



ELSEVIER

Computer Methods and Programs in Biomedicine 53 (1997) 163–173

Computer Methods
and Programs
in Biomedicine

SWEEPS: a program for the acquisition and analysis of neurophysiological data.

Gerald S. Pollack *

Department of Biology, McGill University, 1205 Avenue Dr Penfield, Montreal, Quebec H3A 1B1, Canada

Received 12 August 1996; received in revised form 13 February 1997; accepted 21 February 1997

Abstract

I describe SWEEPS, a program for the acquisition and analysis of neurophysiological data written with LabWindows/CVI. SWEEPS includes many features of general interest to neurobiologists, such as digital filtering, window discrimination, and the construction of peristimulus time histograms. As the program is written using LabWindows/CVI, a C-programming system which includes routines for many data-processing and display tasks, it can be easily modified to accommodate new analysis needs as they arise. © 1997 Elsevier Science Ireland Ltd.

Keywords: Neurophysiology; Data acquisition; Analog-to-digital conversion; LabWindows

1. Introduction

In this article I describe a program for the acquisition and analysis of neurophysiological data. SWEEPS implements a virtual oscilloscope that acquires waveforms from analog sources, performs analyses and transformations of the waveforms, allows them to be saved to disk, either in native format or as ASCII files for export to other software, and produces hard-copy output.

SWEEPS was written using LabWindows/CVI, a C-language programming environment specifi-

cally designed for data acquisition and analysis. The particular features of SWEEPS are customized to my laboratory's research on the acoustic behavior and physiology of insects. Many of these features, e.g. digital filters, spike counting, peristimulus time histograms, may be of general interest to neurobiologists, while other features required by some users will undoubtedly be missing. However, because LabWindows/CVI includes libraries of routines for many data acquisition and analysis tasks, as well as for the creation of a windowing environment that allows intuitive communication between the program and the user, SWEEPS can readily be tailored to fit the needs of a particular laboratory.

* Tel.: +1 514 3986418; fax: +1 514 3985069; e-mail: GPOLLACK@BIO1.LAN.MCGILL.CA

SWEEPS offers much of the functionality of many other software packages, both commercial and otherwise e.g. [1–3]. It is unique in its combination of a modern windowing interface, ease of modification, and the use of a standard, powerful, programming language.

2. Description of LabWindows/CVI

LabWindows CVI is a C program development system sold by National Instruments (<http://www.natinst.com>), that includes a source-code editor, integrated debugging tools, and a compiler and linker that produce 32-bit programs. Most importantly, LabWindows comes with a number of libraries of pre-written routines for data acquisition, analysis and presentation, as well as for the construction of a graphical user interface. Programming is simplified by a number of tools, such as a drag-and-drop user interface editor that is used to construct a graphical user interface, and 'function panels', which are forms that allow the programmer to write the code for calling LabWindows library functions simply by filling in the fields that correspond to the functions' parameters (thereby avoiding many errors of typing and syntax).

LabWindows CVI is available both for IBM-compatible computers running Microsoft Windows and for SUN workstations. My experience (and this article) is limited to the Microsoft Windows version (specifically, version 3.01).

3. Description of SWEEPS

3.1. User-interface features

The interface of SWEEPS consists of a number of 'panels' (LabWindows' term for a screen of information to be entered and/or displayed), each populated by a number of 'controls' (LabWindows' term for software devices responsible for obtaining input from the user and for providing output). For example, upon starting the program the user is presented with the main panel (Fig. 1), which includes the main menu bar, the oscillo-

scope screen (a 'graph' control), adjustments for vertical gains, offsets, and time base ('numeric' controls), readouts of cursor coordinates (also numeric controls), and boxes that, when clicked on with the mouse, toggle the oscilloscope in or out of storage mode, display or hide the cursor, etc. ('toggle button' controls). This is far from an exhaustive list of the types of controls used throughout SWEEPS, which themselves represent only a subset of the types available with LabWindows. Most of the controls are programmable so that their behaviour can be modified according to current conditions. For example, whenever new data are either acquired or loaded from disk, the maximum permissible value of the control labeled 'ms per div' on the main panel is set to one tenth the duration of a sweep. This limits the length of a displayed sweep (10 divisions) to the duration of the actual data sweeps, and thus prevents attempts to display data that do not exist.

3.2. Operation of SWEEPS

SWEEPS' main features can be classified into the following general areas: data acquisition, data display, data processing, and file operations.

3.2.1. Data acquisition

The unit of data acquisition is a 'sweep', i.e. a defined number of data points acquired from a defined array of analog input channels at a defined sampling rate. A data set consists of one or more such sweeps. Parameters relevant to data acquisition, including which of the eight available channels to sample, sampling rate, sweep duration, and trigger source (either a signal applied to the acquisition card's trigger input line, or a key stroke or mouse click by the user), are specified by the user on the setup panel.

Just before acquisition begins the user selects whether a defined number of sweeps should be acquired and retained in memory or whether successive sweeps should instead be acquired continuously until terminated by the user, in which case only the most recent sweep is retained. Memory to accommodate the data is allocated dynamically according to the acquisition parameters (duration and sampling rate) and the number of sweeps

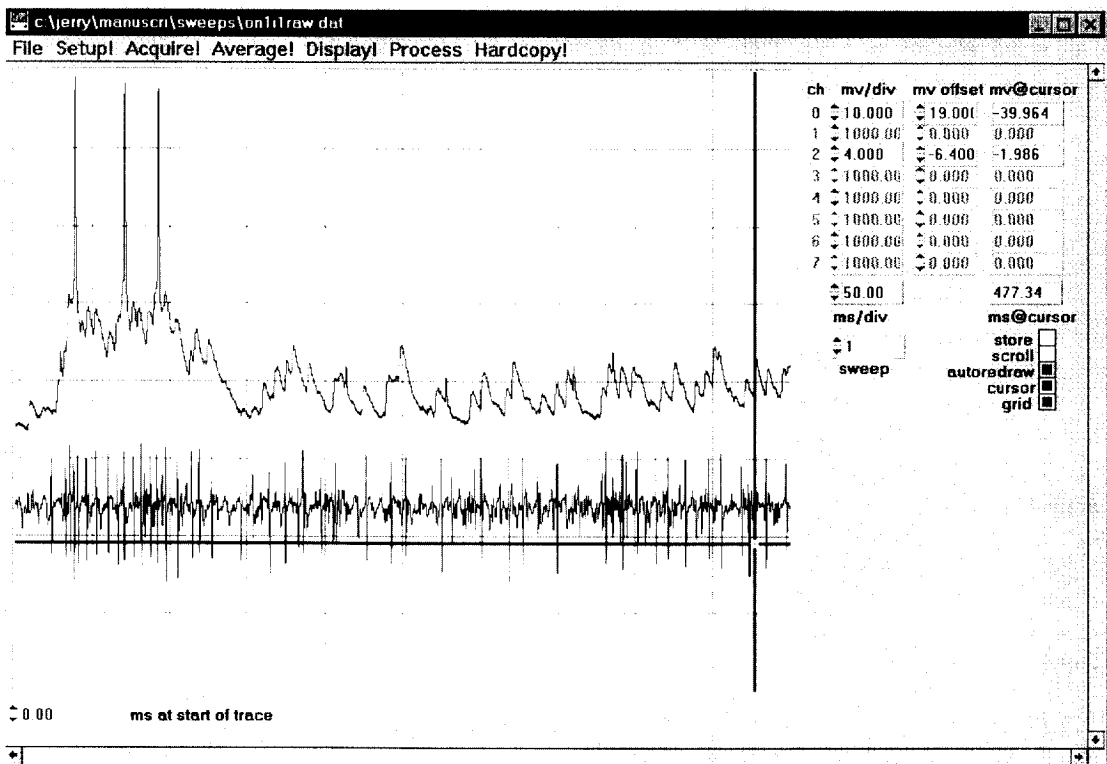


Fig. 1. The main panel of SWEEPS. Program features are selected by clicking on the appropriate entries on the menu bar near the top of the panel. Data are displayed on a virtual oscilloscope screen. The traces shown here are recordings from auditory neurons of crickets. Data were acquired on channels 0 and 2; the controls pertaining to the other, inactive, channels are 'dimmed'. The grid divides the screen into 8 vertical and 10 horizontal divisions, the scaling of which are set by the mv div and ms div controls. Features of the display, and the functions of the controls, are described in the text (Section 3.2.2).

expected. Enough memory is allocated to accommodate one sweep more than expected. This 'extra' sweep is used to display the results of processing operations (see below).

The data acquisition features of SWEEPS are customized for the acquisition hardware used in my laboratory (ATMI016F5 and ATMI064F5 multifunction interfaces from National Instruments); users of other devices might have to modify the program. The data acquisition library supplied by National Instruments only supports that company's products, but custom functions can be written for other interface cards, and many such functions are available either from the card's manufacturer or from the community of LabWindows users. Data from other sources can also be imported into SWEEPS for display and analysis so long as they can be converted to SWEEPS' file

format (which is available upon request from the author).

3.2.2. Data display

The central feature of the display is, of course, the virtual oscilloscope screen (Fig. 1). As with standard oscilloscopes, waveforms can be displayed at a variety of time and voltage scales (by adjusting mv div and ms/div), and traces can be positioned vertically (mv offset). The features of the display are also affected by a number of 'toggle buttons' that determine whether the grid and cursor are displayed, whether or not the screen emulates a storage oscilloscope, and whether or not the display should be updated immediately to reflect changes in scaling, etc. (autoredraw).

An important feature of SWEEPS (and a marked improvement on many storage oscilloscopes) is that long data records can be acquired with a high sampling rate. If such sweeps are viewed at high temporal resolution (i.e. with a 'fast' time base) then only a portion of the record can be displayed at once, and the oscilloscope screen serves as a window into a subset of the data. The data can be scrolled through this window automatically by clicking the scroll button or, alternatively, the beginning of the display can be set to correspond to any section of the data record by entering the appropriate time value in the control labeled 'ms at start of trace'.

In addition to displaying data on screen, SWEEPS can generate hard-copy output, either by printing to any bit-map compatible Windows device (e.g. a laser printer) or, indirectly, by producing a file in Hewlett Packard Graphics Language, which then can be imported into external graphics programs.

3.2.3. Data processing

Data processing operations are of three general types. Some (e.g. filtering) change the original waveforms; others (e.g. peak measurements) make simple measurements from waveforms, and still others (e.g. power spectrum) present graphical displays of functions or relationships derived from the waveforms. The user can choose to apply most procedures either to the currently displayed sweep or to the entire collection of sweeps in memory. For operations that modify waveforms, if the procedure is applied only to the current sweep the results are displayed as the 'extra' sweep described in Section 3.2.1; if the procedure is applied to all sweeps, then the original data are overwritten. For procedures that simply make measurements, the results for operations on single sweeps are shown on screen; if the analysis is to be performed on all of the sweeps in memory, the results are saved in an ASCII file.

Space constraints prohibit a detailed explanation of all of the processing features, but the following brief descriptions give some idea of the current capabilities of SWEEPS. In Section 4 of this paper I present examples that illustrate how some of the processing features are used together

to do useful work. Most of the computations performed during processing use functions included in LabWindows libraries.

3.2.3.1. Average. Computes the arithmetic means of the waveforms in a group of selected sweeps, treating each channel's data separately, and displays the result as the 'extra' sweep that is reserved for the output of processing operations (see Sections 4.1 and 4.2).

3.2.3.2. Correlate. Computes the correlation function for two selected input channels (from the same or different sweeps). If the two input waveforms are identical, then 'correlate' computes the waveform's autocorrelation function; if the two inputs are different, then 'correlate' displays their cross-correlation function (see Section 4.2).

3.2.3.3. Differentiate. Differentiates the data for a selected channel with respect to time (see Section 4.2).

3.2.3.4. Filter. LabWindows includes a broad array of digital filters, only two classes of which are included in SWEEPS. The Butterworth filters are infinite impulse response filters, while the Window filters are finite impulse response filters; the main difference between the two types is their effect on the phase spectrum of the signal being filtered. The precise characteristics of the filters are set by selecting cut-off frequencies and parameters that affect sharpness (see Section 4.1).

3.2.3.5. Integrate. Integrates the waveform on a selected channel, over a selected time window (see Section 4.1).

3.2.3.6. Peak measurement. Reports the most-positive and most-negative values of a waveform within a specified time window, as well as their difference.

3.2.3.7. Power spectrum. Computes the power spectrum of a selected channel for that portion of a sweep that is currently visible on the oscilloscope screen.

3.2.3.8. Rectify. Multiplies all negative values in the input waveform by -1.0 (see Section 4.1).

3.2.3.9. Window discriminator. Searches the specified channel of a selected series of sweeps for events that meet a specified voltage condition within a specified time window. The voltage can either be required to cross a threshold, with specified sign of slope, or it can be required to fall in a window (i.e. to satisfy both lower and upper thresholds), entering with specified slope. The waveform can be binarized (i.e. replaced by a series of uniform positive pulses, of selected pulse width, at the points at which the events occur), or the number of events can simply be counted, or the times of the events can be 'marked', i.e. kept track of in memory. Marking is used for two purposes: (1) to create, from a single long sweep, a series of shorter sweeps that are registered in time according to the occurrence of events of interest, and (2) to create peristimulus-time histograms (see Section 4.2).

3.2.3.10. Peristimulus time histogram. Accumulates the times of occurrence of events (which must first have been 'marked' by the window discriminator) over a selected series of sweeps and plots a histogram of the number of events occurring in successive time 'bins' (see Section 4.2).

3.2.4. File operations

In addition to routine operations such as saving and loading data files, SWEEPS includes some operations that facilitate data analysis. If events within a sweep have been marked by the window discriminator, then a series of new sweeps can be generated, one for each event, in which the events occur at the same (and specified) time relative to the beginning of each new sweep. The new series of sweeps can then be superimposed or averaged, in order to highlight features that are time-locked to the events detected by the window discriminator (see Section 4.2). The File menu also includes a utility to copy waveforms from one channel to another, either within or between sweeps. This is useful for preparing sweeps that have multiple versions of the same trace (e.g. a 'native' and processed version). Another utility allows one to

save a new set of sweeps in which only every 2nd, 3rd,... nth data point of the original sweep is retained; this can save disk space, memory, and computing time in those cases where the waveforms can tolerate such decreases in effective sampling rate.

4. Sample runs

In this section I present two examples of how SWEEPS can be used to extract useful information from physiological recordings.

4.1. Analysis of compound responses recorded from a whole sensory nerve

The experiment summarized in Fig. 2 examines the frequency-sensitivity of the auditory nerve of the cricket, *Teleogryllus oceanicus*. Extracellular recordings were made from the whole nerve while the animal was stimulated with 20 different sound frequencies, all at the same intensity (80 dB), and each repeated ten times. The data set thus consists of 200 sweeps. Acquisition of each sweep was triggered externally by a signal that preceded the stimulus by 10 ms; thus the sweeps are all aligned in time relative to the stimulus.

One of the original recordings (a response to a 4.5 kHz stimulus) is shown as the top trace of Fig. 2(a). The response consists of spikes from many individual auditory receptors, and the events seen in the trace are mainly compound action potentials rather than spikes from single receptors. Integration was used to quantify these compound responses, according to a three-step protocol. First, the responses were high-pass filtered (Butterworth, 100 Hz cut-off) so as to remove any DC-offset in the recording, which would otherwise be problematic in the next step of the procedure, rectification. The top trace of Fig. 2(a) has been so filtered. Next, the responses were rectified, so as to ensure that both negative and positive excursions would contribute cooperatively to the integral of the response, rather than canceling one another out. The middle trace of Fig. 2(a) shows the rectified version of the response. Finally, the rectified responses were integrated over an appro-

A

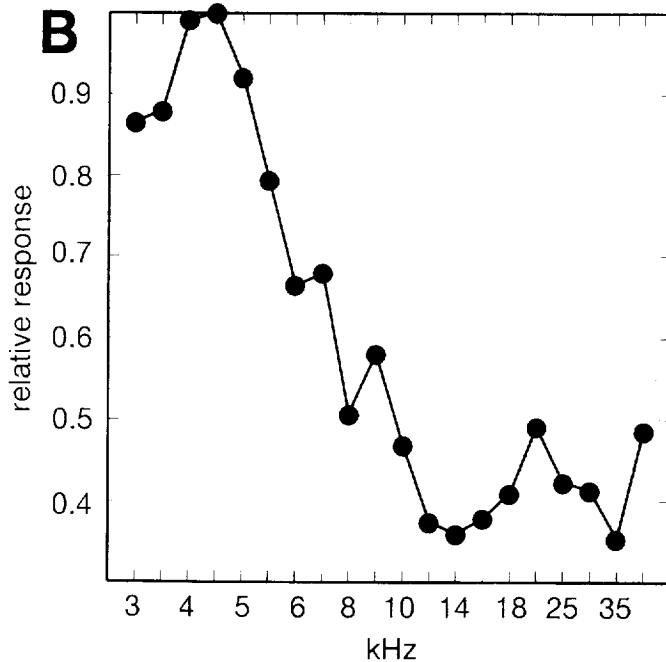
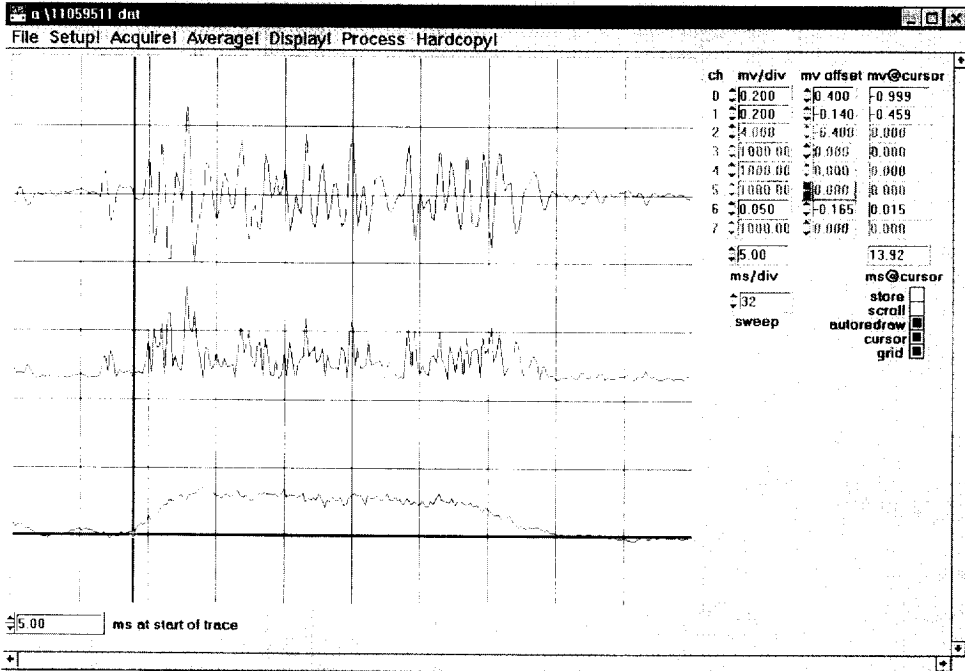


Fig. 2. Recordings from a cricket's auditory nerve. (a): The top trace shows a single response to a 4.5 kHz, 80 dB stimulus, after high-pass filtering. The middle trace is the same response after rectification. The bottom trace is the average of all 200 recordings in the data set, each of which was first high-pass filtered and rectified. (b): A graph of the mean normalized integrated response of the nerve as a function of sound frequency, prepared with another program, SYSTAT, after importing data from an ASCII file created by SWEEPS. See text for further details.

priate time window, and the results stored in an ASCII file. The time window for integration was determined by averaging all 200 sweeps in the data set and reading the response onset and offset times from this averaged trace (shown as the bottom trace of Fig. 2(a)). The ASCII file containing the integrated responses was then imported into a statistics program, SYSTAT, where the mean response to each frequency was calculated and plotted (Fig. 2(b)). The nerve responds most strongly to sounds with frequencies in the range 4–5 kHz [4], which corresponds to the frequency range of intraspecific acoustic signals [5].

4.2. Analysis of temporal correlation between two neurons

The recordings shown in Fig. 1 are responses of two identified auditory interneurons of crickets: the upper trace is an intracellular recording from the neuron ON1 (which has been hyperpolarized to suppress most action potentials and allow synaptic potentials to be seen more clearly), and the lower trace is an extracellular recording from the neuron AN2. Fig. 1 shows the beginning of the response to a 5s-long, 30 kHz sound stimulus. Such stimuli activate a small number of auditory receptors that are sensitive to ultrasound and that provide powerful common inputs to both neurons, resulting in temporally correlated activity [6]. The following example shows how SWEEPS can be used to analyze the temporal relationship between the spikes recorded from AN2 and EPSPs recorded in ON1.

The window discriminator was used to 'mark' the occurrence of AN2 spikes throughout the 5 s-long sweep. These time points were then used to create a new series of 197 sweeps, each 40 ms in duration, and each with the detected AN2 spike occurring halfway through this period. Fig. 3(a) shows 30 of these sweeps superimposed. There is a hint here of more frequent occurrence of EPSPs in ON1 near the time of AN2 spikes, but this is difficult to discern clearly. The relationship is clearer in Fig. 3(b), which shows the average of all 197 sweeps. The broad peak in the ON1 trace shows that its EPSPs are indeed temporally coupled to the spikes in AN2.

Another way to demonstrate this relationship is to create a histogram of the number of ON1 EPSPs occurring in discrete time bins throughout the duration of the AN2-spike-aligned sweeps. This requires that the ON1 EPSPs are first 'marked' by the window discriminator. In order to enhance the contrast between the relatively rapidly rising EPSPs and the more slowly shifting membrane potential upon which they are superimposed (Fig. 4(a), top trace), the waveforms were differentiated. Differentiation also emphasized high-frequency noise in the recordings, which was removed by low-pass filtering. The bottom trace in Fig. 4(a) shows the results of this processing. As the cursor shows, the large EPSPs that are apparent in the top trace can now easily be discriminated. The sequence of differentiation followed by low-pass filtering followed by window discrimination resulted in the detection of 734 EPSPs over the series of 197 sweeps. A histogram of their times of occurrence is shown in Fig. 4(b), where it can be seen that ON1 EPSPs are most frequent between 11 and 18 ms, i.e. a few ms before the AN2 spikes, which occur 20 ms after the beginning of each sweep.

A third way to investigate temporal correlation between the two waveforms is by correlation analysis. A 1-s long extract of the original recording was 'binarized' with the window discriminator, i.e. both the AN2 spikes and ON1 EPSPs were converted to 5 ms-long pulses of uniform amplitude. Binarizing converts both the EPSPs and spikes into events of uniform shape, and thus ensures that the result of the subsequent correlation step is determined not by the fine details of the shapes of the original events, but only by their relative timing. The cross-correlation function for the binarized traces is shown in Fig. 4(c). The prominent peak, at a lag of about -5 ms, again shows that ON1 EPSPs tend to occur shortly before AN2 spikes.

5. Modification of SWEEPS

One of the main advantages of using custom, rather than pre-packaged, software is that one can be sure that the program includes all of the fea-

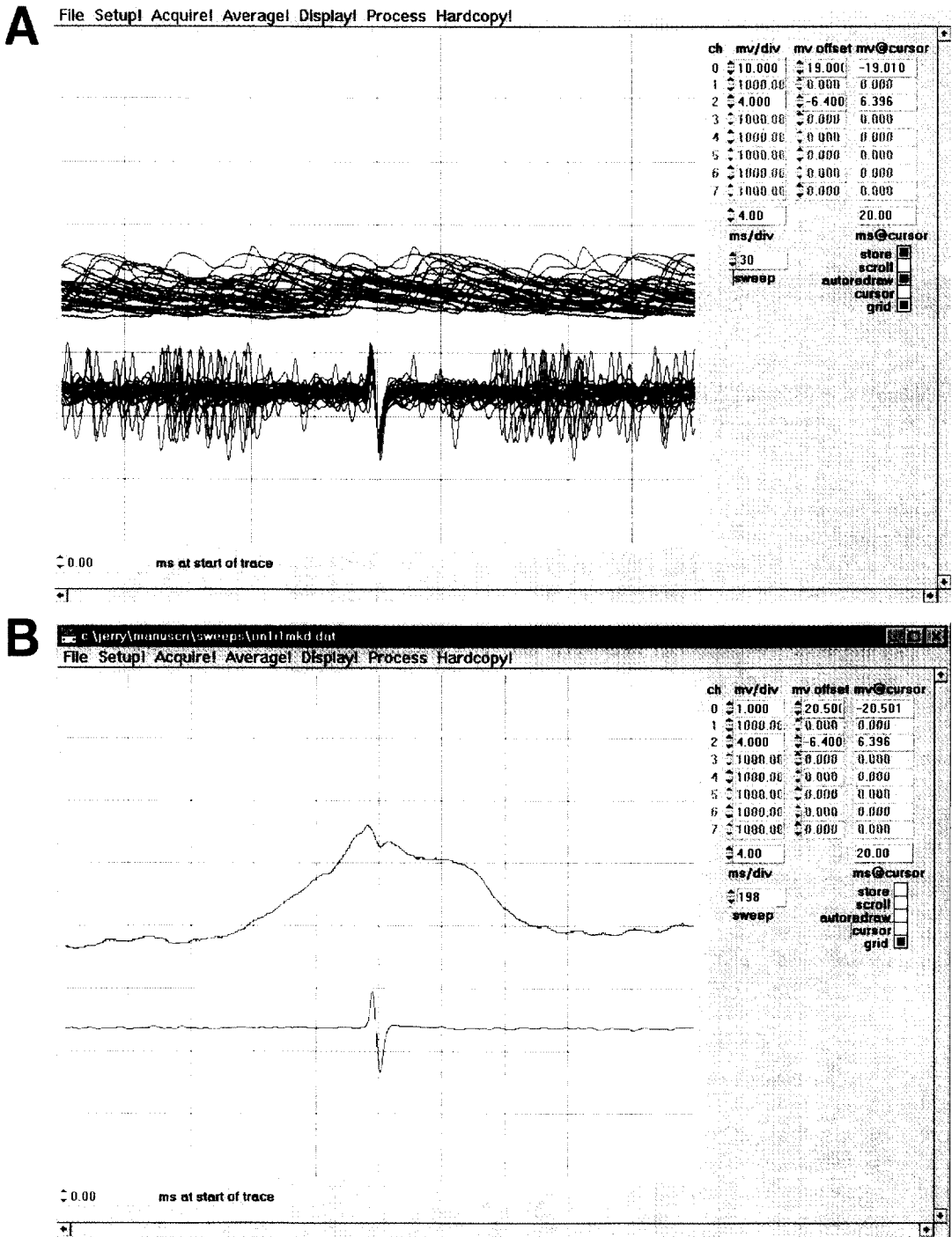


Fig. 3. Temporal relationships between events in two neurons. (a): Superposition of 30 sweeps (of a total of 197) of intracellularly recorded EPSPs in the ON1 neuron (top traces) and extracellularly recorded spikes from the AN2 neuron (bottom traces) that were created by marking spikes in the AN2 neuron in a 5 s-long original sweep. (b): Average of the 197 traces.

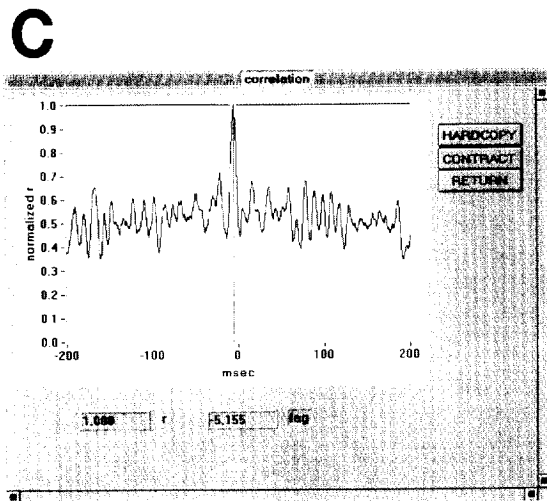
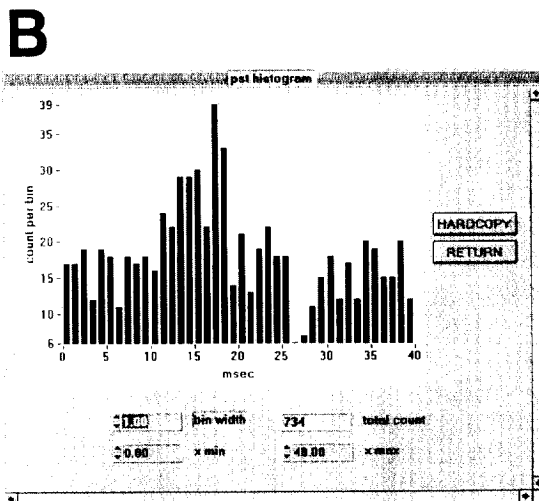
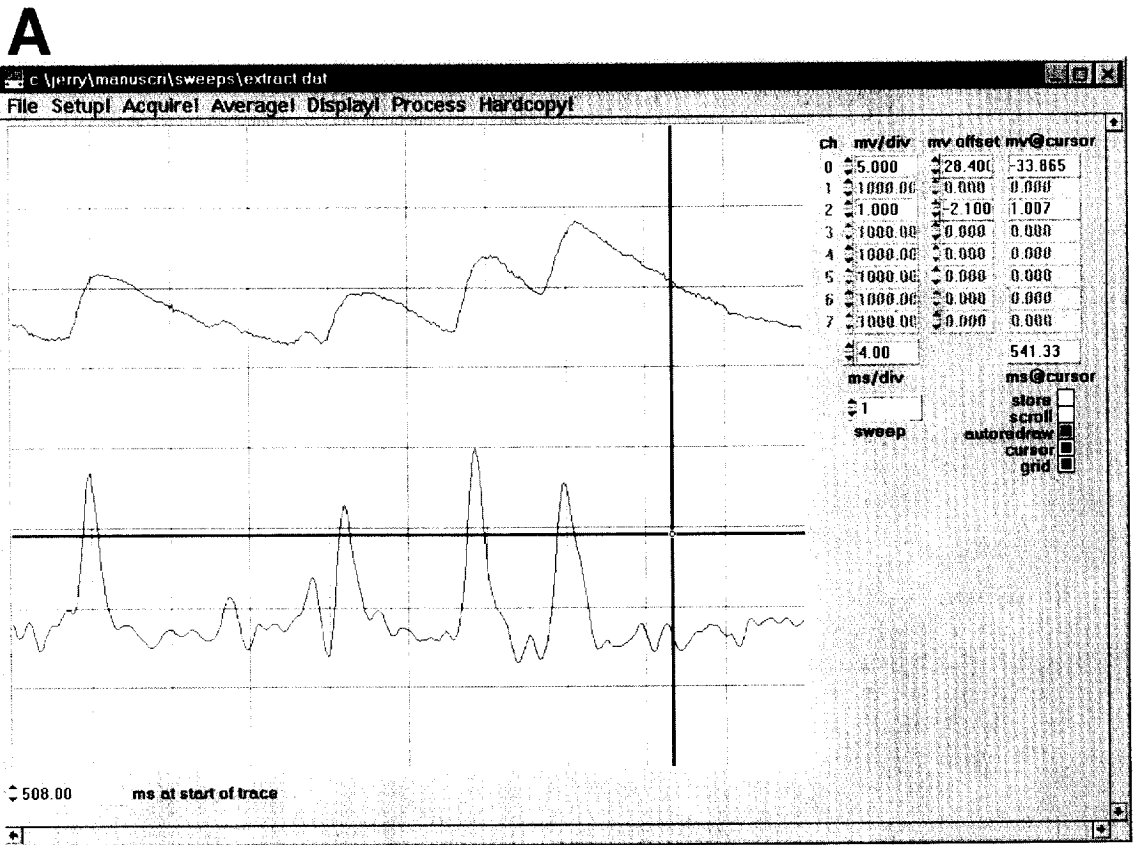


Fig. 4. Analysis of temporal relationships using peristimulus time histogram and correlation. (a): The top trace shows EPSPs in the ON1 neuron; the bottom trace is the same recording after differentiation and low-pass filtering. (b): A histogram of ON1 EPSPs across a series of AN2-aligned sweeps. (c): Cross-correlation function between AN2 spikes and ON1 EPSPs over a 1-s long period. See text for further details.

tures that one requires (and, nearly as important, that the program is not cluttered by unnecessary features). Some central features of SWEEPS, e.g. file input and output, data display, are essential to its operation while others, e.g. some of the data analysis procedures, may be of interest to some users but not others. As SWEEPS was designed to meet my research needs, it will doubtless lack some features that are required by others. Some users might also wish to use algorithms for performing computations that are different from those that are used in SWEEPS (which relies mainly on LabWindows libraries). For example, the FFT function in LabWindows' library requires that the number of data points in its input array be an integer power of two [7]; some users might wish to replace this with a more flexible algorithm [8,9]. Or, some users might wish to replace the LabWindows functions for integration and differentiation, which use, respectively, Simpson's rule and three-point-midpoint algorithms [7], with more accurate alternatives [8,9]. Fortunately, both the modification of existing features and the addition of new ones are fairly straightforward, though they do require basic C-programming skills and purchase of the LabWindows software system.

The functions that accomplish the data processing features listed in Section 3.2 are similar in overall structure. First, a 'panel' is loaded and displayed, so that the user can specify the channel whose data are to be processed and any other parameters appropriate for the procedure. Next, the data to be processed are copied into a local array variable. (As a consequence of the method by which the data acquisition hardware samples multiple channels, data from different channels are interleaved within the data structure that represents a sweep; the data for a single channel must thus be teased out of this structure.) This array is then submitted to the function that actually processes the data. Finally, the results of the processing are displayed, according to the type of operation performed (waveform-modifying, simple measurement, or relationship derived from waveforms; see Section 3.2.3). In order to replace an algorithm currently used in SWEEPS with an alternative, all that is required is to write a func-

tion that implements the new algorithm and substitute the call to the LabWindows library function with a call to its replacement. Adding a new analytical capability requires editing SWEEPS' menu structure so that the new procedure can be selected by the user; creating a panel; writing a function that will perform the analysis (or using one from a LabWindows library); and writing a higher level control function to load the panel, accept user input, and display the results. In most cases, both the panel and the control function can be modeled on others already implemented in SWEEPS. As LabWindows uses an ANSI C compiler, any analytical procedure that can be implemented in C can be incorporated into SWEEPS.

6. Hardware and software requirements

SWEEPS will run on microcomputers equipped with at least an 80386 processor and 8 MB of ram (though a 486 system, with 16 MB ram, is recommended); if the microprocessor is of an -sx variety, then a math coprocessor is required. SWEEPS requires Microsoft Windows 3.1 or higher. If SWEEPS is to be used without modification for data acquisition, then a data-acquisition card from National Instruments is required. SWEEPS was written for, and has been tested on, ATMIO16F5 and ATMIO64F5 cards, though it should work with other National Instruments models as well. SWEEPS supports eight data channels, but it can easily be modified to accommodate a larger number. Modification of SWEEPS requires LabWindows CVI (National Instruments, 6504 Bridge Point Parkway, Austin, Texas, 78730-5039).

The performance of SWEEPS depends, of course, on the specific hardware used. The total amount of data that SWEEPS can accommodate is limited by memory: with 16 MB of RAM, and 24 MB of Windows 3.1 virtual memory, approximately 35 MB total are available for data storage. This is sufficient to acquire approximately 15 min worth of recording of a single channel sampled at 10 kHz (which is adequate for most neurophysiological recordings) or almost 2 min from 8 chan-

nels sampled at 10 kHz each. The maximum overall sampling rate depends both on the data acquisition hardware (the boards listed above can sample at 200 kHz), and on the speed of the computer and its accessories. A 33 MHz 80486-based computer, with a rather slow hard drive, is capable of sustained sampling at 100 kHz (i.e., more than enough to sample 8 channels, each at 10 kHz); sampling at 200 kHz is possible on this machine only if the total memory required can be accommodated without resorting to virtual RAM.

7. Availability of the program

SWEEPS can be obtained from the author, either as an executable program or, for those who own LabWindows/CVI, as source code. SWEEPS file formats are also available for those who wish to import data into SWEEPS for analysis.

Acknowledgements

Supported by the Natural Sciences and Engineering Research Council of Canada. I thank Dr R. Chase for his helpful comments on the manuscript.

References

- [1] D.A. Turner and M. Schlieckert. Data acquisition and analysis system for intracellular neuronal signals. *J. Neurosci. Methods* 35 (1990) 241–251.
- [2] B. Hedwig and M. Knepper. NEUROLAB, a comprehensive program for the analysis of neurophysiological and behavioural data. *J. Neurosci. Methods* 45 (1992) 135–148.
- [3] M.A. Nordstrom, E.A. Mapletoft and T.S. Miles. Spike-train acquisition, analysis and real-time experimental control using a graphical programming language (LabView). *J. Neurosci. Methods* 62 (1995) 93–102.
- [4] G.S. Pollack and K. Imaizumi. Representation of low and high sound frequencies in the auditory nerve of the cricket *Teleogryllus oceanicus*. 10th Int. Meet. on Insect Sound and Vibration. Woods Hole (1996).
- [5] R. Balarishnan and G.S. Pollack. Recognition of courtship song in the field cricket *Teleogryllus oceanicus*. *Animal Behav.* 51 (1996) 353–366.
- [6] G.S. Pollack. Synaptic inputs to the omega neuron of the cricket *Teleogryllus oceanicus*: differences in EPSP waveforms evoked by low and high sound frequencies. *J. Compar. Physiol.* 174 (1994) 83–89.
- [7] LabWindows CVI Advanced analysis library reference manual. (National Instruments Corp., Austin TX 1994)
- [8] W.H. Press, S.A. Teukolsky, W.T. Vetterling and B.P. Flannery. *Numerical Recipes in C: the Art of Scientific Computing*. 2nd Edn. (Cambridge University Press, Cambridge, MA, 1992).
- [9] J.D. Faires and R.L. Burden. *Numerical methods*, (PWS-Kent, Boston, MA, 1993).