Specification, 2EOAE Program

Version 2.2, 8/22/2005
Doug Keefe

Embed Mathtype fonts:

Revision History:

Version 2.2, 8/2005. Revised the one-buffer mode description.

Version 2.1, 1/2005. Correct calculation of spectral SPL.

Version 2.0, 10/2003. Add adaptive version.of 2EOAE. Add fixed attenuation level option to Record Mode (2). Add 2E acoustic reflex test. Maximum NumLevels=20, NumBuffers=128.

Version 1.61, 7/2003: 2 channel ADC. *[Denis: In record mode, when we ask for* `NumBuffers=16` *buffers, we appear to only get 15 in the file that str stored. We might like to record only 1 buffer in some instances, but then we might get 0 buffers stored. Please look into this. It is happening with Shawn's experiment list for which you are providing new Matlab functions. Also, allow for real L1 and L2 values in the experiment list.]*

Version 1.5, 6/2002: Add Record Mode (Mode=2), run parameters FirFilt and BufferLength, Two-buffer and one-buffer RunModes. V1.51: change number of averages in autolevel from 16 to 8.

Version 1.4, 2/2002: Add phase response measurement and I/O function plots.

Version 1.3, 12/14/2001: Minor corrections to conform with present program structure.

Version 1.2, 5/2/2001: Clarify SSOAE processing; describe I/O function extraction for single frequency bin for transients using frequency-specific stimuli and responses. Added brief description of TFR calculations that differ from those in ref. [6].

I. Introduction

Purpose: [1] Acquire, analyze, store and plot 2EOAE responses for otoacoustic emissions (OAE) of the following types: continuous and transient evoked otoacoustics emissions (EOAE). The continuous types include stimulus frequency otoacoustic emissions (SFOAE) and distortion product otoacoustic emissions (DPOAE). The transient types include a general transient otoacoustic emission (TEOAE), which includes any of the more specific transient types: click-evoked otoacoustic emission (CEOAE), chirp-evoked otoacoustic emission (ChEOAE), and gated-SFOAE and gated DPOAE.

[2] Acquire, analyze, store and plot synchronous-evoked spontaneous otoacoustic emission (SSOAE) response using a simplification of the 2EOAE recording paradigm. This simplified paradigm includes conventional DPOAE measurement based on simultaneous presentation of two buffers.[3] Calculate the time-frequency representation (TFR) of a TEOAE and store, plot with 2-D contour and 3-D graphics, analyze individual trajectories to extract parameters related to onset, sustain and decay, as well as multiple reflections. The user should be able to calculate the TFR using the cone kernel or spectrogram techniques.
[4] Output results from multiple subjects into a data base readable by Excel.

System: Windows compatible sound cards controlled by Windows Audio API software, 24 bit converters. Develop in Borland Delphi-compatible software environment. The CardDeluxe sound card is used. Public domain C code is used for the cone-kernel TFR, by constructing a DLL with Borland C Builder. The BTNRH enhancements of this code (which have been implemented in Matlab) are included such that the TFR has circular (rotational) symmetry within the buffer (corresponding to circular symmetry of the input waveform,) and a digital filter in the cone-kernal lag-time domain based on the use of a Kaiser filter. Code from Numerical Recipes (1992) is used for cubic spline interpolation. Previous to version 1.6, the quadruple-buffer acquisition setting was used in the Windows communication with the driver. In version 1.6, evaluate whether the double-buffer mode works and is stable.

Development Guidelines: The 2EOAE application uses the PHAL Data Acquisition Module. It is assumed that the probe has two loudspeakers and one microphone. There will be parameter switches to control whether the program operates in 2EOAE mode (standard or Record mode, with the latter intended for waveform analysis outside the 2EOAE program using each separate stored buffer), or in some other mode (e.g., Reflex mode, or Iso-level Reflectance mode, Full (iso-level and nonlinear) Reflectance mode (includes 2EOAE also)). The sound-card hardware is controlled using standard Windows calls using a 24-bit sound card in full duplex mode. The default software configuration is for the DACs set to operate at +4 dBu (professional digital audio standard), and the ADCs at -10 dBV (consumer digital audio standard). However, the software configuration should be switchable so that the software can run on sound cards with alternative choices of professional/consumer levels.

II. Buffer Processing

A. 2E Technique

The double-evoked (2E) technique for measuring OAEs has a framework across many different types of OAEs, and uses a common definition of OAE *signal* and *noise* across continuous and transient OAEs. Basic information on 2E measurements is in refs. [1] and [2]. The approach here departs in specifics from those references. For each of two digital to analog converter (DAC) channels, a 2E stimulus buffer is defined based on three elementary buffers S1, S2 and S12, each of length `N`. For channel A, the contents of S1 and S12 are identical, and S2 is zero-filled. For channel B, the contents of S2 and S12 are identical, and S1 is zero-filled. Each DAC channel drives a loudspeaker in the probe, and a composite acoustical stimulus is produced in the ear canal. The probe microphone measures the total acoustical response, comprised of the signals transmitted by the loudspeakers and the response in the ear canal (or coupler).

The 2E response is of length 3N, and is partitioned into three responses P1, P2 and P12, each of length N. P1 is the response due to the two-channel stimuli S1, P2 is the response due to S2, and P12 is the response due to S12. Because of the pair of zero-filled elementary buffers in the two-channel, P1 is due to signal S1 presented through DACA, P2 is due to signal S2 presented through DACB, and P12 is due to signals S1 presented through DACA and S2 simultaneously presented through DACB. The distortion response P_D at the n-th sample is defined by

$$P_D[n] = P_{12}[n] - \left\{ P_1[n] + P_2[n] \right\}. \qquad [2]$$

If the measurement system is linear, then this distortion response is due to the system being tested, i.e., the EOAE response of the ear. The distortion response is called the 2E waveform. If either measurement system channel including loudspeaker has distortion that is independent of the other channel, then this distortion is removed from the 2E waveform. The only system distortion that remains is intermodulation distortion.

Digital filtering and artifact rejection are carried out during data acquisition and averaging based on the entire ADC1 buffer, which has a length of *3N* samples in the present example. The digital filter is stored as a external text file with a default file name (*.FIR) and read in by the program; the default file name input by the program is fhp.fir. Other digital filter names can be specified in the Experiment list (see below).

The above 2E stimulus construction method is adequate for TEOAE stimulus for which the following conditions must be satisfied: [1] each buffer begins and ends with zero amplitude, and [2] each elementary buffer (length N) is sufficiently long that the TEOAE response of the ear decays within the buffer except for synchronous-evoked spontaneous otoacoustic emissions (SSOAEs) (see refs. [3]-[4]).

The 2E stimulus construction is inadequate for continuous EOAEs using sine-tone testing, for which condition [1] is not satisfied. All the sine-tone evoked OAEs should use sine tones with an integral number of periods within each elementary buffer of length N. However, the inclusion in the 2E stimulus of elementary buffers with zero values means that the sine-tone stimulus of each loudspeaker is turned on and off within each 2E period of 3N samples. Coupled with the finite-duration impulse response of the loudspeaker, it follows that the acoustic signal in response to a rectangular-windowed sine tone is not a continuous response, and the ear responds to the transient component of this signal with a TEOAE response. The resolution is to leave the the sine tone on for a sufficiently long duration that the transient decays before measuring the "continuous" response. Because of condition [2] above, the EOAE should decay within one elementary buffer.

The design is to construct a 2E stimulus buffer with multiple repetitions M of each of S1, S2 and S12. For the case such that M equal to 4, the 2E stimulus is

S1 S1 S1 S1 S2 S2 S2 S2 S12 S12 S12 S12

The total length of the 2E buffer with M repetitions is 3MN samples. Digital filtering and artifact rejection is based on this total length.

Suppose the total buffer length is N' (equal to 3MN for a 2E buffer, but this filtering operation is used for any other EOAE technique, e.g., SSOAE discussed below) and the window length is `L'=2L+1` (the filter length is always odd). The digital filtering operation is implemented as a linear convolution so that the filtering of the stream of input total buffers is similar to that achieved by an analog filter at the input. For purposes of filtering, the total buffer to be filtered must be longer than `N'` samples to accommodate the impulse response of the finite impulse response (FIR) filter. Thus it includes the current total buffer and the end of the previous total buffer. Its length is augmented to `L'-1+N'`. The initial `L'-1` samples of this augmented input buffer are loaded with the last `L'-1` samples of the previous total buffer; if this is the filtering operation for the initial total buffer, then the previous buffer is assumed to hold zero values. The next `N'` samples are the current total buffer. This filter buffer is convolved with the impulse response of the filter, which is an array of length `L'`. The linear convolution command `nspdConv` in the Intel Signal Processing Library is used. The output array is returned with a length equal to one less than the sum of the lengths of the two input arrays, or `L'-1+N'+L'-1=2L'+N'-2` samples. The output filtered total buffer must have the same length as the input array `(N')`. This output array includes the `N'` samples beginning at sample number `L'-1` in the output array returned by the linear convolution (sample number `L'-1` is the *L*-th sample in the array, since the initial array index is 0. The last `L'` samples of the original total buffer are copied to serve as the previous total-buffer contents for the filtering of the next total buffer.

The next operation is artifact rejection. If the buffer is valid, then the total buffer must be rotated to remove the group delay of the filter, before the total buffer is decomposed into its elementary sub-buffers. The group delay of the filter is equal to `L=(L'-1)/2` samples, so that the output total-buffer array is left-rotated (towards the origin) by `L` samples. This rotation must follow the completion of the linear convolution, but may either precede or follow artifact rejection. It has been described here as following artifact rejection. Either way, the rotation must occur *after* the `N'` samples of the output of the linear convolution have been selected.

The 2E response buffer in each ADC buffer has the form

P1 P1 P1 P1 P2 P2 P2 P2 P12 P12 P12 P12

Time-domain averaging of `A` 2E buffers during data acquisition means that each elementary response in the above is an average of `A` valid responses. Because of condition [2] above, the first P1 response of the `M` repetitions of P1 contains both a transient and continuous component, with the transient component due to having turned loudspeaker B off at the onset of the first S1 stimulus. Thus, the program should have the option of discarding the first P1 buffer (or the first and additional buffers, as determined by the value of the run parameter `DiscardValue`, which in the case of discarding the first buffer only is set to `1`), and similarly for the first P2 and P12 buffers. `K=M-1` response buffers of P1, P2 and P12 are retained, or 3 each in this example. This approach enables the measurement of a continuous EOAE response using stimuli containing discontinuities.

The number of buffers *K=M-1* ultimately constrains the ability to decompose the total distortion spectrum into signal and noise components. To increase the number of buffers, an ADC repetition parameter `Q` is introduced. Rather than directly averaging valid buffers in a single 2E memory location, a set of `Q` memory locations is defined, each the address of a 2E buffer of length `3MN`. An index `q` cycles through these memory locations from `1` to `Q`, and the next increment is again `1`. Each valid 2E buffer is summed into the current memory location and the index q is incremented to the next memory location. By summing into the current memory location, it is possible to time average A number of times into each of the 3MNQ elementary buffers. For each value of index q, the program should extract M-1 buffers, so that the total number of independent 2E buffers is K=Q*(M-1), each 2E buffer the time average of A valid responses.

For each of the elementary response buffers $m = 1,\ldots,K$, the 2E waveform $p_D^m[n]$ is defined by

$$p_D^m[n] = p_{12}^m[n] - \left\{p_1^m[n] + p_2^m[n]\right\}. \qquad [3]$$

The inputs to this 2E buffer processing section include N, M, A, DiscardValue, and Q.

B. SSOAE Processing

The one-buffer technique for recording EOAEs uses the joint presentation of two stimuli presented individually in DAC1 and DAC2. Whereas the 2EOAE technique uses separate presentations of S1, S2 and S12 and measures a corresponding trio of ADC responses P1, P2 and P12, the one-buffer technique presents only S12 and measures an ADC response P12.

The one-buffer stimulus buffer can still have multiple repetitions M of each of S1, S2 and S12. For the case such that M equal to 4, the 2E stimulus is

S12 S12 S12

The total length of the simplified buffer with M repetitions is MN samples. Digital filtering and artifact rejection is based on this total length. The ADC buffer has the form

P12 P12 P12 P12

A SSOAE measurement is an example that uses the one-buffer technique in which time windowing is used to separate the SSOAE from the stimulus and the shorter-latency CEOAE.

The SSOAE technique differs from the 2E technique in that: (1) it always uses a click file waveform, (2) it returns more than one EOAE response for spectral and waveform plots (the 2E has three sub-buffers but returns only one emission response), (3) it returns one EOAE waveform for TFR analysis (2E also returns one waveform), and (4) the calculations in the SSOAE are different in that only a single stimulus buffer is presented (denoted as S12 because each channel contains a click) and there is only a single response buffer (denoted as P12) . For the purposes of the stopping rule, one EOAE response is calculated during data acquisition so that its signal and noise contributions can be evaluated and made available for use in the stopping rule algorithm. It is also available in real-time plotting. Other calculations take place immediately after data collection has been completed, and before the next run in the Experiment List.

Before SSOAE data collection starts, the following (half)-Hanning window (squared cosine function) that acts over $N_h + 1$ samples (from 0 to N_h) is calculated:

$$w[n] = 0.5 * \left[1 - \cos\left(\pi \frac{n}{N_h} \right) \right]. \qquad [4]$$

This window increases from 0 up to 1 over this sample range. The first half of the window is used to define an onset ramp window, and the second half an offset ramp window. For the onset ramp, i.e., beginning at *0* and ending at *1*, the index *n* should vary between *0* up to N_h. For the offset ramp, i.e., beginning at *1* and ending at *0*, the index *n* should vary by reversing the onset ramp. N_h is the experiment list parameter `ShortRamp` with a default value of 22 samples (corresponding to approximately 1 ms at a 22.050 kHz sample rate.

The first response (*P12*) is the entire window. This is used during data acquisition to show the stimulus response including the click excitation. The second response (*Pd*) is a click-evoked OAE (CEOAE). During data collection, this CEOAE is defined for each valid buffer as follows:

a) The buffer index at which the absolute value of the response is a maximum is defined as the time origin of the click. This index may be different for different valid buffers, although it should tend to be similar.
b) This sample and the N_{click}*-1* samples form the CEOAE buffer that will contain $p_d[n]$ (the system parameter here is `Nclick` with a default value of 512 at the 22050 Hz sample rate). The program should check in the run list that `Nclick` is a multiple of 2, suitable for DFT analysis, and the program should generate an error condition is `Nclick` exceeds `N` (the buffer length of $p_{12}[n]$). A typical value of *`N`* in a SSOAE run is 2048 samples. If the time origin is within the final N_{click} samples of the entire buffer of length *N*, then the CEOAE buffer is effectively circular-rotated so that its end occurs towards the beginning of the entire buffer. This should be managed by appropriate choices of index range. However, it is expected that the time origin will be placed before the final N_{click} samples so that such a circular rotation will not be necessary.
c) The initial response in the click buffer following the click maximum amplitude, which occurs at the first sample of the click buffer, is zeroed. The total number of sample to zero is `ShortDelay,` an experiment list parameter with a default (i.e., typical) value of 44 samples. This is approximately 2 ms at a 22.05 kHz sample rate. These samples are 0 to `ShortDelay-1` in the click buffer.
d) The next section of the click buffer is multiplied by the onset ramp--these are the samples beginning at `ShortDelay` and extending to `ShortDelay` $+ N_h$.
e) The final $N_h + 1$ samples of the click buffer are multipled by the offset ramp so that the very last sample (`Nclick` -1) is set to 0.

All the signal processing in Eqs. [7] through [19] is carried out for this newly defined p_d as before, and the resulting OAE signal and noise are plotted as a CEOAE. The signal and noise thus defined are used in the stopping rules, as needed. In all plots, selection of the `Pd` button plots this CEOAE spectrum.

After all valid buffers have been acquired in the SSOAE run, the program should calculate all intermediate SSOAE results. At this moment in data acquistion (at the end of the SSOAE run and before the next run in the experiment list), the buffers are stored in separate buffer locations in memory. Locations for P12 and Pd are in use, but only the P12 buffers are used in subsequent processing. The average of the P12 response is calculated, and the sample index is determined for which the absolute value of the signal is a maximum. This is the time origin of the averaged response and is used in further SSOAE processing. Optionally, all buffers may be rotated so that the sample corresponding to the time origin is the first sample of each buffer. The second response Pd as defined above is calculated for this averaged response, in which the same time origin is used for each P12 buffer to define a Pd buffer (the CEOAE response). Thus, the new contents of the Pd buffers will be slightly different, in general, than the old contents. These are used to calculate the CEOAE signal (average) and noise (variance) that is stored in the file.

The third response is the SSOAE signal, which is defined based on an additional system parameter N_{ss} (Nss). Typically, Nss is chosen so that the initial 20 ms of the response following the maximum click is zeroed (Nss must not exceed N else the program should generate an error message). Assuming all buffers have been rotated as in the above (else appropriate circular processing of the buffers should be used), a window of length N is constructed such that the initial N_{ss} samples are 0, and the next N_h+1 samples use the onset ramp defined above. The remaining samples in the window have amplitude of 1, except for the final ShortDelay+N_h samples, which are loaded with the offset ramp of duration N_h+1 followed by zeros for ShortDelay-1 samples. This window is multiplied by the (rotated) P12 buffer to form the SSOAE buffer. This window $w_{ss}[n]$ is multiplied by the $p_{12}[n]$, and the product is denoted by the symbol $p_{ssoae}[n]$. All the signal processing in Eqs. [7] through [19] is carried out for this newly defined p_{ssoae} as before, and the resulting SSOAE signal and noise are plotted and labelled as a SSOAE. Internally, the program might use the P1 buffers since these are otherwise unused. This SSOAE signal is derived from the late part of the click response, in which the original CEOAE response is assumed to have become small and all that remains is the synchronous-evoked SOAE response. In this example, the SSOAE is calculated using a 2048 point DFT and the CEOAE is calculated using a 512 point DFT.

The fourth response is the TFR response, which uses a longer non-zero window to create a buffer for subsequent uses in TFR analysis. The distinguishing feature is that more of the original waveform in retained than in the other windowed responses. This TFR response is calculated by multiplying the rotated P12 buffer and a TFR window defined as follows. The initial ShortDelay samples are 0. The remaining window values are 1 except for an offset ramp and zero portion identical to that used in the SSOAE: the final ShortDelay+N_h samples are loaded with the offset ramp of duration N_h+1 followed by zeros for ShortDelay-1 samples. Thus, there are ShortDelay-1 zeros preceding and following the sample at which the original stimulus response had maximum amplitude. The intent is to remove the energy associated with the click stimulus.

Thus, the default values are such that the response is zeroed for 0-2 ms following the maximum (absolute value of the) click, and the onset ramp transitions between 0 and 1 between 2-3 ms following the maximum click. The same window acts on the response before the maximum click.

This TFR window multiplies each P12 buffer response to form the distortion waveform, $p_d[n] = w[n]* p_{12}[n]$. Next, the calibration and calculations of signal and noise in Eqs. [7] through [19] are carried out for p_d as defined above for this SSOAE and p_{12}, but not for p_1 and p_2, as they are undefined here. The average $p_d[n]$ and $p_{12}[n]$ buffers are rotated so that the initial sample of each buffer corresponds to the sample of the maximum click (as determined earlier), so that time is measured from an origin at the peak of the click response. These rotated buffers now contain zeros at the beginning and end of the buffer. These responses may be analyzed using the TFR and plotted.

There are details of how the GUI shows the CEOAE/SSOAE/TFR-SSOAE waveforms and spectral responses that differ from the standard 2EOAE plotting GUI. The SSOAE plots do not explicitly contain P1 or P2 controls, but they do contain a P12 control for the original stimulus response (of length *N*), a Pd control for the CEOAE (of length `Nclick`), a control for the SSOAE (of length *N*), and a control for the windowed response transmitted to the TFR (of length *N*). The P1 and P2 controls should be used for these latter controls, with a simple re-labelling of the string labels in an appropriate manner. The additional run parameters in a SSOAE run are `ShortDelay`, `ShortRamp`, `Nclick`, and `Nss`.

C. Record Mode

The purpose of Record Mode is to record long sequences of responses and save these responses for subsequent analysis by another program. It differs from the 2E mode described above and is selected for use in an Experiment List with global parameter `Mode 2`. There is a new run parameter in Record mode not present in 2E mode, `BufferLength`, which is an integer greater than or equal to the value of `StimLength` (see Section IV.C. for Experiment List format). As in 2E mode, `StimLength` governs the length of the DFT and must be equal to a power of 2 (up to 16384). The `BufferLength` can exceed the `StimLength` to attain recorded durations of arbitrary length. The `StimLength` value in Record mode is used solely to define the length of the DFT used in auto-leveling and any other spectral calculations, and provides a sufficiently good means of setting spectral levels.

The **Record** mode does not support the I/O function and TFR capabilities of the other 2E modes. It does not support fractional octave averaging. The file format is similar for the 2E mode (`Mode=1`) and the Record mode (`Mode=2`). An example of an Experiment List for Record mode is shown below. All runs in Record mode are of `Run T, Subtype R` with externally specified file names.

```
2Eexplist
SampleRate                    22050
AR    1                        500
Mode 2
NumRuns  2
CircShift  0
Run   T    OS ; overshoot 1
Subtype R
StoppingRule                     2
File1 noise1.2es
File2 probeStart.2es
NumFreqs   1
NumLevels                        6
```

```
FreqLevel1             0 70 60 50 40 30 20
FreqLevel2             0 20 20 20 20 20 20
StimLength                    16384
BufferLength                  22050
StimReps                        1
NumBuffers                      8
DiscardValue                    0

NumRuns  1
Run   T    OS ; overshoot 2
Subtype R
StoppingRule                    2
File1 noise1.2es
File2 probeMiddle.2es
NumFreqs   1
NumLevels                       6
FreqLevel1             0 70 60 50 40 30 20
FreqLevel2             0 20 20 20 20 20 20
StimLength                    16384
BufferLength                  22050
StimReps                        1
NumBuffers                      8
DiscardValue                    0
```

Record mode provides the capability of recording two channels, each on separate ADCs. In Record mode, there is an additional global parameter `ADCchannels`, which has the default value 1 for single-channel ADC on ADC1, and 2 for dual-channel ADC on ADC1 and ADC2. Calibration is performed only in terms of responses measured on ADC1, but the responses on ADC2 should be stored and digitized in the same manner, but without digital filtering or artifact rejection.

This example shows the use of the `CircShift` global parameters, which performs a circular rotation of the 2E buffer waveform. Its default value is -256 samples, which has the effect of moving the origin of the 2E buffer 256/22.05=11.6 ms to the right. This default may be a good idea for continuous tones because with `CircShift` of 0, the beginning of the 2E buffer actually contains the end of the P12 buffer due to unequal hardware delays in the DAC and ADC channels. This default is often a bad idea because the first and last samples of the original buffer response are discontinuous due to noise and, after a circular rotation, there is a discontinuity in the recorded P1 waveform inasmuch as P1 begins the 2E buffer. The `CircShift` command needs to be placed immediately after either the `NumRuns` or `Atten0` parameters to be properly read. If not provided, it defaults to -256

D. Two-Buffer Mode

[In version 1.6, Stop testing of this Mode pending further changes to specification]

One purpose of the Two-buffer mode is to record SFOAEs, in which the two conditions are an unsuppressed tonal probe signal (a sine tone in a SFOAE), and a suppressed probe signal due to the presence of a suppressor signal. They correspond to the first and second stimuli in the 2E technique. If the intent is to measure the suppression in the first signal due to the presence of the second signal, and if the first signal is tonal (hence, its output is monitored in a single frequency bin), then it is unnecessary to measure the system response to the second signal alone. In terms of the above notation used to describe the 2E technique, the stimulus buffer includes S1 and S12, and the response buffer includes two responses P1 and P12.

A particular Run in an Experiment List is processed in Two-buffer mode if a parameter `RunMode` is present within the set of run parameters, and if it has the value 2. The default value of `RunMode` is 3, corresponding to the normal three-buffers mode of the 2E technique. When this default is used, the `RunMode` parameter need not be present in the Experiment list. If multiple runs in an experiment list should use two-buffer mode, then the parameter line `RunMode 2` must be present in each run list.

The two-buffer response is of length `2N`, and is partitioned into two responses P1 and P12, each of length `N`. P1 is the response due to the two-channel stimuli S1 and P12 is the response due to S12 (as in the 2E technique). Because of the pair of zero-filled elementary buffers in the two-channel, P1 is due to signal S1 presented through DACA, and P12 is due to signals S1 presented through DACA and S2 simultaneously presented through DACB. The two-buffer distortion response P_D at the n-th sample is defined by

$$P_D[n] = P_1[n] - P_{12}[n]. \qquad [5]$$

Elsewhere in the program (plotting, output files), this distortion response is used instead of the 2E distortion response in Eq. (1) as the measured EOAE. This “distortion response” does not have the presence of the P2 stimulus subtracted out, so it is not a pure distortion waveform. Any TFR applications should not use the two-buffer (or single-buffer) run mode (except for SSOAE measurements for which TFRs are well defined).

Either continuous or transient runs may be used except that stimulus S2 is used only in conjunction with S1 to form S12 in the stimulus buffer. Similar to the normal 2E technique, it is necessary to construct a two-buffer stimulus with multiple repetitions `M` of each of S1 and S12. For the case such that `M` equal to 4, the total two-buffer stimulus is

S1 S1 S1 S1 S12 S12 S12 S12

The total response length with `M` repetitions is `2MN` samples. Digital filtering and artifact rejection is based on this total length.

The two-buffer response buffer in the ADC has the form

P1 P1 P1 P1 P12 P12 P12 P12

As in the normal 2E technique, the program should have the option of discarding the first P1 buffer (or the first and additional buffers, as determined by the value of the run parameter `DiscardValue`, which in the case of discarding the first buffer only is set to `1`), and similarly for the first P12 buffers. `K=M-1` response buffers of P1 and P12 are retained, or 3 each in this example. This approach enables the measurement of a continuous EOAE response using stimuli containing discontinuities.

The number of buffers *K*=*M-1* ultimately constrains the ability to decompose the total suppression spectrum into signal and noise components. To increase the number of buffers, an ADC repetition parameter `Q` is introduced in the two-buffer mode analogous to its use in the normal three-buffer 2E run mode. For each of the elementary response buffers $m = 1, \ldots, K$, the distortion waveform $p_D^m[n]$ is defined by

$$p_D^m[n] = p_1^m[n] - p_{12}^m[n]. \qquad [6]$$

This also is analogous to the 2E run mode. All plotting routines should simply make the P2 option (or SPL2, etc.) inactive for data recorded in two-buffer mode. The output file can zero pad the P2 waveform and spectral elements or delete them, whichever is easier for file input/output operations.

E. One-Buffer Mode

-

A conventional DPOAE measurement is an example of the use of the one-buffer mode introduced above in describing the SSOAE measurement, in which additional parameters in the Experiment List instruct the program to search for the DPOAE response at a specific frequency bin (2f1-f2). Except for the SSOAE use described above, the one-buffer run mode is invoked within each run by providing a parameter line `RunMode 1`, with usage similar to that of two-buffer run mode. One application of the one-buffer mode is to measure DPOAEs in the standard manner and another application is to record a response to an arbitrary stimulus. It should be possible to use this one-buffer mode with the stimulus presented at a fixed attenuation level by each DAC.

The one-buffer stimulus buffer can have multiple repetitions *M* of each of S1, S2 and S12. For the case such that M equal to 4, the one-buffer stimulus is

S12 S12 S12 S12

The total length of the one-buffer with *M* repetitions is *MN* samples. Digital filtering and artifact rejection, if used, are based on this total length. The ADC one-buffer has the form

P12 P12 P12 P12

No buffers need to be discarded with this type of OAE measurement. In this run mode, the waveforms and spectra of P1, P2, and PD are undefined. In DPOAE measurements, the stimulus and OAE responses are in separate frequency bins, so that the distortion is evaluated at the distortion frequencies of P12. The data from each of the *M* multiple repetitions should be stored in a Matlab readable file. This file should also contain the stimulus file name(s) and the attenuation level at which the stimulus was presented from the DAC. In some cases only DAC1 is used; the experiment list can either indicate that only DAC1 is used or else a stimulus file with zero values can be presented through DAC2.

F. Fixed-attenuation level record mode

In Record mode (Mode 2), the above discussion assumes that s1 and s2 are calibrated to desired levels for the one-, two-, and three-buffer stimulus sets. If the run parameter `RunCalib 1` is used, then the Record mode should calibration in the standard manner; i.e., `RunCalib 1` is the default. If the run parameter `RunCalib 0` is used, then the levels in the `FreqLevel1` and `FreqLevel2` lines are interpreted as attenuation levels (in dB) relative to 0 dB corresponding to the full-scale amplitude. In this case, each level in the list of `FreqLevel1` and `FreqLevel2` should be less than or equal to 0. The initial field in the `FreqLevel1` and `FreqLevel2` lists should be a 0, as above, but this refers to the absence of frequency-specific information rather than an instruction to use 0 dB attenuation. This is similar to their use in the adaptive mode of the program. The program runtime is decreased when `RunCalib 0` is selected, because the calibration part of data collection and analysis is removed.

III. SPL and Signal and Noise Extraction

A. Conversion to SPL

The units of the time-domain waveforms are in the units of the ADC. Variables in the frequency domain, or the time-frequency domain, are converted into units of pressure, or sound pressure level (SPL). The ADC units are converted into voltage based on the absolute voltage range of the ADC and the number of bits (this is applied to the buffer waveform during data collection and any waveforms are stored as voltage waveforms), and the microphone sensitivity is applied as a ratio of voltage to acoustic pressure in units of *V/Pa* (or *mV/Pa).* Conversion to SPL is performed using the reference pressure level in *Pa*.

In some probes the microphone sensitivity may be regarded as a constant over frequency, but in the general case, the microphone sensitivity varies with pressure. In initial implementation, the phase response of the microphone is neglected so that the only quantity of interest is how the magnitude of the output voltage of the microphone pre-amplifier varies with the magnitude of the pressure at the entry to the probe microphone at various frequencies. This is stored as the microphone sensitivity level $L_{mic}(f) = 20\log_{10}(S_{mic}(f))$, where the microphone sensitivity $S_{mic}(f)$ is expressed in terms of *mV/Pa.* Thus, a microphone sensitivity of 5000 *mV/Pa* has a sensitivity level of 74 dB. A text file `MIC.TXT` stores the microphone calibration in which the first line contains a string label for the probe microphone and every other line consists of a pair of frequency (in Hz) and microphone sensitivity level values, e.g.,

```
MyProbe
0    62
250  62
500  70
1000 76.5
8000 90
10000 75
20000 75
```

By convention, the initial frequency is 0 Hz and the final frequency is 20000 Hz. There may be an arbitrary number of pairs of values in between, but a second convention is that the microphone sensitivity level at 0 Hz is set equal to that for the next highest frequency (62 dB at 250 Hz in this example), the level at 20000 Hz is set equal to that for the next lowest frequency (75 dB at 10000 Hz in this example). In practice, this file is created by calibrating the microphone from 250-10000 Hz, while the lowest and highest frequencies are included to prevent any values outside the measurement range from having unconstrained values and so that the spline analysis is well behaved.

During program initialization, the program should read this microphone calibration file, and convert to sensitivities ($S_{mic} = 10^{L_{mic}/20}$) and construct a cubic spline table using the routine `spline` from Numerical Recipes (NR). In this table construction, the array of frequencies has the role of the input variable `x` and the array of sensitivities has the role of the input variable `y.` The cubic spline table should use "natural" boundary conditions as defined in NR. Once the table is constructed, when the microphone sensitivity value is desired at any particular frequency (corresponding to each DFT bin), the routine `splint` from NR is used for cubic spline interpolation.

In some situations (in data analysis, but never during data collection), the voltage waveforms should be converted to pressure waveforms based on the microphone sensitivity. If the microphone sensitivity is a constant across frequency (as for the Etymotic ER10C), then a single number can be multiplied to change voltage to pressure. The desired output unit is in *mPa=0.001 Pa*. If the microphone sensitivity varies across frequency, then the calibration is first applied in the frequency domain based on the DFT of the voltage waveform $V[k]$ at the k-th frequency bin, and the microphone sensitivity $S_{mic}[k]$ as calculated above. The pressure waveform $p[n]$ at the n-th time step is defined as the inverse DFT of the product of the complex voltage spectrum and microphone sensitivity:

$$p[n] = DFT^{-1}\{V[k] * S_{mic}[k]\} \quad [7]$$

The normalization can be verified first by using a constant microphone sensitivity of unity, and checking that $p[n]$ is identical to $v[n]$, the original voltage waveform; second, by using a constant microphone sensitivity equal to that for the Etymotic ER10C in the +40 switch position (5000 mV/Pa=5 mV/mPa) and checking that the pressure waveforms is 0.2 times the original voltage waveform.

B. Signal and Noise Extraction

Each response buffer contains both signal and noise. Because EOAEs are inherently weak acoustic signals arising from site(s) within the cochlea, it is essential to extract the signal component of the total distortion waveform from the noise component. This depends on the calculation of the coherent and fluctuating parts of the 2E waveform, and is performed in the frequency domain. The implementation is in real time, meaning that the signal to noise ratio (SNR) is calculated as data acquisition is in progress. This allows a stopping rule for data acquisition to use this SNR.

1. Calculate the N-sample discrete Fourier transform (DFT) of $p_1[n], p_2[n], p_{12}[n]$ and $p_D[n]$ using `RealFftNip` in the Intel library. The above are mean waveform responses after averaging across multiple buffers and/or repetitions. In each case, before the DFT is calculated, subtract the average buffer response from each sample to remove the DC response. This can be included as the initial part of the call to the DFT. The complex spectra of P1, P2, P12 and PD are expressed at frequency bin k as $P_1[k]$, $P_2[k]$, $P_{12}[k]$ and $P_D[k]$, respectively, and the function returns complex values from *k=0* up to *N/2*. This is performed for all copies (*m=1,…,K*) of each waveform. In equations, lower-case letters denote time-domain waveforms and upper-case letters denote frequency-domain (complex) spectra. Because the response is highpass filtered during artifact rejection, which attenuates the DC response at *k=0*, there is no need to subtract the mean waveform response before the fast Fourier transform (FFT). The convention on all spectra is that variables such as $P_D^m[k]$ denote the response in the m-th elementary buffer and k-th frequency bin. It is assumed that the values after the FFT are converted from ADC units to voltage and thence to pressure units (Pa) using the cubic spline interpolation table, but this transformation is not otherwise explicit in the notation in this specification. In the case of time-frequency analysis, the calibration is performed on each column of spectral data at each time slice.
For sample rate *SR*, the center frequency of the k-th bin is $f_k = SR*(k-1)/N$. The realizable range of frequencies is up to one-half the sample rate, which corresponds to bin *k=N/2*. The frequency bandwidth of each DFT bin indexed by *k* is $\Delta f = SR/N$.

2. The coherent sum (or average complex spectrum) is defined by

$$< P_D[k] >= \frac{1}{K}\sum_{m=1}^{K} P_D^m[k] \qquad [8]$$

for each frequency index *k*. The Intel function *Add* may be used as it is defined for complex inputs. The coherent energy of the signal (the "EOAE signal energy") is defined by

$$\left|\left\langle P_D[k]\right\rangle\right|^2 = \frac{1}{K^2}\left|\sum_{m=1}^{K} P_D^m[k]\right|^2 . \qquad [9]$$

3. The (real) variance in the complex signal is defined as

$$Var\left(P_D[k]\right) = \frac{\sum_{m=1}^{K}\left|P_D^m[k]\right|^2 - K\left|\left\langle P_D[k]\right\rangle\right|^2}{K-1} \qquad [10]$$

After multiplication by *K/(K-1)* it can be shown that the above variance equals the incoherent (noise) energy from the nonlinear coherence technique. The variance definition is more straightforward to implement using the Intel library functions such as `bPowerSpectr` (for the squared magnitude of a complex vector of numbers, useful in the numerator), etc. The noise energy $|P_N[k]|^2$ in the k-th spectral bin is defined as the standard error of the mean:

$$\left|P_N[k]\right|^2 = \frac{Var\left(P_D[k]\right)}{K} \qquad [11]$$

The OAE distortion or "signal energy" $\left|P_{oae}[k]\right|^2$ in the k-th spectral bin is

$$\left|P_{oae}[k]\right|^2 = \left|\left\langle P_D[k]\right\rangle\right|^2. \qquad [12]$$

The energy spectral density in each bin is $\left|P_{oae}[k]\right|^2 / \Delta f$. That is, the energy spectral density of each bin is that quantity which when integrated over frequencies within the bandwidth of that bin is equal to the energy $\left|P_{oae}[k]\right|^2$. This is a discrete-frequency analogy to the definition of the energy spectral density in a bandwidth Δf .
Intel vector operations can be used for each of the above two equations.

4. Defining the reference pressure as $P_{ref} = 0.0002$ dyne/sq cm (CGS units), the spectral sound pressure levels of the noise and OAE defined from bins *k=1* up to *N/2-1* are

$$\begin{aligned} SPL_N[k] &= 10\log\left(\frac{2}{N^2}\frac{\left|P_N[k]\right|^2/\Delta f}{P_{ref}^2/1}\right), \\ SPL_{oae}[k] &= 10\log\left(\frac{2}{N^2}\frac{\left|P_{oae}[k]\right|^2/\Delta f}{P_{ref}^2/1}\right). \end{aligned} \qquad [13]$$

The reference energy spectral density is the signal energy of P_{ref}^2 measured in a 1-Hz bandwidth, or $P_{ref}^2/1$. The k=0 bin is excluded to prevent singularities in the logarithm function, and the *k=N/2* bin is excluded because the normalization differs. All frequency-domain plots are based on the frequency bin index *k=1* to *N/2-1*, although some plot may be restricted to some sub-range of that. Similarly, the SPLs based on the trio of 2E responses are defined by

$$\begin{aligned} SPL_1[k] &= 10\log\left(\frac{2}{N^2}\frac{\left|P_1[k]\right|^2/\Delta f}{P_{ref}^2}\right), \\ SPL_2[k] &= 10\log\left(\frac{2}{N^2}\frac{\left|P_2[k]\right|^2/\Delta f}{P_{ref}^2}\right), \\ SPL_{12}[k] &= 10\log\left(\frac{2}{N^2}\frac{\left|P_{12}[k]\right|^2/\Delta f}{P_{ref}^2}\right). \end{aligned} \qquad [14]$$

The reference level of each of these spectral SPLs is dB *re:* (2e^{-5} Pa)^2/Hz. The total SPLs of the 2E responses are

$$\begin{aligned}
SPL_{1,tot} &= 10\log\left(\frac{2}{N^2 P_{ref}{}^2}\left(\sum_{k=1}^{N/2-1}\left|P_1[k]\right|^2 + \tfrac{1}{2}\left|P_1[N/2]\right|^2\right)\right),\\
SPL_{2,tot} &= 10\log\left(\frac{2}{N^2 P_{ref}{}^2}\left(\sum_{k=1}^{N/2-1}\left|P_2[k]\right|^2 + \tfrac{1}{2}\left|P_2[N/2]\right|^2\right)\right), \qquad [15]\\
SPL_{12,tot} &= 10\log\left(\frac{2}{N^2 P_{ref}{}^2}\left(\sum_{k=1}^{N/2-1}\left|P_{12}[k]\right|^2 + \tfrac{1}{2}\left|P_{12}[N/2]\right|^2\right)\right).
\end{aligned}$$

The total SPL is the sum of the energy across frequency, so it involves a product of the energy spectral level in each bin and the bandwidth Δf of the bin.

The signal to noise ratio (in dB) is expressed as a spectrum by

$$SNR[k] = SPL_{oae}[k] - SPL_N[k]. \qquad [16]$$

The total noise and signal SPLs are

$$\begin{aligned}
SPL_{N,tot} &= 10\log\left(\frac{2}{N^2 P_{ref}{}^2}\left(\sum_{k=1}^{N/2-1}\left|P_N[k]\right|^2 + \tfrac{1}{2}\left|P_N[N/2]\right|^2\right)\right),\\
& \qquad [19]\\
SPL_{oae,tot} &= 10\log\left(\frac{2}{N^2 P_{ref}{}^2}\left(\sum_{k=1}^{N/2-1}\left|P_{oae}[k]\right|^2 + \tfrac{1}{2}\left|P_{oae}[N/2]\right|^2\right)\right).
\end{aligned}$$

Analogous spectral and total SPLs are defined for the primary signals with $P_D[k]$ replaced by $P_1[k], P_2[k]$ or $P_{12}[k]$, as desired, and averaging over the K independent measurements. In practice, SPL1 and SPL2 for the signal averages of $P_1[k]$ and $P_2[k]$, respectively, are used by the program.

5. In implementation, the desired approach is to calculate the SPL's in real time based on all valid buffers acquired to date, in chunks of the number of buffers transferred from each ADC buffer returned from the DSP to the computer. Then, a stopping rule is utilized based on the OAE SPL and the SNR at a particular bin or across a number of bins, depending on the type of EOAE that is measured. The stopping rule has tests joined by OR connectives (except for MinValidBuffers discussed later) and six input parameters, which are defined as part of a test, as follows:

- Index at which to evaluate noise floor (code as -1 to use total noise SPL),
- Threshold noise SPL.
- Threshold SNR
- MinValidBuffers
- MaxValidBuffers
- MaxTotalBuffers

The last condition is to fix an upper limit on overall measurement time. To implement this, it is necessary to maintain running sums across the K valid buffers obtained to date in order to calculate the signal and noise energies.

All stopping rules are stored in a file and identified by a RuleNumber. The RuleNumber is coded into the Experiment List to select a particular stopping rule. The format is below (square brackets indicate field that needs data entered):

- `Rule    [RuleNumber]`
- `Index   [Int]`
- `ThresholdNoiseSPL    [Int]  [Real]`
- `ThresholdSNR        [Int]   [Real]`
- `MinValidBuffers       [Int]  [Int]`
- `MaxValidBuffers       [Int]  [Int]`
- `MaxTotalBuffers   [Int]   [Int]`

The first integer field in the last four rows is coded as a 1 if the type of rule is included in the stopping rule test, or 0 if it is not. The integer in the `Index` row is a 1 if the threshold rules are applied at a single bin obtained in the Experiment List or -1 if applied across frequency (as total SPL for noise in the Transient class). If the stopping rule is based solely on the number of valid buffers, then `MaxValidBuffers` should be equal to `MinValidBuffers`. If the stopping rule includes an SPL or SNR test, then `MinValidBuffers` must be satisfied before any of the other OR conditions evaluate to true; i.e., `MinValidBuffers` is AND-ed with the output of the OR.

6. The phase response in the DFT of the time-average of each pressure, $P_1[k]$, $P_2[k]$, $P_{12}[k]$ and $P_D[k]$, respectively, should be converted to radians and stored.

IV. 2EOAE Mode

The two broad classes of 2E OAEs are Continuous and Transient OAEs. The Continuous OAEs include

- SFOAE, stimulus frequency OAE, in which a sine tone evokes an EOAE in a single bin,
- DPOAE, distortion product OAE, in which a pair of sine tones, each at a different frequency f1 and f2, evokes DPOAEs at various bins including 2f1-f2, and SFOAEs at f1 and f2.

The Transient OAEs include the use of any general transient signal to evoke OAEs, but in particular include

- CEOAE, click-evoked OAE, in which a short-duration stimulus (the "click", usually defined as the impulse response of a FIR digital filter) evokes an OAE at "all" the frequencies in the measurement range.
- ChEOAE, chirp-evoked OAE, in which a chirp substitutes for the click. The response is "dechirped" to produce an equivalent click response. Higher energy stimuli

(producing more robust emissions) can be used in ChEOAE than CEOAE because the stimulus energy is spread out in time.

- Gated sine-tone emissions in which onset and offset phenomena associated with SFOAE, and DPOAE signals can be studied.

Transient OAEs are particularly suited for time-frequency representations (TFR). Continuous stimuli are created within the program, and transient stimuli are stored as binary files of type 2ES. A program configuration parameter is the default directory for stimulus files. Transient stimuli may or may not have a header. If they have a header, then the parameters defining a stimulus file, which are written to a structure header, are as follows:

- Magic number defining file as type 2ES
- *T* for transient stimulus file
- Length *N* of stimulus, which is the length of the elementary buffer in the 2E technique
- Sample rate *SR* (Hz)
- Transient subtype: *S* for gated sine tone, *C* for click, *H* for chirp, *T* for other transient, *Y* for sYnchronous spontaneous otoacoustic emission, and *F* for a transient with a frequency-specific response
- *StimStart* and *StimEnd* as sample numbers (convention throughout is that sample numbers start from *0*) defining the beginning and end of the stimulus. This provides the range of the non-zero part of the stimulus.
- If subtype *S* or *F*, the frequency bin index *k* for one sine tone at the fundamental sine-tone frequency based on the stimulus length and sample rate.

All stimulus files should have a peak amplitude $PEAK = 2^{23} - 1$ and be stored as reals of type DOUBLE. This peak amplitude is appropriate as a maximum value not to be exceeded for any 24 bit system, but values are stored as reals to enable arbitrary attenuation factors (whether by software or digital attenuator). If the transient stimulus file has no header, then it is stored as array of ANSI standard type double (at BTNRH, most stimulus files are created using Matlab without a header—its standard binary format is compatible with the 2EOAE program) The actual arrays sent to the DACs must be in integer format, via a final rounding operation.

A. Sine Tone Generation

For continuous EOAE tests, the program needs to generate sine tones. Suppose the desired frequency is *f* (e.g., *f=1600* Hz). For an elementary buffer length *N* and sample period *T* (this is the time in seconds for one sample, equal to the inverse of the sample rate in Hz), the desired frequency is present in the spectral bin *k* defined by

$$k = round(fNT) \qquad [20]$$

The actual center frequency f_k of the k-th spectral bin is

$$f_k = \frac{k}{NT} \qquad [21]$$

For a sample rate of 32000 Hz, the index *k=26* for *f=1600* Hz, and the center frequency is *1625* Hz. It is impossible to produce a continuous tone with a frequency of 1600 Hz using a 512 sample buffer; the closest one can come is a sine to of 1625 Hz. The index *k=26* means that 26 complete periods of the sine tone are completed within 512 samples. At the user level, the response is labelled as the *1600* Hz response, but at the machine level, it is indexed by *N*, *T* and *k*.

For single sine tones, it is sufficient to take the phase as zero, and the sine tone is generated at sample index *n* from *0* to *N-1* by

$$s[n] = PEAK * \sin(2\pi nk / N). \qquad [22]$$

B. Transient Stimulus Files

Transient stimuli are created outside of the program. They will be stored as IEEE doubles, in a file with *N* elements, including zero padding. The program should have a menu item to read in such a file, and query for the data fields in which to create a header. The plan is to use Matlab to create these files.

C. Experiment List

The Experiment List is a text file that controls the selection of stimulus files and levels used for each of the two channels in a 2EOAE test. A variant of the Experiment List will be used in the other test modes. The Experiment List contains one or more Run Lists. A run corresponds to a single type of stimulus, e.g., the selection of a particular pair of frequencies or a particular click stimulus. It also contains the buffer processing parameters and one or more pairs of levels at which to set the stimuli. These levels are SPLs at a particular bin for Continuous 2E stimuli or total SPLs for Transient 2E stimuli (except for gated and other windowed sine tones which are also set at a particular bin because of the frequency-specific nature of the stimuli and related response). The global parameters are fixed for all runs and are defined at the beginning of the Experiment List:

- `2Eexplist`
- `SampleRate [int] %24000 or 32000 allowed (depends on sound card)`
- `AR [int] [int] % AR 1 30000`

- `Mode [1` for 2E, 2 for Record Mode`]   % Mode 1`
- `NumRuns     [int]  %in range from 1-12`
- `Atten0 [int] % initial DAC attenuation factor is 0.01`
- `ADCchannels [int] % default 1, else 2 for two channels`

AR stands for artifact rejection, with the first field coded to a 1 if artifact rejection is used and otherwise is a 0, and the second field the initial AR threshold in ADC units (the user should be able to reset this value in the GUI). The initial DAC attenuation factor `Atten0` is defined below as the quantity a_0 in Eq. [23], and must be greater than 0 and less than or equal to 1.

Each run has a keyword `Run` with a first string of 'C' for continuous 2E or 'T' for transient 2E measurement (also including SSOAEs even though these are not measured using the 2E technique), and a second descriptive string field `STR1.` Other parameters defining a run are as follows:

- `Subtype [S,D,C,H,T, Y,F,` or `R]`
- `StoppingRule [number]`
- `File1           % transient only`
- `File2           % transient only`
- `NumFreqs`
- `NumLevels % in range from 1 to 16`
- `FreqLevel1`
- `FreqLevel2`
- `StimLength N % in range from 256 to 16384`
- `BufferLength NB % Mode 2 only, >= StimLength`
- `StimReps M % in range from 1 to 12`
- `NumBuffers Q % in range from 1 to 32`

- `DiscardValue [0=` no discard first buffer, `1=` discard first buffer, >1 to indicate number of buffers to discard but less than `M]`

- `FirFilt  fname.fir  % adopt this FIR filter as new default`

If a continuous run of subtypes *S*, or *D*, there are frequency parameters defining a run as well. A run of subtype *Y* indicates a measurement of sYnchronous-evoked otoacoustics emissions (SSOAE), which does not use the 2E buffer processing. A run of subtype *F* indicates a measurement of a frequency-specific response in an experiment using a run of type 'T'; if the run is of type 'C' (i.e., continuous), a sub-type of *F* should generate an error message.

The `FirFilt` parameter is optional. If absent, then the program uses the default digital filter `fhp.fir`. If present, then the Experiment list uses the new digital filter file specified in the current and subsequent runs in the Experiment List (unless a subsequent Run has another `FirFilt` parameter). This enables the user to change the digital filter design for some runs within the Experiment List. The program should not artificially limit filter length, although the program may not run properly on the computer if the length is too long.

An example of an Experiment List with one run sub-list for continuous OAE of subtype *D* (distortion product emission) and one run sub-list for a transient OAE of type *T* (click-evoked) is given below

```
      2Eexplist
      SampleRate                     22050
      AR    1                        2000         % may not be realistic value
      Mode [1 for 2E]
NumRuns     2
      Atten0  0.01
      Run   C                         DP   % class is Continuous, string is DP
      Subtype                         D
      StoppingRule                    2
      NumFreqs   4
      NumLevels                       6
      FreqLevel1            1600 75 65 55 45 35 25
      FreqLevel2            2000 70 60 50 40 30 20
      FreqLevel1            3200 75 65 55 45 35 25
      FreqLevel2            4000 75 65 55 44 32 10
      FreqLevel1            6400 75 65 55 45 35 25
      FreqLevel2            8000 70 60 50 40 30 20
      FreqLevel1             800 75 65 55 45 35 25
      FreqLevel2            1000 70 60 50 40 30 20
      StimLength                     512
      StimReps                        8
      NumBuffers                      10
      DiscardValue                    1

      Run   T                        CEOAE                   % Transient class
      Subtype                         C
      StoppingRule                    1
      File1 click1.2es        % comments out here
      File2 click2.2es                %
      NumFreqs   1
      NumLevels                       5
      FreqLevel1               0 70 60 50 40 30 ; 0 means no frequency info
      FreqLevel2               0 70 60 50 40 30
      StimLength                     512
      StimReps                        1
      NumBuffers                      20
      DiscardValue                    0
```

The first run uses 5 sets of primary frequencies (`NumFreqs`), and the desired frequency is the first field in the `FreqLevel1` and `FreqLevel2` rows. The remaining fields in each row are the desired test levels for each tone (dB SPL), totaling 6 levels per frequency (`NumLevels`). The order of data acquisition is selection of the first pair of frequencies, generating thestimuli, acquisition of data at each successive level, and then selection of the next pair of frequencies, etc.

The stopping rule number identifies which of the pre-stored stopping rule parameter sets should be used in the run. These stopping-rule parameter sets are stored in a separate configuration file (text file format to be determined) and these values are stored in the binary output file. For a continuous EOAE, the stopping rule may include the noise level in a particular bin. If SFOAE, the bin is the bin of the sine tone frequency. If DPOAE, the bin is at 2k1-k2, where k1 and k2 are the bins containing the desired frequencies f1 and f2. The second run is a transient click-evoked test. For any transient test, `NumFreqs` should be equal to 1, as there is only a single pair of stimulus files (although "Freqs" does not have a meaning of frequency for this stimulus type, there shouldn't be any problem in retaining the name within `NumFreqs` and `FreqLevel*`). The first fields of `FreqLevel1` and `FreqLevel2` are coded as a 0, meaning that the frequency is undefined. (For the case of a transient test combining a click in channel 1 with a sine tone in channel 2, the parameters would be `NumFreqs`=1, and the first field of `FreqLevel1` would be the sine-tone frequency, which would be used in plotting to find the FFT bin number of the sine tone. A related example using a transient with frequency-specific information is presented below). The second and subsequent fields of `FreqLevel1` and `FreqLevel2` provide the test levels as in the first run. The program should check that the sample rate in the transient stimulus file header agrees with the sample rate of the Experiment List.

The second run uses only one repetition of the stimulus in the DAC buffer, and DiscardValue is set to 0 because the first (and only) responses in P1, P2 and P12 are not discarded.

The run list below shows the case of a transient run with frequency-specific information (subtype of `F`). In particular, this is a windowed DPOAE experiment, in which each stimulus file contains a windowed version of a sine tone; the frequencies used in this example are 2000 and 1600 Hz.

```
Run   T  GatedDP % Transient class, frequency specific
Subtype                          F
StoppingRule                     1
File1 gated2000.2es     % comments out here
File2 gated1600.2es              %
NumFreqs   1
NumLevels                        5
FreqLevel1 2000 70 60 50 40 30
FreqLevel2 1600 70 60 50 40 30
StimLength                      512
StimReps                         1
NumBuffers                      20
DiscardValue                     0
```

If the subtype is *F*, then one or both of the first fields of FreqLevel1 and FreqLevel2 must differ from 0. If either first field is negative, the program should halt with an appropriate error message. If only one of the first fields differs from 0, then the response of the program should be in the frequency bin corresponding to the other non-zero first field. If both first fields differ from 0, as in this example in which the first fields identify 2000 and 1600 Hz, respectively. These frequencies are converted into frequency bin indices in exactly the same manner as for the continuous 2EOAE responses. Such frequency-specific responses are plotted in the I/O functions and related output in the plotting GUI, rather than the default transient 2EOAE condition which plots total energy across all bins in the measurement range.

The control flow in processing the Experiment List is that the program proceeds from beginning to end of the list automatically unless the user uses the `Pause` button. If paused, the user has the options to `Continue` (continue data acquisition without modification of internal results or variables), `Restart Current Run` (discard data from current run and re-start at first set of level conditions), or `Stop` (cease data acquisition using the Experiment List). This provides the user the opportunity to re-start a particular run after acquiring data at one or more sets of levels.

It should be possible to start an Experiment List on some other Run than the first Run, in order to provide the operator the ability to Stop the Experiment List and restart it on some other session (some other day, etc.) to complete all runs in the list. The two or more sessions will be tied together to the same subject code. The GUI should have a popup list with the run number on which to begin the test, with valid values from 1 to `NumRums`. The default value is 1. If the operator selects some other value, the program should write any new data to the files of the same name, as determined by the subject ID and test ear.

A system parameter file contains the Site ID and Probe ID.

D. Stimulus Autolevel

The purpose of the autolevel operation is to present tones at sound pressure levels that are closed to desired values. The approach is a simplified one in which the total SPL is matched rather than the SPL in a particular frequency bin. Thus, this autolevel scheme is not applicable for conditions in which two or more sine tones are combined, each at a desired SPL.

The parameter `NumLevels` specifies the number of test levels per stimulus condition. Each stimulus condition corresponds to using a pair of stimulus files. The set of desired sound pressure levels (SPL) is listed as parameters of `FreqLevel1` and `FreqLevel2`. For the first SPL in this parameter list, the level of the stimulus output by each DAC must be autoleveled so as to produce the desired first SPL, denoted by $SPL^d[k] = SPL^d$ for a continuous stimulus (the right-hand side of the equation is the simplification in which the total SPL is approximately equal to the SPL in the k-th bin for a sine tone), or SPL^d for a transient stimulus across all frequencies. Based on Eqs. [15], these desired SPLs are $SPL^d_{1.tot}$ and $SPL^d_{2,tot}$ for DAC1 and DAC2, respectively. Once this autolevel operation is performed for each loudspeaker, the subsequent desired SPLs are obtained by a relative attenuation of the stimulus. Each stimulus is initially presented at a default attenuation level A_d, because it is desirable to attenuate the raw stimuli to control for measurement-system distortion The response averaged over 8 stimulus presentations is measured. Thus, the output stimulus in each DAC channel is scaled by a multiplicative attenuation factor a_0 by

$$a_0 = 10^{-A_d/20} \qquad [23]$$

and then each value is rounded to the nearest integer (because the DAC outputs integer values). Using DAC1 as an example, data are acquired using this stimulus and the resulting total SPLs $SPL^{actual}_{1,tot}$ and $SPL^d_{2,tot}$ are calculated using Eq. [15] for the responses to DAC1 and DAC2 responses (based on 2E waveforms $p_1[n]$ and $p_2[n]$ in response to s_1 and s_2). The relative attenuation levels between the desired and actual are defined by

$$\begin{aligned} \Delta L_1 &= SPL^{actual}_{1,tot} - SPL^d_{1,tot}, \\ \Delta L_2 &= SPL^{actual}_{2,tot} - SPL^d_{2,tot}. \end{aligned} \qquad [24]$$

The corresponding multiplicative channel attenuation factors a_1 and a_2 are defined by

$$\begin{aligned} a_1 &= 10^{-\Delta L_1/20}, \\ a_2 &= 10^{-\Delta L_2/20}, \end{aligned} \qquad [25]$$

The channel attenuation factors may be larger or smaller than 1, but the overall attenuation factors $b_1 = a_0 a_1$ for DAC1 and $b_2 = a_0 a_2$ for DAC2 for the first level in the test list must not exceed 1.

The `FreqLevel1` and `FreqLevel2` internal list of levels may be internally scaled so that the first value is 0 dB, and all subsequent levels subtract this first level. For example, the list

```
FreqLevel1 1600  75   65   55   45   35   25
```

is stored internally as

```
FreqLevel1 1600  0   -10   -20   -30   -40   -50
```

and the internal level differences are stored as δL (equal to 0, -10, -20, -30, -40, -50). An attenuation factor is defined by $c_1 = 10^{\delta L_1/20}$ for `FreqLevel1` and $c_2 = 10^{\delta L_2/20}$ for `FreqLevel2`. The total attenuation factors are $d_1 = b_1 c_1 = a_0 a_1 c_1$ for DAC1 and $d_2 = b_2 c_2 = a_0 a_2 c_2$ for DAC2. The stimulus waveform is multiplied by the total attenuation factor and rounded to integer form. Because all DAC stimuli are stored using the full-range of the DAC converters, a value of overall attenuation exceeding 1 would request a DAC output exceeding its full range. The program should check for this error condition and halt the program if an error is detected. Each DAC stimulus is multiplied by its overall attenuation factor and rounded to the nearest integer (but round towards 0 for the most extreme positive and negative DAC output values to prevent overflow).

E. *Output Files*

During data acquisition, results should be written to the binary output file (type `2EL` for left-ear response, `2ER` for right-ear response, or `2EC` for coupler response) at the conclusion of each run list. The `2ER` and `2EL` files have identical structure, and `2EC` has identical structure except in the introductory globals part. The file name is a concatenation of the subject ID (a unique string) and file number (1,2, etc.). The above is for `ADCchannels` set to the default value of 1. When two ADC channels are acquired, the output from each channel should be stored in separate files with `Ch2` appended to the end of the file name. The header information for channel 2 can be identical to that used in channel 1, when it is based on properties inferred from calibration using properties of channel 1.The file structure is hierarchical with the following data structure (default binary except for selected text fields as-or-if appropriate):

1. Globals: contains Experiment-List information on subject/coupler, details to be determined
   - 1.1.Sample rate
   - 1.2.Site ID

1.3. Operator ID
1.4. Probe ID
1.5. Date and Start Time
1.6. High pass filter information (file name of impulse response)
1.7. Microphone response data structure
1.7.1. Microphone probe label
1.7.2. Pairs of frequency, Microphone sensitivity levels from `MIC.TXT` file

2. Run #: each run list has a corresponding data structure consisting of
2.1. Run List parameters
2.2. Stopping rule
2.2.1. Stopping Rule Parameter File values
2.3. Run/Level #: responses at each level within a run have this data structure:
2.3.1. L1 desired level and actual level
2.3.2. L2 desired level and actual level
2.3.3. Values of attenuation factors for initial stimulus (a_0) and each condition (d_1, d_2)
2.3.4. Stopping rule indicating which OR condition was reached (max buffers, etc.)
2.3.5. General run data
2.3.5.1. Total number of buffers acquired
2.3.5.2. Number of valid buffers
2.3.5.3. Time of completed run
2.3.6. SPL1 (all SPL's from bins *1* to *N/2* excluding DC bin 0)
2.3.7. SPL2
2.3.8. SPL12
2.3.9. SPLoae
2.3.10. SPLnoise
2.3.11. P1 waveform (*N* values in units of mPa=0.001*Pa)
2.3.12. P2 waveform
2.3.13. P12 waveform
2.3.14. distortion waveform
2.3.15. TFR parameters
2.4. Other info may be added (bin numbers at which to estimate sine-tone evoked emissions, etc.)

The time-frequency representation (TFR) is not calculated during data acquisition. The TFRs will be calculated in the analysis mode of the program by selecting either of the 2E responses to the stimuli, or for the 2E waveform, or else in special analysis programs to be devised for batch-mode processing.

It is anticipated that some data base creation program will extract results from these files and write them in Excel-readable form. In the initial program version, the program should be able to read a results file and show all the plots.

E. 2EOAE Mode Plots

The plot window outputs from the 2EOAE data collection include:

1. *SPL Plot*: plot the average SPL1, SPL2, SPL12 and SPLoae as a function of frequency at one test level (selectable from list), each in a different color.
2. *OAE Spectrum Plot*: plot the SPLs of the OAE signal and noise versus frequency at various levels
3. *OAE I/O Function Plot*: plot the SPLs of the OAE signal and noise versus level at various frequencies
4. *Waveform Plot*: plot the P1, P2 and 2E waveform of the averaged responses
5. *Waveform Plot Channel 2*: plot the corresponding ADC2 waveforms in two-channel mode.

For Continuous OAEs, the frequencies are selected as discrete indices, for example, the bin index of the frequency of the SFOAE or 2f1-f2 for a DPOAE (or mf1-nf2 for arbitrary distortion components--this will need to be specified later on). For now, the Continuous OAEs plot results in a single frequency bin. For Transient OAEs, the frequencies are fractional octave averaged responses at given center frequencies, except for subtype `F` described earlier which tends to mimic the continuous-class 2EOAE GUI structure.

The general plot window format is:

- A fixed height and width in pixels
- Type of plot, and subject/coupler file name on window title bar
- Tab to select which Run to view, with string field of the corresponding `Run` line in the Experiment List.
- Popup list of selectable *f2* frequencies if correspond Run list has `NumFreqs`>1. (f1 and f2 are equal for SFOAEs)
- Use string fields 1 and 2 that will be defined in Experiment List to document run information
- Place file name and date on each plot (useful on printouts)
- Provide ability to print any desired plot to the printer, or to file, or cut and paste
- Provide ability to change plot axes, range, toggle legend on or off, as provided in plotting library software, etc.

The particular details of each plot are as follows:

1. *SPL Plot*
   1.1. *Transient Class*
      1.1.1. Show raw spectra versus frequency of SPL1, SPL2, SPL12 and SPLoae using different colors and line styles, use legend
      1.1.2. Provide ability to use fractional octave average (to be specified).

   1.2. *Continuous Class*
      1.2.1. Show raw spectra versus frequency of SPL1, SPL2, SPL12 and SPLoae using different colors and line styles, use legend

The OAE Spectrum plot shows curves parameterized by each pair of levels in FreqLevel1 and FreqLevel2 as a function of frequency across *NumFreqs*.

2. *OAE Spectrum Plot*
   2.1. *Transient Class*
      2.1.1. Show raw spectra versus frequency of SPLoae and SPLnoise using different colors and line styles, add several calculated values on side of plot in text fields (to be done)
      2.1.2. Provide ability to use fractional octave average.
   2.2. *Continuous Class*
      2.2.1. Show spectra versus frequency of SPLoae and SPLnoise at a fixed pair of test levels (LEVEL1 and LEVEL2) using same color and solid/dashed line styles connecting discrete point symbol for the response at one particular bin
      2.2.2. The default bin is 2f1 -f2 bin, which might be denoted *k=2,l=1*.
      2.2.3. Provide popup or radio button list to choose other pairs of *(k, l)*, or enter a pair of arbitrary positive integer *k* and *l*.
      2.2.4. Response is extracted across *NumFreqs* responses at first level, then second level etc.
      2.2.5. Plots at various levels are overlaid, and plotted in different colors
      2.2.6. Ability to select/deselect which level plots are plotted [in plot package?]; this is so user can elect whether or not to display all *NumLevels* plots, or fewer.

The OAE I/O Function plot shows curves parameterized by frequency as a function of level across *NumLevels*. It is based on the same response data as in the OAE Spectrum plot, but plotted as a function of level rather than frequency.

3. *OAE I/O Function Plot*
   3.1. *Transient Class*
      3.1.1. Show fractional-octave averaged SPLoae and SPLnoise as a function of the level (FreqLevel2) using different colors and line styles
      3.1.2. User can select which type of fractional octave to use (12,6,4,3,2,1 for 1/12-th octave, 1/6-th octave, etc.)
      3.1.3. If octave averaging is chosen, show all I/O functions on one plot using different colors (with SPLoae and SPLnoise having the same color for each center frequency, but solid line/dashed line).
      3.1.4. If a frequency-specific transient response is selected, give user the ability to show I/O plots of the phase response of P1, P2, P12 or PD. P1 and P12 should be viewable at the f1 bin; P2 and P12 should be viewable at the f2 bin; PD should be viewable at the m*f1-n*f2 bin.
   3.2. *Continuous Class*
      3.2.1. Show spectra versus level of SPLoae and SPLnoise at a fixed pair of frequencies using same color and solid/dashed line styles connecting discrete point symbol for the response
      3.2.2. The default bin is 2f1 -f2 bin, which might be denoted *k=2,l=1*.
      3.2.3. Provide popup or radio button list to choose other pairs of *(k, l)*, or enter a pair of arbitrary positive integer *k* and *l*.
      3.2.4. Plots at various frequencies are overlaid, and plotted in different colors

3.2.5. Ability to select/deselect which level plots are plotted [in plot package?]; this is so user can elect whether or not to display all *NumFreqs* plots, or fewer.

3.2.6. Give user the ability to show I/O plots of the phase response of P1, P2, P12 or PD. P1 and P12 should be viewable at the f1 bin; P2 and P12 should be viewable at the f2 bin; PD should be viewable at the m*f1-n*f2 bin.

4. *OAE Waveform Plot*

4.1. *Show any ADC buffer waveform as voltage (in mV) versus time (in mS). This is of particular interest for transients, but it should be possible to review any stimulus (P1,P2,P12) or OAE waveform. This is for use in data collection but it may be necessary to review this in data analysis mode.*

4.2. *For data analysis, it should be possible to view the (P1,P2,P12,OAE) waveforms in pressure units (mPa) vs. time.*

4.3.

F. TFR

The time-frequency representations (TFRs) are calculated as described in the Introduction using the discrete-time cone kernel (reference [6]). TFRs should be calculated for each of the *P1, P2* and *OAE* waveforms based on the current *circular rotation* value. This enables comparisons of stimulus and response TFRs, and the user has flexibility in choosing the time origin. In such a calculation, each waveform is first rotated preparatory to the TFR calculation. Each waveform is internally stored as a pressure waveform in *mPa*. As an initial conversion, before the TFR calculation, each waveform should be divided by 0.02 *mPa*, the reference pressure. This simplifies conversion to SPL. Details of the TFR calculation are outside the scope of this document, but follow the convention used in the BTNRH Matlab 2EOAE implementation. One change from reference [6] is the inclusion of buffers of zeros before and after the response, which are used to build in a rotational symmetry condition, namely, that the rotation of the input waveform by *R* samples should produce a rotation in the time axis of the TFR by *R* samples. Also, a particular smoothing window is used in calculating the TFR.

The output of the TFR calculation is a matrix of magnitudes (the complex value in each cell should be converted to magnitude). This section refers to a group of TFR parameters, which have default values and are stored in a local directory. It should be possible to modify these defaults, but the defaults are applied at program startup. These parameters determine how the TFR information is displayed. There are two displays: a TFR color contour plot that plots all TFR information and a 2-D TFR that plots one or more individual signals as a function of time (or frequency).

These are the TFR parameters that control the TFR color contour plot:

- *TFRautoscale:* with the value 1 uses an autoscale procedure, and with the value 0 uses a fixed range procedure,
- *TFRmaxLevel:* If TFRautoscale==0 (fixed range), then the software uses *TFRmaxLevel* as the maximum level (in dB).
- *TFRdynamicRange:* the dynamic range (in dB)

It should be possible to modify these values without re-calculating the TFR except for the calculations described below for the noise floor. An additional parameter *fLinear* should control whether the the frequency axis is plotted linearly (*fLinear*=1) or logarithmically (*fLinear*=0); and *tLinear* should control whether the time axis is plotted linearly (*tLinear*=1) or logarithmically (*tLinear*=0). If *tLinear*=0 is selected, then *tLo* must be positive. The defaults are linear plots.

Four other TFR parameters control the display of frequencies from *fLo* to *fHi*, and times from *tLo* to *tHi*, and affect the calculation of the "noise floor" of the displayed TFR in the color contour plot. The program should store default values of these parameters. Initial default values are: *fLo*=1000 Hz, *fHi*=8000 Hz, *tLo*=-2.5 ms, *tHi* based on buffer length. By definition, the first sample is at *t=0*. Therefore, the value *tLo* = -2.5 ms plots the last 2.5 ms of the buffer at the initial left-hand part of the plot, the aim being to show the temporal extent of an onset stimulus centered at *t=0*. The default for *tHi* is to show the entire buffer in the temporal dimension.

If *TFRautoscale*==1, then the program should calculate the maximum value *TFRmax* across the TFR matrix with frequencies and times in the range specified by *fLo*, *fHi*, *tLo* and *tHi*. This excludes possibly spurious values outside this range. The noise floor of the displayed plot is calculated as $TFRmin = TFRmax * 10^{-TFRdynamicRange/10}$.

If TFRautoscale==0, then the program calculates $TFRmax = 10^{TFRmaxLevel/10}$, and the noise floor *TFRmin* is calculated by the same formula as above.

Each TFR matrix element is constrained not to have a value below TFRmin by calculating $tfr[i,j] = \min(tfr[i,j], TFRmin)$, and the TFR level is calculated as $tfrLevel[i,j] = 10 * \log_{10}(tfr[i,j])$. This should have units of SPL in dB, which will be checked in preliminary tests. The vectors of times and frequencies corresponding to the columns and rows of the TFR matrix are calculated for plotting purposes in labelling the axes, if they are not otherwise available.

Regarding the 2-D TFR plot, if it is a continuous OAE type or if it is a transient OAE of sub-types *C, H*, or *T,* then no plot is generated. If either the first or second stimulus is a transient OAE of sub-type *S* or *O,* then the 2-D TFR plot is generated. In the latter case, either one or both stimuli has a code to indicate the frequency bin index *k* for the sine tone at the fundamental sine-tone frequency based on the stimulus length *N* and sample rate *SR*. If the tone is continuous, then it was internally generated by the program and these parameters should be available, or if the tone is a gated sine-tone externally stored as a file (transient stimulus, sub-type *S*), then these parameters reside in its file header. For each stimulus, the corresponding frequency *f=k*SR/N* is calculated. The array of TFR frequencies is searched to find the index such that the TFR element frequency is closest to the stimulus frequency. This is performed for channels 1 and 2.

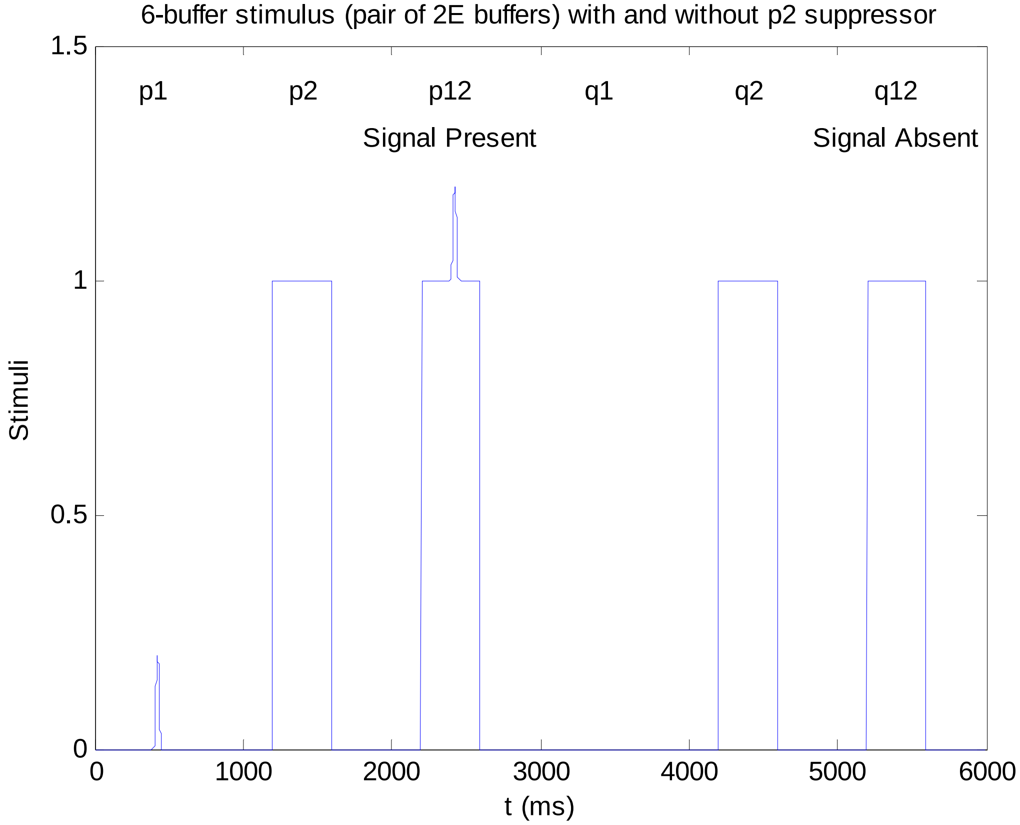
If both channels contain a continuous or gated sine tone, then there are corresponding bin values k_1 and k_2. The corresponding 2EOAE bin is at $k_{OAE} = mk_1 - nk_2$ with default values *m=2* and *n=1* so that the default 2EOAE bin is at $k_{OAE} = 2k_1 - k_2$, and its frequency is calculated and used to search the array of TFR frequencies to find the closest frequency bin. The values of *m* and *n* are under user control (in the existing interface but here applied to the TFR plots). These bins (in the TFR which may or may not align with the bins in the stimuli) are labelled i_1, i_2 and i_{oae}, respectively. If only one channel contains a continuous or gated sine tone (e.g., the other might have been something like a click), assume that it is channel 1 so that k_1 is defined and $k_{oae} = k_1$. In this case, the channel 2 response is displayed at the same bin (i.e., same frequency), $k_2 = k_1$.

The default view for the 2-D TFR is to show the TFR waveforms (actually, the TFR levels calculated above versus time, except that a noise floor parameter *TFRmin2D* is used to define the smallest TFR magnitude in the 2D plot, typically set lower than would be implied by the above *TFRdynamicRange* parameter) at the frequency bins i_1 for $tfr_1[i_1,:]$, i_2 for $tfr_2[i_2,:]$, and i_{oae} for $tfr_{oae}[i_{oae},:]$. There is one plotting window with three plots in it. In the general case in which i_1 and i_2 are unequal, this is a generalized DPOAE response, and the 2-D TFR plot shows that TFR time series of the *p1* response in the *f1* bin, the TFR time series of the *p2* response in the *f2* bin, and the TFR series of the *OAE* pressure response in the *2f1-f2* bin. The user should be able to view all three curves, or any subset of them by appropriate GUI controls (hopefully provided by the plotting package). There should be controls for the user to plot row or column vectors selected at arbitrary times and frequencies, but the above defaults are fixed in the implementation.

V. Adaptive 2EOAE

The basic approach is to implement a sequential adaptive technique (Run parameter `Run A`) to detect a 2EOAE according to a specified criterion, and to vary a stimulus level until the criterion is satisfied. Such a variation is adaptive. The adaptive technique operates in transient mode, with file names (`File1, File2`) specified for the stimuli s1 and s2.
As before, the 2ES files should be normalized such that the maximum DAC amplitude is $2^{23}-1$. For adaptive runs (Run A), the levels (run parameters `FreqLevel1` for s1 and `FreqLevel2` for s2) are initially set with respect to 0 dB as full-scale. Thus, the corresponding run parameter values must be negative, since the attenuation level is reduced relative to 0 dB. Each buffer has `StimLength` samples as in other operations of the program. The conceptually simplest experiment is to consider s1 as a signal and s2 as a noise (i.e., any other signal, which might be tones, random noise, etc.). The threshold task (Run parameter `AdaptMode Threshold`) is to detect whether s1 is present in the distortion pressure spectrum in two presentations of the 2E buffers. Note on parsing the experiment list: for this and all other Run parameter values, it is sufficient that the program convert the string to upper case and test for the first character as ‘`T`’ to encode the value `Threshold`. Different run parameter values will always start with different letters. The run parameter name (`AdaptMode` in this case) must be spelled out in full.

One sequence of 6 stimulus buffers is shown below (Run parameter `AdaptBuffers 6`), in which p1, p2, p12 are the 2E buffers with the signal present, and q1, q2, q12 are the 2E buffers with the signal absent. The corresponding distortion pressures are pd and qd, respectively. Subset of the distortion-pressure signals are used to determine which is larger, with the subset defined as `AdaptStimLength` (Run parameter) number of samples starting at sample `AdaptInitialSample` (Run parameter). Typically, `AdaptStimLength` is chosen to be a multiple of 2 to some integer power. The Run parameter `AdaptWindow` determines whether no window is used (`AdaptWindow Rectangular`) or a Hanning window is used (`AdaptWindow Hanning`). If a Hanning window is used, it is of length `AdaptStimLength` and is multiplied by each of pd and qd at the appropriate start sample. The SPL is calculated for pd/qd using the standard formula in the frequency bin containing the frequency `AdaptFreq` (Run parameter) in Hz. By the same method, the SPL is calculated for p1/q1, p2/q2, and p12/q12. The output file should store all the Run list parameters and values from the Experiment list.



The program calculates which of the two buffers, p12 or q12, has the largest energy (in a manner defined below), thus leading to a larger pd or qd, respectively (based on the assumption that p1 has energy at `AdaptFreq`). The buffer with the largest energy, in this case pd, would be identified as containing the signal. This choice is called Correct. However, if the signals also have external noise (not shown) and p1 is small, then it is possible that qd would have the most energy, and in this case the program choice would be False. Suppose the adaptive paradigm is to fix the noise level (of p2) and vary (i.e., adapt on) the signal level p1 (note that q1 is always zero). This is specified by the Run parameter `Vary 1`.

A trial is the name for each presentation of the 6-buffer stimulus. In the 1-up, 2-down adaptive paradigm (Levitt, 1972; see especially his Fig. 4), if the choice is correct twice in succession, then the p1 level is reduced by a fixed step size (e.g., by 2 dB). If the choice is false, then the p1 level is increased by the same fixed step size. In practice, larger step sizes are used initially and reduced in size after the first *reversal*. A reversal is a set of two or more trials in which the level is reduced followed by a set of one or more trials in which the level is increased. The initial step size (in dB) is given in the Run parameter `InitialStepSize`, and, following the first reversal, the final step size (in dB) is `StepSize`.

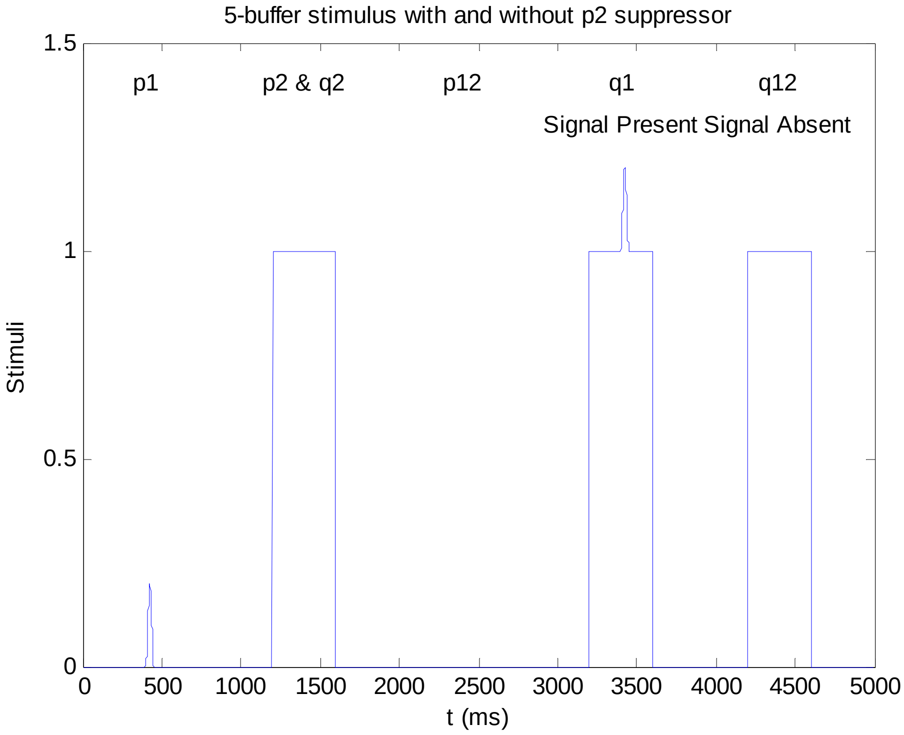
The computer presents the stimuli repetitively, calculates the energies of pd and qd to make the choice, and then varies the level of p1 until either a fixed number of trials are completed following the change in step size (selected by Run parameter `AdaptStoppingRule Trials`) or a fixed number of reversals are completed

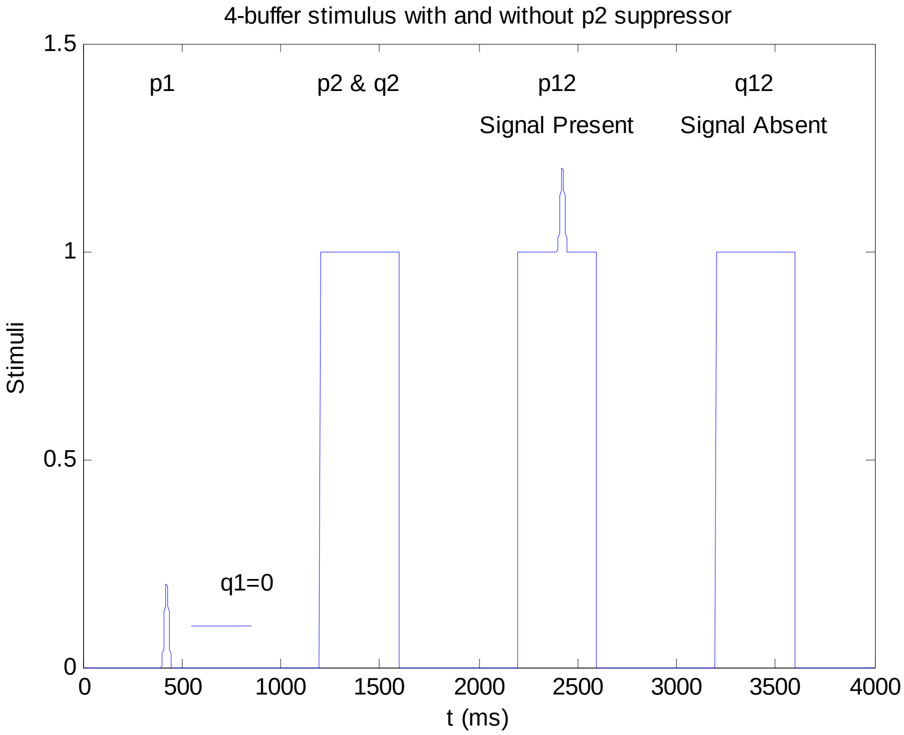
(AdaptStoppingRule Reversals). A single reversal consists of reducing the stimulus level followed by increasing the stimulus level. The fixed number of trials or reversals is given by the Run parameter AdaptFixedNum. For each trial, the program stores in the output file the sequence of SPLs calculated for the AdaptStimLength sample for p1, p2, p12, pd, q1, q2, q12, and qd. The final threshold of p1 is the average and standard deviation of the finite number of trials or finite number of reversals (depending on AdaptStoppingRule)after changing from InitialStepSize to StepSize. The trial number for the first occurrence of StepSize is also stored. It is desirable to store some of the waveforms. The entire 6-buffer set of response waveforms is stored for all trials in the first reversal following the change to StepSize. For each of these same first-reversal trials, the AdaptFixedNum length waveforms and spectra (in SPL) are stored for p1, p2, p12, pd, q1, q2, q12, qd. The program should show all the information in an Info Panel and should plot all waveforms and spectra listed above.

There is no artifact rejection in this paradigm but there is real-time filtering.

The 5-buffer variant of the above using AdaptBuffers 5 has stimulus buffers shown below. In this case, p2 need not be repeated as q2 since the stimuli are identical. Otherwise, the processing is as in the 6-buffer case with q2 results repeated from the p2 results. The 4-buffer variant of the above using AdaptBuffers 4 has stimulus buffers shown below. This is similar to the 5-buffer variant except that, for the case Vary 1, the q1 signal part of the buffer (of length AdaptStimLength) is fitted entirely within the p1 buffer based on a start sample, which is an additional Run parameter Adapt4InitialSample. All samples Adapt4InitialSample to Adapt4InitialSample+AdaptStimLength-1 must fit within the p1 buffer (for Vary 1). Either q1 should follow p1 (as in the figure), or q1 should precede p1. In the first case, if Adapt4InitialSample>AdaptInitialSample, then the program should verify that Adapt4InitialSample+AdaptStimLength-1<StimLength. In the second case, if Adapt4InitialSample<AdaptInitialSample, then the program should verify that Adapt4InitialSample+AdaptStimLength-1< AdaptInitialSample. The reason for including q1 is that qd=q1+q2-q12 must include the effect of the noise in q1 in calculating the distortion signal. In practice, the performance of the adaptive technique will be compared for the 6-, 5- and 4-buffer stimuli (we will have a methods paper here), and the shortest stimulus set without artifact will ultimately be used. The case Vary 2 is obtained by interchanging buffer labels 1 and 2.

In adaptive mode, change from InitialStepSize to StepSize after the first turning point (as a function of the number of trials), with a turning point being a half-reversal. Start averaging the AdaptFixedNum number of reversals or trials in either case after the first two reversals are completed. This has the effect of discarding the initial trials of the measurement before starting the averaging process.


5-buffer stimulus with and without p2 suppressor
1.5
1
0.5
0
Stimuli
p1
p2 & q2
p12
q1
q12
Signal Present Signal Absent
0
500
1000
1500
2000
2500
3000
3500
4000
4500
5000
t (ms)


4-buffer stimulus with and without p2 suppressor
1.5
1
0.5
0
Stimuli
p1
p2 & q2
p12
q12
Signal Present
Signal Absent
q1=0
0
500
1000
1500
2000
2500
3000
3500
4000
t (ms)

To partially control for multi-stimulus buffer order effects, we will randomize on the order of the p/q buffers if `AdaptRandom 1` is selected. No randomization is used for `AdaptRandom 0`. The default value is to randomize, i.e., `AdaptRandom 1`. For `AdaptBuffers 6`, randomize on whether the 3 2E buffers for p are presented first or last. Also randomize for each trial on whether p1 is presented before or after p2, and whether q1 is presented before or after q2. These can be represented as a random choice of the following 8 possible 6-buffer permutations:
p1, p2, p12, q1, q2, q12
p2, p1, p12, q1, q2, q12
p1, p2, p12, q2, q1, q12
p2, p1, p12, q2, q1, q12
q1, q2, q12, p1, p2, p12
q1, q2, q12, p2, p1, p12
q2, q1, q12, p1, p2, p12
q2, q1, q12, p2, p1, p12

The 8 possible 5-buffer permutations are
p1, p2, p12, q1, q12
p2, p1, p12, q1, q12
q1, q12, p1, p2, p12
q1, q12, p2, p1, p12
p1, p12, p2, q1, q12
p2, p12, p1, q1, q12
q1, q12, p1, p12, p2
q1, q12, p12, p1, p2

The 8 possible 4-buffer permutations are
p1, p2, p12, q12
p2, p1, p12, q12
q12, p1, p2, p12
q12, p2, p1, p12
p1, p12, p2, q12
p2, p12, p1, q12
p1, q12, p2, p12
p2, q12, p1, p12

The supra-threshold adaptive test is selected by `AdaptMode Supra`. This is relevant to a paradigm such as two-tone suppression in which a stimulus level is varied so as to obtain a SFOAE level that is a fixed number of dB (usually below a saturated SFOAE level). *This will be implemented later*, because it requires the ability to measure the saturated SFOAE SPL on a preceding run in the Experiment List using Matlab to calculate intermediate results, and pass that value to the adaptive run.

Sample Runs in an experiment list for use in adaptive testing are provided below, based on the the run parameters described above.

```
Run                          A                             ; (A=Adaptive)
File 1                 stim1.2es
File 2                 stim2.2es
NumLevels                    3
FreqLevel1                   0      -20      -20      -20 ; level in dB re:0 dB for full scale
FreqLevel2                   0    -30.3    -10.3    -20.3
StimLength               22050
AdaptMode            Threshold ; Threshold=on larger of s12 bins, Supra=supra-threshold
AdaptBuffers                 5                             ; number of buffers: 4, 5, or 6 as in spec
AdaptInitialSample        1000
AdaptStimLength            512
AdaptWindow            Hanning or =Rectangular
AdaptFreq                 1000                             ; Hz
Vary                         1                             ; if =1, then vary L1; if =2, then vary L2
InitialStepSize              6                              ; dB
StepSize                   1.5                             ; dB
AdaptStoppingRule    Reversals
AdaptFixedNum               10 ; use 10 reversals after changing from InitialStepSize to StepSize
```

```
Run                          A                             ; (A=Adaptive)
File 1                 stim1.2es
File 2                 stim2.2es
NumLevels                    3
FreqLevel1                   0      -20      -20      -20 ; level in dB re:0 dB for full scale
FreqLevel2                   0    -30.3    -10.3    -20.3
StimLength               22050
AdaptMode            Threshold ; Threshold=threshold test, Supra=supra-threshold test
AdaptBuffers                 4                             ; number of buffers: 4, 5, or 6 as in spec
AdaptInitialSample        1000
AdaptStimLength            512
AdaptWindow            Hanning                             ; =Hanning  or =Rectangular
AdaptFreq                 1000                             ; Hz
Vary                         1                             ; if =1, then vary L1; if =2, then vary L2
InitialStepSize              6                              ; dB
StepSize                   1.5                             ; dB
AdaptStoppingRule    Reversals ; use 10 reversals after changing from InitialStepSize to StepSize
AdaptFixedNum               10                             ; q1 follows p1
Adapt4InitialSample       6000                             ; no randomization
AdaptRandom                  0
```

V.A. 2E Acoustic reflex test

In the above adaptive tests, the signal p2 is the suppressor signal, and may be sufficiently high in level to evoke the acoustic reflex. A 2E acoustic reflex threshold (ART) test is designed to measure the threshold level of a suppressor signal that evokes the acoustic reflex (AR) shift on a probe signal p1. The final algorithm will be adaptive, but the present test is designed (in a non-adaptive manner) to provide preliminary data with which to design the final algorithm. The preliminary Run list specifies the run parameter and value `Run R` for the ART mode. This places the program in a transient-like mode, in which stimulus files are read for DAC1 and DAC2, each of a stimulus length given by `StimLength`. The program presents the stimuli at a number of levels given by `NumLevels` (5 in the Run example below). Like the adaptive run lists, the `FreqLevel` run values are expressed as attenuations in dB of the input signal. Thus, no calibration or time averaging takes place for the ART mode of the program.

In preliminary testing, s1 is the probe signal containing tones are ¼ octave frequency spacings from 0.25-1.5 kHz, which is a sensitive frequency range in which to measure the ART. These tones are gated on and off, leaving an inter-stimulus period of silence. Stimulus s2 is the AR elicitor, and its level is varied from relatively high levels (at which the ART will most likely be activated) down to lower levels (in 5 dB steps in this example, but, ultimately, adaptively varied). The s2 elicitor is gated on and off with a shorter duration than s1, and its on window is entirely contained with the on-duration of s1. Thus, the joint presentation of the stimuli as s12 shows an initial period of time with s1 alone, followed by s1 and s2 together, followed by s1 alone. The ART will be evaluated in terms of the relative shifts at each of the s1 frequencies in the pd responses before and after the s2 activator is gated on. The calculations will be done in Matlab until the algorithm is finalized, so the output waveforms should be readable by a Matlab function.

A sample Run list is given below.

```
Run              R         ; (R=Adaptive acoustic reflex test)
File 1           ar1.2es
File 2           stim2.2es
NumLevels               5
FreqLevel1              0      -20      -20      -20      -20      -20
FreqLevel2              0       -5      -10      -15      -20      -25
StimLength          22050
```

References

[1] Keefe, D. H. (1998). Double-evoked otoacoustic emissions: I, Measurement theory and nonlinear coherence. J. Acoust. Soc. Am. 103:3489-3498.

[2] Keefe, D. H. and R. Ling (1998). Double-evoked otoacoustic emissions: II, Intermittent noise rejection, calibration and ear-canal measurments. J. Acoust. Soc. Am. 103:3499-3508.

[3] Keefe, D. H., P. Piskorski and M. P. Gorga (1999). Double-evoked otoacoustic emissions: Time-frequency representations and high-frequency transient-response measurements. Assoc. Res. Otolaryngol. Abs. 22, 395.

[4] Levitt, H. (1972). Transformed up-down methods in psychoacoustics. J. Acoust. Soc. Am. 49, 467-477.

[4] Burns, E. M. and D. H. Keefe (1998). Energy reflectance can exceed unity near SOAE frequencies. J. Acoust. Soc. Am. 103:462-474.

[5] Press, W. H., S.A. Teukolsky, W.T. Vetterling and B.P. Flannery (1992). *Numerical Recipes in C: The Art of Scientific Computing*, Second edition (Cambridge).

[6] Pitton, J. and L.E. Atlas (1995). Discrete-time implementation of the cone-kernel time-frequency representation. IEEE Trans. Signal Proc. 43:1996-1998.