the **MR** Image pattern recognition can be done by computer, the whole process of brain tumour diagnosis will become totally automatic.

2. Try different neural network paradigms.

   Currently the back-propagation feedforward paradigm and supervised learning are applied in brain tumour diagnosis. There are many different paradigms and learning procedures for neural network architecture. Such as neural network with two hidden layers, recursive architecture, or with bm, unsupervised learning and so on. If the different approaches are implemented, by comparing these diagnosis results , a best neural network paradigm will be found out for more aspects of brain tumour diagnosis.

3. Employ different neural network simulator tools.

   There are many tools for neural network simulator. Addition to **Xerion, Aspirin/Migrains, SNN. Xerion** is very good for back-propagation simulator and has very friendly user interface for user to understand and use. **Aspirin/MIGRAINES**[AM 93] is developed in *CMU* and can be run without Sun workstation. It can be used to build the feedforward, feedback and other complex neural network architectures such as feedforward layered networks with arbitrary connections, "skip level" connections, time delay neural networks and so on. **MIGRAINES** is the user interface for **Aspirin**, and it is not as good as **Xerion. SNN** is newly developed simulator tool which has both good user interface as **Xerion** and complex functions as **Aspirin.**

Neural network is a good approach to brain tumour diagnosis. It has a prospect future in its research, representations, learning algorithms and application areas.

Reference

Bibliography

[AM 93] Russell R. Leighton. *The Aspirin/MIGRAINES Neural Network Software User's Manual, V6.0.* Russell Leighton and MITRE Corporation, 1993.

[Arno 91] D.L. Arnold, P.M. Matthews, G. Francis, and J. Antel. *Magnetic resonance spectroscopic imaging (MRSI) allows metabolic characterization of brain lesions.* MedicaMundi, vol.36, no. 2, 1991.

[Anto 91] P. Antognetti and V. Milutinovic Editors. *Neural Networks : Concepts, Applications and Implementations.* Volumn 1, Prentice-Hall Inc, 1991.

[Dayh 90] Judith E. Dayhoff, *Neural network architectures: an introduction.* Van nostrand reinhold, 1990.

[Hint 87] Geofferey E. Hinton. *Connectionist Learning Procedures.* Technical Report CMU-CS-87-115(version 2).

[Judd 90] J. Stephen Judd, *Neural network design and the complexity of learning.* The MIT press, 1990.

[Koho 82] Kohonen, *Clustering, taxonomy, and topological maps of patterns.* In Lang, M., edit, *Proceedings of the Sixth International Conference on Pattern Recognition,* IEEE Computer Society Press, Silver Spring, MD.

[Lau 92] Clifford Lau, *Neural networks: Theoretical foundations and analysis.* IEEE press. 1992.

[Lect 91] *Artificial neural networks. Lecture notes in computer science 540.* 1991.

[McCl 89] James L. McClelland, David E. Rumelhart, *Explorations in parallel distributed processing: a handbook of models, programs and exercises.* 1989 1.V.

[Morg 91] David P. Morgan, Christopher L. Scofield, *Neural networks and speech processing.* Kluwer academic publishers, 1991.

[Preu 94] M. Preul, Z. Caramanos, J-G. Villemure, W. Feinadel, and D. Arnold. *Linear Discriminant Analysis Based on Proton MR Spectroscopic Imaging of Human Brain Tumours Improves Pre-Operative Diagnosis.* Montreal Neurological Institute, McGill University, Montreal, Canada.

[Rume 86] Rumelhart, D. E., Hinton, G. E., and Williams, R. J. 1986. *learning internal representations by back-propagation errors.* Nature, 1986.

[Rume 86] David E. Rumelhart, *Parallel distributed processing.* V.1, 1986.

[Simp 90] Patrick K. Simpson, *Artificial neural systems, foundations, paradigms, applications and implementations.* Pergamon press Inc., 1990.

[Tour 89] David S. Touretzky, *Advances in Neural information processing systems.* Morgan kaufmann publishers, V1, 1989.

[Xeri 93] Drew van Camp. *A User Guide for The* **Xerion** *Neural Network Simulator Version 3.1* Department of Computer Science, University of Toronto, May 3, 1993.

Appendix A

Source Codes for Image Processing

A.1 Source Codes for Developing Sunspec1

A.1.1 data.h

A.1.2 datain.h

A.1.3 dataout.h

A.1.4 readpar.c

A.1.5 roi.c

A.1.6 roi_select.c

A.1.7 rowdataout_winodw.c

```
/*=====================================================================*/
/*                                                                     */
/*      Copyright 1989 N.V. Philips' Gloeilampen Fabrieken.            */
/*      All rights reserved. Reproduction in whole or in part is       */
/*      prohibited without the written consent of the copyright owner. */
/*                                                                     */
/*=====================================================================*/

# define        DATA_PPM                1000000

/*----------------------+
! Extension definitions !
+----------------------*/
# define        DATA_ext_unknown        0
                /*------------------+
                ! Value extensions !
                +------------------*/
# define        DATA_ext_volt           1
# define        DATA_ext_mvolt          2
# define        DATA_ext_v_hz           3
# define        DATA_ext_rad            4
                /*----------------------+
                ! Dimension extensions !
                +----------------------*/
# define        DATA_ext_time           50
# define        DATA_ext_hz             51
# define        DATA_ext_ppm            52
# define        DATA_ext_num            53
                /*-----------------------+
                ! Direction defines      !
                +-----------------------*/
# define        DATA_direction_mu1      70
# define        DATA_direction_kx       71
# define        DATA_direction_ky       72
# define        DATA_direction_kz       73
/*-------------------------+
! Default extension names !
+-------------------------*/
# define        DATA_name_unknown       "[?]"
# define        DATA_name_volt          "[V]"
# define        DATA_name_mvolt         "[mV]"
# define        DATA_name_v_hz          "[V/Hz]"
# define        DATA_name_rad           "[Rad]"
                /*----------------------+
                ! Dimension extensions !
                +----------------------*/
# define        DATA_name_time          "[sec]"
# define        DATA_name_hz            "[Hz]"
# define        DATA_name_ppm           "[ppm]"
# define        DATA_name_num           "[num]"
                /*-----------------------+
                ! Direction names        !
                +-----------------------*/
# define        DATA_name_mu1           "mu1"
# define        DATA_name_kx            "kx"
# define        DATA_name_ky            "ky"
# define        DATA_name_kz            "kz"

/* jane 940707 */
# define        MAX_SPECIFIC_SAVE_ROWS          20

/*--------------------+
! dimension structure !
+--------------------*/
typedef struct {
    int         ext;            /* dimension extension number           */
    int         pnts;           /* number of points                     */
    float       low_val;        /* low scale value                      */
    float       step;           /* stepsize between two neighbour points */

                                /*  jane 940708 */
    int         specific_save_row_total;    /* the total of saved  roi rows       */
    int         specific_save_row_array[MAX_SPECIFIC_SAVE_ROWS];   /* saved row numb
```



```
er  */
    char        specific_save_row_type[MAX_SPECIFIC_SAVE_ROWS][6]; /* saved row type
    */
                                /*  jane 940708 */

    int         direction;      /* direction value                      */
    float       t0_point;       /* zeropoint of the signal              */
} DIM_STRUCT;

typedef struct  {
    /*----------------+
    ! Data parameters !
    +----------------*/
    bool        pars;           /* TRUE if parameters are legal */
    bool        data;           /* TRUE if data is legal        */
    float       *re;            /* pointer to real buffer       */
    float       *im;            /* pointer to imaginairy buffer */
    int         num_dim;        /* number of data buffer dimensions     */
                                /* 1 = 1 row in buffer                  */
                                /* 2 = n rows in buffer ( n > 1 )       */
                                /* 3 = n*m rows in buffer ( n,m > 1 )   */
    bool        resfr_exist;    /* TRUE if resfr is legal       */
    float       resfr;          /* resonance frequency          */
    int         aver;           /* number of averages                   */
    int         meas_type;      /* measurement type                     */
    char        file_name[DEF_STR_LEN];
    /*-------------------------+
    ! Phase correction values !
    +-------------------------*/
    float       phi_0;
    bool        phi_0_exist;
    float       phi_1;
    bool        phi_1_exist;
    /*-----------------+
    ! dimension value !
    +-----------------*/
    int         val_ext;        /* extension number of data values      */
    /*------------------+
    ! Other dimensions !
    +------------------*/
    DIM_STRUCT  dim[4];
} DATA_STRUCT;
```

```
/*-----------------------------------+
! Datastructure for view parameters !
+-----------------------------------*/
typedef struct {
    /*-------------------+
    ! General parameters !
    +-------------------*/
    float       resfr;          /* resonancy frequency                  */
    bool        resfr_exist;    /* if TRUE resonance freq is legal      */
    float       *re;            /* real data buffer                     */
    float       *im;            /* imaginairy data buffer               */
    /*----------------+
    ! Dimension value !
    +----------------*/
    int         val_ext;        /* extension of data values             */
    /*----------------+
    ! Dimension value !
    +----------------*/
    DIM_STRUCT  col;            /* colomn dimension                     */
    /*----------------+
    ! Dimension value !
    +----------------*/
    DIM_STRUCT  row;            /* row dimension                        */
} VIEW_STRUCT;
```



```
/*-------------------------------------+
! Datastructure for function parameters !
+-------------------------------------*/
typedef struct {
    /*-------------------+
    ! General parameters !
    +-------------------*/
    float       resfr;          /* resonancy frequency                  */
    bool        resfr_exist;    /* if TRUE resonance freq is legal      */
    float       *re_in;         /* real  input data buffer              */
    float       *im_in;         /* imaginairy  input data buffer        */
    float       *re_out;        /* real  output data buffer             */
    float       *im_out;        /* imaginairy  output data buffer       */
    /*-------------------------+
    ! Phase correction values !
    +-------------------------*/
    float       phi_0;
    bool        phi_0_exist;
    float       phi_1;
    bool        phi_1_exist;
    /*----------------+
    ! Dimension value !
    +----------------*/
    int         val_ext;        /* extension of data values             */
    /*----------------+
    ! Dimension value !
    +----------------*/
    DIM_STRUCT  col;            /* colomn dimension                     */
    /*----------------+
    ! Dimension value !
    +----------------*/
    DIM_STRUCT  row;            /* row dimension                        */
} FUNC_STRUCT;
/*-----------------------------+
! External routine definitions !
+-----------------------------*/
extern  DATA_STRUCT     *DATA_read_ptr();
extern  bool            DATA_read_resfr();

/*----------------------+
! Data type definitions !
+----------------------*/
# define        DATA_type_unknown               -1
# define        DATA_vax_complex_int            0
# define        DATA_vax_complex_float          1
# define        DATA_sun_complex_float          2
                /* Seperated storage of real and imaginairy rows        */
# define        DATA_sun_complex_float_sep      3

/*---------------------------------+
! I/O parameter string definitions !
+---------------------------------*/
# define  SUN_RELEASE                   "SUNspec1"
# define  SUN_RESONANCE_FREQUENCY       "synthesizer_frequency"
# define  SUN_VALUE_EXTENSION           "SUN_value_extension"
# define  SUN_NUM_DIMENSIONS            "SUN_num_dimensions "

# define  SUN_DIM1_EXT                  "SUN_dim1_ext"
# define  SUN_DIM1_PNTS                 "SUN_dim1_pnts"
# define  SUN_DIM1_LOW_VAL              "SUN_dim1_low_val"
# define  SUN_DIM1_STEP                 "SUN_dim1_step"
# define  SUN_DIM1_DIRECTION            "SUN_dim1_direction"
# define  SUN_DIM1_T0_POINT             "SUN_dim1_t0_point"

# define  SUN_DIM2_EXT                  "SUN_dim2_ext"
# define  SUN_DIM2_PNTS                 "SUN_dim2_pnts"
# define  SUN_DIM2_LOW_VAL              "SUN_dim2_low_val"
# define  SUN_DIM2_STEP                 "SUN_dim2_step"
# define  SUN_DIM2_DIRECTION            "SUN_dim2_direction"
# define  SUN_DIM2_T0_POINT             "SUN_dim2_t0_point"

# define  SUN_DIM3_EXT                  "SUN_dim3_ext"
```

```
# define  SUN_DIM3_PNTS                "SUN_dim3_pnts"
# define  SUN_DIM3_LOW_VAL             "SUN_dim3_low_val"
# define  SUN_DIM3_STEP                "SUN_dim3_step"
# define  SUN_DIM3_DIRECTION           "SUN_dim3_direction"
# define  SUN_DIM3_T0_POINT            "SUN_dim3_t0_point"

# define  SUN_DIM4_EXT                 "SUN_dim4_ext"
# define  SUN_DIM4_PNTS                "SUN_dim4_pnts"
# define  SUN_DIM4_LOW_VAL             "SUN_dim4_low_val"
# define  SUN_DIM4_STEP                "SUN_dim4_step"
# define  SUN_DIM4_DIRECTION           "SUN_dim4_direction"
# define  SUN_DIM4_T0_POINT            "SUN_dim4_t0_point"

# define  SUN_PHI_0                    "SUN_phi_0"
# define  SUN_PHI_1                    "SUN_phi_1"

/*-----------------------------------------+
! Gyroscan I/O paraimeter string definitions !
+-----------------------------------------*/
# define  GYRO_AVERAGES                "averages"
# define  GYRO_SYNT_FREQ               "synthesizer_frequency"
# define  GYRO_SAMPLE_FREQ             "sample_frequency"
# define  GYRO_SAMPLES                 "samples"
# define  GYRO_VALUE_EXT               "spec_sample_extension"
# define  GYRO_DATA_TYPE               "spec_data_type"

# define  GYRO_COL_PNTS                "spec_num_col"
# define  GYRO_COL_LOW                 "spec_col_lower_val"
# define  GYRO_COL_HIGH                "spec_col_upper_val"
# define  GYRO_COL_EXT                 "spec_col_extension"

# define  GYRO_ROW_PNTS                "spec_num_row"
# define  GYRO_ROW_LOW                 "spec_row_lower_val"
# define  GYRO_ROW_HIGH                "spec_row_upper_val"
# define  GYRO_ROW_EXT                 "spec_row_extension"

                                       /*  jane 940708 */
# define  GYRO_ROI_ROWS_TOT           "spec_save_row_total"
# define  GYRO_ROW_SAVE                "spec_save_row_num"
# define  GYRO_ROW_TYPE                "spec_save_row_type"
                                       /*  jane 940708 */
```

```
/*=========================================================================*/
/*                                                                         */
/*      Copyright 1989 N.V. Philips' Gloeilampen Fabrieken.                */
/*      All rights reserved. Reproduction in whole or in part is           */
/*      prohibited without the written consent of the copyright owner.     */
/*                                                                         */
/*=========================================================================*/

/*MPF ::: FUNCTIONS :: DATAIN : MODULE_DESCRIPTION ==========

  DESCRIPTION
        These routines read in parameters and data.
*/
# include <stdio.h>
# include <string.h>
# include <malloc.h>
# include <sys/file.h>

# include "specglo.h"
# include "exception.h"
# include "debug.h"
# include "readpar.h"
# include "data.h"
# include "roi.h"

/*------------------------+
! General data parameters !
+------------------------*/
static  char err_no_pars[]      = "E: no data parameters read (DATAIN)";
static  char err_num_samp[]     = "E: parameter samples missing (DATAIN)";
static  char err_sampfr[]       = "E: parameter sample frequency missing (DATAIN)";
static  char err_buf_allo[]     = "E: no more dynamic memory (DATAIN)";
static  char err_data_type[]    = "E: wrong data type (DATAIN)";
static  char err_select[]       = "E: wrong selection type (DATAIN)";
static  char err_row_range[]    = "E: illegal row range (DATAIN)";
static  char  err_no_prof[]     = "E: no 2d profiles parameter (DATAIN)";
```



```
/*-----------------------------------------------------------+
! Datastructures for reading strings out of parameter files !
+-----------------------------------------------------------*/
static char str1[DEF_STR_LEN];
static char str2[DEF_STR_LEN];

/* name, name_len, par_found, num_char, max_len, val_ptr                */

static PAR_STR_TYPE s_arr[] = {
        { "phase_encoding_enable",      0, FALSE, 0, DEF_STR_LEN, str1 } ,
        { "phase_encoding_direction",   0, FALSE, 0, DEF_STR_LEN, str2 } ,
        { ""    , 0, FALSE, 0, 0, CHAR_NULL_PTR }
};

/* s_arr index          */
# define        PH_CODE_ENABLE          0
# define        PH_CODE_DIR             1
/*-----------------------------------------------------------+
! Datastructures for reading arrays out of parameter files !
+-----------------------------------------------------------*/
static double arr0[1];
static double arr1[1];
static double arr2[1];
static double arr3[1];
static double arr4[1];
static double arr5[1];
static double arr6[1];
static double arr7[1];

/* name, name_len, par_found, wanted, num_val , arr_len, arr_ptr        */

static PAR_DOUBLE_TYPE d_arr[] = {
        { GYRO_SAMPLES                  , 0, FALSE, 1, 0, 1, arr0 } ,
        { GYRO_SAMPLE_FREQ              , 0, FALSE, 1, 0, 1, arr1 } ,
        { GYRO_SYNT_FREQ                , 0, FALSE, 1, 0, 1, arr2 } ,
        { GYRO_AVERAGES                 , 0, FALSE, 1, 0, 1, arr3 } ,
        { "nr_phase_encoding_profiles"  , 0, FALSE, 1, 0, 1, arr4 } ,
        { "t0_mu1_direction"            , 0, FALSE, 1, 0, 1, arr5 } ,
        { "t0_kx_direction"             , 0, FALSE, 1, 0, 1, arr6 } ,
        { "t0_ky_direction"             , 0, FALSE, 1, 0, 1, arr7 } ,
        { ""                            , 0, FALSE, 0, 0, 0, DOUBLE_NULL_PTR }
};

/* d_arr index                  */
# define        SAMP            0
# define        SFR             1
# define        SYNFR           2
# define        AVER            3
# define        PROF            4
# define        T0_MU1          5
# define        T0_KX           6
# define        T0_KY           7
```

```
/*-------------------------------------------------------------------+
!       get_si_measurement                                          !
!                                                                   !
+-------------------------------------------------------------------*/
static  int             /* return : nothing                        */

get_si_measurement( ptr )

DATA_STRUCT     *ptr;   /* out : read parameters                   */

{ /*-----------------+
  ! Local parameters !
  +-----------------*/
  char  *str;
  int   profiles;

  DEB( "\nGet_si_measurement     (DATAIN)" );

  /*----------------------------+
  ! Check for 2D-SI measurement !
  +----------------------------*/
  if( s_arr[PH_CODE_ENABLE].par_found == TRUE ) {
    /*---------------------------------+
    ! Check on phase encoding enable !
    +---------------------------------*/
    str = s_arr[PH_CODE_ENABLE].val_ptr;
    FUNCGEN_str_lower( str );
    if( str[0] == 'y' ) {
      /*-----------------------------------+
      ! check on phase encoding direction !
      +-----------------------------------*/
      str = s_arr[PH_CODE_DIR].val_ptr;
      FUNCGEN_str_lower( str );
      if( strcmp( str ,   "cor" ) == 0 ||
          strcmp( str , "trans" ) == 0 ||
          strcmp( str ,   "sag" ) == 0  )  {
        /*-----------------------+
        ! 2D-SI-measurement      !
        ! Get number of profiles !
        +-----------------------*/
        if( d_arr[PROF].par_found == TRUE && d_arr[PROF].num_val == 1 ) {
          profiles = (int)d_arr[PROF].arr_ptr[0];
        }else{
          EXC( err_no_prof );
        }
        /*------------+
        ! dimension 1 !
        +------------*/
        ptr->dim[1].ext         = DATA_ext_num;
        ptr->dim[1].direction   = DATA_direction_kx;
        ptr->dim[1].pnts        = profiles;
        ptr->dim[1].step        = 1.0;
        ptr->dim[1].low_val     = 1.0;
        /*------------+
        ! dimension 2 !
        +------------*/
        ptr->dim[2].ext         = DATA_ext_num;
        ptr->dim[2].direction   = DATA_direction_ky;
        ptr->dim[2].pnts        = profiles;
        ptr->dim[2].step        = 1.0;
        ptr->dim[2].low_val     = 1.0;
        /*--------------------+
        ! Set measurment type !
        +--------------------*/
        ptr->meas_type = MEAS_2d_si;
      }else if( strcmp( str , "ap" ) == 0 ||
                strcmp( str , "cc" ) == 0 ||
                strcmp( str , "lr" ) == 0  )  {
        /*-----------------------+
        ! 1D-SI-measurement      !
        ! Get number of profiles !
        +-----------------------*/
```



```
        if( d_arr[PROF].par_found == TRUE && d_arr[PROF].num_val == 1 ) {
          profiles = (int)d_arr[PROF].arr_ptr[0];
        }else{
          EXC( err_no_prof );
        }
        /*------------+
        ! dimension 1 !
        +------------*/
        ptr->dim[1].ext         = DATA_ext_num;
        ptr->dim[1].direction   = DATA_direction_kx;
        ptr->dim[1].pnts        = profiles;
        ptr->dim[1].step        = 1.0;
        ptr->dim[1].low_val     = 1.0;
        /*--------------------+
        ! Set measurment type !
        +--------------------*/
        ptr->meas_type = MEAS_1d_si;
      }
    }
  }
  /*-------------+
  ! Routine exit !
  +-------------*/
  DEB( "\nGet_si_measurement   meas = %d  exit  (DATAIN)",  ptr->meas_type );
}
```

```
/*-------------------------------------------------------------------+
!       get_general_acq_pars                                        !
!                                                                   !
+-------------------------------------------------------------------*/
static  int             /* return : nothing                        */

get_general_acq_pars( ptr )

DATA_STRUCT     *ptr;   /* out : read parameters                   */

{ /*-----------------+
  ! Local parameters !
  +-----------------*/
  DEB( "\nGet_general_acq_pars     (DATAIN)" );

  /*----------------------------+
  ! Store synthesizer frequency !
  +----------------------------*/
  if( d_arr[SYNFR].par_found == FALSE || d_arr[SYNFR].num_val != 1 ||
      d_arr[SYNFR].arr_ptr[0] < 1.0 ) {
    ptr->resfr          = 0.0;
    ptr->resfr_exist    = FALSE;
  }else{
    ptr->resfr          = d_arr[SYNFR].arr_ptr[0];
    ptr->resfr_exist    = TRUE;
  }

  /*-----------------------+
  ! Sample value extension 1
  +-----------------------*/
  ptr->val_ext = DATA_ext_volt;

  /*------------------+
  ! Number of samples !
  +------------------*/
  if( d_arr[SAMP].par_found == FALSE ||
      d_arr[SAMP].num_val != 1 ||
      d_arr[SAMP].arr_ptr[0] < 1.0 ) EXC(err_num_samp );

  /*-----------------+
  ! Sample frequency !
  +-----------------*/
  if( d_arr[SFR].par_found == FALSE ||
      d_arr[SFR].num_val != 1 ||
      d_arr[SFR].arr_ptr[0] < 1.0 ) EXC( err_sampfr );

  /*-------------------+
  ! Number of averages !
  +-------------------*/
  if( d_arr[AVER].par_found == FALSE ||
      d_arr[AVER].num_val != 1 ||
      d_arr[AVER].arr_ptr[0] < 1.0 )
  {
     ptr->aver = 1;
  }else{
     ptr->aver = (int)d_arr[AVER].arr_ptr[0];
  }

  /*-----------------+
  ! Fill dimension 0 !
  +-----------------*/
  ptr->dim[0].ext       = DATA_ext_time;
  ptr->dim[0].direction = DATA_direction_mu1;
  ptr->dim[0].pnts      = (int)d_arr[SAMP].arr_ptr[0];
  ptr->dim[0].step      = 1/d_arr[SFR].arr_ptr[0];
  ptr->dim[0].low_val   = 0.0;
  /*-------------+
  ! Routine exit !
  +-------------*/
  DEB( "\nGet_general_acq_pars   exit   (DATAIN)" );
}
```



```
/*-------------------------------------------------------------------+
!       get_t0_positions                                            !
!                                                                   !
+-------------------------------------------------------------------*/
static  int             /* return : nothing                        */

get_t0_positions( ptr )

DATA_STRUCT     *ptr;   /* out : read parameters                   */

{ /*-----------------+
  ! Local parameters !
  +-----------------*/
  float value;

  DEB( "\nGet_to_positions     (DATAIN)" );

  /*-----------------------------------+
  ! Store t0-position of mu1-direction !
  +-----------------------------------*/
  ptr->dim[0].t0_point = 0.0;
  if( d_arr[T0_MU1].par_found == TRUE && d_arr[T0_MU1].num_val == 1 ) {
    value = d_arr[T0_MU1].arr_ptr[0];
    if( value >= 0.0 && value <= 100.0 )  ptr->dim[0].t0_point = value;

  }

  /*-----------------------------------+
  ! Store t0-position of kx-direction !
  +-----------------------------------*/
  ptr->dim[1].t0_point = 0.0;
  if( d_arr[T0_KX].par_found == TRUE && d_arr[T0_KX].num_val == 1 ) {
    value = d_arr[T0_KX].arr_ptr[0];
    if( value >= 0.0 && value <= 100.0 )  ptr->dim[1].t0_point = value;

  }

  /*-----------------------------------+
  ! Store t0-position of yx-direction !
  +-----------------------------------*/
  ptr->dim[2].t0_point = 0.0;
  if( d_arr[T0_KY].par_found == TRUE && d_arr[T0_KY].num_val == 1 ) {
    value = d_arr[T0_KY].arr_ptr[0];
    if( value >= 0.0 && value <= 100.0 )  ptr->dim[2].t0_point = value;

  }

  /*-------------+
  ! Routine exit !
  +-------------*/
  DEB( "\nGet_t0_positions   exit   (DATAIN)" );
}
```

```
/*MPF ::: FUNCTIONS :: DATAIN : READS_AND_STORES_ACQ_PARAMETERS ==========

    FUNCTION NAME
        DATAIN_read_par

    DESCRIPTION
        Reads a number of parameters out of the specified parameter file
        and stores these parameters in a structure.

    ANYOTHER
        Author          : Ad Marien
        Creation Date   : 1989-03-02

    CALLING SEQUENCE
*/
int             /* return : nothing                                  */

DATAIN_read_par( file  , ptr )

char            *file;  /* in : parameter file name                  */
DATA_STRUCT     *ptr;   /* out : read parameters                     */

{ /*-----------------+
  ! Local parameters !
  +-----------------*/
  DEB( "\nDATA_read_par (%s)  ptr=%d", file , ptr );

  /*--------------------+
  ! Read parameter file !
  +--------------------*/
  READPAR( file , d_arr , s_arr );
  get_general_acq_pars( ptr );
  get_t0_positions( ptr );

  ptr->meas_type = MEAS_unknown;
  get_si_measurement( ptr );
  if( ptr->meas_type == MEAS_unknown ) {
    /*------------+
    ! dimension 1 !
    +------------*/
    ptr->dim[1].ext       = DATA_ext_num;
    ptr->dim[1].direction = DATA_direction_kx;
    ptr->dim[1].pnts      = 0.0;
    ptr->dim[1].step      = 1.0;
    ptr->dim[1].low_val   = 1.0;
  }
  ptr->pars      = TRUE;

  /*-------------+
  ! Routine exit !
  +-------------*/
  DATAGEN_par_debug( ptr );
  DEB( "\nDATAIN_read_par   exit");
}
```



```
/*MPF ::: FUNCTIONS :: DATAIN: READS_AND_STORES_THE_DATA ==========

    FUNCTION NAME
        DATAIN_read_data

    DESCRIPTION
        Reads the data, converts it to the right format and stores it.

    ANYOTHER
        Author          : Ad Marien
        Creation Date   : 1989-03-02

    CALLING SEQUENCE
*/
int             /* return : nothing                                  */

DATAIN_read_data( file, row_start , row_stop , num_rows, data_type , ptr )

char    *file;          /* in  :  data filename parameterparameter      */
int     row_start;      /* in  : start row range to be read (row_start >= 0 ) */
int     row_stop;       /* in  : stop row range to be read              */
int     num_rows;       /* in  : number of complex rows in file         */
int     data_type;      /* in  : vax complex integer                    */
                        /*     : vax complex float                      */
                        /*     : sun complex float                      */
                        /*     : sun complex floats, seperated storage */
DATA_STRUCT     *ptr;   /* out : structure to read data                 */

{ /*-----------------+
  ! Local parameters !
  +-----------------*/
  int           re_buf_len;     /* bytelenght of real and imag buffer   */
  int           compl_buf_len;  /* bytelenght of complex buffer         */
  int           fp;             /* file byte position pointer           */
  int           i;              /* loop counter                         */
  int           rows;           /* number of rows to be read            */
  complex       *cptr;          /* complex buffer pointer               */
  int           num_pnts;       /* number of complex points to be read  */

  DEB( "\nDATAIN_read_data  (%s) ptr=%d", file , ptr);
  DEB( "\n range=(%d,%d)    (%d)  type = %d",
                        row_start, row_stop , num_rows, data_type);

  if( ptr->pars == FALSE ) EXC( err_no_pars );
  /*-------------------------------------------------------------+
  ! Estimate number of samples and number of rows to be read !
  +-------------------------------------------------------------*/
  if( row_start < 0 || row_start > row_stop || row_stop >= num_rows ) {
    EXC( err_row_range );
  }
  /*-----------------------------+
  ! Allocate real and imag buffer !
  +-----------------------------*/
  rows   = row_stop - row_start +1;
  num_pnts   = rows * ptr->dim[0].pnts;
  re_buf_len = num_pnts * sizeof( float );
  DEB( "\n re_buf_len = %d  row=%d   pnts=%d", re_buf_len, rows, num_pnts );
  ptr->re = FLOAT_PTR malloc( re_buf_len );
  ptr->im = FLOAT_PTR malloc( re_buf_len );
  if( ptr->re==FLOAT_NULL_PTR || ptr->im == FLOAT_NULL_PTR ) EXC(err_buf_allo );

  /*----------------------+
  ! Read and convert data !
  +----------------------*/
  if( data_type == DATA_vax_complex_int   ||
      data_type == DATA_vax_complex_float ||
      data_type == DATA_sun_complex_float       ) {
    /*-----------------------------------------------------------+
    ! Read Vax complex integers or floats or Sun complex floats !
    +-----------------------------------------------------------*/
    compl_buf_len = num_pnts * sizeof( complex );
    DEB("\ncompl_buf_len = %d", compl_buf_len );
```

```
    cptr = COMPLEX_PTR malloc( compl_buf_len );
    if( cptr == COMPLEX_NULL_PTR ) EXC(err_buf_allo );
    E_begin
      fp = row_start * ptr->dim[0].pnts * sizeof( complex );
      READDATA( file, fp , compl_buf_len , (char *)cptr );
      /*--------------------------------+
      ! Convert read data to Sun floats !
      +--------------------------------*/
      if( data_type == DATA_vax_complex_int )
        DATACONV( CHAR_PTR cptr, 2*num_pnts , "vax_i", CHAR_PTR cptr, "sun_f" );
      else if( data_type == DATA_vax_complex_float )
        DATACONV( CHAR_PTR cptr, 2*num_pnts , "vax_f", CHAR_PTR cptr, "sun_f" );
      /*-----------------------------------------------+
      ! Convert complex to real and imaginairy buffers !
      +-----------------------------------------------*/
      for( i=0; i<num_pnts; i++ ) {
        ptr->re[i] = cptr[i].re;
        ptr->im[i] = cptr[i].im;
      }
      free( cptr );
    E_handle
    E_others
        free( cptr );
        EXC_reraise
    Exc_end
  }else if( data_type == DATA_sun_complex_float_sep ) {
    /*-----------------------------------+
    ! Read Sun complex floats , real part !
    +-----------------------------------*/
    fp = row_start * ptr->dim[0].pnts * sizeof( float );
    READDATA( file, fp , re_buf_len , (char *)ptr->re );
    /*--------------------------------------+
    ! Read Sun complex floats , imaginary part !
    +--------------------------------------*/
    fp = fp + num_rows * ptr->dim[0].pnts * sizeof( float );
    READDATA( file, fp , re_buf_len , (char *)ptr->im );
  }else{
     EXC( err_data_type );
  }

  /*------------------+
  ! Update parameters !
  +------------------*/
  if( ptr->meas_type == MEAS_2d_si ) {
    if( rows == num_rows ) {
        DEB("\n 2D SI measurement" );
        ptr->num_dim = 3;
    }else{
        DEB("\n Truncated 2D SI set" );
        ptr->meas_type  = MEAS_unknown;
        ptr->dim[1].pnts = rows;
        if( rows > 1 )  ptr->num_dim = 2;
        else            ptr->num_dim = 1;
    }
  }else{
    if( rows > 1 ) ptr->num_dim = 2;
    else           ptr->num_dim = 1;
    ptr->dim[1].pnts = rows;
  }
  ptr->data     = TRUE;
  /*--------------------------------+
  ! Correct for number of averages !
  +--------------------------------*/
  if( ptr->aver > 1 ) {
      for( i=0; i<num_pnts; i++ ) ptr->re[i] /= ptr->aver;
      for( i=0; i<num_pnts; i++ ) ptr->im[i] /= ptr->aver;
  }
  /*-------------+
  ! Routine exit !
  +-------------*/
  DATAGEN_par_debug( ptr );
  DEB( "\nDATAIN_read_data    exit" );
}
```

```
/*=====================================================================*/
/*                                                                     */
/*      Copyright 1989 N.V. Philips' Gloeilampen Fabrieken.            */
/*      All rights reserved. Reproduction in whole or in part is       */
/*      prohibited without the written consent of the copyright owner. */
/*                                                                     */
/*=====================================================================*/

/*MPF ::: FUNCTIONS :: DATAOUT : MODULE_DESCRIPTION ==========

  DESCRIPTION
        These routines write data and parameters to files.
*/
# include <stdio.h>
# include <string.h>
# include <malloc.h>
# include <sys/file.h>

# include "specglo.h"
# include "exception.h"
# include "debug.h"
# include "readpar.h"
# include "data.h"

/*-------------------------+
! General data parameters !
+-------------------------*/
static  char err_open[]          = "E: open file (DATAOUT)";
static  char err_buf_allo[]      = "E: no more dynamic memory (DATAOUT)";
static  char err_data_type[]     = "E: wrong data type (DATAOUT)";
static  char err_too_few[]       = "E: too few bytes written (DATAOUT)";
```



```
/*MPF ::: FUNCTIONS :: DATAOUT : WRITE_DATA_ROW_TO_FILE ==========

    FUNCTION NAME
        DATAOUT_write_data

    DESCRIPTION
        Write data rows to file.

    ANYOTHER
        Author          : Ad Marien
        Creation Date   : 1989-03-02

    CALLING SEQUENCE
*/

int             /* return : nothing                                    */

DATAOUT_write_data( file, re, im, num_complex, data_type )

char    *file;          /* in :  data filename                         */
float   *re;            /* in : real buffer pointer                    */
float   *im;            /* in : imaginairy  buffer pointer             */
int     num_complex;    /* in : number of complex points               */
int     data_type;      /* in : vax complex float                      */
                        /*    : sun complex floats, seperated storage  */

{ /*-----------------+
  ! Local parameters !
  +-----------------*/
  int           i;
  int           fd;
  int           num_bytes;
  int           n;
  int           compl_buf_len;  /* bytelenght of complex buffer        */
  complex       *cptr = COMPLEX_NULL_PTR;
                                /* complex buffer pointer              */

  DEB( "\nDATAOUT_write_data  (%s)", file );
  DEB( "\n buf = (%d,%d)   num = %d    datatype = %d",
                          re, im, num_complex, data_type);

  /*---------------+
  ! Open data file !
  +---------------*/
  fd = creat( file , 0744 );
  if( fd == -1 ) EXC( err_open );

  if( data_type == DATA_vax_complex_float ) {
    /*----------------------+
    ! Write VAX output file !
    +----------------------*/
    compl_buf_len =  num_complex * sizeof( complex );
    DEB("\ncompl_buf_len = %d", compl_buf_len );
    cptr = COMPLEX_PTR malloc( compl_buf_len );
    if( cptr == COMPLEX_NULL_PTR ) EXC( err_buf_allo );
    DEB("\n cptr = %d" , cptr);
    /*---------------------------+
    ! Copy data to complex buffer !
    +---------------------------*/
    for( i=0; i<num_complex; i++ ) {
        cptr[i].re = re[i];
        cptr[i].im = im[i];
    }
    DATACONV( CHAR_PTR cptr, 2*num_complex , "sun_f", CHAR_PTR cptr, "vax_f" );
    n = write( fd, cptr, compl_buf_len );
    free( cptr );
    if( n != compl_buf_len ) EXC( err_too_few );
 }else if( data_type == DATA_sun_complex_float_sep ) {
    /*----------------------+
    ! Write SUN output file !
    +----------------------*/
    num_bytes = num_complex * sizeof( float );
```

```
    n = write( fd, re, num_bytes );
    if( n != num_bytes ) EXC( err_too_few );
    n = write( fd, im, num_bytes );
    if( n != num_bytes ) EXC( err_too_few );
  }else{
    close( fd );
    EXC( err_data_type );
  }
  /*-----------+
  ! Close file !
  +-----------*/
  close( fd );
  /*-------------+
  ! Routine exit !
  +-------------*/
  DEB( "\nDATAOUT_write_data    exit" );
}
```



```
/*-----------------------------------------------------------------+
!       r5_ext                                                     !
!                                                                  !
+-----------------------------------------------------------------*/
static          int             /* return : nothing              */

r5_ext( str , ext )

char    *str;           /* out : r5 extension string                  */
int     ext;            /*  in : extension number                     */

{  DEB("\nr5_ext  ext=%d  (DATAOUT)", ext );
  /*-------------------------+
  ! Convert extension number !
  +-------------------------*/
  switch( ext ) {
    case DATA_ext_volt  : strcpy( str , "[V]"          );      break;
    case DATA_ext_mvolt : strcpy( str , "[mV]"         );      break;
    case DATA_ext_v_hz  : strcpy( str , "[V/Hz]"       );      break;
    case DATA_ext_rad   : strcpy( str , "[Rad]"        );      break;
    case DATA_ext_time  : strcpy( str , "[sec]"        );      break;
    case DATA_ext_hz    : strcpy( str , "[Hz]"         );      break;
    case DATA_ext_ppm   : strcpy( str , "[ppm]"        );      break;
    case DATA_ext_num   : strcpy( str , "[index]"      );      break;
    default             : strcpy( str , "???"          );      break;
  }
  /*-------------+
  ! Routine exit !
  +-------------*/
  DEB("\nr5_ext  ext=(%s)  exit   DATAOUT)", str );
}
```

```
/*-----------------------------------------------------------------+
!       DATAOUT_r5_ext                                            !
!                                                                 !
+-----------------------------------------------------------------*/
int             /* return : nothing             */

DATAOUT_r5_ext( str , ext )

char    *str;           /* out : r5 extension string                */
int     *ext;           /*  in : extension number                   */

{ DEB("\nDATAOUT_r5_ext  ext=(%s)", str );
  /*-----------------------------------+
  ! Convert string to extension number !
  +-----------------------------------*/
  if( strncmp( "[V]", str, strlen( "[V]" ) ) == 0 )
        *ext = DATA_ext_volt;
  else if( strncmp( "[V/Hz]", str, strlen( "[V/Hz]" )  ) == 0 )
        *ext = DATA_ext_v_hz;
  else if( strncmp( "[Rad]", str, strlen( "[Rad]" ) ) == 0 )
        *ext = DATA_ext_rad;
  else if( strncmp( "[Sec]", str, strlen( "[Sec]" ) ) == 0 )
        *ext = DATA_ext_time;
  else if( strncmp( "[sec]", str, strlen( "[sec]" ) ) == 0 )
        *ext = DATA_ext_time;
  else if( strncmp( "[Hz]", str, strlen( "[Hz]" ) ) == 0 )
        *ext = DATA_ext_hz;
  else if( strncmp( "[ppm]", str, strlen( "[ppm]" ) ) == 0 )
        *ext = DATA_ext_ppm;
  else if( strncmp( "[index]", str, strlen( "[index]" ) ) == 0 )
        *ext = DATA_ext_num;
  else
        *ext = DATA_ext_unknown;
  /*-------------+
  ! Routine exit !
  +-------------*/
  DEB("\nDATAOUT_r5_ext  ext=%d  exit   (DATAOUT)", *ext );
}
```



```
/*MPF ::: FUNCTIONS :: DATAOUT : TRANSLATE_EXTENSION_NUMBER_TO_STRING ==========

    FUNCTION NAME
        DATAOUT_r6_ext

    DESCRIPTION
        Translates extension number to string.

    ANYOTHER
        Author          : Ad Marien
        Creation Date   : 1989-03-02

    CALLING SEQUENCE
*/
int                     /* return : nothing             */

DATAOUT_r6_ext( str , ext )

char    *str;           /* out : r6 extension string                */
int     ext;            /*  in : extension number                   */

{  /* DEB("\nDATAOUT_r6_ext  ext=%d", ext ); */
  /*-------------------------+
  ! Convert extension number !
  +-------------------------*/
  switch( ext ) {
    case DATA_ext_volt          : strcpy( str , DATA_name_volt );       break;
    case DATA_ext_mvolt         : strcpy( str , DATA_name_mvolt );      break;
    case DATA_ext_v_hz          : strcpy( str , DATA_name_v_hz );       break;
    case DATA_ext_rad           : strcpy( str , DATA_name_rad );        break;
    case DATA_ext_time          : strcpy( str , DATA_name_time );       break;
    case DATA_ext_hz            : strcpy( str , DATA_name_hz );         break;
    case DATA_ext_ppm           : strcpy( str , DATA_name_ppm );        break;
    case DATA_ext_num           : strcpy( str , DATA_name_num );        break;
    case DATA_direction_mu1     : strcpy( str , DATA_name_mu1 );        break;
    case DATA_direction_kx      : strcpy( str , DATA_name_kx );         break;
    case DATA_direction_ky      : strcpy( str , DATA_name_ky );         break;
    case DATA_direction_kz      : strcpy( str , DATA_name_kz );         break;
    default                     : strcpy( str , DATA_name_unknown );    break;
  }
  /*-------------+
  ! Routine exit !
  +-------------*/
  /* DEB("\nDATAOUT_r6_ext  ext=(%s)  exit", str ); */
}
```

```
/*-------------------------------------------------------------+
!       r6_dim_write                                          !
!                                                             !
+-------------------------------------------------------------*/
static          int             /* return : nothing          */

r6_dim_write( fc, dim_ptr , dim_num )

FILE            *fc;            /* in : file pointer         */
DIM_STRUCT      *dim_ptr;       /* in : dimension parameters */
int             dim_num;        /* in : dimension number     */

{ /*----------------+
  ! Local parameter !
  +----------------*/
  char  ext_str[DEF_STR_LEN];
  char  dir_str[DEF_STR_LEN];

  DEB("\nr6_dim_write  dim=%d  ptr=%d (DATAOUT)", dim_num , dim_ptr );

  DATAOUT_r6_ext( ext_str , dim_ptr->ext );
  DATAOUT_r6_ext( dir_str , dim_ptr->direction );
  /*---------------------------+
  ! Write dimension parameters !
  +---------------------------*/
  switch( dim_num ) {
    case  1 :
        fprintf( fc, "\n%s       : %s", SUN_DIM1_EXT , ext_str );
        fprintf( fc, "\n%s      : %d", SUN_DIM1_PNTS , dim_ptr->pnts );
        fprintf( fc, "\n%s   : %f", SUN_DIM1_LOW_VAL , dim_ptr->low_val );
        fprintf( fc, "\n%s      : %f", SUN_DIM1_STEP , dim_ptr->step );
        fprintf( fc, "\n%s : %s", SUN_DIM1_DIRECTION , dir_str );
        fprintf( fc, "\n%s  : %f", SUN_DIM1_T0_POINT , dim_ptr->t0_point );
        break;
    case  2 :
        fprintf( fc, "\n%s       : %s", SUN_DIM2_EXT , ext_str );
        fprintf( fc, "\n%s      : %d", SUN_DIM2_PNTS , dim_ptr->pnts );
        fprintf( fc, "\n%s   : %f", SUN_DIM2_LOW_VAL , dim_ptr->low_val );
        fprintf( fc, "\n%s      : %f", SUN_DIM2_STEP , dim_ptr->step );
        fprintf( fc, "\n%s : %s", SUN_DIM2_DIRECTION , dir_str );
        fprintf( fc, "\n%s  : %f", SUN_DIM2_T0_POINT , dim_ptr->t0_point );
        break;
    case  3 :
        fprintf( fc, "\n%s       : %s", SUN_DIM3_EXT , ext_str );
        fprintf( fc, "\n%s      : %d", SUN_DIM3_PNTS , dim_ptr->pnts );
        fprintf( fc, "\n%s   : %f", SUN_DIM3_LOW_VAL , dim_ptr->low_val );
        fprintf( fc, "\n%s      : %f", SUN_DIM3_STEP , dim_ptr->step );
        fprintf( fc, "\n%s : %s", SUN_DIM3_DIRECTION , dir_str );
        fprintf( fc, "\n%s  : %f", SUN_DIM3_T0_POINT , dim_ptr->t0_point );
        break;
    case  4 :
        fprintf( fc, "\n%s       : %s", SUN_DIM4_EXT , ext_str );
        fprintf( fc, "\n%s      : %d", SUN_DIM4_PNTS , dim_ptr->pnts );
        fprintf( fc, "\n%s   : %f", SUN_DIM4_LOW_VAL , dim_ptr->low_val );
        fprintf( fc, "\n%s      : %f", SUN_DIM4_STEP , dim_ptr->step );
        fprintf( fc, "\n%s : %s", SUN_DIM4_DIRECTION , dir_str );
        fprintf( fc, "\n%s  : %f", SUN_DIM4_T0_POINT , dim_ptr->t0_point );
default :
        break;
  }
  /*-------------+
  ! Routine exit !
  +-------------*/
  DEB("\nr6_dim_write    exit    (DATAOUT)" );
}
```



```
/*MPF ::: FUNCTIONS :: DATAOUT : WRITE_DATA_PARAMETERS_TO_FILE ==========

    FUNCTION NAME
        DATAOUT_write_pars

    DESCRIPTION
        Write data parameters to file.

    ANYOTHER
        Author          : Ad Marien
        Creation Date   : 1989-03-02

    CALLING SEQUENCE
*/
int             /* return : nothing                          */

/* jane 940716
DATAOUT_write_pars(file, ptr ,row_start, row_stop, row_max ,r5_pars )
*/
DATAOUT_write_pars(file, ptr ,row_start, row_stop, row_max ,r5_pars, spec_sv_row_tot
al, specific_sv_row_num, specific_sv_row_type )

char            *file;          /* in : parameter filename           */
DATA_STRUCT     *ptr;           /* in : data structure pointer       */
int             row_start;      /* in : row start                    */
int             row_stop;       /* in : row stop range               */
int             row_max;        /* in : maximum row number           */
bool            r5_pars;        /* in : TRUE = write also r5 pars    */
/* jane 940716 */
int             spec_sv_row_total;     /* in : roi save row total     */
int             specific_sv_row_num[];   /* in : roi save row number  */
char            specific_sv_row_type[][6];    /* in : roi save row type */

{ /*-----------------+
  ! Local parameters !
  +-----------------*/
  int           col;
  float         val;
  int           rows;
  int           i;
  double        sampl_freq;
  FILE          *fc = NULL;
  char          str[DEF_STR_LEN];
  char          sep_str[DEF_STR_LEN];
  DIM_STRUCT    dim;

  DEB( "\nDATAOUT_write_pars  (%s) ptr=%d", file , ptr );
  DEB("\n row_range = (%d,%d) max = %d   r5=%d",
        row_start, row_stop, row_max , r5_pars );

  /*--------------------+
  ! Open parameter file !
  +--------------------*/
  fc = fopen( file , "w" );
  if( fc == NULL ) EXC( err_open );

  strcpy( sep_str ,
        "\n!-------------------------------------------------------------------");
  /*----------------------------+
  ! Write SUN format parameters !
  +----------------------------*/
  fprintf( fc, "Outputfile_from : %s", SUN_RELEASE );
  fprintf( fc, sep_str );
  /*--------------------------+
  ! Write resonance frequency !
  +--------------------------*/
  if( ptr->resfr_exist == TRUE ) {
    fprintf( fc, "\n%s : %d", SUN_RESONANCE_FREQUENCY ,  (int)ptr->resfr );
  }
  /*-----------------------------+
  ! Write sample value extension !
  +-----------------------------*/
```

```
DATAOUT_r6_ext( str , ptr->val_ext );
fprintf( fc, "\n%s : %s", SUN_VALUE_EXTENSION , str );

if( row_start == 1 && row_stop == row_max ) {
  /*-------------------------------+
  ! Write all dimension parameters !
  +-------------------------------*/
  fprintf( fc, "\n%s  : %d", SUN_NUM_DIMENSIONS , ptr->num_dim );
  for( i=0; i<ptr->num_dim; i++ ) {
      fprintf( fc, sep_str );
      r6_dim_write( fc, ptr->dim+i , i+1 );
  }
}else{
  /*-------------------------------+
  ! Write 2D reduced parameterset !
  +-------------------------------*/
  rows = row_stop - row_start + 1;
  if( rows == 1 ) fprintf( fc, "\n%s  : 1", SUN_NUM_DIMENSIONS );
  else            fprintf( fc, "\n%s  : 2", SUN_NUM_DIMENSIONS );
  /*-----------------------+
  ! Write first dimension !
  +-----------------------*/
  fprintf( fc, sep_str );
  r6_dim_write( fc, ptr->dim , 1 );
  /*-------------------------------+
  ! Write updated second dimension !
  +-------------------------------*/
  if( rows != 1 ) {
      fprintf( fc, sep_str );
      DATAGEN_byte_copy( &dim , ptr->dim+1, sizeof( DIM_STRUCT ) );
      dim.low_val = dim.low_val - (row_start-1) * dim.step;
      dim.pnts    = rows;
      r6_dim_write( fc, &dim, 2 );
  }
}

/*-----------------------------+
! Write Gyroscan R5 parameters !
+-----------------------------*/
if( r5_pars == TRUE ) {
      fprintf( fc, sep_str );
      fprintf( fc, "\n!--------------------------------------------+" );
      fprintf( fc, "\n! Gyroscan parameters for preprocessed data !" );
      fprintf( fc, "\n!--------------------------------------------+" );
      if( ptr->resfr_exist == TRUE ) {
        fprintf( fc, "\n%s  : %d", GYRO_SYNT_FREQ, (int)ptr->resfr );
      }
      fprintf( fc, "\n%s                : %d" ,
                   GYRO_SAMPLES, ptr->dim[0].pnts );
      /*--------------------------+
      ! Estimate sample frequency !
      +--------------------------*/
      sampl_freq = 1.0;
      for( i=0; i<ptr->num_dim; i++ ) {
        if( ptr->dim[i].direction == DATA_direction_mul ) {
            if( ptr->dim[i].ext == DATA_ext_time ) {
              sampl_freq = 1.0/ptr->dim[i].step;
            }else if( ptr->dim[i].ext == DATA_ext_hz ) {
              sampl_freq = ptr->dim[i].pnts * ptr->dim[i].step;
            }else if( ptr->dim[i].ext == DATA_ext_ppm ) {
              if( ptr->resfr_exist == TRUE )
                sampl_freq = ptr->dim[i].pnts * ptr->dim[i].step *
                             ptr->resfr/DATA_PPM;
            }
        }
      }
      if( sampl_freq < 0.0 ) sampl_freq *= -1.0;
      fprintf( fc, "\n%s       : %d" ,
                   GYRO_SAMPLE_FREQ,(int)sampl_freq );
      fprintf( fc, "\nspectro_preprocessed_parameters :" );
      fprintf( fc, "\n!---------------------------------------+" );
      fprintf( fc, "\n! Extra parameters for preprocessed data !" );
      fprintf( fc, "\n!---------------------------------------+" );
```



```
      fprintf( fc, "\n%s : cf" , GYRO_DATA_TYPE );
      r5_ext( str , ptr->val_ext );
      fprintf( fc, "\n%s : %s", GYRO_VALUE_EXT, str );
      fprintf( fc, "\n!-------------------------+" );
      fprintf( fc, "\n! Column parameters       !" );
      fprintf( fc, "\n!-------------------------+" );
      col = ptr->dim[0].pnts;
      fprintf( fc, "\n%s : %d", GYRO_COL_PNTS, col );
      fprintf( fc, "\n%s : %f", GYRO_COL_LOW, ptr->dim[0].low_val );
      fprintf( fc, "\n%s : %f", GYRO_COL_HIGH,
             ptr->dim[0].low_val + (float)((col)*ptr->dim[0].step ));
      r5_ext( str , ptr->dim[0].ext );
      fprintf( fc, "\n%s : %s", GYRO_COL_EXT, str );
      fprintf( fc, "\n!-------------------------+" );
      fprintf( fc, "\n! Row parameters          !" );
      fprintf( fc, "\n!-------------------------+" );
      fprintf( fc, "\n%s : %d", GYRO_ROW_PNTS, (int)(row_stop-row_start+1) );
      val = ptr->dim[1].low_val + (row_start-1) * ptr->dim[1].step;
      fprintf( fc, "\n%s : %f",  GYRO_ROW_LOW, val );
      val = ptr->dim[1].low_val + (row_stop-1) * ptr->dim[1].step;
      fprintf( fc, "\n%s : %f",  GYRO_ROW_HIGH, val );
      r5_ext( str , ptr->dim[1].ext );
      fprintf( fc, "\n%s : %s", GYRO_ROW_EXT, str );

/* jane 940716
      fprintf( fc, "\n!-------------------------+" );
      fprintf( fc, "\n! Selected ROI row and type  !" );
      fprintf( fc, "\n!-------------------------+" );
      ptr->dim[1].specific_save_row_total = spec_sv_row_total;
      fprintf( fc, "\n%s : %d", GYRO_ROI_ROWS_TOT, ptr->dim[1].specific_save_row_t
otal);

      for ( i=0; i<spec_sv_row_total; i++)
      {
         ptr->dim[1].specific_save_row_array[i] = specific_sv_row_num[i];
         strcpy(ptr->dim[1].specific_save_row_type[i], specific_sv_row_type[i]);

         fprintf( fc, "\n%s : %4d, %s : %s ", GYRO_ROW_SAVE, ptr->dim[1].specific_
save_row_array[i], GYRO_ROW_TYPE, ptr->dim[1].specific_save_row_type[i]);
         DEB( "\n%s : %4d-%d ; %s : %4d, %s : %s ", ptr->dim[1].specific_save_row_
total, i, GYRO_ROW_SAVE, ptr->dim[1].specific_save_row_array[i], GYRO_ROW_TYPE, ptr->
dim[1].specific_save_row_type[i]);
      }
      fprintf( fc, "\n");
 jane 940716 */

/* jane 940717 */
      fprintf( fc, "\n!-------------------------+" );
      fprintf( fc, "\n! Selected ROI row and type  !" );
      fprintf( fc, "\n!-------------------------+" );
      fprintf( fc, "\n%s : %d", GYRO_ROI_ROWS_TOT, spec_sv_row_total);

      for ( i=0; i<spec_sv_row_total; i++)
      {
         fprintf( fc, "\n%s : %4d, %s : %s ", GYRO_ROW_SAVE, specific_sv_row_num[i
], GYRO_ROW_TYPE, specific_sv_row_type[i]);
         DEB("\n%s : %4d, %s : %s ", GYRO_ROW_SAVE, specific_sv_row_num[i], GYRO_R
OW_TYPE, specific_sv_row_type[i]);
      }
      fprintf( fc, "\n");
/* jane 940717 */


      fprintf( fc, "\n!-------------------------+" );
      fprintf( fc, "\n! Processing commands     !" );
      fprintf( fc, "\n!-------------------------+" );
      fprintf( fc, "\nbegin_commands :" );
      fprintf( fc, "\nend_commands   :" );
  }
/* jane 940717 */
  else {
      fprintf( fc, "\n!-------------------------+" );
      fprintf( fc, "\n! Selected ROI row and type  !" );
```

```
        fprintf( fc, "\n!---------------------------+" );
        fprintf( fc, "\n%s : %d", GYRO_ROI_ROWS_TOT, spec_sv_row_total);

        for ( i=0; i<spec_sv_row_total; i++)
        {
           fprintf( fc, "\n%s : %4d, %s : %s ", GYRO_ROW_SAVE, specific_sv_row_num[i
], GYRO_ROW_TYPE, specific_sv_row_type[i]);
           DEB("\n%s : %4d, %s : %s ", GYRO_ROW_SAVE, specific_sv_row_num[i], GYRO_R
OW_TYPE, specific_sv_row_type[i]);
        }
        fprintf( fc, "\n");
  }
/* jane 940717 */


  /*-----------+
  ! Close file !
  +-----------*/
  fclose( fc );
  /*-------------+
  ! Routine exit !
  +-------------*/
  DEB( "\nDATAOUT_write_pars    exit" );
}
```

```
/*=====================================================================*/
/*                                                                     */
/*      Copyright 1989 N.V. Philips' Gloeilampen Fabrieken.            */
/*      All rights reserved. Reproduction in whole or in part is       */
/*      prohibited without the written consent of the copyright owner. */
/*                                                                     */
/*=====================================================================*/

/*-------------------+
! System definitions !
+-------------------*/
# include <stdio.h>
# include <string.h>
/*-----------------+
! User definitions !
+-----------------*/
# include "specglo.h"
# include "exception.h"
# include "debug.h"
# include "readpar.h"
# include "str_func.h"

# define        MAX_LEN         202
# define        SKIP_LEADING    while( *line == ' ' || *line == '\t' ) line++;
# define        SKIP_CONT       if( skip_comment_end() == '-' ) read_line();
# define        SKIP_NAME       while( *line != ':' && *line != '\0' ) line++;

static  FILE            *fc = NULL;
static  int             line_count;
static  char            line_str[MAX_LEN];
static  char            *line;

static char     err_open[] = "E: open parameter file (READPAR)";
```



```
/*MPF ::: FUNCTIONS :: READPAR.C : PRINT_READ_PARAMETERS ==========

    FUNCTION NAME
        READPAR_print_str

    DESCRIPTION
        Prints the string parameters.

    EXCEPTIONS

    ANYOTHER
        Author          : Ad Marien
        Creation Date   : 1989-02-25

    CALLING SEQUENCE
*/
int                             /* return : none                */

READPAR_print_str( str_arr )

PAR_STR_TYPE    str_arr[];

/*EMP=================================================================*/

{ /*----------------+
  ! Local variables !
  +----------------*/
  int i = 0;

  DEB( "\n===> Read string parameters <===" );
  if( str_arr != (PAR_STR_TYPE *) 0 ) {
    while( str_arr[i].par_name[0] != '\0' ) {
      if( str_arr[i].par_found == TRUE ) {
        DEB( "\n%s (%s)", str_arr[i].par_name , str_arr[i].val_ptr );
      }else{
        DEB( "\nparameter =>%s<= not found", str_arr[i].par_name );
      }
      i++;
    }
  }
}
```

```
/*MPF ::: FUNCTIONS :: READPAR.C : PRINT_READ_PARAMETERS ==========

    FUNCTION NAME
        READPAR_print_arr

    DESCRIPTION
        Prints the double value arrays

    EXCEPTIONS

    ANYOTHER
        Author          : Ad Marien
        Creation Date   : 1989-02-25

    CALLING SEQUENCE
*/
int                             /* return : none                */

READPAR_print_arr( d_arr )

PAR_DOUBLE_TYPE d_arr[];

/*EMP=============================================================================*/

{ /*----------------+
  ! Local variables !
  +----------------*/
  int   i = 0;
  int   j;

  DEB( "\n===> Read array parameters <===" );
  if( d_arr != (PAR_DOUBLE_TYPE *) 0 ) {
    while( d_arr[i].par_name[0] != '\0' ) {
      if( d_arr[i].par_found == TRUE ) {
        if( d_arr[i].num_val <= 0 ) {
          DEB( "\n%s : no values read", d_arr[i].par_name );
        }else{
          DEB( "\n%s : %15.6e", d_arr[i].par_name , d_arr[i].arr_ptr[0] );
          for( j=1; j<d_arr[i].num_val; j++ )
                DEB( "\n %3d : %15.6e", j , d_arr[i].arr_ptr[j] );
        }
      }else{
        DEB( "\nparameter =>%s<= not found", d_arr[i].par_name );
      }
      i++;
    }
  }
}
```



```
/*-------------------------------------------------------------------------+
!       -- skip_comment_end                                                !
!                                                                         !
+-------------------------------------------------------------------------*/
static char             /* return : last string character or '\0'  */

skip_comment_end()

{ /*----------------+
  ! Local parameters !
  +----------------*/
  char  *ptr = line;

  /*-------------------------+
  ! Search for comment part !
  +-------------------------*/
  while( *ptr != '!' && *ptr != '\0' ) *ptr++;
  if( *ptr == '!' ) *ptr = '\0';
  /*----------------------------+
  ! Skip ending spaces and tabs !
  +----------------------------*/
  ptr--;
  while( ptr >= line && ( *ptr == ' ' || *ptr == '\t' ) ) ptr--;
  *(ptr+1) = '\0';
  if( ptr < line ) return( '\0' );
  else             return( *ptr );
}
```

```
/*-------------------------------------------------------------------------+
!       -- read_line                                                       !
!                                                                         !
+-------------------------------------------------------------------------*/
static int              /* return : nothing                             */

read_line()

{ /*-----------------+
  ! Local parameters !
  +-----------------*/
  int   len;

  /*---------------------------+
  ! Read next non comment line !
  +---------------------------*/
  do{
    /*--------------------+
    ! Read next file line !
    +--------------------*/
    line_count++;
    STR_FUNC_read_line( fc, line_str , MAX_LEN );
    /*----------------------------+
    ! Skip leading spaces and tabs !
    +----------------------------*/
    line = line_str;
    SKIP_LEADING
  } while( *line == '!' );
  /*-------------+
  ! Routine exit !
  +-------------*/
  DEB( "\n%d(%s)", line_count , line );
}
```



```
/*-------------------------------------------------------------------------+
!       -- read_str                                                        !
!                                                                         !
! Parameter file format :                                                 !
!                                                                         !
! <parameter_name> : <parameter_value> <comment> <eol>                    !
! <parameter_name> : -                 <comment> <eol>                    !
!                    <parameter_value> <comment> <eol>                    !
!                                                                         !
! <comment> = <empty string> || '!'<string>                               !
!                                                                         !
! Possible begin and end quote (") are removed from the parameter value   !
! string.                                                                 !
+-------------------------------------------------------------------------*/
static int              /* return : nothing                             */

read_str( str_arr , ind )

PAR_STR_TYPE    str_arr[];
int             ind;

{ /*-----------------+
  ! Local parameters !
  +-----------------*/
  int   len;
  char  last_char;

  DEB( "\nRead_str  (%s) (%d)", str_arr[ind].par_name , ind );

  /*--------------------+
  ! Skip parameter name !
  +--------------------*/
  SKIP_NAME
  /*  DEB( "\n1 (%s)", line ); */
  /*---------------------------+
  ! Check on separation sign !
  +---------------------------*/
  if( *line == ':' ) {
    /*-----------------------------------------------------------+
    ! Skip separation sign, leading and ending spaces and tabs !
    +-----------------------------------------------------------*/
    line++;
    SKIP_LEADING
    /* DEB( "\n2 (%s)", line ); */
    last_char = skip_comment_end();
    /*----------------------------+
    ! Check on continuation sign !
    +----------------------------*/
    /* DEB( "\n3 (%s)", line ); */
    if( strcmp( line , "-" ) == 0 ) {
      read_line();
      last_char = skip_comment_end();
    }
    /*------------------------+
    ! Skip possible end quote !
    +------------------------*/
    if( last_char == '"' ) {
        len = strlen( line );
        line[ len-1 ] = '\0';
        skip_comment_end();
    }
    /*--------------------------+
    ! Skip possible start quote !
    +--------------------------*/
    if( *line == '"' ) {
        line++;
        SKIP_LEADING
    }
    /*------------------+
    ! Store string part !
    +------------------*/
    /* DEB( "\n4 (%s)", line ); */
```

```
    strncpy( str_arr[ind].val_ptr , line , str_arr[ind].max_len-1 );
    str_arr[ind].val_ptr[ str_arr[ind].max_len-1 ] ='\0';
    /* DEB( "\n5 (%s)", str_arr[ind].val_ptr ); */
    str_arr[ind].num_char = strlen( line );
  }else{
    DEB( "\nE: No ':' sign for %s ( line = %d )",
                      str_arr[ind].par_name , line_count );
  }
  /*-------------+
  ! Routine exit !
  +-------------*/
  DEB( "\n(%s)", str_arr[ind].val_ptr );
  DEB( "\nRead_str    num_char = %d   exit", str_arr[ind].num_char );
}
```



```
/*-------------------------------------------------------------------------+
!       -- read_d                                                          !
!                                                                          !
! Parameter file format :                                                  !
!                                                                          !
! <parameter_name> : <parameter_values>    <comment> <eol>                 !
! <parameter_name> : <parameter_values>,- <comment> <eol>                  !
!                    .............                                         !
!                    <parameter_values>,- <comment> <eol>                  !
!                    <parameter_values>    <comment> <eol>                 !
! <parameter_name> : - <comment> <eol>                                     !
!                    .............                                         !
!                    <parameter_values>,- <comment> <eol>                  !
!                    <parameter_values>    <comment> <eol>                 !
!                                                                          !
! <parameters_values > =                                                   !
!       <parameter_value> ||                                               !
!       (number) <parameter_value> ||                                      !
!       <parameter_values> , <parameter_values>                            !
!                                                                          !
! <comment> = <empty string> || '!' <string>                               !
+-------------------------------------------------------------------------*/
static int              /* return : nothing                    */

read_d( d_arr , ind )

PAR_DOUBLE_TYPE d_arr[];
int             ind;

{ /*-----------------+
  ! Local parameters !
  +-----------------*/
  int           mult;
  int           j;
  int           max_pars;
  float         temp_fl;
  bool          error = FALSE;
  bool          stop  = FALSE;

  DEB( "\nRead_d  (%s) (%d)", d_arr[ind].par_name, ind );

  /*--------------------------------+
  ! Set number of values to be read !
  +--------------------------------*/
  if( d_arr[ind].wanted <= 0 ) {
    max_pars = d_arr[ind].arr_len;
  }else{
    max_pars = d_arr[ind].wanted;
    if( max_pars > d_arr[ind].arr_len ) max_pars = d_arr[ind].arr_len;
  }

  /*-------------------------+
  ! Skip parameter name part !
  +-------------------------*/
  SKIP_NAME
  /* DEB( "\n1 (%s)", line ); */

  /*-------------------------+
  ! Check on separation sign !
  +-------------------------*/
  if( *line == ':' ) {
    /*-----------------------------------------------------------+
    ! Skip separation sign, leading and ending spaces and tabs !
    +-----------------------------------------------------------*/
    line++;
    SKIP_LEADING
    /* DEB( "\n2 (%s)", line ); */
    skip_comment_end();
    /*---------------------------+
    ! Check on continuation sign !
    +---------------------------*/
    /* DEB( "\n3 (%s)", line ); */
```

```
    if( strcmp( line , "-" ) == 0 ) read_line();

    do{
      /*-----------------------------+
      ! Skip leading spaces and tabs !
      +-----------------------------*/
      SKIP_LEADING
      /* DEB( "\n2 (%s)", line ); */
      /*--------------------------+
      ! Check on mutilply factor !
      +--------------------------*/
      if( *line == '(' ) {
        if( sscanf( line , "( %d ) %f", &mult , &temp_fl ) == 2 ) {
          /*----------------------+
          ! Store multiple values !
          +----------------------*/
          /* DEB( "\n mult=%d   val=%15.6e", mult , temp_fl ); */
          j = 0;
          while( j<mult && d_arr[ind].num_val < max_pars ) {
            d_arr[ind].arr_ptr[ d_arr[ind].num_val ] = temp_fl;
            d_arr[ind].num_val++;
            j++;
          }
          /* DEB( "\n5 (%s)", line ); */
        }else{
          error = TRUE;
          DEB( "\nE: multiple value error " );
        }
      }else{
        /*-------------+
        ! Single value !
        +-------------*/
        if( sscanf( line, " %f", &temp_fl ) == 1 ) {
          /*--------------------+
          ! Store single value !
          +--------------------*/
          /* DEB( "\n6 single value = %15.6e", temp_fl ); */
          if( d_arr[ind].num_val < max_pars ) {
            d_arr[ind].arr_ptr[ d_arr[ind].num_val ] = temp_fl;
            d_arr[ind].num_val++;
            /* DEB( "\n7 (%s)", line ); */
          }else{
            error = TRUE;
          }
        }else{
          error = TRUE;
          DEB( "\nE: single value conversion error" );
        }
      }
      if( error == FALSE && d_arr[ind].num_val < max_pars ) {
        /*----------------------+
        ! Skip interpreted part !
        +----------------------*/
        while( *line != ',' && *line != '!' && *line != '\0' ) line++;
        switch( *line ) {
          case '\0' : stop = TRUE;
                      break;
          case '!'  : stop = TRUE;
                      break;
          case ','  : line++;
                      /*---------------------------+
                      ! Check on continuation sign !
                      +---------------------------*/
                      SKIP_LEADING
                      if( *line == '-' ) {
                        skip_comment_end();
                        if( strcmp( line , "-" ) == 0 ) read_line();
                      }
                      break;
          default   : error = TRUE;
                      break;
        }
      }
```



```
      } while( d_arr[ind].num_val < max_pars && stop == FALSE && error == FALSE );
      /*-------------------------------+
      ! Skip non used parameter lines !
      +-------------------------------*/
      if( error == FALSE ) SKIP_CONT
    }else{
      DEB( "\nE: No ':' sign" );
    }
    /*-------------+
    ! Routine exit !
    +-------------*/
    DEB( "\nRead_d  %d values read.   Exit", d_arr[ind].num_val);
}
```

```
/*MPF ::: LOGGING :: READPAR : READ_PARAMETER_FILE ============

    FUNCYION NAME
        READPAR

    PACKAGE

    DESCRIPTION
        Reads the requested parameters out of the parameter file. The parameter
        value strings are filled with the file contents.

    EXCEPTIONS
        Exceptions raised :
        - E: parameterfile could not be opened

    ANYOTHER
        Author          : Ad Marien
        Creation Date   : 1985-02-25

    CALLING SEQUENCE
*/

int                     /* return : none                        */

READPAR( file , d_arr , str_arr )

char            file[];         /* in  : name of parameter file         */
PAR_DOUBLE_TYPE d_arr[];
PAR_STR_TYPE    str_arr[];

/*EMP=================================================================*/

{ /*----------------+
  ! Local variables !
  +----------------*/
  int           i;
  int           num_str_pars = 0;
  int           num_d_pars = 0;
  int           max_str_pars = 0;
  int           max_d_pars = 0;
  bool          d_par_found;
  bool          str_par_found;

  DEB( "\nREADPAR (%s)", file );

  /*--------------------+
  ! Open parameter file !
  +--------------------*/
  fc = fopen( file , "r" );
  if( fc == NULL ) EXC( err_open );

  /*-----------------------------------+
  ! Reset all string parameter values !
  +-----------------------------------*/
  if( str_arr != (PAR_STR_TYPE *) 0 ) {
    i = 0;
    while( str_arr[i].par_name[0] != '\0' ) {
      str_arr[i].name_len   = strlen( str_arr[i].par_name );
      str_arr[i].par_found  = FALSE;
      str_arr[i].val_ptr[0] = '\0';
      str_arr[i].num_char   = 0;
      i++;
    }
    max_str_pars = i;
  }else{
    max_str_pars = 0;
  }
  DEB( "\nMax_str_pars = %d", max_str_pars );
  /*-----------------------------------------+
  ! Reset all double float parameter values !
  +-----------------------------------------*/
  if( d_arr != (PAR_DOUBLE_TYPE *) 0 ) {
```



```
    i = 0;
    while( d_arr[i].par_name[0] != '\0' ) {
      d_arr[i].name_len   = strlen( d_arr[i].par_name );
      d_arr[i].par_found = FALSE;
      d_arr[i].num_val   = 0;
      i++;
    }
    max_d_pars = i;
  }else{
    max_d_pars = 0;
  }
  DEB( "\nMax_d_pars = %d", max_d_pars );

  /*----------------+
  ! Read parameters !
  +----------------*/
  line_count = 0;
  E_begin
    while( num_str_pars < max_str_pars || num_d_pars < max_d_pars ) {
      d_par_found   = FALSE;
      str_par_found = FALSE;
      read_line();
      /*---------------------------+
      ! Check for string parameter !
      +---------------------------*/
      if( num_str_pars < max_str_pars ) {
        i = 0;
        while( i < max_str_pars && str_par_found == FALSE) {
          if( str_arr[i].par_found == FALSE &&
              strncmp( str_arr[i].par_name, line, str_arr[i].name_len)==0 ) {
            /*------------------+
            ! Check name length !
            +------------------*/
            switch( line[str_arr[i].name_len]  ) {
            case      ':'  :
            case      ' '  :
            case      '\0' :
            case      '\t' : str_arr[i].par_found = TRUE;
                             str_par_found = TRUE;
                             num_str_pars++;
                             read_str( str_arr , i );
                             break;
            default        : i++;
                             break;
            }
          }else{
            i++;
          }
        }
      }
      /*---------------------------------+
      ! Check for double array parameter !
      +---------------------------------*/
      if( str_par_found == FALSE && num_d_pars < max_d_pars ) {
        i = 0;
        while( i < max_d_pars && d_par_found == FALSE ) {
          if( d_arr[i].par_found == FALSE &&
              strncmp( d_arr[i].par_name , line , d_arr[i].name_len ) == 0 ){
            /*------------------+
            ! Check name length !
            +------------------*/
            switch( line[d_arr[i].name_len]  ) {
            case      ':'  :
            case      ' '  :
            case      '\0' :
            case      '\t' : d_arr[i].par_found = TRUE;
                             d_par_found = TRUE;
                             num_d_pars++;
                             read_d( d_arr , i );
                             break;
            default        : i++;
                             break;
            }
```

```
            }else{
              i++;
            }
          }
        }
        /*------------------------+
        ! Skip continuation lines !
        +------------------------*/
        if( str_par_found == FALSE && d_par_found == FALSE ) SKIP_CONT
    }
    E_handle
    E_others
    Exc_end

    /*-----------+
    ! Close file !
    +-----------*/
    fclose( fc );

    /*-------------+
    ! Routine exit !
    +-------------*/
    if( DEB_is_on() == TRUE ) {
      READPAR_print_str( str_arr );
      READPAR_print_arr( d_arr );
      DEB( "\nREADPAR      exit.\n" );
    }
}
```

```
/*=================================================================*/
/*                                                                 */
/*      Copyright 1989 N.V. Philips' Gloeilampen Fabrieken.        */
/*      All rights reserved. Reproduction in whole or in part is   */
/*      prohibited without the written consent of the copyright owner. */
/*                                                                 */
/*=================================================================*/

/*-----------------------------+
! Measurement type definitions !
+-----------------------------*/
# define        MEAS_unknown            0
# define        MEAS_1_voi              10
# define        MEAS_2_voi              20
# define        MEAS_t1                 30
# define        MEAS_t2                 40
# define        MEAS_editing            50
# define        MEAS_1d_si              60
# define        MEAS_2d_si              70
# define        MEAS_3d_si              80

# define        MEAS_1_voi_str          "1 VOI measurement"
# define        MEAS_2_voi_str          "2 VOI measurement"
# define        MEAS_t1_str             "T1 measurement"
# define        MEAS_t2_str             "T2 measurement"
# define        MEAS_editing_str        "Editing measurement"
# define        MEAS_1d_si_str          "1D SI measurement"
# define        MEAS_2d_si_str          "2D SI measurement"
# define        MEAS_3d_si_str          "3D SI measurement"

# define        MAX_1D_SI_SLICE_SIZE    (1e+06)

# define        MAX_SPECIFIC_ROWS       20 /* jane 940708 */

/*--------------------------------+
! Spectroscopic image orientation !
+--------------------------------*/
# define        PLUSA_PLUSB             90
# define        PLUSA_MINB              91
# define        MINA_PLUSB              92
# define        MINA_MINB               93
# define        PLUSB_PLUSA             94
# define        PLUSB_MINA              95
# define        MINB_PLUSA              96
# define        MINB_MINA               97

/*--------------+
! ROI-structure !
+--------------*/
typedef struct {
        bool    roi_available;          /* TRUE = roi parameters are legal */
                                        /* else  FALSE  */
        float   ap_off_center;          /* [mm]                 */
        float   lr_off_center;          /* [mm]                 */
        float   cc_off_center;          /* [mm]                 */
        float   ap_roi_shift;           /* [mm]                 */
        float   lr_roi_shift;           /* [mm]                 */
        float   cc_roi_shift;           /* [mm]                 */
        float   ap_size;                /* [mm]                 */
        float   lr_size;                /* [mm]                 */
        float   cc_size;                /* [mm]                 */
        float   ap_angulation;          /* [degrees]            */
        float   lr_angulation;          /* [degrees]            */
        float   cc_angulation;          /* [degrees]            */
        int     ap_voxels;              /* number of ap voxels  */
        int     lr_voxels;              /* number of lr voxels  */
        int     cc_voxels;              /* number of cc voxels  */
        int     slice_orientation;      /* slice orientation    */
        int     spec_image_orient;      /* spec.image orientation*/
} ROI_STRUCT;

struct {
        int sel_row_total;
```



```
        int sel_row_array[MAX_SPECIFIC_ROWS];
} specific_rows;

/*---------------------+
! Double ROI structure !
+---------------------*/
typedef struct {
        int             meas_type;
        ROI_STRUCT      roi_1;
        ROI_STRUCT      roi_2;
} ROI2_STRUCT;

/*--------------------+
! External procedures !
+--------------------*/
extern  ROI2_STRUCT     *ROIIN_ptr();
```

```
/*=============================================================================*/
/*                                                                             */
/*      Copyright 1989 N.V. Philips' Gloeilampen Fabrieken.                    */
/*      All rights reserved. Reproduction in whole or in part is               */
/*      prohibited without the written consent of the copyright owner.         */
/*                                                                             */
/*=============================================================================*/
/*******************************************************************************/
/*                            module ROI_select.c                              */
/*                                                                             */
/* Author: P.v.Gerwen                                                          */
/* Date  : 1 june 1989                                                         */
/*******************************************************************************/

# include <stdio.h>
# include <string.h>
# include <suntool/sunview.h>
# include <suntool/panel.h>
# include <math.h>

# include "specglo.h"
# include "imageglo.h"
# include "exception.h"
# include "debug.h"
# include "data.h"
# include "roi.h"
# include "row_range.h"
# include "axis_layout.h"
# include "curve_layout.h"
# include "curve_window.h"
# include "roi_disp.h"
# include "image_param.h"

static  char err_mode[]  = "E: wrong mode (ROI_select)";
static  char err_row[]   = "E: wrong row number (ROI_select)";
static  char err_num[]   = "E: wrong number of rows (ROI_select)";
static  char err_allo[]  = "E: no dynamic memory (ROI_select)";

static int            *selected_rows = INT_NULL_PTR;

static int            *Global_selected_rows = INT_NULL_PTR;

static char           image_error[]      = "E: selection image not available";
static char           no_ROI_error[]     = "E: no ROI in selection image";
static char           invalid_ROI_msg[]  = "E: invalid ROI for selection";
static char           exceeded_row_msg[] = "E: rownumber out of range";
static char           invalid_num_rows[] = "E: invalid number of rows";
static char           empty_msg[]        = "";

static ROI_disp       ROI;
static Axis_layout    axis_layout;
static Curve_layout   curve_layout;
static Curve_window   curve_window;
static VIEW_STRUCT    data_pars;
static Row_range      *row_range = (Row_range *) 0;
static IMAGE_param    *image     = (IMAGE_param *) 0;

static bool           cursor_in_image[MAX_NUM_IMAGES];
static bool           cursor_displayed[MAX_NUM_IMAGES];
static Pixwin         *cursor_pw[MAX_NUM_IMAGES];
static int            cursor_x[MAX_NUM_IMAGES];
static int            cursor_y[MAX_NUM_IMAGES];

static Frame          selection_frame;
static Panel          selection_panel;
static Panel_item     selection_item;
static int            selection_image;
static Panel_item     image_item[MAX_NUM_IMAGES];
static bool           image_selected[MAX_NUM_IMAGES];
static int            selection_frame_x_pos      = 866;
static int            selection_frame_y_pos      = 22;
static char           pos_selection_frame_file[] = "ROI_select_selection.pos";
static bool           selection_first            = TRUE;
```



```
static Frame          frame;
static Panel          panel;
static Panel_item     num_rows_text;
static Panel_item     pos_x_text, pos_y_text, number_text;
static int            frame_x_pos      = 658;
static int            frame_y_pos      = 22;
static char           pos_frame_file[] = "roi_select.pos";
static bool           first            = TRUE;

static Pixwin         *curve_pw;

static float          *data_ptr;

static bool           erase_row;
static int            curve_num_rows = 5;
static int            prev_row;
static int            row_number;

static void           create_image_selection_panel();
static void           selection_frame_msg();
static void           save_selection_frame_location();
static void           destroy_selection_frame();
static void           ROI_select_proc();
static void           create_panel();
static void           frame_msg();
static void           calc_vertical_window();
static void           init_image_events();
static void           disable_image_events();
static void           start_row_selection();
static void           init_selected_row_array();
static void           adapt_selected_row_array();
static void           delete_selected_row_array();
static void           handle_event();
static void           calc_row_number();
static void           reorient_si();
static void           adapt_accompany_cursor();
static void           draw_accompany_cursor();
static void           write_curve();
static void           save_frame_location();
static void           proceed_proc();
static void           refresh_proc();
static void           remove_proc();
static void           cancel_proc();

extern float          *VIEW_data_ptr();

void average_selected_row_array();
void average_and_save_selected_row_array();

static int  Global_row_number;

ROI_select()
{
      DEB("\nROI_select  in");

      E_begin
        DATAVIEW_read_par(&data_pars);
      E_handle
      E_others
        return;
      Exc_end

      E_begin   SPECTRUM_number_remove();
      E_handle  E_others       Exc_end

      create_image_selection_panel();

      DEB("\nROI_select  exit");
}

/*******************************************************************************/
```

```
/* Initialization of the panel in which to select the image in which the      */
/* selection in the ROI will be made.                                         */
/*****************************************************************************/
static void create_image_selection_panel()
{
      char  string[DEF_STR_LEN];
      int   i;
      int   x_pos, y_pos;

      for (i = 0; i < MAX_NUM_IMAGES; i++) image_selected[i] = 1;

      /*-------------------------+
      ! Get saved frame position !
      +-------------------------*/
      if (selection_first == TRUE ) {
          selection_first = FALSE;
          E_begin
            READGEN_get_frame_pos(pos_selection_frame_file, &x_pos, &y_pos);
            selection_frame_x_pos = x_pos;
            selection_frame_y_pos = y_pos;
          E_handle
          E_others
          Exc_end
      }

      selection_frame = window_create(NULL, FRAME,
                        FRAME_EMBOLDEN_LABEL, TRUE,
                        FRAME_LABEL         , "Select image for ROI selection",
                        FRAME_NO_CONFIRM    , TRUE,
                        WIN_X               , selection_frame_x_pos,
                        WIN_Y               , selection_frame_y_pos,
                        WIN_MOUSE_XY        , selection_frame_x_pos + 100,
                                              selection_frame_y_pos + 50,
                        WIN_SHOW            , TRUE,
                        0);
      READGEN_window_change_frame_menu(selection_frame);

      selection_panel = window_create(selection_frame, PANEL,
                                      WIN_WIDTH , ATTR_COL(34),
                                      WIN_HEIGHT, ATTR_ROW(9),
                                      0);

      selection_item = panel_create_item(selection_panel, PANEL_CHOICE,
                       PANEL_ITEM_X        , ATTR_COL(3),
                       PANEL_ITEM_Y        , ATTR_ROW(0),
                       PANEL_LABEL_BOLD    , TRUE,
                       PANEL_LABEL_STRING  , "Image nr.:",
                       PANEL_CHOICE_STRINGS, " 1 ", " 2 ", " 3 ", " 4 ", 0,
                       PANEL_FEEDBACK      , PANEL_INVERTED,
                       0);

      panel_create_item(selection_panel, PANEL_MESSAGE,
                        PANEL_ITEM_X      , ATTR_COL(0),
                        PANEL_ITEM_Y      , ATTR_ROW(2),
                        PANEL_LABEL_BOLD  , TRUE,
                        PANEL_LABEL_STRING,
                        "Display an accompanying cursor in:",
                        0);

      for (i = 0; i < MAX_NUM_IMAGES; i++) {
          sprintf(string, "Image %1d       ", i+1);

          image_item[i] = panel_create_item(selection_panel, PANEL_CHOICE,
                          PANEL_ITEM_X        , ATTR_COL(6),
                          PANEL_ITEM_Y        , ATTR_ROW(3+i),
                          PANEL_LABEL_STRING  , string,
                          PANEL_CHOICE_STRINGS, "YES", "NO", 0,
                          PANEL_VALUE         , image_selected[i],
                          PANEL_FEEDBACK      , PANEL_INVERTED,
                          0);
      }

      panel_create_item(selection_panel, PANEL_BUTTON,
```



```
                        PANEL_ITEM_X     , ATTR_COL(3),
                        PANEL_ITEM_Y     , ATTR_ROW(8),
                        PANEL_LABEL_BOLD , TRUE,
                        PANEL_LABEL_IMAGE,
                        panel_button_image(selection_panel, "Proceed", 10, 0),
                        PANEL_NOTIFY_PROC, ROI_select_proc,
                        0);
      panel_create_item(selection_panel, PANEL_BUTTON,
                        PANEL_ITEM_X     , ATTR_COL(19),
                        PANEL_ITEM_Y     , ATTR_ROW(8),
                        PANEL_LABEL_BOLD , TRUE,
                        PANEL_LABEL_IMAGE,
                        panel_button_image(selection_panel, "Cancel", 10, 0),
                        PANEL_NOTIFY_PROC, destroy_selection_frame,
                        0);

      window_fit(selection_frame);
      SUNSPEC1_block_function_selection();
}


/*****************************************************************************/
/* Function writes a string to the frame label of the image selection frame. */
/*****************************************************************************/
static void selection_frame_msg(char_ptr)

char *char_ptr;
{
      window_set(selection_frame, FRAME_LABEL, char_ptr, 0);
}


/*****************************************************************************/
/* Function stores the current location of the frame.                        */
/*****************************************************************************/
static void save_selection_frame_location()
{
      int  x_pos, y_pos;

      /*--------------------+
      ! Save frame location !
      +--------------------*/
      x_pos = (int) window_get(selection_frame, WIN_X, 0);
      y_pos = (int) window_get(selection_frame, WIN_Y, 0);
      if(x_pos != selection_frame_x_pos  ||  y_pos != selection_frame_y_pos) {
          selection_frame_x_pos = x_pos;
          selection_frame_y_pos = y_pos;
          E_begin
            READGEN_store_frame_pos(pos_selection_frame_file, x_pos, y_pos);
          E_handle
          E_others
          Exc_end
      }
}


/*****************************************************************************/
/* Function destroys the selection window.                                   */
/*****************************************************************************/
static void destroy_selection_frame()
{
      save_selection_frame_location();
      window_set( selection_frame , WIN_SHOW, FALSE, 0 );
      window_destroy(selection_frame);
      SUNSPEC1_free_function_selection();
      PROC_SEQ_window_start_next_comm( FUNC_NAME_ROI_SELECT );
}


/*****************************************************************************/
/* Global function to select and display rowdata from an ROI in an image.    */
/*****************************************************************************/
static void ROI_select_proc()
```

```
{
      int          i;
      IMAGE_param  *accompany_image = (IMAGE_param *) 0;
      char         string[DEF_STR_LEN];

  E_begin
      /*-------------------------------------------------------------------+
      ! Get the number of the selected image and read it's parameters. !
      +-------------------------------------------------------------------*/
      selection_image = (int)panel_get_value(selection_item) + 1;
      image           = (IMAGE_param *)IMAGE_param_ptr(selection_image);

      /*-----------------------------------------------+
      ! Error if the selected image contains no data. !
      +-----------------------------------------------*/
      if (!image->raw.filled) EXC(image_error);
      /*-----------------------------------------------+
      ! Update the number of voxels in the ROI in case !
      ! of any zero filling in the spatial dimensions. !
      +-----------------------------------------------*/
      ROI_update();
      /*-------------------------+
      ! Read the ROI parameters. !
      +-------------------------*/
      ROI_read_disp(&ROI);

      /*-------------------------------------+
      ! Error if the ROI does not belong to a !
      ! spectroscopic imaging measurement.    !
      +-------------------------------------*/
      if (ROI.measurement_type != MEAS_1d_si  &&
          ROI.measurement_type != MEAS_2d_si  &&
          ROI.measurement_type != MEAS_3d_si) EXC(invalid_ROI_msg);

      /*-------------------------------------------------------------------+
      ! Check in which images an accompanying cursor has to be displayed. !
      +-------------------------------------------------------------------*/
      for (i = 0; i < MAX_NUM_IMAGES; i++) {
          image_selected[i] = (int)panel_get_value(image_item[i]);

          if (image_selected[i] == 0 || i == (selection_image -1 )  ) {
              /*-----------------------------------------------------+
              ! Read the parameters of the selected accompany image. !
              +-----------------------------------------------------*/
              accompany_image = (IMAGE_param *)IMAGE_param_ptr(i+1);
              /*-----------------------------------------------------+
              ! If the image contains no data, display an error message. !
              +-----------------------------------------------------*/
              if (!accompany_image->raw.filled) {
                  sprintf(string, "Error: no data in image %1d", i+1);
                  EXC(string);
              }
              /*-----------------------------------------------------+
              ! If it's slice orientation matches the slice orientation !
              ! of the selection image, initialize the parameters for   !
              ! displaying an accompany cursor.                         !
              +-----------------------------------------------------*/
              if (accompany_image->raw.slice_orientation ==
                  image->raw.slice_orientation) {
                  cursor_in_image[i]  = TRUE;
                  cursor_displayed[i] = FALSE;
              }
              /*---------------------------------+
              ! If not display an error message. !
              +---------------------------------*/
              else {
                  sprintf(string, "Slice orientation image %1d invalid", i+1);
                  EXC(string);
              }
          }
          /*---------------------------------+
          ! Here if the image is not selected !
          ! to write an accompany cursor to.  !
```



```
          +---------------------------------*/
          else {
              cursor_in_image[i]  = FALSE;
              cursor_displayed[i] = FALSE;
          }
      }

      /*---------------------------------+
      ! Read the stacked curve parameters. !
      +---------------------------------*/
      DATAVIEW_read_par(&data_pars);
      DATAVIEW_par_debug(&data_pars);

      /*---------------------------------+
      ! Read other viewing parameters and !
      ! initialize some other parameters. !
      +---------------------------------*/
      VIEW_read_curve_layout(&curve_layout);
      VIEW_read_curve_window(&curve_window);
      VIEW_read_axis_layout(&axis_layout);
      row_range  = (Row_range *)VIEW_row_range_ptr();
      curve_pw   = (Pixwin *)get_base_pixwin_ptr();
      data_ptr   = (float *)VIEW_data_ptr();
      erase_row  = FALSE;
      row_number = 0; Global_row_number = 0;
      prev_row   = 0;

      save_selection_frame_location();
      window_set( selection_frame , WIN_SHOW, FALSE, 0 );
      window_destroy(selection_frame);
  E_handle
  E_others
      selection_frame_msg(EXC_code);
      return;
  Exc_end


  E_begin
      create_panel();
      calc_vertical_window();
      clear_pixwin_region(curve_pw, 0, 0, 1151, 899);
      init_hidden_line();
      init_selected_row_array();
      VIEW_draw_axis();
      init_image_events();
      start_row_selection();
  E_handle
  E_others
      frame_msg(EXC_code);
  Exc_end
}


/*****************************************************************************/
/* Initialization of the control panel.                                      */
/*****************************************************************************/
static void create_panel()
{
      int   x_pos, y_pos;
      char  string[DEF_STR_LEN];

      /*-------------------------+
      ! Get saved frame position !
      +-------------------------*/
      if (first == TRUE ) {
          first = FALSE;
          E_begin
            READGEN_get_frame_pos(pos_frame_file, &x_pos, &y_pos);
            frame_x_pos = x_pos;
            frame_y_pos = y_pos;
          E_handle
          E_others
          Exc_end
```

```
    }

    frame = window_create(NULL, FRAME,
                          FRAME_EMBOLDEN_LABEL, TRUE,
                          FRAME_LABEL         , "ROI selection",
                          FRAME_NO_CONFIRM    , TRUE,
                          WIN_X               , frame_x_pos,
                          WIN_Y               , frame_y_pos,
                          WIN_MOUSE_XY        , frame_x_pos + 100,
                                                frame_y_pos + 50,
                          WIN_SHOW            , TRUE,
                          0);
    READGEN_window_change_frame_menu(frame);

    panel = window_create(frame, PANEL,
                          WIN_WIDTH , ATTR_COL(60),
                          WIN_HEIGHT, ATTR_ROW(10),
                          0);

    panel_create_item(panel, PANEL_MESSAGE,
                      PANEL_ITEM_X      , ATTR_COL(2),
                      PANEL_ITEM_Y      , ATTR_ROW(0),
                      PANEL_LABEL_BOLD  , TRUE,
                      PANEL_LABEL_STRING, "Cursor position:",
                      0);
    pos_x_text   = panel_create_item(panel, PANEL_TEXT,
                     PANEL_ITEM_X             , ATTR_COL(20),
                     PANEL_ITEM_Y             , ATTR_ROW(0),
                     PANEL_LABEL_BOLD         , TRUE,
                     PANEL_LABEL_STRING       , "x =",
                     PANEL_VALUE_DISPLAY_LENGTH, 8,
                     0);
    pos_y_text   = panel_create_item(panel, PANEL_TEXT,
                     PANEL_ITEM_X             , ATTR_COL(32),
                     PANEL_ITEM_Y             , ATTR_ROW(0),
                     PANEL_LABEL_BOLD         , TRUE,
                     PANEL_LABEL_STRING       , "y =",
                     PANEL_VALUE_DISPLAY_LENGTH, 8,
                     0);

    panel_create_item(panel, PANEL_MESSAGE,
                      PANEL_ITEM_X      , ATTR_COL(2),
                      PANEL_ITEM_Y      , ATTR_ROW(1),
                      PANEL_LABEL_BOLD  , TRUE,
                      PANEL_LABEL_STRING, "Spectrum number:",
                      0);
    number_text  = panel_create_item(panel, PANEL_TEXT,
                     PANEL_ITEM_X             , ATTR_COL(20),
                     PANEL_ITEM_Y             , ATTR_ROW(1),
                     PANEL_LABEL_BOLD         , TRUE,
                     PANEL_LABEL_STRING       , "",
                     PANEL_VALUE_DISPLAY_LENGTH, 8,
                     0);

  panel_create_item(panel, PANEL_MESSAGE,
                      PANEL_ITEM_X      , ATTR_COL(8),
                      PANEL_ITEM_Y      , ATTR_ROW(3),
                      PANEL_LABEL_BOLD  , TRUE,
                      PANEL_LABEL_STRING,
                      "Use mouse to select a spectrum from the ROI",
                      0);
    panel_create_item(panel, PANEL_MESSAGE,
                      PANEL_ITEM_X      , ATTR_COL(10),
                      PANEL_ITEM_Y      , ATTR_ROW(4),
                      PANEL_LABEL_BOLD  , TRUE,
                      PANEL_LABEL_STRING,
                      "Press left mouse button to fix spectrum",
                      0);

    num_rows_text = panel_create_item(panel, PANEL_TEXT,
                      PANEL_ITEM_X             , ATTR_COL(19),
                      PANEL_ITEM_Y             , ATTR_ROW(6),
                      PANEL_LABEL_BOLD         , TRUE,
```



```
                      PANEL_LABEL_STRING       , "Number of rows: ",
                      PANEL_VALUE_DISPLAY_LENGTH, 6,
                      0);
      sprintf(string, "%6d", curve_num_rows);
      panel_set_value(num_rows_text, string);

      panel_create_item(panel, PANEL_BUTTON,
                        PANEL_ITEM_X     , ATTR_COL(0),
                        PANEL_ITEM_Y     , ATTR_ROW(8),
                        PANEL_LABEL_BOLD , TRUE,
                        PANEL_LABEL_IMAGE,
                        panel_button_image(panel, "Adapt num rows", 16, 0),
                        PANEL_NOTIFY_PROC, adapt_selected_row_array,
                        0);
      panel_create_item(panel, PANEL_BUTTON,
                        PANEL_ITEM_X     , ATTR_COL(21),
                        PANEL_ITEM_Y     , ATTR_ROW(8),
                        PANEL_LABEL_BOLD , TRUE,
                        PANEL_LABEL_IMAGE,
                        panel_button_image(panel, "Refresh rows", 16, 0),
                        PANEL_NOTIFY_PROC, refresh_proc,
                        0);
      panel_create_item(panel, PANEL_BUTTON,
                        PANEL_ITEM_X     , ATTR_COL(42),
                        PANEL_ITEM_Y     , ATTR_ROW(8),
                        PANEL_LABEL_BOLD , TRUE,
                        PANEL_LABEL_IMAGE,
                        panel_button_image(panel, "Remove all rows", 16, 0),
                        PANEL_NOTIFY_PROC, remove_proc,
                        0);

      panel_create_item(panel, PANEL_BUTTON,
                        PANEL_ITEM_X     , ATTR_COL(10),
                        PANEL_ITEM_Y     , ATTR_ROW(9),
                        PANEL_LABEL_BOLD , TRUE,
                        PANEL_LABEL_IMAGE,
                        panel_button_image(panel, "Proceed", 10, 0),
                        PANEL_NOTIFY_PROC, proceed_proc,
                        0);

      panel_create_item(panel, PANEL_BUTTON,
                        PANEL_ITEM_X     , ATTR_COL(38),
                        PANEL_ITEM_Y     , ATTR_ROW(9),
                        PANEL_LABEL_BOLD , TRUE,
                        PANEL_LABEL_IMAGE,
                        panel_button_image(panel, "Cancel", 10, 0),
                        PANEL_NOTIFY_PROC, cancel_proc,
                        0);
      window_fit(frame);
      SUNSPEC1_block_function_selection();
}


/*****************************************************************************/
/* Function writes a string to the frame label of the image selection frame. */
/*****************************************************************************/
static void frame_msg(char_ptr)

char *char_ptr;
{
          window_set(frame, FRAME_LABEL, char_ptr, 0);
}


/*****************************************************************************/
/* Function calculates the extrema of the datarows stored in memory, so that */
/* a vertical window can be applied to build up a stacked curve in which all */
/* rows will be displayed with the same vertical scale.                      */
/*****************************************************************************/
static void calc_vertical_window()
{
      int    i, x_left, x_right;
      float  row_max, row_min, scale_max, scale_min;
```

```
        /*-----------------------------------------------+
        ! If there is already a vertical window applied, !
        ! no vertical window needs to be calculated.     !
        +-----------------------------------------------*/
        if (curve_window.VER_SET) return;

        /*-------------------------------------------------------+
        ! Determine the domain on which to examine the extrema. !
        +-------------------------------------------------------*/
        if (curve_window.HOR_SET) {
            x_left  = curve_window.X_LEFT;
            x_right = curve_window.X_RIGHT;
        }
        else {
            x_left  = 0;
            x_right = data_pars.col.pnts - 1;
        }

        scale_min =  MAXFLOAT;
        scale_max = -MAXFLOAT;

        for (i = 0; i < data_pars.row.pnts; i++) {
            get_max_min_float_data(data_ptr + i * data_pars.col.pnts + x_left,
                                   x_right - x_left + 1, &row_max, &row_min);

            if (row_min < scale_min) scale_min = row_min;
            if (row_max > scale_max) scale_max = row_max;
        }

        /*-------------------------------------------------------+
        ! Store the extrema in the vertical window parameters.   !
        ! Note that this is just a local copy of the vertical    !
        ! window, so it will only be effective within this module. !
        +-------------------------------------------------------*/
        curve_window.VER_SET = 1;
        curve_window.Y_MAX   = scale_max;
        curve_window.Y_MIN   = scale_min;
}


/*****************************************************************************/
/* Function initializes the events for the image window.                     */
/*****************************************************************************/
static void init_image_events()
{
        window_set(image->disp.canvas,
                   WIN_CONSUME_PICK_EVENTS, WIN_NO_EVENTS,
                                            LOC_MOVE,
                                            MS_LEFT,
                                            0,
                   WIN_EVENT_PROC         , handle_event,
                   0);
}


/*****************************************************************************/
/* Function disables the events for the image window.                        */
/*****************************************************************************/
static void disable_image_events()
{
        window_set(image->disp.canvas,
                   WIN_CONSUME_PICK_EVENTS, WIN_NO_EVENTS, 0,
                   0);
}


/*****************************************************************************/
/* Function to start the row selection from the image: the cursor is placed  */
/* in the center of the selection image and the row, corresponding with this */
/* position is displayed.                                                    */
/*****************************************************************************/
static void start_row_selection()
```



```
{
        int    factor, x_pos, y_pos;
        float  x_coord, y_coord;

        factor = image->disp.zoom_factor * image->disp.format_factor;
        /*-----------------------------------------------------+
        ! Calculate the coordinates of the center of the image. !
        +-----------------------------------------------------*/
        x_pos = factor * image->disp.matrix_x / 2;
        y_pos = factor * image->disp.matrix_y / 2;
        /*-----------------------------------------------------+
        ! Put the cursor in the middle of the selection image. !
        +-----------------------------------------------------*/
        window_set(image->disp.canvas, WIN_MOUSE_XY, x_pos, y_pos, 0);
        /*----------------------------+
        ! Draw the corresponding row. !
        +----------------------------*/
        calc_row_number(x_pos, y_pos);
        /*--------------------------------+
        ! Write the accompanying cursors. !
        +--------------------------------*/
        x_coord = image->disp.offset_x - image->disp.FOV_x / 2. +
                  (float)x_pos / (factor * image->disp.matrix_x - 1) *
                  image->disp.FOV_x;
        y_coord = image->disp.offset_y + image->disp.FOV_y / 2. -
                  (float)y_pos / (factor * image->disp.matrix_y - 1) *
                  image->disp.FOV_y;
        adapt_accompany_cursor(x_coord, y_coord);
}


/*****************************************************************************/
/* Allocate a sufficient amount of memory to store the selected rows in.     */
/*****************************************************************************/
static void init_selected_row_array()
{
        int  num_bytes;

        /*---------------------------------------+
        ! Allocate a sufficient amount of memory !
        ! to store the selected rows in.         !
        +---------------------------------------*/
        num_bytes     = curve_num_rows * sizeof(int);
        selected_rows = (int *)malloc(num_bytes);
}


/*****************************************************************************/
/* Function adapts the allocated memory for increasion or decreasion of the  */
/* the number of selectable rows. The selected rows are refreshed.           */
/*****************************************************************************/
static void adapt_selected_row_array()
{
        int   num_bytes, new_num_rows;
        char  string[DEF_STR_LEN];

        /*-----------------------------------------+
        ! Get the new number of rows from the panel. !
        +-----------------------------------------*/
        strcpy(string, panel_get_value(num_rows_text));
        if (sscanf(string, "%6d", &new_num_rows) != 1) {
            frame_msg(invalid_num_rows);
            sprintf(string, "%6d", curve_num_rows);
            panel_set_value(num_rows_text, string);
            return;
        }
        /*-----------------------------------------+
        ! If not valid, display an error message.   !
        +-----------------------------------------*/
        if (new_num_rows < 1) {
            frame_msg(invalid_num_rows);
            sprintf(string, "%6d", curve_num_rows);
            panel_set_value(num_rows_text, string);
```

```
        return;
    }
    else {
        curve_num_rows = new_num_rows;
    }
    /*---------------------------------------+
    ! Adapt the allocated amount of memory to !
    ! the new number of rows.                 !
    +---------------------------------------*/
    num_bytes     = curve_num_rows * sizeof(int);
    selected_rows = (int *)realloc(selected_rows, num_bytes);
    /*-------------------------------------------------------------+
    ! If the current number of rows exceeds the new number of rows,  !
    ! decrease the current number of rows to the new number of rows. !
    +-------------------------------------------------------------*/
    if (row_number > curve_num_rows) {
      row_number = curve_num_rows;
      Global_row_number = row_number;
    }
    /*-------------------------------------------------------------+
    ! Refresh the displayed rows (a new layout will be created). !
    +-------------------------------------------------------------*/
    refresh_proc();
}

/*****************************************************************************/
/* Function frees the allocated memory for the selected rows.                */
/*****************************************************************************/
static void delete_selected_row_array()
{
    free((char *)selected_rows);
}


/*****************************************************************************/
/* Function to handle the events in the image window.                        */
/*****************************************************************************/
static void handle_event(canvas_local, event)

Canvas  canvas_local;
Event   *event;
{
    switch (event_id(event)) {
        case LOC_MOVE: calc_row_number(event_x(event), event_y(event));
                       break;
        case MS_LEFT : erase_row = FALSE;
                       if (row_number < curve_num_rows) {
                           selected_rows[row_number] = prev_row;
                           row_number++;
                           Global_row_number = row_number;
                       }
                       break;
        default      : break;
    }
}


/*****************************************************************************/
/* Function determines the rowselection from the position of the cursor.     */
/*****************************************************************************/
static void calc_row_number(x_pos, y_pos)

int x_pos, y_pos;
{
    float  x_pos_cursor, y_pos_cursor, delta_x, delta_y;
    float  angulated_x, angulated_y, alpha;
    int    factor, x_row, y_row, sel_row;
    char   string[DEF_STR_LEN];

    DEB("\nCalc_row_number  x_pos = %d  y_pos = %d spec.orient=%d (ROI_select)",
          x_pos, y_pos, image->disp.spec_image_orient);

    factor = image->disp.zoom_factor * image->disp.format_factor;
```



```
    /*-------------------------------------------------------------+
    ! Determine the physical coordinates of the cursor in the image, !
    ! and write them to the control panel.                           !
    +-------------------------------------------------------------*/
    x_pos_cursor = image->disp.offset_x - image->disp.FOV_x / 2. +
                   (float)x_pos /
                   (factor * image->disp.matrix_x - 1) * image->disp.FOV_x;
    y_pos_cursor = image->disp.offset_y + image->disp.FOV_y / 2. -
                   (float)y_pos /
                   (factor * image->disp.matrix_y - 1) * image->disp.FOV_y;

    sprintf(string, "%7.2f", x_pos_cursor);
    panel_set_value(pos_x_text, string);

    sprintf(string, "%7.2f", y_pos_cursor);
    panel_set_value(pos_y_text, string);

    /*-------------------------------------------------+
    ! Adapt the positions of the accompanying cursors. !
    +-------------------------------------------------*/
    adapt_accompany_cursor(x_pos_cursor, y_pos_cursor);

    /*-------------------------------------------------+
    ! Check if image is spec.image from which orientation  !
    ! is adapted according to "Spec.image!" parm in *.spar.!
    +-------------------------------------------------*/
    if ( image->disp.spec_image_orient != PLUSA_PLUSB )
    {
      reorient_si(&x_pos, &y_pos);
      /*---------------------------------------+
      ! Now recalculate the physical coordinates in !
      ! order to select the corresponding spectrum. !
      +---------------------------------------*/
      x_pos_cursor = image->disp.offset_x - image->disp.FOV_x / 2. +
                     (float)x_pos /
                     (factor * image->disp.matrix_x - 1) * image->disp.FOV_x;
      y_pos_cursor = image->disp.offset_y + image->disp.FOV_y / 2. -
                     (float)y_pos /
                     (factor * image->disp.matrix_y - 1) * image->disp.FOV_y;

      DEB("\nReoriented  x_pos = %d  y_pos = %d  (ROI_select)",
            x_pos, y_pos);
    }

    /*-----------------------------------+
    ! Calculate the offset of the cursor  !
    ! according to the center of the ROI. !
    +-----------------------------------*/
    delta_x = x_pos_cursor - ROI.ROI_1.offset_x;
    delta_y = y_pos_cursor - ROI.ROI_1.offset_y;
    /*-----------------------------------------------------------------------+
    ! To check if any row is selected by the cursor, the position of the     !
    ! cursor has to be compared with that of the ROI.                        !
    ! In case of an angulated ROI, this would demand intricate calculations. !
    ! Instead, the position of the cursor can be angulated over the negative !
    ! angulation of the ROI, and be compared with the position of the        !
    ! unangulated ROI.                                                       !
    +-----------------------------------------------------------------------*/
    alpha       = -(ROI.ROI_1.angulation / 360. * 2 * M_PI);
    angulated_x = ROI.ROI_1.offset_x +
                  delta_x * cos(alpha) - delta_y * sin(alpha);
    angulated_y = ROI.ROI_1.offset_y +
                  delta_x * sin(alpha) + delta_y * cos(alpha);
    /*-------------------------------------------------------------+
    ! If the angulated cursorposition lies within the unangulated ROI, !
    ! and the maximum number of rows in the stacked curve is not yet   !
    ! reached, determine the selected row and display it.              !
    +-------------------------------------------------------------*/
    if (angulated_x >= ROI.ROI_1.offset_x - ROI.ROI_1.size_x / 2.  &&
        angulated_x <= ROI.ROI_1.offset_x + ROI.ROI_1.size_x / 2.  &&
        angulated_y >= ROI.ROI_1.offset_y - ROI.ROI_1.size_y / 2.  &&
        angulated_y <= ROI.ROI_1.offset_y + ROI.ROI_1.size_y / 2.  &&
```

```
        row_number   <  curve_num_rows) {

        x_row =
        (int)(((angulated_x - (ROI.ROI_1.offset_x - ROI.ROI_1.size_x / 2.))
              / ROI.ROI_1.size_x) * ROI.ROI_1.num_voxels_x + 1);
        y_row =
        (int)((-(angulated_y - (ROI.ROI_1.offset_y + ROI.ROI_1.size_y / 2.))
              / ROI.ROI_1.size_y) * ROI.ROI_1.num_voxels_y + 1);
        sel_row = (y_row - 1) * ROI.ROI_1.num_voxels_x + x_row;

        if ( sel_row < 1 || sel_row > data_pars.row.pnts) {
            frame_msg(exceeded_row_msg);
            return;
        }
        else {
            frame_msg(empty_msg);
        }

        sprintf(string, "%d", sel_row);
        panel_set_value(number_text, string);

        if (sel_row != prev_row  ||  !erase_row) {
            if (erase_row) {
                restore_prev_hidden_line();
                write_curve(prev_row, row_number, FOREGROUND);
                restore_prev_hidden_line();
            }

            save_hidden_line();
            write_curve(sel_row, row_number, BACKGROUND);

            prev_row = sel_row;
        }

        erase_row = TRUE;
    }
}


/*****************************************************************************/
/* Function fixes the reoriention, evt. done in case of spectroscopic images. */
/*****************************************************************************/
static void reorient_si( x_pos_ptr, y_pos_ptr )

int *x_pos_ptr, *y_pos_ptr;
{
  /*------------+
  ! Local parms !
  +------------*/
  int nr_x_points = image->disp.matrix_x;
  int nr_y_points = image->disp.matrix_y;
  int spec_orient = ROI.ROI_1.spec_image_orient;
  int xposi       = *(x_pos_ptr);
  int yposi       = *(y_pos_ptr);
  int remember;

  DEB("\nReorient SI  x_pos = %d  y_pos = %d spec_orient=%d, matrix=(%d,%d)",
        *x_pos_ptr, *y_pos_ptr, spec_orient, nr_x_points, nr_y_points);

  if ( spec_orient == PLUSA_MINB || spec_orient == MINA_MINB ||
       spec_orient == PLUSB_MINA || spec_orient == MINB_MINA )
  {
    /*-------------------------------------+
    ! Rows have been reordered:            !
    ! nr.last -> nr.1 ... nr.1 -> nr.last: !
    +-------------------------------------*/
    yposi = (nr_y_points - yposi);
  }

  if ( spec_orient == MINA_PLUSB || spec_orient == MINA_MINB ||
       spec_orient == MINB_PLUSA || spec_orient == MINB_MINA )
  {
    /*-------------------------------------+
```



```
    ! Columns have been reordered:         !
    ! nr.last -> nr.1 ... nr.1 -> nr.last: !
    +-------------------------------------*/
    xposi = (nr_x_points - xposi);
  }

  if ( spec_orient == PLUSB_PLUSA || spec_orient == PLUSB_MINA ||
       spec_orient == MINB_PLUSA || spec_orient == MINB_MINA  )
  {
    /*-------------------------------------+
    ! X- and y-direction have been exchanged !
    +-------------------------------------*/
    remember = xposi;
    xposi = yposi;
    yposi = remember;
  }

  *(x_pos_ptr) = xposi;
  *(y_pos_ptr) = yposi;

  DEB("\nSI reoriented  x_pos = %d  y_pos = %d  (ROI_select)",
        *x_pos_ptr, *y_pos_ptr);

}
/*****************************************************************************/
/* Function writes a selected curve to the display.                          */
/*****************************************************************************/
static void write_curve(row_nr, rank_nr, color)

int  row_nr, rank_nr, color;
{
      float  x_step, y_step;

      DEB("\nWrite_curve  row_nr=%d  rank_nr=%d   color=%d   (ROI_select)",
                row_nr , rank_nr, color );

      /*-----------------------------------------------------------+
      ! Calculate the layout step size between 2 successive rows. !
      +-----------------------------------------------------------*/
      x_step = (float)(axis_layout.DIM3_X2 - axis_layout.DIM3_X1) /
               (curve_num_rows - 1);
      y_step = (float)(axis_layout.DIM3_Y2 - axis_layout.DIM3_Y1) /
               (curve_num_rows - 1);
      /*----------------------------+
      ! Display the selected curve. !
      +----------------------------*/
      display_data(curve_pw, data_ptr + (row_nr - 1) * data_pars.col.pnts,
                   data_pars.col.pnts,
                   curve_layout.X_LEFT   + (int)rint(rank_nr * x_step),
                   curve_layout.X_RIGHT  + (int)rint(rank_nr * x_step),
                   curve_layout.Y_BOTTOM + (int)rint(rank_nr * y_step),
                   curve_layout.Y_TOP    + (int)rint(rank_nr * y_step),
                   curve_window.HOR_SET,
                   curve_window.X_LEFT, curve_window.X_RIGHT,
                   curve_window.VER_SET,
                   curve_window.Y_MIN, curve_window.Y_MAX,
                   curve_layout.HIDDEN_LINES,
                   curve_layout.HIDDEN_LINE_PTR,
                   color);


      DEB("\nWrite_curve    exit    (ROI_select)");
}


/*****************************************************************************/
/* Function adapts the position of the accompany cursors.                    */
/*****************************************************************************/
static void adapt_accompany_cursor(x_pos, y_pos)

float x_pos, y_pos;
{
      IMAGE_param  *acc_image;
```

```
        float       offset_x, offset_y, FOV_x, FOV_y;
        int         matrix_x, matrix_y, factor;
        int         i;

        for (i = 0; i < MAX_NUM_IMAGES; i++) {
            if (cursor_in_image[i]) {
                /*-----------------------------------------+
                ! Read the image parameters of the image   !
                ! in which the cursor has to be displayed. !
                +-----------------------------------------*/
                acc_image = (IMAGE_param *)IMAGE_param_ptr(i + 1);

                cursor_pw[i] = acc_image->disp.pw;
                /*-------------------------------------------------------+
                ! If the cursor is currently being displayed, erase it. !
                +-------------------------------------------------------*/
                if (cursor_displayed[i])
                    draw_accompany_cursor(cursor_pw[i],
                                          cursor_x[i], cursor_y[i]);

                offset_x = acc_image->disp.offset_x;
                offset_y = acc_image->disp.offset_y;
                FOV_x    = acc_image->disp.FOV_x;
                FOV_y    = acc_image->disp.FOV_y;
                matrix_x = acc_image->disp.matrix_x;
                matrix_y = acc_image->disp.matrix_y;
                factor   = acc_image->disp.zoom_factor *
                           acc_image->disp.format_factor;

                /*-----------------------------------------+
                ! Calculate the pixwin coordinates according !
                ! to the physical coordinates of the cursor. !
                +-----------------------------------------*/
                cursor_x[i] = (int)rint(((x_pos - (offset_x - FOV_x / 2.)) /
                                         FOV_x) * (factor * matrix_x - 1));
                cursor_y[i] = (int)rint((factor * matrix_y - 1) -
                                        ((y_pos - (offset_y - FOV_y / 2.)) /
                                         FOV_y) * (factor * matrix_y - 1));
                /*-------------------------------------------------------+
                ! Only draw the accompanying cursor if it lies within !
                ! the displayed region of the image it belongs to.     !
                +-------------------------------------------------------*/
                if (cursor_x[i] >= 0  &&  cursor_x[i] < factor * matrix_x  &&
                    cursor_y[i] >= 0  &&  cursor_y[i] < factor * matrix_y) {
                    draw_accompany_cursor(cursor_pw[i],
                                          cursor_x[i], cursor_y[i]);
                    cursor_displayed[i] = TRUE;
                }
                else {
                    cursor_displayed[i] = FALSE;
                }
            }
        }
}

/*****************************************************************************/
/* Function draws the accompany cursor in the specified pixwin.              */
/*****************************************************************************/
static void draw_accompany_cursor(pixwin, x_pos, y_pos)

Pixwin *pixwin;
int    x_pos, y_pos;
{
        int  i, x1, y1, x2, y2;

        /*-------------------------------------------------+
        ! Draw the horizontal cursor stripe (3 lines thick). !
        +-------------------------------------------------*/
        x1 = x_pos - 5;
        x2 = x_pos + 5;
        pw_vector(pixwin, x1, y_pos-1, x2, y_pos-1, PIX_SRC ^ PIX_DST, 255);
        pw_vector(pixwin, x1, y_pos  , x2, y_pos  , PIX_SRC ^ PIX_DST, 127);
        pw_vector(pixwin, x1, y_pos+1, x2, y_pos+1, PIX_SRC ^ PIX_DST, 255);
```



```
        /*-------------------------------------------------+
        ! Draw the vertical cursor stripe (3 lines thick). !
        +-------------------------------------------------*/
        y1 = y_pos - 5;
        y2 = y_pos + 5;
        pw_vector(pixwin, x_pos-1, y1, x_pos-1, y2, PIX_SRC^PIX_DST, 255);
        pw_vector(pixwin, x_pos  , y1, x_pos  , y2, PIX_SRC^PIX_DST, 127);
        pw_vector(pixwin, x_pos+1, y1, x_pos+1, y2, PIX_SRC^PIX_DST, 255);
}


/*****************************************************************************/
/* Function stores the current location of the frame.                        */
/*****************************************************************************/
static void save_frame_location()
{
        int  x_pos, y_pos;

        /*--------------------+
        ! Save frame location !
        +--------------------*/
        x_pos = (int) window_get(frame, WIN_X, 0);
        y_pos = (int) window_get(frame, WIN_Y, 0);
        if(x_pos != frame_x_pos  ||  y_pos != frame_y_pos) {
            frame_x_pos = x_pos;
            frame_y_pos = y_pos;
            E_begin
              READGEN_store_frame_pos(pos_frame_file, x_pos, y_pos);
            E_handle
            E_others
            Exc_end
        }
}


/*****************************************************************************/
/* Function to update row range array                                        */
/*****************************************************************************/
static void spectrum_number_update( row_arr, num_rows )

int     row_arr[];      /* in : array with row sequence          */
int     num_rows;       /* in : number of rows                   */

{ /*-----------------+
  ! Local parameters !
  +-----------------*/
  float         *x_ptr = FLOAT_NULL_PTR;
  float         *y_ptr = FLOAT_NULL_PTR;
  float         voxel_size;
  float         x;
  float         y;
  int           i;
  int           x_row;
  int           y_row;
  double        alpha;
  double        sin_alpha;
  double        cos_alpha;
  ROI_disp       ROI;

  DEB("\nSpectrum_number_update  rows = %d  (ROI_select)", num_rows );

  specific_rows.sel_row_total = 0;                              /*  jane 940708 */

  if( num_rows > 0 ) {
                                                /*  jane 940708 */
    for ( i=0; i<MAX_SPECIFIC_ROWS; i++)
        specific_rows.sel_row_array[i] = 0;

    specific_rows.sel_row_total = num_rows;
    for ( i=0; i<num_rows; i++)
        specific_rows.sel_row_array[i] =  row_arr[i];
                                                /*  jane 940708 */
```

```
    SPECTRUM_number_reset(  &x_ptr, &y_ptr, num_rows );
    DEB("\n x_ptr = %d   y_ptr = %d", x_ptr, y_ptr );
    /*-------------------------+
    ! Read the ROI parameters. !
    +-------------------------*/
    ROI_update();
    ROI_read_disp(&ROI);
    /*-----------------------------------+
    ! Error if the ROI does not belong to a !
    ! spectroscopic imaging measurement.    !
    +-----------------------------------*/
    if (ROI.measurement_type != MEAS_1d_si  &&
        ROI.measurement_type != MEAS_2d_si  &&
        ROI.measurement_type != MEAS_3d_si) {
        EXC( invalid_ROI_msg );
    }
    alpha = (ROI.ROI_1.angulation / 360. * 2 * M_PI);
    sin_alpha = sin( alpha );
    cos_alpha = cos( alpha );

    for( i=0; i<num_rows; i++ ) {
        y_row = ((row_arr[i]-1)/ROI.ROI_1.num_voxels_x) + 1;
        x_row = row_arr[i] - (y_row-1) * ROI.ROI_1.num_voxels_x;
        DEB("\n%d :  num=%d   x_row = %d   y_row = %d",
                i, row_arr[i], x_row, y_row );
        voxel_size = ROI.ROI_1.size_x/ROI.ROI_1.num_voxels_x;
        x = ((float)x_row -0.5) * voxel_size - ROI.ROI_1.size_x/2.0;
        voxel_size = ROI.ROI_1.size_y/ROI.ROI_1.num_voxels_y;
        y = -((float)y_row-0.5) * voxel_size + ROI.ROI_1.size_y/2.0;
        DEB("\n x_pos = %f   y_pos = %f", x_ptr[i], y_ptr[i] );
        /*-----------------+
        ! Apply angulation !
        +-----------------*/
        x_ptr[i] =  x * cos_alpha - y * sin_alpha + ROI.ROI_1.offset_x;
        y_ptr[i] =  x * sin_alpha + y * cos_alpha + ROI.ROI_1.offset_y;
        DEB("\n x_pos = %f   y_pos = %f", x_ptr[i], y_ptr[i] );
    }
  }
  /*-------------+
  ! Routine exit !
  +-------------*/
  DEB("\nSpectrum_number_update  exit  (ROI_select)" );
}
/*****************************************************************************/
/* Function to update row range array                                        */
/*****************************************************************************/
static void update_row_range_array( row_arr, num_rows )

int     row_arr[];      /* in : array with row sequence         */
int     num_rows;       /* in : number of rows                  */

{ /*-----------------+
  ! Local parameters !
  +-----------------*/
  int   i;
  int   num_bytes;
  int   start;
  int   stop;
  char  str[DEF_STR_LEN];
  char  val_str[DEF_STR_LEN];

  DEB("\nUpdate_row_array  rows = %d  (ROI_select)", num_rows );

  if( num_rows > 0 ) {
      /*------------------+
      ! Check row numbers !
      +------------------*/
      for( i=0; i<num_rows; i++ ) {
        if( row_arr[i]<1 || row_arr[i]>data_pars.row.pnts ) EXC( err_row );
      }
      /*----------------+
      ! Store row range !
      +----------------*/
```



```
      row_range  = (Row_range *)VIEW_row_range_ptr();
      row_range->num_rows = num_rows;
      free((char *)row_range->array);
      num_bytes          = row_range->num_rows * sizeof(int);
      row_range->array = (int *)malloc(num_bytes);
      if( row_range->array == INT_NULL_PTR ) EXC( err_allo );
      for (i = 0; i < row_range->num_rows; i++)
        row_range->array[i] = row_arr[i];
      /*---------------+
      ! History loging !
      +---------------*/
      if( num_rows == 1 ) {
        HIS("# %s   num=1  rows=%d", FUNC_NAME_ROI_SELECT, row_arr[0] );
      }else{
        /*-----------------+
        ! Log first 8 rows !
        +-----------------*/
        if( num_rows <=8 ) stop = num_rows;
        else               stop = 8;
        sprintf( str, "%d", row_arr[0] );
        for (i = 1; i <stop; i++) {
            sprintf( val_str, ",%d", row_arr[i] );
            strcat( str, val_str );
        }
        HIS("# %s   num=%d  rows=%s", FUNC_NAME_ROI_SELECT, num_rows , str );
        /*------------------------+
        ! Log mutiples of 16 rows !
        +------------------------*/
        start = -7;
        do{
          start = start + 16;
          stop  = start + 15;
          if( stop > num_rows ) stop = num_rows;
          if( stop >= start ) {
            sprintf( str, "    %d", row_arr[start-1] );
            for (i = start; i <stop; i++) {
              sprintf( val_str, ",%d", row_arr[i] );
              strcat( str, val_str );
            }
            HIS("  %s ", str );
          }
        }while( stop > start );
      }
      /*-----------------------------+
      ! Update spectrum number arrays !
      +-----------------------------*/
      spectrum_number_update( row_arr, num_rows );
  }
  /*-------------+
  ! Routine exit !
  +-------------*/
  DEB("\nUpdate_row_array  exit  (ROI_select)" );
}

/*****************************************************************************/
/* Function proceeds with the stacked curve of the selected rows.            */
/*****************************************************************************/
static void proceed_proc()
{
   int  i, num_bytes;

   E_begin
      /*--------------------------------+
      ! Erase the accompanying cursors. !
      +--------------------------------*/
      for (i = 0; i < MAX_NUM_IMAGES; i++) {
          if (cursor_in_image[i]  &&  cursor_displayed[i])
              draw_accompany_cursor(cursor_pw[i], cursor_x[i], cursor_y[i]);
      }

      /*---------------------------------------+
      ! Adapt the row range array in such way   !
      ! that it contains the selected datarows. !
```

```
      +--------------------------------------*/
      update_row_range_array( selected_rows, row_number );

Global_row_number = row_number;

      VIEW_curve_init( 0, 0 );
  E_handle
  E_others
       selection_frame_msg(EXC_code);
       return;
  Exc_end

  num_bytes     = Global_row_number * sizeof(int);

  if (Global_selected_rows != NULL)
    free(Global_selected_rows);

  Global_selected_rows = (int *)malloc(num_bytes);

  for(i=0; i<Global_row_number; ++i)
    Global_selected_rows[i] = selected_rows[i];

  delete_selected_row_array();
  disable_image_events();
  save_frame_location();
  window_set( frame , WIN_SHOW, FALSE, 0 );
  window_destroy(frame);
  SUNSPEC1_free_function_selection();
  PROC_SEQ_window_start_next_comm( FUNC_NAME_ROI_SELECT );
}


/*****************************************************************************/
/* Function refreshes the displayed curves by redisplaying them.            */
/*****************************************************************************/
static void refresh_proc()
{
      int  i;

      /*----------------------------------+
      ! Hidden line removal is restarted. !
      +----------------------------------*/
      init_hidden_line();
      /*------------------+
      ! Clear the pixwin. !
      +------------------*/
      clear_pixwin_region(curve_pw, 0, 0, MAX_X_PIXEL-1, MAX_Y_PIXEL-1);
      /*------------------+
      ! Refresh the axis. !
      +------------------*/
      VIEW_draw_axis();
      /*------------------+
      ! Refresh the rows. !
      +------------------*/
      for (i = 0; i < row_number; i++)
          write_curve(selected_rows[i], i, BACKGROUND);
      /*-----------------------------+
      ! Save the current hidden line. !
      +-----------------------------*/
      save_hidden_line();
}


/*****************************************************************************/
/* Function removes all selected rows.                                       */
/*****************************************************************************/
static void remove_proc()
{
      /*----------------------+
      ! Reset the row numbers. !
      +----------------------*/
      row_number = 0; Global_row_number = row_number;
      prev_row   = 0;
```



```
      /*--------------------+
      ! Refresh the screen. !
      +--------------------*/
      refresh_proc();
      /*---------------------------+
      ! Restart the row selection. !
      +---------------------------*/
      start_row_selection();
}


/*****************************************************************************/
/* Function cancels the selected curves and continues with the old stacked   */
/* curve.                                                                    */
/*****************************************************************************/
static void cancel_proc()
{
      int  i;

      /*---------------------------------+
      ! Erase the accompanying cursors. !
      +---------------------------------*/
      for (i = 0; i < MAX_NUM_IMAGES; i++) {
          if (cursor_in_image[i]  &&  cursor_displayed[i])
              draw_accompany_cursor(cursor_pw[i], cursor_x[i], cursor_y[i]);
      }

      E_begin
        VIEW_curve_init( 0, 0 );
      E_handle
      E_others
      Exc_end

      delete_selected_row_array();
      disable_image_events();
      save_frame_location();
      window_set( frame , WIN_SHOW, FALSE, 0 );
      window_destroy(frame);
      SUNSPEC1_free_function_selection();
      PROC_SEQ_window_start_next_comm( FUNC_NAME_ROI_SELECT );
}
```

```
/*MPF ::: FUNCTIONS ::  ROI_SELECTION =========

    FUNCTION NAME
        ROI_select_proc_seq

    DESCRIPTION
        Selection of rows.
        .

    ANYOTHER
        Author          : Ad Marien
        Creation Date   : 1989-03-02

    CALLING SEQUENCE
*/
int             /* return : nothing                                  */

ROI_select_proc_seq( fc,  mode , interactive )

FILE    *fc;            /* in : processing sequence file pointer        */
char    mode;           /* in : 'b' = batch mode, 'n' = normal mode     */
char    interactive;    /* in : 'i' = interactive else ' '              */

{ /*-----------------+
  ! Local parameters !
  +-----------------*/
  char          str[DEF_STR_LEN];
  char          *ptr;
  int           i = 0;
  int           num_bytes;
  int           max_rows = 0;
  int           *row_ptr = INT_NULL_PTR;
  bool          stop = FALSE;

  if( DEB_is_on() == TRUE ) {
    DEB("\nROI_select_proc_seq");
    DEB("\n fc = %d    mode = %c  interactive = (%c)",
                fc , mode , interactive );
  }

  if( mode == 'n' ) {
    if( interactive == 'i' ) {
        /*---------------------+
        ! Interactive function !
        +---------------------*/
        ROI_select();
    }else{
      /*-------------------------+
      ! Non interactive function !
      +-------------------------*/
      DATAVIEW_read_par(&data_pars);
      DATAVIEW_par_debug(&data_pars);
      PARINPUT_window_next_str( fc , str );
      ptr = str;
      /*-------------------------------+
      ! Get number of rows parameter   !
      +-------------------------------*/
      STR_FUNC_skip_char( '=' , 'y' , &ptr );
      if( sscanf(ptr ,"%d", &max_rows ) != 1 ) EXC( err_num );
      if( max_rows < 1 ) EXC( err_num );
      num_bytes = max_rows * sizeof(int);
      row_ptr = INT_PTR malloc(num_bytes);
      if( row_ptr == INT_NULL_PTR ) EXC( err_allo );

      E_begin
        /*-----------------------------------------+
        ! Get row range parameters                  !
        ! Check on : range = <value> - <value> - ... !
        +-----------------------------------------*/
        STR_FUNC_skip_char( '=' , 'y' , &ptr );
        i = 0;
        if( sscanf(ptr ,"%d", &row_ptr[i] ) != 1 ) EXC( err_row );
```



```
        if( row_ptr[i]<1 || row_ptr[i]>data_pars.row.pnts ) EXC( err_row );
        i++;
        do{
          E_begin       STR_FUNC_skip_char( ',' , 'y' , &ptr );
          E_handle
          E_others      stop = TRUE;
          Exc_end
          if( stop == FALSE ) {
            /*---------------------------+
            ! Check on line continuation !
            +---------------------------*/
            if( *ptr == '\0' ) {
                PARINPUT_window_next_str( fc , str );
                ptr = str;
            }
            /*----------+
            ! Get value !
            +----------*/
            if( sscanf(ptr ,"%d", &row_ptr[i] ) != 1 ) EXC( err_row );
            if( row_ptr[i]<1 || row_ptr[i]>data_pars.row.pnts ) EXC(err_row);
            i++;
          }
        } while( stop == FALSE && i < max_rows );
        /*----------------------+
        ! Display row selection !
        +----------------------*/
        SPECTRUM_number_remove();
        update_row_range_array( row_ptr, i );
        free( row_ptr );
        VIEW_curve_init( 0, 0 );
      E_handle
      E_others  free( row_ptr );
                EXC_reraise;
      Exc_end
    }
  }else{
        EXC( err_mode );
  }
  /*-------------+
  ! Routine exit !
  +-------------*/
  DEB("\nROI_select_proc_seq   exit" );
}



/*****************************************************************************/
/* Function averages the data for the seleted rows.
/* The average row is displayed.
/*****************************************************************************/
void average_selected_row_array()
{
      int   num_bytes, new_num_rows;
      char  string[DEF_STR_LEN];
      int i,j;
      float *avg_data,*f_ptr;


      /*-----------------------------------------+
      ! If not valid, display an error message.    !
      +-----------------------------------------*/
      if (Global_row_number < 1) {
        printf ("Illegal value for  Global_row_number = %d\n",Global_row_number);
        return;
      }

      DATAVIEW_read_par(&data_pars);

      /*---------------------------------+
      ! Read other viewing parameters and !
      ! initialize some other parameters. !
      +---------------------------------*/
      VIEW_read_curve_layout(&curve_layout);
```

```
      VIEW_read_curve_window(&curve_window);
      VIEW_read_axis_layout(&axis_layout);
      row_range  = (Row_range *)VIEW_row_range_ptr();
      curve_pw   = (Pixwin *)get_base_pixwin_ptr();
      data_ptr   = (float *)VIEW_data_ptr();
      erase_row  = FALSE;

      /*---------------------------------+
      ! Hidden line removal is restarted. !
      +---------------------------------*/
      init_hidden_line();
      /*------------------+
      ! Clear the pixwin. !
      +------------------*/
      clear_pixwin_region(curve_pw, 0, 0, MAX_X_PIXEL-1, MAX_Y_PIXEL-1);
      /*------------------+
      ! Refresh the axis. !
      +------------------*/
      VIEW_draw_axis();
      VIEW_calc_scale();

      /*------------------+
      ! Refresh the rows. !
      +------------------*/

      avg_data = (float *)malloc(data_pars.col.pnts * sizeof(float));

      for (j=0; j < data_pars.col.pnts; j++ ) {

        avg_data[j] = 0.0;

        for (i = 0; i < Global_row_number; i++) {

          f_ptr = (float *)(data_ptr + (((Global_selected_rows[i] - 1) * data_pars.c
ol.pnts) + j)) ;
          avg_data[j] += *f_ptr;

          }

        avg_data[j] /= Global_row_number;

      }


      /*----------------------------+
      ! Display the average curve. !
      +----------------------------*/
      display_data(curve_pw, avg_data,
                   data_pars.col.pnts,
                   curve_layout.X_LEFT   ,
                   curve_layout.X_RIGHT  ,
                   curve_layout.Y_BOTTOM ,
                   curve_layout.Y_TOP    ,
                   curve_window.HOR_SET,
                   curve_window.X_LEFT, curve_window.X_RIGHT,
                   curve_window.VER_SET,
                   curve_window.Y_MIN, curve_window.Y_MAX,
                   curve_layout.HIDDEN_LINES,
                   curve_layout.HIDDEN_LINE_PTR,
                   BACKGROUND);


      /*-----------------------------+
      ! Save the current hidden line. !
      +-----------------------------*/
      save_hidden_line();

}

void average_and_save_selected_row_array()
{
  average_selected_row_array();
}
```

```
/*====================================================================*/
/*                                                                    */
/*      Copyright 1989 N.V. Philips' Gloeilampen Fabrieken.           */
/*      All rights reserved. Reproduction in whole or in part is      */
/*      prohibited without the written consent of the copyright owner.*/
/*                                                                    */
/*====================================================================*/

# include <stdio.h>
# include <string.h>
# include <suntool/sunview.h>
# include <suntool/panel.h>

# include "specglo.h"
# include "exception.h"
# include "debug.h"
# include "readgen_window.h"
# include "data.h"
# include "roi.h"                       /* jane 940708 */

# define        FILE_LEN                40
# define        ROW_DIG                 6
# define        VAX_FILE                0
# define        SUN_FILE                1

static          bool            first = TRUE;

static  char err_mode[] = "E: wrong mode (ROWDATAOUT_window)";
static  char err_type[] = "E: wrong filetype (ROWDATAOUT_window)";
static  char err_file[] = "E: wrong file specification (ROWDATAOUT_window)";
static  char err_rows[] = "E: wrong row specification (ROWDATAOUT_window)";
/*---------------------+
! Read data file items !
+---------------------*/
static  Frame           frame_id;
static  int             frame_x_pos = 618;
static  int             frame_y_pos = 22;
static  char            pos_frame_file[] = "rowdataeout.pos";
static  Panel           panel_id;
static  Panel_item      row_start_item;
static  Panel_item      row_stop_item;
static  Panel_item      file_item;
static  Panel_item      type_item;

/* jane 940708 */
static  Panel_item      specific_sv_row_item[MAX_SPECIFIC_SAVE_ROWS];
static  Panel_item      specific_sv_row_type_item[MAX_SPECIFIC_SAVE_ROWS];

bool            specific_row_sv_flag[MAX_SPECIFIC_SAVE_ROWS];
int             roi_sv_row_array[MAX_SPECIFIC_SAVE_ROWS];
char            roi_sv_row_type[MAX_SPECIFIC_SAVE_ROWS][6];

int             spec_sv_row_total;
int             specific_sv_row_num[MAX_SPECIFIC_SAVE_ROWS];
char            specific_sv_row_type[MAX_SPECIFIC_SAVE_ROWS][6];
/* jane 940708 */

static  int             row_start = 0;
static  int             row_stop = 0;
static  int             row_max = 0;
static  int             file_type = 0;
static  char            file_name[FILE_LEN]= { "" };
```



```
/*-------------------------------------------------------------+
!       cancel_proc                                           !
!                                                             !
! Exit routine for rowdataout window.                         !
+-------------------------------------------------------------*/
static  int             /* return : nothing                  */

cancel_proc()

{ /*-----------------+
  ! Local parameters !
  +-----------------*/
  int   x_pos;
  int   y_pos;

  DEB("\nCancel_proc  (ROWDATAOUT_window)" );
  /*--------------------+
  ! Save frame location !
  +--------------------*/
  x_pos = (int) window_get( frame_id , WIN_X , 0 );
  y_pos = (int) window_get( frame_id , WIN_Y , 0 );
  if( x_pos != frame_x_pos || y_pos != frame_y_pos ) {
    frame_x_pos = x_pos;
    frame_y_pos = y_pos;
    E_begin   READGEN_store_frame_pos( pos_frame_file , x_pos, y_pos );
    E_handle
    E_others
    Exc_end
  }
  /*---------------+
  ! Destroy window !
  +---------------*/
  window_set( frame_id , FRAME_NO_CONFIRM, TRUE, 0 );
  window_set( frame_id , WIN_SHOW, FALSE, 0 );
  window_destroy( frame_id );
  SUNSPEC1_free_function_selection();
  /*-------------+
  ! Routine exit !
  +-------------*/
  DEB("\nCancel_proc  exit  x,y = (%d,%d) (ROWDATAOUT_window)" ,
                  frame_x_pos , frame_y_pos );
  PROC_SEQ_window_start_next_comm( FUNC_NAME_WRITEDATA ); }
```

```
/*-----------------------------------------------------------------+
!       msg                                                       !
!                                                                 !
! Displays a message string in the window frame.                  !
+-----------------------------------------------------------------*/
static  int             /* return : nothing                      */

msg( msg )

char    *msg;

{ window_set( frame_id, FRAME_LABEL , msg , 0 ); }
```



```
/*-----------------------------------------------------------------+
!       write_rows_to_file                                        !
!                                                                 !
+-----------------------------------------------------------------*/
static  int             /* return : nothing                      */

write_rows_to_file( file, file_type, row_start , row_stop , row_max )

char    *file;
int     file_type;
int     row_start;
int     row_stop;
int     row_max;

{ /*-----------------+
  ! Local parameters !
  +-----------------*/
  int           index;
  int           num_complex;
  int           data_type;
  float         *re = FLOAT_NULL_PTR;
  float         *im = FLOAT_NULL_PTR;
  char          str[DEF_STR_LEN];
  char          par_file[DEF_STR_LEN];
  char          data_file[DEF_STR_LEN];
  bool          r5_file = FALSE;
  DATA_STRUCT   *in_par;

  DEB("\nWrite_rows_to_file  (ROWDATAOUT_window)" );
  DEB("\n (%s)   range = (%d,%d)  max=%d  type=%d",
                file, row_start, row_stop, row_max, file_type );

  if( file_type == VAX_FILE ) {
    /*---------------+
    ! Make filenames !
    +---------------*/
    sprintf(  par_file , "%s.spar", file );
    sprintf( data_file , "%s.sdat", file );
    data_type = DATA_vax_complex_float;
    r5_file = TRUE;
  }else if( file_type == SUN_FILE ) {
    /*---------------+
    ! Make filenames !
    +---------------*/
    sprintf(  par_file , "%s.sp", file );
    sprintf( data_file , "%s.sd", file );
    data_type = DATA_sun_complex_float_sep;
    r5_file = FALSE;
  }else{
    EXC( err_type );
  }

  /*---------------------+
  ! Read data parameters !
  +---------------------*/
  in_par = DATA_read_ptr( 'i' );
  DATAGEN_par_debug(  in_par );

  /*-----------+
  ! Write data !
  +-----------*/
  index = (row_start-1) * in_par->dim[0].pnts;
  re = &in_par->re[index];
  im = &in_par->im[index];
  num_complex = (row_stop-row_start+1) * in_par->dim[0].pnts;
  DATAOUT_write_data( data_file, re, im, num_complex, data_type );

  /*-----------------+
  ! Write parameters !
  +-----------------*/
/* jane 940716
  DATAOUT_write_pars( par_file, in_par, row_start, row_stop, row_max, r5_file );
```

```
*/
  DATAOUT_write_pars( par_file, in_par, row_start, row_stop, row_max, r5_file, spec_
sv_row_total, specific_sv_row_num, specific_sv_row_type );
  /*-------+
  ! Loging !
  +-------*/
  HIS("# %s   file = %s   rows = %d , %d",
        FUNC_NAME_WRITEDATA , par_file, row_start, row_stop );
  /*-------------+
  ! Routine exit !
  +-------------*/
  DEB("\nWrite_rows_to_file   exit  (ROWDATAOUT_window)" );
}
```



```
/*-------------------------------------------------------------+
!       write_all_proc                                         !
!                                                              !
+-------------------------------------------------------------*/
static  int             /* return : nothing                  */

write_all_proc()

{ /*-----------------+
  ! Local parameters !
  +-----------------*/
  int   i;              /* jane 940717 */
  DEB("\nWrite_all_proc  (ROWDATAOUT_window)" );


/* Jane 940717 */
  /*---------------------------+
  ! Save spec roi row and type !
  +---------------------------*/
  spec_sv_row_total = 0;
  for ( i = 0; i < specific_rows.sel_row_total; i++)
  {
      roi_sv_row_array[spec_sv_row_total] = specific_rows.sel_row_array[i];
      specific_sv_row_num[spec_sv_row_total] = specific_rows.sel_row_array[i];
      sscanf( panel_get_value( specific_sv_row_type_item[i] ) , "%s", roi_sv_row_typ
e[i] );
      strcpy( specific_sv_row_type[spec_sv_row_total], roi_sv_row_type[i]);
      spec_sv_row_total++;
  }


  /*-------------+
  ! Get filename !
  +-------------*/
  strcpy( file_name , panel_get_value( file_item ) );
  if( strlen( file_name ) <= 0 ) {
    msg( "Select output file" );
    return;
  }
  /*-------------+
  ! Get filetype !
  +-------------*/
  file_type = (int)panel_get_value( type_item );
  /*-----------+
  ! write rows !
  +-----------*/
  E_begin
        write_rows_to_file( file_name, file_type, 1, row_max , row_max );
  E_handle
  E_others
        msg( EXC_code );
        return;
  Exc_end
  cancel_proc();
  /*-------------+
  ! Routine exit !
  +-------------*/
  DEB("\nWrite_all_proc   exit  (ROWDATAOUT_window)" );
}
```

```
/*-----------------------------------------------------------------+
!       proceed_proc                                              !
!                                                                 !
+-----------------------------------------------------------------*/
static  int             /* return : nothing                      */

proceed_proc()

{ /*-----------------+
  ! Local parameters !
  +-----------------*/
  int   i;              /* jane 940715 */
  char  str[DEF_STR_LEN];

  DEB("\nProceed_proc  (ROWDATAOUT_window)" );

  /*-------------+
  ! Get rowrange !
  +-------------*/
  sscanf( panel_get_value( row_start_item ) , "%d", &row_start );
  sscanf( panel_get_value( row_stop_item  ) , "%d", &row_stop  );
  DEB("\n row_range = (%d,%d)", row_start, row_stop );
  if( row_start < 1 || row_start > row_stop || row_stop > row_max ) {
    msg( "E: wrong rowrange" );
    return;
  }

/* Jane 940710 */
  spec_sv_row_total = 0;
  for ( i = 0; i < specific_rows.sel_row_total; i++)
  {
      specific_row_sv_flag[i] = (bool)panel_get_value(specific_sv_row_item[i]);
      if (specific_row_sv_flag[i] == 0)
      {
          roi_sv_row_array[spec_sv_row_total] = specific_rows.sel_row_array[i];
          specific_sv_row_num[spec_sv_row_total] = specific_rows.sel_row_array[i];
          sscanf( panel_get_value( specific_sv_row_type_item[i] ) , "%s", roi_sv_row
_type[i] );
          strcpy( specific_sv_row_type[spec_sv_row_total], roi_sv_row_type[i]);
          spec_sv_row_total++;
      }
  }
/* jane 940708 */


  /*-------------+
  ! Get filename !
  +-------------*/
  strcpy( file_name , panel_get_value( file_item ) );
  if( strlen( file_name ) <= 0 ) {
    msg( "Select output file" );
    return;
  }
  sprintf( str, "file (%s)", file_name );
  /*-------------+
  ! Get filetype !
  +-------------*/
  file_type = (int)panel_get_value( type_item );
  /*-----------+
  ! write rows !
  +-----------*/
  E_begin
        write_rows_to_file( file_name, file_type,
                row_start, row_stop , row_max );
  E_handle
  E_others
        msg( EXC_code );
        return;
  Exc_end

  cancel_proc();
  /*-------------+
```



```
  ! Routine exit !
  +-------------*/
  DEB("\nProceed_proc   exit  (ROWDATAOUT_window)" );
}
```

```
/*-------------------------------------------------------------+
!       init_rowdataout_window                                !
!                                                             !
+-------------------------------------------------------------*/
static  int             /* return : nothing                  */

init_rowdataout_window()

{ /*-----------------+
  ! Local parameters !
  +-----------------*/
  int   i;                      /* jane 940708 */
  int   x_pos;
  int   y_pos;
  char  str[DEF_STR_LEN];

  DEB("\nInit_rowdataout_window  (ROWDATAOUT_window)" );

  /*--------------------------+
  ! Get saved frame position !
  +--------------------------*/
  if( first == TRUE ) {
    first = FALSE;
    E_begin
        READGEN_get_frame_pos( pos_frame_file , &x_pos, &y_pos );
        frame_x_pos = x_pos;
        frame_y_pos = y_pos;
    E_handle
    E_others
    Exc_end
  }

  /*-------------+
  ! Rowout panel !
  +-------------*/
  frame_id = window_create( NULL, FRAME, WIN_SHOW, FALSE,
                FRAME_LABEL,    "Save rowdata to file" ,
                WIN_ERROR_MSG,  "E: Can't create rowdata output window",
                WIN_X,          frame_x_pos,
                WIN_Y,          frame_y_pos,
                WIN_MOUSE_XY,   frame_x_pos + 100,
                                frame_y_pos + 50,
                0 );

  /*--------------------------+
  ! Change default frame menu !
  +--------------------------*/
  READGEN_window_change_frame_menu( frame_id);

  /*---------------------+
  ! Create rowout panels !
  +---------------------*/
  panel_id = window_create( frame_id , PANEL,
        WIN_COLUMNS,    65,
/*  jane 940715
        WIN_ROWS,       7,
*/
        WIN_ROWS,       10 + specific_rows.sel_row_total,
        0 );

  sprintf( str , "Number of rows in buffer : %d", row_max );
  (void)panel_create_item( panel_id , PANEL_MESSAGE,
        PANEL_ITEM_X,                   ATTR_COL(0),
        PANEL_LABEL_Y,                  ATTR_ROW(0),
        PANEL_LABEL_STRING,             str ,
        PANEL_LABEL_BOLD,               TRUE,
        0 );

  /*-----------------------------------------+
  ! Create rowrange and file selection panels !
  +-----------------------------------------*/
  sprintf( str , "%*d",  ROW_DIG ,row_start );
```



```
  row_start_item = panel_create_item( panel_id , PANEL_TEXT,
        PANEL_ITEM_X,                   ATTR_COL(0),
        PANEL_LABEL_Y,                  ATTR_ROW(2),
        PANEL_VALUE_DISPLAY_LENGTH,     ROW_DIG,
        PANEL_VALUE_STORED_LENGTH,      ROW_DIG,
        PANEL_LABEL_STRING,             "Write rows                 : " ,
        PANEL_VALUE,                    str ,
        PANEL_LABEL_BOLD,               TRUE,
        0 );

  sprintf( str , "%*d", ROW_DIG, row_stop );
  row_stop_item = panel_create_item( panel_id , PANEL_TEXT,
        PANEL_LABEL_Y,                  ATTR_ROW(2),
        PANEL_VALUE_DISPLAY_LENGTH,     ROW_DIG,
        PANEL_VALUE_STORED_LENGTH,      ROW_DIG,
        PANEL_LABEL_STRING,             "thru : " ,
        PANEL_VALUE,                    str ,
        PANEL_LABEL_BOLD,               TRUE,
        0 );

/* jane 940708 */
  /*-----------------------------------------+
  ! Initial specific_sv_row parameters      !
  +-----------------------------------------*/
  for ( i = 0; i < MAX_SPECIFIC_SAVE_ROWS; i++)
  {
      specific_row_sv_flag[i] = 1;
      roi_sv_row_array[i] = 0;
      specific_sv_row_num[i] = 0;
      strcpy(roi_sv_row_type[i], "");
      strcpy(specific_sv_row_type[i], "");
  }

  panel_create_item(panel_id, PANEL_MESSAGE,
                    PANEL_ITEM_X      , ATTR_COL(10),
                    PANEL_ITEM_Y      , ATTR_ROW(4),
                    PANEL_LABEL_BOLD  , TRUE,
                    PANEL_LABEL_STRING, "Select the rows and types to be saved:",
                    0);

  for ( i = 0; i < specific_rows.sel_row_total; i++) {
      sprintf(str, "SpecNo : %4d    ", specific_rows.sel_row_array[i]);
      specific_sv_row_item[i] = panel_create_item( panel_id, PANEL_CHOICE,
                                PANEL_ITEM_X        , ATTR_COL(2),
                                PANEL_ITEM_Y        , ATTR_ROW(5+i),
                                PANEL_LABEL_STRING  ,    str,
                                PANEL_CHOICE_STRINGS, "YES", "NO", 0,
                                PANEL_VALUE         , specific_row_sv_flag[i],
                                PANEL_FEEDBACK      , PANEL_INVERTED,
                                0);
      sprintf( str , "%s",  roi_sv_row_type[i] );
      specific_sv_row_type_item[i] = panel_create_item( panel_id , PANEL_TEXT,
        PANEL_LABEL_X,                  ATTR_COL(40),
        PANEL_LABEL_Y,                  ATTR_ROW(5+i),
        PANEL_LABEL_STRING,             "Type: " ,
        PANEL_VALUE_DISPLAY_LENGTH,     ROW_DIG+9,
        PANEL_VALUE_STORED_LENGTH,      ROW_DIG+9,
        PANEL_VALUE,                    roi_sv_row_type[i] ,
        PANEL_LABEL_BOLD,               TRUE,
        0 );
  }


  file_item = panel_create_item( panel_id , PANEL_TEXT,
        PANEL_ITEM_X,                   ATTR_COL(0),
        PANEL_LABEL_Y,                  ATTR_ROW(3+3+i),   /* jane */
        PANEL_VALUE_DISPLAY_LENGTH,     FILE_LEN,
        PANEL_VALUE_STORED_LENGTH,      FILE_LEN,
        PANEL_LABEL_STRING,             "Outputfile (no extension) : " ,
        PANEL_VALUE,                    file_name ,
        PANEL_LABEL_BOLD,               TRUE,
        0 );
  /*---------------------+
```

```
  ! Create filetype panel !
  +---------------------*/
  type_item = panel_create_item( panel_id , PANEL_CHOICE,
        PANEL_ITEM_X,                   ATTR_COL(0),
        PANEL_LABEL_Y,                  ATTR_ROW(4+3+i),  /* jane */
        PANEL_LABEL_STRING,             "Filetype                    : ",
        PANEL_CHOICE_STRINGS,            " GYROSCAN_R5 " , " SUNspec1 ",   0,
        PANEL_VALUE,                    file_type ,
        PANEL_FEEDBACK,                 PANEL_INVERTED,
        PANEL_LABEL_BOLD,               TRUE,
        0 );

  /*-------------------------------+
  ! Create window "Proceed" button !
  +-------------------------------*/
  (void) panel_create_item( panel_id, PANEL_BUTTON,
        PANEL_ITEM_X,                   ATTR_COL(0),
        PANEL_ITEM_Y,                   ATTR_ROW(6+3+i),   /* jane */
        PANEL_LABEL_IMAGE, panel_button_image( panel_id,"Proceed", 0 , 0 ),
        PANEL_NOTIFY_PROC, proceed_proc,
        0 );

  /*---------------------------------+
  ! Create window "Write all" button !
  +---------------------------------*/
  (void) panel_create_item( panel_id, PANEL_BUTTON,
        PANEL_ITEM_Y,                   ATTR_ROW(6+3+i), /* jane */
        PANEL_LABEL_IMAGE,
        panel_button_image( panel_id,"Write all + Proceed", 0 , 0 ),
        PANEL_NOTIFY_PROC, write_all_proc,
        0 );

  /*-------------------------------+
  ! Create window "Cancel" button !
  +-------------------------------*/
  (void) panel_create_item( panel_id, PANEL_BUTTON,
        PANEL_ITEM_Y,                   ATTR_ROW(6+3+i), /* jane */
        PANEL_LABEL_IMAGE, panel_button_image( panel_id,"Cancel", 0 , 0 ),
        PANEL_NOTIFY_PROC, cancel_proc,
        0 );

  window_fit( frame_id);

  SUNSPEC1_block_function_selection();

  /*-------------+
  ! Routine exit !
  +-------------*/
  DEB("\nInit_rowdataout_window  exit    (ROWDATAOUT_window)" );
}
```



```
/*MPF ::: FUNCTIONS :: ROW_OUT : STORE_STACKED_DATA_TO_FILE  =========

    FUNCTION NAME
        ROWDATAOUT_window

    DESCRIPTION
        Saves rowdata to file.

    ANYOTHER
        Author          : Ad Marien
        Creation Date   : 1989-03-02

    CALLING SEQUENCE
*/
int             /* return : nothing                                    */

ROWDATAOUT_window()

{ /*-----------------+
  ! Local parameters !
  +-----------------*/
  DEB("\nROWDATAOUT_window");

  /*------------------------+
  ! Check on data available !
  +------------------------*/
  E_begin       DATA_available( 'i' );
  E_handle
  E_others      return;
  Exc_end

  /*--------------+
  ! Get data pars !
  +--------------*/
  E_begin
    /*-----------------------------------+
    ! Estimate number of rows and columns !
    +-----------------------------------*/
    row_max = DATAFUNC_num_rows( 'i' );
    DEB("\n  rows = %d" , row_max );
    if( row_stop  < 1 || row_stop  > row_max  ) row_stop  = row_max;
    if( row_start < 1 || row_start > row_stop ) row_start = 1;
    /*------------------+
    ! Initialise window !
    +------------------*/
    init_rowdataout_window();
    window_set( frame_id , WIN_SHOW, TRUE, 0);
  E_handle
  E_others
  Exc_end
  /*-------------+
  ! Routine exit !
  +-------------*/
  DEB("\nROWDATAOUT_window   exit" );
}
```

```
/*MPF ::: FUNCTIONS ::  ROWDATAOUT: STORE_STACKED_DATA_TO_FILE =========

    FUNCTION NAME
        ROWDATAOUT_proc_seq

    DESCRIPTION
        Saves rowdata to file.

    ANYOTHER
        Author          : Ad Marien
        Creation Date   : 1989-03-02

    CALLING SEQUENCE
*/
int             /* return : nothing                                  */

ROWDATAOUT_proc_seq( fc,  mode , interactive )

FILE    *fc;            /* in : processing sequence file pointer        */
char    mode;           /* in : 'b' = batch mode, 'n' = normal mode     */
char    interactive;    /* in : 'i' = interactive else ' '              */

{ /*-----------------+
  ! Local parameters !
  +-----------------*/
  int   i;                  /* jane 940715 */
  int   file_type;
  int   row_max;
  int   low;
  int   high;
  char  str[DEF_STR_LEN];
  char  file_str[DEF_STR_LEN];
  char  type_str[DEF_STR_LEN];
  char  *ptr;

  if( DEB_is_on() == TRUE ) {
    DEB("\nROWDATAOUT_proc_seq");
    DEB("\n fc = %d    mode = %c  interactive = (%c)",
                fc , mode , interactive );
  }

  if( mode == 'n' && interactive == 'i' ) {
    /*---------------------+
    ! Interactive function !
    +---------------------*/
    ROWDATAOUT_window();
  }else if( mode == 'b' || mode == 'n' ) {
        /*---------------------------------+
        ! Batch or processing sequence mode !
         +---------------------------------*/
        DATA_available( 'i' );
        /*-------------------------------------------------------------+
        ! Get parameters                                               !
        ! Check on :                                                   !
        !       file = <file>  type = <type> rows = <low> , <high>     !
        !       file = <file>  type = <type> rows = all                !
        +-------------------------------------------------------------*/
        PARINPUT_window_next_str( fc , str );
        /*-----------------+
        ! Exctract filename !
        +-----------------*/
        ptr = str;
        STR_FUNC_skip_char( '=' , 'y' , &ptr );
        if( sscanf( ptr , "%s", file_str ) != 1 ) EXC( err_file );
        /*-----------------+
        ! Exctract file type !
        +-----------------*/
        STR_FUNC_skip_char( '=' , 'y' , &ptr );
        if( sscanf( ptr , "%s", type_str ) != 1 ) EXC( err_type );
        switch( type_str[0] ) {
          case 'G'      : file_type = VAX_FILE;         break;
          case 'S'      : file_type = SUN_FILE;         break;
```



```
          default       : EXC( err_type);               break;
        }
        /*---------------+
        ! Read row range !
        +---------------*/
        STR_FUNC_skip_char( '=' , 'y' , &ptr );
        if( strncmp( ptr , "all", 3 ) == 0 ) {
          /*---------------+
          ! Read all rows !
          +---------------*/
          low = 1;
          high = DATAFUNC_num_rows( 'i' );
        }else{
          if( sscanf( ptr , "%d", &low ) != 1 ) EXC( err_rows );
          STR_FUNC_skip_char( ',' , 'y' , &ptr );
          if( sscanf( ptr , "%d", &high ) != 1 ) EXC( err_rows );
        }

        /*------------------------+
        ! Write row range to file !
        +------------------------*/
        row_max = DATAFUNC_num_rows( 'i' );
        write_rows_to_file( file_str, file_type, low , high , row_max );
  }else{
        EXC( err_mode );
  }
  /*-------------+
  ! Routine exit !
  +-------------*/
  DEB("\nROWDATAOUT_proc_seq   exit" );
}
```

A.2 Source codes for Feature Extraction

A.2.1 FeatureExtract.c

```
/*****************************************************************/
/* Name: FeaureExtract.c                                         */
/* Date: July 30, 1994                                           */
/*                                                               */
/* Functions: Feature extraction from MR Spectroscopic Imaging's ROI */
/*            according to the spectrosocopic base line and its ppm. */
/*            Six feature elements are Cho,Cr,NAA,Ala,Lac and Lip.   */
/*            Six feature values are the six feature element values  */
/*                to the ratio of averaged normal Cr from the same   */
/*                patient.                                           */
/*            The feature value types defined by extracted value type*/

/*                peak, area, averaged peak, averaged area.          */

/*                                                               */
/* Compile: cc FeatureExtract.c -o FeatureExtract                */
/* Run    : FeatureExtract XX.sd YY.par ZZ.sp                    */
/*                                                               */
/* Input: .sd  file: spectrum data file.                         */
/*        .par file: MRSI parameter file.                        */
/*        .sp (or .spar) file: spectrum parameter file.          */
/*                                                               */
/* Output:   .peak file : Peak feature values of individual spectrum */
/*                        for one patient.                       */
/*           .area file : Area feature values of individual spectrum */
/*                        for one patient.                       */
/*         LSTpeak file : Peak feature values of individual spectrum */
/*                        for a group of patients.               */
/*         LSTarea file : Area feature values of individual spectrum */
/*                        for a group of patients.               */
/*         AVEpeak file : A group of patients feature values,and each*/
/*                        patient has Averaged Peak feature values   */
/*                        of tumour and normal spectra.          */
/*         AVEarea file : A group of patients feature values,and each*/
/*                        patient has Averaged Area feature values   */
/*                        of tumour and normal spectra.          */
/*           .dump file : Trace necessary information during feature */
/*                        extraction.                            */
/*                                                               */
/* Used by: Tumour_nn_data.c                                     */
/*                                                               */
/* Update : each file's path name                                */
          /*****************************************************************/

#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <math.h>

#define MAXSP_X 32
#define MAXSP_Y 32
#define MAXSP_LEN 512

#define ABS(x) ( ((x) < 0.0) ? -(x) : (x) )
#define CEIL(x) ( ((int)(x) == x) ? (x) : (int)(x)+1 )

#define MAX( x, y ) ( ((x) >= (y)) ? (x) : (y) )
#define MIN( x, y ) ( ((x) <= (y)) ? (x) : (y) )

#define PNAA 80
#define PCr 46
#define PCho 41
#define SPAN 10
#define RANGE0 6

#define PARANUM 26
#define NAA_P 0
#define CHO_P 1
#define CR_P  2
#define NAA_A 3
#define CHO_A 4
#define CR_A  5
#define NAA_RP 6
```



```
#define CHO_RP 7
#define NAA_RA 8
#define CHO_RA 9
#define NAA_RA_CHO_CR 10
#define NAA_RP_CHO 11
#define NAA_RA_CHO 12
#define CHO_CR 13

#define ALA_P 14
#define LAC1_P 15
#define LAC2_P 16
#define LAC3_P 17
#define LAC_P 18
#define LIP_P 19

#define ALA_A 20
#define LAC12_A 21
#define LAC23_A 22
#define LAC13_A 23
#define LAC_A 24
#define LIP_A 25

#define MAX_CNT_SPEC 20
#define MAXCHAR 100
#define SPECNUM 20
#define SPECDAT 150

typedef struct
{
   char    samp_name[20];
   float   elem[6];
   int     target[6];
}ROI_SAMPLE;

ROI_SAMPLE roi_six_peaks[MAX_CNT_SPEC];
ROI_SAMPLE roi_six_areas[MAX_CNT_SPEC];

int SP_X,SP_Y;
int SP_LEN;

int nml_cr_cnt;
int ab_nml_cnt;
float ab_nml_peak[6]; /* patient tumour voxel peak features */
float pt_nml_peak[6]; /* patient normal voxel peak features */
float ab_nml_area[6]; /* patient tumour voxel area features */
float pt_nml_area[6]; /* patient normal voxel area features */
float nml_ave_cr;     /* Contra-Cr, the averaged peak for normal Cr  */
float nml_ave_cr_area;/* Contra-Cr, the averaged area for normal Cr  */
int   roi_tot;        /* total of ROI spectra */
char curr_in_name[50];
char spname[20];
char tar_vector[6];   /* target output vector */
char curr_roi_tm_type[5];
char aver_roi_tm_type[5];
char aver_title[20];

FILE *roi_data_dump;  /* .dump file */

int vox_idx;          /* voxel index */
int  roi_vox_num[MAX_CNT_SPEC];     /* ROI voxel number      */
char roi_tm_type[MAX_CNT_SPEC][5];  /* ROI voxel tumour type */
int curr_vox;
int curr_vox_flag;
int ppm_start;
int ppm_end;
int ppm_step;

float ave_cho_lx, ave_cho_ly, ave_cr_rx, ave_cr_ry;
float ave_naa_lx, ave_naa_ly, ave_naa_rx, ave_naa_ry;
float ave_ala_lx, ave_ala_ly, ave_lac_rx, ave_lac_ry;
float ave_lip_lx, ave_lip_ly, ave_lip_rx, ave_lip_ry;
```

```
int peak[20];
float naa_a,cr_a,cho_a,naa_p,cr_p,cho_p,naa_rp,cho_rp;
float naa_ra,cho_ra,naa_ra_cho_cr;
float ala_p, lac_p, lac1_p, lac2_p, lac3_p, lip_p, ala_a;
float lac_a, lac12_a, lac23_a, lac13_a, lip_a;
float naa_ra_cho,naa_rp_cho;
char *filename,*parname;
char transferedname[20];
int ar0,ar1,ar2,ar3,ar4;
float intsy;
float spbuf_real[MAXSP_LEN];
int lineint();


/****************************************/
/*  Set target output for each sample    */
/****************************************/
void Set_target_output()
{
   int i;

   for (i=0; i<6; i++)
      tar_vector[i] = 0;

   if (!(strcmp(curr_roi_tm_type, "as1")))
      tar_vector[0] = 1;
   if (!(strcmp(curr_roi_tm_type, "as2")))
      tar_vector[1] = 1;
   if (!(strcmp(curr_roi_tm_type, "gbm")))
      tar_vector[2] = 1;
   if (!(strcmp(curr_roi_tm_type, "men")))
      tar_vector[4] = 1;
   if (!(strcmp(curr_roi_tm_type, "met")))
      tar_vector[3] = 1;
   if (!(strcmp(curr_roi_tm_type, "nml")))
      tar_vector[5] = 1;
}

/****************************************/
/*       Set Average file title         */
/****************************************/
void Set_aver_title()
{
     int normal_flag = 0;

     if ((strcmp(aver_title, "nml")) == 0)
        normal_flag = 1;
     strcpy(aver_title, "");

     aver_title[0]  = '/';
     aver_title[1]  = '*';
     aver_title[2]  = ' ';
     aver_title[3]  = curr_in_name[0];
     aver_title[4]  = curr_in_name[1];
     aver_title[5]  = curr_in_name[2];
     if (normal_flag)
     {
         aver_title[6] = 'N';
         aver_title[7] = 'o';
         aver_title[8] = 'r';
         aver_title[9] = 'm';
         aver_title[10] = ' ';
         aver_title[11]  = 'n';
         aver_title[12]  = 'm';
         aver_title[13] = 'l';
         aver_title[14]  = ' ';
         aver_title[15]  = '*';
         aver_title[16]  = '/';
     }
     else {
         aver_title[6] = 'T';
         aver_title[7] = 'u';
         aver_title[8] = 'm';
```



```
         aver_title[9] = 'r';
         aver_title[10] = ' ';
         aver_title[11]  = aver_roi_tm_type[0];
         aver_title[12]  = aver_roi_tm_type[1];
         aver_title[13] = aver_roi_tm_type[2];
         aver_title[14]  = ' ';
         aver_title[15]  = '*';
         aver_title[16]  = '/';
     }
}

/****************************************/
/*  Save AVEpeak and AVEarea files      */
/****************************************/
void Write_aver_patient_files()
{
   FILE *roi_aver_patient_peak;
   FILE *roi_aver_patient_area;
   int   aver_target[6];
   float temp_peak;
   float temp_area;
   char  aver_peak_file[100];
   char  aver_area_file[100];
   int   i, j, normal_flag = 0;

   strcpy(aver_peak_file, "/home/jane/featureExtract/out/");
   strcat(aver_peak_file, "AVEpeak");
   if ((roi_aver_patient_peak = fopen(aver_peak_file, "a")) == NULL)
   {
      printf("\n Can not open average patient file %s", aver_peak_file);
      exit(0);
   }


   strcpy(aver_area_file, "/home/jane/featureExtract/out/");
   strcat(aver_area_file, "AVEarea");
   if ((roi_aver_patient_area = fopen(aver_area_file, "a")) == NULL)
   {
      printf("\n Can not open average patient file %s", aver_area_file);
      exit(0);
   }

   while (normal_flag < 2)
   {
     strcpy(aver_title, "");
     normal_flag++;
     if ((nml_cr_cnt != 0) && (normal_flag == 1))
     {
       strcpy(aver_title, "nml");
       strcpy(curr_roi_tm_type, "nml");
       Set_aver_title();
       Set_target_output();
       for (j=0; j<6; j++)
           aver_target[j] = tar_vector[j];

       fprintf(roi_aver_patient_peak, "\n%s\n", aver_title);
       fprintf(roi_aver_patient_area, "\n%s\n", aver_title);
       for (i=0; i<6; i++)
       {
           temp_peak = pt_nml_peak[i]/nml_cr_cnt;
           fprintf(roi_aver_patient_peak, "%-8.2f ", temp_peak/nml_ave_cr);
           temp_area = pt_nml_area[i]/nml_cr_cnt;
           fprintf(roi_aver_patient_area, "%-8.2f ", temp_area/nml_ave_cr_area);
       }
       fprintf(roi_aver_patient_peak, "\n");
       fprintf(roi_aver_patient_area, "\n");
       for (i=0; i<6; i++)
       {
           fprintf(roi_aver_patient_peak, "%-8d ", aver_target[i]);
           fprintf(roi_aver_patient_area, "%-8d ", aver_target[i]);
       }
       fprintf(roi_aver_patient_peak, "\n");
       fprintf(roi_aver_patient_area, "\n");
```

```
      }
      else {
        if (ab_nml_cnt != 0)
        {
#ifdef DEBUG
          printf("\n aver_roi_tm_type=%s\n ", aver_roi_tm_type);
#endif
          strcpy(aver_title, aver_roi_tm_type);
          strcpy(curr_roi_tm_type, aver_roi_tm_type);
          Set_aver_title();
          Set_target_output();
          for (j=0; j<6; j++)
              aver_target[j] = tar_vector[j];

          fprintf(roi_aver_patient_peak, "\n%s\n", aver_title);
          fprintf(roi_aver_patient_area, "\n%s\n", aver_title);
          for (i=0; i<6; i++)
          {
              temp_peak = ab_nml_peak[i]/ab_nml_cnt;
              temp_area = ab_nml_area[i]/ab_nml_cnt;
              if (nml_cr_cnt == 0)
              {
                 fprintf(roi_aver_patient_peak, "%-8.2f ", temp_peak);
                 fprintf(roi_aver_patient_area, "%-8.2f ", temp_area);
              }
              else {
                 fprintf(roi_aver_patient_peak, "%-8.2f ", temp_peak/nml_ave_cr);
                 fprintf(roi_aver_patient_area, "%-8.2f ", temp_area/nml_ave_cr_area
);
             }
          }
          fprintf(roi_aver_patient_peak, "\n");
          fprintf(roi_aver_patient_area, "\n");
          for (i=0; i<6; i++)
          {
              fprintf(roi_aver_patient_peak, "%-8d ", aver_target[i]);
              fprintf(roi_aver_patient_area, "%-8d ", aver_target[i]);
          }
          fprintf(roi_aver_patient_peak, "\n");
          fprintf(roi_aver_patient_area, "\n");
        }
        normal_flag++;
      }
    }
    fclose(roi_aver_patient_peak);
    fclose(roi_aver_patient_area);
}


/*************************************************************/
/*  Read .sp(or .spar) file and get each ROI number and Type */
/*************************************************************/
void Read_sp_file()
{
    FILE *sp_file;
    char spfile[MAXCHAR];
    char line_buf[MAXCHAR];
    char temp[MAXCHAR];
    char temp_tm_type[5];
    int  i,j,k,temp_small_vox, index = 0;
    int  find_ppm_flag = 0;

    strcpy(spfile, "/home/jane/data/");
    strcat(spfile, spname);

    if ((sp_file = fopen(spfile, "r")) == NULL)
    {
       printf("\n Can not open parameter file %s : \n", spfile );
       exit(0);
    }
    else
    {
       while ((fgets(line_buf, MAXCHAR, sp_file)) != NULL)
```



```
       {
          if ((( strstr(line_buf, "ppm")) != NULL) || ((strstr(line_buf, "[Hz]")) !=
NULL))
             find_ppm_flag++;

          if (((strstr(line_buf, "_low_val")) != NULL) && (find_ppm_flag == 1))
             sscanf( strchr(line_buf, ':'),  "%s %f", temp, &ppm_start);

/*
          if (((strstr(line_buf, "spec_col_upper_val")) != NULL) && (find_ppm_flag ==
 1))
             sscanf( strchr(line_buf, ':'), "%s %f", temp, &ppm_end);
*/
          if (((strstr(line_buf, "_step")) != NULL) && (find_ppm_flag == 1))
          {
             sscanf( strchr(line_buf, ':'), "%s %f", temp, &ppm_step);
             find_ppm_flag++;
          }

          if (((strstr(line_buf, "spec_save_row_total")) != NULL) && (find_ppm_flag !
= 0))
             sscanf( strchr(line_buf, ':'), "%s %d", temp, &roi_tot);

          if (((strstr(line_buf, "spec_save_row_num")) != NULL) && (find_ppm_flag !=
0))
          {
             sscanf( strchr(line_buf, ':'), "%s  %d", temp, &roi_vox_num[index]);
             sscanf( strchr(line_buf, ','), "%s  %s %s %s", temp, temp, temp, roi_tm_
type[index]);

             index++;
          }
       }
       if (ppm_start < -3)
       {
          ppm_start = -0.214066;
          ppm_step = -0.030581;
#ifdef DEBUG
          printf("\n Column extension is not ppm. \n");
#endif
       }
    }

    for (i=0; i<roi_tot-1; i++)
     for (j=i+1; j<roi_tot; j++)
       if (roi_vox_num[i] > roi_vox_num[j])
       {
          temp_small_vox = roi_vox_num[i];
          roi_vox_num[i] = roi_vox_num[j];
          roi_vox_num[j] = temp_small_vox;
          strcpy(temp_tm_type, roi_tm_type[i]);
          strcpy(roi_tm_type[i], roi_tm_type[j]);
          strcpy(roi_tm_type[j], temp_tm_type);

       }
#ifdef DEBUG
    for (i=0; i<roi_tot; i++)
       printf("\n roi_vox_num[%2d] = %4d  %s", i, roi_vox_num[i], roi_tm_type[i]);
#endif
    fclose(sp_file);
}

/*****************************************/
/*  Get baseline function                */
/*****************************************/
int getlinefunc(x0,y0,x1,y1,AK)
float x0,y0,x1,y1,AK[2];
{
        if((x1-x0) == 0)
        return(-1);
        AK[0] = (y1-y0)/(x1-x0);
        AK[1] = y0-AK[0]*x0;
        return(1);
```

```
}

/*****************************************/
/*  Get areas feature values             */
/*****************************************/
void Get_areas_feature()
{
  int i,j;
  float ave0,ave1,ints,dt;
  float pos[20][2],tx,ty;
  float posx0,posy0,posx1,posy1;
  float a,b,c,r;
  float AK[2],AK1[2],AK2[2],AK3[2],AK4[2];
  float A1,A2,A3,A4,A5,S;

  /* peak[0] left of cho */
  /* peak[1] peak of cho */
  /* peak[2] right of cho, left of cr */
  /* peak[3] peak of cr */
  /* peak[4] right of cr */
  /* peak[5] left of naa */
  /* peak[6] peak of naa */
  /* peak[7] right of naa */

  cr_a = 0.; cho_a = 0.;  naa_a=0.;
  ala_a = lac_a = lip_a = lac12_a = lac23_a = lac13_a = 0.0; /* jane */

  ave0=0.;  ave1=0.;

  posx0=ave_cho_lx; posy0=ave_cho_ly; /* get avg position, left of cho */
  posx1=ave_cr_rx;  posy1=ave_cr_ry;  /* get avg position, right of cr */
                                      /* estimate baseline fn under cho+cr */
  if(getlinefunc(posx0,posy0,posx1,posy1,AK) == -1)    {
    printf("unreasonable line function for cho-cr\n");
    printf("x0=%f y0=%f x1=%f y1=%f\n",posx0,posy0,posx1,posy1);
    printf("%d %d %d %d %d %d %d %d\n",
          peak[0],peak[1],peak[2],peak[3],peak[4],peak[5],peak[6],peak[7]);
    for (i=0; i<150; ++i)
        printf ("%f\n",spbuf_real[i]);
    exit(-1);
  }
  A1=0.; A2=0.; A3=0.; A4=0.;   /* areas under CHO and CR */


                                /* calc CHO area */
  for(i=peak[0];i<=peak[2];i++)  {
    if(spbuf_real[i] > AK[0]*i+AK[1]) { /* if data above estimated baseline */
      A1+=spbuf_real[i];                /* add it to area calculation */
      A2+=AK[0]*i+AK[1];
    }
  }

                                /* calc CR area */
  for(j=peak[2]; j<=peak[4]; j++)  {
    if(spbuf_real[j] > AK[0]*j+AK[1]) { /* if data above estimated baseline */
      A3+=spbuf_real[j];                /* add it to area calculation */
      A4+=AK[0]*j+AK[1];
    }
  }

  cho_a=A1-A2;                  /* cho_a is the area over the baseline */
  cr_a=A3-A4;                   /* creatine area over baseline */

           /*  estimate baseline under the Naa peak: */
  ave0=0.;  ave1=0.;

  posx0=ave_naa_lx; posy0=ave_naa_ly; /* get avg position, left of naa */
  posx1=ave_naa_rx;  posy1=ave_naa_ry;  /* get avg position, right of naa */
  if(getlinefunc(posx0,posy0,posx1,posy1,AK)==-1) {
    printf("unreasonable line function for naa\n"); exit(-1);
  }

  A1=0.;  A2=0.;
```



```
                                /* calc Naa area */
  for(i=peak[6]-RANGE0/2;i<peak[6]+RANGE0/2;i++)  {
    if(spbuf_real[i] > AK[0]*i+AK[1]) {
      A1+=spbuf_real[i];
      A2+=AK[0]*i+AK[1];
    }
  }

  naa_a = A1-A2;                /* area of Naa peak, above the
                                   estimated baseline */

  if(cr_a<=0.001)
    cr_a = 0.001;

  naa_ra = naa_a/cr_a;
  cho_ra = cho_a/cr_a;

  naa_ra_cho_cr = naa_a/(cr_a + cho_a);
        if(cho_a <= 0.001) cho_a = 0.001;
        naa_ra_cho = naa_a/cho_a;

  /* jane 940727 add areas for ala_a, lac_a, lip_a */
  ave0 = ave1 = 0.0;
  posx0=ave_ala_lx; posy0=ave_ala_ly; /* avg positions, left  of ala */
  posx1=ave_lac_rx; posy1=ave_lac_ry;
  if(getlinefunc(posx0,posy0,posx1,posy1,AK)==-1) {
    printf("unreasonable line function for ala\n"); exit(-1);
  }

  A1=0.;  A2=0.;

                                /* calc ala area */
  for(i=peak[8];i<=peak[10];i++)  {
    if(spbuf_real[i] > AK[0]*i+AK[1]) {
      A1+=spbuf_real[i];
      A2+=AK[0]*i+AK[1];
    }
  }
  ala_a = A1-A2;

  A1=0.;  A2=0.;

                                /* calc lac area */
  for(i=peak[11];i<=peak[13];i++)  {
    if(spbuf_real[i] > AK[0]*i+AK[1]) {
      A1+=spbuf_real[i];
      A2+=AK[0]*i+AK[1];
    }
  }
  lac_a = A1-A2;


  A1 = A2 = 0.0;                        /* calc lac1,2 area */
  for(i=peak[11];i<=peak[15];i++)  {
    if(spbuf_real[i] > AK[0]*i+AK[1]) {
      A1+=spbuf_real[i];
      A2+=AK[0]*i+AK[1];
    }
  }
  lac12_a = A1-A2;

  A1 = A2 = 0.0;                        /* calc lac23 area */
  for(i=peak[15];i<=peak[13];i++)  {
    if(spbuf_real[i] > AK[0]*i+AK[1]) {
      A1+=spbuf_real[i];
      A2+=AK[0]*i+AK[1];
    }
  }
  lac23_a = A1-A2;
  lac13_a = lac12_a + lac23_a; /* separate calc lac sum area */

  ave0 = ave1 = 0.0;
```

```
   posx0=ave_lip_lx; posy0=ave_lip_ly; /* avg positions, left  of lip */
   posx1=ave_lip_rx; posy1=ave_lip_ry;
   if(getlinefunc(posx0,posy0,posx1,posy1,AK)==-1) {
     printf("unreasonable line function for lip\n"); exit(-1);
   }
   A1 = A2 = 0.0;                          /* calc lip area */
   for(i=peak[17];i<=peak[19];i++)  {
     if(spbuf_real[i] > AK[0]*i+AK[1]) {
       A1+=spbuf_real[i];
       A2+=AK[0]*i+AK[1];
     }
   }
   lip_a = A1-A2;


   if (curr_vox_flag == 1)
   {
#ifdef DEBUG
    printf("\n curr_vox_flag=%d, type=%s", curr_vox_flag, curr_roi_tm_type);
#endif
    if (strstr(curr_roi_tm_type, "nml") != NULL)
    {
       nml_ave_cr_area = nml_ave_cr_area + cr_a;
#ifdef DEBUG
       printf("\n Increase nml_ave_cr_area=%f , cr_a=%f ", nml_ave_cr_area, cr_a);
#endif

       /* calculate the average areas for all normal spectra */
       pt_nml_area[0] = pt_nml_area[0] + cho_a;
       pt_nml_area[1] = pt_nml_area[1] + cr_a;
       pt_nml_area[2] = pt_nml_area[2] + naa_a;
       pt_nml_area[3] = pt_nml_area[3] + ala_a;
       pt_nml_area[4] = pt_nml_area[4] + lac_a;  /* use average lac here */
       pt_nml_area[5] = pt_nml_area[5] + lip_a;
    }
    else {
       /* calculate the average areas for all abnormal spectra */
       ab_nml_area[0] = ab_nml_area[0] + cho_a;
       ab_nml_area[1] = ab_nml_area[1] + cr_a;
       ab_nml_area[2] = ab_nml_area[2] + naa_a;
       ab_nml_area[3] = ab_nml_area[3] + ala_a;
       ab_nml_area[4] = ab_nml_area[4] + lac_a;  /* use average lac here */
       ab_nml_area[5] = ab_nml_area[5] + lip_a;
    }
    roi_six_areas[vox_idx].elem[0] = cho_a;
    roi_six_areas[vox_idx].elem[1] = cr_a;
    roi_six_areas[vox_idx].elem[2] = naa_a;
    roi_six_areas[vox_idx].elem[3] = ala_a;
    roi_six_areas[vox_idx].elem[4] = lac_a;
    roi_six_areas[vox_idx].elem[5] = lip_a;

       fprintf(roi_data_dump, "\n\n ---AREA for patient %s, voxel ***   %d \n", curr_
in_name, curr_vox);
       fprintf(roi_data_dump, "\n  %f ", cho_a);
       fprintf(roi_data_dump, "  %f ", cr_a);
       fprintf(roi_data_dump, "  %f ", naa_a);
       fprintf(roi_data_dump, "  %f ", ala_a);
       fprintf(roi_data_dump, "  %f ", lac_a);
       fprintf(roi_data_dump, "  %f \n", lip_a);
    vox_idx++;
  }

}

/****************************************/
/*  Get peaks feature values            */
/****************************************/
float Get_peaks_feature()
{
  int i,j,k;
  char peak_title[20][15];
  int pmaxnaa,pmaxcr,pminnaa1,pminnaa2,pmincr1,pmincr2,pmaxcho,pmincho1,pmincho2;
  int pmaxala,pminala_1, pminala_r, pmaxlac,pminlac_1,pminlac_r,pmaxlip,pminlip_1,
```



```
  pminlip_r, pmaxlac1, pmaxlac2, pmaxlac3;
  int peak_ala_1, peak_ala_r, peak_lac_1, peak_lac_r, peak_lip_1, peak_lip_r;
  float peak_ala, peak_lac1, peak_lac2, peak_lac3, peak_lac, peak_lip;
  float area_ala,  area_lac, area_lip;
  int pnaa,pcr;
  int pcho=PCho;

  int pala, plac, plip;
  int plac_end, get_lac1_flag = 0, get_lac2_flag = 0, get_lac3_flag = 0;
  float ave0,ave1,ave2,ave3;
  float AK[2];
  float
    sum,
    max = 0,min1=6000,min2=6000,nom,den;
  pnaa = PNAA;  pcr = PCr;
  pmaxnaa=pnaa; pmaxcr=pcr;
  pminnaa1=pnaa-SPAN; pminnaa2=pnaa+SPAN;
  pmincr1=pcr-SPAN;   pmincr2=pcr-SPAN;

  for (i=0; i<20; i++)
     strcpy(peak_title[i], "");

  strcpy(peak_title[0], "pmincho2");
  strcpy(peak_title[1], "pmaxCHO");
  strcpy(peak_title[2], "mid_cho_cr");
  strcpy(peak_title[3], "pmaxCR");
  strcpy(peak_title[4], "pmincr1");
  strcpy(peak_title[5], "pminnaa2");
  strcpy(peak_title[6], "pmaxNAA");
  strcpy(peak_title[7], "pminnaa1");
  strcpy(peak_title[8], "pminala_1");
  strcpy(peak_title[9], "pmaxALA");
  strcpy(peak_title[10], "pminala_r");
  strcpy(peak_title[11], "pminlac_1");
  strcpy(peak_title[12], "pmaxLAC");
  strcpy(peak_title[13], "pminlac_r");
  strcpy(peak_title[14], "pmaxlac1");
  strcpy(peak_title[15], "pmaxlac2");
  strcpy(peak_title[16], "pmaxlac3");
  strcpy(peak_title[17], "pminlip_1");
  strcpy(peak_title[18], "pmaxLIP");
  strcpy(peak_title[19], "pminlip_r");

  sum=0.0;
  for (i=0; i<SP_LEN; ++i)
    sum += spbuf_real[i];

  if (sum < 0.000001)
    return(0.0);

  /* find peak */

  for(i=pnaa-SPAN;i<pnaa+SPAN;i++)
    {
      if(max < spbuf_real[i]) {max=spbuf_real[i];       pmaxnaa=i;}
    }

  for(i=pmaxnaa;i<pmaxnaa+SPAN;i++)
    {
      if(min1 > spbuf_real[i]) {min1=spbuf_real[i]; pminnaa1=i;}
    }
  for(i=pmaxnaa-SPAN;i<pmaxnaa;i++)
    {
      if(min2 > spbuf_real[i]) {min2=spbuf_real[i]; pminnaa2=i;}
    }
/* added to find peaks of cho and cr by ppm */

        pcr = pmaxnaa-32;       /* */
        pcho = pmaxnaa-40;      /* */

  max = 0;min1=6000;min2=6000;
  for(i=pcho-3;i<=pcho+3;i++)
```

```
    {
      if(max < spbuf_real[i]) {max=spbuf_real[i]; pmaxcho=i;}
    }
  for(i=pmaxcho;i<pmaxcho+4;i++)
    {
      if(min1 > spbuf_real[i]) {min1=spbuf_real[i]; pmincho1=i;}
    }
  for(i=pmaxcho-SPAN;i<pmaxcho;i++)
    {
      if(min2 > spbuf_real[i]) {min2=spbuf_real[i]; pmincho2=i;}
    }

  max = 0;min1=6000;min2=6000;
  for(i=pcr-3;i<pcr+3;i++)
    {
      if(max < spbuf_real[i]) {max=spbuf_real[i]; pmaxcr=i;}
    }
  for(i=pmaxcr;i<pmaxcr+SPAN;i++)
    {
      if(min1 > spbuf_real[i]) {min1=spbuf_real[i]; pmincr1=i;}
    }
  for(i=pmaxcr-3;i<pmaxcr;i++)
    {
      if(min2 > spbuf_real[i]) {min2=spbuf_real[i]; pmincr2=i;}
    }

  max = 0;min1=6000;min2=6000;
  for(i=pmaxcho;i<=pmaxcr;i++)
    {
      if(min1 > spbuf_real[i]) {min1=spbuf_real[i]; pmincho1=i;}
    }


 /*   find ala position range */
  pala  = pmaxnaa + 20;          /* */
  pmaxala = pala;
  max = 0;min1=6000;min2=6000;
  for(i=pala-1;i<pala+2;i++)
    {
      if(max < spbuf_real[i]) {max=spbuf_real[i]; pmaxala=i;}
    }
  peak_ala = max;
  for(i=pmaxala;i<pmaxala+2;i++)
    {
      if(min1 > spbuf_real[i]) {min1=spbuf_real[i]; pminala_r=i;}
    }
  for(i=pmaxala-5;i<pmaxala;i++)
    {
      if(min2 > spbuf_real[i]) {min2=spbuf_real[i]; pminala_l=i;}
    }

 /* find lac position range */
  if ((pmaxala - pmaxnaa) == 21)
  {
     plac  = pmaxnaa + 26;       /* */
     plac_end = plac + 5;
  }
  else {
     plac  = pmaxnaa + 25;       /* */
     plac_end = plac + 6;
  }
  max = 0;min1=6000;min2=6000;

  peak_lac2 = 1.0;
  for(i=pmaxala+1; i< plac_end; i++)
  {
      if((peak_lac1 < spbuf_real[i]) && (get_lac1_flag == 0))
      {
          peak_lac1=spbuf_real[i];
          pmaxlac1=i;
      } /* hl */
      else {
          get_lac1_flag = 1;
```



```
          if((peak_lac2 > spbuf_real[i]) && (get_lac3_flag == 0))
          {
              peak_lac2=spbuf_real[i];
              pmaxlac2=i;
              get_lac2_flag = 1;
          } /* low */
          else {
              get_lac3_flag = get_lac1_flag * get_lac2_flag;

              if ((peak_lac3 < spbuf_real[i]) && (get_lac3_flag == 1))
              {
                  peak_lac3=spbuf_real[i];
                  pmaxlac3=i;
              } /* hr */
          }
      }
  }
  if ((abs(pmaxlac2 - pmaxlac1) > 2) || (abs(pmaxlac3 - pmaxlac2) > 2))
  {
      peak_lac1 = peak_lac2 = peak_lac3 = 0;
      for(i=pmaxala+1;i<plac_end;i++)
      {
          if (peak_lac2 < spbuf_real[i])
          {
              peak_lac2 = spbuf_real[i];
              pmaxlac2 = i;
          }
      }
      pmaxlac1 = pmaxlac2 - 1;
      peak_lac1 = spbuf_real[pmaxlac1];
      pmaxlac3 = pmaxlac2 + 1;
      peak_lac3 = spbuf_real[pmaxlac3];
  }
  if (pmaxlac1 > pmaxlac2)
      pmaxlac1 = pmaxlac2;
  peak_lac = (peak_lac1 + peak_lac2 + peak_lac3) / 3;
  pmaxlac = (int)(pmaxlac1 + pmaxlac2 + pmaxlac3) / 3;

  for(i=pmaxlac3;i<pmaxlac3+6;i++)
    {
      if(min1 > spbuf_real[i]) {min1=spbuf_real[i]; pminlac_r=i;}
    }
  if (pmaxala+1 < pmaxlac1)
  {
     for(i=pmaxala+1;i<pmaxlac1;i++)
        if(min2 > spbuf_real[i]) {min2=spbuf_real[i]; pminlac_l=i;}
  }
  else
    pminlac_l = pmaxala;

 /* find lip position range */
  plip  = pmaxnaa + 38;          /* */
  max = 0;min1=6000;min2=6000;
  for(i=plip-4;i<plip+6;i++)
    {
      if(max < spbuf_real[i]) {max=spbuf_real[i]; pmaxlip=i;}
    }
  peak_lip = max;
  for(i=pmaxlip;i<pmaxlip+6;i++)
    {
      if(min1 > spbuf_real[i]) {min1=spbuf_real[i]; pminlip_r=i;}
    }
  for(i=pmaxlip-4;i<pmaxlip;i++)
    {
      if(min2 > spbuf_real[i]) {min2=spbuf_real[i]; pminlip_l=i;}
    }

        peak[0] = pmincho2;
        peak[1] = pmaxcho;
        peak[2] = (int)((pmaxcho+pmaxcr)/2);
        peak[3] = pmaxcr;
        peak[4] = pmincr1;
```

```
        peak[5] = pminnaa2;
        peak[6] = pmaxnaa;
        peak[7] = pminnaa1;

        peak[8] = pminala_l;
        peak[9] = pmaxala;
        peak[10] = pminala_r;
        peak[11] = pminlac_l;
        peak[12] = pmaxlac;
        peak[13] = pminlac_r;
        peak[14] = pmaxlac1;
        peak[15] = pmaxlac2;
        peak[16] = pmaxlac3;
        peak[17] = pminlip_l;
        peak[18] = pmaxlip;
        peak[19] = pminlip_r;

        ave0=0; ave1=0; ave2=0; ave3=0;
        for(i=peak[0];i>peak[0]-6;i--)
                ave0+=spbuf_real[i]/6;
        ave1 = (peak[0]-(6-1)/2);
        for(i=peak[4];i<peak[4]+6;i++)
                ave2+=spbuf_real[i]/6;
        ave3 = (peak[4]+(6-1)/2);

        ave_cho_lx = ave1;
        ave_cho_ly = ave0;
        ave_cr_rx  = ave3;
        ave_cr_ry  = ave2;
        getlinefunc(ave1,ave0,ave3,ave2,AK);
        nom = spbuf_real[peak[1]]-(AK[0]*peak[1]+AK[1]);
        den = spbuf_real[peak[3]]-(AK[0]*peak[3]+AK[1]);
        if (den<=0.001) den=0.001;
        cho_rp = nom/den;
        cho_p = nom;
        cr_p = den;

        ave0=0; ave1=0; ave2=0; ave3=0;
        for(i=peak[5];i>peak[5]-6;i--)
                ave0+=spbuf_real[i]/6;
        ave1 = (peak[5]-(6-1)/2);
        for(i=peak[7];i<peak[7]+6;i++)
                ave2+=spbuf_real[i]/6;
        ave3 = (peak[7]+(6-1)/2);
        ave_naa_lx = ave1;
        ave_naa_ly = ave0;
        ave_naa_rx  = ave3;
        ave_naa_ry  = ave2;
        getlinefunc(ave1,ave0,ave3,ave2,AK);
        nom = spbuf_real[peak[6]]-(AK[0]*peak[6]+AK[1]);
        if(den<=0.001) den=0.001;
        naa_p = nom;
        if(naa_p <0) naa_p = 0;
        naa_rp = naa_p/den;
        if(cho_p <= 0.001)
           naa_rp_cho = naa_p/0.001;
        else
           naa_rp_cho = naa_p/cho_p;

        ave0=0; ave1=0; ave2=0; ave3=0;
        for(i=peak[8];i>peak[8]-6;i--)
                ave0+=spbuf_real[i]/6;
        ave1 = (peak[8]-(6-1)/2);
        for(i=peak[13];i<peak[13]+6;i++)
                ave2+=spbuf_real[i]/6;
        ave3 = (peak[13]+(6-1)/2);
        ave_ala_lx = ave1;
        ave_ala_ly = ave0;
        ave_lac_rx = ave3;
        ave_lac_ry = ave2;
        getlinefunc(ave1,ave0,ave3,ave2,AK);
        nom = spbuf_real[peak[9]]-(AK[0]*peak[9]+AK[1]);
        den = spbuf_real[peak[12]]-(AK[0]*peak[12]+AK[1]);
```



```
        if (nom<=0.001) nom=0.001;
        if (den<=0.001) den=0.001;
        ala_p = nom;
        lac_p = den;
        lac1_p = spbuf_real[peak[14]]-(AK[0]*peak[14]+AK[1]);
        lac2_p = spbuf_real[peak[15]]-(AK[0]*peak[15]+AK[1]);
        lac3_p = spbuf_real[peak[16]]-(AK[0]*peak[16]+AK[1]);
        if (lac1_p<=0.001) lac1_p = 0.001;
        if (lac2_p<=0.001) lac2_p = 0.001;
        if (lac3_p<=0.001) lac3_p = 0.001;

        ave0=0; ave1=0; ave2=0; ave3=0;
        for(i=peak[17];i>peak[17]-6;i--)
                ave0+=spbuf_real[i]/6;
        ave1 = (peak[17]-(6-1)/2);
        for(i=peak[19];i<peak[19]+6;i++)
                ave2+=spbuf_real[i]/6;
        ave3 = (peak[19]+(6-1)/2);
        ave_lip_lx = ave1;
        ave_lip_ly = ave0;
        ave_lip_rx = ave3;
        ave_lip_ry = ave2;
        getlinefunc(ave1,ave0,ave3,ave2,AK);
        nom = spbuf_real[peak[18]]-(AK[0]*peak[18]+AK[1]);
        if (nom<=0.001) nom=0.001;
        lip_p = nom;


  if (curr_vox == roi_vox_num[vox_idx])
  {
      curr_vox_flag = 1;
      printf("  selected_spectra %4d  \n", curr_vox);

   strcpy(curr_roi_tm_type, roi_tm_type[vox_idx]);

   if (strstr(curr_roi_tm_type, "nml") != NULL)
   {
      nml_cr_cnt++;
      nml_ave_cr = nml_ave_cr + cr_p;
#ifdef DEBUG
      printf("\n Increase nml_ave_cr_peak=%f ,cr_p= %f, nml_cnt=%d ", nml_ave_cr, cr
_p, nml_cr_cnt);
#endif

      /* calculate the average peaks for all normal spectra */
      pt_nml_peak[0] = pt_nml_peak[0] + cho_p;
      pt_nml_peak[1] = pt_nml_peak[1] + cr_p;
      pt_nml_peak[2] = pt_nml_peak[2] + naa_p;
      pt_nml_peak[3] = pt_nml_peak[3] + ala_p;
      pt_nml_peak[4] = pt_nml_peak[4] + lac_p;  /* use average lac here */
      pt_nml_peak[5] = pt_nml_peak[5] + lip_p;
   }
   else {
      ab_nml_cnt++;
      /* calculate the average peaks for all abnormal spectra */
      ab_nml_peak[0] = ab_nml_peak[0] + cho_p;
      ab_nml_peak[1] = ab_nml_peak[1] + cr_p;
      ab_nml_peak[2] = ab_nml_peak[2] + naa_p;
      ab_nml_peak[3] = ab_nml_peak[3] + ala_p;
      ab_nml_peak[4] = ab_nml_peak[4] + lac_p;  /* use average lac here */
      ab_nml_peak[5] = ab_nml_peak[5] + lip_p;
   }
   roi_six_peaks[vox_idx].elem[0] = cho_p;
   roi_six_peaks[vox_idx].elem[1] = cr_p;
   roi_six_peaks[vox_idx].elem[2] = naa_p;
   roi_six_peaks[vox_idx].elem[3] = ala_p;
   roi_six_peaks[vox_idx].elem[4] = lac_p;
   roi_six_peaks[vox_idx].elem[5] = lip_p;


      fprintf(roi_data_dump, "\n\n *******   FEATURE positions for patient %s, voxel
***    %d \n", curr_in_name, curr_vox);
      for (k=0; k<20; k++)
```

```
      {
         if ((strstr(peak_title[k], "pmax")) != NULL)
            fprintf(roi_data_dump, "          %14s  %4d \n", peak_title[k], peak[k])
;
         else
            fprintf(roi_data_dump, " %14s  %4d \n", peak_title[k], peak[k]);
      }
      fprintf(roi_data_dump, "\n ---PEAKS for patient %s, voxel ***   %d", curr_in_n
ame, curr_vox);
      fprintf(roi_data_dump, "\n  %f ", cho_p);
      fprintf(roi_data_dump, "  %f ", cr_p);
      fprintf(roi_data_dump, "  %f ", naa_p);
      fprintf(roi_data_dump, "  %f ", ala_p);
      fprintf(roi_data_dump, "  %f ", lac_p);
      fprintf(roi_data_dump, "  %f \n", lip_p);
  }
  return(sum);
}

/****************************************/
/*  Calculate mean sum                  */
/****************************************/
float mean_sum(i,j,fp1,fp2)
int i,j;
float *fp1, *fp2;
{
  int
    x,y,count;
  float
    v1,v2,
    sum;

  count = 0;
  sum = 0.0;

  for ( x=i-1; x<=i+1; ++x)
    for ( y=j-1; y<=j+1; ++y) {
                v1 = 0.0;       v2 = 0.0;
      if (x>=0 && y>=0 && x<SP_X && y<SP_X) {
        v1 = *(fp1 + x*SP_X+y);
        v2 = *(fp2 + x*SP_X+y);

        if ((v1+v2) > 0.0) {
          ++count;
          sum += v1+v2;
        }
      }
    }

  if (count>0)
    return (sum/count);
  else
    return ( 0.0001 );

}

#define MAXLEN 100

/****************************************/
/*  Ignore those tiny feature values    */
/****************************************/
void extrem_cut(num,img,dim)
float img[PARANUM][MAXSP_X*MAXSP_Y];
int dim[4],num;
{

  int  extremlonum=8,extremhinum=4;
  int    index[MAXSP_X*MAXSP_Y],tmp,status=1,i,j=0;

  for(i=0;i<dim[0];i++)
    for(j=0;j<SP_X;j++)
      img[num][j*SP_X+i]=0;
  for(i=dim[1];i<SP_X;i++)
```



```
    for(j=0;j<SP_X;j++)
      img[num][j*SP_X+i]=0;
  for(i=0;i<SP_X;i++)
    for(j=0;j<dim[2];j++)
      img[num][j*SP_X+i]=0;
  for(i=0;i<SP_X;i++)
    for(j=dim[3];j<SP_X;j++)
      img[num][j*SP_X+i]=0;
  j=0;
  for(i=0;i<SP_X*SP_X;i++)
    if(img[num][i]!=0) { index[j]=i;    j++; }

  while (status==1) {
      status=0;
      for(i=1;i<j;i++) {
        if(img[num][index[i-1]] < img[num][index[i]]) {
          tmp=index[i]; index[i]=index[i-1]; index[i-1]=tmp; status=1;}
      }
    }
  for(i=0;i<extremlonum;i++) {
   img[num][index[j-i-1]]=img[num][index[j-extremlonum-1]];
  }
  for(i=0;i<extremhinum;i++) {
    img[num][index[i]]=img[num][index[extremhinum]];
  }
}

/****************************************/
/*  Read .par file                      */
/****************************************/
void readroi(dim)
int dim[4];
{
    FILE *fp,*fp_par;
    float par_ap_off_center,par_lr_off_center,par_ap_size,par_lr_size,
          phase_encoding_fov,SP_res;
    char buff[MAXLEN],tmp[MAXLEN];
    float tu,tv,SU[4],SV[4],tx,ty;
    int i;

    if ((fp_par = fopen(parname,"r"))==NULL) {
    sprintf (tmp,"cant open %s.\n",parname);
#ifdef DEBUG
    printf (tmp);
#endif
    return;

  }

  /* read in .par parameters:                                    */
  while ((fgets(buff,MAXLEN,fp_par) != NULL)) {
    if ( (strstr(buff,"ap_off_center")!=NULL) ) {
      sscanf(  strchr(buff,':') ,"%s %f",tmp,&par_ap_off_center);
#ifdef DEBUG
      printf("ap_off_center=%f\n",par_ap_off_center);
#endif
    }
    if ( (strstr(buff,"lr_off_center")!=NULL) ) {
      sscanf(  strchr(buff,':') ,"%s %f",tmp,&par_lr_off_center);
#ifdef DEBUG
      printf("lr_off_center=%f\n",par_lr_off_center);
#endif
    }
    if ( (strstr(buff,"ap_size")!=NULL) ) {
      sscanf(  strchr(buff,':') ,"%s %f",tmp,&par_ap_size);
#ifdef DEBUG
      printf("ap_size=%f\n",par_ap_size);
#endif
    }
    if ( (strstr(buff,"lr_size")!=NULL) ) {
      sscanf(  strchr(buff,':') ,"%s %f",tmp,&par_lr_size);
#ifdef DEBUG
      printf("lr_size=%f\n",par_lr_size);
```

```
#endif
    }

    if( (strstr(buff,"phase_encoding_fov")!=NULL) ) {
      sscanf( strchr(buff,')'), "%s %f",tmp,&phase_encoding_fov);
#ifdef DEBUG
      printf("phase_encoding_fov=%f\n",phase_encoding_fov);
#endif
    }
  }

  fclose(fp_par);
  /* calculate world coords of spectro box (in mm):
     (assuming head-first and supine)                              */
  tu=(SP_X)*par_lr_size/phase_encoding_fov;
  tv=(SP_X)*par_ap_size/phase_encoding_fov;

  /* establish voxel coordinates, for upper-left corner of VOI */

                                /* left border,mm*/
                                /* from left edge of FOV */
  tx=(phase_encoding_fov-par_lr_size)/2 - par_lr_off_center;
  tx=tx*(SP_X)/phase_encoding_fov; /* translate to voxels. */

                                /* top border,mm*/
                                /* from top edge of FOV */
  ty=(phase_encoding_fov-par_ap_size)/2 - par_ap_off_center;
  ty=ty*(SP_X)/phase_encoding_fov;

                                /* build corner coordinates of VOI,
                                   in SPECTRO VOXEL COORDINATES */
  SU[0] = tx+0;          SV[0] = ty+0;
  SU[1] = tx+0;          SV[1] = ty+tv;
  SU[2] = tx+tu;         SV[2] = ty+tv;
  SU[3] = tx+tu;         SV[3] = ty+0;
    dim[0]=CEIL(SU[0]); dim[1]=(int)SU[3]; dim[2]=CEIL(SV[0]); dim[3]=(int)SV[1];
}


/*********************************************/
/*  Save .peak, .area, LSTpeak, LSTarea files. */
/*********************************************/
void Write_roi_files()
{
   int    i, j;
   char   peak_file[80];
   char   area_file[80];
   char   list_peak_file[80];
   char   list_area_file[80];
   FILE   *roi_peak_file;
   FILE   *roi_area_file;
   FILE   *list_peaks;
   FILE   *list_areas;

   nml_ave_cr = nml_ave_cr / nml_cr_cnt;
   nml_ave_cr_area = nml_ave_cr_area / nml_cr_cnt;

#ifdef DEBUG
   printf("\n Before write, nml_ave_cr=%f, nml_ave_cr_area=%f, nml_cr_cnt=%d, ", nml
 ave_cr, nml_ave_cr_area, nml_cr_cnt);
#endif

   strcpy(peak_file, "");
   strcpy(area_file, "");

   strcpy(peak_file, "/home/jane/featureExtract/out/");
   strcpy(area_file, "/home/jane/featureExtract/out/");

   strcat(peak_file, spname);
   strcat(peak_file, ".peak");

   strcat(area_file, spname);
   strcat(area_file, ".area");
```



```
   if ((roi_peak_file=fopen(peak_file, "w")) == NULL)
   {
      printf("\n Can not open peak file");
      exit(0);
   }
   else {
      if ((roi_peak_file=fopen(peak_file, "a")) == NULL)
      {
          printf("\n Can not open peak file");
          exit(0);
      }
   }

   if ((roi_area_file=fopen(area_file, "w")) == NULL)
   {
      printf("\n Can not open area file");
      exit(0);
   }
   else {
      if ((roi_area_file=fopen(area_file, "a")) == NULL)
      {
          printf("\n Can not open area file");
          exit(0);
      }
   }
/*
   strcpy(list_peak_file, "");
   strcpy(list_peak_file, "/home/jane/featureExtract/out/");
   strcat(list_peak_file, "LSTpeak");
   if ((list_peaks = fopen(list_peak_file, "a")) == NULL)
    {
        printf("\n Can not open list_peaks file %s \n", list_peak_file);
        exit(0);
    }

   strcpy(list_area_file, "");
   strcpy(list_area_file, "/home/jane/featureExtract/out/");
   strcat(list_area_file, "LSTarea");
   if ((list_areas = fopen(list_area_file, "a")) == NULL)
    {
        printf("\n Can not open list_areas file %s \n", list_area_file);
        exit(0);
    }
*/
   for (i = 0; i < roi_tot; i++)
   {
      /* write peaks to the individual peak file */
      fprintf(roi_peak_file, "\n%s\n", roi_six_peaks[i].samp_name);
      for (j=0; j<6; j++)
         fprintf(roi_peak_file, "%-8.2f ", roi_six_peaks[i].elem[j]/nml_ave_cr);
      fprintf(roi_peak_file, "\n");
      for (j=0; j<6; j++)
         fprintf(roi_peak_file, "%-8d ", roi_six_peaks[i].target[j]);
      fprintf(roi_peak_file, "\n");

      /* write areas to the individual area file */
      fprintf(roi_area_file, "\n%s\n", roi_six_areas[i].samp_name);
      for (j=0; j<6; j++)
         fprintf(roi_area_file, "%-8.2f ", roi_six_areas[i].elem[j]/nml_ave_cr_area)
;
      fprintf(roi_area_file, "\n");
      for (j=0; j<6; j++)
         fprintf(roi_area_file, "%-8d ", roi_six_areas[i].target[j]);
      fprintf(roi_area_file, "\n");
   }

   strcpy(list_peak_file, "");
   strcpy(list_peak_file, "/home/jane/featureExtract/out/");
   strcat(list_peak_file, "LSTpeak");
   if ((list_peaks = fopen(list_peak_file, "a")) == NULL)
    {
        printf("\n Can not open list_peaks file %s \n", list_peak_file);
```

```
        exit(0);
    }
    else {
       for (i = 0; i < roi_tot; i++)
       {
          fprintf(list_peaks, "\n%s\n", roi_six_peaks[i].samp_name);
          for (j=0; j<6; j++)
             fprintf(list_peaks, "%-8.2f ", roi_six_peaks[i].elem[j]/nml_ave_cr);
          fprintf(list_peaks, "\n");
          for (j=0; j<6; j++)
             fprintf(list_peaks, "%-8d ", roi_six_peaks[i].target[j]);
          fprintf(list_peaks, "\n");
       }
    }
    strcpy(list_area_file, "");
    strcpy(list_area_file, "/home/jane/featureExtract/out/");
    strcat(list_area_file, "LSTarea");
    if ((list_areas = fopen(list_area_file, "a")) == NULL)
    {
        printf("\n Can not open list_areas file %s \n", list_area_file);
        exit(0);
    }
    else {
       for (i = 0; i < roi_tot; i++)
       {
          fprintf(list_areas, "\n%s\n", roi_six_areas[i].samp_name);
          for (j=0; j<6; j++)
             fprintf(list_areas, "%-8.2f ", roi_six_areas[i].elem[j]/nml_ave_cr_area
);
          fprintf(list_areas, "\n");
          for (j=0; j<6; j++)
             fprintf(list_areas, "%-8d ", roi_six_areas[i].target[j]);
          fprintf(list_areas, "\n");
       }
    }

    fclose(roi_peak_file);
    fclose(roi_area_file);
    fclose(list_peaks);
    fclose(list_areas);
}


/*****************************************/
/*  Write transfer file                  */
/*****************************************/
void  Write_trans_file(spimgbuf)
float  spimgbuf[PARANUM][MAXSP_X*MAXSP_Y];
{
   int i;
   FILE *fp;

  printf("\n Enter the trans file name ===> ");
  scanf("%s", transferedname);
  fp = fopen(transferedname,"w+b");
  if(fp == NULL){printf("cannot open file\n"); exit(-1);}
  if ((fp = fopen(transferedname,"a"))==NULL)
  {
     printf("\n can not open trans file ");
     exit(0);
  }
/*  for(i=0;i<PARANUM;i++)
    fwrite(spimgbuf[i],sizeof(float),SP_X*SP_Y,fp);
    fclose(fp);
*/
   fwrite(spimgbuf[NAA_A],sizeof(float),SP_X*SP_Y,fp);
   fwrite(spimgbuf[CHO_A],sizeof(float),SP_X*SP_Y,fp);
   fwrite(spimgbuf[CR_A],sizeof(float),SP_X*SP_Y,fp);
   fwrite(spimgbuf[NAA_RA],sizeof(float),SP_X*SP_Y,fp);
   fwrite(spimgbuf[NAA_RA_CHO_CR],sizeof(float),SP_X*SP_Y,fp);
   fwrite(spimgbuf[CHO_RA],sizeof(float),SP_X*SP_Y,fp);
  fclose(fp);
}
```



```
/*****************************************/
/*  MRSI feature extraction              */
/*****************************************/
void  Spectra_image_feature_extract()
{
  char title[20];
  int temp_vox_num;
  char temp_vox_digit[5];
  float
    spimgbuf[PARANUM][MAXSP_X*MAXSP_Y];
  FILE *fp;
  int i,j, k, tmp_vox;
  float
    nom,den;
  int dim[4];
  fp = fopen(filename,"r");
  if(fp == NULL){printf("Cannot open sd file\n"); exit(-1); }

  for(i=0;i<SP_X;i++)
    for(j=0;j<SP_Y;j++) {
       curr_vox_flag = 0;

      /* read in spectrum for voxel. */
      curr_vox = i * SP_X + j + 1;
      fread(spbuf_real,sizeof(float),SP_LEN,fp);


      if (Get_peaks_feature()!=0.0) {    /* find peaks  of Cho,Cr,Naa,Ala,Lac,Lip un
der baseline  */
        Get_areas_feature();             /* find areas of Cho,Cr,Naa,Ala,Lac,Lip  un
der base line */
        spimgbuf[NAA_P][i*SP_X+j]  = naa_p;
        spimgbuf[CHO_P][i*SP_X+j]  = cho_p;
        spimgbuf[CR_P][i*SP_X+j]   = cr_p;
        spimgbuf[NAA_A][i*SP_X+j]  = naa_a;
        spimgbuf[CHO_A][i*SP_X+j]  = cho_a;
        spimgbuf[CR_A][i*SP_X+j]   = cr_a;
        spimgbuf[NAA_RP][i*SP_X+j] = naa_rp;
        spimgbuf[CHO_RP][i*SP_X+j] = cho_rp;
        spimgbuf[NAA_RA][i*SP_X+j] = naa_ra;
        spimgbuf[CHO_RA][i*SP_X+j] = cho_ra;
        spimgbuf[NAA_RA_CHO_CR][i*SP_X+j] = naa_ra_cho_cr;
        spimgbuf[NAA_RP_CHO][i*SP_X+j] = naa_rp_cho;
        spimgbuf[NAA_RA_CHO][i*SP_X+j] = naa_ra_cho;
        spimgbuf[CHO_CR][i*SP_X+j] = cho_a+cr_a;

        spimgbuf[ALA_P][i*SP_X+j]  = ala_p;
        spimgbuf[LAC_P][i*SP_X+j]  = lac_p;
        spimgbuf[LAC1_P][i*SP_X+j]  = lac1_p;
        spimgbuf[LAC2_P][i*SP_X+j]  = lac2_p;
        spimgbuf[LAC3_P][i*SP_X+j]  = lac3_p;
        spimgbuf[LIP_P][i*SP_X+j]  = lip_p;

        spimgbuf[ALA_A][i*SP_X+j]  = ala_a;
        spimgbuf[LAC_A][i*SP_X+j]  = lac_a;
        spimgbuf[LAC12_A][i*SP_X+j]  = lac12_a;
        spimgbuf[LAC23_A][i*SP_X+j]  = lac23_a;
        spimgbuf[LAC13_A][i*SP_X+j]  = lac13_a;
        spimgbuf[LIP_A][i*SP_X+j]  = lip_a;
      }
      else {
        spimgbuf[NAA_P][i*SP_X+j]  = 0.0;
        spimgbuf[CHO_P][i*SP_X+j]  = 0.0;
        spimgbuf[CR_P][i*SP_X+j]   = 0.0;
        spimgbuf[NAA_A][i*SP_X+j]  = 0.0;
        spimgbuf[CHO_A][i*SP_X+j]  = 0.0;
        spimgbuf[CR_A][i*SP_X+j]   = 0.0;
        spimgbuf[NAA_RP][i*SP_X+j] = 0.0;
        spimgbuf[CHO_RP][i*SP_X+j] = 0.0;
        spimgbuf[NAA_RA][i*SP_X+j] = 0.0;
        spimgbuf[CHO_RA][i*SP_X+j] = 0.0;
        spimgbuf[NAA_RA_CHO_CR][i*SP_X+j] = 0.0;
```

```
        spimgbuf[NAA_RP_CHO][i*SP_X+j]   = 0.0;
        spimgbuf[NAA_RA_CHO][i*SP_X+j] = 0.0;
        spimgbuf[CHO_CR][i*SP_X+j] = 0.0;

        spimgbuf[ALA_P][i*SP_X+j]  = 0.0;
        spimgbuf[LAC_P][i*SP_X+j]  = 0.0;
        spimgbuf[LAC1_P][i*SP_X+j]  = 0.0;
        spimgbuf[LAC2_P][i*SP_X+j]  = 0.0;
        spimgbuf[LAC3_P][i*SP_X+j]  = 0.0;
        spimgbuf[LIP_P][i*SP_X+j]  = 0.0;

        spimgbuf[ALA_A][i*SP_X+j]  = 0.0;
        spimgbuf[LAC_A][i*SP_X+j]  = 0.0;
        spimgbuf[LAC12_A][i*SP_X+j]  = 0.0;
        spimgbuf[LAC23_A][i*SP_X+j]  = 0.0;
        spimgbuf[LAC13_A][i*SP_X+j]  = 0.0;
        spimgbuf[LIP_A][i*SP_X+j]  = 0.0;

      }
#ifdef DEBUG
     printf("%f        %f    %d  %d\n",naa_a,cho_a,i,j);
#endif
  }
  readroi(dim);
  extrem_cut(4,spimgbuf,dim);
  extrem_cut(5,spimgbuf,dim);

  for(i=0;i<SP_X;i++)
    for(j=0;j<SP_Y;j++) {
      spimgbuf[NAA_RA_CHO_CR][i*SP_X+j] =
        spimgbuf[NAA_A][i*SP_X+j] / mean_sum(i,j,spimgbuf[CHO_A],spimgbuf[CR_A]);
      spimgbuf[NAA_RA][i*SP_X+j] = 2*
        spimgbuf[NAA_A][i*SP_X+j]/mean_sum(i,j,spimgbuf[CR_A],spimgbuf[CR_A]);
      spimgbuf[NAA_RP][i*SP_X+j] = 2*
        spimgbuf[NAA_P][i*SP_X+j]/mean_sum(i,j,spimgbuf[CR_P],spimgbuf[CR_P]);
      spimgbuf[CHO_RP][i*SP_X+j] = 2*
        spimgbuf[CHO_P][i*SP_X+j]/mean_sum(i,j,spimgbuf[CR_P],spimgbuf[CR_P]);
      spimgbuf[CHO_RA][i*SP_X+j] = 2*
        spimgbuf[CHO_A][i*SP_X+j]/mean_sum(i,j,spimgbuf[CR_A],spimgbuf[CR_A]);
    }
  fclose(fp);

  /* for save space, not use this file now.      940729
  Write_trans_file(spimgbuf);
  */

  for (i=0; i<roi_tot; i++)
  {
      strcpy(curr_roi_tm_type, roi_tm_type[i]);
      /* get abnormal tumour type  at the first time */
      if (strcmp(curr_roi_tm_type, "nml"))
          strcpy(aver_roi_tm_type, curr_roi_tm_type);
      Set_target_output();
      strcpy(curr_in_name, spname);

      /* get the 4 digits voxel number 0001-1024 */
      strcpy(temp_vox_digit, "");
      temp_vox_num = roi_vox_num[i];
      temp_vox_digit[0] = (int)temp_vox_num/1000 + 0x30;
      temp_vox_num = temp_vox_num % 1000;
      temp_vox_digit[1] = (int)temp_vox_num/100 + 0x30;
      temp_vox_num = temp_vox_num % 100;
      temp_vox_digit[2] = (int)temp_vox_num/10 + 0x30;
      temp_vox_num = temp_vox_num % 10;
      temp_vox_digit[3] = (int)temp_vox_num + 0x30;


         strcpy(title, "");
         title[0]  = '/';
         title[1]  = '*';
         title[2]  = ' ';
         title[3]  = curr_in_name[0];
         title[4]  = curr_in_name[1];
```



```
         title[5]  = curr_in_name[2];
         for (j=6; j<10; j++)
            title[j] = temp_vox_digit[j-6];
/*
         if (i < 10)
            title[6] = i + 0x30;
         else
            title[6] = i + 0x37;
         title[7]  = '_';
         title[8]  = curr_roi_tm_type[0];
         title[9]  = curr_roi_tm_type[1];
         title[10] = curr_roi_tm_type[2];
         title[11]  = ' ';
         title[12]  = '*';
         title[13]  = '/';
*/
         title[10]  = '_';
         title[11]  = curr_roi_tm_type[0];
         title[12]  = curr_roi_tm_type[1];
         title[13]  = curr_roi_tm_type[2];
         title[14]  = ' ';
         title[15]  = '*';
         title[16]  = '/';
         strcpy(roi_six_peaks[i].samp_name, title);
         strcpy(roi_six_areas[i].samp_name, title);
         for (j=0; j<6; j++)
         {
            roi_six_peaks[i].target[j] = tar_vector[j];
            roi_six_areas[i].target[j] = tar_vector[j];
         }
   }
   Write_roi_files();
   Write_aver_patient_files();
   fclose(roi_data_dump);
}

/*****************************************/
/*  Initial variables                    */
/*****************************************/
Init_variables()
{
   int    i, j;
   char   dump_file[80];

   roi_tot = 0;
   for (i=0; i<20; i++)
   {
     roi_vox_num[i] = 0;
     strcpy(roi_tm_type[i], "");
   }

   Read_sp_file();

   ab_nml_cnt = 0;
   nml_cr_cnt = 0;
   nml_ave_cr =0.0;
   nml_ave_cr_area = 0.0;
   strcpy(curr_in_name, "");

   for (i=0; i<6; i++)
   {
      ab_nml_peak[i] = 0.0;
      pt_nml_peak[i] = 0.0;
      ab_nml_area[i] = 0.0;
      pt_nml_area[i] = 0.0;
   }

   for (i=0; i<MAX_CNT_SPEC; i++)
   {
      strcpy(roi_six_peaks[i].samp_name, "");
      strcpy(roi_six_areas[i].samp_name, "");
      for (j=0; j<6; j++)
      {
```

```
            roi_six_peaks[i].elem[j] = 0.0;
            roi_six_areas[i].elem[j] = 0.0;
        }
    }

    /* open dump file */
    strcpy(dump_file, "");
    strcpy(dump_file, "/home/jane/featureExtract/out/");

    strcat(dump_file, spname);
    strcat(dump_file, ".dump");

    if ((roi_data_dump=fopen(dump_file, "w")) == NULL)
    {
       printf("\n Can not open des file");
       exit(0);
    }
    else {
       if ((roi_data_dump=fopen(dump_file, "a")) == NULL)
       {
           printf("\n Can not open dump file\n");
           exit(0);
       }
    }
}

/************************************************/
/*  Main routine                                */
/*  called by: FeatureExtract XX.sd YY.par ZZ.sp */
/************************************************/
main(argc,argv)
int argc;
char *argv[];
{
  int i, j, temp_small_vox;
  char *voxname;

  if(argc < 2) {
    printf("USAGE: sp2img .sd_filename  .par_filename .sp_filename \n");
    exit(-1);
  }
  voxname = argv[3];
  strcpy(spname, voxname);
  parname=argv[2];
  filename = argv[1];
  SP_LEN = 150;
  SP_X = 32;
  SP_Y=SP_X;
  printf("\n .sd file= %s,     .par file= %s,     .sp file= %s\n ", filename, parnam
e, spname);
  Init_variables();

  Spectra_image_feature_extract();
  exit(0);
}
```

Appendix B

Source Codes for Neural Network

B.1 Source Codes for transforming data to Neural Net Format

B.1.1 Tumour_nn_data.c

```
/*************************************************************/
/* Name: Tumour_nn_data.c                                    */
/* Date: July 26, 1994                                       */
/*                                                           */
/* Functions:  Compress the data into [0,1] from the raw data*/
/*             to fit the neural nets package's constrains.  */
/*                                                           */
/* Input: in_file (the output file produced by FeatureExtract.c)*/
/* Output:in_file.nn (defalut tm_data.nn data set in [0,1]   */
/*                                                           */
/* Called by: Tumour_set_test.c                              */
/* Calls: None                                               */
/*                                                           */
/* Used by: Tumour_set_test.c                                */
/*************************************************************/
#include <stdio.h>
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

/*
#define MAX_SAMPLE 105
*/

#define MAXSAMPLE  1000
#define MAX_ELEM   6


  char name[MAXSAMPLE][15];

  char in[MAX_ELEM][8];
  char out[MAX_ELEM][8];

  double in_f[MAXSAMPLE][6];
  double out_f[MAXSAMPLE][6];

  double in_max[MAX_ELEM];
  double out_max[MAX_ELEM];

  double in_max_data;
  double out_max_data;

  double in_pre_data[MAXSAMPLE][6];
  double out_pre_data[MAXSAMPLE][6];

int MAX_SAMPLE;


void Clear_buffer()
{
   int i;
   for (i=0; i<MAXSAMPLE; i++)
      strcpy(name[i], "");
}

void Clear_vars()
{
   int i,j;

   for (i=0; i<MAX_ELEM; i++)
   {
      strcpy(in[i], "");
      strcpy(out[i], "");
   }
}

/************************************************/
/* Get the maximum feature value from all samples */
/************************************************/
void Get_max_elem()
{
   int i, j;
   double in_max[MAX_ELEM];
```



```
   double out_max[MAX_ELEM];


   for (j=0; j<MAX_ELEM; j++)
   {
      in_max[j] = in_f[0][j];
      out_max[j] = out_f[0][j];

      for (i=1; i<MAX_SAMPLE; i++)
      {
         if (in_f[i][j] > in_max[j])
            in_max[j] = in_f[i][j];

         if (out_f[i][j] > out_max[j])
            out_max[j] = out_f[i][j];

      }
   }

#ifdef DEBUG
   printf("\n\n MAX_in_data==> ");
   for (j=0; j<MAX_ELEM; j++)
       printf("  %f", in_max[j]);

   printf("\n\n MAX_out_data==> ");
   for (j=0; j<MAX_ELEM; j++)
       printf("  %f", out_max[j]);
#endif

   in_max_data = in_max[0];
   out_max_data = out_max[0];
   for (j=1; j<MAX_ELEM; j++)
   {
       if (in_max[j] > in_max_data)
          in_max_data = in_max[j];

       if (out_max[j] > out_max_data)
          out_max_data = out_max[j];
   }

#ifdef DEBUG
   printf("\n==================================");
   printf("\n\n In_MAX_DATA %f ", in_max_data);
   printf("\n\n Out_MAX_DATA %f ", out_max_data);
   printf("\n==================================\n");
#endif
}

/************************************************/
/* Suppress data to [0,1] divided by maximum value*/
/************************************************/
void Get_01_data()
{
   int i, j;

   for (i=0; i<MAX_SAMPLE; i++)
      for (j=0; j<MAX_ELEM; j++)
      {
         in_pre_data[i][j]  = 0.0;
         out_pre_data[i][j] = 0.0;
      }


   for (i=0; i<MAX_SAMPLE; i++)
      for (j=0; j<MAX_ELEM; j++)
      {
         in_pre_data[i][j]  = in_f[i][j]/in_max_data;
         out_pre_data[i][j] = out_f[i][j]/out_max_data;
      }
}


/************************************************/
```

```
/* Main routine for suppress data to [0,1]        */
/************************************************/
main()
{
   FILE *fp, *fp_out1;
   char in_file[80];
   char out_file[80];
   char in_name[20];
   char scan_in_name[20];
   int i,j, elem;
   char temp[20];
   char rtemp[10];
   int index = 0;

   Clear_buffer();
   Clear_vars();

   strcpy(scan_in_name, "");
   strcpy(in_name, "");
   strcpy(scan_in_name, "tm_data.org");
   strcpy(in_file, "/home/apollo/jane/project/my_xerion/bp/tumour/data/pk/");
   strcpy(out_file, "/home/apollo/jane/project/my_xerion/bp/tumour/data/");
   printf("\n Enter the original data file name (default: tm_data.org) ==>");
   scanf("%s", scan_in_name);

   strcat(in_file, scan_in_name);
   if ((fp = fopen(in_file, "r")) == NULL)
   {
      printf("\n can not open file");
      exit(0);
   }

/* Make data neural network data file main name */
   if (!(strcmp(scan_in_name, "LSTpeak")) ||
       !(strcmp(scan_in_name, "LSTarea")) ||
       !(strcmp(scan_in_name, "AVEpeak")) ||
       !(strcmp(scan_in_name, "AVEarea")))
      strcpy(in_name, scan_in_name);
   else {
      i = 0;
      j = 0;
      while (scan_in_name[i] != '.')
         in_name[j++] = scan_in_name[i++];

/* produce peak or area data file extension name */
      if (strstr(scan_in_name, "peak") != NULL)
      {
         in_name[j++] = '.';
         in_name[j] = 'p';
      }
      else {
         if (strstr(scan_in_name, "area") != NULL)
         {
            in_name[j++] = '.';
            in_name[j] = 'a';
         }
         else {
            printf("\n ERROR: Wrong in file name %s", scan_in_name);
            exit(0);
         }
      }
   }

/* make .nn suffix to data file name */
   strcat(out_file, in_name);
   strcat(out_file, ".nn");
   if ((fp_out1 = fopen(out_file, "w")) == NULL)
   {
      printf("\n can not open file");
      exit(0);
   }
   fclose(fp_out1);
```



```
   if ((fp_out1 = fopen(out_file, "a")) == NULL)
   {
      printf("\n can not open file");
      exit(0);
   }


   i = 0;
   fscanf(fp, "%s", name[index]);
   while (name[index][0] != NULL)
   {
      Clear_vars();
/* get title */
         if (name[index][0] == '/')
            fscanf(fp, "%s", name[index]);
         fscanf(fp, "%s", rtemp);

/* get input feature values */
         for (elem=0; elem<MAX_ELEM; elem++)
         {
            fscanf(fp, "%s", in[elem]);
            in_f[index][elem] = (double) atof(in[elem]);
         }

/* get target output */
         for (elem=0; elem<MAX_ELEM; elem++)
         {
            fscanf(fp, "%s", out[elem]);
            out_f[index][elem] = (double) atof(out[elem]);
         }

#ifdef DEBUG
         printf("\n name=%s", name[index]);
         printf("\n in-->");
         for (elem=0; elem<MAX_ELEM; elem++)
           printf(" %f ", in_f[index][elem]);

         printf("\n out-->");
         for (elem=0; elem<MAX_ELEM; elem++)
           printf(" %f ", out_f[index][elem]);
#endif
       index++;
       i++;
      fscanf(fp, "%s", name[index]);
   }
   MAX_SAMPLE = i;
#ifdef DEBUG
   printf("\n");
#endif

   Get_max_elem();
   Get_01_data();

/* Print out neural network data files */
   for (i=0; i<MAX_SAMPLE; i++)
   {
#ifdef DEBUG
       printf("\n name=%s", name[i]);
       printf("\n pre_in-->");
#endif

       fprintf(fp_out1, "\n");

       strcpy(temp, "");
       temp[0] = '/';
       temp[1] = '*';
       temp[2] = ' ';

       for (j=3; j<14; j++)
          temp[j] = name[i][j-3];

       temp[14] = ' ';
       temp[15] = '*';
```

```
        temp[16] = '/';
        temp[17] = '\0';

        fprintf(fp_out1, " %s", temp);

        fprintf(fp_out1, "\n");
        for (j=0; j<MAX_ELEM; j++)
           fprintf(fp_out1, " %f ", in_pre_data[i][j]);

#ifdef DEBUG
        printf("\n pre_out-->");
#endif
        fprintf(fp_out1, ", \n");
        for (j=0; j<MAX_ELEM; j++)
           fprintf(fp_out1, " %f ", out_pre_data[i][j]);
        fprintf(fp_out1, "; \n");
    }
     fclose(fp);
     fclose(fp_out1);
}
```

B.2 Source Codes for Creating Data Sets

B.2.1 Tumour_set_test.c

```
/*************************************************************/
/*  Name : data_set_test.c                                   */
/*  Date : July 28, 1994                                     */
/*                                                           */
/*  Function : Produce training data file and test data file */
/*             in xerion format automatically.               */
/*                                                           */
/*  Input  : .nn file produced by Tumour_nn_data.c           */
/*                                                           */
/*  Output : Xerion format train and test data sets.         */
/*                                                           */
/*    1. tr_XX.p and ts_XX.p : one patient area file.        */
/*         ts_XX.p : patient XX peak file.                   */
/*         tr_XX.p : rest of patients peak file but XX.      */
/*                                                           */
/*    2. tr_XX.a and ts_XX.a : one patient area file.        */
/*         ts_XX.a : patient XX area file.                   */
/*         tr_XX.a : rest of patients area file but XX.      */
/*                                                           */
/*    3. tr_LSTpeak or ts_LSTpeak : many patients peak file. */
/*                                                           */
/*    4. tr_LSTarea or ts_LSTarea : many patients area file. */
/*                                                           */
/*    5. tr_AVEpeak or ts_AVEpeak :                          */
/*         Averaged tumour and normal two peak samples for   */
/*         each patient, and gether many patients' data.     */
/*                                                           */
/*    6. tr_AVEarea or ts_AVEarea :                          */
/*         Averaged tumour and normal two area samples for   */
/*         each patient, and gether many patients' data.     */
/*                                                           */
/*    7. tr_XX.p.123 and ts_XX.p.123 : voxel peak data sets  */
/*         ts_XX.p.123 : voxel 123 peak file.                */
/*         tr_XX.p.123 : rest of voxels peak file but        */
/*                       voxel 123 for one patient.          */
/*                                                           */
/*    8. tr_XX.a.123 and ts_XX.a.123 : voxel area data sets  */
/*         ts_XX.a.123 : voxel 123 area file.                */
/*         tr_XX.a.123 : rest of voxels area file but        */
/*                       voxel 123 for one patient.          */
/*                                                           */
/*    Here XX is input file main name, and 123 is arbitrary  */
/*    ROI spectroscopic number for one patient.              */
/*                                                           */
/*                                                           */
/*  Run : 1. Enter the .nn file name by prompt.              */
/*            i.e. XX.p.nn                                  */
/*                                                           */
/*        2. Enter the test set name by prompt.              */
/*           if enter "all":                                 */
/*             all samples to test set ts_XX.p, no train set.*/
/*           if enter "none":                                */
/*             all samples to train set tr_XX.p, no test set.*/
/*           if enter YY:                                    */
/*             sample YY to test set ts_XX.p.YY,             */
/*             rest of samples to train set tr_XX.p.YY.      */
/*                                                           */
/*  Used by: addExamples in Xerion                           */
/*                                                           */
/*  Update : The source and destination file paths           */
/*           The data file size.                             */
/*************************************************************/
#include <stdio.h>
#include <stdlib.h>

#define MAX_ELEM    7
#define MAX_SAMPLE  1000


char train_buf[MAX_ELEM][10];
char test_buf[MAX_ELEM][10];

char org_file[100];
```



```
char tr_file[100];
char ts_file[100];
char org_path[65];
char des_path[65];
char org_name[25];

FILE *fp_org;
FILE *fp_tr_des, *fp_ts_des;

void Clear_vars()
{
   int i;
   for (i=0; i<MAX_ELEM; i++)
   {
      strcpy(train_buf[i], "");
      strcpy(test_buf[i], "");
   }
}

/*****************************************/
/* Open clean data                       */
/*****************************************/
void Open_org()
{
   strcpy(org_name, "tm_data.org.nn");

   strcpy(org_file, org_path);
   printf("\n Enter the nn data file name (default: tm_data.org.nn) ==> ");
   scanf("%s", org_name);
   strcat(org_file, org_name);
   if ((fp_org = fopen(org_file, "r")) == NULL)
   {
      printf("\n can not open file %s ", org_file);
      exit(0);
   }
}

/*****************************************/
/* Open train data set file              */
/*****************************************/
void Open_train_des()
{
   if ((fp_tr_des = fopen(tr_file, "w")) == NULL)
   {
      printf("\n can not open train file %s", tr_file);
      exit(0);
   }
   fclose(fp_tr_des);

   if ((fp_tr_des = fopen(tr_file, "a")) == NULL)
   {
      printf("\n can not open train file %s", tr_file);
      exit(0);
   }
}

/*****************************************/
/* Open test data set file               */
/*****************************************/
void Open_test_des()
{
   if ((fp_ts_des = fopen(ts_file, "w")) == NULL)
   {
      printf("\n can not open test file %s \n", ts_file);
      exit(0);
   }
   fclose(fp_ts_des);

   if ((fp_ts_des = fopen(ts_file, "a")) == NULL)
   {
      printf("\n can not open test file %s ", ts_file);
      exit(0);
   }
```

```
}


/*****************************************/
/* Set train data set                    */
/*****************************************/
void Set_train_file(char * sample_name)
{
   int elem;

   fprintf(fp_tr_des, "\n");

   fprintf(fp_tr_des, "%s", sample_name);
   fprintf(fp_tr_des, "\n");

   /* scan the input line */
   for (elem=0; elem<MAX_ELEM; elem++)
   {
      fscanf(fp_org, "%s", train_buf[elem]);
      fprintf(fp_tr_des, " %s ", train_buf[elem]);
   }
   fprintf(fp_tr_des, "\n");

   /* scan the target output line */
   for (elem=0; elem<MAX_ELEM; elem++)
   {
      fscanf(fp_org, "%s", train_buf[elem]);
      fprintf(fp_tr_des, " %s ", train_buf[elem]);
   }
   fprintf(fp_tr_des, "\n");
}

/*****************************************/
/* Set test  data set                    */
/*****************************************/
void Set_test_file(char * sample_name)
{
   int elem;

   fprintf(fp_ts_des, "\n");

   fprintf(fp_ts_des, "%s", sample_name);
   fprintf(fp_ts_des, "\n");

   /* scan the input line */
   for (elem=0; elem<MAX_ELEM; elem++)
   {
      fscanf(fp_org, "%s", test_buf[elem]);
      fprintf(fp_ts_des, " %s ", test_buf[elem]);
   }
   fprintf(fp_ts_des, "\n");

   /* scan the target output line */
   for (elem=0; elem<MAX_ELEM; elem++)
   {
      fscanf(fp_org, "%s", test_buf[elem]);
      fprintf(fp_ts_des, " %s ", test_buf[elem]);
   }
   fprintf(fp_ts_des, "\n");
}

/*****************************************/
/* Main routine for Set train and test sets*/
/*****************************************/
main()
{
   int i,j, found_flag;

   char in_file[20];
   char cut_nn_ext_name[25];

   char title[80];
   char tag_name[30];
```



```
   char temp[100];

   found_flag = 0;
   strcpy(org_path, "");
   strcpy(des_path, "");
   strcpy(org_file, "");
   strcpy(tr_file, "");
   strcpy(ts_file, "");

   strcpy(title, "# '$Id: tm.in, v1.0, 94/06/02 Jane EXP $' ");
   strcpy(org_path, "/home/apollo/jane/project/my_xerion/bp/tumour/data/");
   strcpy(des_path, "/home/apollo/jane/project/my_xerion/bp/tumour/");

   /* Open input .nn data file  */
   Open_org();

   /* Get rid of .nn suffix and get main name */
   strcpy(cut_nn_ext_name, org_name);
   j = 0;
   i = sizeof(cut_nn_ext_name);
   while (j < 2)
   {
      if (cut_nn_ext_name[i] == 'n')
         j++;
      i--;
   }
   cut_nn_ext_name[i] = '\0';

   /* Input test set name */
   printf("\n Enter the testing sample name, please ==> ");
   scanf("%s", in_file);

   if (strcmp(in_file, "") != 0)
   {
         strcpy(tr_file, des_path);
         strcpy(ts_file, des_path);
      if (strcmp(in_file, "none") == 0)
      {
         /* Only produce training set */
         found_flag = 1;
         strcat(tr_file, "tr_");
         strcat(tr_file, cut_nn_ext_name);

         printf("\n Training data set file ==> %s \n", tr_file);
         Open_train_des();

         fprintf(fp_tr_des, "\n");
         fprintf(fp_tr_des, "%s", title);
      }
      else {

         if (strcmp(in_file, "all") == 0)
         {
         /* Only produce testing set */
            strcat(ts_file, "ts_");
            strcat(ts_file, cut_nn_ext_name);
            printf("\n Testing data set file ==> %s \n", ts_file);

            Open_test_des();

            fprintf(fp_ts_des, "\n");
            fprintf(fp_ts_des, "%s", title);
         }
         else {
            /* Set both train and test sets' names */
            strcat(tr_file, "tr_");
            strcat(tr_file, cut_nn_ext_name);
            strcat(tr_file, ".");
            strcat(tr_file, in_file);

            printf("\n Training data set file ==> %s \n", tr_file);
            Open_train_des();
```

```
            strcat(ts_file, "ts_");
            strcat(ts_file, cut_nn_ext_name);
            strcat(ts_file, ".");
            strcat(ts_file, in_file);

            printf("\n Testing data set file ==> %s \n", ts_file);
            Open_test_des();
            fprintf(fp_tr_des, "\n");
            fprintf(fp_tr_des, "%s", title);
            fprintf(fp_ts_des, "\n");
            fprintf(fp_ts_des, "%s", title);
         }
      }

      Clear_vars();

      /* Save train and test sets file */
      strcpy(temp, "");
      fscanf(fp_org, "%s", temp); /* comment start */
      while (strcmp(temp, ""))
      {
         strcpy(tag_name, "");

         /* scan the comment line to get the tag name */
         if (temp[0] == '/')
         {
            strcpy(tag_name, "tag: ");
            fscanf(fp_org, "%s", temp); /* name */
            strcat(tag_name, temp);

            if ((strstr(temp, in_file) != NULL) || (strcmp(in_file, "all") == 0))
            {
               found_flag = 1;
               fscanf(fp_org, "%s", temp); /* comments end */
               Set_test_file(tag_name);
            }
            else
            {
               fscanf(fp_org, "%s", temp); /* comments end */
               Set_train_file(tag_name);
            }

            strcpy(temp, "");
            fscanf(fp_org, "%s", temp); /* comments start */
         }
         else
            printf("\n ERROR: Wrong string is %s ", temp);
      }
      if (found_flag == 0)
         printf("\n ERROR: Not found test data record %s ", in_file);
      fclose(fp_ts_des);
      fclose(fp_tr_des);
      printf("\n\n");
   }
   fclose(fp_org);

}
```

B.3 Script file for Build Neural Network:

B.3.1 Tumour_net_run.in

B.3.2 Tumour_net.layout

```
##################################################################
# Name : Tumour_net_run.in                                       #
#                                                                #
# Function : Build netural network for brain tumour diagnosis,   #
#            and initialize some variables. Start to run it      #
#            using bp simulator in Xerion.                       #
#                                                                #
# Run : bp-> read Tumour_net_run.in                              #
# Called by: None                                                #
# Calls    : Tumour_net.layout                                   #
##################################################################


############################################################
# an 6-?-6 tm network, change the value of NUMHIDDEN
# to set the number of hidden units.
# '$Id: tm.in,v 2.0 95/05/25 drew Exp $'
############################################################
# we need to unset this for some of our tests

unset unset-error
############################################################
# create NUMHIDDEN if it doesn't exist, and set it to a
# default value (3)
############################################################
if same "$NUMHIDDEN" "" ; then
    var int NUMHIDDEN
    set NUMHIDDEN = 3
fi
set unset-error

############################################################
# Now actually build the net
############################################################
addNet   "6-${NUMHIDDEN}-6 Tumour"
useNet   "6-${NUMHIDDEN}-6 Tumour"

addGroup -t INPUT  "Input"      6
addGroup           "Hidden"     $NUMHIDDEN
addGroup -t OUTPUT "Output"     6

disconnectGroups Bias           Input

connectGroups    "Input"        "Hidden"
connectGroups    "Hidden"       "Output"

orderGroups Bias Input Hidden Output

randomize

doLayout Tumour_net.layout
# loadWeights tm1.wts


set currentNet.weightCost = 0.0002
set currentNet.extension.zeroErrorRadius = 0.01
```

```
# '$Id: Tumour_net.layout,v 1.2 94/6/23 18:19:20 xerion Exp $'
#########################################################
# Name : Tumour_net.layout                              #
#                                                       #
# Function : Set up neural network strucure layout for  #
#            Activition area in Activition Display and  #
#            in Connection Display of Xerion simulator. #
# Called by: Tumour_net_run.in                          #
# Calls    : None                                       #
#########################################################
activity-layout  cell-size 15  margin 3  per-row 8
#       per-row $(calc 6 < $NUMHIDDEN ? $NUMHIDDEN : 6)
fill, all Output, fill
fill, all Hidden, fill
fill, all Input,  fill

minor-layout  cell-size 10  margin 3 per-row 8
fill, all Hidden, fill
fill, all Input, fill, blank 1, all Bias

major-layout  margin 3  per-row 3
[6] fill, next Output 1, fill
fill, all Hidden, fill
```

B.4 Script file for Tracing Neural Network:

B.4.1 runExamples

B.4.2 printTraceInfo

B.4.3 printAexample

```
#!xerion
# '$Id: runExamples,v 1.0 94/06/13 11:12:19 Jane Exp $'
#                                                              #
# Name : runExamples                                           #
#                                                              #
# Input : train_set test_set examine_bound                     #
#                                                              #
# Output: log/tr_XX.ex                                         #
#         log/ts_XX.ex                                         #
#                                                              #
# Run   : bp-> runExamples train_set test_set examine_bound#
# Calls : xerion/printTraceInfo                                #
#                                                              #
# Used by: Tumour_stat.c                                       #
################################################################

################################################################
# create exam_tr, exam_ts, examine_bound and y-n-response  #
# if they don't exist, prompt the usage format and exit    #
################################################################
unset unset-error
if same "$exam_tr" "" ; then
    var String exam_tr
fi
if same "$exam_ts" "" ; then
    var String exam_ts
fi
if same "$examine_bound" "" ; then
    var Real examine_bound
    set examine_bound = 0.2
fi
if same "$y-n-response" "" ; then
        var String y-n-response
        set y-n-response = n
fi
set unset-error

################################################################
# if no command args, prompt for the usage                 #
################################################################
if same $(calc ${tokc} < 3) "1"; then
    echo "  "
    echo "usage: ${tokv[0]} [train_set] [test_set] [examine_bound]"
    exit
else
    set exam_tr="${tokv[1]}"
    set exam_ts="${tokv[2]}"
    set examine_bound="${tokv[3]}"
fi

################################################################
# echo "    " $exam_tr $exam_ts $examine_bound              #
################################################################
var String set_train_test
var String exam_log
var String exam_ex

echo  "deleteExamples "
deleteExamples -t TRAINING
deleteExamples -t TESTING
echo "  "

########################################
set set_train_test = "TRAINING"
########################################

set exam_ex = $exam_tr

echo "addExamples " -n $exam_ex
addExamples -t $set_train_test $exam_ex
# doLayout tm.layout

# set currentNet.weightCost = 0.0002
```



```
# set currentNet.extension.zeroErrorRadius = 0.01


set exam_log = $exam_ex
sh rm -f log/$exam_log

echo "Calculate output and errors "
exec xerion/printTraceInfo Output > log/$exam_log
# sh rm -f $exam_ex

echo " "

########################################
set set_train_test = "TESTING"
########################################

set exam_ex = $exam_ts

echo "addExamples " -n $exam_ex
addExamples -t $set_train_test $exam_ex
# doLayout tm.layout

sh rm -f log/$exam_ex

echo "Calculate output and errors "
exec xerion/printTraceInfo Output > log/$exam_ex
# sh rm -f $exam_ex

echo "  "
```

```
#!xerion
# '$Id: xerion/printTraceInfo,v 1.0 94/06/26 11:12:26 Jane Exp $'
##############################################################
# Name : printTraceInfo                                      #
#                                                            #
# Description:                                               #
# Sets up a trace for 'doExamples' that prints the input,    #
# target output and the absolute error of all the units      #
# in an output group. If the error is more than the          #
# examine_bound, put the sample in failure catagory.         #
#                                                            #
# Functions: print out the target, output and error for      #
#               every example in Training set with the       #
#               example name as a identifier.                #
#                                                            #
# Called by : runExamples                                    #
# Calls     : printAexample                                  #
##############################################################

##############################################################
# create OutputName and y-n-response if they don't exist,    #
# then set them to default values ("Output", and "n")        #
##############################################################
unset unset-error
if same "$OutputName" "" ; then
    var String OutputName
    set OutputName = "Output"
fi
if same "$y-n-response" "" ; then
        var String y-n-response
        set y-n-response = n
fi
set unset-error

##############################################################
# if no command args, prompt for a value for OutputName      #
##############################################################
if same "$tokc" "1" ; then
    prompt OutputName "Enter the name of the output group: ($OutputName) "

##############################################################
# if one command argument, set OutputName to it              #
##############################################################
elif same "$tokc" "2" ; then
    set OutputName="${tokv[1]}"

##############################################################
# anything else is an error                                  #
##############################################################
else
    echo "usage: ${tokv[0]} [OutputName]"
    exit
fi

##############################################################
# Check if the Group exists                                  #
##############################################################
if groupLongName "$OutputName" > /dev/null ; then
    var String group
    set group = $(groupLongName "$OutputName")
else
    ${tokv[0]}
    exit
done

##############################################################
# create variables for holding the errors (an array)         #
##############################################################
var Real error[${${group}.numUnits}]
var Real target[${${group}.numUnits}]
var Real output[${${group}.numUnits}]
var Real input[${${group}.numUnits}]
```



```
var String dash_char
set dash_char = "-"

########################################
# for output file colum_7 to show the  #
# examine bound and Fail info.         #
########################################
var String input_7
var String output_7
var String target_7
var Real   error_7
set input_7 = $dash_char
set target_7 = $dash_char
set error_7 = $examine_bound

########################################
# for failure catagory                 #
########################################
var int  fail_max_idx
set fail_max_idx = 600

var String failure_exam[$fail_max_idx]
var Real fail_out0[$fail_max_idx]
var Real fail_out1[$fail_max_idx]
var Real fail_out2[$fail_max_idx]
var Real fail_out3[$fail_max_idx]
var Real fail_out4[$fail_max_idx]
var Real fail_out5[$fail_max_idx]
var int  failure_i
var int  failure_j
set failure_i = 0
set failure_j = 0

########################################
# clear failure variables              #
########################################
while same $(calc $failure_j < $fail_max_idx) "1" ; do
  set failure_exam[failure_j] = ""
  set fail_out0[failure_j] = 0.0
  set fail_out1[failure_j] = 0.0
  set fail_out2[failure_j] = 0.0
  set fail_out3[failure_j] = 0.0
  set fail_out4[failure_j] = 0.0
  set fail_out5[failure_j] = 0.0
  set failure_j = $(calc $failure_j + 1)
done
  set failure_j = 0

##############################################################
# generate a string containing "0 1 2 .. group.numUnits - 1"#
##############################################################
var int    idx
set idx = ${${group}.numUnits}

var String exam_set
var int example_id
set example_id = 0

##############################################################
# set up corresponding data set for the input data set       #
##############################################################
echo "      "
if same "$set_train_test" "TRAINING" ; then
   set exam_set = ${currentNet.trainingExampleSet.name}
elif same "$set_train_test" "TESTING" ; then
   set exam_set = ${currentNet.testingExampleSet.name}
else
   set exam_set = ${currentNet.validationExampleSet.name}
fi
print exam_set  exam_ex

##############################################################
# Now add the trace to 'doExamples' (we use another          #
```

```
# script file to make all the parsing simpler)              #
############################################################
 addTrace +0 doExamples "xerion/printAexample"

############################################################
# run doExamples for corresponding data set                 #
############################################################
 doExamples -t $set_train_test

############################################################
# delete the command we added to the trace                  #
############################################################
deleteTrace doExamples 0

############################################################
# print out failure samples                                 #
############################################################
echo "     "
echo " Failure        " "abs(TargetOutput-ActualOutput)=" $examine_bound

while same $(calc ${failure_j}< $fail_max_idx) "1" ; do
 if  same "${failure_exam[failure_j]}"  "" ; then
 elif same "${failure_exam[failure_j]}" "${failure_exam[$(calc $failure_j + 1)]}" ;
then
 else
  print exam_ex "    " failure_exam[failure_j] "  " fail_out0[failure_j] "%4.2f"    f
ail_out1[failure_j] "%4.2f" fail_out2[failure_j] "%4.2f"     fail_out3[failure_j] "%4
.2f" fail_out4[failure_j] "%4.2f" fail_out5[failure_j] "%4.2f"
 fi
  set failure_j = $(calc $failure_j + 1)
done
  echo ""
```

```
#!xerion
# '$Id: printAexample,v 1.0 94/06/26 11:12:20 Jane Exp $'
#####################################################################
# Name : printAexample                                              #
#                                                                   #
# Description:                                                      #
#  calculates and prints the error of all units on a specific       #
#  example and examines the failure samples.                        #
#  MUST BE CALLED BY printTraceInfo                                 #
#                                                                   #
#  Functions: Trace each example's information in net.              #
#             print out the target output, actual output,           #
#             error (abs(target output - actual output))            #
#             for all units on a specific example with the          #
#             example name as its identifier.                       #
#                                                                   #
#  Called by: printTraceInfo                                        #
#  Calls    : none                                                  #
#####################################################################
var int units_idx
var String units_name
var String example_name
var int example_max_num

var int curr_idx

var int failure_bound
set failure_bound = 0
var String exch

var int failure_memo
set failure_memo = 0

##############################################
# set up corresponding data set              #
##############################################
if same "$set_train_test" "TRAINING" ; then
    set example_max_num =  `calc $example_id < ${currentNet.trainingExampleSet.numEx
amples}`
    set exch = "training"
elif same "$set_train_test" "TESTING" ; then
    set example_max_num =  `calc $example_id < ${currentNet.testingExampleSet.numExa
mples}`
    set exch = "testing"
else
    set example_max_num =  `calc $example_id < ${currentNet.validationExampleSet.num
Examples}`
    set exch = "validation"
fi

##############################################
# For valid example, get its sample's name   #
##############################################
if  same "$example_max_num" "1" ; then
    set example_name = ${currentNet.${exch}ExampleSet.example[example_id].name}

##############################################
# Get input vector                           #
##############################################
set units_idx = 0
set curr_idx = ${currentNet.${exch}ExampleSet.example[example_id].numInputs}
while same $(calc $units_idx < $curr_idx) "1"; do
    set input[units_idx]  = ${currentNet.${exch}ExampleSet.example[example_id].input
Vector[0][units_idx]}
    set units_idx = $(calc $units_idx + 1)
done
    set example_id = $(calc $example_id + 1)

set units_idx = 0
set output_7 = $dash_char

##############################################
# Get target,actual output and error vectors#
```



```
##############################################
    print example_name  "  Cho      Chr      NAA      Ala      LA       Lip    Succ_
Fail"
fi
echo          "======== ======== ======== ======== ======== ======== ======== =====
====="
while same $(calc $units_idx < $idx) "1"; do
    set output[units_idx] =  ${${group}.unit[units_idx].output}
    set target[units_idx] =  ${${group}.unit[units_idx].target}

    set error[units_idx] = $(calc "abs(  ${target[units_idx]} - ${output[units_idx]}
)")

#    set units_name = ${${group}.unit[units_idx].name}

#####################################################
# Get failure samples, which error > examine-bound #
#####################################################
   set failure_bound =  `calc ${error[units_idx]} "\>" $examine_bound`

   if same "$failure_bound" "1" ; then
       set output_7 = "Fail"
       set failure_memo = 1
   fi

   set units_idx = $(calc $units_idx + 1)
done

   if same "$failure_memo" "1" ; then
       set failure_exam[failure_i] = $example_name

       set fail_out0[failure_i] = ${output[0]}
       set fail_out1[failure_i] = ${output[1]}
       set fail_out2[failure_i] = ${output[2]}
       set fail_out3[failure_i] = ${output[3]}
       set fail_out4[failure_i] = ${output[4]}
       set fail_out5[failure_i] = ${output[5]}

       set failure_i = $(calc $failure_i + 1)
       set failure_bound = 0.0
   fi

#####################################################
# Save trace information for one example           #
#####################################################
#    set units_idx = $(calc $units_idx + 1)
# done
    print "Input  "  input[0] "%9.6f" input[1] "%8.6f" input[2] "%8.6f" input[3] "%8
.6f" input[4] "%8.6f" input[5] "%8.6f" input_7 "%7s"
    print "Target "  target[0] "%9.6f" target[1] "%8.6f" target[2] "%8.6f" target[3]
 "%8.6f" target[4] "%8.6f" target[5] "%8.6f" target_7 "%7s"
    print "Output "  output[0] "%9.6f" output[1] "%8.6f" output[2] "%8.6f" output[3]
 "%8.6f" output[4] "%8.6f" output[5] "%8.6f" output_7 "%7s"
    print "Error  "  error[0] "%9.6f" error[1] "%8.6f" error[2] "%8.6f" error[3] "%8
.6f" error[4] "%8.6f" error[5] "%8.6f" error_7 "%7.2f"
   echo "          "
```

```
#!xerion
# '$Id: printAexample,v 1.0 94/06/26 11:12:20 Jane Exp $'
################################################################
# Name : printAexample                                         #
#                                                              #
# Description:                                                 #
#  calculates and prints the error of all units on a specific  #
#  example and examines the failure samples.                   #
#  MUST BE CALLED BY printTraceInfo                            #
#                                                              #
#  Functions: Trace each example's information in net.         #
#             print out the target output, actual output,      #
#             error (abs(target output - actual output))       #
#             for all units on a specific example with the     #
#             example name as its identifier.                  #
#                                                              #
#  Called by: printTraceInfo                                   #
#  Calls    : none                                             #
################################################################
var int units_idx
var String units_name
var String example_name
var int example_max_num

var int curr_idx

var int failure_bound
set failure_bound = 0
var String exch

var int failure_memo
set failure_memo = 0

##############################################
# set up corresponding data set              #
##############################################
if same "$set_train_test" "TRAINING" ; then
    set example_max_num =  `calc $example_id < ${currentNet.trainingExampleSet.numEx
amples}`
    set exch = "training"
elif same "$set_train_test" "TESTING" ; then
    set example_max_num =  `calc $example_id < ${currentNet.testingExampleSet.numExa
mples}`
    set exch = "testing"
else
    set example_max_num =  `calc $example_id < ${currentNet.validationExampleSet.num
Examples}`
    set exch = "validation"
fi

##############################################
# For valid example, get its sample's name   #
##############################################
if  same "$example_max_num" "1" ; then
    set example_name = ${currentNet.${exch}ExampleSet.example[example_id].name}

##############################################
# Get input vector                           #
##############################################
set units_idx = 0
set curr_idx = ${currentNet.${exch}ExampleSet.example[example_id].numInputs}
while same $(calc $units_idx < $curr_idx) "1"; do
    set input[units_idx]  = ${currentNet.${exch}ExampleSet.example[example_id].input
Vector[0][units_idx]}
    set units_idx = $(calc $units_idx + 1)
done
    set example_id = $(calc $example_id + 1)

set units_idx = 0
set output_7 = $dash_char

##############################################
# Get target,actual output and error vectors#
```



```
##############################################
    print example_name "  Cho      Chr      NAA      Ala      LA       Lip    Succ_
Fail"
fi
echo          "======== ======== ======== ======== ======== ======== ======== =====
====="
while same $(calc $units_idx < $idx) "1"; do
    set output[units_idx] =  ${${group}.unit[units_idx].output}
    set target[units_idx] =  ${${group}.unit[units_idx].target}

    set error[units_idx] = $(calc "abs(  ${target[units_idx]} - ${output[units_idx]}
)")

#    set units_name = ${${group}.unit[units_idx].name}

#####################################################
# Get failure samples, which error > examine-bound  #
#####################################################
   set failure_bound =  `calc ${error[units_idx]} "\>" $examine_bound`

   if same "$failure_bound" "1" ; then
       set output_7 = "Fail"
       set failure_memo = 1
   fi

   set units_idx = $(calc $units_idx + 1)
done

   if same "$failure_memo" "1" ; then
       set failure_exam[failure_i] = $example_name

       set fail_out0[failure_i] = ${output[0]}
       set fail_out1[failure_i] = ${output[1]}
       set fail_out2[failure_i] = ${output[2]}
       set fail_out3[failure_i] = ${output[3]}
       set fail_out4[failure_i] = ${output[4]}
       set fail_out5[failure_i] = ${output[5]}

       set failure_i = $(calc $failure_i + 1)
       set failure_bound = 0.0
   fi

#####################################################
# Save trace information for one example            #
#####################################################
#    set units_idx = $(calc $units_idx + 1)
# done
    print "Input  "  input[0] "%9.6f" input[1] "%8.6f" input[2] "%8.6f" input[3] "%8
.6f" input[4] "%8.6f" input[5] "%8.6f" input_7 "%7s"
    print "Target "  target[0] "%9.6f" target[1] "%8.6f" target[2] "%8.6f" target[3]
 "%8.6f" target[4] "%8.6f" target[5] "%8.6f" target_7 "%7s"
    print "Output "  output[0] "%9.6f" output[1] "%8.6f" output[2] "%8.6f" output[3]
 "%8.6f" output[4] "%8.6f" output[5] "%8.6f" output_7 "%7s"
    print "Error  "  error[0] "%9.6f" error[1] "%8.6f" error[2] "%8.6f" error[3] "%8
.6f" error[4] "%8.6f" error[5] "%8.6f" error_7 "%7.2f"
   echo "          "
```

Appendix C

Source Codes for Statistical Analysis

C.1 Tumour_stat.c

```
/*************************************************************/
/*  Name  : Tumour_stat.c                                    */
/*  Date  : July30 26, 1994                                  */
/*                                                           */
/*  Function:                                                */
/*   1. Add:  Statistic summ_XX report for tumour diagnosis. */
/*            Calculate the success rate  and failure rate   */
/*            from the total testing samples. If there is    */
/*            any sample failed, give the actual output      */
/*            and compare with the target output, give the   */
/*            actual brain tumour group diagnosis.           */
/*                                                           */
/*        In : the testing result file name in log directory*/
/*        Out: the statistics summary report, refer to       */
/*             summ_XX or REPORT in current directory.       */
/*             the detailed output for each sample,refer to  */
/*             log/tr_XX.ex log/ts_XX.ex                     */
/*                                                           */
/*   2. Delete: Delete the statistic data from summ_XX       */
/*                 and history summary report files.         */
/*                                                           */
/*        In : the being delete file name in log directory   */
/*        Out: the update statistics summary report,refer to*/
/*             summ_XX or REPORT in current directory.       */
/*             the detailed output for each sample,refer to  */
/*             log/tr_XX.p.YY log/ts_XX.p.YY                 */
/*                                                           */
/*   3. Quit:   Exit statistic routine.                      */
/*                                                           */
/*                                                           */
/*  Run : 1. Enter summary report file name by prompt.       */
/*                                                           */
/*        2. Choose add, delete and quit functions:          */
/*                                                           */
/*           Select A option to add the analysis result.     */
/*             enter the data set tr_XX or ts_XX for adding. */
/*             If you want to start new summary report,      */
/*                type Y, otherwise enter N;                 */
/*                                                           */
/*           Select D option to delte some analysis records. */
/*             enter the delete data set name for deleting.  */
/*                                                           */
/*           Select Q for exit when you finished.            */
/*                                                           */
/*                                                           */
/*  Used by: Evaluate the neural network performance.        */
/*  Note: Summary report types refers to Tumour_set_test.c   */
/*  Update : The source and destination file path            */
/*                                                           */
/*************************************************************/
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

#define MAX_ELEM   8
#define FAIL_NUM   600

FILE *fp_org, *fp_report, *fp_summary;

char org_path[60];          /* the traced nets reports path       */
char des_path[60];          /* the SUMMARY  and REPORT path       */
char summ_path[60];         /* the summ_XX  path                  */
char org_file[80];          /* traced nets reports file name      */
char des_file[80];          /* history records for summary report */
char summary_file[80];      /* summary report file path and name  */
char set_name[10];          /* training or testing data set name  */
char temp_fail_actual[4];   /* temporary for actual tumour group  */
char in_file[20];
int as1_cnt;
int as2_cnt;
int gbm_cnt;
int men_cnt;
int met_cnt;
```



```
int nml_cnt;
int round_out;
float float_out;

typedef struct {
   char s_name[4];          /* as1, as2, gbm, men, met, nml       */
   int  s_total;            /* total testing samples              */
   int  s_succ;             /* total success samples              */
   int  s_fail;             /* total failure samples              */
   int  s_fail_case_idx;    /* current failure sample pointer     */
   char s_examine[FAIL_NUM][5];  /* abs(TargetOutput-ActualOutput)*/
   char s_fail_case[FAIL_NUM][30]; /* failure sample set and name */
   int  s_fail_out[FAIL_NUM][6];   /* failure sample actual output*/
   char s_actual[FAIL_NUM][4];     /* failure sample actual group */
}report;

report as1;
report as2;
report gbm;
report men;
report met;
report nml;

/*****************************************************************/
/* clear all records of summary report and history summary records */
/*****************************************************************/
void Clear_records(report rept)
{
   int i,j;

   rept.s_total = 0;
   rept.s_succ = 0;
   rept.s_fail = 0;
   rept.s_fail_case_idx = 0;
   for (i=0; i<FAIL_NUM; i++)
   {
       strcpy(rept.s_fail_case[i], "");
       strcpy(rept.s_actual[i], "");
       strcpy(rept.s_examine[i], "");
       for (j=0; j<6; j++)
          rept.s_fail_out[i][j] = 0;
   }
}

/*****************************************************************/
/* Open the traced nets log data file for statistic analysis      */
/*****************************************************************/
void Open_org_file()
{
   if ((fp_org = fopen(org_file, "r")) == NULL)
   {
      printf("\n can not open file %s\n\n", org_file);
      exit(0);
   }
}

/*****************************************************************/
/* Clear history summary records report                          */
/*****************************************************************/
void Open_report_file()
{
   if ((fp_report = fopen(des_file, "w")) == NULL)
   {
      printf("\n can not open file %s\n\n", des_file);
      exit(0);
   }
}


/*****************************************************************/
/* Open summary report                                           */
/*****************************************************************/
void Open_summary_file()
```

```
{
   if ((fp_summary = fopen(summary_file, "w")) == NULL)
   {
      printf("\n can not open SUMMARY file\n\n");
      exit(0);
   }
}

/*****************************************************************/
/* Write one tumour record to summary and its history record files */
/*****************************************************************/
void Write_a_record(report rept)
{
   int i, j;
   char temp_slash[2];
   int  temp_fail_out;

   temp_slash[0] = '/';
   temp_slash[1] = '\0';

   fprintf(fp_report, "\n");
   fprintf(fp_summary, "\n");
   fprintf(fp_report, "%3s  %3d %s %2d %4d %4d\n", rept.s_name, rept.s_succ, temp_sl
ash, rept.s_total, rept.s_fail, rept.s_fail_case_idx);
   fprintf(fp_summary, "%3s  %3d %s %2d %4d \n", rept.s_name, rept.s_succ, temp_slas
h, rept.s_total, rept.s_fail);
   for (i=0; i<rept.s_fail_case_idx; i++)
   {
      fprintf(fp_report,  " %35s", rept.s_examine[i]);
      fprintf(fp_report,  " %37s", rept.s_fail_case[i]);
/*
      fprintf(fp_summary, " %28s", rept.s_examine[i]);
      fprintf(fp_summary, " %30s", rept.s_fail_case[i]);
*/
      fprintf(fp_summary, " %23s", rept.s_examine[i]);
      fprintf(fp_summary, " %28s", rept.s_fail_case[i]);
      fprintf(fp_report,  "%s", " ");
      fprintf(fp_summary, "%s", " ");

      for (j=0; j<6; j++)
      {
          temp_fail_out = rept.s_fail_out[i][j];
          if (temp_fail_out == 100)
          {
             fprintf(fp_report, " %.3d", temp_fail_out);
             fprintf(fp_summary, " %.3d", temp_fail_out);
/*
             fprintf(fp_report, " %.3d", rept.s_fail_out[i][j]);
             fprintf(fp_summary, " %.3d", rept.s_fail_out[i][j]);
*/
          }
          else {
             fprintf(fp_report, "  %.2d", temp_fail_out);
             fprintf(fp_summary, "  %.2d", temp_fail_out);
          }
      }
      fprintf(fp_report, "%8s\n", rept.s_actual[i]);
      fprintf(fp_summary, " %4s\n", rept.s_actual[i]);
   }
}


/*****************************************************************/
/* Write all tumour records to summary and history record files    */
/*****************************************************************/
void Write_records()
{
   char title[80];

   fprintf(fp_report, "\n");
   fprintf(fp_summary, "\n");
   strcpy(title, "Name Succ Tot  Fail EXBD        Fail_Sample                Failure_
Output%      Diag");
```



```
   fprintf(fp_summary, "%s", title);

   strcpy(title, "==== ==== ==== ==== ==== ========================== ===========
=========== =====");
   fprintf(fp_summary, "\n");
   fprintf(fp_summary, "%s", title);

   strcpy(title, "Name Succ  Tot  Fail F_examine F_idx        Fail_Sample
 Failure_Output%      Actual_Group");
   fprintf(fp_report, "%s", title);
   fprintf(fp_report, "\n");
   strcpy(title, "==== ===== ==== ==== ========= ===== ======================  ===
==================== ============");
   fprintf(fp_report, "%s", title);

   Write_a_record(as1);
   Write_a_record(as2);
   Write_a_record(gbm);
   Write_a_record(men);
   Write_a_record(met);
   Write_a_record(nml);
   fprintf(fp_report, "\n");
   fprintf(fp_summary, "\n");
}

/*****************************************************************/
/* Read one tumour record from history summary records report      */
/* Only for continue doing statistic analysis for summary report   */
/*****************************************************************/
report Read_a_record(report rept)
{
   int i, j;
   char temp_total[4];
   char temp_succ[4];
   char temp_fail[4];
   char temp_examine[5];
   char temp_fail_idx[4];
   char temp_fail_out[5];
   char temp_slash[2];

   strcpy(temp_total, "");
   strcpy(temp_succ, "");
   strcpy(temp_fail, "");
   strcpy(temp_examine, "");
   strcpy(temp_fail_idx, "");
   strcpy(temp_fail_out, "");

   fscanf(fp_report, "%6s %s %s %s %s %s\n", rept.s_name, temp_succ, temp_slash, tem
p_total, temp_fail, temp_fail_idx);
   rept.s_total = atoi(temp_total);
   rept.s_succ = atoi(temp_succ);
   rept.s_fail = atoi(temp_fail);
   rept.s_fail_case_idx = atoi(temp_fail_idx);
   for (i=0; i<rept.s_fail_case_idx; i++)
   {
      fscanf(fp_report, "%s", rept.s_examine[i]);
      fscanf(fp_report, "%s", rept.s_fail_case[i]);
      for (j=0; j<6; j++)
      {
          fscanf(fp_report, "%s", temp_fail_out);
          rept.s_fail_out[i][j] = atoi(temp_fail_out);
      }
      fscanf(fp_report, "%s\n", rept.s_actual[i]);
   }
   return rept;
}


/*****************************************************************/
/* Read all tumour records from history summary records report     */
/* Only for continue doing statistic analysis for summary report   */
/*****************************************************************/
void Read_records()
```

```
{
   int i, j;
   char title[80];

   if ((fp_report = fopen(des_file, "r")) == NULL)
   {
#ifdef DEBUG
      printf("\n can not open file");
#endif
      exit(0);
   }

   for (j=0; j<9; j++)
      fscanf(fp_report, "%s", title);
   for (j=0; j<9; j++)
      fscanf(fp_report, "%s", title);

   as1 = Read_a_record(as1);
   as2 = Read_a_record(as2);
   gbm = Read_a_record(gbm);
   men = Read_a_record(men);
   met = Read_a_record(met);
   nml = Read_a_record(nml);
   fclose(fp_report);
}

/*********************************************************************/
/* Comparing the actual output with the target output to classify  */
/* the actual tumour group                                         */
/*********************************************************************/
void Get_fail_actual(int pos)
{
   strcpy(temp_fail_actual, "-");
   switch (pos)
   {
      case 0: strcpy(temp_fail_actual, "as1");
              break;
      case 1: strcpy(temp_fail_actual, "as2");
              break;
      case 2: strcpy(temp_fail_actual, "gbm");
              break;
      case 3: strcpy(temp_fail_actual, "met");
              break;
      case 4: strcpy(temp_fail_actual, "men");
              break;
      case 5: strcpy(temp_fail_actual, "nml");
              break;
      default: break;
   }
}


void Get_round_output()
{
    round_out = (int) float_out;
    if ((float_out - round_out) >= 0.5)
       round_out = round_out + 1;
}

/*********************************************************************/
/* Analyze the traced nets data file to make statistical summary   */
/* The total diagnosis samples, the success, the failure total     */
/* diagnosis. For the failure diagnosis, give its sample name and  */
/* its actual diagnosis tumour group.                              */
/*********************************************************************/

void Add_records()
{
   int i, j;

   char temp[10][20];
   char temp_head[30];
   char temp_fail[30];
```



```
   char temp_examine[5];
   char temp_examine_prompt[40];

   char temp_fail_out_i[4];
   float temp_fail_out;
   int temp_fail_max;
   int temp_fail_pos;

   int temp_sum_out;
   int temp_one_out;
   float temp_float_out;
   int round_out;

   char yes_no[4];

   int first_failure = 1;
   temp_fail_out= 0;
   temp_fail_max = 0;
   temp_fail_pos = 8;
   temp_sum_out = 0;
   temp_one_out = 0;
   strcpy(yes_no, "Y");

   strcpy(temp_head, "");
   for (j=0; j<10; j++)
       strcpy(temp[j], "");

   strcpy(org_file, org_path);
   strcat(org_file, in_file);

   if (strcmp(in_file, "") != 0)
      Open_org_file();
   else
      exit(0);

   strcpy(des_file, des_path);
   strcpy(des_file, "REPORT");
/*
   strcpy(summary_file, des_path);
   strcpy(summary_file, "SUMMARY");
*/

   /* if you want to continue the statistic summary analysis or not */
   printf("\n Do you want to RESET the prevous records(Y/N)");
   scanf("%s", yes_no);
   if ((yes_no[0] == 'y' ) || (yes_no[0] == 'Y'))
   {
   /* for starting the new statistic analysis, clear the history summary records */
      Clear_records(as1);
      Clear_records(as2);
      Clear_records(gbm);
      Clear_records(men);
      Clear_records(met);
      Clear_records(nml);
   }
   else
   /* for continuing the statistic analysis, read the history summary records */
      Read_records();

   Open_report_file();
   Open_summary_file();

   strcpy(as1.s_name, "as1");
   strcpy(as2.s_name, "as2");
   strcpy(gbm.s_name, "gbm");
   strcpy(men.s_name, "men");
   strcpy(met.s_name, "met");
   strcpy(nml.s_name, "nml");

   if (strcmp(in_file, "") != 0)
   {
      fscanf(fp_org, "%s", temp_head); /*  set title */
#ifdef DEBUG
```

```
      printf("\n Set_name=%s  ", temp_head);
#endif
      fscanf(fp_org, "%s", temp_head); /*  set name */
      strcpy(set_name, temp_head);
      printf(" Test_data_set=%s\n", set_name);

      fscanf(fp_org, "%s", temp_head); /*  example name */
#ifdef DEBUG
      printf("\n  %s\n", temp_head);
#endif

      while (strcmp(temp_head, ""))
      {
          if (strcmp(temp_head, "Failure") == 0)      /* fall into Fail samples */
          {

             if (first_failure)
             {
                strcpy(temp_examine_prompt, "");
                strcpy(temp_examine, "");
                fscanf(fp_org, "%s", temp_examine_prompt); /* title for fail_examine
 */
                fscanf(fp_org, "%s", temp_examine); /* read fail_examine bound */
                first_failure = 0;
             }

             for (j=0; j<10; j++)
                 strcpy(temp[j], "");

             for (j=0; j<8; j++)
                 fscanf(fp_org, "%s", temp[j]); /* read one fail sample record  */
                 /* temp[0]: set name,       i.e tr_berp_met.ex  */
                 /* temp[1]: example name,   i.e coom_as2 */
                 /* temp[2]: actual output[0],   i.e 0.02 */
                 /* temp[3]: actual output[1],   i.e 0.96 */
                 /* temp[4]: actual output[2],   i.e 0.01 */
                 /* temp[5]: actual output[3],   i.e 0.00 */
                 /* temp[6]: actual output[3],   i.e 0.00 */
                 /* temp[7]: actual output[3],   i.e 0.04 */

             if (strcmp(temp[0], "") == 0)      /* if the set name is empty, done */
                 strcpy(temp_head, "");
             else                               /* the set name exist, analyze fail
*/
             {
                 float_out = 0.0;
                 round_out = 0;
                 temp_sum_out = 0;
                 temp_one_out = 0;
                 strcpy(temp_fail_out_i, "");
                 for (j=0; j<6; j++)
                 {
                    if (temp[j+2][0] == '1')    /* case 1.00 */
                        temp_one_out = 100;
                    else {                      /* less than 1 */
                     temp_fail_out_i[0] = temp[j+2][2];
                     temp_fail_out_i[1] = temp[j+2][3];
                     temp_one_out = atoi(temp_fail_out_i); /* actual output * 100 */
                    }
                    temp_sum_out = temp_sum_out + temp_one_out;
                 }

                 temp_fail_max = 0;
                 temp_fail_pos = 8;             /* default is non_group */
                 strcpy(temp_fail_out_i, "");

                 strcpy(temp_fail, temp[0]);
                 strcat(temp_fail, "::");
                 strcat(temp_fail, temp[1]);    /* make fail id = set_name::example_
name */

                 if (strstr(temp[1], "_as1"))  /* as1 */
                 {
```



```
                    strcpy(as1.s_examine[as1.s_fail_case_idx], temp_examine);
                    strcpy(as1.s_fail_case[as1.s_fail_case_idx], temp_fail);
                    for (j=0; j<6; j++)
                    {
                      if (temp[j+2][0] == '1') /* case 1.00 */
                          temp_fail_out = 100;
                      else {                   /* less than 1.00 */
                       temp_fail_out_i[0] = temp[j+2][2];
                       temp_fail_out_i[1] = temp[j+2][3];
                       temp_fail_out = atof(temp_fail_out_i); /* actual output * 10
0 */
                      }

                       float_out = 100 * temp_fail_out/temp_sum_out;

                       round_out = (int) float_out;
                       if ((float_out - round_out) >= 0.5)
                       round_out = round_out + 1;

                       as1.s_fail_out[as1.s_fail_case_idx][j] = round_out;
                       if (temp_fail_out > temp_fail_max)
                       {
                          temp_fail_max = temp_fail_out;
                          temp_fail_pos = j;
                       }
                    }
                    Get_fail_actual(temp_fail_pos);  /* get actual tumour group */
                    strcpy(as1.s_actual[as1.s_fail_case_idx], temp_fail_actual);
                    as1.s_fail++;
                    as1.s_fail_case_idx++;
                 }
                 if (strstr(temp[1], "_as2")) /* as2 */
                 {
                    strcpy(as2.s_examine[as2.s_fail_case_idx], temp_examine);
                    strcpy(as2.s_fail_case[as2.s_fail_case_idx], temp_fail);
                    for (j=0; j<6; j++)
                    {
                      if (temp[j+2][0] == '1') /* case 1.00 */
                          temp_fail_out = 100;
                      else {                   /* less than 1.00 */
                       temp_fail_out_i[0] = temp[j+2][2];
                       temp_fail_out_i[1] = temp[j+2][3];
                       temp_fail_out = atof(temp_fail_out_i);
                      }

                       float_out = 100 * temp_fail_out/temp_sum_out;
                       round_out = (int) float_out;
                       if ((float_out - round_out) >= 0.5)
                       round_out = round_out + 1;

                       as2.s_fail_out[as2.s_fail_case_idx][j] = round_out;
                       if (temp_fail_out > temp_fail_max)
                       {
                          temp_fail_max = temp_fail_out;
                          temp_fail_pos = j;
                       }
                    }
                    Get_fail_actual(temp_fail_pos);
                    strcpy(as2.s_actual[as2.s_fail_case_idx], temp_fail_actual);
                    as2.s_fail++;
                    as2.s_fail_case_idx++;
                 }
                 if (strstr(temp[1], "_gbm")) /* gbm */
                 {
                    strcpy(gbm.s_examine[gbm.s_fail_case_idx], temp_examine);
                    strcpy(gbm.s_fail_case[gbm.s_fail_case_idx], temp_fail);
                    for (j=0; j<6; j++)
                    {
                      if (temp[j+2][0] == '1') /* case 1.00 */
                          temp_fail_out = 100;
                      else {                   /* less than 1.00 */
                       temp_fail_out_i[0] = temp[j+2][2];
                       temp_fail_out_i[1] = temp[j+2][3];
```

```
                temp_fail_out = atof(temp_fail_out_i);
               }

                float_out = 100 * temp_fail_out/temp_sum_out;
                round_out = (int) float_out;
                if ((float_out - round_out) >= 0.5)
                round_out = round_out + 1;
                gbm.s_fail_out[gbm.s_fail_case_idx][j] = round_out;
                if (temp_fail_out > temp_fail_max)
                {
                   temp_fail_max = temp_fail_out;
                   temp_fail_pos = j;
                }
            }
            Get_fail_actual(temp_fail_pos);
            strcpy(gbm.s_actual[gbm.s_fail_case_idx], temp_fail_actual);
            gbm.s_fail++;
            gbm.s_fail_case_idx++;
        }
        if (strstr(temp[1], "_men")) /* men */
        {
            strcpy(men.s_examine[men.s_fail_case_idx], temp_examine);
            strcpy(men.s_fail_case[men.s_fail_case_idx], temp_fail);
            for (j=0; j<6; j++)
            {
              if (temp[j+2][0] == '1') /* case 1.00 */
                  temp_fail_out = 100;
              else {                   /* less than 1.00 */
               temp_fail_out_i[0] = temp[j+2][2];
               temp_fail_out_i[1] = temp[j+2][3];
               temp_fail_out = atof(temp_fail_out_i);
              }

               float_out = 100 * temp_fail_out/temp_sum_out;
               round_out = (int) float_out;
               if ((float_out - round_out) >= 0.5)
               round_out = round_out + 1;

               men.s_fail_out[men.s_fail_case_idx][j] = round_out;
               if (temp_fail_out > temp_fail_max)
               {
                  temp_fail_max = temp_fail_out;
                  temp_fail_pos = j;
               }
            }
            Get_fail_actual(temp_fail_pos);
            strcpy(men.s_actual[men.s_fail_case_idx], temp_fail_actual);
            men.s_fail++;
            men.s_fail_case_idx++;
        }
        if (strstr(temp[1], "_met")) /* met */
        {
            strcpy(met.s_examine[met.s_fail_case_idx], temp_examine);
            strcpy(met.s_fail_case[met.s_fail_case_idx], temp_fail);
            for (j=0; j<6; j++)
            {
              if (temp[j+2][0] == '1') /* case 1.00 */
                  temp_fail_out = 100;
              else {                   /* less than 1.00 */
               temp_fail_out_i[0] = temp[j+2][2];
               temp_fail_out_i[1] = temp[j+2][3];
               temp_fail_out = atof(temp_fail_out_i);
              }

               float_out = 100 * temp_fail_out/temp_sum_out;
               round_out = (int) float_out;
               if ((float_out - round_out) >= 0.5)
               round_out = round_out + 1;

               met.s_fail_out[met.s_fail_case_idx][j] = round_out;
               if (temp_fail_out > temp_fail_max)
               {
                  temp_fail_max = temp_fail_out;
```



```
                  temp_fail_pos = j;
               }
            }
            Get_fail_actual(temp_fail_pos);
            strcpy(met.s_actual[met.s_fail_case_idx], temp_fail_actual);
            met.s_fail++;
            met.s_fail_case_idx++;
        }
        if (strstr(temp[1], "_nml")) /* nml */
        {
            strcpy(nml.s_examine[nml.s_fail_case_idx], temp_examine);
            strcpy(nml.s_fail_case[nml.s_fail_case_idx], temp_fail);
            for (j=0; j<6; j++)
            {
              if (temp[j+2][0] == '1') /* case 1.00 */
                  temp_fail_out = 100;
              else {                   /* less than 1.00 */
               temp_fail_out_i[0] = temp[j+2][2];
               temp_fail_out_i[1] = temp[j+2][3];
               temp_fail_out = atof(temp_fail_out_i);
              }

               float_out = 100 * temp_fail_out/temp_sum_out;
               round_out = (int) float_out;
               if ((float_out - round_out) >= 0.5)
               round_out = round_out + 1;

               nml.s_fail_out[nml.s_fail_case_idx][j] = round_out;
               if (temp_fail_out > temp_fail_max)
               {
                  temp_fail_max = temp_fail_out;
                  temp_fail_pos = j;
               }
            }
            Get_fail_actual(temp_fail_pos);
            strcpy(nml.s_actual[nml.s_fail_case_idx], temp_fail_actual);
            nml.s_fail++;
            nml.s_fail_case_idx++;
        }
      }
    }
    else /* normal reports */
    {
       if ((temp_head[0] > 0x60) && (temp_head[0] <= 'z'))
       {
           for (j=0; j<MAX_ELEM; j++)
               fscanf(fp_org, "%s", temp[j]); /* traced data */

           if (strstr(temp_head, "_as1")) /* as1 */
               as1.s_total++;
           if (strstr(temp_head, "_as2")) /* as2 */
               as2.s_total++;
           if (strstr(temp_head, "_gbm")) /* gbm */
               gbm.s_total++;
           if (strstr(temp_head, "_men")) /* men */
               men.s_total++;
           if (strstr(temp_head, "_met")) /* met */
               met.s_total++;
           if (strstr(temp_head, "_nml")) /* nml */
               nml.s_total++;
       }
       printf("\n");
       fscanf(fp_org, "%s", temp_head); /* second  */
#ifdef DEBUG
       printf(" %s", temp_head);
#endif
       }
    }
    /* set success total */
    as1.s_succ = as1.s_total - as1.s_fail;
    as2.s_succ = as2.s_total - as2.s_fail;
    gbm.s_succ = gbm.s_total - gbm.s_fail;
    men.s_succ = men.s_total - men.s_fail;
```

```
      met.s_succ = met.s_total - met.s_fail;
      nml.s_succ = nml.s_total - nml.s_fail;
      Write_records();
   }
   fclose(fp_org);
   fclose(fp_report);
   fclose(fp_summary);
}

report Check_set(report rept, int set_cnt)
{
   int i, j;
   int new_fail_idx;

   i = 0;
   new_fail_idx = 0;
   while (i < rept.s_fail_case_idx)
   {
      if (strstr(rept.s_fail_case[i], in_file) == 0) /* not match */
      {
         strcpy(rept.s_fail_case[new_fail_idx], rept.s_fail_case[i]);
         strcpy(rept.s_actual[new_fail_idx], rept.s_actual[i]);
         for (j=0; j<6; j++)
            rept.s_fail_out[new_fail_idx][j] = rept.s_fail_out[new_fail_idx][j];
         new_fail_idx++;
      }
      i++;
   }
   rept.s_fail_case_idx = new_fail_idx;
   rept.s_fail = new_fail_idx;
   rept.s_total = rept.s_total - set_cnt;
   rept.s_succ = rept.s_total - rept.s_fail;
   return rept;
}

void Get_count()
{
    char temp_id[20];
    char temp_str[8][20];
    int i, exit_flag = 1;

    fscanf(fp_org, "%s", temp_str);
    fscanf(fp_org, "%s", temp_id);
    while (exit_flag)
    {
       for (i=0; i<8; i++)
       {
          strcpy(temp_str[i], "");
          fscanf(fp_org, "%s", temp_str[i]);
       }
       strcpy(temp_id, temp_str[0]);
       if ((strcmp(temp_id, "") == 0) || (strcmp(temp_id, "Failure") == 0))
          exit_flag = 0;
       else {
          if (strstr(temp_id, "_as1"))
             as1_cnt++;
          if (strstr(temp_id, "_as2"))
             as2_cnt++;
          if (strstr(temp_id, "_gbm"))
             gbm_cnt++;
          if (strstr(temp_id, "_men"))
             men_cnt++;
          if (strstr(temp_id, "_met"))
             met_cnt++;
          if (strstr(temp_id, "_nml"))
             nml_cnt++;
       }
    }
}

void Delete_records()
{
```



```
   as1_cnt = 0;
   as2_cnt = 0;
   gbm_cnt = 0;
   men_cnt = 0;
   met_cnt = 0;
   nml_cnt = 0;
   if (strcmp(in_file, ""))
   {
      strcpy(org_file, org_path);
      strcat(org_file, in_file);
      Open_org_file();
      Get_count();
      fclose(org_file);

      strcpy(des_file, des_path);
      strcpy(des_file, "REPORT");
/*
      strcpy(summary_file, des_path);
      strcpy(summary_file, "SUMMARY");
*/
      Read_records();

      Open_report_file();
      Open_summary_file();

      if (strcmp(in_file, "") != 0)
      {
         as1 =  Check_set(as1, as1_cnt);
         as2 =  Check_set(as2, as2_cnt);
         gbm =  Check_set(gbm, gbm_cnt);
         men =  Check_set(men, men_cnt);
         met =  Check_set(met, met_cnt);
         nml =  Check_set(nml, nml_cnt);
      }
      Write_records();
   }
   fclose(fp_report);
   fclose(fp_summary);
}


main()
{
   int i,j;
   char file_option[4];
   int exit_flag;
   char summ_name[20];


   strcpy(in_file, "");
   strcpy(file_option, "");

   strcpy(org_path, "");
   strcpy(org_file, "");
   strcpy(des_path, "");
   strcpy(des_file, "");
   strcpy(summ_path, "");
   strcpy(summary_file, "");

   /* the path for traced nets data files for statistic analysis */
   strcpy(org_path, "/home/apollo/jane/project/my_xerion/bp/tumour/log/");
   strcpy(des_path, "/home/apollo/jane/project/my_xerion/bp/tumour/");
   strcpy(summ_path, "/home/apollo/jane/project/my_xerion/bp/tumour/summ30/");

   printf("\n Enter the summary file name ==>");
   scanf("%s", summ_name);
   strcpy(summary_file, summ_path);
   strcat(summary_file, summ_name);

   do {
      exit_flag = 1;
      /* the file operation selection */
```

```
      printf("\n\n");
      printf("\n Enter the file option :  ");
      printf("\n       A. Adding the statistics data. ");
      printf("\n       D. Delete the statistics data. \n");
      printf("\n       Q. Quit.  \n\n");
      printf("\n Your choice ==> ");
      scanf("%s", file_option);


      if (!strcmp(file_option, "Q"))
         exit_flag = 0;
      else {
         if (!strcmp(file_option, "A"))
      /* the file name of traced nets data files for statistic analysis */
         {
            printf("\n Enter the test result file ==> ");
            scanf("%s", in_file);
            Add_records(in_file);
         }
         else {
            if (!strcmp(file_option, "D"))
         /* the data set statistics items will be deleted from the summary file */
            {
               printf("\n Enter the being deleted data file name ==> ");
               scanf("%s", in_file);
               Delete_records(in_file);
            }
         }
      }
   } while (exit_flag);
   printf("\n\n");
}
```

Appendix D

Data Files for Brain Tumour Diagnosis

D.1 Data Files

D.1.1 Clean Data File

D.1.2 Noisy Data File Samples

1. LSTpeak-Group patients Peak List data.
2. LSTarea-Group patients Area List data.
3. AVEpeak-Group patients Averaged Peak data.
4. AVEarea-Group patients Averaged Area data.
5. .peak-Sample of Peak data for one patient.
6. .area-Sample of Area data for one patient.

```
/*  baxw_as1  */
2.42    0.99    1.89    0       0.44    0
1       0       0       0       0       0
/*  boim_as1  */
1.98    1.13    2.01    0       0.65    0
1       0       0       0       0       0
/*  boug_as1  */
2.63    0.87    3.22    0       0.71    0
1       0       0       0       0       0
/*  brin_as1  */
2.71    1.22    3.11    0       0.58    0
1       0       0       0       0       0
/*  clel_as1  */
2.62    1.32    3.5     0       0.75    0
1       0       0       0       0       0
/*  gohm_as1  */
2.59    1.2     2.29    0       0.21    0
1       0       0       0       0       0
/*  hasy_as1  */
2.37    1.19    2.6     0       0.43    0
1       0       0       0       0       0
/*  kamj_as1  */
2.2     1.21    2.44    0       0.54    0
1       0       0       0       0       0
/*  labm_as1  */
2.64    0.97    1.77    0       0.47    0
1       0       0       0       0       0
/*  lang_as1  */
2.57    1.33    2.01    0       0.5     0
1       0       0       0       0       0
/*  leca_as1  */
2.29    1.07    1.84    0       0.57    0
1       0       0       0       0       0
/*  marj_as1  */
2.92    1.1     1.23    0       0.09    0
1       0       0       0       0       0
/*  ravr_as1  */
2.39    1       2.08    0       0.63    0
1       0       0       0       0       0
/*  teda_as1  */
2.5     1.29    1.34    0       0.72    0
1       0       0       0       0       0
/*  adac_as1  */
2.72    1.05    2.03    0       0.55    0
1       0       0       0       0       0
/*  pelj_as1  */
2.88    1.26    2.44    0       0.6     0
1       0       0       0       0       0
/*  robr_as1  */
2.38    1.28    1.67    0       0.27    0
1       0       0       0       0       0
/*  savy_as1  */
2.84    1.04    1.22    0       0.52    0
1       0       0       0       0       0
/*  tres_as1  */
2.56    1.18    1.47    0       0.7     0
1       0       0       0       0       0
/*  yuek_as1  */
2.69    1.22    3.3     0       0.82    0
1       0       0       0       0       0
/*  anaa_as2  */
2.76    0.94    1.1     0       1.44    0
0       1       0       0       0       0
/*  bisg_as2  */
2.93    0.73    0.8     0       1.65    0
0       1       0       0       0       0
/*  bouh_as2  */
2.49    0.87    1.2     0       1.71    0
0       1       0       0       0       0
/*  bour_as2  */
2.59    0.92    1.1     0       1.58    0
0       1       0       0       0       0
/*  casg_as2  */
```



```
2.4     0.92    1.5     0       1.75    0
0       1       0       0       0       0
/*  chia_as2  */
2.59    0.99    0.98    0       1.21    0.31
0       1       0       0       0       0
/*  coom_as2  */
2.77    0.89    1.6     0       1.43    0
0       1       0       0       0       0
/*  croa_as2  */
2.62    0.75    1.4     0       1.54    0
0       1       0       0       0       0
/*  dasa_as2  */
2.47    0.8     1.17    0       1.47    0
0       1       0       0       0       0
/*  desm_as2  */
2.54    0.89    1       0       1.5     0.21
0       1       0       0       0       0
/*  dorl_as2  */
2.49    0.92    1.34    0       1.57    0
0       1       0       0       0       0
/*  golz_as2  */
2.26    0.73    1.23    0       1.09    0
0       1       0       0       0       0
/*  lavf_as2  */
3.2     0.6     0.88    0       1.63    0
0       1       0       0       0       0
/*  lits_as2  */
2.7     1       1.3     0       1.72    0.22
0       1       0       0       0       0
/*  marj_as2  */
2.84    0.55    1       0       1.55    0
0       1       0       0       0       0
/*  marm_as2  */
2.6     0.86    1.4     0       1.6     0
0       1       0       0       0       0
/*  prom_as2  */
2.44    0.98    1       0       1.27    0.3
0       1       0       0       0       0
/*  ratn_as2  */
2.83    0.99    1.22    0       1.52    0
0       1       0       0       0       0
/*  robc_as2  */
2.59    1       1.35    0       1.7     0
0       1       0       0       0       0
/*  rosm_as2  */
3.41    0.82    1.12    0       1.82    0.4
0       1       0       0       0       0
/*  teda_as2  */
2.9     0.93    0.84    0       0.8     0.22
0       1       0       0       0       0
/*  vilj_as2  */
2.83    0.92    0.79    0       1.47    0
0       1       0       0       0       0
/*  auga_gbm  */
2.46    0.44    0.6     0       2.44    0.23
0       0       1       0       0       0
/*  bedh_gbm  */
2.73    0.43    0.8     0       2.65    0.63
0       0       1       0       0       0
/*  catg_gbm  */
2.39    0.47    0.62    0       2.71    0.23
0       0       1       0       0       0
/*  cols_gbm  */
2.19    0.52    0.21    0       2.58    0.45
0       0       1       0       0       0
/*  cour_gbm  */
2.1     0.52    0.65    0       2.75    0
0       0       1       0       0       0
/*  croa_gbm  */
2.19    0.39    0.38    0       2.21    0.31
0       0       1       0       0       0
/*  forp_gbm  */
2.37    0.27    0.56    0       2.23    0.37
```

```
0       0       1       0       0       0
/*  jels_gbm  */
2.62    0.19    0.9     0       2.54    0.28
0       0       1       0       0       0
/*  kisb_gbm  */
2.27    0.4     0.67    0       2.47    0.39
0       0       1       0       0       0
/*  knia_gbm  */
2.34    0.39    0.7     0       2.5     0.64
0       0       1       0       0       0
/*  krem_gbm  */
2.29    0.42    0.48    0       2.57    0
0       0       1       0       0       0
/*  lonr_gbm  */
2       0.73    0.38    0       2.09    0.73
0       0       1       0       0       0
/*  mord_gbm  */
2       0.18    0.8     0       2.43    0.61
0       0       1       0       0       0
/*  pauw_gbm  */
2.2     0.22    0.77    0       2.72    0.43
0       0       1       0       0       0
/*  pers_gbm  */
2.64    0.23    0.72    0       2.55    0.66
0       0       1       0       0       0
/*  poig_gbm  */
2.6     0.36    0.64    0       2.6     0.44
0       0       1       0       0       0
/*  rosg_gbm  */
2.1     0.47    0.83    0       2.27    0.37
0       0       1       0       0       0
/*  sanj_gbm  */
2.3     0.99    0.45    0       2.52    0.2
0       0       1       0       0       0
/*  siga_gbm  */
2.59    0.57    0.84    0       2.7     0.43
0       0       1       0       0       0
/*  teda_gbm  */
2.2     0.82    0.62    0       2.82    0.38
0       0       1       0       0       0
/*  turg_gbm  */
1.9     0.36    0.18    0       2.8     0.18
0       0       1       0       0       0
/*  valj_gbm  */
2.53    0.25    0.38    0       2.47    0.57
0       0       1       0       0       0
/*  vilj_gbm  */
2.78    0.22    0.49    0       2.05    0.37
0       0       1       0       0       0
/*  wons_gbm  */
2.89    0.09    0.3     0       1.98    0.47
0       0       1       0       0       0
/*  alll_men  */
1.62    0.64    0       0.87    0       0
0       0       0       0       1       0
/*  covd_men  */
1.21    0.55    0.2     0.54    0       0
0       0       0       0       1       0
/*  cred_men  */
1.56    0.87    0.3     0.64    0       0
0       0       0       0       1       0
/*  four_men  */
1.63    0.93    0       0.86    0       0
0       0       0       0       1       0
/*  gian_men  */
1.25    0.59    0.21    0.73    0       0
0       0       0       0       1       0
/*  hard_men  */
1.32    0.75    0.23    0.39    0       0
0       0       0       0       1       0
/*  hebj_men  */
1.78    0.75    0.44    0.64    0       0
0       0       0       0       1       0
```



```
/*  pfey_men  */
1.79    0.84    0.17    0.75    0       0
0       0       0       0       1       0
/*  quii_men  */
1.43    0.85    0       0.83    0.69    0.32
0       0       0       0       1       0
/*  berp_met  */
1.2     0.3     0.3     0       3.9     0.28
0       0       0       1       0       0
/*  bram_met  */
0.76    0.5     0.4     0       3.93    0.39
0       0       0       1       0       0
/*  brol_met  */
0.87    0.48    0.12    0       3.83    0.64
0       0       0       1       0       0
/*  bror_met  */
0.78    0.23    0.07    0       2.32    0.18
0       0       0       1       0       0
/*  gagg_met  */
1.1     0.37    0.16    0       3.36    0.73
0       0       0       1       0       0
/*  gure_met  */
0.87    0.21    0.2     0       3.89    0.61
0       0       0       1       0       0
/*  mymm_met  */
0.73    0.1     0.32    0       2       0.43
0       0       0       1       0       0
/*  ramr_met  */
1.32    0.08    0.07    0       2.98    0.66
0       0       0       1       0       0
/*  ridk_met  */
1.29    0.05    0.18    0       3.42    0.44
0       0       0       1       0       0
/*  schi_met  */
1.55    0.5     0       0       3.2     0.37
0       0       0       1       0       0
/*  soul_met  */
1.47    0.04    0.32    0       3.54    0.2
0       0       0       1       0       0
/*  valu_met  */
0.74    0.31    0.31    0       3.21    0.43
0       0       0       1       0       0
/*  berp_met  */
0.82    0.27    0.28    0       2.3     0.38
0       0       0       1       0       0
/*  hunr_met  */
0.92    0.31    0       0       3.4     0.18
0       0       0       1       0       0
/*  leto_met  */
1.18    0.19    0.08    0       2.89    0.59
0       0       0       1       0       0
/*  macj_met  */
0.7     0.24    0.09    0       2.93    0.27
0       0       0       1       0       0
/*  brij_nml  */
1.12    1       3.76    0       0       0
0       0       0       0       0       1
/*  codd_nml  */
1.1     0.9     3.55    0       0       0
0       0       0       0       0       1
/*  cora_nml  */
0.99    0.92    3.52    0       0       0
0       0       0       0       0       1
/*  desn_nml  */
1.13    1.1     3.8     0       0       0
0       0       0       0       0       1
/*  fiej_nml  */
1       1       3.7     0       0       0
0       0       0       0       0       1
/*  fuli_nml  */
1.1     1.05    3.67    0       0       0
0       0       0       0       0       1
/*  lari_nml  */
```

```
1.11    0.98    3.87    0       0       0
0       0       0       0       0       1
/*  lowl_nml  */
1.18    1.2     3.67    0       0       0
0       0       0       0       0       1
/*  manf_nml  */
1.21    1       3.59    0       0       0
0       0       0       0       0       1
/*  medv_nml  */
1.16    1       3.72    0       0       0
0       0       0       0       0       1
/*  olej_nml  */
1.17    1       3.56    0       0       0
0       0       0       0       0       1
/*  perb_nml  */
1.2     0.96    3.55    0       0       0
0       0       0       0       0       1
/*  treb_nml  */
0.95    0.93    3.69    0       0       0
0       0       0       0       0       1
/*  zwij_nml  */
1.05    1       3.53    0       0       0
0       0       0       0       0       1
```

```
/* bis0268_as2 */
1.66    0.99    0.85    0.62    0.76    0.30
0       1       0       0       0       0

/* bis0269_as2 */
1.51    0.50    0.70    0.58    0.75    0.18
0       1       0       0       0       0

/* bis0308_nml */
1.09    0.99    2.80    0.02    0.02    0.13
0       0       0       0       0       1

/* bis0331_as2 */
2.12    0.69    0.74    0.26    0.50    0.13
0       1       0       0       0       0

/* bis0335_as2 */
1.18    0.30    0.74    0.55    0.74    0.13
0       1       0       0       0       0

/* bis0340_nml */
1.20    1.15    3.08    0.04    0.06    0.21
0       0       0       0       0       1

/* bis0363_as2 */
2.44    1.07    0.81    0.15    0.41    0.19
0       1       0       0       0       0

/* bis0364_as2 */
3.14    0.97    0.73    0.50    0.52    0.19
0       1       0       0       0       0

/* bis0365_as2 */
2.71    0.61    0.66    0.31    0.47    0.15
0       1       0       0       0       0

/* bis0366_as2 */
1.88    0.30    0.94    0.50    0.80    0.07
0       1       0       0       0       0

/* bis0372_nml */
1.16    0.92    3.34    0.19    0.02    0.21
0       0       0       0       0       1

/* bis0404_nml */
0.99    0.99    3.68    0.11    0.08    0.21
0       0       0       0       0       1

/* bis0436_nml */
0.96    0.98    3.86    0.03    0.06    0.18
0       0       0       0       0       1

/* bis0468_nml */
1.19    0.97    3.32    0.09    0.24    0.26
0       0       0       0       0       1

/* bri0365_nml */
1.38    1.23    3.06    0.23    0.26    0.18
0       0       0       0       0       1

/* bri0397_nml */
1.31    1.11    2.59    0.02    0.15    0.21
0       0       0       0       0       1

/* bri0401_as1 */
2.94    1.31    1.27    0.02    0.58    0.06
1       0       0       0       0       0

/* bri0402_as1 */
2.33    1.19    1.74    0.02    0.45    0.18
1       0       0       0       0       0
```



```
/* bri0429_nml */
1.16    0.96    3.16    0.07    0.20    0.16
0       0       0       0       0       1

/* bri0432_as1 */
2.69    1.35    2.50    0.23    0.22    0.13
1       0       0       0       0       0

/* bri0433_as1 */
4.12    1.69    1.75    0.28    0.02    0.04
1       0       0       0       0       0

/* bri0434_as1 */
2.74    1.13    2.18    0.07    0.26    0.20
1       0       0       0       0       0

/* bri0461_nml */
0.68    0.85    3.07    0.02    0.34    0.17
0       0       0       0       0       1

/* bri0465_as1 */
2.66    1.50    2.06    0.53    0.02    0.15
1       0       0       0       0       0

/* bri0500_nml */
0.72    1.05    4.02    0.22    0.17    0.32
0       0       0       0       0       1

/* bri0565_nml */
0.93    0.80    3.54    0.42    0.02    0.12
0       0       0       0       0       1

/* cov0332_men */
1.77    0.74    0.84    0.84    0.72    0.06
0       0       0       0       1       0

/* cov0333_men */
0.81    0.52    0.49    0.64    0.50    0.09
0       0       0       0       1       0

/* cov0364_men */
1.62    0.79    0.49    1.09    0.96    0.13
0       0       0       0       1       0

/* cov0365_men */
0.96    0.55    0.50    0.15    0.17    0.10
0       0       0       0       1       0

/* cov0373_nml */
1.23    1.07    3.07    0.04    0.09    0.21
0       0       0       0       0       1

/* cov0396_men */
1.66    0.73    0.53    1.02    0.18    0.16
0       0       0       0       1       0

/* cov0397_men */
1.02    0.56    0.61    0.54    0.50    0.24
0       0       0       0       1       0

/* cov0405_nml */
1.31    1.29    2.85    0.20    0.02    0.18
0       0       0       0       0       1

/* cov0428_men */
1.79    0.54    0.65    1.16    1.06    0.17
0       0       0       0       1       0

/* cov0429_men */
1.25    0.69    0.55    0.94    0.78    0.02
0       0       0       0       1       0

/* cov0437_nml */
```

```
1.01      0.76      2.48      0.17      0.02      0.14
0         0         0         0         0         1

/* cov0469_nml */
0.90      0.87      2.17      0.12      0.09      0.12
0         0         0         0         0         1

/* for0372_nml */
0.91      0.85      2.46      0.16      0.02      0.18
0         0         0         0         0         1

/* for0404_nml */
0.82      0.81      2.64      0.18      0.02      0.18
0         0         0         0         0         1

/* for0405_nml */
1.03      0.97      2.54      0.21      0.13      0.24
0         0         0         0         0         1

/* for0436_nml */
0.99      1.09      3.31      0.48      0.02      0.22
0         0         0         0         0         1

/* for0437_nml */
1.00      1.28      3.33      0.15      0.22      0.18
0         0         0         0         0         1

/* for0493_gbm */
1.54      0.53      1.18      0.31      0.43      0.33
0         0         1         0         0         0

/* for0494_gbm */
1.64      0.18      0.58      0.23      1.51      0.34
0         0         1         0         0         0

/* for0495_gbm */
1.44      0.12      0.56      0.54      1.09      0.50
0         0         1         0         0         0

/* for0496_gbm */
1.74      0.29      0.59      0.75      1.64      0.73
0         0         1         0         0         0

/* for0497_gbm */
1.67      0.30      0.54      1.34      2.73      0.65
0         0         1         0         0         0

/* for0498_gbm */
1.30      0.78      0.66      0.24      2.40      0.37
0         0         1         0         0         0

/* for0526_gbm */
1.98      0.12      0.42      0.49      1.25      0.47
0         0         1         0         0         0

/* for0530_gbm */
2.21      0.55      0.79      0.93      0.93      0.36
0         0         1         0         0         0

/* for0558_gbm */
2.82      0.76      0.80      1.33      1.06      0.21
0         0         1         0         0         0

/* for0559_gbm */
3.51      1.28      1.47      1.21      1.62      0.29
0         0         1         0         0         0

/* ful0405_nml */
1.16      0.55      3.36      0.11      0.22      0.14
0         0         0         0         0         1

/* ful0437_nml */
1.08      0.54      4.00      0.12      0.02      0.21
```



```
0         0         0         0         0         1

/* ful0461_nml */
1.30      1.14      4.44      0.09      0.13      0.18
0         0         0         0         0         1

/* ful0469_nml */
0.80      0.75      4.36      0.04      0.05      0.25
0         0         0         0         0         1

/* ful0500_nml */
1.35      0.92      4.15      0.14      0.05      0.15
0         0         0         0         0         1

/* ful0501_nml */
0.96      0.80      4.42      0.13      0.24      0.12
0         0         0         0         0         1

/* ful0525_nml */
1.27      1.26      4.96      0.11      0.26      0.14
0         0         0         0         0         1

/* ful0533_nml */
1.30      0.92      4.34      0.18      0.02      0.16
0         0         0         0         0         1

/* ful0556_nml */
1.17      1.31      4.40      0.07      0.07      0.10
0         0         0         0         0         1

/* ful0557_nml */
1.20      1.33      4.78      0.09      0.17      0.15
0         0         0         0         0         1

/* ful0565_nml */
1.44      1.01      4.64      0.15      0.15      0.24
0         0         0         0         0         1

/* ful0588_nml */
1.23      1.45      4.47      0.02      0.13      0.10
0         0         0         0         0         1

/* mym0372_nml */
1.36      0.83      2.84      0.07      0.03      0.11
0         0         0         0         0         1

/* mym0405_nml */
1.03      0.77      3.04      0.19      0.01      0.06
0         0         0         0         0         1

/* mym0493_nml */
1.28      1.28      3.37      0.27      0.01      0.11
0         0         0         0         0         1

/* mym0498_met */
2.36      0.72      0.83      2.04      3.49      0.63
0         0         0         1         0         0

/* mym0499_met */
1.71      0.48      0.60      1.70      4.53      0.97
0         0         0         1         0         0

/* mym0532_met */
1.32      0.54      0.84      1.08      5.88      1.32
0         0         0         1         0         0

/* mym0557_nml */
1.11      1.13      3.13      0.05      0.47      0.14
0         0         0         0         0         1

/* mym0558_nml */
1.13      1.04      2.73      0.37      0.62      0.17
0         0         0         0         0         1
```

```
/* mym0563_met */
1.21    0.39    0.36    0.20    7.39    1.76
0       0       0       1       0       0

/* mym0564_met */
1.25    0.44    0.66    0.48    6.75    1.54
0       0       0       1       0       0

/* mym0590_nml */
1.13    0.96    2.99    0.23    0.08    0.06
0       0       0       0       0       1

/* pfe0271_men */
1.26    0.43    0.27    0.29    0.21    0.20
0       0       0       0       1       0

/* pfe0272_men */
1.03    0.47    0.14    0.07    0.03    0.11
0       0       0       0       1       0

/* pfe0273_men */
0.93    0.73    0.18    0.46    0.18    0.10
0       0       0       1       0       0

/* pfe0303_men */
1.14    0.55    0.46    0.19    0.02    0.11
0       0       0       0       1       0

/* pfe0304_men */
1.10    0.60    0.34    0.15    0.05    0.15
0       0       0       0       1       0

/* pfe0305_men */
1.08    0.50    0.20    0.17    0.18    0.10
0       0       0       0       1       0

/* pfe0306_men */
-0.13   0.02    0.09    0.02    0.06    0.16
0       0       0       0       1       0

/* pfe0338_men */
1.43    0.71    0.55    0.14    0.13    0.02
0       0       0       0       1       0

/* pfe0467_nml */
1.58    1.26    2.90    0.09    0.10    0.18
0       0       0       0       0       1

/* pfe0499_nml */
1.13    0.98    2.69    0.12    0.08    0.07
0       0       0       0       0       1

/* pfe0532_nml */
1.01    0.78    3.31    0.24    0.16    0.23
0       0       0       0       0       1

/* pfe0557_nml */
1.73    1.31    2.96    0.02    0.29    0.14
0       0       0       0       0       1

/* pfe0563_nml */
0.99    0.78    3.18    0.43    0.06    0.03
0       0       0       0       0       1

/* pfe0564_nml */
1.19    0.90    3.60    0.18    0.12    0.32
0       0       0       0       0       1

/* rav0372_nml */
1.12    1.04    4.06    0.06    0.09    0.17
0       0       0       0       0       1
```



```
/* rav0373_nml */
0.69    0.74    4.15    0.04    0.02    0.17
0       0       0       0       0       1

/* rav0399_as1 */
2.19    1.15    1.91    0.47    0.41    0.16
1       0       0       0       0       0

/* rav0400_as1 */
3.00    1.34    1.47    0.32    0.42    0.22
1       0       0       0       0       0

/* rav0404_nml */
1.19    1.06    4.67    0.17    0.02    0.14
0       0       0       0       0       1

/* rav0405_nml */
0.89    0.83    4.29    0.26    0.24    0.08
0       0       0       0       0       1

/* rav0431_as1 */
2.34    1.31    2.06    0.35    0.02    0.13
1       0       0       0       0       0

/* rav0432_as1 */
3.36    1.54    1.41    0.42    0.46    0.26
1       0       0       0       0       0

/* rav0436_nml */
1.14    1.19    4.85    0.25    0.02    0.12
0       0       0       0       0       1

/* rav0437_nml */
0.88    1.13    4.61    0.16    0.10    0.10
0       0       0       0       0       1

/* rav0463_as1 */
2.52    1.57    2.28    0.45    0.02    0.18
1       0       0       0       0       0

/* rav0464_as1 */
3.68    1.71    1.51    0.30    0.50    0.11
1       0       0       0       0       0

/* rav0495_as1 */
2.40    1.65    2.02    0.41    0.02    0.15
1       0       0       0       0       0

/* rav0526_as1 */
1.70    1.29    2.50    0.47    0.02    0.17
1       0       0       0       0       0

/* rav0527_as1 */
2.20    1.61    1.99    0.54    0.39    0.15
1       0       0       0       0       0

/* sav0276_nml */
1.37    0.92    4.18    0.04    0.04    0.33
0       0       0       0       0       1

/* sav0301_as1 */
1.38    0.46    1.52    1.09    0.43    0.17
1       0       0       0       0       0

/* sav0302_as1 */
1.61    0.47    1.08    0.33    0.37    0.24
1       0       0       0       0       0

/* sav0308_nml */
1.64    1.42    5.25    0.12    0.39    0.28
0       0       0       0       0       1

/* sav0334_as1 */
```

```
1.68    1.02    3.15    1.19    0.04    0.44
1       0       0       0       0       0

/* sav0335_as1 */
1.52    0.57    2.35    1.17    0.04    0.29
1       0       0       0       0       0

/* sav0372_nml */
1.51    1.14    5.66    0.26    0.23    0.44
0       0       0       0       0       1

/* sav0373_nml */
0.99    0.78    3.96    0.46    0.33    0.37
0       0       0       0       0       1

/* sav0404_nml */
1.14    0.74    5.12    0.14    0.35    0.50
0       0       0       0       0       1

/* sig0372_nml */
0.83    0.91    3.01    0.20    0.33    0.08
0       0       0       0       0       1

/* sig0404_nml */
1.04    1.14    4.06    0.27    0.02    0.25
0       0       0       0       0       1

/* sig0436_nml */
0.99    1.08    3.92    0.55    0.06    0.29
0       0       0       0       0       1

/* sig0469_nml */
0.68    0.96    3.32    0.26    0.44    0.13
0       0       0       0       0       1

/* sig0525_gbm */
1.42    0.77    0.77    0.81    1.47    0.33
0       0       1       0       0       0

/* sig0526_gbm */
1.42    0.79    1.00    1.21    1.33    0.33
0       0       1       0       0       0

/* sig0533_nml */
0.90    0.91    3.59    0.27    0.51    0.21
0       0       0       0       0       1

/* sig0556_gbm */
1.31    0.51    0.93    0.73    0.98    0.29
0       0       1       0       0       0

/* sig0557_gbm */
1.04    0.36    0.58    0.80    1.38    0.53
0       0       1       0       0       0

/* sig0558_gbm */
1.12    0.29    0.46    1.46    1.63    0.29
0       0       1       0       0       0

/* sig0623_gbm */
1.23    0.64    0.75    0.94    0.95    0.30
0       0       1       0       0       0

/* vaj0434_gbm */
1.61    0.90    1.45    2.01    1.52    0.35
0       0       1       0       0       0

/* vaj0465_gbm */
1.89    1.29    1.28    1.71    3.55    0.59
0       0       1       0       0       0

/* vaj0466_gbm */
1.12    0.64    0.62    2.61    3.10    0.67
```



```
0       0       1       0       0       0

/* vaj0467_gbm */
0.67    0.50    0.84    3.25    4.03    0.47
0       0       1       0       0       0

/* vaj0497_gbm */
1.47    0.90    0.83    1.68    2.70    0.66
0       0       1       0       0       0

/* vaj0498_gbm */
1.08    0.63    0.66    2.28    3.20    0.63
0       0       1       0       0       0

/* vaj0499_gbm */
1.05    0.53    0.86    2.09    4.04    0.50
0       0       1       0       0       0

/* vaj0531_gbm */
1.64    0.66    0.97    1.92    4.16    0.66
0       0       1       0       0       0

/* vaj0563_gbm */
1.19    0.58    0.85    1.99    4.96    1.06
0       0       1       0       0       0

/* vaj0620_nml */
1.24    0.89    2.64    0.03    0.07    0.15
0       0       0       0       0       1

/* vaj0654_nml */
1.23    0.88    2.96    0.11    0.42    0.10
0       0       0       0       0       1

/* vaj0655_nml */
1.47    1.07    3.50    0.04    0.16    0.06
0       0       0       0       0       1

/* vaj0656_nml */
1.42    1.07    2.92    0.02    0.21    0.07
0       0       0       0       0       1

/* vaj0657_nml */
1.27    1.08    2.61    0.09    0.31    0.11
0       0       0       0       0       1

/* vau0492_nml */
1.26    0.80    3.15    0.24    0.24    0.21
0       0       0       0       0       1

/* vau0556_nml */
1.25    0.92    3.03    0.12    0.10    0.15
0       0       0       0       0       1

/* vau0557_nml */
1.31    1.18    4.18    0.12    0.03    0.28
0       0       0       0       0       1

/* vau0589_nml */
1.29    1.09    3.65    0.13    0.03    0.30
0       0       0       0       0       1

/* vau0628_met */
4.71    0.85    0.56    0.35    7.27    1.89
0       0       0       1       0       0

/* vau0654_nml */
1.20    1.05    3.77    0.03    0.17    0.20
0       0       0       0       0       1

/* vau0658_met */
3.93    0.90    0.74    0.64    5.97    2.08
0       0       0       1       0       0
```

```
/* vau0659_met */
5.09      0.91      0.57      0.22      8.99      2.88
0         0         0         1         0         0

/* vau0660_met */
5.03      0.96      0.62      0.32      8.16      2.01
0         0         0         1         0         0

/* vau0687_nml */
1.15      0.95      3.08      0.10      0.33      0.12
0         0         0         0         0         1

/* vog0396_nml */
0.92      1.00      1.66      0.07      0.17      0.17
0         0         0         0         0         1

/* vog0397_nml */
1.08      0.90      1.95      0.08      0.26      0.19
0         0         0         0         0         1

/* vog0402_nml */
0.85      0.89      1.72      0.30      0.20      0.20
0         0         0         0         0         1

/* vog0428_nml */
1.09      1.45      1.97      0.05      0.02      0.22
0         0         0         0         0         1

/* vog0434_nml */
1.25      1.13      2.31      0.05      0.07      0.17
0         0         0         0         0         1

/* vog0435_nml */
0.75      0.64      1.93      0.02      0.08      0.21
0         0         0         0         0         1

/* vog0464_non */
0.60      0.28      0.34      0.85      0.42      0.02
0         0         0         0         0         0

/* vog0496_non */
0.39      0.54      0.24      0.17      0.91      0.08
0         0         0         0         0         0

/* vog0527_non */
0.39      0.68      0.34      0.25      0.81      0.19
0         0         0         0         0         0

/* vog0528_non */
0.69      0.22      0.37      0.73      0.92      0.27
0         0         0         0         0         0

/* vog0560_non */
0.62      0.39      0.31      0.68      0.68      0.18
0         0         0         0         0         0

/* vog0561_non */
0.12      0.02      0.11      0.02      0.84      0.13
0         0         0         0         0         0
```

```
/* bis0268_as2 */
1.85    1.11    1.11    0.78    1.08    0.23
0       1       0       0       0       0

/* bis0269_as2 */
2.04    0.67    1.11    0.57    0.98    0.19
0       1       0       0       0       0

/* bis0308_nml */
1.29    0.91    3.21    0.00    0.16    0.18
0       0       0       0       0       1

/* bis0331_as2 */
2.43    0.88    0.63    0.15    1.28    0.18
0       1       0       0       0       0

/* bis0335_as2 */
1.45    0.47    0.86    0.44    0.98    0.07
0       1       0       0       0       0

/* bis0340_nml */
1.55    1.18    3.50    0.02    0.04    0.25
0       0       0       0       0       1

/* bis0363_as2 */
2.82    1.20    0.81    0.07    0.89    0.30
0       1       0       0       0       0

/* bis0364_as2 */
3.29    0.89    0.72    0.28    1.50    0.08
0       1       0       0       0       0

/* bis0365_as2 */
2.61    0.46    0.67    0.12    1.76    0.19
0       1       0       0       0       0

/* bis0366_as2 */
1.67    0.42    1.03    0.25    0.85    0.12
0       1       0       0       0       0

/* bis0372_nml */
1.33    0.99    3.68    0.14    0.21    0.27
0       0       0       0       0       1

/* bis0404_nml */
1.11    0.97    4.01    0.05    0.06    0.11
0       0       0       0       0       1

/* bis0436_nml */
0.93    0.87    4.24    0.01    0.05    0.16
0       0       0       0       0       1

/* bis0468_nml */
1.09    1.08    3.77    0.03    0.17    0.17
0       0       0       0       0       1

/* bri0365_nml */
1.21    1.16    3.51    0.09    0.23    0.08
0       0       0       0       0       1

/* bri0397_nml */
1.07    1.15    3.02    0.00    0.12    0.14
0       0       0       0       0       1

/* bri0401_as1 */
2.55    1.43    1.22    0.00    0.46    0.02
1       0       0       0       0       0

/* bri0402_as1 */
1.92    1.36    2.06    0.00    0.59    0.07
1       0       0       0       0       0
```



```
/* bri0429_nml */
0.99    1.21    3.39    0.05    0.24    0.13
0       0       0       0       0       1

/* bri0432_as1 */
2.57    1.60    2.71    0.29    0.20    0.05
1       0       0       0       0       0

/* bri0433_as1 */
4.01    2.11    1.88    0.49    0.56    0.01
1       0       0       0       0       0

/* bri0434_as1 */
2.71    1.51    2.60    0.13    0.25    0.21
1       0       0       0       0       0

/* bri0461_nml */
0.74    0.85    4.03    0.00    0.22    0.11
0       0       0       0       0       1

/* bri0465_as1 */
3.27    1.80    2.02    0.42    0.61    0.10
1       0       0       0       0       0

/* bri0500_nml */
0.70    0.94    4.31    0.15    0.13    0.18
0       0       0       0       0       1

/* bri0565_nml */
0.80    0.69    3.57    0.28    0.23    0.08
0       0       0       0       0       1

/* cov0332_men */
1.76    0.62    0.67    0.81    0.94    0.17
0       0       0       0       1       0

/* cov0333_men */
0.97    0.60    0.53    0.69    0.54    0.03
0       0       0       0       1       0

/* cov0364_men */
1.47    0.65    0.39    1.41    1.24    0.14
0       0       0       0       1       0

/* cov0365_men */
1.11    0.50    0.61    0.13    0.11    0.12
0       0       0       0       1       0

/* cov0373_nml */
1.42    1.05    3.72    0.03    0.09    0.16
0       0       0       0       0       1

/* cov0396_men */
1.66    0.68    0.40    1.25    0.78    0.13
0       0       0       0       1       0

/* cov0397_men */
1.10    0.67    0.57    0.46    0.58    0.16
0       0       0       0       1       0

/* cov0405_nml */
1.50    1.39    3.11    0.11    0.20    0.08
0       0       0       0       0       1

/* cov0428_men */
1.73    0.41    0.53    1.43    1.05    0.15
0       0       0       0       1       0

/* cov0429_men */
1.33    0.71    0.53    0.94    0.89    0.00
0       0       0       0       1       0

/* cov0437_nml */
```

```
1.29      0.73      2.78      0.16      0.16      0.26
0         0         0         0         0         1

/* cov0469_nml */
1.24      0.83      2.38      0.07      0.13      0.14
0         0         0         0         0         1

/* for0372_nml */
0.88      0.97      2.84      0.10      0.16      0.07
0         0         0         0         0         1

/* for0404_nml */
0.77      0.87      3.05      0.09      0.19      0.15
0         0         0         0         0         1

/* for0405_nml */
0.99      0.97      2.74      0.22      0.06      0.23
0         0         0         0         0         1

/* for0436_nml */
0.94      0.98      3.48      0.36      0.61      0.26
0         0         0         0         0         1

/* for0437_nml */
0.97      1.20      3.30      0.08      0.09      0.16
0         0         0         0         0         1

/* for0493_gbm */
1.35      0.48      1.35      0.16      1.78      0.27
0         0         1         0         0         0

/* for0494_gbm */
1.45      0.65      0.55      0.15      3.08      0.27
0         0         1         0         0         0

/* for0495_gbm */
1.53      0.34      0.90      0.31      4.13      0.56
0         0         1         0         0         0

/* for0496_gbm */
1.72      0.30      0.64      0.31      5.53      0.86
0         0         1         0         0         0

/* for0497_gbm */
2.02      0.30      0.79      0.70      4.61      0.77
0         0         1         0         0         0

/* for0498_gbm */
1.22      1.02      0.72      0.24      4.90      0.36
0         0         1         0         0         0

/* for0526_gbm */
2.06      0.08      0.54      0.42      4.64      0.52
0         0         1         0         0         0

/* for0530_gbm */
2.02      0.55      0.78      0.41      3.62      0.24
0         0         1         0         0         0

/* for0558_gbm */
2.91      0.90      0.76      0.82      3.18      0.24
0         0         1         0         0         0

/* for0559_gbm */
3.42      1.29      1.41      0.74      2.05      0.23
0         0         1         0         0         0

/* ful0405_nml */
1.42      0.69      4.33      0.08      0.17      0.11
0         0         0         0         0         1

/* ful0437_nml */
1.39      0.56      5.02      0.09      0.19      0.17
```



```
0         0         0         0         0         1

/* ful0461_nml */
1.66      1.43      5.69      0.06      0.14      0.21
0         0         0         0         0         1

/* ful0469_nml */
1.14      0.81      5.39      0.02      0.25      0.23
0         0         0         0         0         1

/* ful0500_nml */
1.50      0.97      4.86      0.12      0.32      0.06
0         0         0         0         0         1

/* ful0501_nml */
1.31      0.81      5.34      0.12      0.26      0.08
0         0         0         0         0         1

/* ful0525_nml */
1.49      1.29      5.17      0.12      0.17      0.12
0         0         0         0         0         1

/* ful0533_nml */
1.77      0.96      5.28      0.15      0.27      0.10
0         0         0         0         0         1

/* ful0556_nml */
1.11      1.12      4.20      0.03      0.15      0.08
0         0         0         0         0         1

/* ful0557_nml */
1.29      1.21      4.45      0.06      0.15      0.10
0         0         0         0         0         1

/* ful0565_nml */
1.67      0.93      4.89      0.18      0.12      0.11
0         0         0         0         0         1

/* ful0588_nml */
1.35      1.23      4.32      0.00      0.16      0.08
0         0         0         0         0         1

/* mym0372_nml */
1.82      1.06      3.44      0.07      0.01      0.10
0         0         0         0         0         1

/* mym0405_nml */
1.27      0.81      3.68      0.17      0.09      0.08
0         0         0         0         0         1

/* mym0493_nml */
1.29      1.18      3.83      0.26      0.13      0.07
0         0         0         0         0         1

/* mym0498_met */
2.53      0.94      1.01      1.39      5.57      0.93
0         0         0         1         0         0

/* mym0499_met */
2.00      0.75      0.65      1.21      8.62      1.71
0         0         0         1         0         0

/* mym0532_met */
1.66      0.72      0.90      0.86      9.77      2.27
0         0         0         1         0         0

/* mym0557_nml */
1.31      1.14      3.76      0.04      0.41      0.11
0         0         0         0         0         1

/* mym0558_nml */
1.39      0.93      3.42      0.38      0.74      0.17
0         0         0         0         0         1
```

```
/* mym0563_met */
1.58      0.61      0.50      0.33      12.03     3.09
0         0         0         1         0         0

/* mym0564_met */
1.67      0.57      0.79      0.46      10.37     2.58
0         0         0         1         0         0

/* mym0590_nml */
1.28      0.88      3.50      0.18      0.39      0.07
0         0         0         0         0         1

/* pfe0271_men */
1.87      0.41      0.17      0.19      0.15      0.12
0         0         0         0         1         0

/* pfe0272_men */
1.80      0.50      0.08      0.03      0.03      0.15
0         0         0         0         1         0

/* pfe0273_men */
1.02      1.05      0.15      0.46      0.50      0.07
0         0         0         1         0         0

/* pfe0303_men */
1.80      0.49      0.31      0.11      0.10      0.06
0         0         0         0         1         0

/* pfe0304_men */
1.90      0.48      0.19      0.10      0.06      0.14
0         0         0         0         1         0

/* pfe0305_men */
1.76      0.36      0.10      0.12      0.16      0.14
0         0         0         0         1         0

/* pfe0306_men */
0.01      0.01      0.04      0.00      0.02      0.10
0         0         0         0         1         0

/* pfe0338_men */
2.05      0.73      0.49      0.11      0.14      0.00
0         0         0         0         1         0

/* pfe0467_nml */
1.46      1.07      2.65      0.03      0.26      0.11
0         0         0         0         0         1

/* pfe0499_nml */
1.02      0.97      2.64      0.07      0.22      0.02
0         0         0         0         0         1

/* pfe0532_nml */
0.99      0.83      3.44      0.15      0.12      0.13
0         0         0         0         0         1

/* pfe0557_nml */
1.43      1.18      2.99      0.06      0.26      0.05
0         0         0         0         0         1

/* pfe0563_nml */
1.05      0.94      3.69      0.40      0.30      0.02
0         0         0         0         0         1

/* pfe0564_nml */
1.20      1.01      3.67      0.11      0.10      0.18
0         0         0         0         0         1

/* rav0372_nml */
1.15      0.99      4.85      0.03      0.07      0.10
0         0         0         0         0         1
```



```
/* rav0373_nml */
0.87      0.69      5.07      0.02      0.16      0.28
0         0         0         0         0         1

/* rav0399_as1 */
2.42      1.40      2.26      0.27      0.48      0.19
1         0         0         0         0         0

/* rav0400_as1 */
3.76      1.66      1.59      0.25      0.43      0.21
1         0         0         0         0         0

/* rav0404_nml */
1.16      1.07      5.42      0.10      0.02      0.10
0         0         0         0         0         1

/* rav0405_nml */
0.89      0.81      5.58      0.23      0.34      0.15
0         0         0         0         0         1

/* rav0431_as1 */
2.60      1.58      2.26      0.25      0.53      0.06
1         0         0         0         0         0

/* rav0432_as1 */
4.08      2.09      1.42      0.37      0.49      0.22
1         0         0         0         0         0

/* rav0436_nml */
1.26      1.21      5.50      0.18      0.28      0.14
0         0         0         0         0         1

/* rav0437_nml */
0.97      1.22      5.65      0.07      0.10      0.11
0         0         0         0         0         1

/* rav0463_as1 */
2.91      1.92      2.35      0.51      0.80      0.13
1         0         0         0         0         0

/* rav0464_as1 */
4.08      2.21      1.51      0.33      0.52      0.07
1         0         0         0         0         0

/* rav0495_as1 */
2.95      2.08      1.98      0.36      0.66      0.13
1         0         0         0         0         0

/* rav0526_as1 */
1.77      1.72      2.53      0.37      0.59      0.23
1         0         0         0         0         0

/* rav0527_as1 */
2.69      2.02      2.20      0.37      0.49      0.16
1         0         0         0         0         0

/* sav0276_nml */
1.42      0.84      4.85      0.00      0.30      0.42
0         0         0         0         0         1

/* sav0301_as1 */
1.59      0.47      1.93      0.89      1.71      0.06
1         0         0         0         0         0

/* sav0302_as1 */
1.42      0.62      1.05      0.14      3.06      0.20
1         0         0         0         0         0

/* sav0308_nml */
1.74      1.22      5.41      0.07      0.23      0.19
0         0         0         0         0         1

/* sav0334_as1 */
```

```
2.04      1.09      3.34      0.84      1.31      0.31
1         0         0         0         0         0

/* sav0335_as1 */
1.78      0.70      2.59      0.96      1.52      0.34
1         0         0         0         0         0

/* sav0372_nml */
1.66      1.17      5.81      0.14      0.09      0.41
0         0         0         0         0         1

/* sav0373_nml */
0.86      0.99      4.59      0.29      0.29      0.26
0         0         0         0         0         1

/* sav0404_nml */
1.03      0.78      5.98      0.10      0.39      0.45
0         0         0         0         0         1

/* sig0372_nml */
0.89      0.98      3.27      0.09      0.64      0.03
0         0         0         0         0         1

/* sig0404_nml */
1.13      1.14      4.03      0.35      0.24      0.24
0         0         0         0         0         1

/* sig0436_nml */
1.01      1.01      4.30      0.85      0.37      0.17
0         0         0         0         0         1

/* sig0469_nml */
0.68      0.97      4.09      0.25      0.39      0.06
0         0         0         0         0         1

/* sig0525_gbm */
1.64      0.74      0.74      0.42      3.87      0.63
0         0         1         0         0         0

/* sig0526_gbm */
1.51      0.81      0.88      1.43      2.02      0.33
0         0         1         0         0         0

/* sig0533_nml */
1.06      0.90      4.52      0.50      0.42      0.22
0         0         0         0         0         1

/* sig0556_gbm */
1.64      0.50      0.99      0.60      2.81      0.29
0         0         1         0         0         0

/* sig0557_gbm */
1.28      0.42      0.51      0.54      3.65      0.67
0         0         1         0         0         0

/* sig0558_gbm */
1.32      0.31      0.46      1.32      3.20      0.49
0         0         1         0         0         0

/* sig0623_gbm */
1.18      0.70      0.73      0.41      3.23      0.25
0         0         1         0         0         0

/* vaj0434_gbm */
1.46      0.83      1.34      1.81      3.56      0.45
0         0         1         0         0         0

/* vaj0465_gbm */
1.70      1.14      1.00      0.67      4.48      0.68
0         0         1         0         0         0

/* vaj0466_gbm */
1.05      0.50      0.44      1.20      8.91      1.08
```



```
0         0         1         0         0         0

/* vaj0467_gbm */
0.73      0.39      0.77      3.10      4.46      0.90
0         0         1         0         0         0

/* vaj0497_gbm */
1.33      0.70      0.60      0.71      7.37      0.90
0         0         1         0         0         0

/* vaj0498_gbm */
1.02      0.56      0.57      1.05      8.85      0.86
0         0         1         0         0         0

/* vaj0499_gbm */
1.08      0.57      0.77      0.94      5.75      0.78
0         0         1         0         0         0

/* vaj0531_gbm */
1.57      0.67      0.83      1.04      6.17      1.13
0         0         1         0         0         0

/* vaj0563_gbm */
1.20      0.52      0.71      1.10      7.19      1.59
0         0         1         0         0         0

/* vaj0620_nml */
1.12      0.78      2.32      0.05      0.05      0.22
0         0         0         0         0         1

/* vaj0654_nml */
1.08      0.81      2.83      0.06      0.44      0.09
0         0         0         0         0         1

/* vaj0655_nml */
1.30      1.04      3.62      0.04      0.18      0.06
0         0         0         0         0         1

/* vaj0656_nml */
1.37      1.14      3.25      0.00      0.15      0.03
0         0         0         0         0         1

/* vaj0657_nml */
1.19      1.23      2.86      0.07      0.33      0.08
0         0         0         0         0         1

/* vau0492_nml */
1.20      0.79      3.73      0.14      0.41      0.17
0         0         0         0         0         1

/* vau0556_nml */
1.04      0.79      3.36      0.14      0.21      0.07
0         0         0         0         0         1

/* vau0557_nml */
1.45      1.20      4.32      0.06      0.05      0.20
0         0         0         0         0         1

/* vau0589_nml */
1.44      1.26      4.06      0.11      0.20      0.15
0         0         0         0         0         1

/* vau0628_met */
5.11      1.12      0.54      0.17      10.16     2.39
0         0         0         1         0         0

/* vau0654_nml */
1.17      1.01      3.57      0.00      0.18      0.20
0         0         0         0         0         1

/* vau0658_met */
3.88      1.21      0.62      0.33      8.47      2.88
0         0         0         1         0         0
```

```
/* vau0659_met */
5.52      1.37      0.46      0.08      13.09     4.19
0         0         0         1         0         0

/* vau0660_met */
5.83      1.22      0.49      0.11      11.33     2.98
0         0         0         1         0         0

/* vau0687_nml */
1.07      0.95      3.27      0.25      0.53      0.08
0         0         0         0         0         1

/* vog0396_nml */
1.00      1.13      1.65      0.03      0.16      0.21
0         0         0         0         0         1

/* vog0397_nml */
1.04      0.97      2.09      0.04      0.17      0.21
0         0         0         0         0         1

/* vog0402_nml */
1.37      0.84      1.89      0.20      0.32      0.09
0         0         0         0         0         1

/* vog0428_nml */
1.12      1.58      1.94      0.03      0.08      0.13
0         0         0         0         0         1

/* vog0434_nml */
1.36      1.04      2.32      0.02      0.03      0.16
0         0         0         0         0         1

/* vog0435_nml */
0.77      0.44      2.35      0.00      0.03      0.19
0         0         0         0         0         1

/* vog0464_non */
1.19      0.45      0.31      0.97      1.06      0.02
0         0         0         0         0         0

/* vog0496_non */
0.53      0.65      0.31      0.07      1.86      0.05
0         0         0         0         0         0

/* vog0527_non */
0.52      1.56      0.32      0.18      1.39      0.10
0         0         0         0         0         0

/* vog0528_non */
1.21      0.46      0.41      1.17      0.82      0.32
0         0         0         0         0         0

/* vog0560_non */
1.27      0.55      0.46      1.18      0.96      0.24
0         0         0         0         0         0

/* vog0561_non */
0.05      0.01      0.07      0.00      1.78      0.06
0         0         0         0         0         0
```

```
/* bisNorm_nml */
1.10      1.00      3.34      0.08      0.08      0.20
0         0         0         0         0         1

/* bisTumr_as2 */
2.08      0.68      0.77      0.43      0.62      0.17
0         1         0         0         0         0

/* briNorm_nml */
1.03      1.00      3.24      0.17      0.19      0.19
0         0         0         0         0         1

/* briTumr_as1 */
2.91      1.36      1.92      0.19      0.26      0.13
1         0         0         0         0         0

/* covNorm_nml */
1.11      1.00      2.64      0.13      0.06      0.16
0         0         0         0         0         1

/* covTumr_men */
1.36      0.64      0.58      0.80      0.61      0.12
0         0         0         0         1         0

/* forNorm_nml */
0.95      1.00      2.86      0.24      0.08      0.20
0         0         0         0         0         1

/* forTumr_gbm */
1.98      0.49      0.76      0.74      1.46      0.42
0         0         1         0         0         0

/* fulNorm_nml */
1.19      1.00      4.36      0.10      0.12      0.16
0         0         0         0         0         1

/* mymNorm_nml */
1.17      1.00      3.02      0.20      0.21      0.11
0         0         0         0         0         1

/* mymTumr_met */
1.57      0.51      0.66      1.10      5.61      1.25
0         0         0         1         0         0

/* pfeNorm_nml */
1.27      1.00      3.11      0.18      0.14      0.16
0         0         0         0         0         1

/* pfeTumr_men */
0.98      0.50      0.28      0.19      0.11      0.12
0         0         0         0         1         0

/* ravNorm_nml */
0.99      1.00      4.44      0.15      0.08      0.13
0         0         0         0         0         1

/* ravTumr_as1 */
2.60      1.46      1.91      0.41      0.25      0.17
1         0         0         0         0         0

/* savNorm_nml */
1.33      1.00      4.83      0.20      0.27      0.38
0         0         0         0         0         1

/* savTumr_as1 */
1.55      0.63      2.02      0.95      0.22      0.29
1         0         0         0         0         0

/* sigNorm_nml */
0.89      1.00      3.58      0.31      0.27      0.19
0         0         0         0         0         1
```



```
/* sigTumr_gbm */
1.26      0.56      0.75      0.99      1.29      0.34
0         0         1         0         0         0

/* vajNorm_nml */
1.33      1.00      2.93      0.06      0.23      0.10
0         0         0         0         0         1

/* vajTumr_gbm */
1.30      0.74      0.93      2.17      3.47      0.62
0         0         1         0         0         0

/* vauNorm_nml */
1.24      1.00      3.48      0.12      0.15      0.21
0         0         0         0         0         1

/* vauTumr_met */
4.69      0.90      0.62      0.38      7.60      2.22
0         0         0         1         0         0

/* vogNorm_nml */
0.99      1.00      1.93      0.10      0.14      0.19
0         0         0         0         0         1

/* vogTumr_non */
0.47      0.36      0.29      0.45      0.76      0.14
0         0         0         0         0         0
```

```
/* bisNorm_nml */
1.21    1.00    3.74    0.04    0.11    0.19
0       0       0       0       0       1

/* bisTumr_as2 */
2.27    0.76    0.87    0.33    1.17    0.17
0       1       0       0       0       0

/* briNorm_nml */
0.92    1.00    3.64    0.10    0.19    0.12
0       0       0       0       0       1

/* briTumr_as1 */
2.84    1.63    2.08    0.22    0.45    0.08
1       0       0       0       0       0

/* covNorm_nml */
1.36    1.00    3.00    0.09    0.15    0.16
0       0       0       0       0       1

/* covTumr_men */
1.39    0.61    0.53    0.89    0.77    0.11
0       0       0       0       1       0

/* forNorm_nml */
0.91    1.00    3.08    0.17    0.22    0.17
0       0       0       0       0       1

/* forTumr_gbm */
1.97    0.59    0.84    0.43    3.75    0.43
0       0       1       0       0       0

/* fulNorm_nml */
1.42    1.00    4.91    0.09    0.20    0.12
0       0       0       0       0       1

/* mymNorm_nml */
1.39    1.00    3.61    0.18    0.30    0.10
0       0       0       0       0       1

/* mymTumr_met */
1.89    0.72    0.77    0.85    9.28    2.12
0       0       0       1       0       0

/* pfeNorm_nml */
1.19    1.00    3.18    0.14    0.21    0.09
0       0       0       0       0       1

/* pfeTumr_men */
1.52    0.50    0.19    0.14    0.14    0.10
0       0       0       0       1       0

/* ravNorm_nml */
1.05    1.00    5.35    0.11    0.16    0.15
0       0       0       0       0       1

/* ravTumr_as1 */
3.03    1.85    2.01    0.34    0.55    0.16
1       0       0       0       0       0

/* savNorm_nml */
1.34    1.00    5.33    0.12    0.26    0.35
0       0       0       0       0       1

/* savTumr_as1 */
1.71    0.72    2.23    0.71    1.90    0.23
1       0       0       0       0       0

/* sigNorm_nml */
0.95    1.00    4.04    0.40    0.41    0.15
0       0       0       0       0       1
```



```
/* sigTumr_gbm */
1.43    0.58    0.72    0.79    3.13    0.44
0       0       1       0       0       0

/* vajNorm_nml */
1.21    1.00    2.98    0.04    0.23    0.10
0       0       0       0       0       1

/* vajTumr_gbm */
1.24    0.65    0.78    1.29    6.30    0.93
0       0       1       0       0       0

/* vauNorm_nml */
1.23    1.00    3.72    0.12    0.26    0.15
0       0       0       0       0       1

/* vauTumr_met */
5.08    1.23    0.52    0.17    10.76   3.11
0       0       0       1       0       0

/* vogNorm_nml */
1.11    1.00    2.04    0.05    0.13    0.17
0       0       0       0       0       1

/* vogTumr_non */
0.79    0.61    0.31    0.59    1.31    0.13
0       0       0       0       0       0
```

```
/* bis0268_as2 */
1.66      0.99      0.85      0.62      0.76      0.30
0         1         0         0         0         0

/* bis0269_as2 */
1.51      0.50      0.70      0.58      0.75      0.18
0         1         0         0         0         0

/* bis0308_nml */
1.09      0.99      2.80      0.02      0.02      0.13
0         0         0         0         0         1

/* bis0331_as2 */
2.12      0.69      0.74      0.26      0.50      0.13
0         1         0         0         0         0

/* bis0335_as2 */
1.18      0.30      0.74      0.55      0.74      0.13
0         1         0         0         0         0

/* bis0340_nml */
1.20      1.15      3.08      0.04      0.06      0.21
0         0         0         0         0         1

/* bis0363_as2 */
2.44      1.07      0.81      0.15      0.41      0.19
0         1         0         0         0         0

/* bis0364_as2 */
3.14      0.97      0.73      0.50      0.52      0.19
0         1         0         0         0         0

/* bis0365_as2 */
2.71      0.61      0.66      0.31      0.47      0.15
0         1         0         0         0         0

/* bis0366_as2 */
1.88      0.30      0.94      0.50      0.80      0.07
0         1         0         0         0         0

/* bis0372_nml */
1.16      0.92      3.34      0.19      0.02      0.21
0         0         0         0         0         1

/* bis0404_nml */
0.99      0.99      3.68      0.11      0.08      0.21
0         0         0         0         0         1

/* bis0436_nml */
0.96      0.98      3.86      0.03      0.06      0.18
0         0         0         0         0         1

/* bis0468_nml */
1.19      0.97      3.32      0.09      0.24      0.26
0         0         0         0         0         1
```

```
/* bis0268_as2 */
1.85      1.11      1.11      0.78      1.08      0.23
0         1         0         0         0         0

/* bis0269_as2 */
2.04      0.67      1.11      0.57      0.98      0.19
0         1         0         0         0         0

/* bis0308_nml */
1.29      0.91      3.21      0.00      0.16      0.18
0         0         0         0         0         1

/* bis0331_as2 */
2.43      0.88      0.63      0.15      1.28      0.18
0         1         0         0         0         0

/* bis0335_as2 */
1.45      0.47      0.86      0.44      0.98      0.07
0         1         0         0         0         0

/* bis0340_nml */
1.55      1.18      3.50      0.02      0.04      0.25
0         0         0         0         0         1

/* bis0363_as2 */
2.82      1.20      0.81      0.07      0.89      0.30
0         1         0         0         0         0

/* bis0364_as2 */
3.29      0.89      0.72      0.28      1.50      0.08
0         1         0         0         0         0

/* bis0365_as2 */
2.61      0.46      0.67      0.12      1.76      0.19
0         1         0         0         0         0

/* bis0366_as2 */
1.67      0.42      1.03      0.25      0.85      0.12
0         1         0         0         0         0

/* bis0372_nml */
1.33      0.99      3.68      0.14      0.21      0.27
0         0         0         0         0         1

/* bis0404_nml */
1.11      0.97      4.01      0.05      0.06      0.11
0         0         0         0         0         1

/* bis0436_nml */
0.93      0.87      4.24      0.01      0.05      0.16
0         0         0         0         0         1

/* bis0468_nml */
1.09      1.08      3.77      0.03      0.17      0.17
0         0         0         0         0         1
```

D.2 Samples of Data Sets and Statistical Report

D.2.1 Train Set Sample

D.2.2 Test Set Sample

D.2.3 Statistical Report

```
# '$Id: tm.in, v1.0, 94/06/02 Jane EXP $'
tag: bis0268_as2
 0.141329  0.084798  0.084798  0.059587  0.082506  0.017571  ,
 0.000000  1.000000  0.000000  0.000000  0.000000  0.000000  ;

tag: bis0269_as2
 0.155844  0.051184  0.084798  0.043545  0.074866  0.014515  ,
 0.000000  1.000000  0.000000  0.000000  0.000000  0.000000  ;

tag: bis0308_nml
 0.098549  0.069519  0.245225  0.000000  0.012223  0.013751  ,
 0.000000  0.000000  0.000000  0.000000  0.000000  1.000000  ;

tag: bis0331_as2
 0.185638  0.067227  0.048128  0.011459  0.097785  0.013751  ,
 0.000000  1.000000  0.000000  0.000000  0.000000  0.000000  ;

tag: bis0335_as2
 0.110772  0.035905  0.065699  0.033613  0.074866  0.005348  ,
 0.000000  1.000000  0.000000  0.000000  0.000000  0.000000  ;

tag: bis0340_nml
 0.118411  0.090145  0.267380  0.001528  0.003056  0.019099  ,
 0.000000  0.000000  0.000000  0.000000  0.000000  1.000000  ;

tag: bis0363_as2
 0.215432  0.091673  0.061879  0.005348  0.067991  0.022918  ,
 0.000000  1.000000  0.000000  0.000000  0.000000  0.000000  ;

tag: bis0364_as2
 0.251337  0.067991  0.055004  0.021390  0.114591  0.006112  ,
 0.000000  1.000000  0.000000  0.000000  0.000000  0.000000  ;

tag: bis0365_as2
 0.199389  0.035141  0.051184  0.009167  0.134454  0.014515  ,
 0.000000  1.000000  0.000000  0.000000  0.000000  0.000000  ;

tag: bis0366_as2
 0.127578  0.032086  0.078686  0.019099  0.064935  0.009167  ,
 0.000000  1.000000  0.000000  0.000000  0.000000  0.000000  ;

tag: bis0372_nml
 0.101604  0.075630  0.281131  0.010695  0.016043  0.020626  ,
 0.000000  0.000000  0.000000  0.000000  0.000000  1.000000  ;

tag: bis0404_nml
 0.084798  0.074102  0.306341  0.003820  0.004584  0.008403  ,
 0.000000  0.000000  0.000000  0.000000  0.000000  1.000000  ;

tag: bis0436_nml
 0.071047  0.066463  0.323911  0.000764  0.003820  0.012223  ,
 0.000000  0.000000  0.000000  0.000000  0.000000  1.000000  ;

tag: bis0468_nml
 0.083270  0.082506  0.288006  0.002292  0.012987  0.012987  ,
 0.000000  0.000000  0.000000  0.000000  0.000000  1.000000  ;

tag: bri0365_nml
 0.092437  0.088617  0.268144  0.006875  0.017571  0.006112  ,
 0.000000  0.000000  0.000000  0.000000  0.000000  1.000000  ;

tag: bri0397_nml
 0.081742  0.087853  0.230710  0.000000  0.009167  0.010695  ,
 0.000000  0.000000  0.000000  0.000000  0.000000  1.000000  ;

tag: bri0401_as1
 0.194805  0.109244  0.093201  0.000000  0.035141  0.001528  ,
 1.000000  0.000000  0.000000  0.000000  0.000000  0.000000  ;

tag: bri0402_as1
 0.146677  0.103896  0.157372  0.000000  0.045073  0.005348  ,
 1.000000  0.000000  0.000000  0.000000  0.000000  0.000000  ;
```



```
tag: bri0429_nml
 0.075630  0.092437  0.258976  0.003820  0.018335  0.009931  ,
 0.000000  0.000000  0.000000  0.000000  0.000000  1.000000  ;

tag: bri0432_as1
 0.196333  0.122231  0.207028  0.022154  0.015279  0.003820  ,
 1.000000  0.000000  0.000000  0.000000  0.000000  0.000000  ;

tag: bri0433_as1
 0.306341  0.161192  0.143621  0.037433  0.042781  0.000764  ,
 1.000000  0.000000  0.000000  0.000000  0.000000  0.000000  ;

tag: bri0434_as1
 0.207028  0.115355  0.198625  0.009931  0.019099  0.016043  ,
 1.000000  0.000000  0.000000  0.000000  0.000000  0.000000  ;

tag: bri0461_nml
 0.056532  0.064935  0.307869  0.000000  0.016807  0.008403  ,
 0.000000  0.000000  0.000000  0.000000  0.000000  1.000000  ;

tag: bri0465_as1
 0.249809  0.137510  0.154316  0.032086  0.046600  0.007639  ,
 1.000000  0.000000  0.000000  0.000000  0.000000  0.000000  ;

tag: bri0500_nml
 0.053476  0.071811  0.329259  0.011459  0.009931  0.013751  ,
 0.000000  0.000000  0.000000  0.000000  0.000000  1.000000  ;

tag: bri0565_nml
 0.061115  0.052712  0.272727  0.021390  0.017571  0.006112  ,
 0.000000  0.000000  0.000000  0.000000  0.000000  1.000000  ;

tag: cov0332_men
 0.134454  0.047364  0.051184  0.061879  0.071811  0.012987  ,
 0.000000  0.000000  0.000000  0.000000  1.000000  0.000000  ;

tag: cov0333_men
 0.074102  0.045837  0.040489  0.052712  0.041253  0.002292  ,
 0.000000  0.000000  0.000000  0.000000  1.000000  0.000000  ;

tag: cov0364_men
 0.112299  0.049656  0.029794  0.107716  0.094729  0.010695  ,
 0.000000  0.000000  0.000000  0.000000  1.000000  0.000000  ;

tag: cov0365_men
 0.084798  0.038197  0.046600  0.009931  0.008403  0.009167  ,
 0.000000  0.000000  0.000000  0.000000  1.000000  0.000000  ;

tag: cov0373_nml
 0.108480  0.080214  0.284186  0.002292  0.006875  0.012223  ,
 0.000000  0.000000  0.000000  0.000000  0.000000  1.000000  ;

tag: cov0396_men
 0.126814  0.051948  0.030558  0.095493  0.059587  0.009931  ,
 0.000000  0.000000  0.000000  0.000000  1.000000  0.000000  ;

tag: cov0397_men
 0.084034  0.051184  0.043545  0.035141  0.044309  0.012223  ,
 0.000000  0.000000  0.000000  0.000000  1.000000  0.000000  ;

tag: cov0405_nml
 0.114591  0.106188  0.237586  0.008403  0.015279  0.006112  ,
 0.000000  0.000000  0.000000  0.000000  0.000000  1.000000  ;

tag: cov0428_men
 0.132162  0.031322  0.040489  0.109244  0.080214  0.011459  ,
 0.000000  0.000000  0.000000  0.000000  1.000000  0.000000  ;

tag: cov0429_men
 0.101604  0.054240  0.040489  0.071811  0.067991  0.000000  ,
 0.000000  0.000000  0.000000  0.000000  1.000000  0.000000  ;
```

```
tag: cov0437_nml
 0.098549  0.055768  0.212376  0.012223  0.012223  0.019862  ,
 0.000000  0.000000  0.000000  0.000000  0.000000  1.000000  ;

tag: cov0469_nml
 0.094729  0.063407  0.181818  0.005348  0.009931  0.010695  ,
 0.000000  0.000000  0.000000  0.000000  0.000000  1.000000  ;

tag: ful0405_nml
 0.108480  0.052712  0.330787  0.006112  0.012987  0.008403  ,
 0.000000  0.000000  0.000000  0.000000  0.000000  1.000000  ;

tag: ful0437_nml
 0.106188  0.042781  0.383499  0.006875  0.014515  0.012987  ,
 0.000000  0.000000  0.000000  0.000000  0.000000  1.000000  ;

tag: ful0461_nml
 0.126814  0.109244  0.434683  0.004584  0.010695  0.016043  ,
 0.000000  0.000000  0.000000  0.000000  0.000000  1.000000  ;

tag: ful0469_nml
 0.087089  0.061879  0.411765  0.001528  0.019099  0.017571  ,
 0.000000  0.000000  0.000000  0.000000  0.000000  1.000000  ;

tag: ful0500_nml
 0.114591  0.074102  0.371276  0.009167  0.024446  0.004584  ,
 0.000000  0.000000  0.000000  0.000000  0.000000  1.000000  ;

tag: ful0501_nml
 0.100076  0.061879  0.407945  0.009167  0.019862  0.006112  ,
 0.000000  0.000000  0.000000  0.000000  0.000000  1.000000  ;

tag: ful0525_nml
 0.113827  0.098549  0.394958  0.009167  0.012987  0.009167  ,
 0.000000  0.000000  0.000000  0.000000  0.000000  1.000000  ;

tag: ful0533_nml
 0.135218  0.073338  0.403361  0.011459  0.020626  0.007639  ,
 0.000000  0.000000  0.000000  0.000000  0.000000  1.000000  ;

tag: ful0556_nml
 0.084798  0.085561  0.320856  0.002292  0.011459  0.006112  ,
 0.000000  0.000000  0.000000  0.000000  0.000000  1.000000  ;

tag: ful0557_nml
 0.098549  0.092437  0.339954  0.004584  0.011459  0.007639  ,
 0.000000  0.000000  0.000000  0.000000  0.000000  1.000000  ;

tag: ful0565_nml
 0.127578  0.071047  0.373568  0.013751  0.009167  0.008403  ,
 0.000000  0.000000  0.000000  0.000000  0.000000  1.000000  ;

tag: ful0588_nml
 0.103132  0.093965  0.330023  0.000000  0.012223  0.006112  ,
 0.000000  0.000000  0.000000  0.000000  0.000000  1.000000  ;

tag: mym0372_nml
 0.139037  0.080978  0.262796  0.005348  0.000764  0.007639  ,
 0.000000  0.000000  0.000000  0.000000  0.000000  1.000000  ;

tag: mym0405_nml
 0.097021  0.061879  0.281131  0.012987  0.006875  0.006112  ,
 0.000000  0.000000  0.000000  0.000000  0.000000  1.000000  ;

tag: mym0493_nml
 0.098549  0.090145  0.292590  0.019862  0.009931  0.005348  ,
 0.000000  0.000000  0.000000  0.000000  0.000000  1.000000  ;

tag: mym0498_met
 0.193277  0.071811  0.077158  0.106188  0.425516  0.071047  ,
 0.000000  0.000000  0.000000  1.000000  0.000000  0.000000  ;

tag: mym0499_met
```



```
 0.152788  0.057296  0.049656  0.092437  0.658518  0.130634  ,
 0.000000  0.000000  0.000000  1.000000  0.000000  0.000000  ;

tag: mym0532_met
 0.126814  0.055004  0.068755  0.065699  0.746371  0.173415  ,
 0.000000  0.000000  0.000000  1.000000  0.000000  0.000000  ;

tag: mym0557_nml
 0.100076  0.087089  0.287242  0.003056  0.031322  0.008403  ,
 0.000000  0.000000  0.000000  0.000000  0.000000  1.000000  ;

tag: mym0558_nml
 0.106188  0.071047  0.261268  0.029030  0.056532  0.012987  ,
 0.000000  0.000000  0.000000  0.000000  0.000000  1.000000  ;

tag: mym0563_met
 0.120703  0.046600  0.038197  0.025210  0.919022  0.236058  ,
 0.000000  0.000000  0.000000  1.000000  0.000000  0.000000  ;

tag: mym0564_met
 0.127578  0.043545  0.060351  0.035141  0.792208  0.197097  ,
 0.000000  0.000000  0.000000  1.000000  0.000000  0.000000  ;

tag: mym0590_nml
 0.097785  0.067227  0.267380  0.013751  0.029794  0.005348  ,
 0.000000  0.000000  0.000000  0.000000  0.000000  1.000000  ;

tag: pfe0271_men
 0.142857  0.031322  0.012987  0.014515  0.011459  0.009167  ,
 0.000000  0.000000  0.000000  0.000000  1.000000  0.000000  ;

tag: pfe0272_men
 0.137510  0.038197  0.006112  0.002292  0.002292  0.011459  ,
 0.000000  0.000000  0.000000  0.000000  1.000000  0.000000  ;

tag: pfe0273_men
 0.077922  0.080214  0.011459  0.035141  0.038197  0.005348  ,
 0.000000  0.000000  0.000000  0.000000  1.000000  0.000000  ;

tag: pfe0303_men
 0.137510  0.037433  0.023682  0.008403  0.007639  0.004584  ,
 0.000000  0.000000  0.000000  0.000000  1.000000  0.000000  ;

tag: pfe0304_men
 0.145149  0.036669  0.014515  0.007639  0.004584  0.010695  ,
 0.000000  0.000000  0.000000  0.000000  1.000000  0.000000  ;

tag: pfe0305_men
 0.134454  0.027502  0.007639  0.009167  0.012223  0.010695  ,
 0.000000  0.000000  0.000000  0.000000  1.000000  0.000000  ;

tag: pfe0306_men
 0.000764  0.000764  0.003056  0.000000  0.001528  0.007639  ,
 0.000000  0.000000  0.000000  0.000000  1.000000  0.000000  ;

tag: pfe0338_men
 0.156608  0.055768  0.037433  0.008403  0.010695  0.000000  ,
 0.000000  0.000000  0.000000  0.000000  1.000000  0.000000  ;

tag: pfe0467_nml
 0.111536  0.081742  0.202445  0.002292  0.019862  0.008403  ,
 0.000000  0.000000  0.000000  0.000000  0.000000  1.000000  ;

tag: pfe0499_nml
 0.077922  0.074102  0.201681  0.005348  0.016807  0.001528  ,
 0.000000  0.000000  0.000000  0.000000  0.000000  1.000000  ;

tag: pfe0532_nml
 0.075630  0.063407  0.262796  0.011459  0.009167  0.009931  ,
 0.000000  0.000000  0.000000  0.000000  0.000000  1.000000  ;

tag: pfe0557_nml
 0.109244  0.090145  0.228419  0.004584  0.019862  0.003820  ,
```

```
 0.000000  0.000000  0.000000  0.000000  0.000000  1.000000  ;

tag: pfe0563_nml
 0.080214  0.071811  0.281895  0.030558  0.022918  0.001528  ,
 0.000000  0.000000  0.000000  0.000000  0.000000  1.000000  ;

tag: pfe0564_nml
 0.091673  0.077158  0.280367  0.008403  0.007639  0.013751  ,
 0.000000  0.000000  0.000000  0.000000  0.000000  1.000000  ;

tag: rav0372_nml
 0.087853  0.075630  0.370512  0.002292  0.005348  0.007639  ,
 0.000000  0.000000  0.000000  0.000000  0.000000  1.000000  ;

tag: rav0373_nml
 0.066463  0.052712  0.387319  0.001528  0.012223  0.021390  ,
 0.000000  0.000000  0.000000  0.000000  0.000000  1.000000  ;

tag: rav0399_as1
 0.184874  0.106952  0.172651  0.020626  0.036669  0.014515  ,
 1.000000  0.000000  0.000000  0.000000  0.000000  0.000000  ;

tag: rav0400_as1
 0.287242  0.126814  0.121467  0.019099  0.032850  0.016043  ,
 1.000000  0.000000  0.000000  0.000000  0.000000  0.000000  ;

tag: rav0404_nml
 0.088617  0.081742  0.414057  0.007639  0.001528  0.007639  ,
 0.000000  0.000000  0.000000  0.000000  0.000000  1.000000  ;

tag: rav0405_nml
 0.067991  0.061879  0.426280  0.017571  0.025974  0.011459  ,
 0.000000  0.000000  0.000000  0.000000  0.000000  1.000000  ;

tag: rav0431_as1
 0.198625  0.120703  0.172651  0.019099  0.040489  0.004584  ,
 1.000000  0.000000  0.000000  0.000000  0.000000  0.000000  ;

tag: rav0432_as1
 0.311688  0.159664  0.108480  0.028266  0.037433  0.016807  ,
 1.000000  0.000000  0.000000  0.000000  0.000000  0.000000  ;

tag: rav0436_nml
 0.096257  0.092437  0.420168  0.013751  0.021390  0.010695  ,
 0.000000  0.000000  0.000000  0.000000  0.000000  1.000000  ;

tag: rav0437_nml
 0.074102  0.093201  0.431627  0.005348  0.007639  0.008403  ,
 0.000000  0.000000  0.000000  0.000000  0.000000  1.000000  ;

tag: rav0463_as1
 0.222307  0.146677  0.179526  0.038961  0.061115  0.009931  ,
 1.000000  0.000000  0.000000  0.000000  0.000000  0.000000  ;

tag: rav0464_as1
 0.311688  0.168831  0.115355  0.025210  0.039725  0.005348  ,
 1.000000  0.000000  0.000000  0.000000  0.000000  0.000000  ;

tag: rav0495_as1
 0.225363  0.158900  0.151261  0.027502  0.050420  0.009931  ,
 1.000000  0.000000  0.000000  0.000000  0.000000  0.000000  ;

tag: rav0526_as1
 0.135218  0.131398  0.193277  0.028266  0.045073  0.017571  ,
 1.000000  0.000000  0.000000  0.000000  0.000000  0.000000  ;

tag: rav0527_as1
 0.205500  0.154316  0.168067  0.028266  0.037433  0.012223  ,
 1.000000  0.000000  0.000000  0.000000  0.000000  0.000000  ;

tag: sav0276_nml
 0.108480  0.064171  0.370512  0.000000  0.022918  0.032086  ,
 0.000000  0.000000  0.000000  0.000000  0.000000  1.000000  ;
```



```
tag: sav0301_as1
 0.121467  0.035905  0.147441  0.067991  0.130634  0.004584  ,
 1.000000  0.000000  0.000000  0.000000  0.000000  0.000000  ;

tag: sav0302_as1
 0.108480  0.047364  0.080214  0.010695  0.233766  0.015279  ,
 1.000000  0.000000  0.000000  0.000000  0.000000  0.000000  ;

tag: sav0308_nml
 0.132926  0.093201  0.413293  0.005348  0.017571  0.014515  ,
 0.000000  0.000000  0.000000  0.000000  0.000000  1.000000  ;

tag: sav0334_as1
 0.155844  0.083270  0.255157  0.064171  0.100076  0.023682  ,
 1.000000  0.000000  0.000000  0.000000  0.000000  0.000000  ;

tag: sav0335_as1
 0.135982  0.053476  0.197861  0.073338  0.116119  0.025974  ,
 1.000000  0.000000  0.000000  0.000000  0.000000  0.000000  ;

tag: sav0372_nml
 0.126814  0.089381  0.443850  0.010695  0.006875  0.031322  ,
 0.000000  0.000000  0.000000  0.000000  0.000000  1.000000  ;

tag: sav0373_nml
 0.065699  0.075630  0.350649  0.022154  0.022154  0.019862  ,
 0.000000  0.000000  0.000000  0.000000  0.000000  1.000000  ;

tag: sav0404_nml
 0.078686  0.059587  0.456837  0.007639  0.029794  0.034377  ,
 0.000000  0.000000  0.000000  0.000000  0.000000  1.000000  ;

tag: sig0372_nml
 0.067991  0.074866  0.249809  0.006875  0.048892  0.002292  ,
 0.000000  0.000000  0.000000  0.000000  0.000000  1.000000  ;

tag: sig0404_nml
 0.086325  0.087089  0.307869  0.026738  0.018335  0.018335  ,
 0.000000  0.000000  0.000000  0.000000  0.000000  1.000000  ;

tag: sig0436_nml
 0.077158  0.077158  0.328495  0.064935  0.028266  0.012987  ,
 0.000000  0.000000  0.000000  0.000000  0.000000  1.000000  ;

tag: sig0469_nml
 0.051948  0.074102  0.312452  0.019099  0.029794  0.004584  ,
 0.000000  0.000000  0.000000  0.000000  0.000000  1.000000  ;

tag: sig0525_gbm
 0.125286  0.056532  0.056532  0.032086  0.295646  0.048128  ,
 0.000000  0.000000  1.000000  0.000000  0.000000  0.000000  ;

tag: sig0526_gbm
 0.115355  0.061879  0.067227  0.109244  0.154316  0.025210  ,
 0.000000  0.000000  1.000000  0.000000  0.000000  0.000000  ;

tag: sig0533_nml
 0.080978  0.068755  0.345302  0.038197  0.032086  0.016807  ,
 0.000000  0.000000  0.000000  0.000000  0.000000  1.000000  ;

tag: sig0556_gbm
 0.125286  0.038197  0.075630  0.045837  0.214668  0.022154  ,
 0.000000  0.000000  1.000000  0.000000  0.000000  0.000000  ;

tag: sig0557_gbm
 0.097785  0.032086  0.038961  0.041253  0.278839  0.051184  ,
 0.000000  0.000000  1.000000  0.000000  0.000000  0.000000  ;

tag: sig0558_gbm
 0.100840  0.023682  0.035141  0.100840  0.244461  0.037433  ,
 0.000000  0.000000  1.000000  0.000000  0.000000  0.000000  ;
```

```
tag: sig0623_gbm
 0.090145  0.053476  0.055768  0.031322  0.246753  0.019099  ,
 0.000000  0.000000  1.000000  0.000000  0.000000  0.000000  ;

tag: vaj0434_gbm
 0.111536  0.063407  0.102368  0.138273  0.271963  0.034377  ,
 0.000000  0.000000  1.000000  0.000000  0.000000  0.000000  ;

tag: vaj0465_gbm
 0.129870  0.087089  0.076394  0.051184  0.342246  0.051948  ,
 0.000000  0.000000  1.000000  0.000000  0.000000  0.000000  ;

tag: vaj0466_gbm
 0.080214  0.038197  0.033613  0.091673  0.680672  0.082506  ,
 0.000000  0.000000  1.000000  0.000000  0.000000  0.000000  ;

tag: vaj0467_gbm
 0.055768  0.029794  0.058824  0.236822  0.340718  0.068755  ,
 0.000000  0.000000  1.000000  0.000000  0.000000  0.000000  ;

tag: vaj0497_gbm
 0.101604  0.053476  0.045837  0.054240  0.563025  0.068755  ,
 0.000000  0.000000  1.000000  0.000000  0.000000  0.000000  ;

tag: vaj0498_gbm
 0.077922  0.042781  0.043545  0.080214  0.676089  0.065699  ,
 0.000000  0.000000  1.000000  0.000000  0.000000  0.000000  ;

tag: vaj0499_gbm
 0.082506  0.043545  0.058824  0.071811  0.439267  0.059587  ,
 0.000000  0.000000  1.000000  0.000000  0.000000  0.000000  ;

tag: vaj0531_gbm
 0.119939  0.051184  0.063407  0.079450  0.471352  0.086325  ,
 0.000000  0.000000  1.000000  0.000000  0.000000  0.000000  ;

tag: vaj0563_gbm
 0.091673  0.039725  0.054240  0.084034  0.549274  0.121467  ,
 0.000000  0.000000  1.000000  0.000000  0.000000  0.000000  ;

tag: vaj0620_nml
 0.085561  0.059587  0.177235  0.003820  0.003820  0.016807  ,
 0.000000  0.000000  0.000000  0.000000  0.000000  1.000000  ;

tag: vaj0654_nml
 0.082506  0.061879  0.216196  0.004584  0.033613  0.006875  ,
 0.000000  0.000000  0.000000  0.000000  0.000000  1.000000  ;

tag: vaj0655_nml
 0.099312  0.079450  0.276547  0.003056  0.013751  0.004584  ,
 0.000000  0.000000  0.000000  0.000000  0.000000  1.000000  ;

tag: vaj0656_nml
 0.104660  0.087089  0.248281  0.000000  0.011459  0.002292  ,
 0.000000  0.000000  0.000000  0.000000  0.000000  1.000000  ;

tag: vaj0657_nml
 0.090909  0.093965  0.218487  0.005348  0.025210  0.006112  ,
 0.000000  0.000000  0.000000  0.000000  0.000000  1.000000  ;

tag: vau0492_nml
 0.091673  0.060351  0.284950  0.010695  0.031322  0.012987  ,
 0.000000  0.000000  0.000000  0.000000  0.000000  1.000000  ;

tag: vau0556_nml
 0.079450  0.060351  0.256684  0.010695  0.016043  0.005348  ,
 0.000000  0.000000  0.000000  0.000000  0.000000  1.000000  ;

tag: vau0557_nml
 0.110772  0.091673  0.330023  0.004584  0.003820  0.015279  ,
 0.000000  0.000000  0.000000  0.000000  0.000000  1.000000  ;

tag: vau0589_nml
```



```
 0.110008  0.096257  0.310160  0.008403  0.015279  0.011459  ,
 0.000000  0.000000  0.000000  0.000000  0.000000  1.000000  ;

tag: vau0628_met
 0.390374  0.085561  0.041253  0.012987  0.776165  0.182582  ,
 0.000000  0.000000  0.000000  1.000000  0.000000  0.000000  ;

tag: vau0654_nml
 0.089381  0.077158  0.272727  0.000000  0.013751  0.015279  ,
 0.000000  0.000000  0.000000  0.000000  0.000000  1.000000  ;

tag: vau0658_met
 0.296409  0.092437  0.047364  0.025210  0.647059  0.220015  ,
 0.000000  0.000000  0.000000  1.000000  0.000000  0.000000  ;

tag: vau0659_met
 0.421696  0.104660  0.035141  0.006112  1.000000  0.320092  ,
 0.000000  0.000000  0.000000  1.000000  0.000000  0.000000  ;

tag: vau0660_met
 0.445378  0.093201  0.037433  0.008403  0.865546  0.227655  ,
 0.000000  0.000000  0.000000  1.000000  0.000000  0.000000  ;

tag: vau0687_nml
 0.081742  0.072574  0.249809  0.019099  0.040489  0.006112  ,
 0.000000  0.000000  0.000000  0.000000  0.000000  1.000000  ;
```

```
# '$Id: tm.in, v1.0, 94/06/02 Jane EXP $'
tag: for0372_nml
 0.067227  0.074102  0.216960  0.007639  0.012223  0.005348  ,
 0.000000  0.000000  0.000000  0.000000  0.000000  1.000000  ;

tag: for0404_nml
 0.058824  0.066463  0.233002  0.006875  0.014515  0.011459  ,
 0.000000  0.000000  0.000000  0.000000  0.000000  1.000000  ;

tag: for0405_nml
 0.075630  0.074102  0.209320  0.016807  0.004584  0.017571  ,
 0.000000  0.000000  0.000000  0.000000  0.000000  1.000000  ;

tag: for0436_nml
 0.071811  0.074866  0.265852  0.027502  0.046600  0.019862  ,
 0.000000  0.000000  0.000000  0.000000  0.000000  1.000000  ;

tag: for0437_nml
 0.074102  0.091673  0.252101  0.006112  0.006875  0.012223  ,
 0.000000  0.000000  0.000000  0.000000  0.000000  1.000000  ;

tag: for0493_gbm
 0.103132  0.036669  0.103132  0.012223  0.135982  0.020626  ,
 0.000000  0.000000  1.000000  0.000000  0.000000  0.000000  ;

tag: for0494_gbm
 0.110772  0.049656  0.042017  0.011459  0.235294  0.020626  ,
 0.000000  0.000000  1.000000  0.000000  0.000000  0.000000  ;

tag: for0495_gbm
 0.116883  0.025974  0.068755  0.023682  0.315508  0.042781  ,
 0.000000  0.000000  1.000000  0.000000  0.000000  0.000000  ;

tag: for0496_gbm
 0.131398  0.022918  0.048892  0.023682  0.422460  0.065699  ,
 0.000000  0.000000  1.000000  0.000000  0.000000  0.000000  ;

tag: for0497_gbm
 0.154316  0.022918  0.060351  0.053476  0.352177  0.058824  ,
 0.000000  0.000000  1.000000  0.000000  0.000000  0.000000  ;

tag: for0498_gbm
 0.093201  0.077922  0.055004  0.018335  0.374332  0.027502  ,
 0.000000  0.000000  1.000000  0.000000  0.000000  0.000000  ;

tag: for0526_gbm
 0.157372  0.006112  0.041253  0.032086  0.354469  0.039725  ,
 0.000000  0.000000  1.000000  0.000000  0.000000  0.000000  ;

tag: for0530_gbm
 0.154316  0.042017  0.059587  0.031322  0.276547  0.018335  ,
 0.000000  0.000000  1.000000  0.000000  0.000000  0.000000  ;

tag: for0558_gbm
 0.222307  0.068755  0.058060  0.062643  0.242934  0.018335  ,
 0.000000  0.000000  1.000000  0.000000  0.000000  0.000000  ;

tag: for0559_gbm
 0.261268  0.098549  0.107716  0.056532  0.156608  0.017571  ,
 0.000000  0.000000  1.000000  0.000000  0.000000  0.000000  ;
```

Name	Succ	Tot	Fail	EXBD	Fail_Sample	Failure_Output%						Diag
as1	2	3	1									
				0.3	ts_AVEarea.sav::savTumr_as1	00	00	82	02	00	17	gbm
as2	0	1	1									
				0.3	ts_AVEarea.bis::bisTumr_as2	54	01	13	00	32	00	as1
gbm	0	3	3									
				0.3	ts_AVEarea.for::forTumr_gbm	14	31	55	00	00	00	gbm
				0.3	ts_AVEarea.sig::sigTumr_gbm	00	24	73	00	03	00	gbm
				0.3	ts_AVEarea.vaj::vajTumr_gbm	00	00	53	47	00	00	gbm
men	1	2	1									
				0.3	ts_AVEarea.cov::covTumr_men	00	38	08	00	54	00	men
met	1	2	1									
				0.3	ts_AVEarea.mym::mymTumr_met	00	02	36	63	00	00	met
nml	12	12	0									

Name	Succ	Tot	Fail	EXBD	Fail_Sample	Failure_Output%	Diag
as1	2 /	3	1				
				0.3	ts_AVEpeak.sav::savTumr_as1	08 00 04 00 00 88	nml
as2	0 /	1	1				
				0.3	ts_AVEpeak.bis::bisTumr_as2	06 01 06 00 86 00	men
gbm	1 /	3	2				
				0.3	ts_AVEpeak.sig::sigTumr_gbm	00 01 42 00 57 00	men
				0.3	ts_AVEpeak.vaj::vajTumr_gbm	00 00 45 54 01 00	met
men	1 /	2	1				
				0.3	ts_AVEpeak.cov::covTumr_men	00 40 47 00 13 00	gbm
met	1 /	2	1				
				0.3	ts_AVEpeak.mym::mymTumr_met	00 00 75 01 00 24	gbm
nml	12 /	12	0				

Name	Succ Tot	Fail	EXBD	Fail_Sample	Failure_Output%	Diag
as1	18 / 19	1				
			0.3	tr_LSTarea.for::sav0302_as1	00 32 68 00 00 00	gbm
as2	8 / 8	0				
gbm	23 / 25	2				
			0.3	ts_LSTarea.for::for0493_gbm	12 88 00 00 00 00	as2
			0.3	ts_LSTarea.for::for0559_gbm	93 00 00 07 00 00	as1
men	16 / 16	0				
met	9 / 9	0				
nml	72 / 72	0				