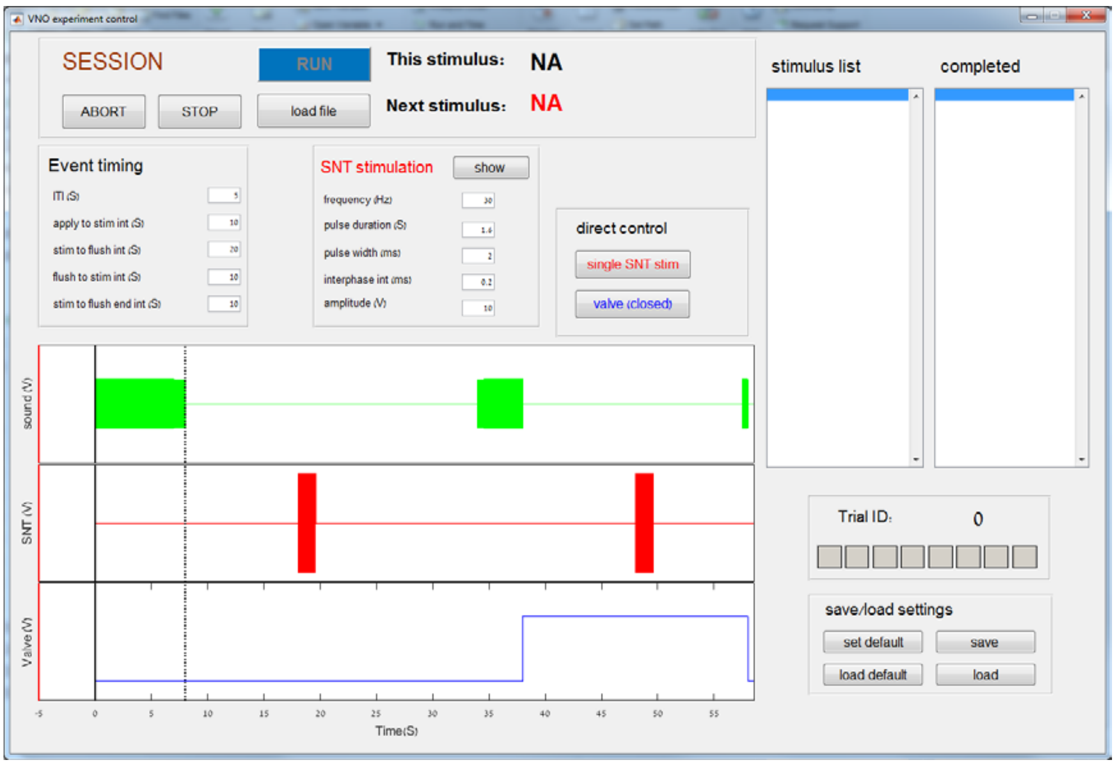
**Instructions for the Session Control Program (RUN_VNS_EXP) interface and related programs.**

The software is part of an experimental setup for stimulus delivery to the VNO, typically combined with electrophysiological recordings. The complete setup is described in the article that this manual accompanies. In addition to the regular MATLAB product, the program requires the MATLAB data acquisition (DAQ) toolbox which is primarily designed for use with National Instruments devices. See the MATLAB DAQ documentation for instructions on installing the required board driver software. The code has been tested with the R2016b release of MATLAB. Older versions may not be compatible with the graphic interface.

To run the session control program, add the VNS_EXP folder to the MATLAB path (see MATLAB documentation on the path and how to add folders to it). To create the GUI (**Figure 1**), type RUN_VNS_EXP on the MATLAB command prompt. Note that the GUI will only function if the program can establish a connection with an input output board which supports 3 analog output channels and 8 digital output channels (e.g. NI-USB-6343, National Instruments). If a successful connection is established, the program will create directories for the log files, stimulus files, and parameter files (all are child directories of the folder containing the MALTAB code).

Advanced notes on establishing connections with I/O board: If using a board that is not compatible with the default definitions, the establishment of communication protocols should be modified. The relevant definitions appear in a section with a header title "define DAQ settings section" within the function *RUN_VNS_EXP_OpeningFcn*, which itself is part of the main GUI function, namely, *RUN_VNS_EXP*. The *RUN_VNS_EXP_OpeningFcn* also specifies the temporal resolution of the outputs to the I/O board (scans per second, set by default at 10000). For communication with the board the program creates a digital and a continuous session. The digital session is required for the trial ID bits (see below), while the continuous session controls the output analog channels. During the session, the digital outputs are set using the *digiS.outputSingleScan()* function. The analog channels are set using the *continS.queueOutputData()* and *continS.startBackground()* functions.



This stimulus: NA
Next stimulus: NA

Event timing

| | |
|---|---|
| ITI (S) | 5 |
| apply to stim int (S) | 10 |
| stim to flush int (S) | 20 |
| flush to stim int (S) | 10 |
| stim to flush end int (S) | 10 |

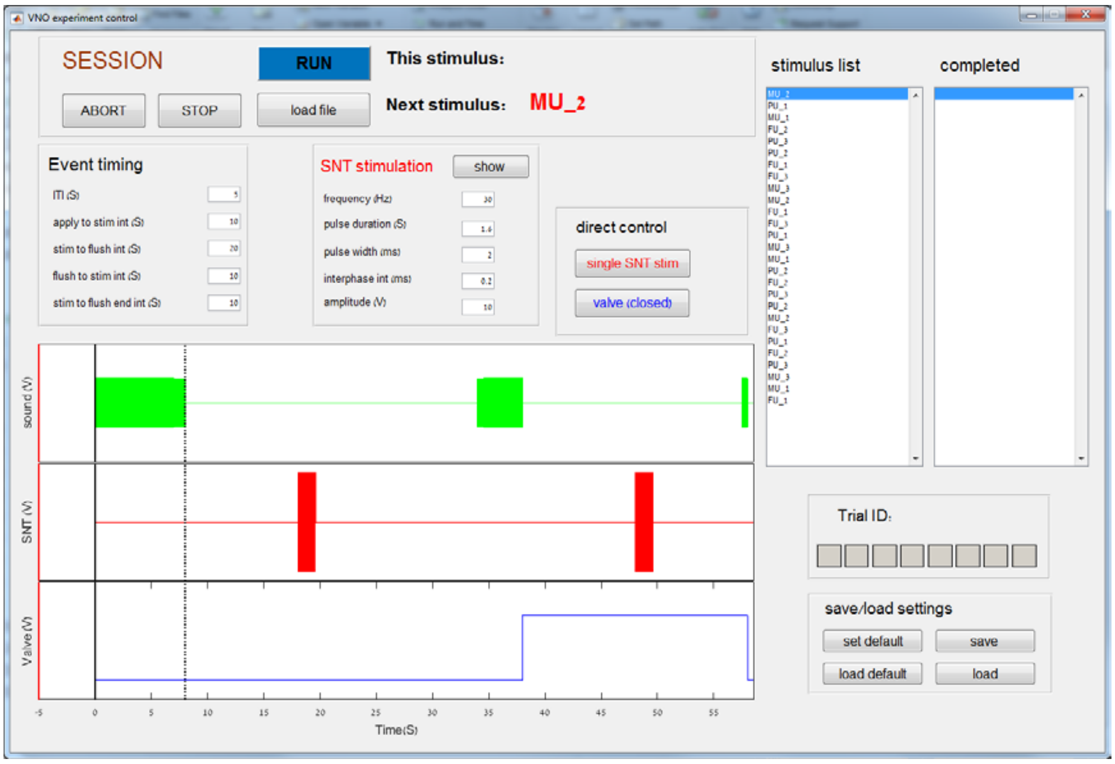
SNT stimulation

| | |
|---|---|
| frequency (Hz) | 30 |
| pulse duration (S) | 1.4 |
| pulse width (ms) | 2 |
| interphase int (ms) | 0.2 |
| amplitude (V) | 10 |

Trial ID: 0

**Figure 1:** The GUI, before setting a stimulus file.

Before running a session, a stimulus list must be loaded. The stimulus list is specified in a text file, where each line is one stimulus. Files can be written manually, or created with the function *create_stimulus_file*. The function must be edited to specify the list of stimuli and the number of blocks. It will then create a .txt file that includes a repeating list of stimuli, shuffled within each block. The file name is determined by the date of its creation. To use multiple files created on the same day, it is required to change their names manually (so that they will not be overwritten). Use the *load file* button on the GUI to select one such text files. Once done, the list of stimuli as appears in the file, will populate the left edit box (**Figure 2**). It is only after a stim file is loaded, that the RUN button on the GUI is enabled (compare **Figure 1** and **Figure 2**). When the RUN button is pressed, the program, will run through all the stimuli in the list, one trial per item in the list.



This stimulus:
Next stimulus: MU_2

Event timing

| | |
|---|---|
| ITI (S) | 5 |
| apply to stim int (S) | 10 |
| stim to flush int (S) | 20 |
| flush to stim int (S) | 10 |
| stim to flush end int (S) | 10 |

SNT stimulation

| | |
|---|---|
| frequency (Hz) | 30 |
| pulse duration (S) | 1.6 |
| pulse width (ms) | 2 |
| interphase int (ms) | 0.2 |
| amplitude (V) | 10 |

stimulus list: MU_2, PU_1, MU_1, FU_2, PU_3, PU_2, FU_1, FU_3, MU_3, MU_2, FU_1, FU_3, PU_1, MU_3, MU_1, PU_2, FU_2, PU_3, PU_2, MU_2, FU_3, PU_1, FU_2, PU_3, MU_3, MU_1, FU_1

**Figure 2:** The GUI, after selecting a stimulus file.

The trial structure and stimulation waveform are defined in the *Event Timing* and *SNT stimulation* panels, respectively. The parameter names should be self-explanatory. The analog signals, which define the trial structure, are shown in the lower left display. The first is the sound signal, which instructs the experimenter when to apply the stimuli and wash solution. The second is the command voltage to the electric stimulator, while the third controls valve opening and closing. The continuous vertical black line indicates the time of trial start (at the end of inter trial interval), while the broken vertical line indicates the time of stimulus application. When any parameter is changed by the user, the display updates. Because the display has a limited resolution, small changes will not be visible in this display. For example small changes in the frequency of the SNT stimulation waveform will not be

<u>Advanced notes on changing parameter values:</u> All changes to GUI parameters are handled by the function *update_stim_params*, within the main *RUN_VNS_EXP* function. The function itself contains a definition of the valid range for each parameter and a default value, which is assigned if the user selects a parameter value that is out of the valid range. These values can be edited. When called, the *update_stim_params* function calls the function *generate_single_VNS_trial_scan* which prepares the analog signals for each of the trials. This function also contains the definition of the actual voltage ranges, which depend on the specific devices controlled by them (and thus must be changed for adapting to different devices). The function also calls the function *generate_trial_sounds* which creates the sound waveforms to be delivered to the speakers. The sounds can therefore be changed using this function. Once the signals have been created according to the current parameters, the *update_stim_params* function plots the signals in the GUI axes (thus updating the display) and stores them in the GUI figure handle DATA structure.

apparent in the display. The *show* button in the *SNT stimulation panel* will generate a new figure showing only a single stimulation command train, providing finer resolution (**Figure 3**).

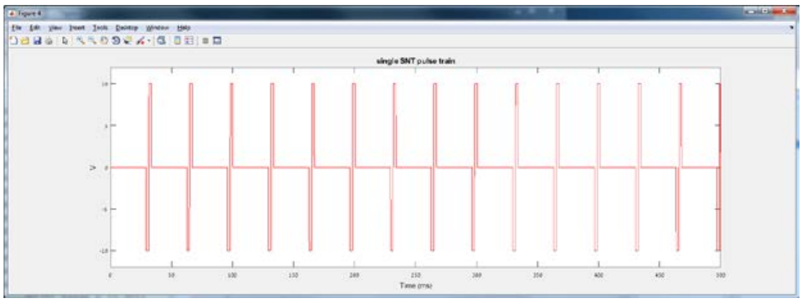
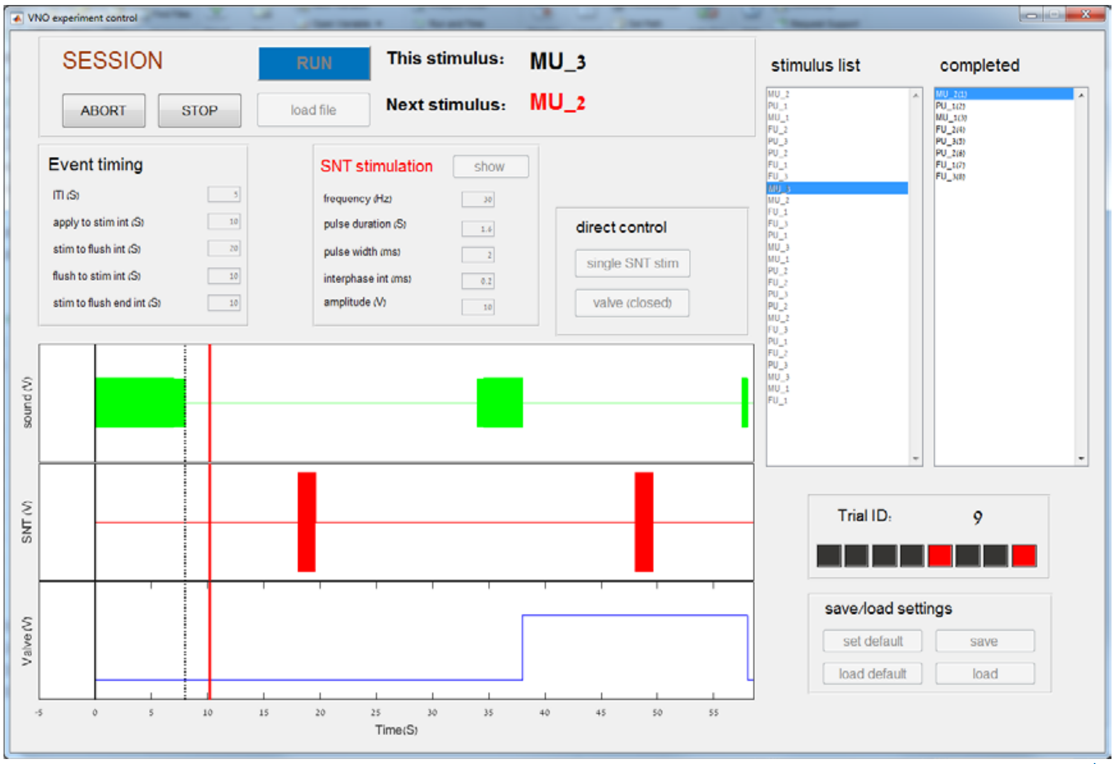


**Figure 3:** A display of the stimulation command waveform, evoked after pressing the *SHOW* button.

**Figure 4** shows the GUI as a session is in progress, that is, after the RUN button was pressed and several trials were completed. Several aspects should be noted:
1. The current time within the trial is indicated by the moving red vertical cursor.
2. The current stimulus is shown in black at the top of the GUI
3. The next stimulus is shown in red at the top of the GUI, allowing the experimenter to prepare it for the next trial.
4. The trial ID is indicated at the lower right side of the GUI. In addition to its conventional representation, the trial ID is also represented as the bitmap which is sent to the data acquisition board for logging. The trial ID is a unique identifier of the trial within the session in progress. Each trial, even if aborted during execution, is associated with a unique ID. This is critical to allow associating each trial with the stimulus that was presented.
5. The list box on the right of the GUI shows the stimuli that were already presented. Note that this list also shows the unique trial ID for each stimulus presentation (in parenthesis).
6. Most other buttons are disabled during the run.

While running, session execution can be stopped in two different ways. When the *STOP* button is pressed, the current trial will continue until completed, and then session execution will pause. If the *ABORT* button is pressed, the trial will immediately stop, and the program will pause execution. In both cases, the session can be resumed by pressing the RUN button. Trial parameters and SNT stimulation parameters can be changed when the program is paused, so that different trials within a session may be associated with different parameters. The specific parameters valid for each trial are stored in the log file (see section on log file below).



This stimulus: MU_3

Next stimulus: MU_2

Event timing

| | |
|---|---|
| ITI (S) | 5 |
| apply to stim int (S) | 10 |
| stim to flush int (S) | 20 |
| flush to stim int (S) | 10 |
| stim to flush end int (S) | 10 |

SNT stimulation

| | |
|---|---|
| frequency (Hz) | 30 |
| pulse duration (S) | 1.6 |
| pulse width (ms) | 2 |
| interphase int (ms) | 0.2 |
| amplitude (V) | 10 |

stimulus list

MU_2
PU_1
MU_1
FU_2
PU_3
PU_2
FU_1
FU_3
MU_3
MU_2
FU_1
FU_3
PU_1
MU_3
MU_1
PU_2
FU_2
PU_3
PU_2
MU_2
FU_3
PU_1
FU_2
PU_3
MU_3
MU_1
FU_1

completed

MU_2(1)
PU_1(2)
MU_1(3)
FU_2(4)
PU_3(5)
PU_2(6)
FU_1(7)
FU_3(8)

Trial ID: 9

**Figure 4:** The GUI, while a session is in progress. The program completed 8 trials, and the 9th trial is in progress.

Advanced notes regarding program execution: The run button initiates a loop that unless stopped or aborted, runs over all stimuli in the list. In each trial, it calls the *run_once_button_Callback* function which sends the signals to the I/O board (using the built in *queueOutputData* function). The *run_once_button_Callback* function also creates a timer object that updates the cursor position (red vertical bar shown in **Figure 4**) during the trial. It has an update rate of 0.2 seconds.

Changing the stimulus: When the program is not running, the current position in the list can be changed, thus allowing for example to skip or alternatively, to repeat trials. This is useful if there was some problem during delivery of a particular stimulus. To change the next stimulus, simply select it from the stimulus list (not the list of completed stimuli). A warning message is issued to prevent accidentally changing the list of stimuli.

Direct control of stimulation and of the suction valve: It is often necessary to control the valve directly, for example when checking the efficiency of suction. This can be done via the interface using the *valve* button. A single SNT stimulation train can be delivered using the *single SNT stim* button. The pulse train is specified by the settings in the *SNT settings* panel. When delivered using the *single SNT stim* button, the stimulation is delivered immediately.

Saving and loading settings: The parameter settings specified in the *Event timing* and *SNT stimulation* panels can be saved and loaded using the *save/load settings* panel. The *set default* button will save the current parameters as the default (in the file *default_params.mat* in the exp_params_files directory). The *load default* button will load those default parameters from the file. If default parameters have been saved previously, then the GUI will open with them. If default parameters have not been set in this manner, then the GUI will initialize with the parameter values specified in the figure file (see below). The *save* button will save the current settings into a user specified file. The load button will load and apply settings from a file in which settings have been stored previously. The *load* and save *buttons* allow storing multiple trial definitions (and not just a single one, as can be done with the *set default* and *load default* buttons).

The log file: Upon completion of each trial, the program updates the log file. The log file is a MATLAB data file which is saved in the log_files directory (created automatically under the main program directory). The log file name includes the date of creation. Thus, one log file is applicable for an entire day. This means that if a session was stopped and resumed within a single day, even if a new MATLAB session was made, the previously completed trials will populate the *completed stimulus* list. If running multiple experiments on a single day, it is possible to save them in separate log files. This can be accomplished by modifying the name of one such log file (for example by adding a unique suffix, such as '"_a"). Once renamed, the file will not be recognized by the program as a preexisting log file, and a new one will be generated.
The log file itself contains a structure called *log_data* with one element for each trial. It contains the following fields:
*good_trial*: 1 is trial was completed, 0 if it is was aborted
*stim_name*: character string with name of the stimulus
*unique_trial_id*: the unique trial ID of the trial
*now*: the time in which the entry was logged
*params*: a structure with two fields, name and value.
The params structure has 12 elements (parameters). The first 10 correspond to edit boxes in the GUI (their identity should be self evident from the name). The 11th parameter is the sampling rate (scan rate of the I/O board). The 12th is the time of stimulus application since trial start, it is the sum of the ITI (inter trial interval) and the duration of the sound sequence, and is shown by the broken black vertical bar. The 12 parameters are:
1. ITI_edit
2. application_to_stim_delay_edit
3. stim_to_wash_delay_edit
4. wash_start_to_wash_stim_delay_edit
5. wash_stim_to_wash_end_delay_edit
6. VNO_stim_frequency_edit
7. VNO_stim_duration_edit
8. VNO_stim_pulse_width_edit
9. VNO_stim_interphase_delay_edit
10. VNO_stim_amplitude_edit
11. SR
12. apply time

For example, if the first element is:

```
name: 'ITI_edit'
value: 5
```

Then the ITI_edit box had a value of 5, which means that the ITI during that trial was 5 seconds.

Applying more extensive changes to the GUI: The GUI was largely written using the MATLAB GUIDE tool. To modify the layout of the GUI or the callbacks associated with each of the buttons, it is highly advised to use the GUIDE. Typing the command >> guide RUN_VNS_EXP.fig on the MATLAB command prompt will open the GUIDE tool with the interface.

For help or bug reports, please contact Yoram Ben-Shaul
yoramb@ekmd.huji.ac.il