ERG Analysis – Matlab Scripts, User Guide

1. Introductions

The scripts ERGAnalysis_SinglePulses.m and ERGAnalysis_Flicker.m assist in analyzing ERG data by finding data points that are useful for the data analysis such as analyzing on-transients, maximal photoreceptor responses, off-transients, baseline recovery and finding the critical flicker fusion frequency. The text in the PDF file needs to be pasted into MATLAB .m files. Make sure that this process does not add new lines into the code. In order to apply any of the scripts, recorded data first needs to be exported with the recording software (in our case AD Instruments) into MATLAB format, and then loaded into MATLAB (The MathWorks Inc.). The scripts generate several figures and save all settings and analyzed data automatically as a .mat file within a result folder that is automatically generated by the scripts.

The script ERGAnalysis_SinglePulses.m assists in analyzing individual ERG responses. Typically these result from stimulating the eye with relatively long lasting and clearly separated light pulses in which concurrent stimuli arrive after the response has recovered from previous stimuli. To use this script you can record as many responses as you like within each ‘data block’. The script ERGAnalysis_Flicker.m assist in analyzing ERG responses to relatively high frequency pulses and assists in finding the critical flicker fusion frequency. To be able to use this script, record each stimulus train as separate ‘data block’ (stop the recording after each train and start the recording again before the next stimulus train).

Both programs will find important points, the specific meanings of which vary between species and stimulus conditions. Note that in not all cases all computed points are meaningful. For example, some points are designed to find on and off transients that are observed in flies but often are absent from other recordings. Therefore it is important that the user carefully evaluates his/her data to determine which of the points are most meaningful for the desired analysis.

2. Load the data into MATLAB

First, export the recorded data in MATLAB format (.mat) using the recording software and load the data into MATLAB, for instance by simply dragging the file into the MATLAB workspace window or command line window.

3. Necessary variables and accepted data formats

After loading the data, there should be several variables available in the workspace window of MATLAB, not all of which however are needed for this analysis. You can use the following command in the command line to check the format of a variable: >> whos VariableName

An example of the command line that you then will see is:

```
>> whos data
    Name      Size           Bytes  Class          Attributes
    data     1x20356286 162850288 double
```

List of necessary variables and their formats:

<table>
<thead>
<tr>
<th>Name</th>
<th>Size</th>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>data</td>
<td>1xN</td>
<td>double</td>
<td>N is the total number of data points that were collected in the trial data contains the stimulus and recording data of all blocks</td>
</tr>
<tr>
<td>datastart</td>
<td>2xN</td>
<td>double</td>
<td>N is the number of data blocks, the first row contains the start indices of the response data blocks the second row contains start indices of the stimulus data blocks</td>
</tr>
<tr>
<td>dataend</td>
<td>2xN</td>
<td>double</td>
<td>has the same format as datastart but holds the data ending indices of the response and stimulus data blocks</td>
</tr>
</tbody>
</table>
4. How to initiate and run the scripts

The programs are organized to analyze each ‘data block’ individually. To do so first open either script in MATLAB and scroll to the section entitled ‘initiation’. This section contains several variables that you can change to adjust the program to your need. To get started you need to assign the correct block number to variable blocknumber. You find this variable under: ‘a’. This variable defines which block (recording sequence) is analyzed. A list and description of all variables that you might want to adjust is below. Finally run the script by clicking on the green triangular ‘run’ symbol of the editor window. Note that you can run the script multiple times without reloading your raw data.

a. blocknumber: determines which recording sequence is being analyzed.

b. savedata: if this variable is set to false then the data will not be saved, while if it is set to true, all data will be saved.

c. stim.scale: determines how big the binary stimulus is displayed on the figures.

stim.shift: determines where (relative to the response) the stimulus is displayed.
In both scripts these variables are needed in step 3.2. Change these variables if you need to adjust the appearance of the stimulus in the figures.

d. response.w_smooth: In both scripts this variable is needed in step 4 and determines the window size that is used for a moving average to smooth the raw data (response.raw) and compute response.smoothed. The larger the window size, the stronger the data is smoothed. NOTE: You will lose information if the window size is too large.

e. point_1.w_average: is the number of data points that are averaged.

In ERGAnalysis_singlePulses.m in step 6.1 it determines how many data points of response.smoothed will be averaged just before each stimulus onset to compute each point_1.average of point 1.

In ERGAnalysis_Flicker.m in step 6.1.1. and 6.1.2 it determines how many data points of response.smoothed will be averaged just before the first stimulus onset to compute the first point_1.average of point 1.

f. These variables are needed in ERGAnalysis_singlePulse.m in step 6.3. to compute point 3a, point 3b and point 3c. point_3a.w_average, point_3b.w_average and point_3c.w_average: number of data points of response.smoothed that are averaged to compute point_3a.average, point_3b.average and point_3c.average respectively.

point_3b.skip: In step 6.3.c it determines how many data points will be skipped after stimulus onset to compute point 3b. This variable allows the program to find the most negative point in the sustained ERG response, even if there is a more negative point tightly following the onset of the stimulus onset.

g. These variables are needed in both scripts in step 6.4 to compute point 4 point_4.window: In ERGAnalysis_singlePulses.m the variable determines the number of data points that will be included after the end of the stimulus when searching for point 4 (minimum response value of response.smoothed) after each stimulus offset. In ERGAnalysis_Flicker.m this variable only applies to the last stimulus offset. This point is useful if there is a negative transient signal after the stimulus is terminated.

point_4.w_average: This variable determines how many data points of response.smoothed will be averaged to compute point_4.average.

h. point_5.window: In both scripts this variable is needed in step 6.5 to compute point 5. In ERGAnalysis_singlePulses.m the variable determines the number of data points that will be included after the end of the stimulus when searching for point 5 after each stimulus offset. In ERGAnalysis_Flicker.m this variables only applies to the last stimulus offset.
This point is useful if there is a positive transient signal after the stimulus is terminated.

5. **Data analysis output**

The scripts generate several figures and store these figures with all data in a result directory which is called: “results_block_N” where N is the block number.

5.1. **figure_rawdata: raw data**

The first figure that is generated shows the raw data of the response (*response.raw*) and plots the stimulus (*stim.plot*) below it. The figure on the left is an example generated with the script `ERGAnalysis_SinglePulses.m` and the one on the right is an example generated with the script `ERGAnalysis_Flicker.m`. The figure legend indicates the maximum and minimum values that were used in the stimulus.

5.2. **figure_stim_onoff: smoothed data with stimulus on and offsets**

The second figure shows the original response (*response.raw*) in blue, the smoothed response (*response.smoothed*) in green, the stimulus (*stim.plot*) in black and data points that mark the beginning and end of the stimulus as red and magenta stars respectively. The figure on the left shows an example generated by the script `ERGAnalysis_SinglePulses.m` and on the right is an example generated by the script `ERGAnalysis_Flicker.m`. In order to better keep track of the stimuli, the number of onsets and offsets are indicated in the figure legend (in brackets).
5.3 figure_points: generated by ERGAnalysis_SinglePulses.m: points of interest

This figure shows response.smoothed in blue, stim.plot in black, and the points 1 through 6 in various colors (see figure legend).

point 1: is the average response value of response.smoothed just before the stimulus onset, point_1.w_average determined how many data points were averaged. We suggest using this value as the baseline value as it immediately precedes the onset of the stimulus.

point 2: is the maximum response value of response.smoothed between stimulus onset and offset.

point 3a: is the average minimum response value of response.smoothed between stimulus onset and offset, point_3a.w_average determined how many data points were averaged. This point is potentially useful as the strongest response value (manifested as the most negative response) of the photoreceptor.

point 3b: is also an average minimum response value of response.smoothed between stimulus onset and offset, but this time it is possible to exclude a set number of points at the beginning of the sequence (as defined in point_3b.skip), point_3b.average determined how many data points were averaged. This point can be used to evaluate the sustained photoreceptor response even if there is an overshoot at the beginning (as in our example).

point 3c: is the average response value of response.smoothed at the time of the end of the stimulus. In some cases this is the most informative point to calculate the sustained photoreceptor points, point_3b.average determined how many data points were averaged. This point can also be used as the baseline for calculating off-transient sizes.

point 4: is the average minimum response value of response.smoothed within a specified time period (defined by point_4.window) following the end of the stimulus. The variable point_4.w_average determined how many data points of response.smoothed were averaged for this point.

point 5: is the average maximum response value of response.smoothed within a window specified as for point 4. This point could be used to evaluate a transient positive signal that follows the end of the stimulus. The variable point_5.window determined how many data points after stimulus offset were included in the search and point_5.w_average determined how many data points were averaged.

point 6: indicates when the baseline value (point_1.average) is first recovered after the end of each stimulus. The number in brackets shows if the baseline was recovered and how often the baseline was recovered if the trace contains multiple stimuli. In our case, the baseline was not recovered so the number is zero and the point is absent from the graph. Had the baseline recovered, this number would be 1.
5.4 figure_points_v1 and figure_points_v2: generated by ERGAnalysis_Flicker.m: points of interest version 1/version 2

These figures show the \textit{response.smoothed} in green, the \textit{response.average} in blue, the stimulus in black, and points 1 through 5 as described below. To assess if photoreceptors follow the stimulus it is necessary to evaluate the figures of both versions. As long as one of the two versions consistently displays points 1 and 3, photoreceptors tend to follow the stimulus. However if there are failures in both versions, then photoreceptors are starting to fail following the stimulus (as illustrated in Figure 10 of the manuscript).

\textbf{point 1:} the first data point of point 1 is the average response value just before the beginning of the stimulus where \texttt{point\_1.w\_average} determined how many data points of \textit{response.smoothed} were averaged. All following data points of point 1 are maximum response values of \textit{response.smoothed} (as specified below).

\textbf{point 3:} are minimum response values of \textit{response.smoothed}.

There are two versions in which point 1 and point 3 are found. One or the other version should find consistent values as long as the response continues to follow the stimulus. The two versions are necessary because due to a delay in photoreceptor response the stimulus end can precede the strongest response. Depending on the magnitude of this delay relative to the stimulus frequency, maximum/minimum responses occur either between the end of two consecutive stimuli or between the beginning of two consecutive stimuli. Version 1 searches for maximum responses between two consecutive stimulus ends and minimum responses between two consecutive stimulus beginnings. Version 2 does the opposite (it searches for maximum responses between two consecutive stimulus beginnings and minimum responses between two consecutive stimulus endings). In this example version 2 worked better than version 1, in which points are inconsistent. Note that flicker fusion data should be examined closely for instances in which photoreceptors start to skip stimuli. This can be done easily by zooming into the resulting figure.

\textbf{point 2:} is the maximum response value of \textit{response.smoothed} between the first two stimulus onsets. This point is useful to calculate on transients if they exist.

\textbf{point 4:} is the average minimum response value within a window right after the end of the last stimulus.

The variable \texttt{point\_4.window} determines how many data points after stimulus offset are included in the search. The variable \texttt{point\_4.w\_average} determines how many data points of \textit{response.smoothed} are averaged.

\textbf{point 5:} is the average maximum response value of \textit{response.smoothed} within a window right after the end of the last stimulus. The variable \texttt{point\_5.w\_average} determines how many data points are averaged and the variable
**point_5.window** determines how many data points are included in the search.

**point 6:** is the first time when **point_1.v_1.average(1)** (baseline) is reached after the ending of the stimulus. The number in the brackets indicates if the original baseline was reached again at the end of the recording (1: it was reached, 0: it was not reached).

### 5.2 Available Variables and the content of results.mat

After running the scripts or after loading **results.mat** into the workspace of MATLAB, there are a variety of variables available for further data analysis. To access the variables, simply double click on them. A variable window will then open. If the variable is a **struct** such as **data**, it will show the content of the **struct** (such as a series of arrays). For any variable that is an array, or a single value such as **blocknumber**, clicking it will display a spreadsheet that shows the values. You then can copy the values of the spreadsheet for instance into a program like Excel for further data analysis. The following section summarizes available variables and their meanings.

#### 5.2.1 Available variables when running the script ERGAnalysis_SinglePulse.m

- **blocknumber**: is the number of the block that was analyzed
- **result_directory**: is the name of the folder into which all files were saved
- **savedata**: determines if the data is saved, 1 -&gt; is saved, 0 -&gt; is not saved
- **rawdata**: is a **struct** that holds the raw data that was initially imported into MATLAB, as illustrated in the figure.

- **data**: holds the raw recorded data
- **dataend**: holds the block end indices
- **datastart**: holds the block start indices
- **samplerate**: holds the sample rates

**point_1**: is a **struct** that holds all variables of **point 1**.

- **w_average**: is the number of data points that were averaged of **response_smoothed** before each stimulus onset to compute **value**
- **value**: is the average response value of **w_average** data points before each stimulus onset
- **index**: is the data index of **value**
- **time**: is the real time of **value**

**point_2**: is a **struct** that holds all data of **point 2**.

- **value**: is the maximum value of **response_smoothed** between the stimulus on and offset.
- **time**: is the real time of **value**
- **index**: is the data index of **value**
delay: is the time between the first stimulus onset and \textit{time}

\textit{magnitude:} is the difference between \textit{value} (which is \texttt{point\_2.value}) and \texttt{point\_1.value} of the same stimulus pulse. In many cases this directly describes the size of an on transient.

\texttt{point\_3a:} is a \texttt{struct} that holds all data of \texttt{point\_3a}.

\texttt{w\_average:} the number of data points of \texttt{response.smoothed} around \texttt{index} that are averaged to compute average \texttt{value:} is the minimum value of \texttt{response.smoothed} between stimulus on and offset \texttt{index:} is the data index of \texttt{value} \texttt{time:} is the real time of \texttt{value} \texttt{average:} is the average of \texttt{w\_average} data points of \texttt{response.smoothed} around \texttt{index} \texttt{magnitude:} is the difference between \textit{value} (which is \texttt{point\_3a.value}) and \texttt{point\_1.value} of the same stimulus pulse.

In many cases this is a good description of the magnitude of the sustained photoreceptor response

\texttt{point\_3b:} is a \texttt{struct} that holds all data of \texttt{point\_3b}.

\texttt{w\_average:} the number of data points that are averaged as for \texttt{point\_3a} \texttt{skip:} is the number of data points that will be skipped after stimulus onset \texttt{value:} is the minimum value of \texttt{response.smoothed} between \texttt{skip} data points after stimulus onset and stimulus offset. \texttt{index:} is the data index of \texttt{value} \texttt{average:} is the average response value of \texttt{response.smoothed} as for \texttt{point\_3a} \texttt{time:} is the real time of \texttt{value} \texttt{magnitude:} is the difference between \textit{value} (which is \texttt{point\_3b.value}) and \texttt{point\_1.value} of the same stimulus pulse.

In many cases this is a good description of the magnitude of the sustained photoreceptor response

\texttt{point\_3c:} is a \texttt{struct} that holds all data of \texttt{point\_3c}.

\texttt{w\_average:} the number of data points that are averaged as for \texttt{point\_3a} \texttt{index:} is the data index of \texttt{value} \texttt{value:} is the value of \texttt{response.smoothed} just before stimulus offset \texttt{average:} is the average response value of \texttt{response.smoothed} as for \texttt{point\_3a} \texttt{time:} is the real time of \texttt{value} \texttt{magnitude:} is the difference between \textit{value} (which is \texttt{point\_3c.value}) and \texttt{point\_1.value} of the same stimulus pulse.

In many cases this is a good description of the magnitude of the sustained photoreceptor response

\texttt{point\_4:} is a \texttt{struct} that holds all data of \texttt{point\_4}.

\texttt{window:} is the number of data points that are included after stimulus offset \texttt{w\_average:} the number of data points that are averaged as for \texttt{point\_3a} \texttt{value:} is the minimum value of \texttt{response.smoothed} within a \texttt{window} of data points after stimulus offset. \texttt{index:} is the data index of \texttt{value}
**time**: is the real time of value

**delay**: is the time between stimulus offset and time

**average**: is the average response value of response.smoothed as for point 3a

**magnitude_1**: is the difference between average (which is point_4.average) and point_1.average of the same stimulus pulse. This describes the total change in magnitude between the baseline and the lowest point (frequently the off transient).

**magnitude_2**: is the difference between average (which is point_4.average) and point_3c.average of the same stimulus pulse. This often is a good quantification of the off transient.

**point_5**: is a struct that holds all data of point 5.

- **window**: is the number of data points that are included after stimulus offset
- **w_average**: the number of data points that are averaged as for point 3a
- **value**: is the maximum value of response.smoothed within window data points after stimulus offset
- **index**: is the data index of value
- **average**: is the average response value of response.smoothed as for point 3a
- **time**: is the real time of value
- **delay**: is the time between the last stimulus offset and time
- **magnitude_1**: is the difference between average (which is point_5.average) and point_1.average of the same stimulus pulse. This is useful to determine the magnitude between the initial baseline reading and a maximum that occurs after the stimulus has terminated.

**magnitude_2**: is the difference between point_5.average and point_3c.average of the same stimulus pulse. This is useful to determine the magnitude between the end of the sustained photoreceptor response, and a maximum that occurs after the stimulus has terminated.

**point_6**: is a struct that holds all data of point 6. If this point does not exist, its variables (except for reachedbaseline) will hold NaN (not a number)

- **reachedbaseline**: indicates if and after how many stimuli point_1.value (baseline) was reached if the block contains more than one stimulus.
- **index**: is the data index of value
- **value**: is the response value of response.smoothed for the first time when baseline was reached after each stimulus offset
- **time**: is the real time of value
- **delay**: is the time between the last stimulus offset and time

**response**: is a struct that holds all data that is related to the response traces.

- **raw**: holds the raw response data of the analyzed block
- **num_of_points**: is the number of data points of the analyzed block
- **time**: is the real time of the block

**raw**:

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>w_smooth</td>
<td>50</td>
</tr>
<tr>
<td>raw</td>
<td>1x213928 double</td>
</tr>
<tr>
<td>num_of_points</td>
<td>213928</td>
</tr>
<tr>
<td>time</td>
<td>1x213928 double</td>
</tr>
<tr>
<td>smoothed</td>
<td>1x213928 double</td>
</tr>
</tbody>
</table>
**w_smooth**: is the window size to compute **smoothed**.

**smoothed**: is the moving average of **raw**

**stim**: is a **struct** that holds all data that is related to the stimulus traces.

**raw**: is the raw stimulus trace of the analyzed block

**max**: is the maximal stimulus value of the block

**min**: is the minimum stimulus value of the block

**binary**: is the stimulus in binary format

**scale**: defines by how much **binary** will be scaled to prepare the stimulus to be plotted

**scaled**: is the **binary** stimulus after scaling as defined by **scale**

**shift**: defines by how much **scaled** is shifted relative to the minimum response value to plot the stimulus

**plot**: is **scaled** after shifting as defined by **shift**

**onset**: is a **struct** that holds all stimulus onset data

**offset**: is a **struct** that holds all stimulus end (offset) data

The **structs** **onset** and **offset** hold the variables:

**index**: are indices of all stimulus on/offsets

**time**: are the times of each index

**num**: is the number of stimulus on/offsets

**value**: is the value of **response.smoothed** at **time**

### 5.2.2 Available variables when running the script **ERGAnalysis_Flicker.m**

The following variables are exactly the same as when running the script **ERGAnalysis_SinglePulses.m**:

- **blocknumber**
- **result_directory**
- **rawdata**
- **response**
- **savedata**
- **stim**
The following variables are somewhat different from the SinglePulses script described above:

**point_1** is a struct that holds all data of point 1.

- **w_average**: number of data points of response_smoothed that were averaged before the first stimulus onset to compute value(1)
- **v_1**: is a struct that holds all data of **point 1** version 1
- **v_2**: is a struct that holds all data of **point 1** version 2

  - **value**: are response values of all **point 1**
  - **index**: are data indices of all **value**
  - **time**: is the real time of all **value**

  version 1: **value(1)**, the first value within the array, is the average of **w_average** data points of response_smoothed just before the first stimulus onset. All other **value** are the maximum value of response_smoothed between two consecutive stimulus offsets.

  version 2: **value(1)**, the first value within the array, is the same as in version 1. All other **value** are the maximum value of response_smoothed between two consecutive stimulus onsets.

**point_2** is a struct that holds all data of **point 2**.

- **value**: is the maximum value of response_smoothed between the first stimulus onset and the second stimulus offset.
- **index**: is the data index of **value**
- **time**: is the real time of **value**
- **delay**: is the time between the first stimulus onset and **time**
- **magnitude**: is the difference between **point_1.v_1.value(1)** (baseline before stimulus onset) and **value**

**point_3** is a struct that holds all data of **point 3**

- **v_1**: holds all data of **point 3** version 1
- **v_2**: holds all data of **point 3** version 2

Each of the structs **v_1** and **v_2** holds the variables:

- **value**: are the response values of response_smoothed of all **point 3**
- **index**: is the data indices of all **value**
**time**: is the real time of all **value**

**magnitude**: is the difference b/w **value** (point_3.value) and **point_1.value** of the same stimulus pulse and version. It is the difference b/w the maximal and minimal response value of each stimulus pulse.

version 1: **value** are the minimum value of **response.smoothed** between two consecutive stimulus onsets.
version 2: **value** are the minimum value of **response.average** between two consecutive stimulus offsets.

**point_4**: is a **struct** that holds all data of **point 4**.

- **window**: number of data points that are included when searching for **point 4** right after the last stimulus offset
- **value**: is the minimum value of **response.smoothed** within **window** data points after the last stimulus offset.
- **index**: is the data index of **value**
- **time**: is the real time of **value**
- **delay**: is the time between the last stimulus offset and **time**
- **w_average**: is the number of data points of **response.smoothed** around **index** that are averaged to compute **average**.
- **average**: is the average of **w_average** data points of **response.smoothed** around **index**

**point_5**: is a **struct** that holds all data of **point 5**.

- **window**: is the number of data points that are included when searching for **point 5** right after the last stimulus offset
- **value**: is the maximum value of **response.average** within **window** data points right after the last stimulus offset
- **index**: is the data index of **value**
- **time**: is the real time of **value**
- **delay**: is the time between the last stimulus offset and **time**

**point_6**: is a **struct** that holds all data of **point 6**. If this point does not exist, its variables (except for **reachedbaseline**) will hold **NaN** (not a number)

- **reachedbaseline**: indicates if baseline (**point_1.v_1.value(1)**) was reached after the end of the stimulation
- **index**: is the data index of **value**
- **value**: is the response value of **response.smoothed** for the first time
when baseline was reached after the end of the stimulation

*time*: is the real time of *value*

*delay*: is the time between the last stimulus offset and *time*

6. Trouble shooting, the meaning of error messages that you might get

The scripts contain several error traps. These traps protect you from accidentally crashing the programs. The following explains the meaning of the error messages (in red) which will be displayed in the command window if one of these error traps terminated the script to keep it from crashing.

**the number of stimulus onsets and offsets are not equal, check stimulus trace for artifacts**

This error means that the number of stimulus beginnings and endings that were detected by the script is unequal. One reason for this could be that the recording was stopped before the end of the last stimulus. It could also be that the raw stimulus recording contains large artifacts so that the script falsely accepted one or multiple of these artifacts as a stimulus beginning or ending. If you get this error, look at your raw stimulus recording to find out what happened. If you stopped the recording before the ending of the last stimulus, you could remove your last incomplete response recording. If there are large artifacts, you need to find a way to remove these artifacts or you could use another stimulus recording if the recording was taken with the exact same timing (the time of the beginnings and endings of the stimulus must be exactly the same).

**point_1.w_average precedes the beginning of the recording**

This error means that *point_1.w_average* is too large.

**point_3b.skip is too large**

This error means that the variable *point_3b.skip* is too large. Choose a smaller value so that the starting point for searching for *point 3b* does not exceed the ending of the stimulus.

**point_4.window is too large for pulse:N or point_4.window exceeds the end of the recording**

If you get this error choose a smaller *point_4.window* so that the window within which the script searches for *point 4* will not exceed the beginning of the next stimulus or the ending of the recording.

**point_5.window is too large for pulse:N or point_5.window exceeds end of recording**

If you get this error choose a smaller *point_5.window* so that the window within which the script searches for *point 5* will not exceed the beginning of the next stimulus or the ending of the recording.
ERG ANALYSIS Single Pulses

First, the script clears all data of a previous run except for the rawdata
if exist('point_1')
    clear -regexp ^point ^fig response stim blocknumber savedata;
end

Next, there are several values that can be adjusted.
Please see the user guide for detailed explanations.

% This variable defines which block is analyzed.
blocknumber=1;
% If this variable is set to:
% true: all data will be saved
% false: data will not be saved
savedata = true;
% These variables are needed in step 3.2 to optimally plot the stimulus.
stim.scale = 100; % scales stim.binary
stim.shift = 0.02; % shifts the stim.scaled
% This variable is needed in step 4.
response.w_smooth=100; %window size for moving average
% This variable is needed in step 7.1
point_1.w_average=5000; % number of averaged data points
% This variable is needed in step 7.3, 7.4, and 7.5
point_3a.w_average = 51; % number of averaged data points
point_3b.w_average = 51; % number of averaged data points
point_3b.skip = 1000; % minimum number of data points that are between the
% 'onset of the stimulus and point 3b.
% This variables are needed in step 7.6
point_4.window = 500; % window size
point_5.window = 1000; % window size
% This variable is needed in step 7.7
point_5.w_average = 51;
stim.binary = round(stim.raw / 10);
stim.binary = stim.binary - min(stim.binary);
stim.binary = round(stim.binary / max(stim.binary)); % stimulus in binary

% 3.2 It then prepares the stimulus for the figure.
stim.plot = stim.binary / stim.scale; % scales the stimulus
stim.plot = stim.plot - (min(stim.raw) - stim.shift); % shifts it

% 4. The raw response data is then smoothed to remove noise. This is
done by computing the moving average. For that, we defined the
window size, which determines how strongly the data will be smoothed.
a = 1;
b = ones(1, response.w_smooth) / response.w_smooth;
response.smoothed = filter(b, a, response.raw); % smoothed response data
clear a b;

% 5. It finds stimulus on and offsets and related values

% 5.1 It finds the data index at which there is a stimulus on or offset
j = 1;
k = 1;
for i = 1:response.num_of_points - 1
  if (stim.binary(i) < stim.binary(i + 1)) % finds stimulus onsets
    stim.onset.index(j) = i;
    j = j + 1;
  elseif (stim.binary(i) > stim.binary(i + 1)) % finds stimulus offsets
    stim.offset.index(k) = i;
    k = k + 1;
  end
end
clear j k

% 5.2 It finds values that are related to stimulus onsets
stim.onset.num = length(stim.onset.index); % number of stimulus onsets
stim.onset.time = response.smoothed(stim.onset.index); % response values at stimulus onsets
stim.onset.time = response.time(stim.onset.index); % stimulus onset times

% 5.3 It finds values that are related to stimulus offsets
stim.offset.num = length(stim.offset.index); % number of stimulus offsets
stim.offset.value = response.smoothed(stim.offset.index); % response values at stimulus offsets
stim.offset.time = response.time(stim.offset.index); % stimulus offset times

% Error trap: in case the number of stimulus onsets is not equal to the
% number of stimulus offsets. This can happen if the stimulus trace contains
% very large artifacts or perhaps the recording was terminated before
% the last stimulus offset. Check the raw stimulus trace and see what happened.

if (not (stim.offset.num == stim.onset.num))
  error('myApp:argChk', 'the number of stimulus onsets and offsets are not equal, check stimulus trace for artifacts');
  return;
end;

% 6. It extracts point 1 through point 6
point_6.reachedbaseline = 0;
for i = 1:stim.onset.num

% 6.1 point_1 is the response value just before each stimulus onset

% Error trap: point_1.w_average precedes the beginning of the recording or
% the ending of the previous stimulus
if (point_1.index(i) - point_1.w_average < 0)
  error('myApp:argChk', 'point_1.w_average precedes the beginning of the recording');
  return;
end;
point_1.average(i) = mean(response.smoothed((point_1.index(i) - point_1.w_average):point_1.index(i)));
point_1.index(i) = point_1.index(i) - round(point_1.w_average / 2);
point_1.time(i) = response.time(point_1.index(i));

% 6.2 point_2 is the maximum response between stimulus on and offset,
% [point_2.value(i), J] = max(response.smoothed(stim.onset.index(i):stim.offset.index(i)));
point_2.index(i) = stim.onset.index(i) - round(point_1.w_average / 2);
point_2.time(i) = response.time(point_2.index(i));
point_2.delay(i) = response.time(J);
point_2.magnitude(i) = point_2.value(i) - point_1.average(i);
clear J;

% 6.3.4 point_3a is the minimum response value b/w stimulus on and offset
% [point_3a.value(i), J] = min(response.smoothed(stim.onset.index(i):stim.offset.index(i)));
point_3a.index(i) = stim.onset.index(i) + J;
begin = point_3a.index(i) - round(point_3a.w_average / 2);
i_end = point_3a.index(i)+round(point_3a.w_average/2);
point_3a.average(i) = mean(response.smoothed(i_begin:i_end));
point_3a.time(i) = response.time(point_3a.index(i));
point_3a.magnitude(i)=point_3a.average(i)- point_1.average(i);

% 6.3.b point_3b is the minimum response value b/w a certain number of
% points after stimulus onset and stimulus offset

%Error trap: point_3b.skip exceeds stimulus offset, make it smaller
if (i_begin > stim.offset.index(i))
    error('myApp:argChk', 'point_3b.skip is too large');
end;

%end of error trap

point_3b.value(i),J]=min(response.smoothed(i_begin:stim.offset.index(i)));
point_3b.index(i) = stim.onset.index(i)+point_3b.skip;
point_3b.time(i) = response.time(point_3b.index(i));
point_3b.magnitude(i)= point_3b.average(i)- point_1.average(i);

% 6.3.c point_3c is the minimum response value just before stimulus offset

point_3c.index(i) = stim.offset.index(i)-1;
point_3c.value(i) = response.smoothed(point_3c.index(i)-1);
point_3c.average(i)= mean(response.smoothed((point_3c.index(i)-point_3c.w_average):point_3c.index(i)));% 6.4 point_4 is the minimum response value b/w the end of the stimulus
% and a defined number of data points thereafter as defined
% by point_4.window.

%Error trap: point_4.window exceeds stimulus onset of the next pulse
% or the end of the recording
if (i<stim.offset.num)
    if((stim.offset.index(i)+point_4.window)>stim.onset.index(i+1))
        error('myApp:argChk', strcat('point_4.window is too large for pulse:',num2str(i+1)));
    return; end;
end;

%end of error trap

point_4.value(i),J]=min(response.smoothed(stim.offset.index(i):(stim.offset.index(i)+point_4.window)));% 6.5 point_5 is the maximal response value b/w the end of the stimulus
% and a defined number of data points thereafter as defined
% by point_5.window.

%Error trap: point_5.window exceeds stimulus onset of the next pulse
% or the end of the recording
if (i<stim.offset.num)
    if((stim.offset.index(i)+point_5.window)>stim.onset.index(i+1))
        error('myApp:argChk', strcat('point_5.window is too large for pulse:',num2str(i+1)));
    return; end;
end;

%end of error trap

point_5.value(i),J]=max(response.smoothed(stim.offset.index(i):(stim.offset.index(i)+point_5.window)));
point_6.delay(i) = response.time(point_6.index(i)-stim.offset.index(i));
point_6.reachedbaseline = point_6.reachedbaseline+1;
end
clear J;
end
clear i;

%-----------------------------------------------------------------------
%                             MAKE FIGURES
%-----------------------------------------------------------------------

% It makes a figure of the smoothed and averaged response data, and
% point_1 through point_6
% figure_points = figure('numbertitle','off', 'name', 'points of interest');
title('stimulus, smoothed response, and useful data points');
ylabel('response'); % y axis label
xlabel('time(s)'); % x axis label
hold on;
plot(response.time,stim.plot,'black');
plot(point_1.time,point_1.average,'ro', 'markersize',9,'Linewidth',2);
plot(point_2.time,point_2.value,'go', 'markersize',9,'Linewidth',2);
plot(point_3a.time,point_3a.average,'mo','markersize',9,'Linewidth',2);
plot(point_3b.time,point_3b.average,'yo','markersize',9,'Linewidth',2);
plot(point_3c.time,point_3c.average,'ko','markersize',9,'Linewidth',2);
plot(point_4.time,point_4.average,'co','markersize',9,'Linewidth',2);
plot(point_5.time,point_5.value,'bo','markersize',9,'Linewidth',2);
plot(point_6.time,point_6.value,'r*','markersize',10,'Linewidth',1.5);
hold off;
legend ('stimulus','response smoothed, block:',num2str(blocknumber),...
    'point 1', 'point 2', 'point 3a', 'point 3b', 'point 3c', 'point 4',...
    'point 5','point 6 (',num2str(point_6.reachedbaseline),')');

% It makes a figure of the stimulus, raw response data, smoothed response
% and data and stimulus on and offsets
figure_stim_onoff = figure('numbertitle','off', 'name', 'smoothed data with stimulus on and offsets');
title('stimulus, raw & smoothed response, and pulse on- & offsets');
ylabel('response'); % y axis label
xlabel('time(s)'); % x axis label
hold on;
plot(response.time,stim.plot,'black');
plot(response.time,response.raw,'b');
plot(response.time(response.w_smooth:end),response.smoothed(response.w_smooth:end),'g');
plot(stim.onset.time,stim.onset.value,'r*','markersize',10,'Linewidth',1.5);
plot(stim.offset.time, stim.offset.value,'m*','markersize',10,'Linewidth',1.5);
hold off;
legend ('stimulus', 'raw response, block:',num2str(blocknumber),...
    'smoothed response',...
    'stimulus onsets (',num2str(stim.onset.num),')',...
    'stimulus offsets (',num2str(stim.offset.num),')');

% It makes a figure of the stimulus and the raw response data
figure_rawdata = figure('numbertitle','off', 'name', 'rawdata');
title(strcat('stimulus and raw response of block:',...
    num2str(blocknumber)));
ylabel('response'); % y axis label
xlabel('time(s)'); % x axis label
hold on;
plot(response.time,response.raw,'b');
plot(response.time,stim.plot,'black');
hold off;
legend ('raw response',strcat('stimulus, max:',...
    num2str(stim.max),', min:',num2str(stim.min)));

%-----------------------------------------------------------------------
%                             SAVES ALL DATA
%-----------------------------------------------------------------------

%It makes a directory into which all figures and results will be saved.
The name of the directory is: results_block_N where N is the blocknumber
if savedata
    result_directory = strcat('results_block_',num2str(blocknumber));
    mkdir(result_directory);
end

%this saves all data that is in the workspace to a .m file.
savestrcat(result_directory, '\results_block',num2str(blocknumber)));
ERG ANALYSIS Flicker

This script assists in analyzing ERG recordings when the eye was
stimulated with light pulses of high frequency. It is recommended to be
used when the frequency is too high to yield proper analysis with the
single pulse script. The script finds five points, makes four figures,
and saves all data as described in the user guide. The script can be
applied after the data is exported and loaded into MATLAB as described
in the user guide.

INITIATION

First, the script clears all data of a previous run except for the rawdata
if exist('point_1')
clear -regexp "point .*fig response stim blocknumber savedata;"
end

Next, there are several values that can be adjusted.

a. This variable defines which block is analyzed.
   blocknumber=5;

b. If this variable is set to:
   true: all data will be saved
   false: no data will not be saved
   savedata = true;

c. These variables are needed in step 3.2 to optimally plot the stimulus.
   stim.scale = 100; % scales stim.binary
   stim.shift = 0.02; % shifts the stim.scaled

d. This variable is needed in step 4.
   response.w_smooth=50; % window size for moving average


e. This variable is needed in step 7.1.1 and 7.1.2
   point_1.w_average=1000; % number of averaged data points

f. These variables are needed in step 7.4
   point_4.window = 5000; % window size
   point_4.w_average = 2; % number of averaged data points

gh. This variable is needed in step 7.5
   point_5.window = 5000; % window size
   point_5.w_average = 51;

COMPUTATIONS DONE BY THE SCRIPT

1. It collects the relevant original data in a struct and clears
   all original data that is not needed
   if exist('dataend')
   rawdata.data = data;
   rawdata.datastart = datastart;
   rawdata.dataend = dataend;
   rawdata.samplerate = samplerate;
   rawdata.titles = titles;
   rawdata.com = range.unittext samplerate blocktimes titles...
   firstsampleoffset tickrate;
   clear -regexp "data.*com.*range.*unittext samplerate blocktimes titles..."

2. It extracts the raw data
   response.raw = rawdata.data(1,blocknumber):rawdata.dataend(1,blocknumber));
   response.time = (1:response.num_of_points)/response.samplerate(1);

3. The stimulus is prepared so that we can easily find stimulus on and
   offsets and it looks nice on the figures
   stimulus = round(100*stim.raw); % full integers of either 0 or 1 (binary)

4. The raw response data is the smoothed to remove noise. This is
   done by computing the moving average. For that, we defined the
   window size, which determines how strongly the data will be smoothed.
   a = 1;
   b = ones(1, response.w_smooth)/response.w_smooth;
   response.smoothed = filter(b, a, response.raw); % smoothed response data
   clear a b;
5. It finds stimulus on and offsets and related values

5.1 It finds the data index at which there is a stimulus on or offset

\[ j = 1; \]
\[ k = 1; \]
for \( i = 1 \) to \( \text{response.num.of.points} - 1 \)
\[
\begin{align*}
\text{if stim.binary(i) < stim.binary(i+1)} & \text{ finds stimulus onsets} \\
\text{stim.onset.index}[j] &= i; \\
\text{else if stim.binary(i) > stim.binary(i+1)} & \text{ finds stimulus offsets} \\
\text{stim.offset.index}[k] &= i;
\end{align*}
\]
\[ k = k + 1; \]
end
\[
\text{clear j k}
\]

5.2 It finds values that are related to stimulus onsets
\[
\text{stim.onset.value} = \text{response.smoothed(stim.onset.index)}; \quad \text{response values at stimulus onsets}
\]
\[
\text{stim.onset.time} = \text{response.time(stim.onset.index)}; \quad \text{stimulus onset times}
\]

5.3 It finds values that are related to stimulus offsets
\[
\text{stim.offset.value} = \text{response.smoothed(stim.offset.index)}; \quad \text{response values at stimulus offsets}
\]
\[
\text{stim.offset.time} = \text{response.time(stim.offset.index)}; \quad \text{stimulus offset times}
\]

Error trap: in case the number of stimulus onsets is not equal to the number of stimulus offsets. This can happen if the stimulus trace contains very large artifacts or perhaps the recording was terminated before the last stimulus offset. Check the raw stimulus trace and see what happened.

\[
\text{if not(stim.offset.num == stim.onset.num)} \\
\text{error('myApp:argChk', 'the number of stimulus onsets and offsets are not equal, check stimulus trace for artifacts');} \\
\text{return; end;}
\]

6. Extracts point 1 through point 5

\[ \text{for i = 1} \text{to stim.onset.num} \]

6.1 It finds relevant points around the first stimulus onset

6.1.1 point_1.v_1(1): first value is the response just before first stimulus begins

\[
\begin{align*}
\text{point_1.v_1.value}[1] &= \text{mean(response.smoothed((stim.onset.index(1)-point_1.w_average):(stim.onset.index(1))));} \\
\text{point_1.v_1.index}[1] &= \text{stim.onset.index(1)-round(point_1.w_average/2);} \\
\text{point_1.v_1.time}[1] &= \text{response.time(point_1.v_1.index(1));}
\end{align*}
\]

6.1.1 point_1.v_2(1): the first value is the response just before the first stimulus begins
\[
\begin{align*}
\text{point_1.v_2.value}[1] &= \text{point_1.v_1.value}[1]; \\
\text{point_1.v_2.index}[1] &= \text{point_1.v_1.index(1);} \\
\text{point_1.v_2.time}[1] &= \text{point_1.v_1.time(1);}
\end{align*}
\]

6.2 point_2: maximum response value b/w the first stimulus
\[
\begin{align*}
\text{point_2.value}[1] &= \text{max(response.smoothed(stim.onset.index(1):stim.offset.index(2)));} \\
\text{point_2.index} &= \text{stim.onset.index(1)+R;} \\
\text{point_2.time} &= \text{response.time(point_2.index);} \\
\text{point_2.delay} &= \text{response.time(R);} \\
\text{point_2.magnitude} &= \text{point_2.value - point_1.v_1.value(1);}
\end{align*}
\]

6.3.1 point_3.v_1(1): is the first minimum response value version 1,
\[
\begin{align*}
\text{point_3.v_1.value}[1] &= \text{min(response.smoothed(stim.onset.index(1):stim.offset.index(2)));} \\
\text{point_3.v_1.index}[1] &= \text{stim.onset.index(1)+R;} \\
\text{point_3.v_1.time}[1] &= \text{response.time(point_3.v_1.index(1));} \\
\text{point_3.v_1.magnitude} &= \text{point_3.v_1.value(1) - point_1.v_1.value(1);}
\end{align*}
\]

6.3.2 point_3.v_2(1): is the first minimum response value version 2,
\[
\begin{align*}
\text{point_3.v_2.value}[1] &= \text{min(response.smoothed(stim.offset.index(1):stim.offset.index(2)));} \\
\text{point_3.v_2.index}[1] &= \text{stim.offset.index(1)+R;} \\
\text{point_3.v_2.time}[1] &= \text{response.time(point_3.v_2.index(1));} \\
\text{point_3.v_2.magnitude} &= \text{point_3.v_2.value(1) - point_1.v_1.value(1);}
\end{align*}
\]

6.4 point_4: minimum response value after last stimulus ends
\[
\begin{align*}
\text{point_4.value}[1] &= \text{min(response.smoothed(stim.offset.index(i):stim.offset.index(i)+point_4.window)));} \\
\text{point_4.index} &= \text{stim.offset.index(i)+R;} \\
\text{point_4.time} &= \text{response.time(point_4.index);}
\end{align*}
\]

Error trap: point_4.window exceeds stimulus onset of the next pulse or the end of the recording

\[
\text{if(stim.offset.index(i)+point_4.window)>response.num.of.points} \\
\text{error('myApp:argChk', 'point_4.window exceeds the end of the recording');} \\
\text{return; end;}
\]

End of error trap
% 6.5 point_5: maximum response value after last stimulus ending
% Error trap: point_5.window exceeds stimulus onset of the next pulse
if((stim.offset.index(i)+point_5.window)>response.num_of_points)
    error('myApp:argChk', 'point_5.window exceeds end of recording');
    return;
end;
% end of error trap
[point_5.value,R] = max(response.smoothed(stim.offset.index(i):(stim.offset.index(i)+point_5.window)));
point_5.index = stim.offset.index(i)+R;
point_5.time = response.time(point_5.index);
point_5.average = mean(response.smoothed(i_begin:i_end));
point_5.magnitude_1 = point_5.average - point_1.v_1.value(1);
clear i_begin i_end R;

% 6.6 point_6 is the first time when baseline is reached after stimulus
% offset
J = find(response.smoothed(stim.offset.index(i):end)>=point_1.v_1.value(1));
if isempty(J)
    point_6.index = NaN(1);
    point_6.value = NaN(1);
    point_6.time = NaN(1);
    point_6.delay = NaN(1);
    point_6.reachedbaseline = 0;
else
    point_6.index = stim.offset.index(i)+J(1);
    point_6.time = response.time(point_6.index);
    point_6.value = response.smoothed(point_6.index);
    point_6.delay = response.time(point_6.index -stim.offset.index(i));
    point_6.reachedbaseline = 1;
end
clear J;

% finds relevant data points between first and last stimulus
else
    % 6.1.1 point_1.v_1: max response values b/w two consecutive stimulus beginnings
    [point_1.v_1.value(i),R] = max(response.smoothed(stim.onset.index(i):stim.onset.index(i+1)));
    point_1.v_1.index(i) = stim.onset.index(i)+R;
    point_1.v_1.time(i) = response.time(point_1.v_1.index(i));
    clear R;
    % 6.1.2 point_1.v_2: max response values b/w two consecutive stimulus endings
    [point_1.v_2.value(i),R] = max(response.smoothed(stim.onset.index(i):stim.onset.index(i+1)));
    point_1.v_2.index(i) = stim.onset.index(i)+R;
    point_1.v_2.time(i) = response.time(point_1.v_2.index(i));
    clear R;
    % 6.2.1 point_2.v_1: min response values b/w two consecutive stimulus beginnings
    [point_2.v_1.value(i),R] = min(response.smoothed(stim.onset.index(i):stim.onset.index(i+1)));%
    point_2.v_1.index(i) = stim.onset.index(i)+R;
    point_2.v_1.time(i) = response.time(point_2.v_1.index(i));
    point_2.v_1.magnitude(i)=point_1.v_1.value(i)-point_3.v_1.value(i);
    clear R;
    % 6.2.2 point_2.v_2: min response values b/w two consecutive stimulus endings
    [point_2.v_2.value(i),R] = min(response.smoothed(stim.onset.index(i):stim.onset.index(i+1)));%
    point_2.v_2.index(i) = stim.onset.index(i)+R;
    point_2.v_2.time(i) = response.time(point_2.v_2.index(i));
    point_2.v_2.magnitude(i)=point_1.v_2.value(i)-point_3.v_2.value(i);
    clear R;
end
 clear i;

%-----------------------------------------------------------------------
%                             MAKE FIGURES
%-----------------------------------------------------------------------

figure_points_v2= figure('numbertitle','off','name','points of interest version 2');
title('points of interest version 2');
xlabel('time(s)'); % x axis label
ylabel(strcat('response'));% y axis label
hold on;
plot(response.time,stim.plot,'black');
plot(response.time(response.w_smooth:end),response.smoothed(response.w_smoothend), 'b');
plot(point_1.v_2.time,point_1.v_2.value,'ro','markersize',8,'Linewidth',1.5);
plot(point_2.v_2.time,point_2.v_2.value,'go','markersize',8,'Linewidth',1.5);
plot(point_3.v_2.time,point_3.v_2.value,'mo','markersize',8,'Linewidth',1.5);
plot(point_4.v_2.time,point_4.v_2.value,'co','markersize',8,'Linewidth',1.5);
plot(point_5.v_2.time,point_5.v_2.value,'bo','markersize',8,'Linewidth',1.5);
plot(point_6.v_2.time,point_6.v_2.value,'ro','markersize',8,'Linewidth',1.5);
hold off;
legend ('stimulus','strcat('response smoothed, block:',num2str(blocknumber)),...
    'point 1 version 2', 'point 2', 'point 3 version 2',...,
    'point 4’, 'point 5’,strcat('point 6 (',num2str(point_6.reachedbaseline),')');

%It makes a figure of the smoothed and averaged response data and
%point_1 through point_4 version 1
figure_points_v1= figure('numbertitle','off','name','points of interest version 1');
title('points of interest version 1');

% it makes a figure of the smoothed and averaged response data and
% point_1 through point_4 version 2
figure_points_v2= figure('numbertitle','off','name','points of interest version 2');
title('points of interest version 2');
xlabel('time(s)'); % x axis label
ylabel(strcat('response'));% y axis label
hold on;
plot(response.time,stim.plot,'black');
plot(response.time(response.w_smooth:end),response.smoothed(response.w_smoothend), 'b');
plot(point_1.v_2.time,point_1.v_2.value,'ro','markersize',8,'Linewidth',1.5);
plot(point_2.v_2.time,point_2.v_2.value,'go','markersize',8,'Linewidth',1.5);
plot(point_3.v_2.time,point_3.v_2.value,'mo','markersize',8,'Linewidth',1.5);
plot(point_4.v_2.time,point_4.v_2.value,'co','markersize',8,'Linewidth',1.5);
plot(point_5.v_2.time,point_5.v_2.value,'bo','markersize',8,'Linewidth',1.5);
plot(point_6.v_2.time,point_6.v_2.value,'ro','markersize',8,'Linewidth',1.5);
hold off;
legend ('stimulus','strcat('response smoothed, block:',num2str(blocknumber)),...
    'point 1 version 2', 'point 2', 'point 3 version 2',...,
    'point 4’, 'point 5’,strcat('point 6 (',num2str(point_6.reachedbaseline),')'));

%It makes a figure of the smoothed and averaged response data and
%point_1 through point_4 version 1
figure_points_v1= figure('numbertitle','off','name','points of interest version 1');
title('points of interest version 1');
xlabel('time(s)'); % x axis label
ylabel(strcat('response'));% y axis label
hold on;
plot(response.time,stim.plot,'black'); % plot stimulus
plot(point_1.v_1.time,point_1.v_1.value,'ro','markersize',8,'Linewidth',1.5);
plot(point_2.time, point_2.value,'go','markersize',8,'Linewidth',1.5);
plot(point_3.v_1.time,point_3.v_1.value,'mo','markersize',8,'Linewidth',1.5);
plot(point_4.time,point_4.average,'co','markersize',8,'Linewidth',1.5);
plot(point_5.time,point_5.value,'bo','markersize',8,'Linewidth',1.5);
plot(point_6.time,point_6.value,'r*','markersize',8,'Linewidth',1.5);
hold off;
legend ('stimulus',strcat('response smoothed, block:',num2str(blocknumber)),...
'point 1 version 1', 'point 2','point 3 version 1','point 4','point 5',...
'start point 6 (' num2str(point_6.reachedbaseline)'),');

figure_stim_onoff= figure('numbertitle','off','name','smoothed data with stimulus on and offsets');
title('stimulus, raw & smoothed response data, and pulse on- & offsets');
xlabel('time(s)'); % x axis label
ylabel(strcat('response'));% y axis label
hold on;
plot(response.time,stim.plot,'black'); % plot stimulus
plot(response.time,response.raw,'b'); % plot raw response
plot(response.time(response.w_smooth:end),response.smoothed(response.w_smooth:end),'g'); % plot smoothed response
plot(stim.onset.time,stim.onset.value,'r*','markersize',7,'Linewidth',1);
plot(stim.offset.time, stim.offset.value,'m*','markersize',7,'Linewidth',1);
hold off
legend ('stimulus',strcat('raw response, block:',num2str(blocknumber)),...
'smoothed response',...
'start stimulus offsets (' num2str(stim.onset.num),')',...
'start stimulus offsets (' num2str(stim.offset.num),'));

% It makes a figure of the stimulus and raw response data
figure_rawdata = figure('numbertitle','off','name','rawdata');
title(strcat('stimulus and raw response of block:',...
um2str(blocknumber)));% end of script

% SAVES ALL DATA
result_directory = strcat('results_block_',num2str(blocknumber));
endir(result_directory);

%this saves all data that is in the workspace to a .m file.
save(result_directory, '\results_block',num2str(blocknumber)));