COMPUTER ASSISTED NEUROPHYSIOLOGY BY A DISTRIBUTED JAVA PROGRAM

Luc Jeandenans[1], Michel Gautero[1], François Grize[1], Igor V. Tetko[2,3], Alessandro E.P. Villa[2]

[1)] *Institut d'Informatique, Collège propédeutique, Université de Lausanne, Switzerland*

[2)] *Laboratoire de Neuro-heuristique, Institut de Physiologie, Université de Lausanne, Switzerland*

[3)] *Dept. of Biomedical Applications, IBPC-Academy of Sciences of Ukraine, Kiev-660, Ukraine*

Abstract

We have developed a graphic distributed software package which allows neurophysiologists to have at hand a set of network and graphical tools in order to perform the analysis of data collected from the electrical activity of neurons, studied in the experimental laboratory. This system is built around three major components. The first one processes the experimental data, in order to select desired data files from the networked computer and also displays these in a raster display. The second part selects one type of analysis (like cross-correlation) out of a set of possible choices, after configuration of some specific parameters. Finally, the third component deals with the results of data analyses, which are selected and displayed in a multi-parameter graphical 'ring binder'.

1. INTRODUCTION

Many applications in research or industrial domains are based on a very conventional working scheme which includes data acquisition, actual analytical processing, and finally the evaluation of the analysis, such as their visualisation. In some cases, data, analysis and graphic programs are at the same working place (for example, a workstation) where it is easy to manage all the above mentioned tasks. However, in most cases, it is more difficult because data needs to be on specific experimental devices and the data analysis needs to be on a fast computer platform like a server. Therefore, the simple working scheme involves difficulties due to network management and communication between applications. This should allow us to perform the following sub-tasks: data file acquisition from a computer network, transmission of the data to the analysis program, external analysis program launching, evaluation and display of results. This leads us to the following remarks: (i) retrieval, processing and display of experimental results can involve a large number of computers; (ii) human intervention at each stage is not only a waste of time but also can be a source of many errors: incorrect manipulations can all too easily happen, especially when the user has to work on many different systems.

L. B. Rasmussen et al. (eds.), *Computers and Networks in the Age of Globalization*

In this paper, we would like to present a software application which comprises an evolving integration of all the above mentioned tasks. We have developed software, called SPANAKE, which has been directly tested and validated in Neuroscience. The first objective was to use SPANAKE as a control panel of neurophysiology tasks after experimental assays. Moreover we would like to show how this development led us to the concept of a virtual laboratory.

2. COMPUTER ASSISTED NEUROPHYSIOLOGY

The hypotheses as to how the brain encodes and transmits information, generally referred to as 'brain information processing', can be based either on theoretical models, and more recently on subsequent simulation results, or on experimental observations and analyses of the sequences of spikes — the spike trains. Since the pioneering works of the 1960s (Perkel et al 1967a, 1967b) several methods of time series analysis applied to extracellularly recorded single unit spike trains have been developed. The most commonly used methods are based on first and second order statistics in time (Abeles 1982) and frequency domain (Brillinger et al 1976, Brillinger 1992).

The technological advances achieved in the past decade allow the simultaneous recording of several spike trains so that old and new hypotheses of neural coding based upon cell assemblies can be tested. Most of these hypotheses suggest that higher brain activities related to cognition and sensori-motor association require precise timed relationships between cell assemblies. Thus, several analytical methods such as gravity clustering (Gerstein et al 1985), spatio-temporal patterns of spikes detection algorithms (Dayhoff and Gerstein 1983, Abeles and Gerstein 1988, Tetko and Villa 1997) and joint peri-stimulus-time histogram (Aertsen et al 1989) have been developed to detect and reveal the existence of temporally organised patterns of activity. In addition, recent developments of spike trains analysis in the framework of nonlinear dynamic systems (Aihara et al 1990, Celletti and Villa 1996, Pei and Moss 1996) have suggested additional tools to study the dynamics of experimentally recorded cell assemblies.

All methods mentioned above require massive computations. For this reason they were developed on a large variety of computer platforms according to the computational means available to the investigators. Such a variety of platforms was usually accompanied by customised choices of the operating system, file management and graphic interface. The computational power gained by personal computers in 1997 at affordable price (i.e. at less than 5,000 US$ per unit) together with the communication facilities provided by world wide digital networks has dramatically modified the environment of spike train data analysis in neurophysiology. A global computational approach to such experimental studies, termed Computer Assisted Neurophysiology (CAN) was suggested (Villa et al 1998). In recent years several good software packages were developed, such as XNBC (Vibert et al 1997), Neural Data Analysis (Laboratory for Higher Brain Functions, Dept.

of Physiology, Hebrew University, Jerusalem, Israel) and Stranger (Biographics Inc., Winston-Salem, NC).

We present here the principal features of an approach to CAN based on a new programming environment. Many ideas for the development of our approach have been inspired by the above mentioned packages. We would like to present our software as an evolving approach which should benefit in the future of contributions of several investigators. A key feature of our program is a graphic analysis assistant designed to achieve two essential goals:
(i) to control all useful data processing functions (so as to manage the acquisition of data from the computer network, transmission of data to analysis programs, and the display of the results) from the same console (either a terminal or a personal computer);
(ii) to make this graphical tool easy and quick to use, and structured to allow heterogeneous use. For instance, our software allows the user to visualise some data files while at the same time selecting some other files for analysis, or to launch an analysis while visualising some results from another analysis. We have used the Java programming language (Gosling et al 1996) to reach these two goals. This is a very recent object oriented language and programming environment which offers platform independent software tools including graphic package and standard network protocols. We present different parts of the new software corresponding to segments of data flow from experiment to analyses, showing multiple advantages of the choice of Java programming (Figure 1).

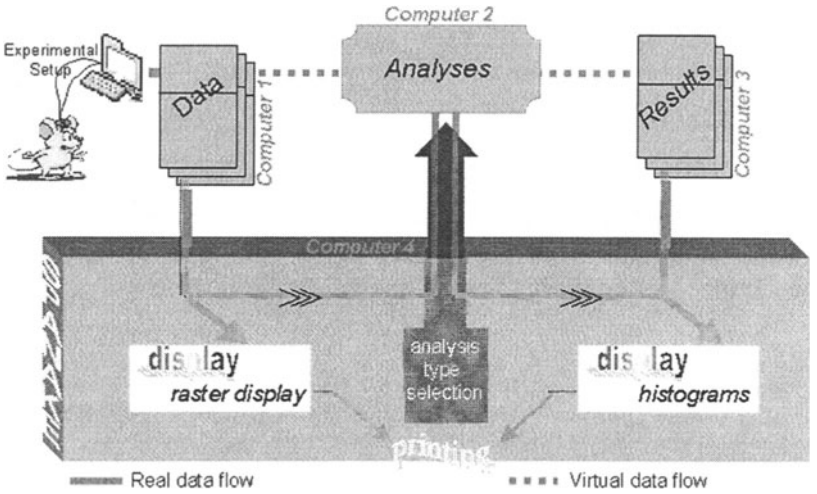


Figure 1. Diagram of the data flow handled by SPANAKE.

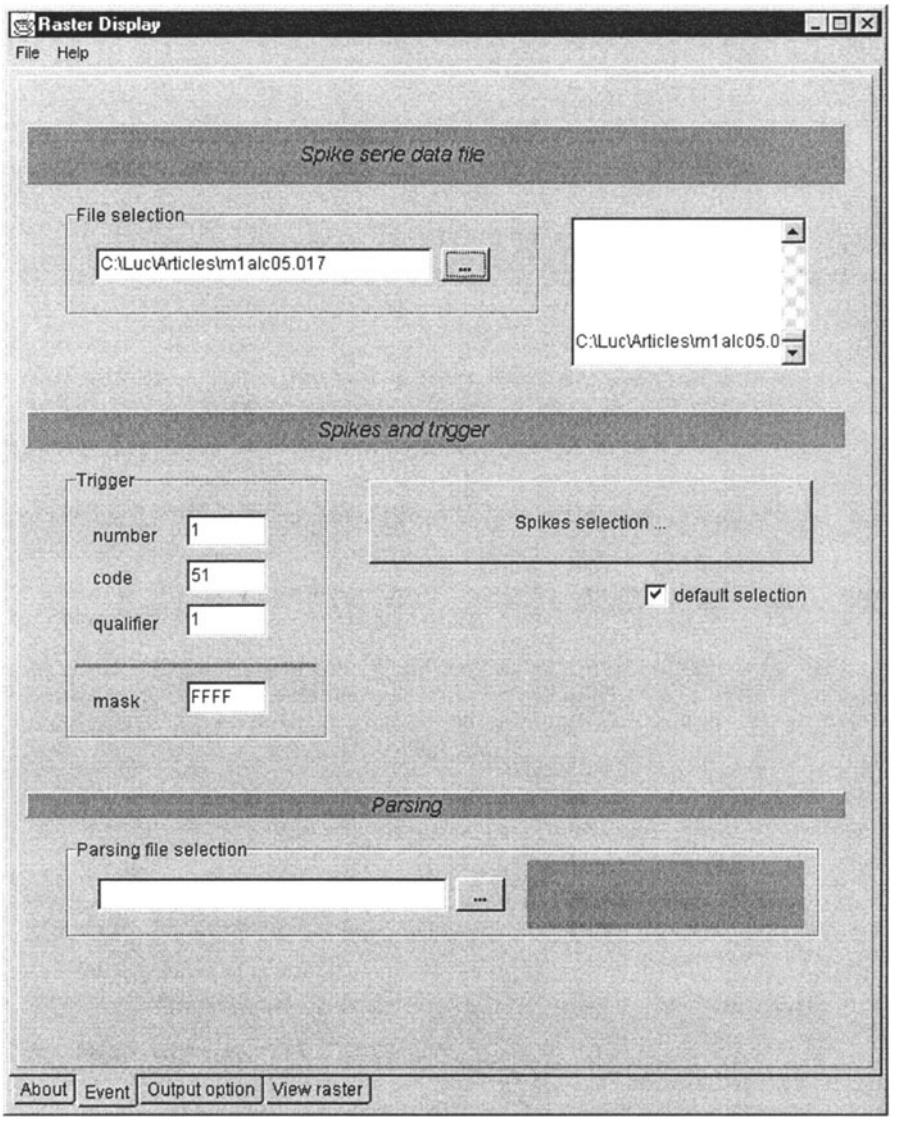
3. TOWARD A VIRTUAL LABORATORY

As we show in Figure 1, SPANAKE performs data analysis through requests launched on a networked computer. It is essential to reach data and analysis programs where they are supposed to be. Indeed, complex data processing should be launched on high-performance machines and it should be possible to import data, like spike data files, from anywhere in the world, specifically from a distant laboratory which we collaborate with. However, analysis and display could be executed on a local computer for domestic needs. Therefore, the working place becomes a virtual laboratory based on local and networked tasks in order to perform the same tasks as in a local place.

3.1 Data file acquisition task

Multiple spike trains were recorded from the cerebral cortex of anaesthetised (Villa et al 1998a) and freely moving rats (Villa et al 1998b). Spike trains are stored, as discrete time series, in readable standard format files. These are ASCII files and thereby they can easily be moved around the computer network.

In SPANAKE, raw data files can be selected singly or as groups for analysis, following an interactive menu for choosing specific time series analyses. In addition, these files can be displayed as dot rasters aligned according to a selected triggering event and customised time scales (Figure 2). In particular, it is possible to drag a cursor on a 'master' dot display and interactively change the onset of the trigger event with respect to the time axis origin. Moreover, these files can be selected and collected from linked network sites. The implementation of this point is explained in further detail in this paper.


Raster Display
File Help
Spike serie data file
File selection
C:\Luc\Articles\m1alc05.017
...
C:\Luc\Articles\m1alc05.0
Spikes and trigger
Trigger
number 1
code 51
qualifier 1
mask FFFF
Spikes selection ...
default selection
Parsing
Parsing file selection
...
About
Event
Output option
View raster

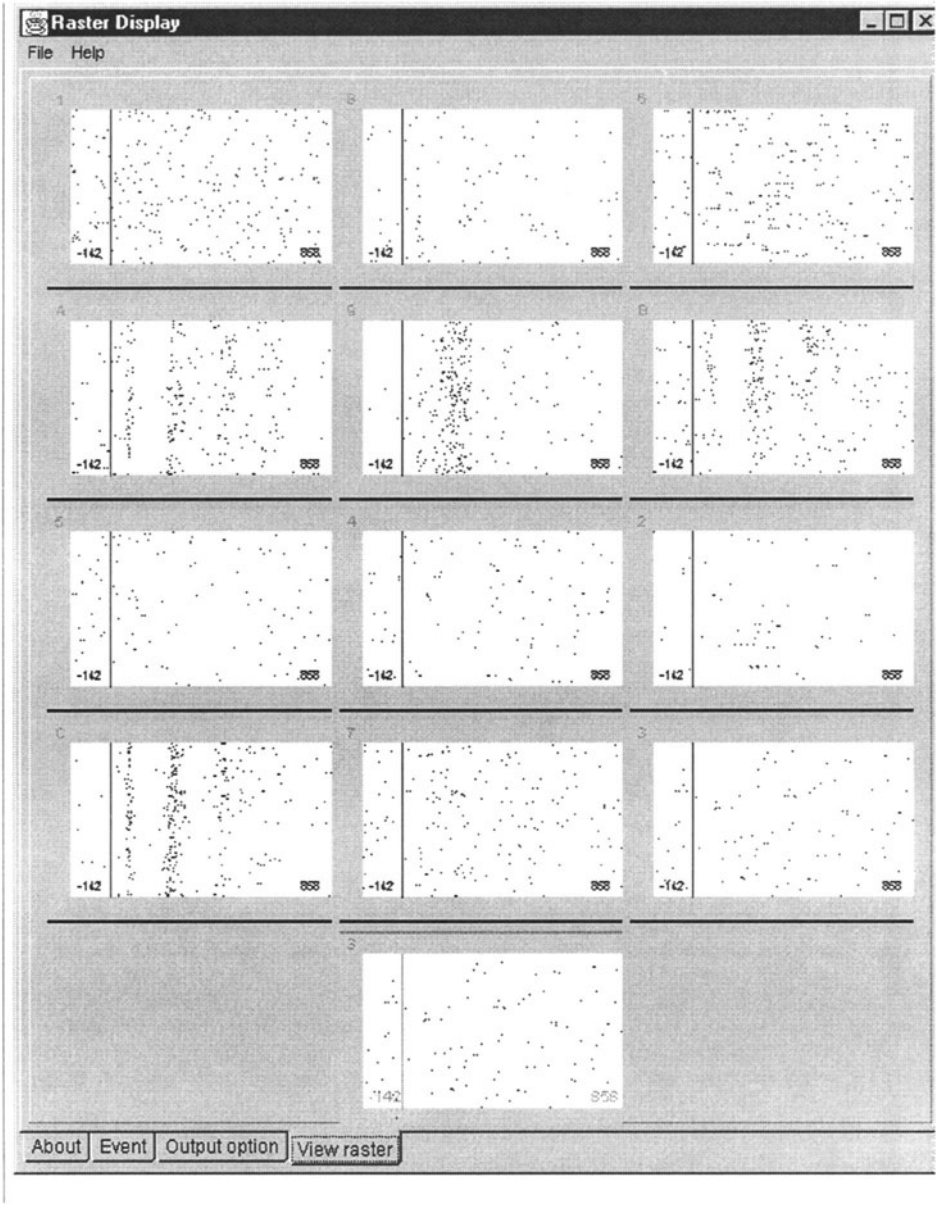


Figure 2. (on previous page and this page) On the left, trigger, spikes and data file selection. On the right, an example of the raster display of thirteen simultaneously recorded neurons aligned to the onset of a stimulus (vertical line). The time scale was set to 1000 ms. The vertical scale represents 100 repetitions of the same stimulus sweep.

For a file transfer via the network we used a specific feature of the Java language, which would have been difficult to implement with another programming language. The files are compressed before being transferred and decompressed at the target computer. Compression and decompression of the data stream are then effected transparently. This feature is particularly important because of the growing traffic load in computer networks which often provokes low transfer rates. It is essential to avoid a situation in which the transfer time exceeds the time needed for the analysis. Finally, on both client and server sides, we used the multithreading techniques afforded by Java. By this way, a server can serve several clients simultaneously and a client is able to process the results of an analysis, while launching the loading of a different series of results.

3.2 Data analysis task

There are two essential requirements for implementing computationally demanding time series analyses. On the one hand, they need speed and therefore an efficient programming language, like C or C++ which is not necessarily the language of a graphical tool. On the other hand, they need high-performance computers. These two specific characteristics often lead to specific application developments on different closed platforms, so that while one analysis program should be implemented in language X1 on a network machine Y1, another one could be implemented in language X2 on a machine Y2. So, it is necessary to establish a link between SPANAKE and customised specialised programs in order to communicate all information (i.e. data files and parameters) needed to launch these programs from the interface, and finally to recover the results. This process should be 'transparent' in order to let the user focus on the analytic procedures. By means of a configuration file handled through an interactive menu, the user can select precise preprocessing parameters in order to determine the exact block of data to be analysed.

To reach this goal, we have chosen a client/server mode; a dialogue box allows the user to select a specific computer over the network (specified as an Internet host) and subsequently the local user's computer interrogates the servers for each of the tasks referred to above and establishes all required communication channels in a 'transparent' way. All control procedures are managed by SPANAKE, which is therefore the client. The remote machines that are contacted must play the role of servers. Therefore we developed a server, which must be installed and run on all the machines involved in data processing. At present, we use the usual client/server mechanisms, i.e. sockets. The client wishing to dialogue with a server requests the latter to open a socket. Then, reading and writing files through the network is performed in exactly the same way as they were stored locally.

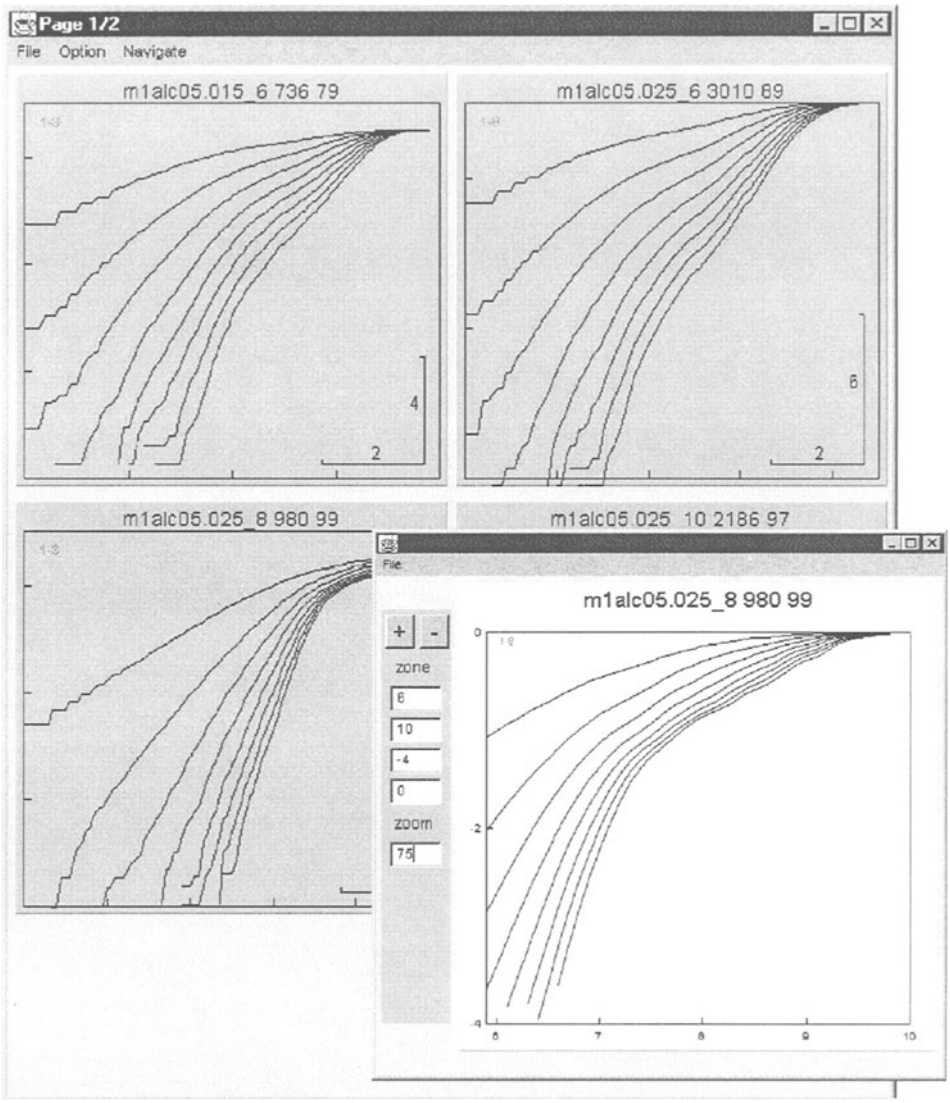
3.3 Native code

It was essential for SPANAKE to be able to access programs written in languages other than Java. We cannot expect users to abandon customised software for complex data analysis and rewrite it in the language chosen by us. Java offers a solution to this problem because it enables programs written in C or C++ to be interfaced through a Java-written control procedure. More unusual is the fact that Java also defines the reverse communication protocol, namely the possibility for a program written in C or C++ to dialogue with the program written in Java. We can therefore ensure communication in both directions between the different applications. This communication is effected independently from a physical computer platform, in a manner that is transparent to the end user. Note that the relative slowness of Java (a language which is two years old can hardly claim to rival in its effectiveness languages that have been optimised for several decades) is not a drawback because the computationally intensive programs for data analysis may be written in C or C++. The example of a user going to add a new analysis to SPANAKE will illustrate these points in detail.

3.4 Display task

The layout of the 'ring binder' that we have implemented in SPANAKE will benefit from a clear separation between the visual aspects and event management. This improvement will be implemented without jeopardising the parallel structure that we introduced into the display. Each page of the 'ring binder' is constructed in a separate thread, which is defined as platform-independent in Java. The last task of such thread is to launch the construction of the next page by calling the result display window. When this window receives the message and all the pages have been constructed, the process stops. In this way, the computer is not overloaded by the simultaneous initialisation of dozens of threads (most machines are still mono-processors). Thus, a page can be displayed as soon as it has been constructed and it becomes possible to navigate freely between all the constructed pages.

For example, as a result of the time series analyses, several graphical files are produced containing several hundreds of curves. They can be displayed after single or multiple file selection. After loading, all curves are available on several pages and a browsing system allows navigation through these pages. On each page, it is possible to determine the number of graphics to be displayed and to zoom into them independently and to print them (Figure 3). The graphical metafiles are based on a very simple format based on a series of x, y coordinates, thus allowing the possibility to enhance the display package for future analyses.


Page 1/2
File Option Navigate
m1alc05.015_6 736 79
m1alc05.025_6 3010 69
m1alc05.025_8 980 99
m1alc05.025_10 2186 97
File
m1alc05.025_8 980 99
+
-
zone
6
10
-4
0
zoom
75

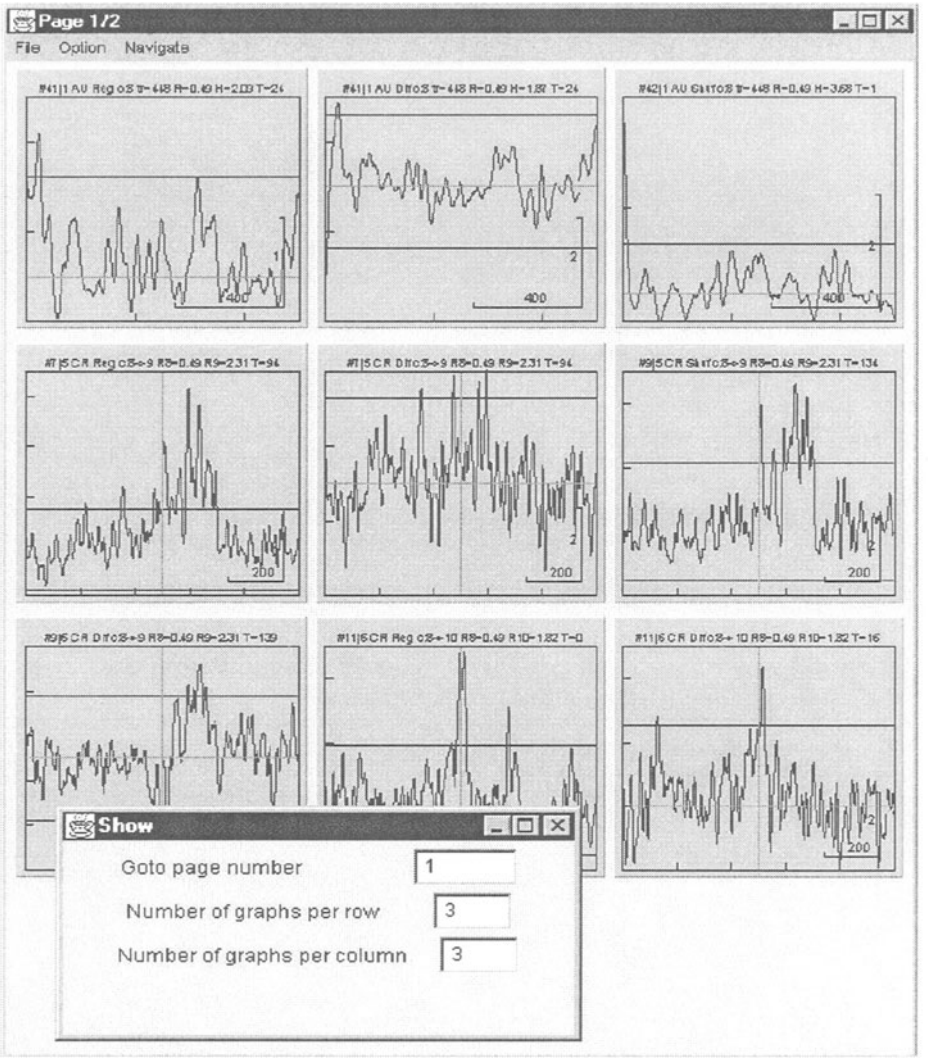


Figure 3.(on privious page and this page) On the left, a display of dynamic system analysis according to Grassberger and Procaccia algorithm (Celletti and Villa 1996). The correlation integral is plotted versus the radius r, in the natural logarithmic scale for embedding dimensions 1 to 8. Note that one display is zoomed and the lines are parallel across a region called scaling region. On the right, a display of auto-renewal density analysis (Abeles 1982). The horizontal axis is time (ms) and the vertical axis in scaled in rate units (spikes/s). Horizontal and vertical bars on each display indicate the scale.

4. DISCUSSION

We have presented the outlines of a software package, SPANAKE, designed to allow neurophysiologists to work in a user-friendly computer environment. The main points of our software are its programming language, Java, and its use across a computer network via the Internet protocol. This software offers an effective tool for neurophysiologists involved in computer-demanding analyses because its modular features allow for easy update and extension. An additional advantage is that investigators can use a distributed architecture based on powerful remote computers for data analysis, and local computers for management and display. However, this tool is based on very general principles which are data file acquisition from a computer network, communication with some external applicat-ions/programs (e.g. transmission data), graphical tools (e.g. interpretation of potential results). Because network and communication management is transparent for the user, SPANAKE builds up a working place as a virtual laboratory. This feature could allow additional applications to take advantage of our graphical tool.

Moreover, the Java programming language allows this software to be run on multiple platforms without recompilation and make usage of programs compiled using native C/C++. In the future, enhanced Internet properties of Java will provide the opportunity to run this software directly from the World Wide Web network. In particular, the socket-based client/server solution might be replaced by communication interfaces based either on the RMI (Remote Method Interface) protocol specifically designed for Java, or on the CORBA protocol, which is a standard for distributive programming. Furthermore, the use of the Java language and the standard J.D.K. version enables one to envisage the conversion of the graphic application into an applet which is accessible from any Java-compatible Web browser. Finally, security is handled with little effort because Java includes embedded security rules for network use.

Acknowledgements

We thank Brian Hyland and Marco Tomassini for providing useful comments and suggestions to improve the manuscript. This grant was partially supported by Swiss grant FNRS 2150-045689.95.

REFERENCES

Abeles, M. and Gerstein, G. L. (1988) Detecting spatio-temporal firing patterns among simultaneously recorded single neurons. *Journal of Neurophysiology* **60** 909-924.

Abeles, M. (1982) Quantification, smoothing, and confidence limits for single unit histogram. *Journal of Neuroscience Methods* **5** 317-325.

Aertsen, A. M. H. J., Gerstein, G. L., Habib, M. K. and Palm, G. (1989) Dynamics of neuronal firing correlation: modulation of 'effective connectivity'. *Journal of Neurophysiology* **61** 900-917.

Aihara, K., Takabe, T. and Toyoda, M. (1990) Chaotic neural networks. *Physics Letters A* **144** 333-340.

Brillinger, D.R. (1992) Nerve cell spike train data analysis: a progression of technique. *J. American Statistical Assoc.* **87** 260-271.

Brillinger, D.R., Bryant, H.L. and Segundo, J.P. (1976) Identification of synaptic interactions. *Biological Cybernetics* **22** 213-228.

Celletti, A. and Villa, A.E.P. (1996) Determination of chaotic attractors in the rat brain. *Journal of Statistical Physics* **84** 1379-1386.

Dayhoff, J. E. and Gerstein, G. L. (1983) Favored patterns in spike trains. I. Detection. *Journal of Neurophysiology* **49** 1334-1348.

Gerstein, G.L., Perkel, D.H. and Dayhoff, J.E. (1985) Cooperative firing activity in simultaneously recorded populations of neurons: detection and measurement. *Journal of Neuroscience* **5** 881-889.

Gosling, J. and Yellin, F. (1996) *JDK: The Java Application Programming Interface: Volumes 1 and 2*. Addison Wesley, New York, NY.

Gosling, J., Joy, B. and Steele, G. (1996) T*he Java language Specification*. Addison Wesley, New York, NY.

Pei, X. and Moss, F. (1996) Characterization of low-dimensional dynamics in the crayfish caudal photoreceptor. *Nature* 379 618-621.

Perkel, D. H., Gerstein, G.L. and Moore, G.P. (1967a) Neuronal spike trains and stochastic point processes. I. The single spike train. *Biophysics Journal* **7** 391-418.

Perkel, D. H., Gerstein, G.L. and Moore, G.P. (1967b) Neuronal spike trains and stochastic point processes. II. Simultaneous spike trains. *Biophysics Journal* **7** 419-440.

Tetko I. V. and Villa, A. E. P. (1997) Fast combinatorial methods for estimation of complex temporal patterns of spikes. *Biological Cybernetics* **76** 397-407.

Vibert, J.F., Pakdaman, K., Boussard, E. and Av-Ron, E. (1997) XNBC: a simulation tool. Application to the study of neural coding using hybrid networks. *Biosystems* **40** 211-218.

Villa, A. E. P., Tetko, I. V., Dutoit, P., de Ribaupierre, Y. and de Ribaupierre, F. (1998a) Corticofugal modulation of functional connectivity within the auditory thalamus of rat, guinea pig and cat revealed by cooling deactivation. *Journal of Neuroscience Methods* (in press).

Villa, A.E.P., Hyland, B., Tetko, I.V. and Najem, A. (1998b) Dynamical cell assemblies in the rat auditory cortex in a reaction-time task. *Biosystems* (submitted).