Behavior Research Methods, Instruments, & Computers
2000, 32 (2), 290-296

MATLAB and graphical user interfaces: Tools for experimental management

ERIN M. HARLEY and GEOFFREY R. LOFTUS
University of Washington, Seattle, Washington

MATLAB is a convenient platform for the development and management of psychological experiments because of its easy-to-use programming language, sophisticated graphics features, and statistics and optimization tools. Through implementation of the Brainard–Pelli Psychophysics Toolbox, the MATLAB user gains close temporal and spatial control over the CRT, while retaining the simplicity of an interpreted language conducive to rapid program development. MATLAB's abilities can be further utilized through easily programmable graphical user interfaces (GUIs). We illustrate how a GUI can serve as a powerful and intuitive tool for organizing and controlling all aspects of a psychological experiment, including design, data collection, data analysis, and theory fitting.

We describe here three related topics: first, the use of MATLAB in psychological research; second, the value of graphical user interfaces (GUIs) in organizing and running experiments; and third, our implementation of one such GUI within the MATLAB environment.

MATLAB AND PSYCHOLOGICAL RESEARCH

Over the past few years, MATLAB (a product of The Math Works Inc.) has gained popularity as a means of running psychological experiments, particularly those involving perception and cognition. MATLAB, which stands for *matrix laboratory*, is a powerful platform for high-performance mathematical computation and graphical representation, whose basic data element is an *N*-dimensional matrix (Hanselman & Littlefield, 1996). MATLAB's primary strength as a means for creating and executing psychological experiments lies in its comprehensive package of superior mathematical and graphical tools. Within this single application, all of the following experimental tasks can be achieved: creation and manipulation of stimuli, design and execution of the experiment, data collection, statistical analysis, fit of complex theory to data, and interactive graphical data display.

MATLAB runs on Macintosh,[1] Windows, and UNIX operating systems and includes both a programming language and a large set of mathematically oriented libraries.[2] MATLAB serves as a good alternative to popular platforms, such as PSYSCOPE (Cohen, MacWhinney, Flatt, & Provost, 1993; Yee & Vaughan, 1999) and MEL (St. James & Schneider, 1991; Schneider, 1989), because it provides greater control over experimental design, allowing for the creation of entirely unique experiments. MATLAB also serves as a good alternative to low-level programming languages, such as C and PASCAL, because MATLAB's language is abstracted from the hardware details and is generally easier to learn. As a result, a MATLAB user receives the benefits of a programming language that is both conducive to rapid program development and able to perform with the power and flexibility of a low-level language (primarily through the addition of the Psychophysics Toolbox, discussed below).

The Psychophysics Toolbox

MATLAB is particularly useful for running sensory and perceptual experiments, largely owing to an extensive set of tools—the Psychophysics Toolbox,[3] written by Denis Pelli and David Brainard—that allows extremely close temporal, spatial, and color-related control over CRT-based stimulus displays (Brainard, 1997; Pelli, 1997). Specifically, the toolbox provides access to the computer's display frame buffer and color lookup table, allows synchronization of stimulus display with the vertical retrace, supports millisecond timing, and facilitates the collection of observer responses (Brainard, 1997). The toolbox incorporates these features into a large collection of stimulus–display routines that allow the user to store preformed visual images (e.g., scenes, forms, text, or some combination of these) in the computer's memory and to display these images at rates that are constrained only by the monitor's refresh rate. For example, on a Macintosh G3 computer, a 400 × 400 pixel gray-scale image can be transferred from memory to the screen buffer in less than 10 msec, thereby allowing a sequence of such images (the number of which is limited only by computer memory) to be displayed at the fastest possible rate.

The writing of this manuscript was supported by NIMH Grant MH41637 to G.L. We thank David Brainard for enormous amounts of help in understanding the Psychophysics Toolbox and MATLAB in general. The MATLAB code discussed in this article will be posted on E.H.'s Web site (http://students.washington.edu/eharley/). Please e-mail either of the authors for details. Correspondence concerning this article should be addressed to E. M. Harley, Department of Psychology, University of Washington, Seattle, WA 98195-1525 (e-mail: eharley@u.washington.edu).

The Psychophysics Toolbox has been used by researchers to study a variety of topics in the field of psychology, including face and object recognition, psychophysical thresholds, color matching, visual search, categorization, motion detection, and perceptual learning. Later in this article, we will briefly discuss our use of the toolbox to display high- and low-pass filtered digit strings for very brief exposure durations.

Tools for Data Analysis and Theory Fitting

MATLAB provides a wide range of functions for use in data analysis. For example, the optional Statistics Toolbox includes more than 200 routines covering topics including, but not limited to, descriptive statistics, probability distributions, linear and nonlinear modeling, multivariate statistics, and hypothesis testing. MATLAB also provides graphing tools for the creation of presentation-quality plots of various forms, including line, scatter, stem, pie, and box plots and bar graphs and histograms, as well as three-dimensional contour, surface, and mesh plots (with optional animation).

Experimenters in the field of perception and cognition often wish to fit mathematical theory to their data. MATLAB's optional Optimization Toolbox provides the tools necessary for fitting complex, nonlinear mathematical models with a number of free parameters but no analytic solution. For a given data set, the optimization routines find the values of the free parameters that minimize some desired error function (e.g., the root-mean square error between the data and the theory predictions).

MATLAB's Intuitive Programming Language

MATLAB incorporates a programming language that is similar in structure to many common languages, such as FORTRAN, BASIC, PASCAL, and C. One element that makes MATLAB's language unique is that, as was noted, it uses a matrix as its basic element. This property confers numerous benefits, one being the following: In most programming languages, it becomes time consuming and cumbersome to perform the same mathematical operation on a group of numbers, because the operation must be repeated (generally via loops) for each number, one number at a time. The greater the dimensionality of the data structure on which such operations are performed, the greater the number of embedded loops required. However, in MATLAB's programming language, data structures are construed as matrices, and almost any operation, be it as simple as adding 1 to each structure element or as complex as finding all elements of a structure that are less than 0, can be performed in a single step. Embedded loops for these tasks are thus eliminated in MATLAB, saving the programmer time and cutting down on the length and complexity of programming routines. A second benefit of MATLAB's matrix-based language pertains to the use and manipulation of images. Images are stored in MATLAB as matrices, and because of this, they can be quickly and easily manipulated or altered (e.g. filtering, adding noise, reducing/increasing contrast, etc.) through matrix operations.

Like other programming platforms, MATLAB provides automatic formatting features, such as color coding of comments, strings, and key words, as well as indentation of loops, thus increasing clarity and organization of programming code. The net effect of such features, combined with MATLAB's simple and intuitive programming syntax, is to allow someone with little or no prior programming experience to quickly become a proficient MATLAB programmer and to write sophisticated routines within a matter of weeks.

Graphical User Interfaces

There are essentially three ways in which a user can communicate with the computer via MATLAB: through the Command Window, through the use of scripts and functions, and through GUIs.

The *Command Window* is MATLAB's default I/O technique. As the name implies, it is a window into which any standard MATLAB commands or user-defined commands, such as "2+2" or "answer = conv2 (mask, filter)," can be typed. Although the Command Window is adequate for accomplishing simple tasks, it is often useful to create a file containing a list of such commands. These files are called *scripts* if they simply run a list of commands and *functions* if they accept input arguments and/or return output arguments. Both scripts and functions can be executed either directly from the Command Window or from within other scripts or functions. To provide an example, a function that we wrote and use often is called "ComputeContrast." It requires two input values, foreground color (FC) and background color (BC), and returns the contrast value calculated by the formula $(FC - BC)/(FC + BC)$.

The third communication device, a GUI, provides an intuitive interface between the user and the programming language. A GUI allows the user to bypass MATLAB commands altogether and, instead, to execute programming routines with a simple mouse click or keypress (most common applications, such as Word and Photoshop, as well as the Macintosh and Windows operating systems, are implemented as complex GUIs). No knowledge of MATLAB or computer programming is necessary for a user to successfully navigate a well-designed GUI; indeed, from the user's perspective, the language underlying the GUI is irrelevant. GUIs can range from simple question boxes prompting the user for a Yes/No response, to more complex interfaces, an example of which will be provided below. MATLAB provides the user with intuitive tools for easy construction of GUIs; see Marchand (1999) for a detailed description of the GUI-creation tools provided by MATLAB.

USING GRAPHICAL USER INTERFACES IN PSYCHOLOGICAL EXPERIMENTS

To illustrate the benefits of using GUIs in psychological experiments, we will describe a specific GUI created to accompany a visual perception experiment in our laboratory.

Example: The Filter Experiment

To provide the reader with a foundation for understanding and interpreting our example GUI, we will first briefly describe the experiment for which the GUI was created. The experiment, which we will call the *filter experiment*, was designed to investigate certain kinds of spatial-frequency filtering on perception and memory for briefly presented digit strings (see, e.g., Olds & Engel, 1998; Parish & Sperling, 1991). Stimuli were randomly chosen four-digit strings presented in three spatial-frequency conditions: normal (N), low spatial frequencies only (LSF), and high spatial frequencies only (HSF). On a given trial, the following sequence of events occurred. First, a stimulus in one of the three spatial-filtering conditions was presented at one of six exposure durations; next, the observer attempted to recall the digits in their correct order (guessing if necessary); and third, visual and/or auditory feedback was provided. The programs used in the filter experiment were all written in MATLAB, utilizing standard MATLAB routines in conjunction with the stimulus presentation routines in the Brainard–Pelli Psychophysics Toolbox.

Organizing Multiple Data Sets

Fourteen observers participated in the filter experiment and produced a total of 64 data sets (where a *data set* is a set of experimental trials, all involving the same experimental parameter values). Three observers collected pilot data during the developing stages of the experiment. Three different observers collected data for the final four versions of the experiment (in which mask/ no mask and low-pass filter size were varied). Finally, 8 observers collected data to serve as practice for a follow-up experiment and to determine the appropriate experimental parameters (contrasts, durations, etc.) to be used in that follow-up experiment. Although it might seem as if organizing all of these observers and data sets, with their differing experimental parameters, would be time consuming and complicated, a GUI designed expressly for this experiment allows for simple control over the multiple data sets from the multiple observers.

The Example Graphical User Interface

The GUI created to accompany the filter experiment consists of two windows: a *main window*, pictured in Figure 1, and a *theory window*, pictured in Figure 2, which we will discuss in turn.

The GUI's main window contains four major components: observer information, setup, and stimulus preview; feedback options and data collection; data analysis, plot options, and graphical display; and escape from the main window in the form of either a link to the theory window or an exit button. We will describe each of these components within the context of their contributions to the design, data collection, data analysis, and theory application entailed in the filter experiment.

Manipulation of Experimental Parameters From Within the Graphic User Interface

Like the filter experiment that we use in our example, many experimental projects, particularly those involving sensation and perception, involve a series of interrelated data sets. For instance, a project to investigate the effects of contrast on information acquisition may involve a series of experiments, each with varying values of duration and contrast and, usually, involving choices of other experimental parameters as well. Often, such experiments act as pilot experiments, designed to map out the lay of the perceptual land for individual observers before the main experiment is designed and the final data collected.

As the number of such data sets within a project grows, it becomes increasingly difficult to change the experimental parameters in a systematic and/or optimal fashion, because such parameters are often embedded in the code of the relevant programs. Keeping track of which data sets emerged from which observers also becomes tedious as the number of data sets per observer increases. A GUI is a useful tool for dealing with these problems, since it can do double duty as a mini-database. Through its use, an experimenter can easily set, alter, and keep track of experimental parameters for multiple observers within a single visual display. The design features used for these purposes that are implemented in our example GUI (see Figure 1) are the following.

Observer information. The area in the upper left section of our GUI's main window is devoted to observer information and setup of data files. Here, all of the observers' initials are listed in a pop-up menu. Once a particular observer has been selected, information for that observer is loaded, and the GUI is updated—that is, parameter values are displayed, and any existing data are graphed in the GUI's graphical display. If, as is usually true, the observer under consideration has more than one data set, a specific data set can be called up with the "Selected Data Set" pop-up menu, and again the GUI will be updated. In Figure 1, observer E.U. is selected; she has five existing data sets, of which Data Set 2 has been selected for analysis. The various parameter values for this data set are shown in the edit boxes along the left side of the GUI. This allows the experimenter to immediately see what parameter values characterized Data Set 2 for observer E.U.[4]

Setup. If the experimenter wishes to establish a new data set for an observer, he or she does so by setting all of the relevant experimental parameters to their desired values and simply clicking on the push button labeled "Setup." This button calls a routine that creates a new data file for the observer with the current parameter settings. An important clarification must be made here about the dual function of the GUI's editable parameter list. As was just described, one function of the list is for the creation of new data sets. The second function of the list is to indicate clearly to the experimenter what parameter values were used in any existing data set. For example, when

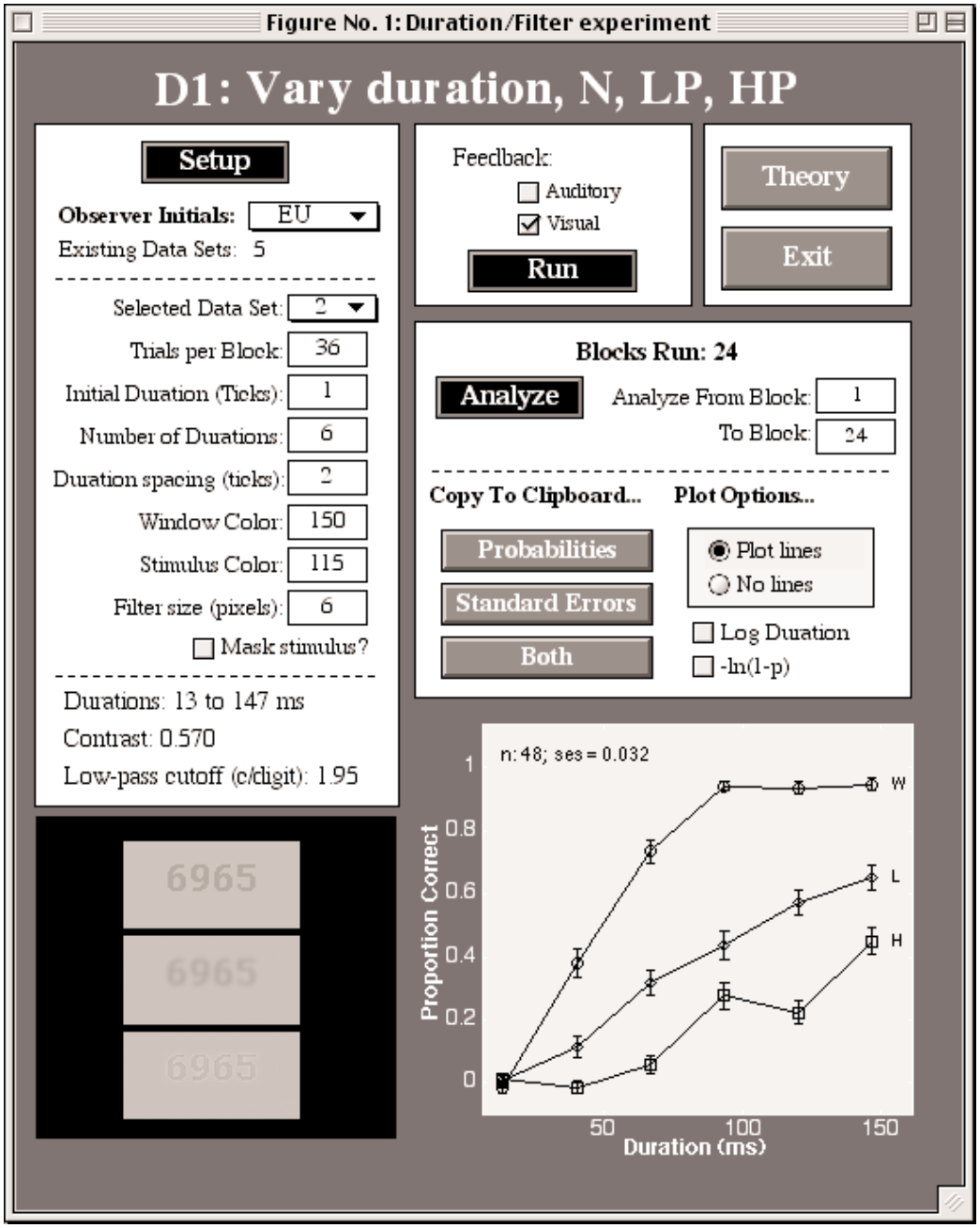

Figure 1. Main window of example graphic user interface (GUI) created to accompany an experiment (the *filter experiment*) designed to investigate the effects of spatial frequency filtering on digit recall. *Note*: This is a black and white reproduction of a multicolored GUI.

a data set is selected for analysis on the GUI, the parameter values corresponding to that data set are automatically loaded and displayed in the editable boxes.

The various pieces of information below the parameter list are updated when parameter values are altered. For example, if the editable text corresponding to (i.e., just to the right of) "Number of Durations" were to be changed from 6 to 8, the static text "Durations: 13 to 147" would change to read "Durations: 13 to 200." Note that although static text cannot be edited on an active GUI, it can be changed in real time by the underlying programming code to reflect changes made to relevant editable items on the GUI.

Stimulus preview. The black box in the bottom left corner of the window shows a preview of what the stimuli will look like during the experiment (showing, from top to bottom, N, LSF, and HSF stimuli). Like the static text below the parameter list, the preview stimuli also

change to reflect the values set in the parameter list. This allows the experimenter to see how manipulating variables, such as stimulus color (gray-scale value ranging from 1 to 256) or filter size, will affect the appearance of the stimuli, without entailing actual execution of the experiment.

Execution of the Experiment from Within the Graphic User Interface

In visual perception experiments, it is common to have few observers but a great deal of data per observer. With the aid of a GUI, observers can easily collect their data without the presence of the experimenter.

Data collection. In the experiments conducted in our laboratory, observers are instructed to open MATLAB and to type the name of the experiment in the Command Window, which calls up and executes the relevant GUI. Once the GUI is open, the observers select their initials from the "Observer Initials" pop-up menu, select the appropriate data set from the "Selected Data Set" pop-up menu, and click on the button entitled "Run." While the experiment is running, the MATLAB code automatically saves data to the observer's data file on completion of each block of trials, thereby eliminating the necessity for the observers to be responsible for saving their own data to a file at the end of a data collection session. At the end of a session, observers are returned to the GUI, where they can view their data in various ways (see the Data Analysis section) or simply quit and leave.

Feedback options and experiment execution. The upper middle section of the GUI's main window provides feedback options, as well as the experiment execution button, entitled "Run." Check boxes are utilized here to provide two feedback options: visual and auditory. These boxes are not mutually exclusive; they can be independently selected and deselected by the user. The effect of clicking on the "Run" button is to execute the filter experiment by calling up MATLAB routines that perform the following operations. First, the requested data set is loaded, and the saved parameter values are evaluated. Then a stimulus-creation routine creates and stores the requisite number of stimuli needed for the first block of experimental trials (i.e., random four-digit strings are generated and filtered). Counterbalancing for the first block of trials is carried out, and finally, the computer screen is cleared, and the observer is prompted to initiate the first trial with a keypress. At the end of each block, the observer is prompted with the option to either continue with a new block of trials or quit and return to the GUI.

Data Analysis

The final section of the GUI's main window, the bottom right, is devoted to data analysis. The static text, "Blocks Run: 24," indicates that the selected observer, E.U., has collected 24 blocks of data for Data Set 2. Edit text boxes allow the experimenter to specify analysis of either the entire set of blocks (the default) or any valid subset of blocks. Data (probabilities, standard errors, or both) can be copied to the clipboard by clicking on the "Copy to Clipboard" buttons. This is particularly useful for transferring data to a spreadsheet application, such as Excel, or to a graphing application, such as Kaleidagraph.

Plot options. Four plot options are provided on the GUI. The first two are displayed as mutually exclusive radio buttons: "Plot lines" and "No lines." These allow the user to choose whether or not to display the lines on the graph or remove them, displaying only the data points and standard error bars. Below the radio buttons are two plot options check boxes. The first, when checked, plots the data on a log duration scale, whereas the second changes the data from probability (p) to a transform of probability, $[-\ln(1-p)]$ (see Loftus, Busey, & Senders, 1993). When any one of these radio or check boxes is selected or deselected, the GUI immediately updates the graph to reflect the change.

Graphical display. The lower right corner of the main window contains a simple graph of the data that automatically updates itself when new observers and/or data sets are selected for analysis. MATLAB provides a multitude of plot options for use in graphing, including various colors, symbols, line styles, axes labels, legends, and so forth. Graphs in MATLAB can be as simple or as complex as the user desires.

Theory Fitting

In our example GUI, we have created a second GUI window, the *theory window*, shown in Figure 2, for all theory-related operations.

Link to theory window and exit button. In the GUI's main window, the user is provided two options for leaving the window: a link to the corresponding theory window, or an exit out of the GUI altogether. The two buttons that perform these tasks are located within a small frame in the upper right portion of the main window. While the "Exit" button simply closes the GUI, the "Theory" button opens the theory window (described below), while leaving open the main window of the GUI. This allows the user to click between the two windows, altering observer and/or data set selection for theory fits.

Theory window. Figure 2 shows the theory window created to accompany the main window of the filter experiment GUI. The code called by various objects within the window utilizes MATLAB's optimization tools to fit a specific quantitative theory to the filter experiment data. For interested readers, the perceptual theory used to fit data from the filter experiment is described in numerous publications (e.g., Busey & Loftus, 1994; Loftus et al., 1993; Loftus & Ruthruff, 1994). We will refer to it as the sensory response/information-acquisition rate (SRIAR) theory.

Although knowledge of the SRIAR theory is not critical for understanding the essence of this article, we provide a very brief synopsis of the parameters, to aid in understanding the GUI's theory window. The SRIAR theory has four parameters. First, n (a unitless positive integer) and τ (a positive real number with units of time) are the

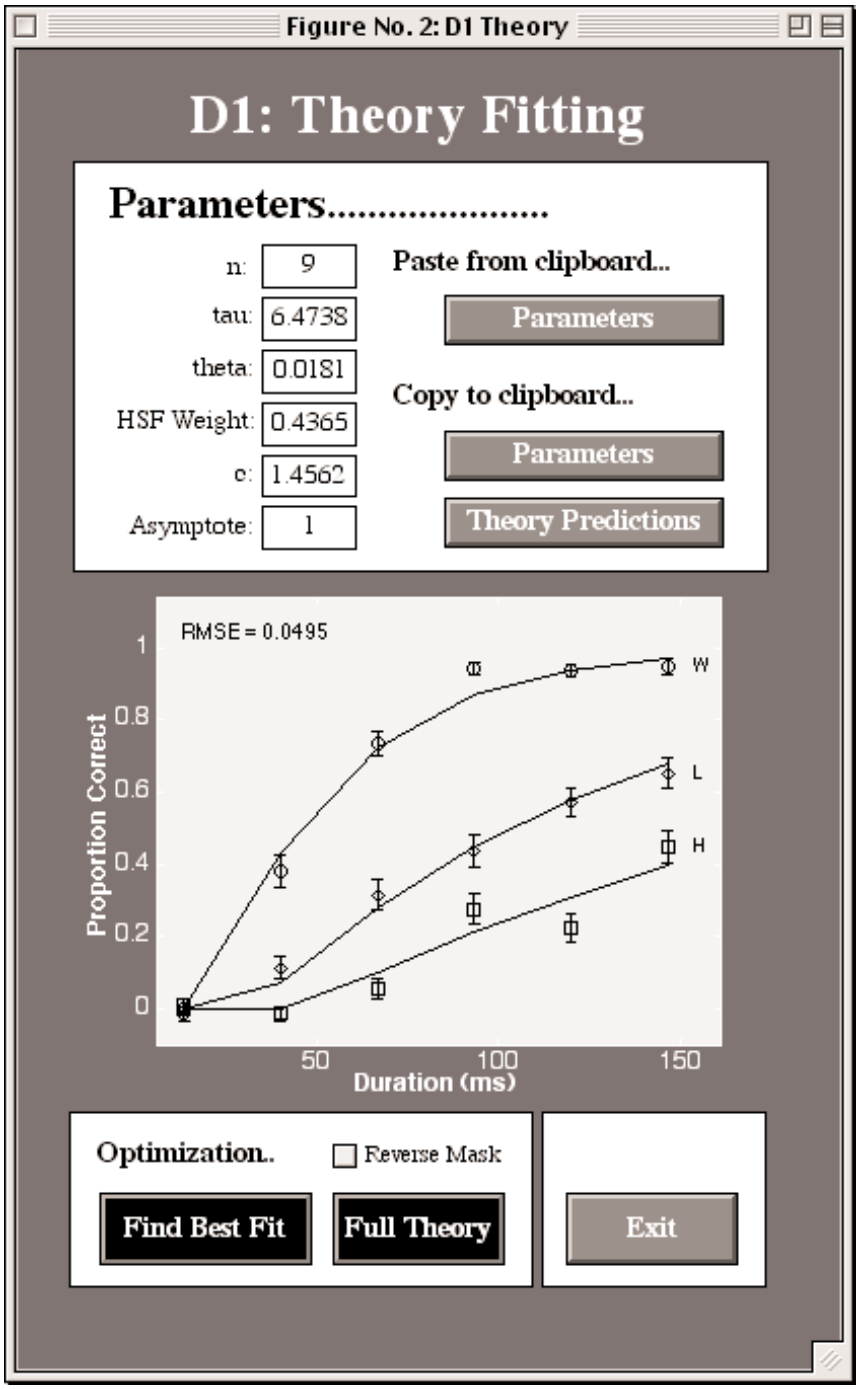

Figure 2. Theory window of the filter experiment graphic user interface (GUI). This window is called by the "Theory" push button on the main window of the GUI and utilizes MATLAB's optimization routines to fit Loftus and Busey's sensory response/information acquisition rate theory to the filter experiment data. Note: This is a black and white reproduction of a multicolored GUI.

parameters of a gamma function that allow generation of an internal *sensory response* to a stimulus of a given temporal shape. There is a *sensory threshold*, θ, such that further information processing takes place only when the value of the sensory response is greater than θ (θ is a unitless number intimately connected to stimulus contrast). A parameter c (with units of time^{-1}) controls the rate at which information is acquired from a given stimulus. For application in the present experiment to the issues of spatial filtering, two additional parameters are required.

A unitless parameter is added that reflects the relative weight of assumed high- versus low-spatial-frequency channels, and finally, because observers appear to be imperfect in several respects, a performance asymptote, *A*, is required.

The resulting six free parameter values for the SRIAR theory are listed in edit text boxes along the top left side of the theory window. Push buttons allow the user the opportunity to paste parameters stored on the clipboard, as well as to copy parameters and theory predictions onto the clipboard. To find the best-fitting theory parameters for a set of data points, the experimenter clicks on the push button entitled "Find Best Fit." This button calls MATLAB optimization routines that search for the values of the free parameters that minimize error between the data and the theory. Once found, the new parameters are copied into the parameter edit text boxes. Each time the parameters are changed, either manually or via the optimization routines, the theory predictions are graphed in the axes as solid lines overlaying the data points. The action initiated by the "Full Theory" button is to smooth out the theory curves by calculating and plotting the theory predictions over a large number of exposure durations, rather than only at the six experimentally tested durations. Finally, the "Reverse Mask" checkbox was created to allow us to see whether the SRIAR theory could fit the data if it was misled about whether or not the stimuli in a particular data set were masked (incidentally, we found that it could not).

SUMMARY AND CONCLUSIONS

MATLAB is a convenient platform for the development and management of psychological experiments, owing to its easy-to-use programming language, sophisticated graphics features, and statistics and optimization tools. Through the additional implementation of the Brainard–Pelli Psychophysics Toolbox, MATLAB's interpreted programming language gains the power and flexibility of a low-level language, giving the user close temporal and spatial control over the CRT display.

MATLAB's benefits to psychological experimentation can be further increased through the use of one graphical feature in particular, the GUI. A well-designed GUI is a powerful tool for organizing and controlling all aspects of running a psychological experiment, including design, data collection, data analysis, and theory fitting.

REFERENCES

Brainard, D. H. (1997). The Psychophysics Toolbox. *Spatial Vision*, **10**, 433-436.

Busey, T. A., & Loftus, G. R. (1994). Sensory and cognitive components of visual information acquisition. *Psychological Review*, **101**, 446-469.

Cohen J., MacWhinney B., Flatt, M., & Provost, J. (1993). PsyScope: An interactive graphic system for designing and controlling experiments in the psychology laboratory using Macintosh computers. *Behavior Research Methods, Instruments, & Computers*, **25**, 257-271.

Hanselman, D., & Littlefield, B. (1996). *Mastering MATLAB: A comprehensive tutorial and reference*. Englewood Cliffs, NJ: Prentice-Hall.

Loftus, G. R., Busey, T. A., & Senders, J. W. (1993). Providing a sensory basis for models of visual information acquisition. *Perception & Psychophysics*, **54**, 535-554.

Loftus, G. R., & Ruthruff, E. R. (1994). A theory of visual information acquisition and visual memory with special application to intensity–duration tradeoffs. *Journal of Experimental Psychology: Human Perception & Performance*, **20**, 33-50.

Marchand, P. (1999). *Graphics and GUIs with MATLAB* (2nd ed.). Boca Raton, FL: CRC Press.

Olds, E. S., & Engel, S. A. (1998). Linearity across spatial frequency in object recognition. *Vision Research*, **38**, 2109-2118.

Parish, D. H., & Sperling, G. (1991). Object spatial frequencies, retinal spatial frequencies, noise, and the efficiency of letter discrimination. *Vision Research*, **31**, 1399-1415.

Pelli, D. G. (1997). The VideoToolbox software for visual psychophysics: Transforming numbers into movies. *Spatial Vision*, **10**, 437-442.

St. James, J. D., & Schneider, W. (1991). Student MEL software support for instructors and teaching assistants in research methods course. *Behavior Research Methods, Instruments, & Computers*, **23**, 149-154.

Schneider, W. (1989). Enhancing a standard experimental delivery system (MEL) for advanced psychological experimentation. *Behavior Research Methods, Instruments, & Computers*, **21**, 240-244.

Yee, P. L., & Vaughan J. (1999). A Web-accessible tutorial for PsyScope based on classic experiments in human cognition. *Behavior Research Methods, Instruments, & Computers*, **31**, 107-112.

NOTES

1. The Math Works announced in 1998 that it would no longer support the Macintosh past version 5.2. It is hoped that The Math Works will reconsider this decision in view of the Macintosh's resurgence.

2. Costwise, MATLAB is an excellent alternative to other experiment creation software packages. The student version of MATLAB (which is a full version, not scaled down) currently sells for $99, with optional toolboxes, such as the Statistics and Optimization toolboxes, costing an additional $59 each. The academic cost of MATLAB for nonstudents is $500 with optional toolboxes costing an additional $200 each.

3. To download or to obtain further information about the free Psychophysics Toolbox available for both Macintosh and Windows, go to http://color.psych.ucsb.edu/psychtoolbox/

4. For use in this paper, the stimulus color listed for observer E.U. in Figure 1 was altered from its actual value of 146 to a lower value of 115. This was done to increase the contrast of the stimuli pictured in the bottom left corner of the GUI and, hence, improve the visibility of these stimuli for the reader.

(Manuscript received November 1, 1999;
revision accepted for publication February 25, 2000.)