Spatial Vision, Vol. 10, No. 4, pp. 447–466 (1997)

Psychophysica: Mathematica notebooks for psychophysical experiments*

(Cinematica—Psychometrica—Quest)

ANDREW B. WATSON[1,**] and JOSHUA A. SOLOMON[2]

[1]*NASA Ames Research Center, Moffett Field, CA 94035-1000, USA*
[2]*Institute of Ophthalmology, Bath Street, London EC1V 9EL, UK*

Received 24 April 1996; revised 7 October 1996; accepted 7 October 1996

Abstract—*Psychophysica* is a set of software tools for psychophysical reasearch. Functions are provided for calibrated visual displays, for fitting and plotting of psychometric functions, and for the QUEST adaptive staircase procedure. The functions are written in the Mathematica programming language.

1. INTRODUCTION

In vision research we must create much of our own experimental apparatus. Increasingly this consists of a computer, a display, and associated software. The generality of this apparatus is a blessing and a curse: it offers great power and flexibility, but exacts a terrible cost in programming effort. This cost is particularly high because of two factors. The first is that, until recently, high-level languages capable of expressing the full variety of experimental design did not exist, consequently low-level languages were employed (Watson *et al.*, 1986). These languages are slow to program and difficult to maintain, and are also nearly impossible to read and hence to share. This leads to the second painful cost multiplier: the duplication of software products. Each laboratory, indeed each experimenter, is obliged to write their own code because they cannot read anyone else's. An additional impetus for private code is that most experimental code is very specific, due in part to the effort involved in creating and maintaining general code in a low-level language.

Recently several high-level languages have appeared that may offer some salvation from this purgatory. Code is developed much more rapidly, and maintenance is simpler. Functions are typically short and relatively legible, and thus encourage sharing.

---

*http://vision.arc.nasa.gov/mathematica/psychophysica.html

**beau@vision.arc.nasa.gov

Here we describe three integrated software packages that we have developed to support vision research. Each is written in the language Mathematica (Wolfram, 1991), which is available for a wide variety of common computer systems. The three packages are *Psychometrica*, which contains functions for fitting and plotting psychometric data, *Quest*, which implements the QUEST adaptive psychometric procedure, and *Cinematica*, for producing calibrated grayscale displays. The first two packages will run on any computer for which Mathematica is available (most common workstations and PCs); the third package depends upon special display routines and runs only on the Apple Macintosh. The three packages are largely independent, but are designed to work together. The three packages are contained in the Mathematica notebooks *Psychometrica.ma*, *Quest.ma*, and *Cinematica.ma* which are available at the web site given on p. 447.

The following sections consist primarily of demonstrations of the use of Mathematica functions. For this purpose we adopt the following conventions: Mathematica input and output are printed in Courier font, with input in boldface. Documentation of individual functions, drawn directly from the Mathematica notebooks, is given in text boxes.

2. CINEMATICA

Cinematica is a software package for producing calibrated grayscale displays on an Apple Macintosh computer from within Mathematica. Cinematica uses the low-level VideoToolbox subroutines developed by Denis Pelli (Pelli and Zhang, 1991; Pelli, 1997) but provides a simple high-level interface that obviates the need for low-level display programming in many experimental situations. A more extensive description of *Cinematica* is given elsewhere (Solomon and Watson, 1996).

As an example of the use of *Cinematica*, to display a 64 × 64 pixel cosine grating with frequency 4 cycles/image at a contrast of 0.5, we would use the following commands. The first function initializes the *Cinematica* package and is typically used once at the start of a session.

```
CinematicaOpen[ ];
```

Then we create the stimulus using standard Mathematica functions.

```
image = Round[128+ 126 Table[ Cos[2 N[Pi] 4 x/64], {64}, {x, 64}]];
```

We can preview the image (Fig. 1) with the following *Mathematica* command.

```
ListDensityPlot[image, FrameTicks->None, Mesh->False];
```

Then we move the image to a portion of the Macintosh memory called the GWorld. From the user's point of view it is just temporary storage for stimuli. The image is assigned an index 1.

```
MovieToGWorld[1, image];
```

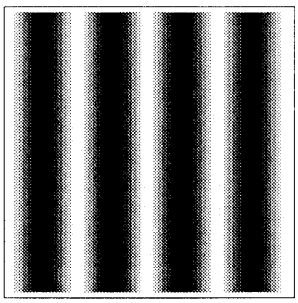

Figure 1. A cosine grating image.

Then we create a color map that will yield 0.5 contrast, and assign it the index 1.

```
ColorMap[1, 0.5];
```

The stimulus is displayed by the following function, which takes a display list consisting of {image index, color map index} pairs for each video frame.

```
GWorldToScreen[{{1, 1}}];
```

Finally at the end of a session one may exit the *Cinematica* package with the following function.

```
CinematicaClose[ ];
```

The *Cinematica* system can display movies defined by any pairing of contrasts and images. This includes such specialized situations as a single image with a particular contrast, a single image with a time-varying contrast, a sequence of images whose individual contrasts are fixed, a sequence of images whose individual contrasts are varied. The lengths of the movies are limited only by the host memory. *Cinematica* contains options for selecting an arbitrary display gamma exponent, and for setting individual graylevels to particular luminances. Limitations of the current system are (1) it displays only monochrome images; and (2) all displayed images are centered on the screen. We conclude this section with a summary of *Cinematica* functions.

CinematicaOpen[ ]: Must be called prior to any other *Cinematica* Functions.

MovieToGWorld[imageIndex, movie]: Load a grayscale image or movie into GWorld(s). *movie* is a rectangular array of numbers (a single image), or a list of equal-sized images. Pixels within each image must be integers between 0 and 255. Each frame of the movie is assigned its own GWorld. The index of the nth frame's GWorld is $imageIndex + n - 1$.

ColorMap[cMapIndex, contrast, grayMin, grayMax, cMapList, gamma]: Calculate a lookup table with index *cMapIndex.contrast* must be a real number between −1. and 1., corresponding to the desired proportion of obtainable contrast (when the minimum luminance is close to zero, this will be equal to luminance contrast). *grayMin* and *grayMax* define the range of image graylevels that will be smoothly mapped to the range of luminances spanned by the minimum and maximum obtainable luminances. *gamma* is the desired exponent of the mapping between graylevels and luminance; if a linear relation is desired, this should be the default value of 1. Any leftover color map entries can be assigned a luminance manually by setting *cMapList*={{gray_1, luminance1}, {gray_2, luminance2}, ...}, where *luminance* is a real number between 0. and 1. on a linear scale of the available luminances. For technical reasons, *cMapList* cannot be empty. However, manual luminance assignments to graylevels between grayMin and grayMax are ignored. Default values are contrast = 1, grayMin = 0, grayMax = 255, cMapList = {{0, .5}}, gamma = 1.

GWorldToScreen[displayList]: Display a movie. *displayList* should be a list of pairs, {{imageIndex1, cMapIndex1}, {imageIndex2, cMapIndex2}, ...}, one pair for each frame of the display. Note that the writing of an image to the screen and the loading of a colormap cannot be truly simultaneous. Cinematica loads the colormap first. To prevent the brief display of an old image with a new colormap, the user may wish to begin displayList with a blank image.

CinematicaClose[ ]: Should be called prior to termination of any Mathematica session involving any *Cinematica* Functions.

FileToGWorld[file, imageIndex, cols, rows, frames]: Load a raw, byte-format, grayscale movie from a file to GWorlds. Each frame of the movie has its own GWorld. The index of the first frame's GWorld is *imageIndex*, the index of the nth frame's GWorld imageIndex + 1.

3. PSYCHOMETRICA

A psychometric function describes the probability of a positive response (Yes, or correct) as a function of some measure of stimulus strength. Fitting a psychometric function to data is a common requirement in vision research. The notebook *Psychometrica.ma* contains functions for fitting and plotting psychometric data as well as definitions of Weibull, Logistic, Normal, and Uniform psychometric functions.

3.1. Psychometric functions

The following are definitions for a number of commonly used psychometric functions. Psychometric functions are typically cumulative probability density functions, amended by guessing and lapsing factors. All the functions defined here follow

a standard form $f[x, p]$ where p is a list of parameters. The parameters for the three examples here are always {threshold, slope, guess, lapse}, although their precise meanings depend somewhat on the particular function. Their general meanings are: threshold—a strength at which a critical probability is reached; slope—a measure of the rate of rise of probability with respect to strength; guess—the probability of a correct response at strength $-\infty$; lapse—the probability of an incorrect response at strength ∞. This standard form allows the functions to be used by fitting routines and by the QUEST procedure (see Section 4). User-written functions should follow the same convention to obtain these capabilities. We provide an example of writing a new function in Section 3.2. By convention, each psychometric function name terminates in the letters 'PF'.

3.1.1. LogWeibullPF. The Weibull function has proven to be a particularly useful description of the human psychometric function for visual detection of threshold stimuli (Quick, 1974; Watson, 1979; Nachmias, 1981; Pelli, 1986). It is based on the Weibull probability distribution (Weibull, 1951). Human contrast detection psychometric functions are typically parallel on a log abscissae. Thus stimulus strength can be conveniently expressed in decibels. The distribution of the log of a Weibull random variable is given by the Log-Weibull distribution (also known as the Extreme Value Distribution, the Gumbel distribution, or the Fisher–Tippet distribution). We therefore define LogWeibullPF as the psychometric function based on the cumulative Log-Weibull distribution.

LogWeibullPF[x, {threshold, slope, guess, lapse}]: The Log-Weibull psychometric function. The stimulus strength is x; *threshold*, *slope*, *guess*, and *lapse* are parameters. The function returns a probability of success.

To illustrate both the function (Fig. 2) and how it may be used we evaluate the following:

```
Plot[LogWeibullPF[x,{-6.,  4.,0,.01}],{x,-20,0}
,PlotRange->{0,1}, Frame->True
,FrameLabel->{"Strength","Probability"}];
```

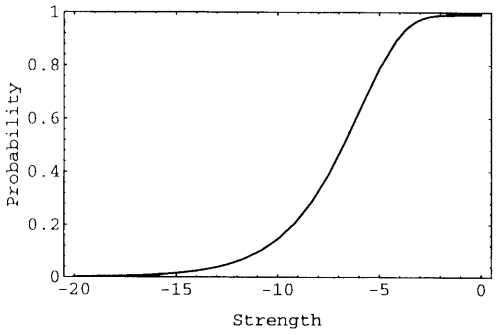

Figure 2. The LogWeibullPF psychometric function.

3.1.2. LogisticPF . The logistic is another widely used psychometric function.

LogisticPF[x, {threshold, slope, guess, lapse}]: The LogisticPF psychometric function. The stimulus strength is x, the parameters are: *threshold*, *slope*, *guess*, and *lapse*. The function returns a probability of success.

3.1.3. NormalPF. The cumulative Normal distribution is yet another widely used psychometric function.

NormalPF[x, {threshold, slope, guess, lapse}]: The NormalPF psychometric function is based on a Normal cumulative distribution with mean *threshold* and standard deviation 1/*slope*. The stimulus strength is x, the guessing rate is *guess*, and the lapsing rate is *lapse*. The function returns a probability of success.

3.1.4. UniformPF. Though not widely used, a uniform psychometric function is useful in some applications.

UniformPF[x, {threshold, slope, guess, lapse}]: The UniformPF psychometric function is based on the cumulative uniform distribution of width 1/*slope*, centered at threshold, ammended by guessing and lapsing parameters guess and lapse. It represents a linear increase in probability from a lower asymptote of *guess* to an upper asymptote of 1-*lapse*, with a rate of increase of $(1 - guess - lapse) * slope$. At *threshold*, the probability is $(1 + guess - lapse)/2$.

3.2. Fitting and plotting psychometric data

Psychometric data typically consist of numbers of correct and incorrect responses at a number of signal strengths. Here are some example data, expressed as triples of the form {strength, # wrong, # right}. We use this raw form, rather than percent correct at each strength, since the latter measure discards useful information regarding number of trials.

```
fakedata = {{2,4,3},{4,3,4},{6,10,34},{8,1,18},{10,0,10}};
```

We fit the data using the function PsychometricFit. The fit is based upon a maximum likelihood method (Watson, 1979).

```
fit = PsychometricFit[fakedata]
```

```
{{{2,4,3},{4,3,4},{6,10,34},{8,1,18},{10,0,10}},
LogWeibullPF, {{6.37345,4.,0.5,0.01},0.489602}}
```

The output structure includes the input data, the name of the psychometric function that is fit to the data, and a list containing the list of psychometric function parameters, followed by the residual error.

By default, PsychometricFit estimates only the first parameter of the psychometric function (threshold), and uses a LogWeibullPF psychometric function. Other parameters may be estimated and other psychometric functions may be selected using the various options to PsychometricFit.

PsychometricFit[data]: Fits psychometric data with a psychometric function. The function returns the input data, the psychometric function name, and the specified and estimated parameters. The fit is based upon a maximum likelihood method, as described in Watson (1979). *data* is a list of triples each of the form {strength, no, yes}, where *strength* is a measure of signal strength, *no* is the number of no or incorrect responses, and *yes* is the number of yes or correct responses. The function also accepts a list of such tables, and produces one fit per table. The function also accepts the data structure produced by QuestExperiment from the package Quest.ma, from which it generates the requisite tables. The option PsychometricFunction (default = LogWeibullPF) allows selection of an alternate psychometric function. The psychometric function must adhere to a standard format: function[x_, parameters_List]. The option PsychometricParameters (default = {Automatic, 4., 0.5, .01}) allows selection of starting points or final values of psychometric function parameters. The value Automatic in the first position indicates that the starting point will be estimated from the data (the largest strength in the input data). Another option is FitParameters (default = {True}) which indicates whether each parameter should be estimated. Unspecified values default to False. For example, to estimate only the first and third parameters, use FitParameters->{True, False, True}. By convention, the first parameter is threshold and the second parameter is slope. This convention is relevant only in functions such as PsychometricPlot, which accepts directly the output of PsychometricFit. A final option is FitLimits, which specifies the bounds on each parameter (default = {{−$MaxMachineNumber, $MaxMachineNumber}, {0, $MaxMachineNumber}, {0, 1}, {0, 1}}.

The output of PsychometricFit can be plotted directly (Fig. 3) using the function PsychometricPlot.

```
PsychometricPlot[fit];
```

By default, PsychometricPlot shows the proportion correct (points), the fitted function (curve), binomial 95% confidence limits (gray vertical lines), the parameters, and the residual error of the fit. The latter three can be suppressed, and options for ListPlot may also be included to vary general properties of the plot such as the font.

PsychometricPlot[fitdata]: Plots a set of psychometric data and a psychometric function, as returned by PsychometricFit. See that function for details. Also draws approximate 95% confidence intervals around each data point, unless the option PsychErrorBars->False is given. To suppress printing of parameters and error of the fit, use the option PsychLegend->False. Options to ListPlot may also be included.

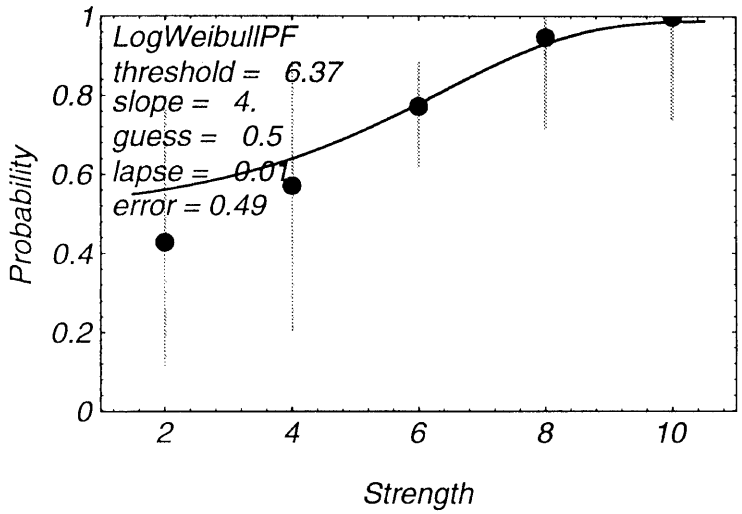

Figure 3. Psychometric data and a fitted psychometric function, produced by the function PsychometricPlot.

3.3. Creating a new Psychometric Function

As noted in the introduction, one virtue of high-level languages is the ease with which new functions are developed, and coordinated with existing functions. To show how the user may create a new type of psychometric function, we define a psychometric function based on the Cauchy distribution, which has infinite variance, and thus reminds us of certain observers we have known. The functions CDF (cumulative density function) and CauchyDistribution are part of Mathematica.

```
CauchyPF[x_, {threshold_, slope_, guess_, lapse_}] :=
  guess + (1 - guess - lapse) *
    CDF[CauchyDistribution[threshold, 1/slope], x]
```

To fit the data with this function, we use the option PsychometricFunction (Fig. 4). We also use the option PsychometricParameters to specify a reasonable starting point for the slope parameter.

```
PsychometricPlot[PsychometricFit[data1
  ,PsychometricFunction->CauchyPF
  ,FitParameters->{True, True}
  ,PsychometricParameters->{Automatic, 1, 0.5, 0.01}]];
```

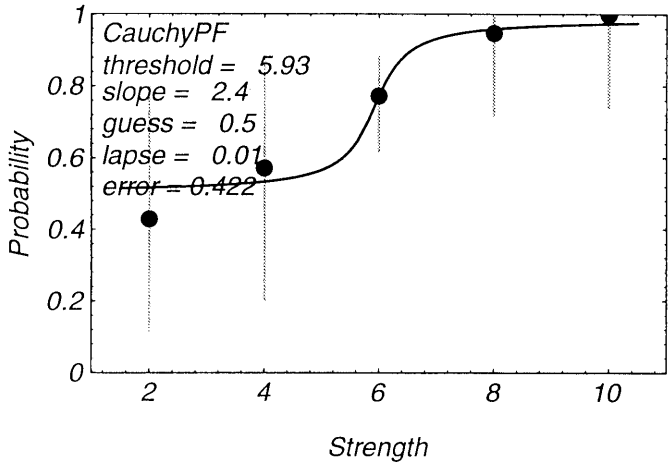

Figure 4. Fitting data with the CauchyPF psychometric function.

4. QUEST

QUEST is an adaptive psychometric procedure for use in psychophysical experiments (Watson and Pelli, 1983; Watson and Fitzhugh, 1990; King-Smith *et al.*, 1994; Treutwein, 1995). In its original form, QUEST placed each trial at the current mode of the posterior probability density for threshold. In later variants, the mean or median of the density have been advocated as the optimal testing point (Emerson, 1986; King-Smith *et al.*, 1994; Treutwein, 1995).

The Quest package consists of three 'layers' of software. At the lowest layer are a set of basic functions from which one may construct a wide variety of specific experimental applications (see Section 4.3). At the next layer is a single function, QuestExperiment, built from the lower level functions, which measures thresholds for a set of abstract stimuli. This function is independent of the particular stimuli or presentation technology, it merely manipulates the abstract quantities *condition* and *strength*. The condition is an integer index (starting at 1) that identifies a particular stimulus. The strength is a real number specifying the stimulus dimension that is varied in the experiment. At the highest layer are functions which control specific psychophysical experiments. We provide one such function, ContrastThreshold, which measures contrast thresholds for arbitrary grayscale image sequences using the Cinematica display software.

4.1. QuestExperiment

We begin with an illustration of QuestExperiment. By default, QuestExperiment will assume a simulated observer. The psychometric function and function parameters used by the simulated observer can be set in the following way. We first run the example with one condition whose true threshold is -20.

```
SetOptions[SimulatedObserver,
  PsychometricFunction->{LogWeibullPF},
  PsychometricParameters->{{-20.,3.5,.5,.01}}];
```

Our guess at threshold is −30. We request 32 trials. We request a plot after each trial. We set a random seed so that this exact sequence can be repeated. We show only three of the 32 plots that are produced by this command (Fig. 5).

```
simdata = QuestExperiment[{-30},32,QuestPlotShow->True
  ,QuestSeed->99];
```

The option QuestPlotShow->True causes QuestExperiment to call the function QuestPlot after each trial. QuestPlot produces a graph that illustrates various aspects of the sequence of trials generated by a Quest staircase. The first two elements of the plot are the ordered sequence of trials, showing strength and response (red = wrong, green = right), a histogram showing the distribution of trials at various strengths (blue bars). Quest operates by continuously updating a Bayesian posterior probability density of the threshold estimate. This density, normalized to one, is shown as a solid gray region. By default, QuestPlot includes the entire range of strengths used by QuestExperiment, which in turn defaults to [−60, 0].

QuestPlot[data, condition]: Plots several summaries of the progress of a QUEST session for one *condition*. QuestPlot[data] plots separate summaries for all conditions. Included on the same figure are 1) the posterior probability density, scaled to have a unitary maximum, 2) the histogram of trial placements, 3) the sequence of trial placements and their individual outcomes (green for yes/correct, red for no/incorrect), 4) the best-fitting psychometric function, and 5) the printed estimates of threshold, slope, guess, lapse, and error. Options to ListPlot and PsychometricPlot can also be included. The argument *data* is the basic data structure maintained by QUEST (see QuestUpdate) and returned by QuestExperiment. The various components of the plot may be toggled by means of the following options, shown with their default values: Fit->False, Density->True, Histogram->True, Trials->True.

Typically, data collection using QUEST is followed by fitting the data with a psychometric function. To do this we use the function PsychometricFit discussed in Section 3.2.

```
fit = PsychometricFit[simdata]
{{{{-30,1,0},{-26,1,0},{-24,0,1},{-22,1,1},{-20,0,6},{-18,2,14},
{-16,0,2},{-14,0,2},{-12,0,1}}, LogWeibullPF, {{-19.4431,4.,
0.5,0.01},3.95957}}}
```

The result can be plotted with PsychometricPlot (Fig. 6). Note that by default this plot only includes strengths at which trials were conducted.

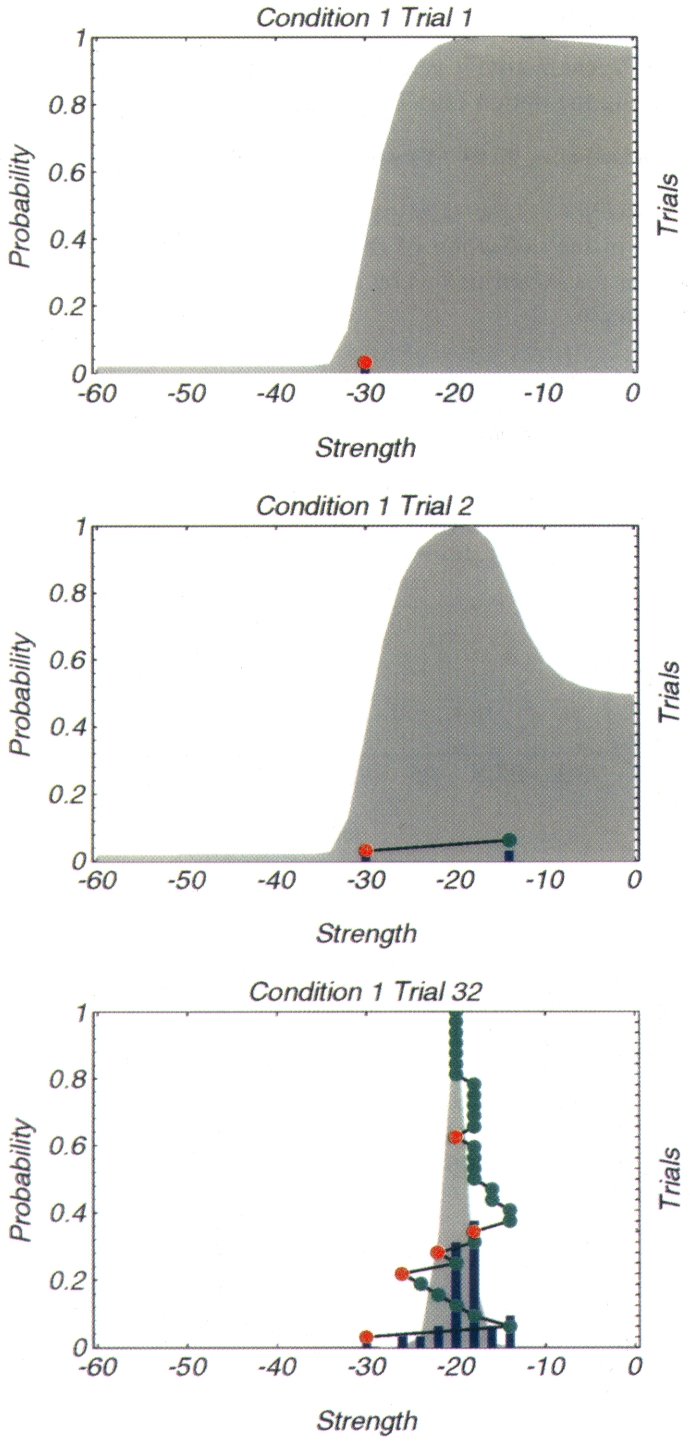

Figure 5. First, second, and last trials from a sequence of 32 trials of QuestExperiment.

```
PsychometricPlot[fit];
```

The plot from PsychometricFit and PsychometricPlot can be embedded within the QuestPlot by setting the option Fit->True (Fig. 7).

```
QuestPlot[simdata, Fit->True];
```

By default, QuestPlot fits the same psychometric function and parameters used by QuestExperiment in the collection of the data, and preserved in the resulting data structure. The user may override this by specifying options to PsychometricFit, as in the following example.

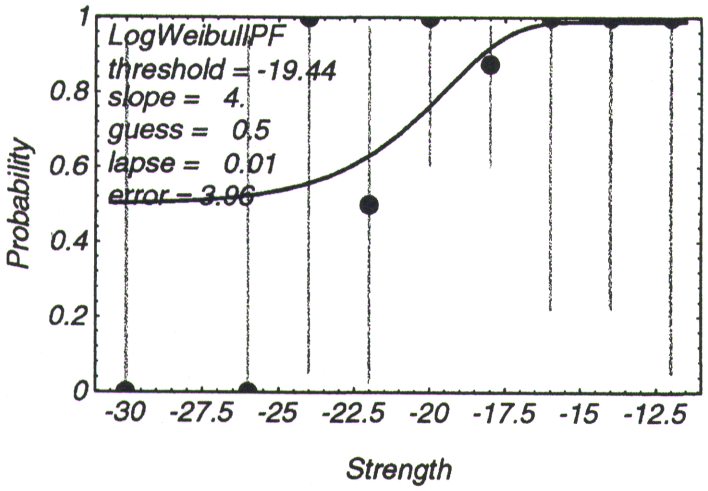

Figure 6. Plot produced by PsychometricPlot of data and psychometric function.

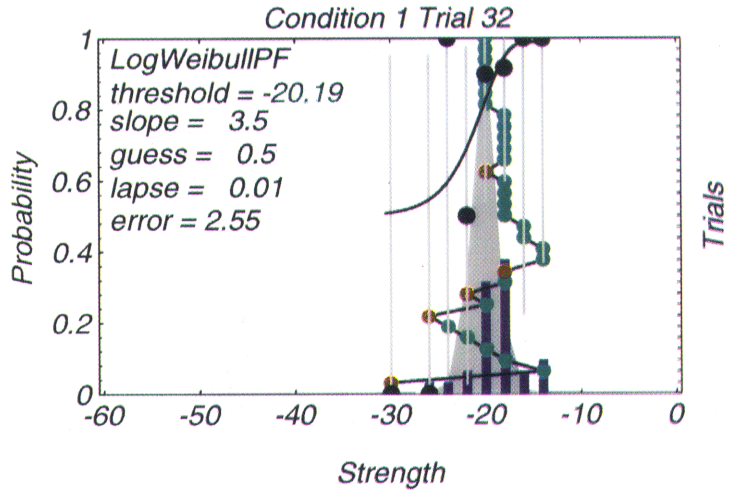

Figure 7. Plot produced by QuestPlot that includes a fitted psychometric function.

```
QuestPlot[simdata, Fit->True
,FitParameters->{True,True}
,PsychometricFunction->LogisticPF
,PsychometricParameters->{Automatic,.2,.5,.01}];
```

The previous examples employed only a single condition. In the next example, we show how to use 2 conditions. We first set the simulated thresholds to −20 and −30.

```
SetOptions[SimulatedObserver,
  PsychometricFunction->{LogWeibullPF,LogWeibullPF},
  PsychometricParameters->{{-20.,4,.5,.01},{-30.,4,.5,.01}}];
  simdata2 = QuestExperiment[{-10,-40},32,QuestSeed->101];
```

We could now use QuestPlot to plot each condition separately (for brevity, all plotting has been suppressed or deleted).

```
QuestPlot[simdata2, Fit->True];
```

And again we could use PsychometricFit to estimate the slope parameters.

```
PsychometricPlot[
  PsychometricFit[simdata2,FitParameters->{True, True}]];
```

4.1.1. Changing QuestMethod. A number of authors have argued that the mean is a more efficient testing point than the mode or median (Emerson, 1986; King-Smith *et al.*, 1994; Treutwein, 1995). The QuestMethod option to QuestExperiment allows the user to change the underlying statistic used by Quest in selecting the next testing point. Here we collect simulated data using the default (QuestMethod->mean) and QuestMethod->mode.

```
dataMean = QuestExperiment[{-30},32,QuestSeed->99];
dataMode = QuestExperiment[{-30},32,QuestSeed->99
,QuestMethod->mode];
```

We plot the two trial sequences, suppressing other elements of QuestPlot.

```
{meanPlot,modePlot} =
  QuestPlot[#,Histogram->False,Density->False]& /@
    {dataMean,dataMode};
```

We show the two plots together, with a heavy line indicating the results for the mode (Fig. 8).

```
Show[meanPlot,Graphics[Thickness[.02]],modePlot];
```

We conclude this section with documentation for QuestExperiment.

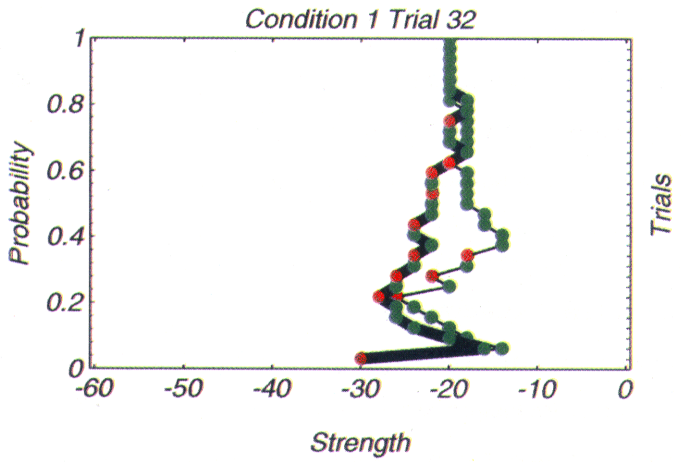

Figure 8. Comparison of simulated trial sequences with QuestMethod->mode (heavy line) and QuestMethod->mean.

QuestExperiment[guesses, trials]: Conducts an experiment using the QUEST adaptive procedure. The user must specify a list, *guesses*, of initial estimates of threshold for each condition. QuestExperiment calls a function f[condition, strength] (where f is the value of the option QuestResponse) which presents a stimulus, collects a response, and serves as the interface to the device- and stimulus-dependent parts of the experiment. QuestExperiment returns a structure of the form {sequence, strengths, functions, parameters}, where *sequence* is the complete trial sequence, each trial specified by a triple {condition, strength, response}. *strengths*, *functions*, and *parameters* are all lists, with lengths equal to the number of conditions. Each element of *strengths* is a list of possible testing strengths. Each element of *functions* is an assumed psychometric function, and parameters are their associated parameters. The following options (shown with their default values) are understood: QuestResponse->SimulatedObserver, QuestFunction->LogWeibullPF, QuestParameters->{3.5, .5, .01}, QuestMethod->mean, QuestPlotShow->False, QuestRange->{−60, 0, 2}, QuestSigma->100, QuestSeed->Automatic. QuestFunction determines the assumed psychometric function, and QuestParameters are its parameters. QuestMethod determines whether mode, median, or mean of the posterior probability function will be used as the testing point. QuestPlotShow indicates whether a plot will be produced after each trial, which is useful for tutorial purposes. QuestSigma is the initial standard deviation (in dB) of the normal prior probability density, centered at the guess. QuestRange is a list specifying the minimum, maximum and increment of the list of possible testing strengths. QuestSeed can be either a number, in which case that number will be used as the seed for all calls to Random (such as those that determine order of presentation of conditions, order within forced-choice presentations, and behavior of the simulated observer), or Automatic, in which case the seeds themselves will be randomly generated.

4.2. *ContrastThreshold*

QuestExperiment is a general mechanism for conducting a wide variety of experiments independent of specific display mechanisms. ContrastThreshold is a simple program built on top of QuestExperiment which collects contrast thresholds for one or several stimuli, using the specific display mechanisms provided by Cinematica on the Apple Macintosh computer. Defined below, ContrastThreshold can be seen to consist of telling QuestExperiment the name of the response function, starting Cinematica, loading the stimuli, evaluating QuestExperiment, closing Cinematica, and returning the data.

```
ContrastThreshold[stimuli_, guesses_, trials_,
  opts___Rule] := Module[{data},
    SetOptions[QuestExperiment, QuestResponse->
      (QuestResponse /.{opts} /.Options[ContrastThreshold])];
    SetOptions[ContrastThreshold, Stimuli:>stimuli];
    CinematicaOpen[ ];
    LoadStimuli[stimuli];
    data = QuestExperiment[guesses, trials, opts];
    CinematicaClose[ ];
  data]
```

Here we will demonstrate the use of ContrastThreshold. First we must create some stimuli. To keep things very simple, we begin with a uniform square, 32 pixels in width, displayed with fixed contrast for one frame. The stimulus format is described more fully in the documentation for ContrastThreshold.

```
square = Table[254, {32}, {32}];
contrast = 1.;
displaylist = {{1, 1}};
stimuli1 = {{square}, {contrast}, {displaylist}};
```

Now we collect a threshold for this stimulus.

```
results1 = ContrastThreshold[stimuli1, {-10.}, 32];
```

ContrastThreshold returns a data structure identical to that returned by QuestExperiment, which may be analyzed and plotted as illustrated in the simulation above. Here we show only one plot, which includes a fit (Fig. 9). We ask for a fit of both threshold and slope, though for these data, with few trials anywhere but near threshold, the estimate of slope should not be taken too seriously.

```
QuestPlot[results1, Fit->True, FitParameters->{True, True}];
```

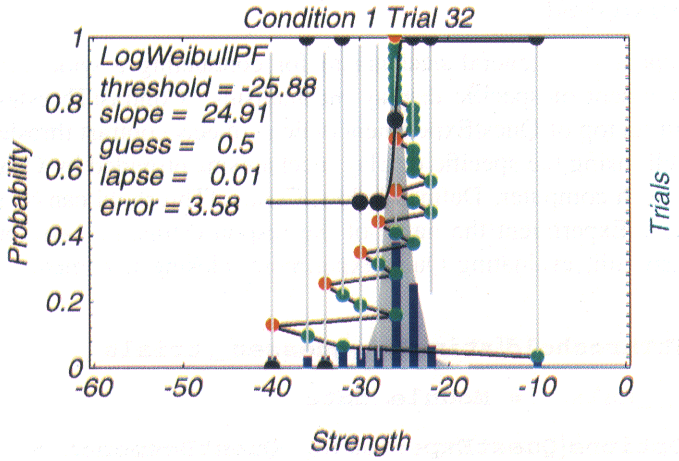

Figure 9. QuestPlot of data from ContrastThreshold in which both threshold and slope have been estimated.

In a slightly more complicated example, we use two stimuli: a bright square and a dark square, each displayed for eight frames.

```
square = Table[254, {32}, {32}];
contrasts = {1., -1.};
dlist1 = Table[{1, 1}, {8}];
dlist2 = Table[{1, 2}, {8}];
stimuli2 = {{square}, contrasts, {dlist1, dlist2}};
```

Now we collect two thresholds, so a guess must be given for each.

```
results2 = ContrastThreshold[stimuli2, {-30., -30.}, 32]
```

Either of the following plots might then be requested.

```
QuestPlot[results2, Fit->True];
PsychometricPlot[fit2 = PsychometricFit[results2]];
```

Note that if we only want the thresholds, they are easy to extract from the output of PsychometricFit.

```
#[[3, 1, 1]]& /@ fit2
{-34.1232, -33.1706}
```

4.2.1. Some options. There are a number of useful options for ContrastThreshold. By default, the fixation point is a single black pixel.

```
Options[ContrastThreshold, FixationImage]
{FixationImage -> {{2}}}
```

We can change the fixation point to any arbitrary image. Here we use a medium gray cross of size 3×3 pixels.

```
ContrastThreshold[stimuli1, {-10.}, 32
  ,FixationImage->{{0, 64, 0}, {64, 64, 64}, {0, 64, 0}}]
```

To make this change permanent within a session, we use SetOptions.

```
SetOptions[ContrastThreshold
  ,FixationImage->{{0, 64, 0}, {64, 64, 64}, {0, 64, 0}}];
```

The following uses a yes/no procedure and appropriate Quest parameters (note that guess, which is now the 'false alarm rate', has been changed to 0.1).

```
ContrastThreshold[stimuli1, {-10.}, 32
,QuestResponse->GetResponseYN
,QuestParameters-> {4., 0.1, 0.01}]
```

There are also options to vary audio warning and feedback sounds, as well as the pause between intervals in a two-alternative forced-choice (2IFC) presentations. In addition, the user can change any of the options to QuestExperiment. We conclude this section with documentation for ContrastThreshold.

ContrastThreshold[stimuli, guesses, trials]: Collect a contrast threshold for each of a number of stimuli, using *trials* trials. *guesses* is a list of the initial estimates of threshold for the stimuli. *stimuli* must be a structure of the following format: {images, contrasts, displayLists}. The first element is a list of images, each a rectangular array of integers ranging between 2 and 254 (we reserve 0, 1, and 255 for other purposes). This list should contain all the images to be used in the experiment, even though they may constitute frames from different stimuli. The second element of the structure is a list of contrasts, real numbers in the range $[-1, 1]$. The contrasts are nominal contrasts, which define the contrast of the stimulus in the unattenuated state. Again, the list should contain all contrasts, even though they may apply to different stimuli. The final element is a list of display lists, one per stimulus. Each consists of a list of number pairs, and each pair defines the image and contrast to be used on a given frame of the display. The stimuli are displayed by means of the *Cinematica* functions. By default a forced-choice method is used, but a yes/no method may be specified with the option QuestResponse->GetResponseYN. The options IntervalPause, FeedbackSounds, and WarningSound may also be specified.

4.3. Quest functions

The Quest package provides a set of component functions from which a wide variety of experiments can be constructed; QuestExperiment and ContrastThreshold are useful examples. We conclude this note with documentation on some of the individual functions of the Quest package.

QuestUpdate[condition, strength, response, qdata]: This function is called after a trial of *condition* at *strength* with outcome *response* (0 or 1). It updates and returns the structure *qdata*, which is a complete record of the experiment. The structure of qdata is {QSequence, QStrengths, QTables, QFunctions, QParameters}. QSequence is a list of triples, each of the form {condition, strength, response} which describes the trial sequence in order of occurrence. QStrengths is a list of lists, one per condition, each specifying the available strengths. QTables is a list of lists, one per condition, each specifying the current un-normalized, log posterior probability of threshold at each strength. QFunctions is a list of function names, one per condition, each specifying the psychometric function used by QUEST. QParameters is a list of lists, one per condition, each specifying the parameters assigned to the corresponding psychometric function.

QuestMean[QTable]: Given a QTable (a list of un-normalized, log posterior probabilities of threshold strength), QuestMean finds the index nearest to the approximate mean of the posterior density.

QuestMedian[QTable]: Given a QTable (a list of un-normalized, log posterior probabilities of threshold strength), QuestMedian finds the index nearest to the approximate median of the posterior density.

QuestMode[QTable]: Given a QTable (a list of un-normalized, log posterior probabilities of threshold strength), QuestMode finds the index of the mode of the posterior density.

NextIndex[QTable, method]: Selects the next index for trial placement, based on the desired method (mode, median, or mean) in the QUEST psychometric procedure. QTable is a list of un-normalized, log posterior probabilities of threshold strength.

SequenceToTable[QSequence]: Converts a QSequence data structure into a table of the form {{{strength, #wrong, #right}, ...}, ...} where there is one list of triples for each condition.

LogGauss[strength, sigma, mean]: This function is used to set a prior probability distribution for each condition, which may be done by initializing the log likelihood list (QTable). It is the log of a Gaussian (a parabola). The arguments are a strength, a standard deviation of the Gaussian, and a mean of the Gaussian.

Shuffle[list]: Randomize the order of a list.

SimulatedObserver[condition, strength]: Simulates a trial for *condition* at *strength*. It returns 1 or 0 for success or failure, respectively. By default, it obeys a LogWeibullPF psychometric function with parameters {0., 3.5, .5, .01, 0.}. It simulates a threshold of -10^* (condition+1). These may be changed for each condition via SetOptions and the option names PsychometricFunction and PsychometricParameters.

GetResponse2IFC[condition, strength]: Conduct a two-interval forced-choice trial for condition at *strength*. This function serves as the interface between the high-level function QuestExperiment and the device-dependent aspects of the function ContrastThreshold. The user may write an alternate version of this function. The arguments to this function are a condition number and a strength. From these numbers, the function should present an appropriate stimulus with an appropriate strength, collect a response from the observer, and return that response (0 or 1) as the value of the function.

GetResponseYN[condition, strength]: Conduct a yes/no trial for *condition* at *strength*. Returns 0 for no and 1 for yes.

DisplayStimulus2IFC[condition, strength, interval]: Display a two-alternative forced-choice trial of stimulus *condition* with contrast *strength* in decibels, with the signal in *interval* (1 or 2). The fixation point is removed during each presentation. Each presentation is preceded by a warning tone. This version uses Cinematica display functions.

DisplayStimulus[condition, strength]: Display a stimulus *condition* with contrast *strength* in decibels. The fixation point is removed during the presentation. The presentation is preceded by a warning tone. Pauses may be inserted before the warning tone, and between the tone and the display, through the options PreWarnPause and PostWarnPause to ContrastThreshold. This version uses Cinematica display functions.

Wait[seconds]: Wait for *seconds*. This is an alternative to the built-in function Pause, which has some peculiar rules about when it is evaluated.

Feedback[response]: Produces feedback sounds. Particular sounds can be specified via the FeedbackSounds option of ContrastThreshold. Defaults are a low-pitch or high-pitch beep, respectively, to a correct (1) or incorrect (0) response.

LoadStimuli[stimuli]: This function takes a *stimuli* data structure, and loads the images into the GWorld using the Cinematica package. The stimulus structure is described under the documentation for ContrastThreshold. The function also loads a blank image, equal in size to the largest image in the stimulus structure, and a fixation image. Two color maps are also automatically loaded, one of zero contrast and one of contrast 1.

REFERENCES

Emerson, P. L. (1986). Observations on maximum likelihood and Bayesian methods of forced-choice sequential threshold estimation. *Percept. Psychophys.* **39**, 151–153.

King-Smith, P. E., Grigsby, S. S., Vingrys, A. J., Benes, S. C. and Supowit, A. (1994). Efficient and unbiased modifications of the QUEST threshold method: Theory, simulations, experimental evaluation, and practical implementation. *Vision. Res.* **34**, 885–912.

Nachmias, J. (1981). On the psychometric function for contrast detection. *Vision Res.* **21**, 215–223.

Pelli, D. G. (1986). Uncertainty explains many aspects of visual contrast detection and discrimination. *J. Opt. Soc. Am. A* **2**, 1508–1532.

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

Pelli, D. G. and Zhang, L. (1991). Accurate control of contrast on microcomputer displays. *Vision Res.* **31**, 1337–1350.

Quick, R. F. (1974). A vector magnitude model of contrast detection. *Kybernetik* **16**, 65–67.

Solomon, J. A. and Watson, A. B. (1996). Cinematica: A system for calibrated, Macintosh-driven displays from within Mathematica. *Behav. Res. Methods Instrum. Comput.* **28**(4), 607–610.

Treutwein, B. (1995). Adaptive psychophysical procedures. *Vision Res.* **35**, 2503–2522.

Watson, A. B. (1979). Probability summation over time. *Vision Res.* **19**, 515–522.

Watson, A. B. and Fitzhugh, A. (1990). The method of constant stimuli is inefficient. *Percept. Psychophys.* **47**, 87–91.

Watson, A. B., Nielsen, K. R. K., Poirson, A., Fitzhugh, A., Bilson, A., Nguyen, K. and Ahumada, Jr., A. J. (1986). Use of a raster framebuffer in vision research. *Behav. Res. Methods Instrum. Comput.* **18**, 587–594.

Watson, A. B. and Pelli, D. G. (1983). QUEST: A Bayesian adaptive psychometric method. *Percept. Psychophys.* **33**, 113–120.

Weibull, W. (1951). A statistical distribution function of wide applicability. *J. Appl. Mech.* **18**, 292–297.

Wolfram, S. (1991). *Mathematica: A System for Doing Mathematics by Computer*, 2nd edn. Addison-Wesley, New-York.