Event-related potential analysis using *Mathematica*

TYLER S. LORIG and THOMAS P. URBACH
Washington and Lee University, Lexington, Virginia

Commercial ERP software is often inflexible and may be of little use to investigators using non-traditional forms of ERP analysis. An approach to this problem is presented using *Mathematica*, a high-level, symbolic mathematics and graphical programming language and environment. A description of the software and its application to ERP data are included, as are examples. Sample graphical output is also provided.

Event-related brain potential (ERP) researchers have long sought efficient software for data collection and analysis (Criswell & Babcock, 1978; Lorig, 1985). Many vendors have provided ERP software with sophisticated user interfaces and visualization capabilities. Much of this software, however, has been designed for single cases and is most appropriate for clinical users. Typically, such software provides limited access to data and an inconvenient format for statistical analysis. Single trials may not be stored, and the incorporation of behavioral measures may not be supported. Additionally, algorithms used for averaging, topographical analysis, or eye-movement correction may be poorly documented and offer few options. Because of the proprietary nature of the software, modification of the code is unlikely.

Clinical and nonclinical researchers have responded to these problems in several ways. Some investigators have constructed their own custom code for ERP analysis. Others have created hybrid analysis systems using some commercial code and writing the rest themselves. Still others have accepted the limitations of clinically oriented systems and have worked around the constraints. Although there may be no single ideal solution for everyone, some compromises are better suited to specific purposes than others. This paper outlines a hybrid system that offers the experimental ERP research laboratory a compromise between turn-key and in-house analysis procedures and that is flexible, extensible, portable, relatively inexpensive, and reasonably easy to learn and maintain.

Phases in ERP Research

ERP research consists of several phases. After the design of the experiment, acquisition of the data is normally the first step. This step is crucially important since

no form of analysis can completely compensate for noisy data. For the concerns of this paper, the goal of this first phase is to convert the analog EEG signal into a digital data file that can be averaged and analyzed. This can be accomplished using commercial turn-key systems designed specifically for EEG/ERP research or by using an analog-to-digital converter board and general-purpose data-logging software. General-purpose data-logging software, while typically inexpensive, is not always the best choice since it is often designed for logging thermocouple, pressure, or similar forms of data. These signals are slow, relative to brain activity, and may lead to a restricted range of choices for sampling rates. A similar problem exists at the other end of the continuum. Digital oscilloscopes designed for electronics work tend to sample at very high rates. While this may be good in some cases, it leads to tremendous data files that are not easily manipulated off-line. These systems are also usually limited to a small number of channels. A further problem with these general-purpose acquisition systems is the way in which individual trials are treated. Because ERP experiments often present several different conditions, the software system must be able to parse the individual trials by condition. This means that a trial's condition must be encoded in the logged data or the software must provide a way of accurately reconciling the logged data with a list of trial conditions. Such a procedure is especially difficult when the subject's response on a trial determines the condition (e.g., correct vs. incorrect classification of a target stimulus).

Commercial acquisition systems can also be problematic. Since many commercial systems are targeted toward clinical markets, there is little regard for the parsing of data by condition. Export of the data to other software packages may also be troublesome because of proprietary data formats. Some clinically oriented systems, while offering easy-to-use, menu-driven software, may not offer choices for artifact rejection or eye-movement correction. Investigators should evaluate these and the general-purpose systems by actually conducting a data collection session with the software if possible. For instance, some software may advertise flexible artifact

Development of this software was supported by NIH Grant DC01323 and Washington and Lee University. The authors wish to thank T. Bashore and two anonymous reviewers for their comments and suggestions on an earlier draft of this paper. Requests for reprints should be addressed to T. Lorig, Department of Psychology, Washington and Lee University, Lexington, VA 24450 (e-mail: tlorig@wlu.edu).

rejection and high acquisition rates, but upon close inspection, when one enters the high-rate acquisition mode, artifact rejection is not available.

Once the data have been collected, the data-reduction phase of the experiment can begin. The data files will be of two types: single trials or averages. Averages may seem convenient since, eventually, the single trials must also be converted to an average. There are, however, drawbacks to this approach. Primary among these limitations is the treatment of artifacts. On-line averaging done by some commercial ERP systems leads to limited artifact detection algorithms since they must be conducted during the interstimulus interval and, once identified as an artifact, the trial is usually excluded from the average rather than corrected. This can lead to excessive exclusion of data, especially if one electrode is producing a poor recording. Additionally, the time course of the ERP can be of great importance in some phenomena and, unless the software creates averages after every n trials, this information is lost. Finally, as algorithms for the treatment of artifacts and digital filtering improve, old data sets that consist of single trials can be reanalyzed—a considerable advantage over on-line averaging.

If the data are of the single-trial type, it is necessary to reduce them to averages for each condition in the experiment. This process can be simple or complex depending upon the investigator's goals. Like on-line averaging, trials with high-voltage artifacts can be excluded from the data and the remaining trials averaged according to experimental condition. A more thorough approach might include sophisticated digital filtering of individual trials followed by artifact rejection and eye-movement correction algorithms that use parameters determined by the other trials in the data set. Eventually, these data would also be averaged by condition. The tools to accomplish either of these goals are not likely to be found in a general-purpose program. Despite their growing sophistication, general-purpose, menu-driven laboratory software made to deal with analog data do not have the tools necessary to reduce ERP single-trial data to averages. Such software may be used effectively to smooth data sets, visualize waveforms, or create power spectra, but it is not likely to do the whole job. Commercial ERP software, while not generally as flexible, can move the data from single trials to averages, even though it may not offer all the options the investigator desires.

Once reduced to averages, visualization of the ERP waveforms and scalp voltage distributions usually takes place. Additionally, features of the data (e.g., P300 amplitude) are determined and exported to files for statistical analysis. General-purpose software may be useful here but can also be cumbersome. For instance, picking peaks is often accomplished manually by placing the cursor over the area of interest then exporting the data. Several commercial ERP packages also use this approach. An even more important limitation is the restricted number of options available to the experimenter regarding extracting the data for statistical analysis. As new techniques are developed, the investigator must rely

on the software vendor adding these programs to the package. At best, this produces a significant lag between a leading laboratory and those who must wait on a vendor to get a product ready for market.

An alternative to using a general-purpose laboratory package or a commercial ERP package would be to construct the software in-house. Constructing custom software for ERP analysis and visualization is very difficult, and standards for acceptable packages have changed rapidly. Command line access to ERP data, while the state-of-the-art several years ago, would be viewed suspiciously by new graduate students accustomed to mouse-based graphical user interfaces. Likewise, standards for data visualization are high, and the construction of this code is especially problematic since many graphics standards seem to be in flux. Should an investigator make the effort to construct an ERP analysis package in spite of these difficulties, he or she will be faced with the daunting task of maintaining and upgrading the software over its life. This is not an insignificant problem. Few investigators can spare the time to construct a large-scale ERP package and, if graduate students assist in the coding of the software, they may be unavailable when the next upgrade is necessary. The whole purpose of creating a custom package may be to experiment with new approaches to ERP analysis not available in clinical packages. If this is the case, the software is likely to be in constant revision.

A Hybrid Approach to ERP Analysis

A hybrid approach that is a compromise between turn-key and in-house procedures seems to be the most useful and efficient way to address the problems involved in ERP analysis. There are a number of excellent general-purpose laboratory packages and commercial ERP systems available that, while unable to meet all of an investigator's goals, can be used effectively. Almost all investigators currently use this hybrid approach in a limited way since many use some form of spreadsheet or graphics package to represent ERP waveforms. Data are collected, analyzed, displayed, and exported to a spreadsheet or graphics program to create a publication-quality graphic.

The advantage of the hybrid approach advocated here is that there is less software to write. The investigator may write a small program that averages the data or completes some unique exploratory analysis. The disadvantages of the hybrid approach are that writing even limited programs can be time consuming and that exporting and importing data to and from heterogeneous packages can be inefficient and occasionally maddening. To reduce the disadvantages of the hybrid approach, we suggest using a single, high-level, customizable language and environment for ERP reduction/analysis and graphics.

Mathematica (Wolfram, 1991) is a sophisticated programming environment that is available on many popular computers (DOS, Windows, Macintosh, Sun, RS6000, NeXT, SGI, etc.). *Mathematica* is unusual because it contains high-level language for mathematics and graphics.

Many of the transforms routinely used by ERP researchers are built-in functions in this language (e.g., Laplace, Integrate). Additionally, *Mathematica* operations (on most of the computers mentioned above) take place in a "notebook" environment that provides a historical record of how an analysis was completed and its numeric and graphical results. It is little wonder that many calculus instructors have turned to this software for teaching purposes.

To use *Mathematica* for ERP analysis, previously collected data are "read into the notebook" in the form of a multidimensional data matrix usually consisting of a subject $\times$ condition $\times$ channel $\times$ timepoint array. Once in the notebook, the data may be analyzed and displayed by a series of built-in and customized commands (customized commands are simply a list of regular *Mathematica* commands defined in a file and assigned a function name). Commands are used in a toolbox approach to manipulate, parse, and display data. Once defined, a data matrix remains available throughout the entire analysis. For instance, a command might be given to average a set of data. Commands to construct a grand mean and graphs of individual waveforms would follow. Peak and latency commands as well as other ERP commands can be used to find and correlate ERP and behavioral data within the *Mathematica* notebook. Export of these data, in ASCII,

allows the creation of data files for high-level statistics packages, such as SAS, BMDP, or !STAT. It would, of course, be possible to create a specialized statistics package using the *Mathematica* language (a one-way between-subjects ANOVA is currently included in the *Mathematica* distribution package). Scalp topography maps may be created using the built-in graphics command `ContourPlot[]` and three-dimensional surfaces or figures may be constructed to represent data using a single command, such as `ListPlot3D[]` (see Figure 1). Novel or seldom-used forms of ERP analysis are relatively easy to implement. Commands may be selected to explore the data as the investigator chooses. This toolbox-type approach is extremely flexible and can be implemented across many different ERP paradigms.

It should be noted that there are several other languages similar to *Mathematica* that may also be useful for ERP analysis. These include such languages as Maple, MATLAB, and ASYST (Tassinary, Marshall-Goodell, & Cacioppo, 1985). Our choice of *Mathematica* 3 years ago was based on features now shared by many of these packages. These critical features included efficient array operations on real data, the ability to easily import massive data files, and the availability of sophisticated and flexible graphics, such as contour plots. Multiplatform

```
ERPPlot[average[[1,18]],8,-8,1]
ERPPlot[average[[3,18]],8,-8,2]
```

```
STM[average[[1, Range[32], 80]],
10,-20]
```

```
Show[STM3DGray[average[[1,
Range[32], 80]],10,-20]]
```

```
ListPlot3D[ midline]
```

Figure 1. Examples of *Mathematica* graphical output and the associated commands.

compatibility was also a key issue. One important feature often overlooked in selecting software is the availability of print and electronic resources. *Mathematica* has an active newsgroup and listserv service and, because of its wide usage in other fields, a variety of books have been published on *Mathematica*. All of the packages, including *Mathematica*, continue to expand and provide new features. Investigators interested in evaluating the various packages should note that some packages treat real data differently from symbolic functions. Thus, while a package may advertise its ability to do a contour plot, it may be able to do this operation only on a symbolic function rather than real data. The features and capabilities of *Mathematica* 2.2 that relate to ERP analysis are described below.

Mathematica Components

At the center of *Mathematica* is a wide array of information-processing functions and the notebook environment. These are complemented with other features that increase the flexibility of the program. The following list is far from complete and only reviews those features that are of particular interest for present purposes.

Notebook environment. This is the glue that holds *Mathematica* together. In initial appearance and behavior, it looks like the screen of a full-screen text processor (see Figures 2 and 3, for examples). Text can be typed in, cut, and pasted, and the entire document can be saved in a file or printed. The difference, however, is that once the text has been typed, for example,

$$2 + 3$$

or

```
q = Fourier[myDataSet];
```

it can then be *executed*. For instance, the first simple expression above would evaluate to 5 and the second would evaluate to the Fourier transform of the time series data in the array variable *myDataSet*. The notebook can be segmented into different units or cells, and these cells can be executed individually or all at once, in sequence.

Symbolic functions. Here, "function" is understood in the sense of mapping a set of inputs to a unique output, but not all functions are numerical. The inputs might be ERP data sets and the output might be the Fourier transform, but the inputs might also be graphical plot. There are a variety of predefined symbolic mathematical functions, ranging from simple arithmetic and matrix operations, through various transforms (e.g., Laplace and Fourier), to real analysis. Many of these functions are useful in the analysis of digitized waveform data.

Visualization. The graphical resources include X-Y scatter plots and line graphs, pie charts, bar graphs, two-dimensional density plots, two-dimensional contour plots, three-dimensional contour plots, and sets of three-dimensional polygons (see Figure 1). These plots can be generated from symbolic functions or explicit lists (arrays) of data points. Display options are flexible and include the ability to add legends, labels, grids, borders,

and arbitrary text, and to display in gray-scale or color, overlap, combine, and resize plots into other plots. In addition, a set of plots can be animated by displaying a temporal sequence of graphics in a single location, thus providing a fourth (temporal) dimension for the visualization of information.

Memory and system management. Theoretically, all implementations of *Mathematica* take advantage of high-capacity hard disks to provide virtual memory. Although using the hard disk slows program execution, it allows enormous data sets (e.g., every data point for every one of 16 subjects to be assigned to a single array variable for subsequent processing). This memory capacity, together with *Mathematica*'s built-in array processing features, permits the entire data set for an experiment to be processed in one operation. In actual use, *Mathematica* is highly dependent on RAM for system overhead and will slow to a stop if RAM is unavailable (16 MB, in our experience, is sufficient for most ERP research). In the analysis of a typical experiment (16 subjects, 8 conditions, 200 time points), 40–50 MB of hard disk space is used as a swap area by the virtual memory system.

Extensible/user-defined functions. If an operation is needed that *Mathematica* does not have built-in, the user can define it either by using a collection of predefined *Mathematica* functions (a macro or *Mathematica* "program") or by running an external program (e.g., FORTRAN, C) from the notebook interface (this feature is not available in the Macintosh version since there is no equivalent of the command line). An illustration of a user-defined function macro for reading ERP files into the notebook is given below.

```
ReadERP[ subjv_, nsubj_] :=
Module[ {},
Table[ Get[StringJoin[ subjv[[i]],
".erp"]], {i, nsubj}]]
```

Here, the command *ReadERP[]* is defined with two variables as arguments: the subject name array (*subjv*) and the number of subjects (*nsubj*). The function joins each subject name to ".erp", then reads the file from disk into an array, such as *all_the_data*. The example below is an illustration of how this command would appear in a notebook.

```
all_the_data = ReadERP[ mysubjects, 16];
```

There is typically a performance penalty for using predefined *Mathematica* functions because of the interpreted nature of the language. Functions that are computationally intensive and often used may be written and compiled in a programming language, such as C, and then called as an external function from the *Mathematica* notebook.

Portability. The versions of *Mathematica* that run on different machines are largely interoperable (some command line features are not available in all versions). We routinely export *Mathematica* notebooks created on SGI and NeXT computers to Windows and Macintosh versions of the software. This means that, as computer hard-

ware is upgraded, the investment in developing *Mathematica* analysis routines will be preserved as long as the new equipment supports this language.

Program control. For conditional execution and repetitive operations, *Mathematica* implements standard procedural control constructs, such as If-Then statements and Do-Loops.

Implementation in ERP Labs

The first step in using *Mathematica* for analyzing ERP data begins with the reduction of the raw single trials into subject means. In our labs, we implement this phase of data analysis with variations of a C program. The programs might have been constructed using *Mathematica*, but, because of the time consuming nature of this first phase and its repetitive use on different subjects, optimized code was preferred. Two different data-collection systems are used in our labs. Both systems are commercially available, although they differ in their capabilities, interface, and file formats. The first system is based on CODAS (Dataq Systems and Coulbourn Instruments), a general-purpose computerized oscilloscope system that is triggered by a second computer generating lexical stimuli on a video monitor. It digitizes 16 channels of EEG data and stores the data in a sequential binary format. Trial and behavioral information is embedded in a separate file. A C program (CODAStoERP.c) reads the EEG and trial data files, parses the trial on the basis of its condition, checks the data for high-voltage artifacts, corrects trials containing eye movements (Semlitsch, Anderer, Schuster, & Presslich, 1986), and

digitally filters the data returning separate *Mathematica* format files for the subject and item means. A summary file of rejected and corrected trials is also produced.

The other system is based on Neuroscan's 32 channel Syn-Amps and ACQUIRE software. It produces a combined trial name and EEG data file in a binary format. This system is used for olfactory and visual ERPs. These data are reduced using a variation of CODAStoERP.c (EEGtoERP.c), and similar output is obtained.

Typically, *Mathematica* is used to "batch process" the subjects' raw data. A Do loop repeatedly executes a command line call to EEGtoERP and substitutes the subject ID strings in the command line, as in:

```
mysubjects={"cls", "cwj", "mma", "fas",
            "tim", "tww"};

Do[Run["EEGtoERP", mysubjects[[i]], "0 20
75 y"], {i, 6}];
```

In this example, an array of subject names is defined as mysubjects. Run[] is a call to the external program: EEGtoERP. Each subject's raw data is filtered outside 0–20 Hz, trials with voltages greater than 75 μ V are rejected, and eye movements are corrected (y). This results in the storage of averages for each subject in a *Mathematica* format file. The next step is to read the set of function definitions used for processing ERP data. The externally defined functions routinely used for ERP analysis are listed in Table 1. These definitions/functions are contained in the file ERP.m and are read into the *Mathematica* notebook by the following command:

Table 1
Extensions (Macros) to *Mathematica* Developed for ERP Research

ReadERP[subjectlist, nsubjects]	Reads sss.erp files into list.
ERPPlot[datalist, maxvolt, minvolt, dashpattern]	Plots ERP waveform specified in datalist.
GrandMean[datalist, nsubjects]	Creates grand mean from datalist.
PickPosPeak[datalist, nsubjects, nconditions, startpoint, endpoint]	Returns list of maximum values within the selected window.
PickNegPeak[datalist, nsubjects, nconditions, startpoint, endpoint]	Returns list of minimum values within the selected window.
MakeAnovaFile[datalist, subjectlist, conditionlist, nsubjects, nconditions, filename]	Creates ANOVA file with data specified in datalist.
IntegratePeak[datalist, nsubjects, nconditions, startpoint, endpoint]	Returns list containing sum of voltages between startpoint and endpoint.
Correlate[datalistA, datalistB, nobervations]	Returns Pearson's product moment correlation coefficient.
STM[datalist of electrode values at single time point, maxvolt, minvolt]	Returns contour plot of 30-channel EEG. Black is more positive.
SCD[datalist of electrode values at single time point]	Deblurs voltage list using estimate of Laplace transform.
STM3D[datalist of electrode values at single time point, maxvolt, minvolt]	Returns color 3-D Graphics object of the 30-electrode montage. Red is positive. Display output using Show[].
STM3DGray[datalist of electrode values at single time point, maxvolt, minvolt]	Returns gray scale 3-D Graphics object of the 30-electrode montage. White is more positive. Display output using Show[].
SCD3D[datalist of electrode values at single time point]	Deblurs voltage list using estimate of 3-D Laplace transform.
Spin[3D graphics object, increment (0-4)]	Creates a series of 3-D images that rotate through 360°. The number of images is dependent on the increment.

```
<<ERP.m
```

Once the definitions are present, the entire set of averaged subject data are then read into the notebook resulting in a subject $\times$ condition $\times$ electrode $\times$ time matrix. The remaining data analysis consists of partitioning, isolating, combining, and manipulating this matrix. Data are read with the following line of code and assigned to the name, *data* in this example:

```
data=ReadERP[mysubjects, 6];
```

To display the data from all the subjects, an investigator mixes ERP-specific functions with built-ins, as in:

```
Do[ERPlot[data[[i, 1, 18]], 25, -25, 2],  
  {i, 6}]
```

In this case, the data for the 18th electrode (Cz) and first condition (1) are plotted for each of the 6 subjects (*i*) within a range of 25 to $-25 \mu\text{V}$, with a line of style 2 (dashed). The function *ERPlot[]* is defined in the file *ERP.m* and *Do[]* is a built-in *Mathematica* command.

To obtain grand mean waveforms and store the resulting file, the following commands are given:

```
grand=GrandMean[data, 6];  
grand>>GrandMean6.erp;
```

where *grand* is the array containing the grand average, and *GrandMean6.erp* is the file that will contain the average of the 6 subjects. This condition $\times$ electrode $\times$ time matrix (*grand*) can be further manipulated. For instance, if conditions are arranged in a factorial fashion, it may make sense to combine some conditions and view the average. One can see the dimensions of *grand* by issuing the command:

```
Dimensions[ grand ]
```

The resulting output would appear as: *grand*{4, 32, 200}, with four conditions, 32 channels, and 200 time-points. Since condition is the first factor of the grand mean matrix, a single condition can be selected by using pairs of double brackets, as in

```
cond=grand[[2]];
```

Now *cond* is the electrode $\times$ time matrix for Condition 2. Thus, the average of two conditions may be given by

```
factorA=(grand[[1]]+grand[[3]])/2;  
factorB=(grand[[2]]+grand[[4]])/2;
```

Grand means or individual subject data may be displayed as a line graph (*ERPlot[]*), a contour plot (*STM[]*), or a three-dimensional surface (*List3DPlot[]*). Electrode 18 for factor A could be plotted with the *ERPlot[]* command used previously, as in

```
ERPlot[factorA[[18]], 15, -15, 1]
```

As noted previously, titles, labels, tic marks, color, and other plotting parameters may be controlled using built-in *Mathematica* graphics directives. Furthermore, animation of any set of graphs may be obtained simply

by double-clicking on a single graphic in the series. For instance, to watch the temporal evolution of grand mean voltage maps (in condition 1), execute the following *Do* loop:

```
Do[STM[grand[[1, Range[32], i]]],  
  {i, 200}]
```

This results in a series of 200 two-dimensional graphics representing the scalp topography maps for 32 electrodes in Condition 1 with *i* representing the time point. A user may double-click on any map in the set, and each map in the series will be successively displayed in the same area. The speed and direction (forward or backward) of this animation can be controlled from the keyboard. Like all graphics generated by *Mathematica*, multiple or single graphic panels can be selected and printed or copied to another application. Since PostScript is used for graphical output, figures copied to other PostScript applications can be resized or further manipulated.

Limited statistical analysis packages have been implemented using *Mathematica*, but it is far more efficient to create a file for use by a commercial statistics program. This is accomplished by first defining a data set (as in peak picking) and using the *MakeAnovaFile[]* command, as in the following example:

```
conditions={"vis mat", "vis nonmat", "olf  
mat", "olf nonmat"};
```

Figure 2. A *Mathematica* notebook (NeXT) showing a typical ERP analysis. Bold type represents commands typed by the investigator. Program output is shown in unhighlighted type.

```
p3=PickPosPeak[data 6, 4, 25, 40];
MakeAnovaFile[p3,mysubjects, 6, 4, "p3.6"];
```

Here, the conditions array is defined with the labels for each condition of the factorial experiment consisting of visual and olfactory conditions crossed with matching and nonmatching stimuli. Then, the largest positive peak (*PickPosPeak*) between time points 25 (250 msec) and 40 (400 msec) was found for each electrode, condition (4), and subject (6). The resulting array (*p3*) is written to an ASCII file (*p3.6*) that includes labels for the subjects, electrodes, and conditions. This file format can be used by SAS, *STAT* (Perlman & Horan, 1986), and a number of other programs. Recently, we completed an experiment involving 15 subjects in four conditions (approximately 100 trials) with 32 electrodes. The entire data analysis, from raw data to finished ANOVAs and graphics, took less than 30 min to complete using *Mathematica* and resulted in an "analysis document" (see Figure 2, for an example) that included the sequence of steps in the analysis with accompanying graphical results. Such

documents may be especially valuable in training students since the series of steps and results are easily evaluated for logical errors—a feature few menu-driven ERP programs offer.

Besides extracting data, *Mathematica* offers excellent opportunities to try out new (or retry old) techniques for the analysis and visualization of data. Lehmann (1987) has reviewed the use of spatiotemporal mapping of EEG and ERPs to make inferences about such data. This type of mapping takes a spatially contiguous series of electrodes and creates a contour map to illustrate their changing voltage over time. Few commercial EEG/ERP programs implement such a visualization tool, and passing the data for a set of contiguous electrodes to a general-purpose contour graphics package would be a multistep operation that would have to take into account file compatibility issues between programs (e.g., binary or ASCII, and data order). Accomplishing this in *Mathematica* is quite easy (as Figure 3 illustrates). After reading in the data, a contour plot is generated using the midline electrodes. If the investigator wished to see a temporal region

Figure 3. A *Mathematica* notebook (SGI) illustrating spatio-temporal mapping. Bold type represents commands typed by the investigator. Program output is shown in unhighlighted type. Text balloons were added.

Figure 4. Illustration of the ListDensityPlot command applied to an experiment with 60 trials and 180 timepoints. Data for electrodes at Cz (top) and VEOG (bottom) are included. Negative voltages are shown in darker shades. Text balloons were added.

in more detail, the plot could be resized by dragging its edge, or the command could be reissued with a more limited range of timepoints, as the following illustrates:

```
ListContourPlot[data[[1, Range[17,20],
Range[100,200]]]]
```

This will cause timepoints 100–200 to be plotted with the midline electrodes (17–20) for Condition 1.

Another unique visualization technique that can prove useful in understanding ERP data is the density plot. This type of plot converts two-dimensional data sets into a rectangular set of pixels according to the values in the data set. Figure 4 illustrates two such plots. These plots represent every trial and timepoint of a 60 trial experiment for electrodes Cz and VEOG (vertical electro-oculogram). The fluctuating voltage recorded at the electrode is automatically assigned a gray-level (density) with greater negativity in darker shades. This is done with the following command:

```
ListDensityPlot[subData[[11]]]
```

where 11 is the number of the VEOG channel in the trial $\times$ timepoint array subData. This technique is very useful for examining artifacts in data. Here, in one graphic,

are all the data from a single channel. As can be seen in the VEOG data presented in Figure 4, a number of white areas appear around timepoint 150. This indicates that this particular subject made blinks (a high positive voltage) around this timepoint in a number of trials. Consistent artifacts such as these can lead to problems in grand averages despite good EOG correction techniques. The density plot for Cz indicates no temporal areas with consistent artifacts. *Mathematica* also offers the ability to combine density or other plots so that data from every electrode can be arranged in a graphics array corresponding to the location of the electrode on the head.

Using *Mathematica*

Mathematica is quite different from C, FORTRAN, or BASIC. Memory allocation and segmentation are controlled automatically and descriptive error messages appear immediately following an error. It also differs somewhat from MATLAB and Maple in syntax and built-in functions. While an excellent tool for ERP analysis, it can also be frustrating. Failure to include the ending semicolon in some commands can cause an entire data set to be printed to the screen—a very time-consuming process. Once defined, arrays occupy memory until the program ends. For instance, if an array called mydata is created, then, later, a new data set is given the same name, memory use is doubled even though the program appears to have written over the first use of mydata. This “feature” can be turned off by using *Mathematica*’s Share[] command. In spite of the arcane nature of this language, within a few hours, a novice can become competent enough to create usable notebooks, and the peculiarities of the syntax become more obvious. Several excellent texts and references for learning *Mathematica* exist (Gaylord, Kamin, & Wellin, 1993; Gray & Glynn, 1992; Wolfram, 1991).

One of the most useful features of this language is its wide usage in other disciplines. This has led to an active listserv group (mathgroup@yoda.physics.unc.edu) and many anonymous ftp sites (e.g., otter.stanford.edu:mma/). These ftp sites contain excellent notebooks for teaching signal analysis and similar topics. We also use *Mathematica* notebooks to teach ERP analysis techniques to undergraduates (TeachERP.ma).

Program Availability

The source code for the programs CODAstoERP.c and EEGtoERP.c, the *Mathematica* notebooks TeachERP.ma and Example.ma, and definition file ERP.m are available via anonymous ftp from liberty.uc.wlu.edu and can be found in the pub/mma/neuro/ExampleERP/ directory. Investigators interested in these programs should note that some modification of the code is likely to be necessary before they can be used in other labs. For instance, the particular mapping of electrodes to channels is fixed in our labs, and the software is similarly fixed. Investigators using a different mapping will need to make changes in the routines that use this data. Such modifications should be straightforward, however, once

the programmer has experience with the *Mathematica* language.

REFERENCES

- CRISWELL, H., & BABCOCK, B. (1978). A flexible low cost signal averager. *Electroencephalography & Clinical Neurophysiology*, **45**, 535-537.
- GAYLORD, R. J., KAMIN, S. N., & WELLIN, P. R. (1993). *Introduction to programming with Mathematica*. New York: Springer-Verlag.
- GRAY, T. W., & GLYNN, J. (1992). *Beginner's guide to Mathematica*. New York: Addison-Wesley.
- LEHMANN, D. (1987). Principles of spatial analysis. In A. Gevins & A. Remond (Eds.), *Handbook of electroencephalography and clinical neurophysiology: Methods of analysis of brain electric and magnetic signals* (pp. 309-354). Amsterdam: Elsevier.
- LORIG, T. S. (1985). Event-related potential data acquisition using the Apple II+/IIe. *Behavior Research Methods, Instruments, & Computers*, **17**, 479-483.
- PERLMAN, G., & HORAN, F. L. (1986). Report on UNIX! STAT Release 5.1: Data analysis programs for UNIX and MSDOS. *Behavior Research Methods, Instruments, & Computers*, **18**, 168-176.
- SEMLITSCH, H. V., ANDERER, P., SCHUSTER, P., & PRESSLICH, O. (1986). A solution for reliable and valid reduction of ocular artifacts, applied to the P300 ERP. *Psychophysiology*, **23**, 696-703.
- TASSINARY, L. G., MARSHALL-GODELL, B. S., & CACIOPPO, J. T. (1985). Microcomputers in social psychophysiological research: An overview. *Behavior Research Methods, Instruments, & Computers*, **17**, 532-536.
- WOLFRAM, S. (1991). *Mathematica: A system for doing mathematics by computer* (2nd ed.). New York: Addison-Wesley.

(Manuscript received February 3, 1994;
revision accepted for publication August 24, 1994.)