Universita´ degli studi di Roma Tor Vergata
Istituto Nazionale di Fisica Nucleare
“**Laboratori Nazionale di Frascati**”

Masters degree Thesis on

DISPLAYING RAP EXPERIMENT GRAPHS USING JPGRAPH

Prepared by:-

Ahmed Omar Jebril

supervised by:-

Giovanni Mazzitelli

October 2003

To my Dear parents

Acknowledgements

I would like to express my deep appreciaton and sincere thanks to my course director Prof. Maria Ferrer, and thesis supervisors Giovanni Mazzitelli and Paolo Valente for their kind guidance and valuable suggestions which helped me so much to finalize this work.

I am also indebted to Fabio Fotugno, Paolo Santangelo, Igor Sfiligoi, Dael Maselli, for their help, and to all tutors for their efforts to realize the objectives of this Masters course.

Finally, I would like to thank my family and friends for their encouragement and support.

Table of contents

Abstract

PHP or "*Hypertext Preprocessor*" is one of the most powerful programming languages used for building dynamic, interactive web sites. This is due to many reasons such as its simplicity which makes it easy to learn, and security which permits it to be put on web servers as CGI "Common Gateway Interface" source codes. Using CGI, only the HTML "Hypertext Markup Language" part is shown on the browser protecting the original source code from being exposed on the web, thus possibly seriously risking the loss of the code´s ownership and related security issues. Moreover, PHP nowdays is supported by most of the ISPs "Internet Service Providers" around the world.

An important tool based on PHP programming language, is JpGraph. It is a class library that easily enables the generation of professional-quality graphs using minimal amount of code.

Generally, in developing PHP scripts to generate graphs using JpGraph, it is useful to take into consideration the main steps shown below:

1. Retrieve and manipulation of the data for plotting.
2. Creation of the Graph-object and setting the graph´s general properties (such as dimensions, colors, and plot axes).
3. Creation of the plots and adding them to the Graph object.
4. Finalization of the Graph-object and displaying the graph.

In this document, we will introduce a JpGraph interface that displays the plots of data collected by the "**RAP experiment"** at the *INFN*´s "***National Laboratories of Frascati***".The aim of this experiment is to measure the effects caused by high energy electrons striking bulk superconductors.The resulting effects were observed as mechanical vibrations on the resonant GW "*Gravitationanl Wave*" detector called NAUTILUS which is a 2300Kg Aluminum alloy cylinder operating at a temperature of 100 mK^o.

Given that NAUTILUS is the most sensitive operating GW detector in the world, very high energy cosmic rays can induce false events that can't be distinguished from GWs. For this reason, it was equipped with a cosmic ray telescope . The rareness of events due to cosmic rays was found unpredictably correlated to the thermodynamic temperature of the apparatus.

Another smaller 35Kg detector (made of the same alloy) was installed to measure the same effects caused by an electron beam generated from BTF *"Beam Test Facility*" at the Laboratories DAFNE accelerator. The RAP experiment -as well as the DAFNE accelerator- data are saved on the Database server as text files with different extensions (this permits to distinguish between different data types).

This interface model retrieves the data from its sources, manipulates them and then displays them on-line via internet according to the graph´s layout characteristics chosen from an "*Enquiry Form*" which provides different options and works as a mediation engine between the user and the graph´s data source.The date of experiment is also displayed on the graph (History or scatter plots) after extracting it from the file's name. In each plot, it is possible to display either one or two plots on the same graph -upon choice- on one of three different scale types (Linear, Semi-Log, or Log). This can be a good instrument in case of comparing three different data types realted to one another at the same time under the same conditions. The model was tested on the DAFNE accelerator database structure that has the same structure as the RAP experiment data and provided satisfactory data plots.

CHAPTER (1)

“Introduction”

Chapter (1): Introduction

PHP stands for Hypertext Pre-processor. It is a scripting language used for different applications on the web. Since it's a preprocessor, it runs on the remote web server and processes the webpages before they are sent to the browser. This makes it a so-called server-side scripting language. Thus, it can be used as a *CGI* (Client Gateway Interface).

Because PHP is meant to be used with webpages, it has a lot of functions to deal with text. PHP is specially designed to deal with webpages so quickly and efficiently where most of its built-in functions are simple and straightforward to use.

Being web-oriented, PHP also contains all the functions needed to do things on the Internet. There are functions for connecting to remote webservers, checking mail via POP3 or IMAP, or URL encoding strings to protect special characters.

PHP is also powerful in creating images and graphs to be displayed on the web. This can be done using the built-in functions that handle image generation using the GD-graphics library.

There are a few different engines, which can run on a server and generate images. Among these engines, the one called *JpGraph*.

JpGraph is a PHP class library implemented in fully Object Oriented basis that easily enables to generate professional quality graphs using a minimal amount of code.

The library supports a large variety of plots to which makes it suitable for providing scientific graphs. Both linear and logarithmic plot scales are supported, in addition to all standard plot types (line, bar, pie, 3D-pie, error, radar, scatter, etc...)

JpGraph also supports the generation of Gantt-charts and supports the use of *CSIM* (Client Side Image Maps) for defining hotspots within the graphs.

JpGraph has also a built-in cache system to reduce load of the server, anti-aliasing of lines and full support for both bit mapped and TTF fonts.

As an image generating engine, we will use a JpGraph model under PHP to display the graphs describing the relationships between the different acquired data from the RAP experiment located at ***LNF***.

RAP (*Rivelazione Acustica di Particelle*) is an italian term that stands for the *Acoustic Detection of Particles in Ultracryogenic Resonant Antenna*.The aim of this experiment is to measure the effects caused by high energy electrons striking bulk superconductors.

The motivation is the anomalous high energy cosmic ray rate detected by the gravitational wave antenna "***NAUTILUS***" when operated in superconducting conditions.

In this document, we will first introduce our model design and then, the way with which the selected graph plots are displayed.

In ***Chapter (2)***, we briefly talk about the RAP experiment with a bit of scientific background, the methods on which the data acquisition is based on and how and where the acquired data are saved.

In ***Chapter (3)***, we describe our model in details and we will explain the written scripting code line by line.

In the last chapter, we discuss the generated graphs, the advantages and disadvantages of using this PHP based model presenting the conclusions and recommendations for future code modifications or further improvements.

Finally, two apendices are included; the first ***(Appendix A)*** contains the various data format saved at the DAFNE Web Server data access facility. The second ***(AppendixB)*** contains the whole PHP code written for this interface model.

CHAPTER (2)

“The RAP Experiment”

A TECHNICAL BACKGROUND

Chapter (2): The RAP Experiment - A Technical Background

(2.1) Overview:-

The RAP experiment located at *Frascati National Laboratories* (LNF) aims to investigate the anomalous effect of *Cosmic Rays* (*CR*) on the GW detector NAUTILUS. For this purpose, another similar but smaller 35kg cylinder cooled at 100mK^o *GW* detector is used [4][13].
The first attempt to study this phenomenon was in 1969, when Beron and Hofstadter [1] [4] carried out their experiments aiming to detect oscillations of piezoelectric disks excited by a GeV electron beam. The results led the authors to suggest that a very large *CR* event could excite mechanical vibrations in a metallic cylinder at its resonance frequency and could provide an accidental background for experiments on GWs [4].
Later on, a group at the University of Milan estimated the possible effects of striking particles on a small aluminum cylinder and made an experiment that verified the calculations, even though with rather large experimental errors.
The idea is based on the mechanical vibrations that originate from a local thermal expansion caused by warming up due to the energy lost by the particles crossing the material. The effect depends on the thermal expansion coefficient and the specific heat of the material. The ratio of these two quantities is referred to as the "*Grüneisen Coefficient*". It turns out that while both the *expansion coefficient* and the *specific heat* vary with temperature, the *Grüneisen coefficient* practically does not. In the case of aluminum, this is certainly true above 1K^o, but no data are available at lower temperatures when the aluminum becomes a superconductor.
Subsequently, several authors [4] made calculations that found to be more refined. All these models agree in predicting, for the vibrational energy (*E)* of the excited fundamental mode of an aluminum cylindrical bar like NAUTILUS, according to the following formula:

$$E = 7.64 \cdot 10^{-9} \, (W^2 f)$$

Where *(E)* is expressed in Kelvin, *(W)* is the energy released by the particle to the bar expressed in GeV, and (*f)* is a geometrical factor of the order of unity. The above formula was verified with an experiment at room temperature [11], using a small aluminum cylinder and an electron beam.

NAUTILUS has been operating in continuous data taking at the *INFN Frascati Laboratory* for five years obtaining the highest sensitivity of any GW detector in the world.

Given that NAUTILUS is equipped with a *CR* detector [1] [4], the data regarding the vibrational energy of the bar were correlated with all data obtained by this *CR* detector. Very large signals, at a rate much greater than expected were detected [4].

In order to investigate this matter, and account for these anomalous results, the particles are measured in a low temperature mechanical oscillator in a controlled environment. This is done using the DAFNE electron *Beam Test Facility* (BTF) at LNF to excite mechanical vibrations in a 35Kg small cylindrical bar made of the same aluminum alloy as NAUTILUS.

The measurements are performed in both superconducting and normal regime of the aluminum bar. The results of this experiment are important to understand the interactions of ionizing particles with bulk superconductors and to understand the limitations caused by *CR*s to the sensitivity of future gravitational resonant detectors not shielded against them.

(2.2) The RAP Experiment

Beron, B.L. and Hofstadter, R., in their pioneering work [1] [4], reported on the observation of radial and compressional modes of vibration of a set of piezoelectric disks impinged by electron beams of 0.2-1.0 (GeV) energy. They also explicitly mentioned that a very large (CR) event could excite mechanical vibrations in a metallic cylinder at its resonant frequencies and could provide an accidental background for experiments on *GW*s.

Grassi Strini, A.M. et al. [1] [4] suggested that the results obtained in the above experiment could not be considered comprehensive. They referred their assumption because a set-up with no calibration capacities was used, and that the interpretation of the results was merely qualitative; and that the observations could have been affected by the energy spread in the absorber caused by the showering electrons.

In their experiment, they used a cylindrical bar, 20cm in length and 3cm in diameter, made of an aluminum alloy and suspended in a vacuum chamber by a steel wire. Two piezoelectric transducers, one placed in correspondence to the central cross section of the bar and the other displaced with respect to that section, provided the readout of the oscillations, also enabling even harmonics detection. The whole system was calibrated by exciting the bar with an oscillating electrical force. The bar was exposed either to a 29.3 MeV pulsed proton beam with a peak intensity of 5*μ*A and a time duration ranging from 10 to 50*μ*s or to a 500eV pulsed electron beam with currents of several mA's and a time duration between 10 and 50μs.

The maximum oscillation amplitude of the first, second and third harmonics as functions of the energy delivered in the beams was experimentally observable, with a quoted total error of ±20%. Comparison with the predictions of a simplified model was performed.

Assuming that:

(1) All the beam energy is transformed into thermal energy.

(2) The bar could be described as a one-dimensional elastic system.

(3) The energy of the beam spreads out uniformly on the whole cross section of the bar.

(4) The beam pulse duration is short with respect to the oscillation period of the highest order harmonics).

The agreement was found very satisfactory when using a proton beam. A less satisfactory agreement found for the electron beams was referred to the limitations of the theoretical model adopted, and in particular to the un-fulfillment of the third hypothesis. Since the electron energy was too low to allow for penetration inside the bar material, leaving the localization of the electron energy loss in the regions near the surface and, thus, not allowing the use of a one-dimensional model (second assumption) as a good approximation.

Verification of the Thermo-Acoustic model was also attempted by Bressi G., et al. [4] using a suspended cylindrical test mass, having a length of 20cm and diameter of 3cm, made of an aluminum alloy with an innovative transducer. It was used to detect the change of the tunneling current in case of a variation of the gap between a tip and the surface of the macroscopic body to be monitored. Various calibrated set-ups were exposed to proton beams in order to detect the effect.

The first attempt was made at LNL with a 5MeV proton beam with a negative observational result, which was described to un-fulfillment of the third assumption in the above-indicated model, since the range of a 5MeV proton is 0.18mm. The second attempt was made at the LNL Tandem by using a proton beam of 30MeV energy, in order to insure the uniformity of the energy release in the bar, being 4mm the proton range at this energy. The negative observational result was related to the time duration of the proton pulse, not optimally matching the fourth assumption of the model.

The last attempt was made at "CERN" using "LEAR" delivering 50MeV protons with pulse time duration shorter than "LNL" Tandem. This set-up was subject to an irrecoverable environmental seismic noise, with effect also in the tunneling current and electrical induction on the sensors. The dismissing of “LEAR" made any improvement of the experimental conditions impossible.

Later experiments[1][4][12] confirmed the applicability of the model by using different setup exposed to a ≈ 600 (MeV) electron beam operating in single-bunch mode with a pulse width of up ≈ 2μs and adjustable intensity of $10^9 - 10^{10}$ electrons per bunch, provided at the *Amsterdam linear accelerator, “MEA”*.

One setup consisted in a cylindrical bar, made of an aluminum alloy and of 20cm length and 3.5cm diameter. The bar was suspended by a plastic string connected to a horizontally movable gliding structure, allowing for displacement of the bar with respect to the beam spot, inside a vacuum chamber. Different arrangements of sensors and calibrators were used in order to verify the feasibility and then to perform the measurement. In the latter case, a piezoelectric sensor was placed 1cm off center on top of the bar, one end of the bar was equipped with an accelerometer, the other end facing a capacitor plate. The calibration of the piezoelectric sensor was performed against the accelerometer, by using the excitation provided by a loudspeaker, and the stability of the piezoelectric device was monitored during the runs by electrically driving the capacitor plate.

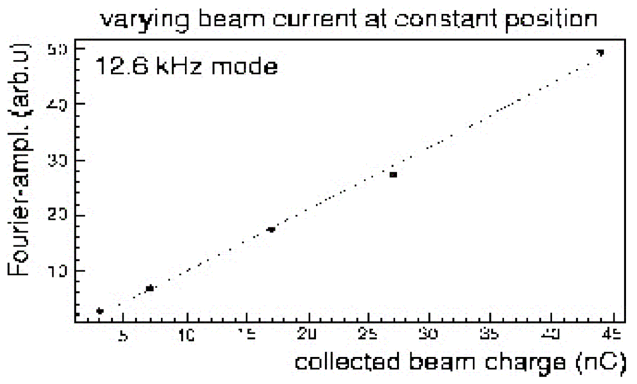


Figure (2.1): Correlation between the Fourier amplitude of the 12.6 kHz vibrational mode and the beam charge. Data points (*) and straight line fit (-).

Calculations using the *Thermo-Acoustic model* [4] agreed well with the data as demonstrated by the variations of the excitation strength with the absorbed energy inferred from the linear dependence between the Fourier amplitude of the modes on the integrated charge in the beam pulse at a fixed beam position (Fig. 2.1). By comparing the amplitudes with the model predictions as a function of the beam hit positions along the cylinder. A conversion factor of 7.4± 1.4 (nm/J) was found for the first longitudinal mode of the resonator, while the prediction of the model is 10 (nm/J).

RAP experiment benefits from a high intensity beam performed by DAFNE's *Beam Test Facility* (BTF). This is done in order to prepare NAUTILUS in various ranges of temperature, particularly when the Aluminum is in its superconducting state and the reception of signals is expected.

(2.3) DAFNE, RAP, and KLOE experiments joint data structure

In order to correlate the RAP experiment data with the rest of the accelerator's data, the same file system and logging structure of the DAFNE accelerator was chosen [5].This is similar to what was done to the KLOE experiment data.

Data in the DAFNE collider are stored by the control system [4] in the local memories of the 45 front-end VME-CPU's distributed all over the accelerator area. The front-end tasks get commands from the high-level user environment and continually update their own database with information from the devices. Data are available through direct memory access to the CPU's memory. This front-end database is constituted by different data types tailored to specific machine elements (non-homogenous database); this means that in order to use this data or to correlate parameters of different devices, specific routines must be implemented.

Two system tasks have been developed in order to collect all the parameters: the *Dumper*, to synchronize and align the data, and the *Storer* to store them to disk [4]. The Dumper continuously fetches data from the front-end memories and writes the different data-types from each machine element aligning them in a homogenous database. This memory resident database is accessible from high-level tasks:

- For monitoring purposes, such as the watchdog process checking for faulty magnets, bad vacuum or CPU failures;
- For the user interface display (from any operator console);
- For online correlation and analysis.

The Storer then synchronizes the memory database and writes it on the mass storage at given time intervals.

On the other hand, the Dumper continuously reads the status of the satellite systems not belonging to the control system environment (such as the spectrum analyzer), merging this data in the homogenous database. In the general scheme of DAFNE controls, sketched in the right part of Fig. (2.3), the main processes and data flow are reported. In addition to the control and monitoring functions and the communication with the KLOE slow control, an other process serves DAFNE data to the controls of the synchrotron radiation facility.

The handling of data related to the DEAR experiment (running at the second DAFNE interaction region) is not shown in the figure, since it is monitored as any other part of the accelerator, and its data can be presented through the DAFNE interface.

As described earlier, the DAFNE storing scheme [5] has the STORER process, running under the DAFNE Control System which writes a set of status and history files on the DAFNE Web server local disks.

While history files are logged on as part of the RAP data and accelerator, the resulting data are logged in binary file formats, especially those files having very large amount of data collected from Fast readout systems. The variables of the devices are logged every 5 and 15 seconds as plain text.

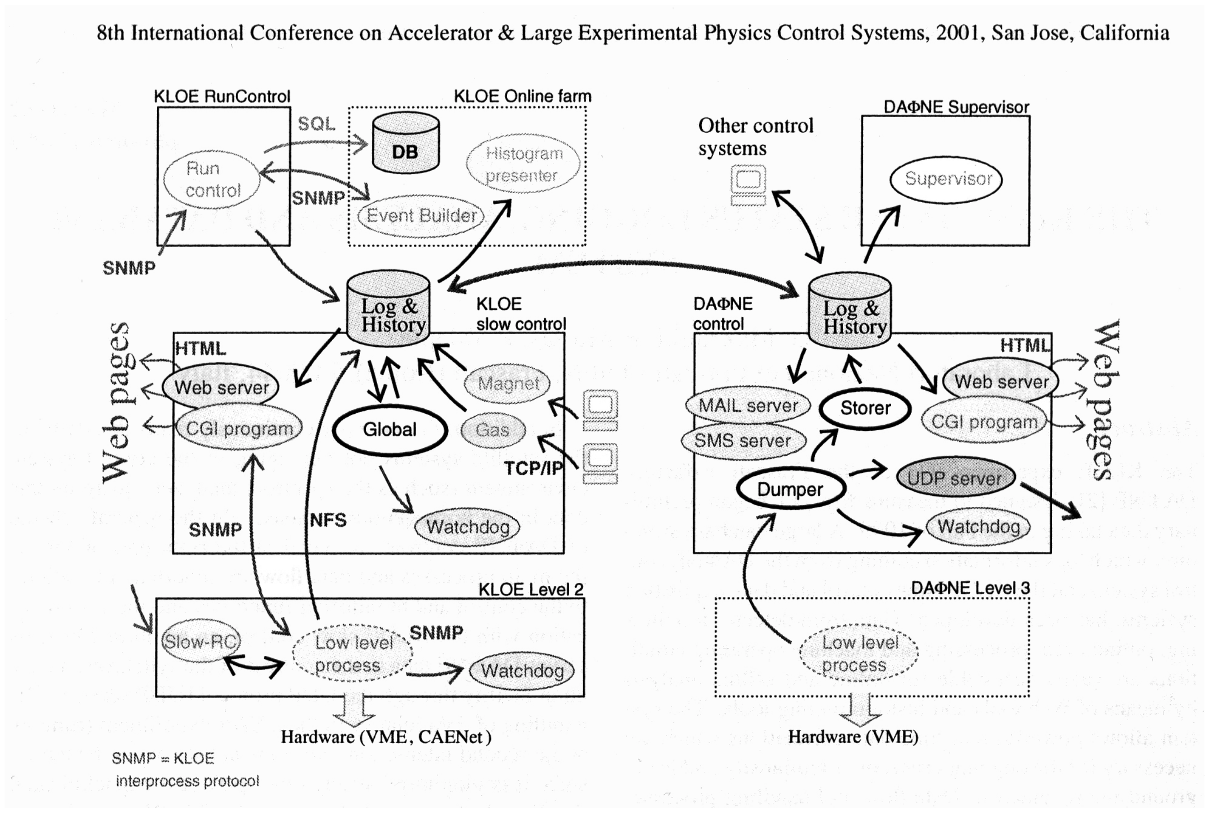


Fig. (2.3): Layout of the KLOE/DAFNE controls and of the joint logging and Database system

(2.4) Data acquisition in RAP experiment

The data acquisition system was developed in the LABVIEW environment with low-level C calls. It is based on a 200 Ksample/s peak sensing 16 bit VME ADC (VMIC 3123) and a VME Pentium III CPU (VMIC 7740) running Linux. This system collects data coming from the PZT (a commercial Piezoelectric Ceramic), the accelerometer, the environmental sensors, and from the beam signals originated by the upstream beam monitor detector and by downstream electromagnetic calorimeter that masures the residual energy of the beam products after the interaction with the detector. The overall data throughput on disk is 0.3 MB/s. An online monitor of the measured amplitude which performs real-time fast Fourier transform analysis was also developed.
Large amounts of auxiliary channels, accelerometers, thermometers, and pressometers are also acquired in order to monitor the detector status.

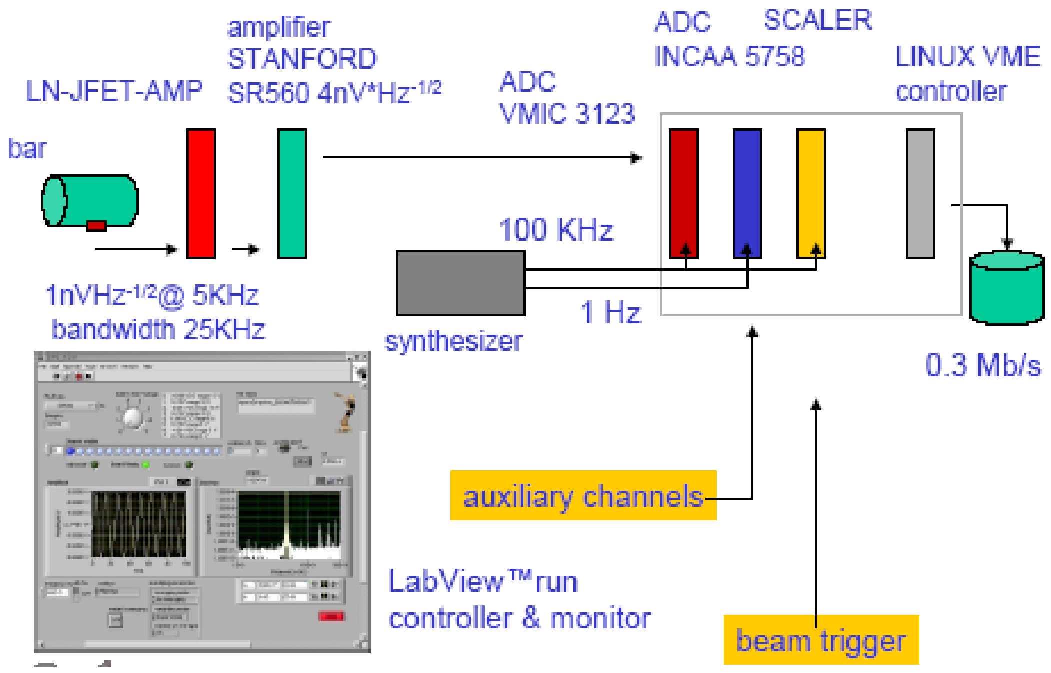


Fig (2.4): RAP experiment data acquisition

CHAPTER (3)

“THE JP graph model”

Chapter (3): The JpGraph model

(3.1) Overview

The Internet is a communication medium. Text and data in tables are both ways to communicate information, but in most cases, a picture is worth a thousand words.

To this end, graphical representation of data (charts or graphs) is often a much more efficient way to communicate than simply presenting the same data in a table. This is particularly true in time-series data, where the user can usually identify variations far more easily in graphical format than by reviewing a series numbers.

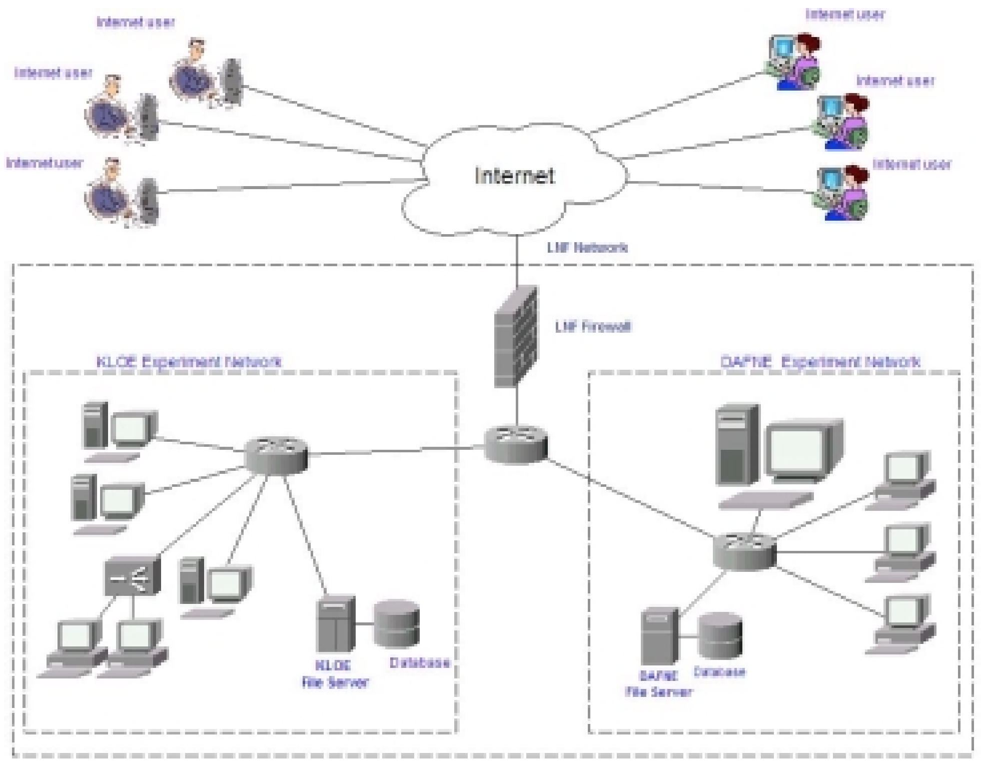


Fig.(3.1): Accessing JpGraph Via Internet

Charts and graphs can be a very powerful means of communication. PHP, compiled with GD support, allows for dynamic image generation. This, combined with PHP's renowned flexibility in retrieving data, gives us the perfect environment to generate charts and graphs dynamically. The JpGraph library of PHP classes is a very useful tool. This set of graphing classes' lets us easily construct professional looking graphs without having to investigate the

GD library calls and their PHP wrapper functions. Having the capability to concentrate on delivering the information and integrating graphs into web applications, instead of building images from low-level graphics primitives.

(3.2) JpGraph supported Graph Type Definitions

The following are the main graph types that can be generated by JpGraph [6] [7]:-

- **GIF** - CompuServe Graphical Interchange Format - image compression format supported by most browsers, limited to 256 colors.
- **JPEG** - Joint Photographic Experts Group (JPG) - image compression format that supports full color (picture quality) images, and enjoys the support of most web browsers. In contrast with GIF or PNG, JPEG is a "lossy" compression algorithm.
- **PNG** - Portable Network Graphics (unofficially: "PNG is Not GIF") - image compression format that is the successor to (in most ways superior to) GIF, and is supported by most modern web browsers.
- **LZW** - Lempel Ziv Welch - the compression algorithm implemented in GIF images, patented by Unisys Corporation (and apparently by IBM 3 weeks earlier, if you are interested in such things)
- **GD** - graphics library, which can be compiled with PHP to allow for image manipulations and generation (Note: From PHP 4.3.0 GD will be bundled with PHP, allowing for ease of configuration.)
- **TTF** - True Type Fonts - fonts based on vector graphics that allow for much better scaling and rotation
- **Time Series Data** - information that can be associated with regular time intervals, as for example; readings per seconds.

(3.3) Prerequisites

To generate graphs with the JpGraph model it is required to:

- Have access to PHP with GD and TTF enabled
- Download and install the JpGraph class and supporting documentation
- Review the JpGraph Quick start and the source code from several of the example charts to have a complete background on the use of the JpGraph classes.

• Import interesting data to graph into the PHP script. (In most real world cases, this data can be retrieved from a database or -like in our interface- from acquisited experimental data text files).

(3.4) How it Works

Starting with version 1.6 [6], JpGraph's author has switched to a QT style license. A brief summary of this license is that the class can be freely used in open source projects, (But for commercial applications, it is necessary to fulifll the licensing terms before using the software).

After downloading the tar ball, it is necessary to unzip and untar the files. Once they were untarred, the entire source should be moved to the PHP library directory to make this graphing capability available to all applications. If of not having access to this directory (in a hosted PHP environment), it can be included in another directory accessible to the application or otherwise copy the class definition files into the user accessible directory.

It might be necessary to make several changes to the used directories paths in the file (jpgraph.php) .These include the definitions of the graph cache directory CACHE_DIR, the server-working directory APACHE_CACHE_DIR and the True Type Font directory TTF_DIR. The CACHE_DIR is a full path, and should be set to a location that is writeable by the web server user. APACHE CACHE the location viewed by the server browser in order to view the results in the CACHE DIR (In our case we used ”/var/www/html/exercises”). TTF_DIR should be set to the location of the TTF files. This means that all of the work done by instantiating a class ***$graph = new graph (...)***, calling the methods that are functions in the class (like ***$graph->setscale ' textlin'***), or changing properties will be saved in that directory.

Being completely Object Oriented, JpGraph permits the structure of code to get much better reuse and makes it easy to correct and modify.

(3.5) PHP Compile Options

To do all this with JpGraph, it is necessary to have several PHP compile options enabled such as the use of True Type Font features ***--with-ttf*** [6]. Once PHP with GD is successfully created, the GD section of the *phpinfo.php* page will look like this:

gd

GD Support	enabled
GD Version	1.6.2 or higher
Free Type Support	enabled
Free Type Linkage	with free type
JPG Support	enabled
PNG Support	enabled
WBMP Support	enabled

Table (3.1): The gd library configuration on PHP

(3.6) Visual Presentation of Data

Having defined the directories, included PHP working with GD, and determined the way to retrieve the data to be plotted, JpGraph provides the ability to easily output the data into charts with a variety of formats (line, bar, pie, spider, scatter or combinations thereof) [6]. For example, given this array of data:

$data = array (7, 10, 9, 11, 54)

It is possible to present this series of data in a variety of formats (code required to generate each graph is included as well). The major graph types that can be implemented are as follows:-

Line graphs:-

Line graphs are good for presenting information that is related, and presented in a sequence. Line graphs are a particularly good choice if you are trying to convey trend information to the user. Examples of this kind of data might include time-series related data like the daily luminosity, the DAFNE current as a function of time, or the amount of acquired data.

Bar Graphs:-

Bar graphs can be used in the same fashion as line graphs to represent a trend. Bar graphs are also useful in representing similar information that is not intended to be in a series. For example, it might be required to present a graph of integrated luminosity per experiment. In this graph, the integrated luminosity could vary significantly from experiment to experiment, and presenting this information as line graph would probably be confusing.However, it should be noted that the special case where presenting non-time series data related to frequency, and are sorted it in descending order is called a Pareto chart. This type of chart is particularly useful to diagnose things like frequency of occurrence of problems.

Pie Charts:-

Pie charts are very useful to convey percentage information, or a sense of relationship between different parts that make up a set of data. Whenever we want like to show how each of the items in a collection of data relates to the others, pie charts are a good means of presentation. For example, the uptime of DAFNE, with the percentage of shutdown, maintenance, data taking periods, etc...

Spider Charts:-

Spider charts (sometimes called radar charts) are used to show the relationships of a number of distinct categories, comparing similar data for each category. An example of this might be comparing luminosity against a plan for a number of luminosity experiments, each luminosity experiment would represent a spoke of the spider's web, and a line would connect to each spoke where the percent of plan for that experiment dictated.

Scatter Charts:-

This type of charts is used to represent data points with two dimensions (an X-axis and a Y-axis component) and we usually expect to draw some conclusions regarding the data by seeing if patterns or groups on the scatter chart. For example, the DAFNE luminosity as a function of electron and positron currents.

(3.7) Use of Memory Caching

JpGraph supports two methods of getting your image to the browser [6]. The first is to stream the image directly from PHP; this is the "*inline*" method. The second is to save the image to disk for later retrieval by the browser as the *src* attribute of an *img* tag; this is the "*cache*" method.

For very dynamic data, which can be retrieved quickly, and the plot would likely be different each time it is requested (for example, poll results), the "inline" method may be preferable. To use this method, it is important to put all of the code to retrieve the data in the graph generating code, and generate the graph into a separate PHP file.

```
<? php
// @fi1e mygraph.php                              '
```

```
$data = get_my_data_somehow() ;
$graph = new graph (200, 100);
//the rest of the graph code
$graph->stroke() ;
?>
```

In the HTML file, referring to <img src=" graph name. php "> to retrieve the image. JpGraph will stream the binary image directly from the PHP file back to the browser, and will take care of sending the correct MIME type header so the browser will know how to correctly render the image. This is a very common use of the JpGraph classes.

If there is data that takes a long time to retrieve, or if the graph is not likely to change over the course of many requests, or if we want to use the same data elsewhere in the page without retrieving it a second time, it may be prefered to use the "cache" method of generating the JpGraph image. To use caching with JpGraph, it is necessary to provide the name for the file to be cached. This can be done as part of the graph Object Creations

```
$graph = new graph (200, 100, ' chart .png’);
```

Where 200 is the graph width, 100 is the height, chart .png is the generated image file saved in the cache directory.

After using the `stroke()` method, it is possible to refer to the image in the `src` attribute of an `<img>` tag. The path to the image cache directory can be set up according to the server-working directory defined while configuring JpGraph.

One advantage of using the cache method is that it is possible to display the graph multiple times from different pages without taxing the system to re-query the data and regenerate the graph image. There are actually five parameters that perform the construction method of the JpGraph graph class:(*image width, image height, cache file name, minutes of cache timeout*, and *the inline streaming flag)*.

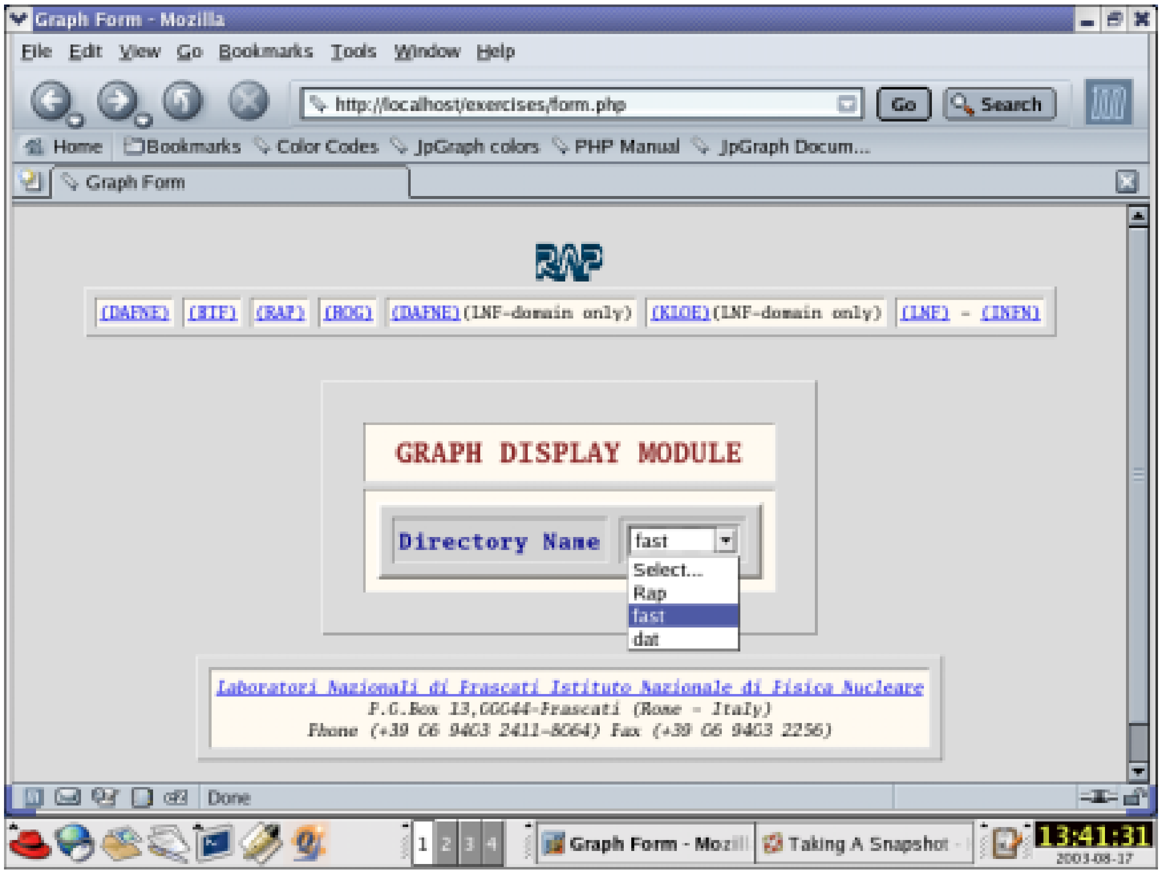
Before writing any scripts, it is required to define the directories where to save them in which they are read and written by the web server. Here the directory containing the PHP scripts is defined as (/PHP). In this directory, the (/jpgraph) directory is created in which the class definition files are located. The CSIM cache directory is defined as (/csimcache) in which the *png* image and the returned HTML map files are saved (useful when displaying mapping points on the graph).

(3.8) The interface model Structure

In this model, two graphs are generated on-line one after the other. The main graph plot is generated right after the enquiry form is accessed,The detailed plot however, is generated from the first one. Both graphs can be either a scatter chart or linear plot with different interval choices. Upon choice, there is a possibility to display one or two Y-axes from this enquiry form. The second plot however, is accessed from the preceding one and has the same plot type. It was designed to show the data in a more detailed manner according to the time start-time and end-time selected.

The enquiry form is a PHP page displayed on the web browser that retrieves the input data variables to be shown on the plot from a special text file named *vars.txt* that located at the same directory where the data are saved. This file contains the names of all variables controlling the data files. Therefore, once the variables are chosen, the graph will select the relative data arrays.

The selection process begins with choosing the directory containing the data files to be displayed from a selection list as shown in fig.(3.1);


Graph Form - Mozilla
File Edit View Go Bookmarks Tools Window Help
http://localhost/exercises/form.php
Go
Search
Home
Bookmarks
Color Codes
JpGraph colors
PHP Manual
JpGraph Docum...
Graph Form
(DAFNE) (BTF) (RAP) (BOG) (DAFNE)(LNF-domain only) (KLOE)(LNF-domain only) (LNF) - (INFN)
Done
Graph Form - Mozill
Taking A Snapshot
13:41:31
2003-08-17

GRAPH DISPLAY MODULE

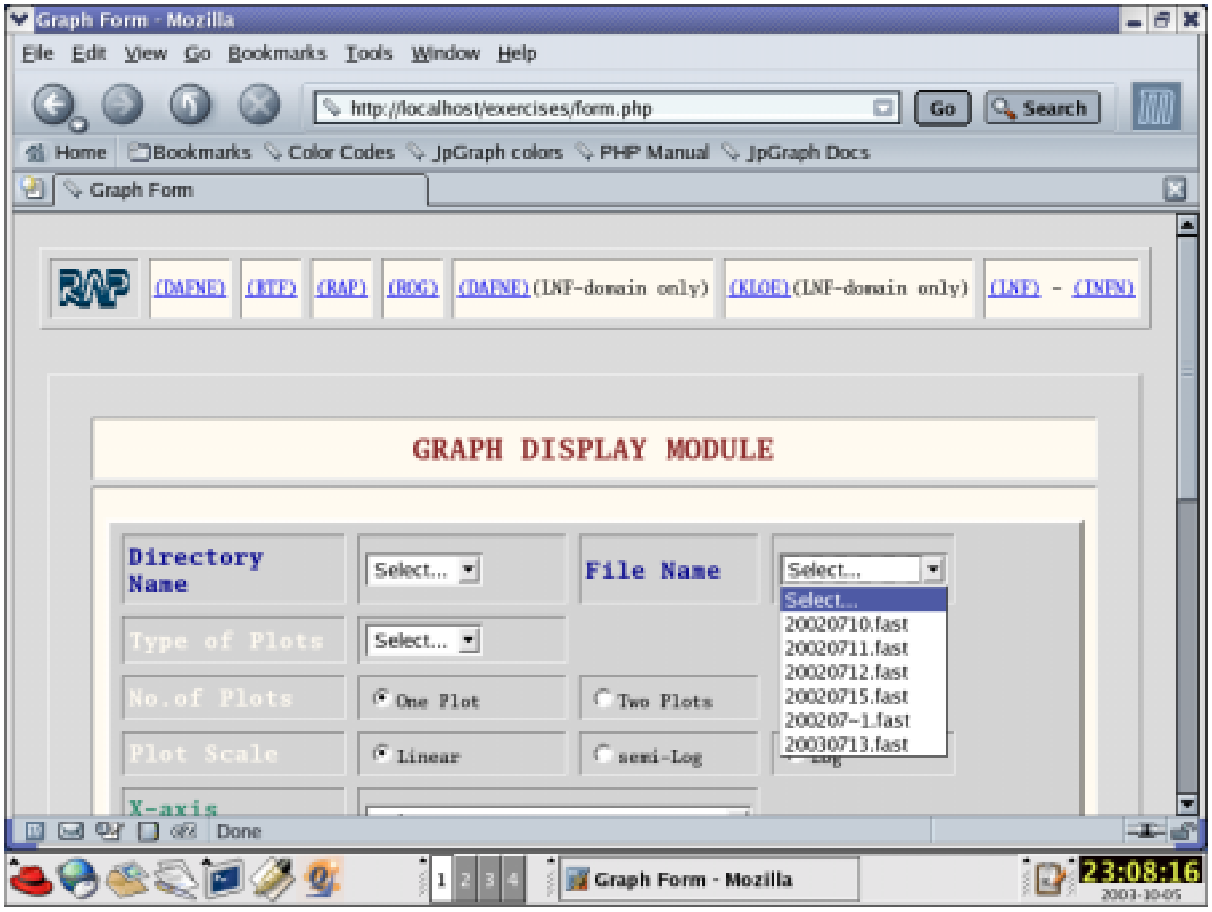
Directory Name: fast

Select...
Rap
fast
dat

Laboratori Nazionali di Frascati Istituto Nazionale di Fisica Nucleare
P.O.Box 13,00044-Frascati (Rome - Italy)
Phone (+39 06 9403 2411-8064) Fax (+39 06 9403 2256)

Fig. (3.2): Selection of graph-plot data directory

Once selected, the rest of the form is displayed. This will permit the selection of the desired file name and all other options from the appropriate selection list as shown below in fig. (3.2);


Graph Form - Mozilla
File Edit View Go Bookmarks Tools Window Help
http://localhost/exercises/form.php
Go
Search
Home
Bookmarks
Color Codes
JpGraph colors
PHP Manual
JpGraph Docs
Graph Form
(DAFNE) (BTF) (RAP) (ROG) (DAFNE)(LNF-domain only) (KLOE)(LNF-domain only) (LNF) - (INFN)
Done
Graph Form - Mozilla
23:08:16
2003-10-05

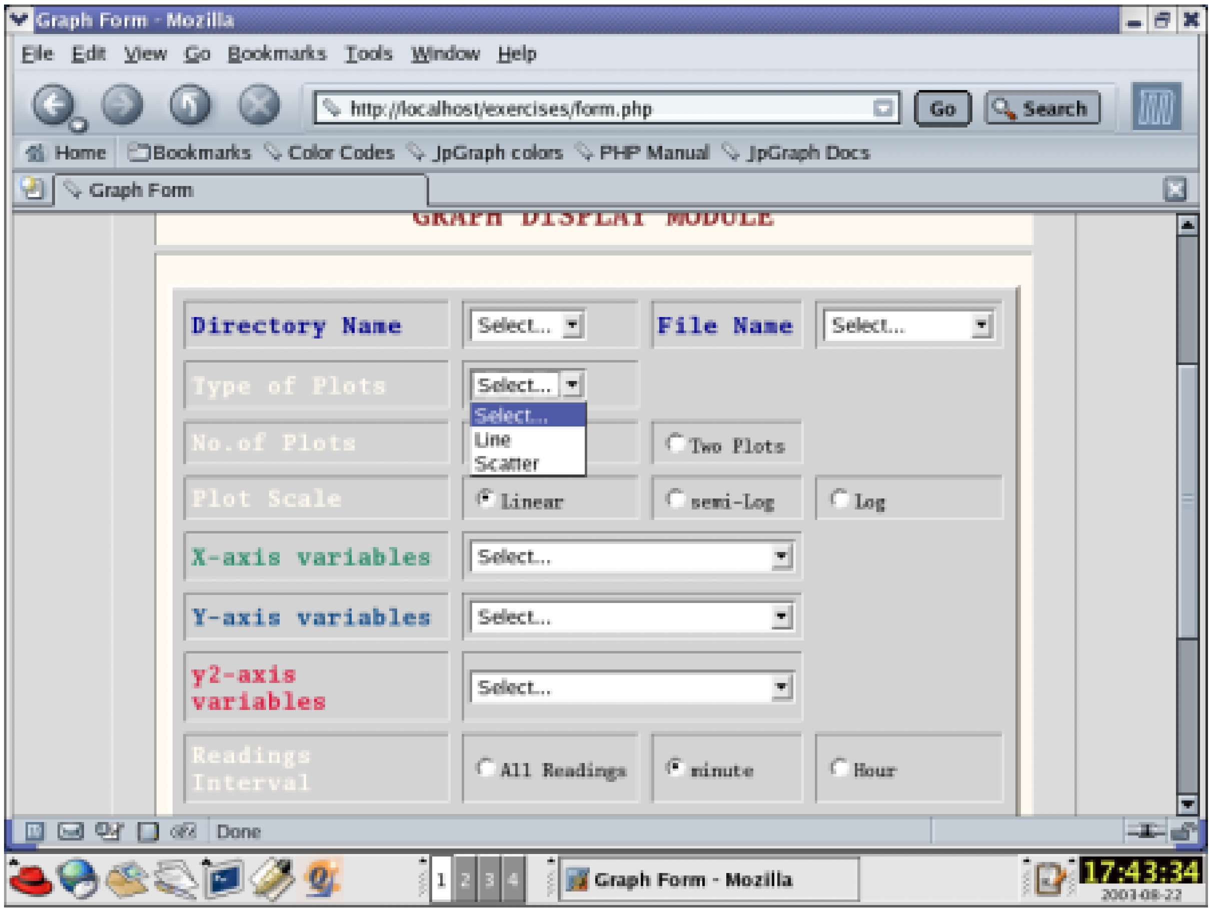
GRAPH DISPLAY MODULE

Directory Name	Select...	File Name	Select...
Type of Plots	Select...		
No.of Plots	● One Plot	○ Two Plots	
Plot Scale	● Linear	○ semi-Log	○ Log
X-axis			

Select...
20020710.fast
20020711.fast
20020712.fast
20020715.fast
200207-1.fast
20030713.fast

Fig. (3.3): Selection of file name

The plot type in turn is chosen from the selection list where there are two options; *linear* history or *Scatter* plots. This will decide the plot type shown in the second graph as well since it is only a detailed plot of the first graph;



Directory Name	Select...	File Name	Select...
Type of Plots	Select... (Select..., Line, Scatter)		
No.of Plots		○ Two Plots	
Plot Scale	[✓ Linear]	○ semi-Log	○ Log
X-axis variables	Select...		
Y-axis variables	Select...		
y2-axis variables	Select...		
Readings Interval	○ All Readings	[✓ minute]	○ Hour

Fig. (3.4): selection of graph plot type

Using the radio buttons options to choose the number of plots that will be displayed and the plot scale from the given options, leads in turn to choose the axes data variables, which are then selected as desired either for one or two plots, and determine which variable to assign for each axis. The variables are retrieved as mentioned earlier from the file *vars.txt* found in each directory.

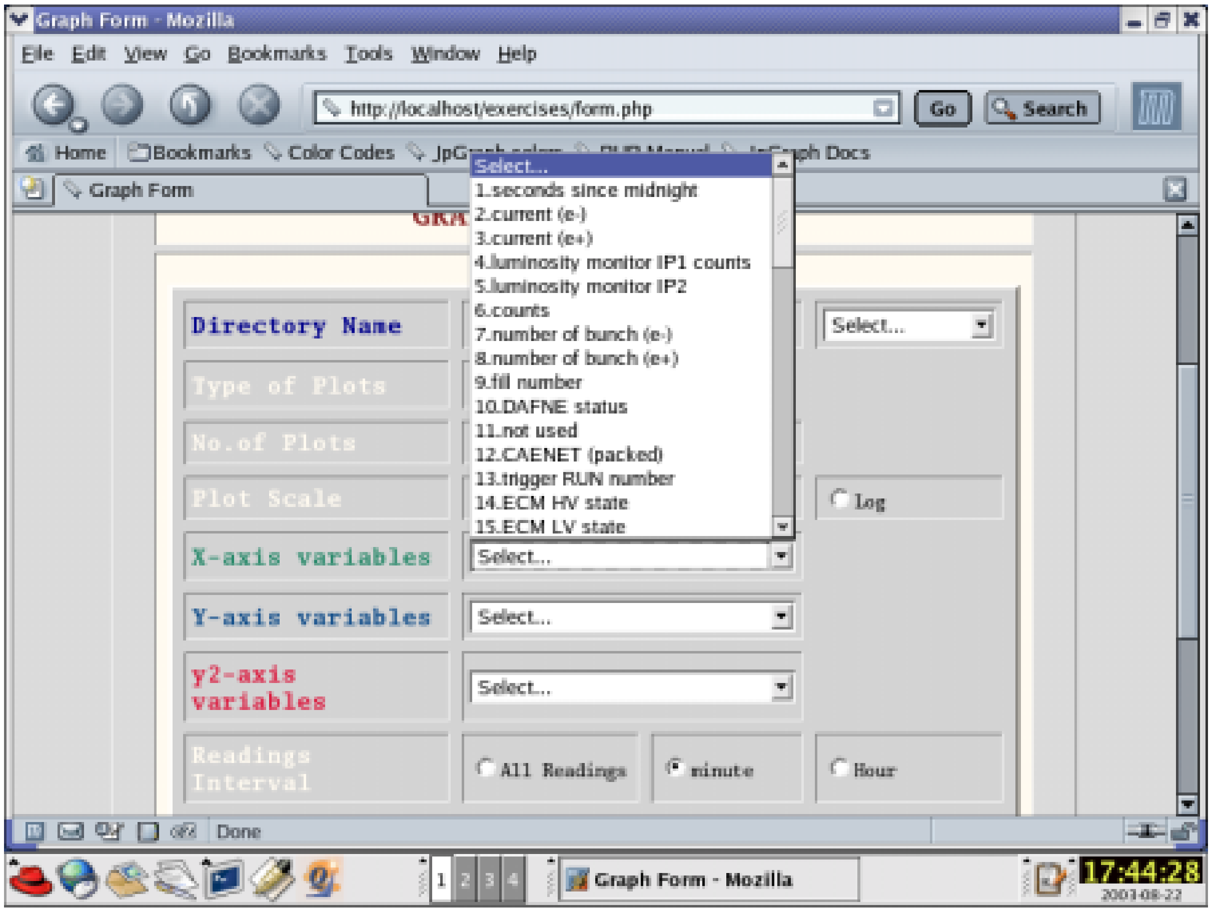


Fig.(3.5): Selection of Axis data variable

As soon as the show plot button is pressed the JpGraph displays the graph according to the selected plot type and interval options (All data, every minute, every hour, or every 2 hours). This is what the resulting plot will look like:

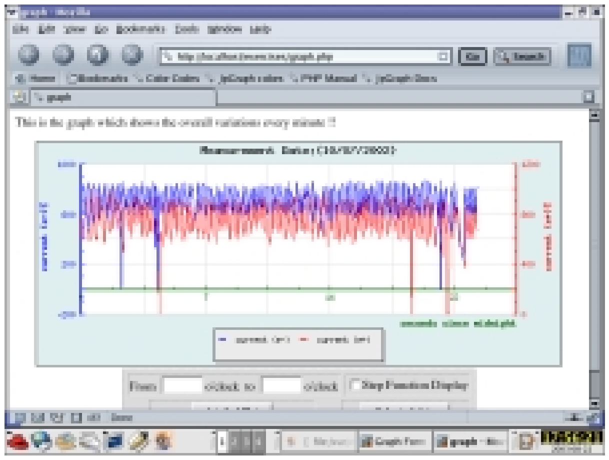


Fig. (3.6): First scatter graph

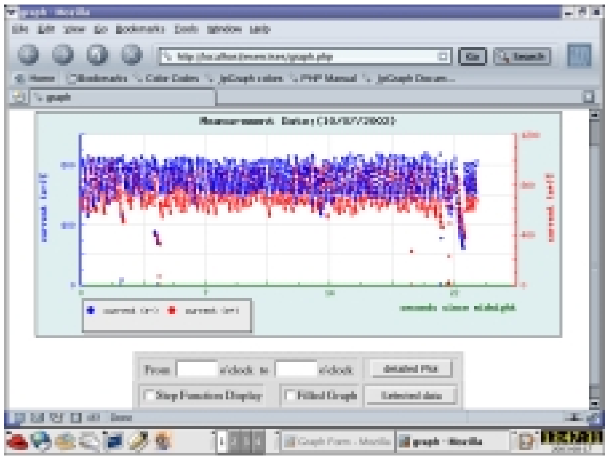


Fig. (3.7): First scatter

From the options in this page, we can display the detailed scatter (or linear) plots for all the readings with the selected start and end times (in hours) and step shape graph (if the option is selected) so output plot will look like this:

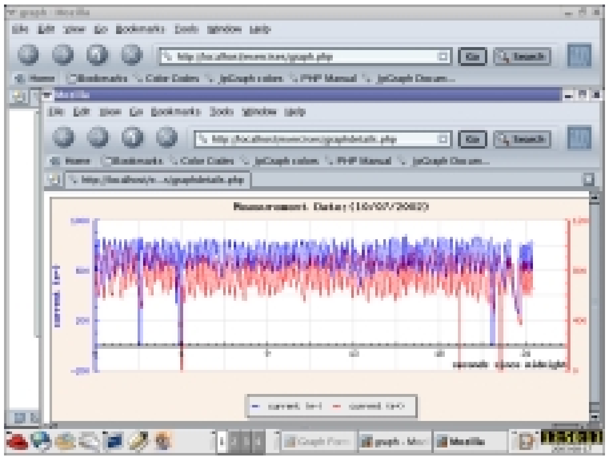


Fig. (3.8): Second line graph

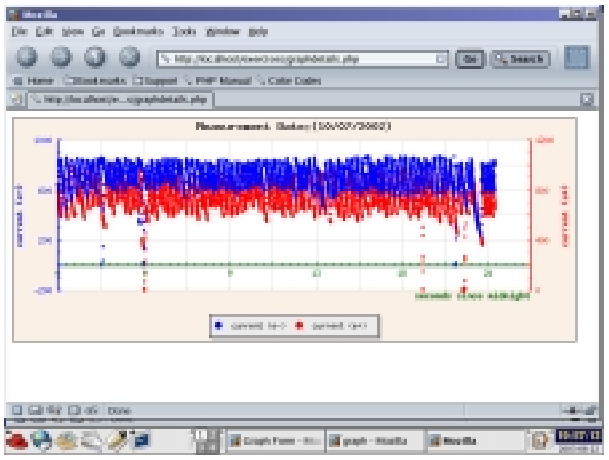


Fig. (3.9): Second scatter graph

It is important to mention here that the code to retrieve the data is inserted in both graph codes this is due to the difficulty in passing an enormous amount of data via the web server!! In addition to other security considerations which are among the PHP scripting drawbacks.

This approach is more useful because it gives us the possibility to see the plots between various data types without the facing the drawbacks of having plots only versus time changes.

(3.9) The script

When using JpGraph objects in PHP scripts, there are three basic steps to generating any graph:-

- The first step is to instantiate and set up the graph object itself. This can be compared to an artist preparing a canvas, to paint on. It includes specifying properties like the size and color of the image.
- The second step is to prepare each of the graphs. This means loading the data and, assigning properties like what type of graph, and what color the graph is.
- The third step is to finalize and output image. This step includes adding the graphs we defined in step two to the final output, placing any titles, legends, etc.

On the graph, General setup for all graphing scripts will include the following class defining files, which are so important since they contain the scripts defining every used class [6]:

```
 Include -or require_once- ('jpgraph/jpgraph.php');#graphing related includes
require_once ('jpgraph/jpgraph_bar.php');   # included only for bar plots
require_once ('jpgraph/jpgraph_line.php');  #included only for line plots
require_once ('jpgraph/jpgraph_pie.php');   # included only for pie plots
require_once('jpgraph/jpgraph_spider.php'); #included only for bar plots
require_once (‘/jpgraph/jpgraph_scatter.php’);# included only for scatter
plots
```

In most cases it also necessary to use some other important class defining files required for each graph such as:-

```
include ("./jpgraph/jpgraph_regstat.php");

include("./jpgraph/jpgraph_log.php" );
```

(3.9.1) The Enquiry Form program

Now that we are done with both graph generating and the data table display codes, lets go to the last part, which is the *Enquiry Form.* This PHP page retrieves the required information from the client to generate the desired graph plots dynamically[7] [10].

The major part of the commands is HTML based. The input data variables are extracted from the source files, all of which will be passed to the graph generating files. The code is shown below;

It starts with the ordinary HTML tags. The head tag contains the *Form Title* and the Meta tag that defines this page to web browsers[2].

```
<html>
<head>
<title>Graph Form</title>
<meta http-equiv="Content-Type"
content="text/html;charset=iso-8859-1">
</head>
```

The body tag begins with a tables containing different sites within the “INFN” **Istituto Nazionale di Fisica Nucleare** (*National Institute of Nuclear Physics*) and the “LNF” **Laboratori Nazionale di Frascati** (*The National Laboratories of Frascati*) [2];

```
<body bgcolor="#DCDCDC">
<center>
<table border = 1 cellpadding =2 cellspacing =6>
<tr>
<img src="pictures/RAPsma.gif" width=50 height=30 hspace=2
vspace=2 border=0>
</td>
<td align = center bgcolor="#FFFAF0">
<a href="http://www.lnf.infn.it/acceleratori/">(DAFNE)</a>
</td>
```

```
<td align = center bgcolor="#FFFAF0">
<a href="http://www.lnf.infn.it/acceleratori/btf/">(BTF)</a>
</td>
<td align = center bgcolor="#FFFAF0">
<a href="http://www.lnf.infn.it/esperimenti/rap">(RAP)</a>
</td>
<td align = center bgcolor="#FFFAF0">
<a href="http://www.roma1.infn.it/rog/">(ROG)</a>
</td>
<td align = center bgcolor="#FFFAF0">
<a href="http://dafne.lnf.infn.it/">(DAFNE)</a>(LNF-domain only)
</td>
<td align = center bgcolor="#FFFAF0">
<a href="http://kloeslow.lnf.infn.it:8000/">(KLOE)</a>(LNF-domain
only)
</td>
<td align = center bgcolor="#FFFAF0">
<a href="http://www.lnf.infn.it/">(LNF)</a> -
<a href="http://www.infn.it/">(INFN)</a>
</td></tr>
</table>
<table border="1" cellspacing="5" cellpadding="8">
<tr>
<td align = center bgcolor="#FFFAF0"><font SIZE=+2
color='#8B2323' face='arial,helvetica'><b>GRAPH DISPLAY
MODULE</b></font>
</td>
<td align = center></td></tr>
```

From here begins the PHP part of the script which retrieves the required inputs from the residing files[7] [10];

```
<?php
```

Definition of the colors used in the form;

```
$blue = '#104E8B';
$blue4 = '#00008B';
$skyblue = '#00BFFF';
$brown = '#8B2323';
$darkgrey = '#A9A9A9';
$gray = '#D4D4D4';
$red = '#DC143C';
$white = '#FFFAF0';
```

Assigning the data source directory;

```
$default_dir = '/var/www/html/exercises/data';
$dir = dir ($default_dir);
echo "<td align = center bgcolor=$white><table border =3
cellpadding =4 cellspacing =8 bgcolor= $gray>";
```

The directory opening engine uses a defined function called "**traverse_dir($dir)**" to explore any directory within the main defined directory and displays all the contents in a selection list from which it is possible to select the file name;

```
echo "<FORM ACTION = '$self_php' METHOD = 'post'>";
echo "<tr><td><font SIZE=+1 color='$blue4'
face='arial,helvetica'><b>Directory Name</b></font></td>";
echo "<td><SELECT NAME='dirname' OnChange='submit()';>";
echo "<OPTION>Select...</OPTION>";
        traverse_dir ($default_dir);
echo "</SELECT></td>";
echo "</FORM>";
```

The returned dir name in (full path) is necessary to display the contained files;

```
$dirname = $_REQUEST ['dirname'];
```

The minimum reading interval is decided from this part of code upon the type of data selected. This is done using the directory name after extracting it from the file path as follows;

```
$directory = basename ($dirname);
switch ($directory){
case ($directory == 'slow'):
$second =1;
$minute = 1;
$hour = 60;
$hours2 = 120;
break;
case ($directory == 'fast'):
$second =1;
$minute = 4;
$hour = 240;
$hours2 = 480;
break;
case ($directory == 'raw'):
$second =1;
$minute = 4;
$hour = 240;
$hours2 = 480;
break;}
```

Having done this, we are ready to go on to the second part of the form to open the file and choose graph options;

```
echo "<FORM ACTION='graph.php' METHOD='post' target='graph'>";
echo "<INPUT TYPE='hidden' NAME= 'dirname' VALUE='$dirname'>";
if ($dirname!= null){
   echo "<td><font SIZE=+1 color='#00008B'
   face='arial,helvetica'><b>File Name</b></font></td>";
```

This selection list calls a function called “**show_files ($dir**)“ to display all contained files inside it. (This function will be describe later);

```
echo "<td><SELECT NAME='filename'>";
   echo "<OPTION SELECTED >Select...</OPTION>";
   show_files ($dirname);
   echo "</SELECT></td>";
```

The following select list shows the graph plot that can be displayed;

```
   echo "<tr><td>”;
   echo "<<font SIZE=+1 color='$white' face='arial,helvetica'>”;
   echo ”<b>Type of Plots</b></font></td>";
   echo "<td><SELECT NAME='tgraph'>";
   echo "<OPTION>Select...</OPTION>";
   echo "<OPTION>Line</OPTION>";
   echo "<OPTION>Scatter</OPTION>";
   echo "</SELECT></td></tr>";
```

The number of plots to be shown on the graph settings is selected from the radio input type written below;

```
echo "<tr><td><font SIZE=+1 color='$white'
face='arial,helvetica'><b>No.of Plots</b></font></td>";
echo "<p><td><INPUT type='radio' name='nplots' value='1'
checked>One Plot</td>";
echo "<td><INPUT type='radio' name='nplots' value= '2' >Two
Plots</td>";
```

The plot scale-type option settings are selected as shown below;

```
echo "<tr><td><font SIZE=+1 color='$white'
face='arial,helvetica'><b>Plot Scale</b></font></td>";
echo "<p><td><INPUT type='radio' name='ptype' value='linlin'
checked>Linear</td>";
echo "<td><INPUT type='radio' name='ptype' value='linlog' >semi-
Log</td>";
```

```
echo "<td><INPUT type='radio' name='ptype' value='loglog'
>Log</td>";
```

The X-axis input variable settings;

```
echo "<p><tr><td>”;
echo "<font SIZE=+1 color='#218868'face='arial,helvetica'>”;
echo "<b>X-axis variables</b>”;
echo "</font></td>";
echo "<td align = center colspan = 2><SELECT NAME='data1'>";
echo "<OPTION SELECTED VALUE = ' '>Select...</OPTION>";
$n_file = file ("$dirname/var.txt");
foreach ($n_file as $key=>$val){
  $rkey =$key + 1 ;
  echo "<OPTION VALUE=$key>$rkey.$val</OPTION><br>";}
echo "</SELECT></td></tr>";
```

The Y-axis input variable settings;

```
echo "<p><tr><td><font SIZE=+1 color='$blue'
face='arial,helvetica'>”;
echo "<b>Y-axis variables</b></font>";
echo "<td align = center colspan = 2><SELECT NAME='data2'>";
echo "<OPTION SELECTED VALUE = ' '>Select...</OPTION>";
$n_file = file ("$dirname/var.txt");
foreach ($n_file as $key=>$val){
  $rkey =$key + 1 ;
  echo "<OPTION VALUE=$key>$rkey.$val</OPTION><br>";}
echo "</SELECT></td></tr>";
```

The Y2-axis input variable settings (only if the option is chosen) [6];

```
echo "<p><tr><td><font SIZE=+1 color='$red'
face='arial,helvetica'>”;
echo "<b>y2-axis variables<b></font>";
echo "<td align = center colspan = 2>”;
echo "<SELECT NAME='data3'>";
echo "<OPTION SELECTED VALUE = ' '>Select...</OPTION>";
```

```
$n_file = file ("$dirname/var.txt");

foreach ($n_file as $key=>$val){
  $rkey =$key + 1 ;
  echo "<OPTION VALUE=$key>$rkey.$val</OPTION><br>";}
echo "</SELECT></td></tr>";
```

The readings interval option settings (All detailes, minute,hour) [7]

```
echo "<p><u><tr><td>”;
echo "<font SIZE=+1 color='$white' face='arial,helvetica'>”;
echo "<b>Readings Interval</b></u></td>";
echo "<td><INPUT type='radio' name='pinterval' value='1'
checked>All Readings</td>";
echo "<td><INPUT type='radio' name='pinterval' value='4'
checked>minute</td>";
echo "<td><INPUT type='radio' name='pinterval'
value='240'>Hour</td></tr>";
```

Setting the Form Submit button [7];

```
echo "<tr><TABLE align= bottom bgcolor= $white><td align =
center>”;
echo "<INPUT TYPE='SUBMIT' NAME='STATUS' VALUE='Show
Plots'>";
```

Setting the Form reset button;

```
  echo "<td align = center >”;
  echo "<INPUT TYPE='RESET' NAME='RESET' VALUE='Reset
Form'></FORM></td>";
```

To show the table containing all data;

```
echo "<FORM ACTION='datashow.php' METHOD='post'
target='_blank'>";
echo "<INPUT TYPE='hidden' NAME= 'dirname' VALUE='$dirname'>";
echo "<INPUT TYPE='hidden' NAME= 'dirname' VALUE='$filename'>";
echo "<INPUT TYPE='hidden' NAME= 'dirname' VALUE='$date'>";
echo "<td align = center><INPUT TYPE='SUBMIT' NAME='Data'
VALUE='All data'></FORM></td></tr>";
}
echo "</TABLE>";
```

Now its time to define the functions [7] [10] mentioned earlier ; the first will trasverse[7] the directories and returns the file names to be shown on the form.The second returns the date of data measurements extracting it from the file name selected .

"Definitions of used functions "

Function [show_files($dir)] This function reads and sorts datafiles within the selected directory;

```
function show_files($dir){
if(!($dp =opendir ($dir))) die("cannot open $dir.");
while ($file = readdir($dp)){
  if($file != "var.dat"){       # if file name is different than
the input variables file
  $filenames[] = $file;
             }
          }
closedir ($dp);
sort ($filenames);        #sorts the read files
for ($i=0; $i < count ($filenames); $i++){
  if ($filenames[$i] !='.' && $filenames[$i] !='..') {
             echo "<OPTION
VALUE='$dir/$filenames[$i]'>$filenames[$i]</OPTION>";
       }
 }
 }
```

Function [trasverse_dir ($dir)] This function navigates through the directory hierarchy reading them all;

```
function traverse_dir($dir) {
  chdir ($dir);
  if(!($dp = opendir($dir))) die("can't open $dir.");
        while($file = readdir($dp)) {
        if(is_dir($file)) {
              if($file != '.' && $file != '..'){
                    echo "<OPTION
VALUE='$dir/$file'>$file</OPTION><br>";
              traverse_dir("$dir/$file"); # keep reading within the directory heirarchy
                          chdir($dir);
                    }
              }
              else echo "$file<br>";
```

```
          }
          closedir($dp);
     }    ?>                                     # end of the PHP part of code
```

At this point, we arrive at the end of the form table, so we conclude with the end table tag:

```
</table>
```

The last part of the page which contains the address and links to the "***National Institute for Nuclear Physics***" and "***Frascati National Laboratories***" is put in another table tag as follows:

```
<table border="1" cellpadding =4 cellspacing =8>
<tr>
<td align = center bgcolor="#FFFAF0">
<address>
<a href="http://www.lnf.infn.it/"> Laboratori Nazionali di
Frascati </a>
<a href="http://www.infn.it/"> Istituto Nazionale di Fisica
Nucleare </a></address>
<address> P.O.Box 13,00044-Frascati (Rome - Italy)</address>
<address> Phone (+39 06 9403 2411-8064) Fax (+39 06 9403
2256)</address></td>
</tr>
</table>
</center>
</body>
```
```
</html>    # end of code
```

(3.9.2) The main-graph program

As for our graph, we will be using Client Side Image Maps (CSIM) to generate a graph page with HTML mappings. Initially, we will include the following at the beginning of the script just like including libraries in C codes:-

```
include ("./jpgraph/jpgraph.php");
include ("./jpgraph/jpgraph_line.php");
include "./jpgraph/jpgraph_scatter.php";
include ("./jpgraph/jpgraph_regstat.php");
include("./jpgraph/jpgraph_log.php" );
```

Then we come to the data retrieving part of the code. Here we need to read the directories containing the required data files. First, we define the location of the graph's *png* image & the returned HTML map file (to be deleted in order to refresh the graph every time this file is called).This is done according to the code shown below:-

```
$graph = "./csimcache/graph.png";

$html = "./csimcache/graph_csim_.html";

if (file_exists($graph)){ unlink($graph);}

if (file_exists($graph)){ unlink($html);}
```

Then, we define the variables to read the POST transmitted controls into

```
$dirname = $_REQUEST ['dirname'];            #  To read the selected path name
$filename = $_REQUEST ['filename'];          #  To read the selected file name
$tgraph = $_REQUEST["tgraph"];               # To read the selected grph plot type
$nplots = $_REQUEST ["nplots"];              #  Decides the no. of axis to display
$hour_start = $_REQUEST["hour_start"];       #  Decides the initial time to display
$hour_end = $_REQUEST["hour_end"];           #  Decides the end time to display
$ptype = $_REQUEST ["ptype"];                #  To select the plot scale
$xkey = $_REQUEST ["data1"];                 #  To read the selected X-axis data
$ykey = $_REQUEST ["data2"];                 #  To read the selected Y-axis data
$y2key = $_REQUEST ["data3"];                #  To read the selected Y2-axis data
$pinterval = $_REQUEST [“pinterval”];        #  To read the selected data interval
$step = $_REQUEST["step"];                   #To display step function plot
```

And define some variables to be used in reading data from the file

```
$file = 0;
```

```
$rcount = 0;
$t_interval = 0;
$t_count = 0;
```

Open the specified file containing data

```
$file_h = fopen("$filename", "r");
```

To call the function that displays the acuisition date by extracting it from the file name (This will be described later on)

```
    $date = d_date ($filename);
```

Then we force the display of all readings if no time limits are selected

```
    if ($hour_start || $hour_end ==null){
    $hour_start = 0;
    $hour_end = 24;}
```

To change the time from hours to seconds in order to select the data readings directly from the file;

```
$second_start = $hour_start * 3600;
$second_end = $hour_end * 3600;
```

To read the lines from the specified file until the end of data or arrival at the end of file;

```
    while (!feof($file_h)) {
```

To read only line by line

```
  $file = fgets($file_h);
```

Then read the lines into an array

```
  $file_arr = preg_split("/\s/",$file,-1,PREG_SPLIT_NO_EMPTY);
```

This part decides the selection of data according to the *Start* and *End* times in seconds;

```
        $secondi = $file_arr[0];
        if ($secondi>=$second_start && $secondi<$second_end)
```

To read only the lines according to the assigned interval (within the selected period) into the array called (file_arr[ ]);

```
  if ($t_count==$t_interval)    {
        for ($key=0; $key < count($file_arr) ; $key += 1 ) {
                $value = $file_arr[$key];
```

To switch all the `file_arr[ ]` array's raw elements into columns in a new array with the name of `accdata[]`:

```
            $accdata[$key][$rcount] = round ($value); }
        $t_interval = $pinterval;
        $rcount++;
        $t_count=0;                              #  counts the total no. of rows

          }
        $t_count++;
        }
     }
      fclose($file_h);
```

To read the variables file in order to display all the axis-titles

```
        $n_file = file ("$dirname/var.dat");
```

Assigning the plot type to be recognised as a JpGraph class. This is done by concatenating the passed graph type into the following form;

```
          $graphplot = $tgraph."Plot";
```

Having done this, we can now create and display our first graph. We first start by defining the variables containing the retrieved data to be shown on the graph; this includes the plot's X-axis, Y-axis & - if selected – the second Y-axis (called Y2-axis) [6][7][10]:

```
          $xdata = $accdata[$xkey];
          $xtitle = $n_file[$xkey];
          $ydata = $accdata[$ykey];
          $ytitle = $n_file[$ykey];
          $y2data = $accdata[$y2key];
          $y2title = $n_file[$y2key];
```

Now its time to call the plot creation class where the (width = 700, height = 300) pixels the timeout is set to infinitly small `(-1)`. This is done by calling the following command [6]:

```
      $graph = new Graph(700,300,"auto",-1);
```

Next, we need to check the csimcache directory to see if there is an old (graph.png) image file -that's done by calling the following command;

```
    $graph->CheckCSIMCache("graph");
```

To set the plot scale into the selected from the enquiry form

```
    $graph->SetScale("$ptype");
```

The layout of the plot (colors, gridlines,…etc) are assigned as follows;

```
    $graph->SetMarginColor('azure2');
    $graph->ygrid->Show();
    $graph->xgrid->Show();
    $graph->SetGridDepth(DEPTH_FRONT);
    $graph->img->SetMargin(60,60,30,70);
    $graph->title->Set("$date");
```

The legend position and layout is assigned using the commands shown below

```
    $graph->legend->Pos(0.25,0.98,"center","bottom");

    $graph->legend->SetLayout(LEGEND_HOR);
```

To turn the time from seconds into hours (when time is selected as the X-axis variable)

```
    if($xkey == 0){
       foreach ($xdata as $index=>$value){
          $s = round ($value/3600);
          $c.=$s.',';
          }
       $x1data = preg_split("/,/",$c,-1,PREG_SPLIT_NO_EMPTY);
     }else {$x1data = $xdata;}
```

This part checks if the values of X-axis data have values more than a hundred or have negative values. If so, it will lower down the axis title for better aspect:

```
    foreach ($xdata as $value){
       if ($xdata>= 100||$xdata<0){
             $graph->xaxis->Settitlemargin(10);
                   }else
                    break;
                }
```

The following commands assign the layout of X-axis

```
$graph->xaxis->SetColor("darkgreen");
$graph->xaxis->SetTickLabels($x1data);
$graph->xaxis->SetLabelFormat("%d");
$graph->xaxis->SetWeight(2);
$graph->xaxis->SetFont(FF_FONT1,FS_NORMAL,12);
$graph->xaxis->title->Set("$xtitle");
$graph->xaxis->title->SetColor("darkgreen");
$graph->xaxis->title->SetFont(FF_FONT1,FS_BOLD,12);
$graph->xaxis->SetTextLabelInterval(2);
```

The Y-axis layout is decided according to the following commands;

```
foreach ($ydata as $value) {
  if ($ydata >= 100){$graph->yaxis->Settitlemargin(30);}else break;}
$graph->yaxis->SetColor("blue");
$graph->yaxis->SetWeight(2);
$graph->yaxis->title->SetColor("blue");
$graph->yaxis->SetFont(FF_FONT1,FS_NORMAL,12);
$graph->yaxis->title->Set("$ytitle");
$graph->yaxis->title->SetFont(FF_FONT1,FS_BOLD,14);
$graph->yaxis->SetTextLabelInterval(2);
$graph->yaxis->Settitlemargin(30);
```

The first linear plot is created by calling the following [6];

```
$splot1=new ScatterPlot($ydata);
$splot1->mark->SetType(MARK_FILLEDCIRCLE);
$splot1->mark->SetFillColor("blue");
$splot1->mark->SetWidth(1);
$splot1->mark->SetColor('blue@0.1');
$splot1->SetLegend ($ytitle);
```

This command shows the first line plot on the graph [6];

```
$graph->Add($splot1);
```

The second plot will only be displayed if this option is selected

```
if ($nplots == 2){
```

If the condition is true, the second linear plot will be created. But fist,lets assign the plot type according to the option selected

```
$pt = substr ($ptype,-3);
```

```
$graph->SetY2Scale("$pt");
$splot2=new ScatterPlot($y2data);
$splot2->mark->SetType(MARK_FILLEDCIRCLE);
$splot2->mark->SetFillColor("red");          # A red color for the second plot
$splot2->mark->SetWidth(1);
$splot2->mark->SetColor('red@0.1');
$splot2 ->SetLegend("$y2title");             # Assign the legend to be shown
```

To adjust the Y2-axis title to permit better visualization of data values [6];

```
foreach ($y2data as $value) {
  if ($y2data >= 100){$graph->y2axis->Settitlemargin(38);}else break;}
$graph->y2axis->SetColor("red");
$graph->y2axis->SetWeight(2);
$graph->y2axis->SetFont(FF_FONT1,FS_NORMAL,12);
$graph->y2axis->title->Set($y2title);
$graph->y2axis->title->SetColor("red");
$graph->y2axis->title->SetFont(FF_FONT1,FS_BOLD,14);
//setting y-axis label interval
$graph->y2axis->SetTextLabelInterval(2);
$graph->AddY2($splot2); }            # Add the second line plot to the graph
```

Finally, this graph plot is displayed using these command

```
$graph->StrokeCsim(basename( __FILE__));
```

Let us quickly review the three steps required to generate the chart. The first three lines that start with *$graph* create the object, set a margin and set the scales for the graphs.
The next step is to add the individual lines. We do this by creating *new LinePlot* objects, then calling their *setcolor* method. Finally, we can permit the display of the graph using [6]:-

```
$graph->StrokeCsim(basename( __FILE__));
```

Now, lets talk about the second graph which displays a more detailed graph of the data. For this purpose, another file called `graphdetails.php` is used. It receives all options regarding the plot and all required passed over variables from the enquiry form in addition to the time interval chosen at the main graph page as the graph zoom. This includes *Time start* and *Time end* of display options, in addition to the *Step-Function* display.

(3.9.3) The detaied graph program:

The detailed graph is created according to the following code;

```
echo "<p><FORM ACTION='graphdetails.php' METHOD='post'
target='_blank'>";
```

The passed controls from the main enquiry form are: the path, the file name, the date, the axis labels and the selected input data values to be shown[7][10];

```
echo "<INPUT TYPE='hidden' NAME= 'd_dirname' VALUE='$dirname'>";
echo "<INPUT TYPE='hidden' NAME= 'f_filename'
VALUE='$filename'>";
echo "<INPUT TYPE='hidden' NAME= 'date' VALUE='$date'>";
echo "<INPUT TYPE='hidden' NAME= 'datas1' VALUE='$data1'>";
echo "<INPUT TYPE='hidden' NAME= 'datas2' VALUE='$data2'>";
echo "<INPUT TYPE='hidden' NAME= 'datas3' VALUE='$data3'>";
echo "<INPUT TYPE='hidden' NAME= 'n_plots' VALUE='$nplots'>";
echo "<INPUT TYPE='hidden' NAME= 'p_type' VALUE='$ptype'>";
echo "<INPUT TYPE='hidden' NAME= 'xtitle' VALUE='$xtitle'>";
echo "<INPUT TYPE='hidden' NAME= 'ytitle' VALUE='$ytitle'>";
echo "<INPUT TYPE='hidden' NAME= 'y2title' VALUE='$y2title'>";
echo "<table border = 1 cellpadding =2 cellspacing =6
bgcolor='#DCDCDC'>";
echo "<tr><td align = center  colspan='6'>From &nbsp;";
echo "<INPUT TYPE='text' size='6' NAME= 'hour_start'>";
echo "&nbsp;o'clock";
echo "&nbsp; to &nbsp;";
echo "<INPUT TYPE='text' size='6' NAME= 'hour_end'>";
echo "&nbsp;o'clock</td>";
echo "<td align = center colspan='1'><INPUT TYPE='SUBMIT'
VALUE='detailed Plot'></td></tr>";
echo "<tr><td align = center colspan='3'>";
echo "<INPUT TYPE='checkbox' NAME= 'step' VALUE='1'>Step Function
Display</td>";
echo "<td></td>";
echo "<td align = left colspan=2><INPUT TYPE='checkbox' NAME=
'fill' VALUE='1'>Filled Graph</td>";
echo "</FORM>";
echo "<FORM ACTION='seldata.php' METHOD='post' target='_blank'>";
echo "<INPUT TYPE='hidden' NAME= 'dirname' VALUE='$dirname'>";
echo "<INPUT TYPE='hidden' NAME= 'filename' VALUE='$filename'>";
echo "<INPUT TYPE='hidden' NAME= 'data_1' VALUE='$data1'>";
echo "<INPUT TYPE='hidden' NAME= 'data_2' VALUE='$data2'>";
echo "<INPUT TYPE='hidden' NAME= 'data_3' VALUE='$data3'>";
echo "<INPUT TYPE='hidden' NAME= 'xtitle' VALUE='$xtitle'>";
echo "<INPUT TYPE='hidden' NAME= 'ytitle' VALUE='$ytitle'>";
echo "<INPUT TYPE='hidden' NAME= 'y2title' VALUE='$y2title'>";
echo "<td align = center><INPUT TYPE='SUBMIT' NAME='Data'
VALUE='Selected data'></FORM></td></tr>";
echo "</table>";
```

The above lines are just ordinary HTML commands written between two quotes in order to be escaped by the PHP compiler.

Finally, at the end of the code, we find the display date function - mentioned ealier – in which the date is displayed by chunking the original file name and putting it in the right order using the following commands;

We first define the function

```
function d_date ($path){
```

The file name is extracted from the rest of the path using [10];

```
$nfile = basename ($path);
```

The first part of the file name before the file type seperated by "."

```
$tok = strtok($nfile,".");
```

The date components are selected in turn as follows;

```
$year = substr("$tok", 0, 4);          # Extracts the year part of the date
$month = substr("$tok", 4, 2);         # Extracts the month part of the date
$fday = substr("$tok",-2);             # Extracts the day part of the date
```

If there are any non digital characters -as for some of the already saved files- they will be eliminated;

```
if (ereg("~",$fday,$reg)) {
  $t = strtok($fday,"~");
  $day = "0$t";
} else $day = $fday;
return $date = ("Measurement Date:($day/$month/$year)");
}
?>
```

Note:- It is mostly important to mention here that the use Client Side Image Maps (CSIM) [6] gives us the possibility to include forms, text,… etc on the same graph page.

Now let us go to the second graph, which displays the same type of graph showing the selected input data plots . In this graph, the data all the data are shown in the least interval as default for the whole 24 hours.

As usual, we include the same necessary class defining files as shown below [6] [7];

```
<?php
#includes the necessary JpGraph files to generate the graph
include ("./jpgraph/jpgraph.php");
include ("./jpgraph/jpgraph_line.php");
include "./jpgraph/jpgraph_scatter.php";
include ("./jpgraph/jpgraph_regstat.php");
include ("./jpgraph/jpgraph_log.php" );
```

Then comes next the file refresh part (the same as described earlier)

```
$graph = "./csimcache/graphdetails.png";
$html = "./csimcache/graphdetails_csim_.html";
if (file_exists($graph)){ unlink($graph);}
if (file_exists($graph)){ unlink($html);}
```

The insertion of the passed control variables from the form and the ones from the preceding main graph plot is the same as the previous graph;

Passed data from the form

```
$d_dirname= $_REQUEST["d_dirname"];
$f_filename= $_REQUEST["f_filename"];
$date = $_REQUEST["date"];
$hour_start = $_REQUEST["hour_start"];
$hour_end = $_REQUEST["hour_end"];
$n_plots = $_REQUEST ["n_plots"];
$p_type = $_REQUEST ["p_type"];
$t_graph= $_REQUEST["t_graph"];
$xkey = $_REQUEST["datas1"];
$ykey = $_REQUEST["datas2"];
$y2key = $_REQUEST["datas3"];
```

The axes titles are passed from the previous graph;

```
$xtitle = $_REQUEST["xtitle"];
$ytitle = $_REQUEST["ytitle"];
$y2title = $_REQUEST["y2title"];
```

A new control is the introduction of the step function option to the graph;

```
$step = $_REQUEST["step"];
```

The rest of the code is the same as the main graph's except for the use of the ordinary caching in stead of using CSIM[7].

```
$file = 0;
$rcount = 0;
$t_interval = 0;
$t_count = 0;
```

The data retrieving part of the code is the same since the same data source is required;

```
$file_h = fopen("$f_filename", "r"); # open the data source file
containing data (read only)
```

Displays all readings if no time limits are selected

```
if ($hour_start == null || $hour_end == null){
$hour_start = 0;
$hour_end = 24; }
```

To switch the selected time from hours to seconds

```
    $second_start = $hour_start * 3600;
    $second_end = $hour_end * 3600;
```

To read the lines from the specified file

```
    while (!feof($file_h)) {
      $file = fgets($file_h);  # to see only the last line
      $file_arr = preg_split("/\s/",$file,-1,PREG_SPLIT_NO_EMPTY);
```

To read only the lines according within the assigned time limits

```
            $secondi = $file_arr[0];
            if ($secondi>=$second_start && $secondi<$second_end)
            {
```

To read only the lines according to the assigned interval

```
      if ($t_count==$t_interval){
                for ($key=0; $key < count($file_arr) ; $key+=1) {
                          $value = $file_arr[$key];
```

Read all column elements in an array

```
            $accdata[$key][$rcount] = round($value);
      }
      $t_interval = 1;
      $rcount++;
      $t_count=0;                          # counts the total no. of rows
      }
      $t_count++;
       }
    }
    fclose($file_h);
```

Assigning the plot type to be recognised as a JpGraph class [6];

```
    $graphplot = $t_graph."Plot";
```

To assign the imported data to the graph

```
$xdata = $accdata[$xkey];
$ydata = $accdata[$ykey];
$y2data = $accdata[$y2key];
```

To change time values from seconds to hours for x-axis data (if time is chosen for X-axis);

```
if ($xkey == 0){
foreach ($xdata as $index=>$value) {
       $s = round ($value/3600);
       $c.=$s.',';    }
  $x1data = preg_split("/,/",$c,-1,PREG_SPLIT_NO_EMPTY);
}else $x1data = $xdata;
```

To create the graph. These two calls are always required [6];

```
$graph = new Graph(750,300,"auto",-1);
#$graph->CheckCSIMCache("graphdetails",1);
$graph->SetScale("$p_type");
$graph->ygrid->Show();          # shows the Y-axis gridlines
$graph->xgrid->Show();          # shows the x-axis gridlines
$graph->img->SetMargin(60,60,30,70);
$graph->title->Set("$date");
$graph->SetMarginColor('linen');
```

Adjust the legend position [6];

```
$graph->legend->SetLayout(LEGEND_HOR);
$graph->legend->Pos(0.5,0.98,"center","bottom");
```

Assigning X-axis specifications:-

To adjust the margin of the X-axis title according to data values

```
foreach ($xdata as $value) {
  if ($xdata >= 100){$graph->xaxis->Settitlemargin(20);}else
break;}
$graph->xaxis->SetColor("darkgreen");
$graph->xaxis->SetWeight(2);
$graph->xaxis->title->Set($xtitle);
```

```
$graph->xaxis->title->SetColor("darkgreen");
$graph->xaxis->title->SetFont(FF_FONT1,FS_BOLD,14);
$graph->xaxis->SetTickLabels($x1data);
$graph->xaxis->SetFont(FF_FONT1,FS_NORMAL,12);
$graph->xaxis->SetTextLabelInterval(2);
```

Assigning Y-axis specifications:-

To adjust the margin of the Y-axis title according to data values;

```
foreach ($ydata as $value) {
  if ($ydata >= 100){$graph->yaxis->Settitlemargin(30);}else
break;}
$graph->yaxis->SetColor("blue");
$graph->yaxis->SetWeight(2);
$graph->yaxis->title->SetColor("blue");
$graph->yaxis->SetFont(FF_FONT1,FS_NORMAL,12);
$graph->yaxis->title->Set("$ytitle");
$graph->yaxis->title->SetFont(FF_FONT1,FS_BOLD,14);
$graph->yaxis->SetTextLabelInterval(2);
$graph->yaxis->scale->SetGrace(10,10 );
$graph->yaxis->Settitlemargin(45);
```

Create the first linear plot;

```
$plot1 = new $graphplot($ydata);
```

To assign the marker in case of scatter plot type for Y-axis

```
if ($t_graph == "Scatter"){
$plot1->mark->SetType(MARK_FILLEDCIRCLE);
$plot1->mark->SetFillColor("blue");
$plot1->mark->SetWidth(1);
$plot1->mark->SetColor('blue@0.1');
}
```

In case the step function option is chosen (Linear Plot);

```
if ($step == 1) {$plot1->SetStepStyle();}
```

Set the legend;

```
$plot1->SetLegend("$ytitle");
$plot1->SetColor("blue");
$plot1->SetWeight("1");
```

Add the first plot to the graph;

```
$graph->Add($plot1);
```

Decides to show the second Y-axis plot upon selection (from the enquiry form)

```
if ($n_plots == 2){
```

Creation of the second plot;

```
$pt= substr ($p_type,-3);   # extract the plot scale part (linear, semi-log, log-log)
$graph->SetY2Scale("$pt");
$graph->y2axis->SetColor("red");
$graph->y2axis->SetWeight(2);
$graph->y2axis->SetFont(FF_FONT1,FS_NORMAL,12);
$graph->y2axis->title->Set("$y2title");
$graph->y2axis->title->SetColor("red");
$graph->y2axis->title->SetFont(FF_FONT1,FS_BOLD,14);
$graph->y2axis->SetTextLabelInterval(2);
```

To adjust the margin of the Y2-axis title according to data values;

```
foreach ($y2data as $value) {
  if ($y2data >= 100){$graph->y2axis->Settitlemargin(38);}else
break;}
```

Then we create the second linear plot;

```
$plot2 = new $graphplot($y2data);
if ($step ==1) {$plot2->SetStepStyle();}
```

To set the legend;

```
$plot2->SetLegend("$y2title");
$plot2->SetColor("red");
$plot2->SetWeight("1");
```

To assign the marker in case of scatter plot type for Y2-axis [6];

```
if ($t_graph == "Scatter"){
   $plot2->mark->SetType(MARK_FILLEDCIRCLE);
   $plot2->mark->SetFillColor("red");
   $plot2->mark->SetWidth(1);
   $plot2->mark->SetColor('red@0.1');
}
```

To add the second plot to the graph [6];

```
$graph->AddY2($plot2);
}
```

Display the graph using the regular stroke command (unlike the case in the main graph) [6];

```
$graph->Stroke();
?>         #    end of the code
```

To display all the data in a table, we use the following code:

```
<?php
```

The directory and file names are imported from the enquiry form;

```
$dirname = $_REQUEST ['dirname'];
$filename = $_REQUEST ['filename'];
```

To put all file contents in an array, we use the following command;

```
$n_file = file ("$dirname/$filename");
```

Then we put every row in an array inside the major one;

```
foreach ($n_file as $key=>$val){
$file_arr[$key] = $val; preg_split("/\s/",$n_file[$key],-
1,PREG_SPLIT_NO_EMPTY);}
```

The output table is constructed using the following HTML table tags;

```
echo "<table align = 'center' border=1>";
```

The table header´s are read from the file containing the variable names in this directory;

```
$h_file = file ("$dirname/var.txt");
echo "<tr>";
foreach ($h_file as $key=>$val){
  $rkey =$key + 1 ;
```

```
     echo "<th align= 'center' ><font SIZE=-1 color='#8B2323'
   face='arial,helvetica'>$val</th>";}
   cho "</tr>";
   foreach ($file_arr as $kkey=>$vals){
   echo "<tr>";
   foreach ($vals as $l_key=>$l_val){
   echo "<td align= 'center'><font SIZE=-2 color='#8B2323'
   face='arial,helvetica'>$l_val</td>";}
   echo "</tr>";}
   echo "</table >";
```

Finally, the table is displayed using the following line;

```
   #print_r($file_arr);
   ?>
```

(3.9.4) The data table program

To show a table of the slected data file, the following code is written. As usual, it starts with the PHP tag:

```
<?php
```

Then receive the passed contro varaibles:

```
$dirname = $_REQUEST ['dirname'];
$filename = $_REQUEST ['filename'];
```

This part will open only the defined file in case no file is selected to be shown:

```
if (!$dirname || !$filename) {
$dirname = "./data/fast";
$filename = "200207~1.fas";}
```

The file contents are put into an array using the following:

```
$n_file = file ("$dirname/$filename");
```

For all lines in the new array, each data line is broken into its components separating each data from the other using the following ,

```
     foreach ($n_file as $key=>$val){
     $file_arr[$key] = preg_split("/\s/",$n_file[$key],-1,
     PREG_SPLIT_NO_EMPTY);}
```

The data readings are then put into a table using the following commands starting with the data variable names (exatracted from the file “`var.txt`” contining them) as the header of the table;

```
echo "<table align = 'center' border=1>";
$h_file = file ("$dirname/var.txt");
echo "<tr>";
foreach ($h_file as $key=>$val){
     $rkey =$key + 1 ;
     echo "<th align= 'center' ><font SIZE=-1 color='#8B2323'
face='arial,helvetica'>$val</th>";}
echo "</tr>";
```

Then, the data are extracted fripm the array and inserted into the table line by line;

```
foreach ($file_arr as $kkey=>$vals){
echo "<tr>";
foreach ($vals as $l_key=>$l_val){
echo "<td align= 'center'><font SIZE=-2 color='#8B2323'
face='arial,helvetica'>$l_val</td>";}
echo "</tr>";}
```

The table closing tag is written after inserting all data into the table;

```
echo "</table >";
```

finally finshing with the PHP end tag to end the program;

```
?>
```

To demonstrate the functionality of this model, lets see the examples below:

(3.9.5) Example:

Assume that we want to see the relationship between the (+ve) and (–ve) currents against time in a Fast type RAP experiment data collected on (10/07/2002). The scale to be used is linear and the graph should be according to the following specifications:

1. Scatter plot, with a reading interval of (1) minute.
2. Linear history plot, to see all experiment readings.
3. Display a table Showing all the data contained in this file.

To do this , First, we want to see scatter plot. Applying this and selecting all options on the form, we get :

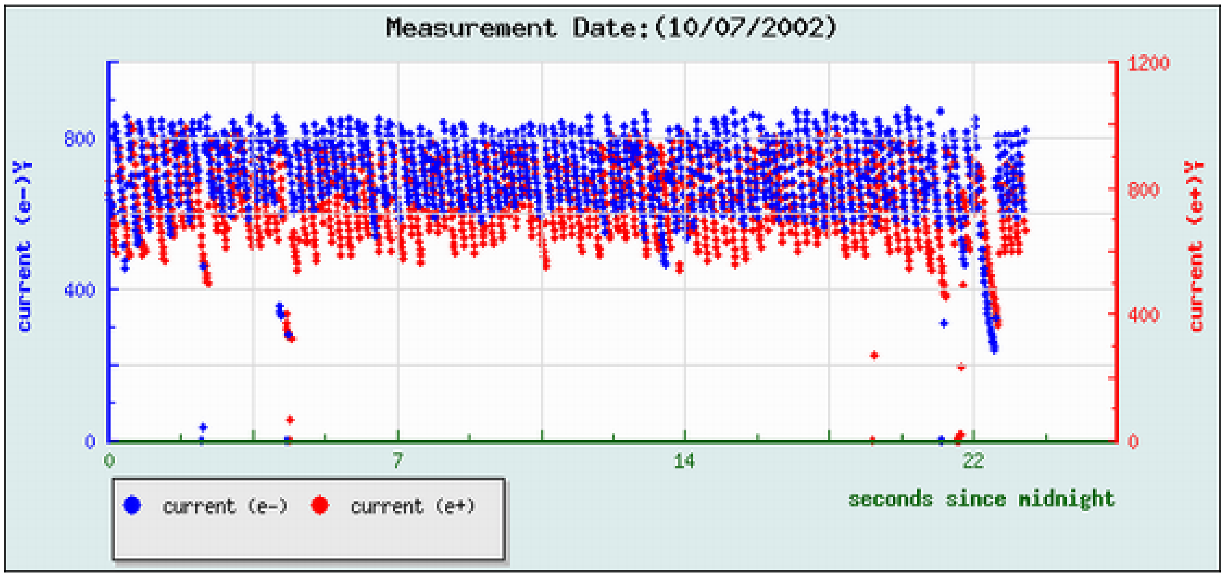


Fig. (3.10): The example scatter plot

The detailed scatter history graph is as shown below:-

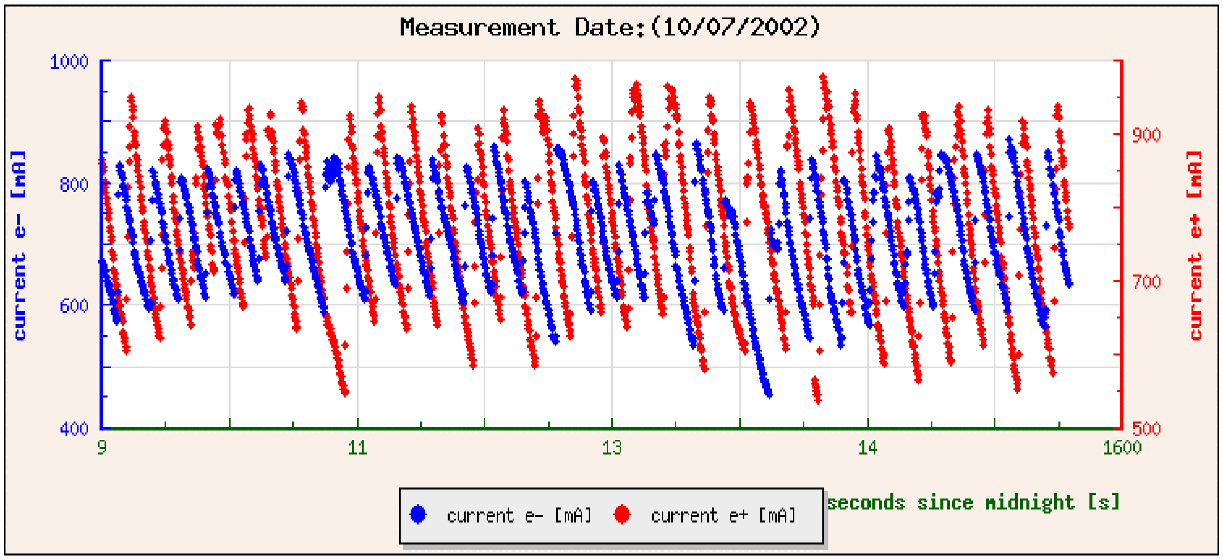


Fig. (3.12): The example detailed scatter plot

Second, by simply selecting Line plot option, and changing the all readings options on the enquiry form, we get:

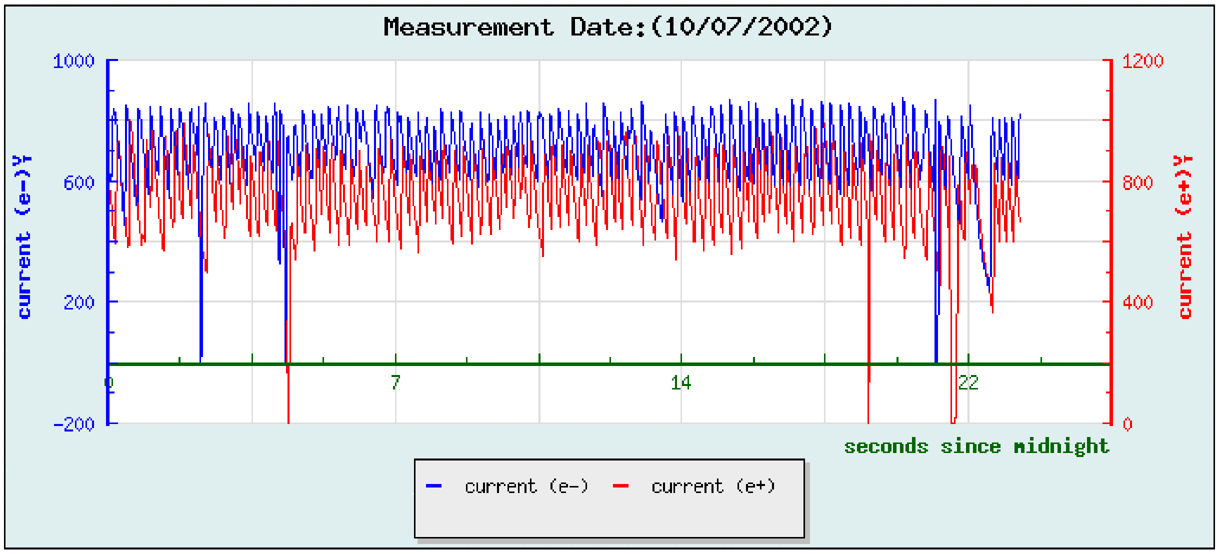


Fig. (3.11): The example line plot

On the other hand, the detailed Linear plot of the example is displayed according to the following:-

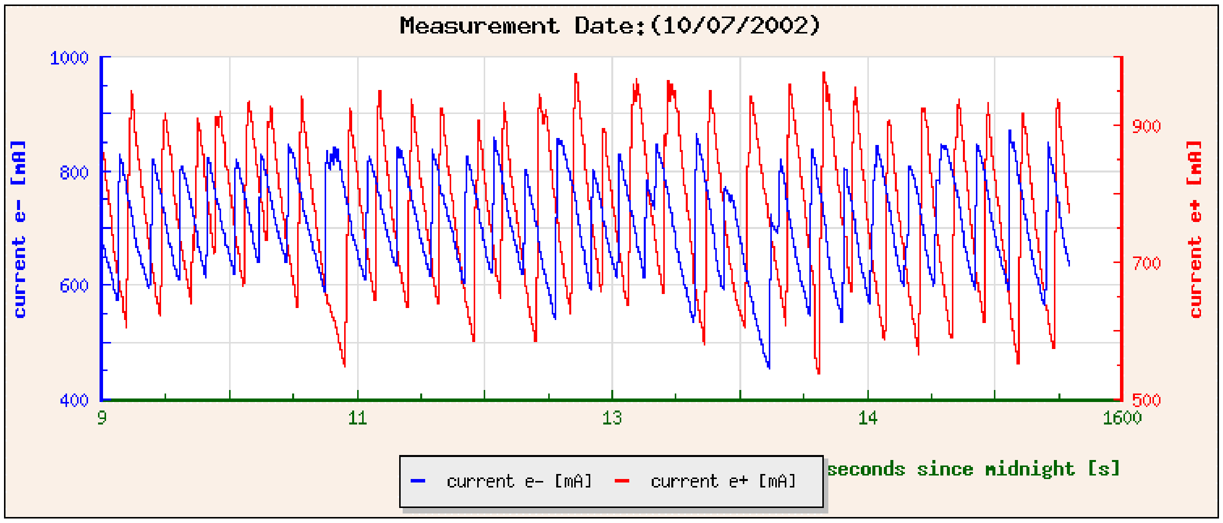
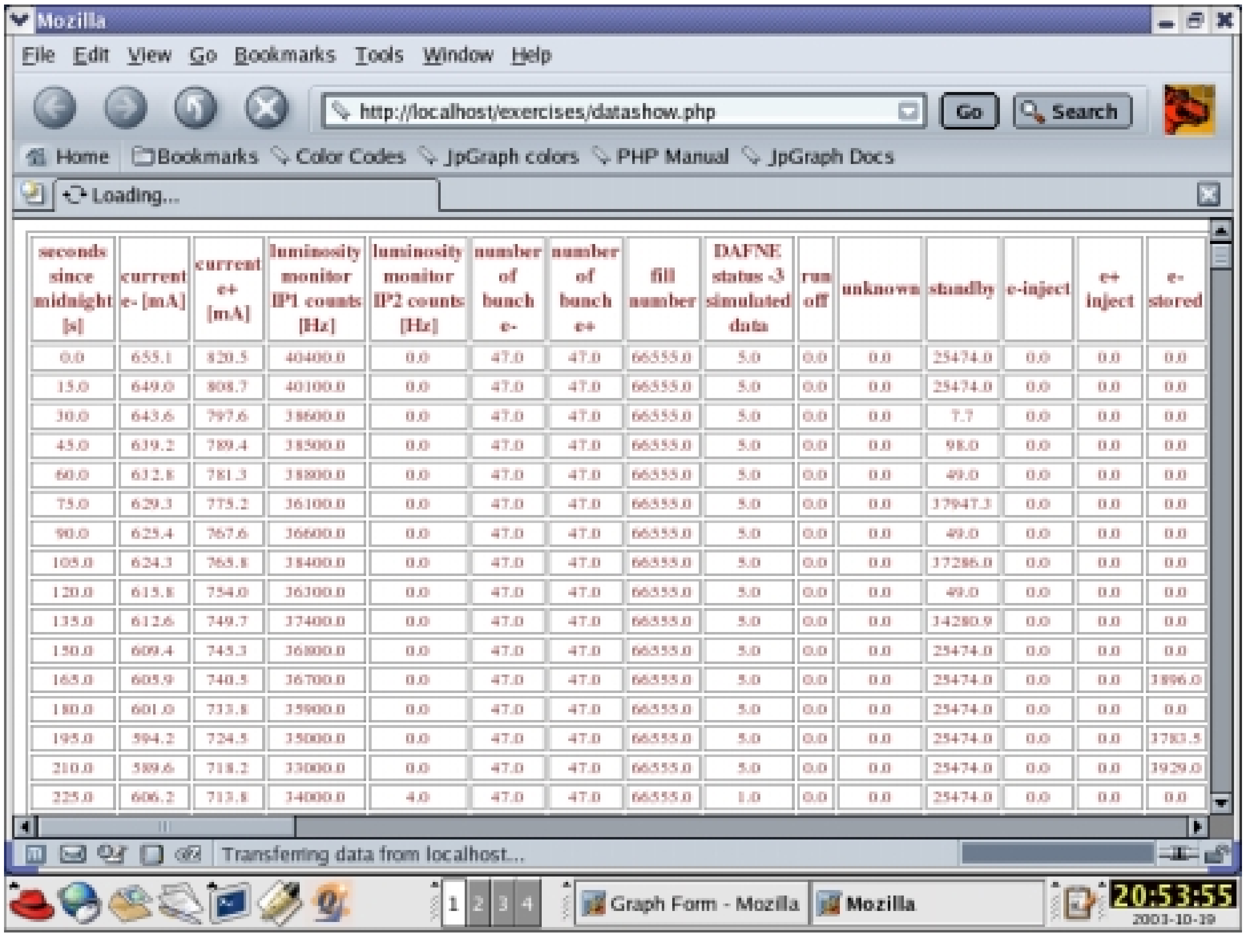


Fig. (3.13): The example detailed line plot

The above graphs clearly show the required plots of the three selected variables assigned for X, Y and Y2 axes from a data file found in a selected directory.

The last part of this example is the data table to be shown on the browsrer. With the same selected options for the graphs, once the Show Table button is pressed, the following table will be shown in another window:



seconds since midnight [s]	current e- [mA]	current e+ [mA]	luminosity monitor IP1 counts [Hz]	luminosity monitor IP2 counts [Hz]	number of bunch e-	number of bunch e+	fill number	DAFNE status -3 simulated data	run off	unknown	standby	e-inject	e+ inject	e- stored
0.0	655.1	820.5	40400.0	0.0	47.0	47.0	66555.0	5.0	0.0	0.0	25474.0	0.0	0.0	0.0
15.0	649.0	808.7	40100.0	0.0	47.0	47.0	66555.0	5.0	0.0	0.0	25474.0	0.0	0.0	0.0
30.0	643.6	797.6	38600.0	0.0	47.0	47.0	66555.0	5.0	0.0	0.0	7.7	0.0	0.0	0.0
45.0	639.2	789.4	38500.0	0.0	47.0	47.0	66555.0	5.0	0.0	0.0	98.0	0.0	0.0	0.0
60.0	632.8	781.3	38800.0	0.0	47.0	47.0	66555.0	5.0	0.0	0.0	49.0	0.0	0.0	0.0
75.0	629.3	775.2	36100.0	0.0	47.0	47.0	66555.0	5.0	0.0	0.0	37947.3	0.0	0.0	0.0
90.0	625.4	767.6	36600.0	0.0	47.0	47.0	66555.0	5.0	0.0	0.0	49.0	0.0	0.0	0.0
105.0	624.3	765.8	38400.0	0.0	47.0	47.0	66555.0	5.0	0.0	0.0	37286.0	0.0	0.0	0.0
120.0	615.8	754.0	36300.0	0.0	47.0	47.0	66555.0	5.0	0.0	0.0	49.0	0.0	0.0	0.0
135.0	612.6	749.7	37400.0	0.0	47.0	47.0	66555.0	5.0	0.0	0.0	34280.9	0.0	0.0	0.0
150.0	609.4	745.3	36800.0	0.0	47.0	47.0	66555.0	5.0	0.0	0.0	25474.0	0.0	0.0	0.0
165.0	605.9	740.5	36700.0	0.0	47.0	47.0	66555.0	5.0	0.0	0.0	25474.0	0.0	0.0	3896.0
180.0	601.0	733.8	35900.0	0.0	47.0	47.0	66555.0	5.0	0.0	0.0	25474.0	0.0	0.0	0.0
195.0	594.2	724.5	35000.0	0.0	47.0	47.0	66555.0	5.0	0.0	0.0	25474.0	0.0	0.0	3783.5
210.0	589.6	718.2	33000.0	0.0	47.0	47.0	66555.0	5.0	0.0	0.0	25474.0	0.0	0.0	3929.0
225.0	606.2	713.8	34000.0	4.0	47.0	47.0	66555.0	1.0	0.0	0.0	25474.0	0.0	0.0	0.0

Fig. (3.14): The example table of data

CHAPTER (4)

“Conclusions”

Chapter (4): Conclusions

Throughout the overall work on this interface model, the following notes are presented as the final conclusions:-

1. RAP experiment performs the Measurment of the temperature dependence of the *Therrmo-Acoustic Model* [4]. The source of the experimental beam is the BTF from the DAFNE accelerator.
2. DAFNE and RAP control systems manage and monitor the machine and detector elements through a set of low-level programs.
3. The high-levels of the two systems are strongly connected and integrated; the information coming at different times from the detector and machine controls are collected, synchronized, stored, preanalyzed and displayed from a number of tools, based on a common general database accessed mainly by the two supervisor Web sites and by dedicated programs.
4. The resulting measured data are saved in a text file format with an extension referring to the type of collected data.
5. These data files are saved on the DAFNE server accoridng to a certain file type (fast, slow,raw,etc…) in the file system structure[4] which represents it. These types are included as file extension.
6. As mentioned in chapter (3), PHP runs on the remote web server and processes the webpages before they are sent to the browser. This makes it a server-side scripting language [7] [8]. The fact that it runs on the server has several benefits, and some drawbacks. Lets take the benefits first:
   a. On the server it's possible to have access to things like a database. This means that we can make a script that sorts through large amounts of data, without the client having to download them first.
   b. Only the output from the script is sent to the client, not the script itself. This means that the script is invisible from the end-user which makes PHP scripts browser-neutral;

they don't depend on some capability of the browser. It also means that there is minor need to worry that the script might be stolen.

c. It is possible to make other programs for use in PHP scripts. Like implementing part of the script in C, and then call the program from the script to make it run faster. PHP is a parsed language, meaning that there are no compiled binaries. Every time a page with PHP-code, is requested, the parser looks through the page and executes any PHP-statements finds.

On the other hand, there are also some drawbacks:

a. By executing everything on the server, more strain is put on it. With many concurrent requests, and large complex scripts, the server might not be able to handle it. But in most cases, this isn't a real concern because the parser in PHP is rapid. And if the server still can't cope with the number of visitors, then there will be a real demand for a bigger server.

b. The pages can't do anything themselves and the server is required to do the whole job. This means that the pages will lose some of their functionality if the site visitors decide to save them to their computer.

8. Concerning the presented JpGraph model, it demonstrated the capability to show various graph plots with a noteable accuracy between diverse control data variables without the limitation of having only particular variables as part of the graph structure.

9. Since this interface is designed in PHP, it needs special server configurations to enable passing of parameters from one file to another. In some cases however, this might cause various security vulnerabilities.

10. For future upgrading purposes, the following notes are recommended:

a. The written code can be easily modifed for future improvents depending upon the demands for graph display developments. For example, it is possible to add other plot type classes, add or modify time intervals, plot scales, etc...

b. Unless another graph is generated (in which the previous graph image is destroyed), a saved image PNG file containing the last generated graph is kept on the JpGraph cache directories (jpgraph_cache, and csim_cache).This could be a good material that can be utilized for other purposes

Appendices

Appendix (A): The RAP Experiment data format

A.1 DAFNE .raw format. Updated every 15 seconds

1. time [UNIX seconds]
2. e- current [mA]
3. e+ current [mA]
4. IP1 luminosity [cm^{-2}s^{-1}]
5. IP1 lum. Int. [nbarn-1]
6. IP2 luminosity [cm^{-2}s^{-1}]
7. IP2 lum. Int. [nbarn^{-1}]
8. bunch 1-32 word*
9. bunch 33-64 word
10. bunch 65-96 word
11. bunch 97-120 word
12. Timing word word
13. Acc. pulse #
14. L0 IP1 [cm^{-2}s^{-1}]
15. I0 e+ IP1 cur [mA]
16. I0 e- IP1 cur [mA]
17. L0 IP2 [cm^{-2}s^{-1}]
18. I0 e+ IP2 cur [mA]
19. I0 e- IP2 cur [mA]
20. ms from start RUN [msec]
21. RF frequency [Hz]
22. Roundness e- [$\int$y/ $\int$x]
23. Roundness e+ [$\int$y/ $\int$x]
24. KLOE field [Gauss]
25. X BPMEL204 [mm]
26. Y BPMEL204 [mm]
27. X BPMEL205 [mm]
28. Y BPMEL205 [mm]
29. X BPMEL206 [mm]
30. Y BPMEL206 [mm]
31. X BPMEL207 [mm]
32. Y BPMEL207 [mm]

*) the bunch word in this format as well as in the KLOE fast format is 1 if the bunch is filled. The most significant bit is the first bunch.

A.2 DAFNE .dat format. Updated every 15 seconds

1. Time [UNIX seconds]
2. e- current [mA]
3. e+ current [mA]
4. Lum1 (IP1 or e+) rate [Hz]
5. Lum2 (IP2 or e-) rate [Hz]
6. Linac mode -1 electrons
   +1 positrons
7. Number of e- bunches #
8. e- bunch 1-32 word bit 32-1
9. e- bunch 33-64 word bit 32-1
10. e- bunch 65-96 word bit 32-1
11. e- bunch 96-120 word bit 32-8
12. Number of e+ bunches #
13. e+ bunch 1-32 word bit 32-1
14. e+ bunch 33-64 word bit 32-1
15. e+ bunch 65-96 word bit 32-1
16. e+ bunch 96-120 word bit 32-8
17. Status e- 0 no beam
   1 acc inject
   2 main ring inject
   3 stored beam
   4 colliding
18. Status e+ 0 no beam

1 acc inject
2 main ring inject
3 stored beam
4 colliding

19. Status DAFNE -3 simulated data
   -2 run off
   -1 unknown
   0 standby
   1 e- inject
   2 e+ inject
   3 e- stored
   4 e+ stored
   5 filled
   6 colliding
20. Fill number #
21. e- lifetime [s] -1 not available
   0 unstable
22. e+ lifetime [s] -1 not available
   0 unstable
23. Lum1 (IP1or e+) [cm-2 s-1]/1e28
24. Lum2 (IP1or e-) [cm-2 s-1]/1e28
25. Interaction flag 0 not colliding
   1 colliding @ IP1
   2 colliding @ IP2

3 colliding @ IP1-IP2

26. RF frequency [Hz]
27. Roundness e- [ʃy/ ʃx]
28. Roundness e+ [ʃy/ ʃx]
29. KLOE field [Gauss]

A.3 DAFNE DMCV, daily most common variable, files format

1. Time [UNIX seconds] 60 seconds
2. e- current [mA] DCTEL001 60 seconds
3. e+ current [mA] DCTPS001 60 seconds
4. IR1 luminosity e+ [cm-2 s-1] DLM00001 60 seconds
5. IR1 rate e+ [Hz] DLM00001 60 seconds
6. IR1 luminosity e- [cm-2 s-1] DLM00002 60 seconds
7. IR1 rate e- [Hz] DLM00002 60 seconds
8. e- number of bunch # 60 seconds
9. e+ number of bunch # 60 seconds
10. MR vacuum IP1 [torr] VUGI1001 300 seconds
11. MR vacuum IP2 [torr] VUGI2001 300 seconds
12. MR vacuum e+ [torr] VUGPL101 300 seconds
13. MR vacuum e+ [torr] VUGPL102 300 seconds
14. MR vacuum e+ [torr] VUGPL103 300 seconds
15. MR vacuum e+ [torr] VUGPS101 300 seconds
16. MR vacuum e+ [torr] VUGPS102 300 seconds
17. MR vacuum e+ [torr] VUGPS103 300 seconds
18. MR vacuum e+ [torr] VUGPS104 300 seconds
19. MR vacuum e+ [torr] VUGPS105 300 seconds
20. MR vacuum e+ [torr] VUGPS201 300 seconds
21. MR vacuum e+ [torr] VUGPS202 300 seconds
22. MR vacuum e+ [torr] VUGPS203 300 seconds
23. MR vacuum e+ [torr] VUGPL201 300 seconds
24. MR vacuum e+ [torr] VUGPL202 300 seconds
25. MR vacuum e+ [torr] VUGPL203 300 seconds
26. MR vacuum e- [torr] VUGEL101 300 seconds
27. MR vacuum e- [torr] VUGEL102 300 seconds
28. MR vacuum e- [torr] VUGEL103 300 seconds
29. MR vacuum e- [torr] VUGES101 300 seconds
30. MR vacuum e- [torr] VUGES102 300 seconds
31. MR vacuum e- [torr] VUGES103 300 seconds
32. MR vacuum e- [torr] VUGES201 300 seconds
33. MR vacuum e- [torr] VUGES202 300 seconds
34. MR vacuum e- [torr] VUGES203 300 seconds
35. MR vacuum e- [torr] VUGEL201 300 seconds
36. MR vacuum e- [torr] VUGEL202 300 seconds
37. MR vacuum e- [torr] VUGEL203 300 seconds

A.4 KLOE .fast format. Updated every 15-30 seconds

1. Seconds since midnight [s]
2. current e- [mA]
3. current e+ [mA]
4. Luminosity monitor IP1 counts [Hz]
5. Luminosity monitor IP2 counts [Hz]
6. Number of bunch e- #
7. Number of bunch e+ #
8. Fill number #
9. DAFNE status
   - -3 simulated data
   - -2 run off
   - -1 unknown
   - 0 standby
   - 1 e- inject
   - 2 e+ inject
   - 3 e- stored
   - 4 e+ stored
   - 5 filled
   - 6 colliding
10. Not used
11. CAENET (packed)
    - BIT 0 HVEMC
    - BIT 1 EMC
    - BIT 2 HVDC
    - BIT 3 DC
    - BIT 4 DAQ
    - BIT 5 DC CURRENT
12. Trigger RUN number #
13. ECM HV state
    - -2 = HV UNDEFINED
    - -1 = WRONG READOUT
    - 0 = HV ON
    - 1 = ECA OFF
    - 2 = BAR OFF
    - 3 = ECB OFF
    - 4 = QCAL OFF
    - 5 = HV OFF
14. ECM LV state
    - -1 = WRONG READOUT
    - 0 = LV OK
    - 1 = PULSING

2 = BAD THRESHOLDS

15. Trigger luminosity [/10^28 cm^{-2}s^{-1}]
16. Trigger number of Bhabha #
17. DC HV state
    - -1 = WRONG READOUT
    - 0 = HV ON
    - 1 = STANDBY
    - 2 = HV OFF
    - 3 = RUMPING UP
    - 4 = RUMPING DOWN
18. DC LV state
    - -1 = WRONG READOUT

0 = LV OK
1 = PULSING
2 = BAD THRESHOLD
3 = BAD WIDTH
4 = BAD DEAD TIME

19. Lifetime e- [s]
20. Lifetime e+ [s]
21. DAQ crates state -1 = WRONG READOUT
    0 = ALL CRATE ON
    1 = ONE OR SAME CRATE OFF
22. L3 run number #
23. Not used
24. L3 luminosity [/10^28 cm-2s-1]
25. KLOE run state -1 NOT RUNNING
    0 NOT READY
    1 READY
    2 PAUSED
    3 RUNNING
26. KLOE RUN number #
27. L3 number of Bhabha #
28. RUN type -1 NOT READY
    0 NORMAL
    1-5 CALIBRATION
29. RUN on disk 0 TRASH
    1 DISK
30. Number of farms #
31. DC trigger level 1 [Hz]
32. DC trigger level 2 [Hz]
33. QCAL coincidence [Hz]
34. QCAL A [Hz]
35. QCAL B [Hz]
36. T2 yes [Hz]
37. T1 free [Hz]
38. QCAL coincidence delayed [Hz]
39. QCAL Bhabha delayed [Hz]
40. QCAL Bhabha [Hz]
41. TRG integrated luminosity [nbarn^{-1}]
42. L3 integrated luminosity [nbarn^{-1}]
43. pe1 e- bunch pattern word
44. pe2 word
45. pe3 word
46. pe4 word
47. pp1 e+ bunch pattern word
48. pp2 word
49. pp3 word
50. pp4 word
51. Run Size [GByte]
52. Event Size [byte]

53. ECM1 [Hz]
54. ECM2 [Hz]
55. ECM3 [Hz]
56. ECM4 [Hz]

A.5 KLOE .slow format. Updated every 45 seconds

1. Seconds from midnight [s]
2. Vacuum at IP 1 (KLOE) [torr]
3. Vacuum at IP 2 (DEAR/FINUDA) [torr]

4. Magnet status	0 cold
	1 undefined
	2 300-77 K
	3 77-4 K
	4 warm map

5. Magnet current [A]
6. Magnet Helium percentage in vessel [%]
7. Magnet coil 1 temperature [K]
8. Magnet coil 2 temperature [K]
9. Magnet coil 3 temperature [K]
10. Magnet coil 4 temperature [K]

11. Gas status	0 working
	1 warning
	2 alarm
	-1 unknown

12. Gas mode	0 standby
	1 closed mode
	2 calibrate
	3 shutdown
	4 open mode (standard)
	5 zero adjust
	6 manual
	-1 unknown

13. gas flow [slm] (standard liters/minute)
14. Atmospheric pressure [mbar]
15. Gas temperature [Celsius]
16. Gas isobutene percentage [%]
17. Gas oxygen content [ppm]
18. Gas water content [ppm]
19. Absolute pressure (side A) [mbar]
20. Absolute pressure (side B) [mbar]
21. Differential pressure (side A) [mbar]
22. Differential pressure (side B) [mbar]
23. Pressure of Helium inlet [mbar]
24. Pressure of isobutene inlet [mbar]
25. Pressure of argon inlet [mbar]
26. TRKMON run number #
27. Beam position x [cm]

28. Beam position y [cm]
29. Beam position z [cm]
30. Beam width x [mm]
31. Beam width y [mm]
32. Beam width z [mm]
33. √ momentum x [MeV/c]
34. √ momentum y [MeV/c]
35. √ momentum z [MeV/c]
36. CALMON run number #
37. ©© endcap energy [MeV]
38. ©© barrel energy [MeV]
39. Bhabha endcap energy [MeV]
40. Bhabha barrel energy [MeV]
41. COSMON run number #
42. DCNOISE run number #
43. Not used
44. Scraper e- LONG IP2 up [mm]
45. Scraper e- LONG IP2 down [mm]
46. Small cells average voltage [V]
47. Big cells average voltage [V]
48. Number of tripped channels #
49. Number of over current channels #
50. Number of channels off #
51. Current of DC sector 1 [∝A]
52. Current of DC sector 2 [∝A]
53. Current of DC sector 3 [∝A]
54. Current of DC sector 4 [∝A]
55. Current of DC sector 5 [∝A]
56. Current of DC sector 6 [∝A]
57. Current of DC sector 7 [∝A]
58. Current of DC sector 8 [∝A]
59. Current of DC sector 9 [∝A]
60. Current of DC sector 10 [∝A]
61. Current of DC sector 11 [∝A]
62. Current of DC sector 12 [∝A]
63. Current of DC sector 13 [∝A]
64. Current of DC sector 14 [∝A]
65. Current of DC sector 15 [∝A]
66. Current of DC sector 16 [∝A]
67. Not used
68. Acci. Clusters >7 MeV WEST endcap [Hz]
69. Acci. clusters >7 MeV EAST endcap [Hz]
70. Accidental clusters >7 MeV barrel [Hz]
71. t-r/c endcap-endcap [ns]
72. t-r/c barrel-barrel [ns]
73. t-l/v endcap-endcap [ns]
74. t-l/v barrel-barrel [ns]

75. Scraper e+ SHORT IP2 in [mm]
76. Scraper e+ SHORT IP2 out [mm]
77. Scraper e+ LONG IP2 up [mm]
78. Scraper e+ LONG IP2 down [mm]
79. Scraper e- SHORT IP2 in [mm]
80. Scraper e- SHORT IP2 out [mm]
81. Scraper e+ LONG IP1 in [mm]
82. Scraper e+ LONG IP1 out [mm]
83. Scraper e- LONG IP1 in [mm]
84. Scraper e- LONG IP1 out [mm]
85. Noise layer 1, sector 1 [Hz]
86. Noise layer 1, sector 2 [Hz]
87. Noise layer 1, sector 3 [Hz]
88. Noise layer 1, sector 4 [Hz]
89. Noise layer 5, sector 1 [Hz]
90. Noise layer 5, sector 2 [Hz]
91. Noise layer 5, sector 3 [Hz]
92. Noise layer 5, sector 4 [Hz]
93. Noise layer 10, sector 1 [Hz]
94. Noise layer 10, sector 2 [Hz]
95. Noise layer 10, sector 3 [Hz]
96. Noise layer 10, sector 4 [Hz]

A.6 *DEAR .dat format. Updated every 2 minutes when DEAR data acquisition is running*

1. Time [UNIX seconds]
2. Time [hh:mm:ss]
3. Daily integrated luminosity [nbarn^{-1}]
4. Luminosity [cm^{-2}s^{-1}]
5. Time between luminosity measures [s]
6. Number of kaons since beginning of RUN #
7. Elapsed run time [s]
8. Integrated luminosity since the begin of the RUN [nbarn^{-1}]
9. Rate of coincidence vetoed by RF/4 [Hz]
10. Rate of coincidence [Hz]
11. Rate in inner scintillator [Hz]
12. Rate in outer scintillator [Hz]

A.7 *DAFNE estimated luminosity (IP1 is e+ illuminometer placed in the KLOE interaction region, IP2 is the e+ illuminometer placed in DEAR region, but can be moved at IP1 as e- illuminometer)*

1. Time [UNIX seconds]
2. IP1 luminosity estimated [cm^{-2} s^{-1}]
3. IP2 luminosity estimated [cm^{-2} s^{-1}]
4. Colliding flag 0 not colliding
   1 colliding @ IP1
   2 colliding @ IP2
   3 colliding @ IP1 and IP2

A.8 *DAFNE slow elements plain files. The most useful variable of elements logged in binary files every 5 minutes are restored in plain text files.*

1. Time [UNIX seconds]
2. IP1 vacuum [torr]
3. IP2 vacuum [torr]
4. Scraper EL201 up [mm]
5. Scraper EL201 down [mm]
6. Scraper PS201 inner [mm]
7. Scraper PS201 outer [mm]
8. Scraper PL201 up [mm]
9. Scraper PL201 down [mm]
10. Scraper ES201 inner [mm]
11. Scraper ES201 outer [mm]
12. Scraper PL101 inner [mm]
13. Scraper PL101 outer [mm]
14. Scraper EL201 inner [mm]
15. Scraper EL201 outer [mm]

A.9 DAFNE daily files format. Daily updated

1. Time [UNIX seconds]
2. e- integrated current [Ah]
3. e+ integrated current [Ah]
4. Time e- stored (timing status) [h]
5. Time e+ stored (timing status) [h]
6. Time standby (timing status) [h]
7. Time filled (Ip >1 or Ie>1) [h]
8. Time delivering (Ip>150 and Ie>150) [h]
9. Daily storing time [%]

A.10 KLOE daily files format. Daily updated

1. Time [UNIX seconds]
2. Daily integrated lumi [nbarn^{-1}]
3. Daily delivered lumi [nbarn^{-1}]
4. Daily running time (on storing time) [%]
5. Daily running time (on 24 h) [%]
6. Daily peak lumi [cm^{-1}s^{-1}]
7. Daily average lumi [cm^{-2}s^{-1}]
8. Daily logging time [%]
9. luminosity/counts ratio [cm^{-2}s^{-1}Hz^{-1}]

Appendix (B): The complete graph model code

1) *The* **"Enquiry Form program"** *code:*

```
<html>
<head>
<title>Graph Form</title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-
8859-1">
</head>
<body bgcolor="#DCDCDC">
<pre>
<center>
<table border = 1 cellpadding =2 cellspacing =6>
<tr><img src="pictures/RAPsma.gif" width=50 height=30 hspace=2
vspace=2 border=0></td>
<td align = center bgcolor="#FFFAF0"><a
href="http://www.lnf.infn.it/acceleratori/">(DAFNE)</a></td>
<td align = center bgcolor="#FFFAF0"><a
href="http://www.lnf.infn.it/acceleratori/btf/">(BTF)</a></td>
<td align = center bgcolor="#FFFAF0"><a
href="http://www.lnf.infn.it/esperimenti/rap">(RAP)</a></td>
<td align = center bgcolor="#FFFAF0"><a
href="http://www.roma1.infn.it/rog/">(ROG)</a></td>
<td align = center bgcolor="#FFFAF0"><a
href="http://dafne.lnf.infn.it/">(DAFNE)</a>(LNF-domain only)</td>
<td align = center bgcolor="#FFFAF0"><a
href="http://kloeslow.lnf.infn.it:8000/">(KLOE)</a>(LNF-domain
only)</td>
<td align = center bgcolor="#FFFAF0"><a
href="http://www.lnf.infn.it/">(LNF)</a> - <a
href="http://www.infn.it/">(INFN)</a></td></tr></table>

<table border="1" cellspacing="5" cellpadding="8">
<tr><td align = center></td></tr>
<tr><td align = center></td>
<td align = center bgcolor="#FFFAF0"><font SIZE=+2 color='#8B2323'
face='arial,helvetica'><b>GRAPH DISPLAY MODULE</b></font></td>
<td align = center></td></tr>
<tr><td align = center ></td>
<?php

#definition of the colors used in the form
$blue = '#104E8B';
$blue4 = '#00008B';
$skyblue = '#00BFFF';
$brown = '#8B2323';
$darkgrey = '#A9A9A9';
$gray = '#D4D4D4';
```

```
$red = '#DC143C';
$white = '#FFFAF0';
$default_dir = '/var/www/html/exercises/data';         #assigning
the data source directory
$dir = dir ($default_dir);

echo "<td align = center bgcolor=$white><table border =3
cellpadding =4 cellspacing =8 bgcolor= $gray>";
#echo "<tr><td align = center colspan = 4 bgcolor=$darkgrey><font
SIZE=+3 color='$brown' face='arial,helvetica'><b>GRAPH DISPLAY
MODULE</b></font></td></tr>";

# The directory open engine

#the returned dir name necessary to file open

$dirname = $_REQUEST ['dirname'];

#extracting the directory name to display in the selection list

$bdir = basename ($dirname);

echo "<FORM ACTION = '$self_php' METHOD = 'post'>";
echo "<tr><td><font SIZE=+1 color='$blue4'
face='arial,helvetica'><b>Directory Name</b></font></td>";
echo "<td><SELECT NAME='dirname' OnChange='submit();'>";
echo "<OPTION>Select...</OPTION>";
     traverse_dir ($default_dir);
echo "</SELECT></td>";
echo "</FORM>";

# here begins the second part of the form to open the file and
choose graph options

echo "<FORM ACTION='graph.php' METHOD='post' target='display'>";
echo "<INPUT TYPE='hidden' NAME= 'dirname' VALUE='$dirname'>";
if ($dirname!= null){
echo "<td><font SIZE=+1 color='#00008B'
face='arial,helvetica'><b>File Name</b></font></td>";
echo "<td  ><SELECT NAME='filename'>";
echo "<OPTION SELECTED >Select...</OPTION>";
show_files ($dirname);
echo "</SELECT></td>";

# The no. of plots to be shown on the graph settings
echo "<tr><td><font SIZE=+1 color='$white'
face='arial,helvetica'><b>No.of Plots</b></font></td>";
echo "<p><td><INPUT type='radio' name='nplots' value='1'
checked>One Plot</td>";
echo "<td><INPUT type='radio' name='nplots' value= '2' >Two
Plots</td>";

# The plot-scale-type option settings
```

```
echo "<tr><td><font SIZE=+1 color='$white'
face='arial,helvetica'><b>Plot Type</b></font></td>";
echo "<p><td><INPUT type='radio' name='ptype' value='linlin'
checked>Linear Plot</td>";
echo "<td><INPUT type='radio' name='ptype' value='linlog' >semi-
Log Plot</td>";
echo "<td><INPUT type='radio' name='ptype' value='loglog' >Log
Plot</td>";

#the X-axis input variable settings
echo "<p><tr><td><font SIZE=+1
color='#218868'face='arial,helvetica'><b>X-axis
variables</b></font></td>";

echo "<td align = center colspan = 2><SELECT NAME='data1'>";
echo "<OPTION SELECTED VALUE = ' '>Select...</OPTION>";
$n_file = file ("$dirname/var.dat");
foreach ($n_file as $key=>$val){
     $rkey =$key + 1 ;
     echo "<OPTION VALUE=$key>$rkey.$val</OPTION><br>";}
echo "</SELECT></td></tr>";

#the Y-axis input variable settings
echo "<p><tr><td><font SIZE=+1 color='$blue'
face='arial,helvetica'><b>Y-axis variables</b></font>";

echo "<td align = center colspan = 2><SELECT NAME='data2'>";
echo "<OPTION SELECTED VALUE = ' '>Select...</OPTION>";

$n_file = file ("$dirname/var.dat");
foreach ($n_file as $key=>$val){
     $rkey =$key + 1 ;
     echo "<OPTION VALUE=$key>$rkey.$val</OPTION><br>";}
echo "</SELECT></td></tr>";

#the Y2-axis input variable settings (only if the option is
chosen)
echo "<p><tr><td><font SIZE=+1 color='$red'
face='arial,helvetica'><b>y2-axis variables<b></font>";
echo "<td align = center colspan = 2><SELECT NAME='data3'>";
echo "<OPTION SELECTED VALUE = ' '>Select...</OPTION>";
$n_file = file ("$dirname/var.dat");
foreach ($n_file as $key=>$val){
     $rkey =$key + 1 ;
     echo "<OPTION VALUE=$key>$rkey.$val</OPTION><br>";}
echo "</SELECT></td></tr>";

#The readings interval settings (every: minute,hour)
echo "<p><u><tr><td><font SIZE=+1 color='$white'
face='arial,helvetica'><b>Readings Interval</b></u></td>";
```

```
echo "<td><INPUT type='radio' name='pinterval' value='4'
checked>minute</td>";
echo "<td><INPUT type='radio' name='pinterval'
value='240'>Hour</td></tr>";

#Setting the submit button

echo "<tr><TABLE align= bottom bgcolor= $white><td align = center
><INPUT TYPE='SUBMIT' NAME='STATUS' VALUE='Show Plots'>";
#Setting the reset button
echo "<td align = center ><INPUT TYPE='RESET' NAME='RESET'
VALUE='Reset Form'></FORM></td>";
echo "<FORM ACTION='datashow.php' METHOD='post' target='_blank'>";
echo "<INPUT TYPE='hidden' NAME= 'dirname' VALUE='$dirname'>";
echo "<INPUT TYPE='hidden' NAME= 'dirname' VALUE='$filename'>";
echo "<td align = center><INPUT TYPE='SUBMIT' NAME='Data'
VALUE='All data'></FORM></td></tr>";

#echo "<FORM ACTION='seldata.php' METHOD='post' target='_blank'>";
#echo "<INPUT TYPE='hidden' NAME= 'dirname' VALUE='$dirname'>";
#echo "<INPUT TYPE='hidden' NAME= 'dirname' VALUE='$filename'>";
#echo "<INPUT TYPE='hidden' NAME= 'xtitle' VALUE='$data1'>";
#echo "<INPUT TYPE='hidden' NAME= 'ytitle' VALUE='$data2'>";
#echo "<INPUT TYPE='hidden' NAME= 'y2title' VALUE='$data3'>";
#echo "<td align = center><INPUT TYPE='SUBMIT' NAME='Data'
VALUE='Selected data'></FORM></td></TABLE></tr>";
}
echo "</TABLE>";


#__________"definitions of used functions "_______________________

# This function reads and sorts datafiles within the selected
directory
function show_files($dir){
if(!($dp =opendir ($dir))) die("cannot open $dir.");
while ($file = readdir($dp)){
     if($file != "var.dat"){       # if file name is different
than the input variables file
     $filenames[] = $file;
               }
             }
closedir ($dp);
sort ($filenames);       #sorts the read files

for ($i=0; $i < count ($filenames); $i++){
     if ($filenames[$i] !='.' && $filenames[$i] !='..') {
               echo "<OPTION
VALUE='$dir/$filenames[$i]'>$filenames[$i]</OPTION>";
          }
 }
 }

# This function navigates through the directory heirarchy reading
them all
function traverse_dir($dir) {
     chdir ($dir);
     if(!($dp = opendir($dir))) die("can't open $dir.");
          while($file = readdir($dp)) {
          if(is_dir($file)) {
               if($file != '.' && $file != '..'){
                    echo "<OPTION
VALUE='$dir/$file'>$file</OPTION><br>";
                    traverse_dir("$dir/$file");   #keep reading
within the directory heirarchy
                    chdir($dir);
                    }
               }

               else echo "$file<br>";
          }
          closedir($dp);
     }

?>
<tr><td>      </td></tr>
</table>
<table border="1" cellpadding =4 cellspacing =8>
<tr>
<td align = center bgcolor="#FFFAF0"><address> <a
href="http://www.lnf.infn.it/"> Laboratori Nazionali di Frascati
</a>
<a href="http://www.infn.it/"> Istituto Nazionale di Fisica
Nucleare </a></address>
<address> P.O.Box 13,00044-Frascati (Rome - Italy)</address>
<address> Phone (+39 06 9403 2411-8064) Fax (+39 06 9403
2256)</address></td>
</tr>
</table>
</center>
</pre>
</body>
</html>
```

2) *The* **“main graph”** *program code:*

```
<html>
<header><title>graph</title></header>
<body>
This is the graph which shows the overall variations every minute
!!
<p>
<center>
<?php
include ("./jpgraph/jpgraph.php");
include ("./jpgraph/jpgraph_line.php");
include "./jpgraph/jpgraph_scatter.php";
include ("./jpgraph/jpgraph_regstat.php");
include("./jpgraph/jpgraph_log.php" );

$graph = "./csimcache/graph.png";
$html = "./csimcache/graph_csim_.html";
if (file_exists($graph)){ unlink($graph);}
if (file_exists($graph)){ unlink($html);}
$dirname = $_REQUEST ['dirname'];
$filename = $_REQUEST ['filename'];
$nplots = $_REQUEST ["nplots"];
$ptype = $_REQUEST ["ptype"];
$xkey = $_REQUEST ["data1"];
$ykey = $_REQUEST ["data2"];
$y2key = $_REQUEST ["data3"];
$pinterval = $_REQUEST ['pinterval'];

$file = 0;
$rcount = 0;
$t_interval = 0;
$t_count = 0;

$file_h = fopen("$filename", "r"); # open the specified file
containing data
$date = d_date ($filename);

# to read the lines from the specified file
while (!feof($file_h)) {
     $file = fgets($file_h);  # to see only the last line
     $file_arr = preg_split("/\s/",$file,-1,PREG_SPLIT_NO_EMPTY);
##$secondi = $file_arr[0];
#    if ($secondi>31000 and $secondi<32000)
#    {
     if ($t_count==$t_interval) # to read only the lines according
to the assigned interval
     {
     for ($key=0; $key < count($file_arr) ; $key += 1 ) {
          $value = $file_arr[$key];
          $accdata[$key][$rcount] = round ($value);  # read all
column elements in an array
          }
     $t_interval = $pinterval;
     $rcount++;
     $t_count=0;                   #counts the total no. of rows
     }

     $t_count++;
#    }
}
fclose($file_h);
#to read the variables file in order to display the axis-titles
$n_file = file ("$dirname/var.dat");

#the  garph generating part

$xdata = $accdata[$xkey];
$xtitle = $n_file[$xkey];
$ydata = $accdata[$ykey];
$ytitle = $n_file[$ykey];
$y2data = $accdata[$y2key];
$y2title = $n_file[$y2key];

// Create the graph. These two calls are always required
$graph = new Graph(700,300,"auto",-1);
$graph->CheckCSIMCache("graph",1);
$graph->SetScale("$ptype");

$graph->SetMarginColor('azure2');
$graph->ygrid->Show();
$graph->xgrid->Show();
$graph->SetGridDepth(DEPTH_FRONT);
$graph->img->SetMargin(60,60,30,70);
$graph->title->Set("$date");

// Adjust the legend position
$graph->legend->Pos(0.25,0.98,"center","bottom");
$graph->legend->SetLayout(LEGEND_HOR);

#to turn the time fromsecnds into hours

if($xkey == 0){
foreach ($xdata as $index=>$value){

$s = round ($value/3600);
   $c.=$s.',';
  }
 $x1data = preg_split("/,/",$c,-1,PREG_SPLIT_NO_EMPTY);

}else {$x1data = $xdata;}
```

```
foreach ($xdata as $value) {
     if ($xdata >= 100 || $xdata < 0 ){$graph->xaxis-
>Settitlemargin(10);}else break;}
$graph->xaxis->SetColor("darkgreen");
$graph->xaxis->SetTickLabels($x1data);
$graph->xaxis->SetLabelFormat("%d");
$graph->xaxis->SetWeight(2);
$graph->xaxis->SetFont(FF_FONT1,FS_NORMAL,12);
$graph->xaxis->title->Set("$xtitle");
$graph->xaxis->title->SetColor("darkgreen");
$graph->xaxis->title->SetFont(FF_FONT1,FS_BOLD,12);
$graph->xaxis->SetTextLabelInterval(2);

foreach ($ydata as $value) {
     if ($ydata >= 100){$graph->yaxis->Settitlemargin(30);}else
break;}
$graph->yaxis->SetColor("blue");
$graph->yaxis->SetWeight(2);
$graph->yaxis->title->SetColor("blue");
$graph->yaxis->SetFont(FF_FONT1,FS_NORMAL,12);
$graph->yaxis->title->Set("$ytitle");
$graph->yaxis->title->SetFont(FF_FONT1,FS_BOLD,14);
$graph->yaxis->SetTextLabelInterval(2);
$graph->yaxis->Settitlemargin(30);
// Create the first linear plot
$splot1=new ScatterPlot($ydata);
$splot1->mark->SetType(MARK_FILLEDCIRCLE);
$splot1->mark->SetFillColor("blue");
$splot1->mark->SetWidth(1);
$splot1->mark->SetColor('blue@0.1');
$splot1->SetLegend ($ytitle);

// Add the first line plot to the graph
$graph->Add($splot1);

if ($nplots == 2){
// Create the second linear plot
$pt = substr ($ptype,-3);

$graph->SetY2Scale("$pt");

$splot2=new ScatterPlot($y2data);
$splot2->mark->SetType(MARK_FILLEDCIRCLE);
$splot2->mark->SetFillColor("red");
$splot2->mark->SetWidth(1);
$splot2->mark->SetColor('red@0.1');
$splot2 ->SetLegend("$y2title");

foreach ($y2data as $value) {
     if ($y2data >= 100){$graph->y2axis->Settitlemargin(38);}else
break;}
$graph->y2axis->SetColor("red");
$graph->y2axis->SetWeight(2);
$graph->y2axis->SetFont(FF_FONT1,FS_NORMAL,12);
$graph->y2axis->title->Set($y2title);
$graph->y2axis->title->SetColor("red");
$graph->y2axis->title->SetFont(FF_FONT1,FS_BOLD,14);
//setting y-axis label interval
$graph->y2axis->SetTextLabelInterval(2);

// Add the second line plot to the graph
$graph->AddY2($splot2);
}

// Display the graph
$graph->StrokeCsim(basename( __FILE__));



echo "<p><FORM ACTION='graphdetails.php' METHOD='post'
target='_blank'>";
echo "<INPUT TYPE='hidden' NAME= 'd_dirname' VALUE='$dirname'>";
echo "<INPUT TYPE='hidden' NAME= 'f_filename' VALUE='$filename'>";
echo "<INPUT TYPE='hidden' NAME= 'date' VALUE='$date'>";
echo "<INPUT TYPE='hidden' NAME= 'datas1' VALUE='$data1'>";
echo "<INPUT TYPE='hidden' NAME= 'datas2' VALUE='$data2'>";
echo "<INPUT TYPE='hidden' NAME= 'datas3' VALUE='$data3'>";
echo "<INPUT TYPE='hidden' NAME= 'n_plots' VALUE='$nplots'>";
echo "<INPUT TYPE='hidden' NAME= 'p_type' VALUE='$ptype'>";
echo "<INPUT TYPE='hidden' NAME= 'xtitle' VALUE='$xtitle'>";
echo "<INPUT TYPE='hidden' NAME= 'ytitle' VALUE='$ytitle'>";
echo "<INPUT TYPE='hidden' NAME= 'y2title' VALUE='$y2title'>";
echo "<table border = 1 cellpadding =2 cellspacing =6
bgcolor='#DCDCDC'>";
echo "<tr><td align = center  colspan='6'>From &nbsp;";
echo "<INPUT TYPE='text' size='6' NAME= 'hour_start'>";
echo "&nbsp;o'clock";
echo "&nbsp; to &nbsp;";
echo "<INPUT TYPE='text' size='6' NAME= 'hour_end'>";
echo "&nbsp;o'clock</td>";
echo "<td align = center colspan='1'><INPUT TYPE='SUBMIT'
VALUE='detailed Plot'></td></tr>";
echo "<tr><td align = center colspan='3'>";
echo "<INPUT TYPE='checkbox' NAME= 'step' VALUE='1'>Step Function
Display</td>";
echo "<td></td>";
echo "<td align = left colspan=2><INPUT TYPE='checkbox' NAME=
'fill' VALUE='1'>Filled Graph</td>";
echo "</FORM>";
echo "<FORM ACTION='seldata.php' METHOD='post' target='_blank'>";
echo "<INPUT TYPE='hidden' NAME= 'dirname' VALUE='$dirname'>";
echo "<INPUT TYPE='hidden' NAME= 'filename' VALUE='$filename'>";
echo "<INPUT TYPE='hidden' NAME= 'data_1' VALUE='$data1'>";
echo "<INPUT TYPE='hidden' NAME= 'data_2' VALUE='$data2'>";
echo "<INPUT TYPE='hidden' NAME= 'data_3' VALUE='$data3'>";
echo "<INPUT TYPE='hidden' NAME= 'xtitle' VALUE='$xtitle'>";
echo "<INPUT TYPE='hidden' NAME= 'ytitle' VALUE='$ytitle'>";
echo "<INPUT TYPE='hidden' NAME= 'y2title' VALUE='$y2title'>";
echo "<td align = center><INPUT TYPE='SUBMIT' NAME='Data'
VALUE='Selected data'></FORM></td></tr>";
echo "</table>";


function d_date ($path){
$nfile = basename ($path);
$tok = strtok($nfile,".");
$year = substr("$tok", 0, 4);
$month = substr("$tok", 4, 2);
$fday = substr("$tok",-2);
if (ereg("~",$fday,$reg)) {
     $t = strtok($fday,"~");
     $day = "0$t";
} else $day = $fday;

return $date = ("Measurement Date:($day/$month/$year)");
}
?>
</center>
</body>
</html>
```

3) *The* **“Detailed graph”** *program code:*

```
<?php
#include the necessary JpGraph files to generate the graph
include ("./jpgraph/jpgraph.php");
include ("./jpgraph/jpgraph_line.php");
include ("./jpgraph/jpgraph_regstat.php");
include("./jpgraph/jpgraph_log.php" );

$graph = "./csimcache/graphdetails.png";
$html = "./csimcache/graphdetails_csim_.html";
if (file_exists($graph)){ unlink($graph);}

if (file_exists($graph)){ unlink($html);}

#passed data from the form
$d_dirname= $_REQUEST["d_dirname"];
$f_filename= $_REQUEST["f_filename"];
$date = $_REQUEST["date"];
$hour_start = $_REQUEST["hour_start"];
$hour_end = $_REQUEST["hour_end"];
$n_plots = $_REQUEST ["n_plots"];
$p_type = $_REQUEST ["p_type"];
$xkey = $_REQUEST["datas1"];
$ykey = $_REQUEST["datas2"];
$y2key = $_REQUEST["datas3"];
$xtitle = $_REQUEST["xtitle"];
$ytitle = $_REQUEST["ytitle"];
$y2title = $_REQUEST["y2title"];
$step = $_REQUEST["step"];
#$fill = $_REQUEST["fill"];
$file = 0;
$rcount = 0;
$t_interval = 0;
$t_count = 0;
$file_h = fopen("$f_filename", "r"); # open the data source file
containing data- read only
$second_start = $hour_start * 3600;
$second_end = $hour_end * 3600;
# to read the lines from the specified file
while (!feof($file_h)) {
     $file = fgets($file_h);  # to see only the last line

     $file_arr = preg_split("/\s/",$file,-1,PREG_SPLIT_NO_EMPTY);

     $secondi = $file_arr[0];
     if ($secondi>=$second_start && $secondi<$second_end)
     {
     if ($t_count==$t_interval){          # to read only the lines
according to the assigned interval
```

```
     for ($key=0; $key < count($file_arr) ; $key+=1) {
          $value = $file_arr[$key];
          $accdata[$key][$rcount] = round($value);  # read all
column elements in an array
     }
     $t_interval = 1;
     $rcount++;
     $t_count=0;                   #counts the total no. of rows
     }

     $t_count++;
     }
}
fclose($file_h);



# to assign the imported data to the graph
$xdata = $accdata[$xkey];
$ydata = $accdata[$ykey];
$y2data = $accdata[$y2key];

#To change time values from seconds to hours for x-axis data
if ($xkey == 0){
foreach ($xdata as $index=>$value) {
          $s = round ($value/3600);
     $c.=$s.',';
     }
     $x1data = preg_split("/,/",$c,-1,PREG_SPLIT_NO_EMPTY);
}else $x1data = $xdata;



// Create the graph. These two calls are always required
$graph = new Graph(750,300,"auto",-1);
$graph->CheckCSIMCache("graphdetails",1);
$graph->SetScale("$p_type");
$graph->ygrid->Show();        # shows the Y-axis gridlines
$graph->xgrid->Show();        # shows the x-axis gridlines
$graph->img->SetMargin(60,60,30,70);
$graph->title->Set("$date");
$graph->SetMarginColor('linen');


// Adjust the legend position
$graph->legend->SetLayout(LEGEND_HOR);
$graph->legend->Pos(0.5,0.98,"center","bottom");

#assign X-axis specifications
foreach ($xdata as $value) {
     if ($xdata >= 100){$graph->xaxis->Settitlemargin(20);}else
break;}

$graph->xaxis->SetWeight(2);
$graph->xaxis->title->Set("$xtitle");
$graph->xaxis->title->SetFont(FF_FONT1,FS_BOLD,14);
$graph->xaxis->SetTitlemargin(5);
$graph->xaxis->SetTickLabels($x1data);
#$graph->xaxis->HideLastTickLabel();
$graph->xaxis->SetFont(FF_FONT1,FS_NORMAL,12);
$graph->xaxis->SetTextLabelInterval(2);

#assign Y-axis specifications
foreach ($ydata as $value) {
     if ($ydata >= 100){$graph->yaxis->Settitlemargin(30);}else
break;}
$graph->yaxis->SetColor("blue");
$graph->yaxis->SetWeight(2);
$graph->yaxis->title->SetColor("blue");
$graph->yaxis->SetFont(FF_FONT1,FS_NORMAL,12);
$graph->yaxis->title->Set("$ytitle");
$graph->yaxis->title->SetFont(FF_FONT1,FS_BOLD,14);
$graph->yaxis->SetTextLabelInterval(2);
$graph->yaxis->scale->SetGrace(10,10 );
$graph->yaxis->Settitlemargin(45);
// Create the first linear plot
$lineplot1=new LinePlot($ydata);
if ($step == 1) {$lineplot1->SetStepStyle();}
// Set the legend
$lineplot1->SetLegend("$ytitle");
$lineplot1->SetColor("blue");
$lineplot1->SetWeight("1");
#$lineplot1->value->Show();
#$lineplot1->value->SetColor("blue");
#$lineplot1->value->SetFont(FF_FONT1,FS_BOLD);

// Add the first plot to the graph
$graph->Add($lineplot1);

# decides to show the second Y-axis plot upon selection
if ($n_plots == 2){
// Create the second linear plot
$pt = substr ($p_type,-3);

$graph->SetY2Scale("$pt");
$graph->y2axis->SetColor("red");
$graph->y2axis->SetWeight(2);
$graph->y2axis->SetFont(FF_FONT1,FS_NORMAL,12);
$graph->y2axis->title->Set("$y2title");
$graph->y2axis->title->SetColor("red");
$graph->y2axis->title->SetFont(FF_FONT1,FS_BOLD,14);
```

```
$graph->y2axis->SetTextLabelInterval(2);
foreach ($xdata as $value) {
     if ($xdata >= 100){$graph->y2axis->Settitlemargin(38);}else
break;}
// Create the second linear plot
$lineplot2=new LinePlot($y2data);
if ($step ==1) {$lineplot2->SetStepStyle();}
// Set the legend
$lineplot2->SetLegend("$y2title");
$lineplot2->SetColor("red");
$lineplot2->SetWeight("1");
#$lineplot2->value->Show();
#$lineplot2->value->SetColor("red");
#$lineplot2->value->SetFont(FF_FONT1,FS_BOLD);

// Add the second plot to the graph
$graph->AddY2($lineplot2);
}

// Display the graph
#$graph->Stroke();
$graph->StrokeCsim(basename( __FILE__));
?>
```

4) *The* **“data table”** *program code:-*

```
<?php
$dirname = $_REQUEST ['dirname'];
$filename = $_REQUEST ['filename'];
if (!$dirname || !$filename) {
$dirname = "./data/fast";
$filename = "200207~1.fas";}
$n_file = file ("$dirname/$filename");
foreach ($n_file as $key=>$val){
$file_arr[$key] = preg_split("/\s/",$n_file[$key],-
1,PREG_SPLIT_NO_EMPTY);}
echo "<table align = 'center' border=1>";
$h_file = file ("$dirname/var.txt");
echo "<tr>";
foreach ($h_file as $key=>$val){
     $rkey =$key + 1 ;
     echo "<th align= 'center' ><font SIZE=-1 color='#8B2323'
face='arial,helvetica'>$val</th>";}
echo "</tr>";
foreach ($file_arr as $kkey=>$vals){
echo "<tr>";
foreach ($vals as $l_key=>$l_val){
echo "<td align= 'center'><font SIZE=-2 color='#8B2323'
face='arial,helvetica'>$l_val</td>";}
echo "</tr>";}
echo "</table >";
#print_r($file_arr);
?>
```

References:

[1] *P. Astone, M. Bassan,P. Bonifazi,P. Carelli,E. Coccia,V. Fafone,S. D'Antonio,S. Frasca,A. Marini,E. Mauceli,G. Mazzitelli,Y. Minenkov,I. Modena,G. Modestino,A. Moleti,G.V. Pallottino, M. A. Papa,G. Pizzella,F. Ronga, R. Terenzi,M. Visco,L. Votano*, "**Cosmic Rays Observed by the Resonant Gravitational Wave Detector NAUTILUS**", LNF Publications, LNF-INFN, Rome, Italy, (Jan. 2000).

[2] *Alan Levine*,"**Writing HTML**", Maricopa center for Learining and Instruction (MCLI), www.mcli.dist.maricopa.edu/tut, (Jun. 2000).

[3] *M.Masciarelli, G. Mazzitelli* "Note: C-19 **DAFNE Web-Server Data Access Facility**", LNF Publications, LNF-INFN, Rome, Italy, (Apr. 2001).

[4] *S. Bertolucci, M. Cirillo, E. Coccia, A. de Waard, D. Di Gioacchino, V. Fafone, G. Frossati3, A. J. Lobo, A. Marini, G. Mazzitelli, V. Merlo, I. Modena, G. Modestino, L. Pellegrino, G. Pizzella, L. Quintieri, G. Raffone, F. Ronga, R. Russo, P. Tripodi, P. Valente*, "**RAP Proposal**", LNF Publications, LNF-INFN, Rome, Italy, (Nov. 2001).

[5] *G. Mazzitelli*, F. Murtas, *P. Valente*, " **The KLOE/DAFNE Status Logging, Analysis, and Database System** ", LNF Publications, LNF-INFN, Rome, Italy, (Dec. 2001).

[6] G. Mazzitelli, A. Stella, "Note: C-20 **DAFNE Server Data Access Facility Update**", LNF Publications, LNF-INFN, Rome, Italy, (Mar. 2002).

[7] *J. E. Sweat*, "**Developing Professional Quality Graphs with PHP**", www.zend.com/zend/tut, (May 2002).

[8] *W. Choi, A. Kent, C. Lea, G. rasad, C. Ullman*,"**Beginning PHP4**", Wrox Press,(Jul. 2002).

[9] *J. E. Sweat*, "**Advanced Features in JpGraph**", PHP architect, www.phparch.com, (Apr. 2003).

[10] *J. Persson*, "**JpGraph and DDDA**", johanp@aditus.Nu, Aditus Consulting, S-128 46,Emagatan 16, Stockholm, Sweden (Apr. 2003).

[11]*Tim Ziegler*, "**PHP from the Ground Up**", http://hotwired.lycos.com, (May 2003).

[12] *P. Valente, S. Bertolucci, E. Coccia, S. D'Antonio, A. De Waard, G. Delle Monache D Di Gioacchino, V. Fafone, G. Frossati, C. Ligi, A. Marini, G. Mazzitelli, G. Modestino, G Pizzella, L.Quintieri, G. Raffone, F. Ronga, P. Tripodi*, "**Acoustic detection of particles in ultracryogenic resonant antenna (RAP)**", LNF Publications, LNF-INFN, Rome, Italy, (Aug. 2003).

[13] *P. Valente, S. Bertolucci, E. Coccia, S. D'Antonio, A. De Waard, G. Delle Monache D Di Gioacchino, V. Fafone, G. Frossati, C. Ligi, A. Marini, G. Mazzitelli, G. Modestino, G Pizzella, L.Quintieri, G. Raffone, F. Ronga, P. Tripodi*."**Thermoacoustic Detection at the DAFNE Beam Test Facility**", LNF Publications, LNF-INFN, Rome, Italy, (Sep. 2003).