Stimulus latency bug

Bruno A. Olshausen

September 8, 2001

Problem

Charlie and I discovered a bug in the stimulus presentation/data acquisition system, which appears to be due to hardware failure on ControlC. We found the error by running some sparse noise test trials in which the data acquisition was triggered off of ControlC. We found that ControlC's trigger to initiate data acquisition was not synchronized with PresenterC's vertical sync signal (nor PresenterC's 'start' signal), so the actual beginning of stimulus presentation was at some random interval (within a refresh cycle) after the initiation of data acquisition. We also noticed that the first trial within a session is usually delayed by two or three vertical syncs from the start of ControlC's trigger, whereas the remaining trials usually begin stimulus presentation upon the first vertical sync after ControlC's trigger.

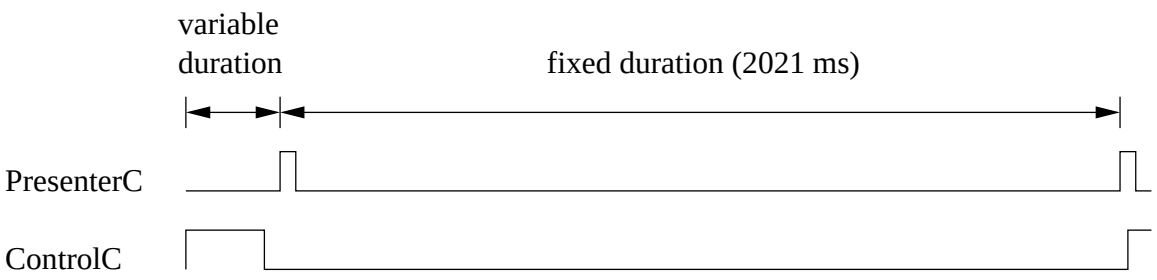
Another set of bugs that were discovered at the same time are that

1. Timing information is not being written to the `.INI` file for sparse noise trials. We believe that most of these trials were run with a 1 ms stimulus latency, 1998 ms stimulus duration, and 1 ms post-stimulus duration. In the future, Presenter needs to be fixed to write out timing information for *all* trials.

2. The refresh rate on PresenterC is not 85 Hz but rather 85.1 Hz. Evidently no one bothered to actually measure this, so it was assumed to be the value reported by the hardware, which is rounded off to the nearest integer. It is recommended that in the future the vertical sync signal be acquired along with neural data. This way, we will be able to verify that PresenterC and ControllerC are synchronized, and if there is any deviation or drift in PresenterC's vertical sync, we will be able to correct for it.

Possible workaround

There appears to be a simple fix for the variable stimulus latency problem. We noticed that for trials subsequent to the first within a session, the distance between PresenterC's 'start' and 'stop' triggers is always the same. For example, for a 1998 ms stimulus duration, we find that the distance between the two pulses is always 2021 ms. The situation is as follows:



The 2021 ms duration can be reconciled by assuming that PresenterC waits until it is *over* the desired stimulus duration by an integer number of frames before commanding a halt to data

acquisition. Once it decides to halt data acquisition, it actually does so on the next refresh. Since there are three refreshes per frame and 85.1 refreshes per second, we have

$$\begin{aligned} \#\text{ of frames presented} &= \text{ceil}\left(\frac{1.998\text{ sec.}}{(3\text{ refreshes/frame})/(85.1\text{ refreshes/sec})}\right) && (1)\\ &= 57 && (2)\end{aligned}$$

where 'ceil' means "round up to nearest integer." Thus, the actual stimulus duration is

$$57\text{ frames} \times (3\text{ refreshes/frame})/(85.1\text{ refreshes/sec}) + (1\text{ refresh})/(85.1\text{ refreshes/sec}) = 2.021\text{ sec} \quad (3)$$

If our assumption is correct, we can re-align the data with the stimulus by simply calculating the amount by which the data file exceeds 2.021 sec (or whatever number is appropriate for the commanded stimulus duration), and chop off the excess from the *beginning* of the file. Note however that this is a valid solution only for trials subsequent to the first trial within a session. For reasons which we still do not understand, the first trials within a session do not follow the above trend.

I have written two matlab functions for realigning the data. The first, called `infer_latency`, calculates the excess time for each data file. The second, `shift_spikes`, subtracts this excess time from the spike times in the session structure generated by `GenerateSessionSpikeTrains`. The raw data files are being left alone for now. Note that these routines are only applicable for data collected before 9/01. Data collected after this should be correctly synced.