Additional simulations of timecourse analysis

Daniel Baker

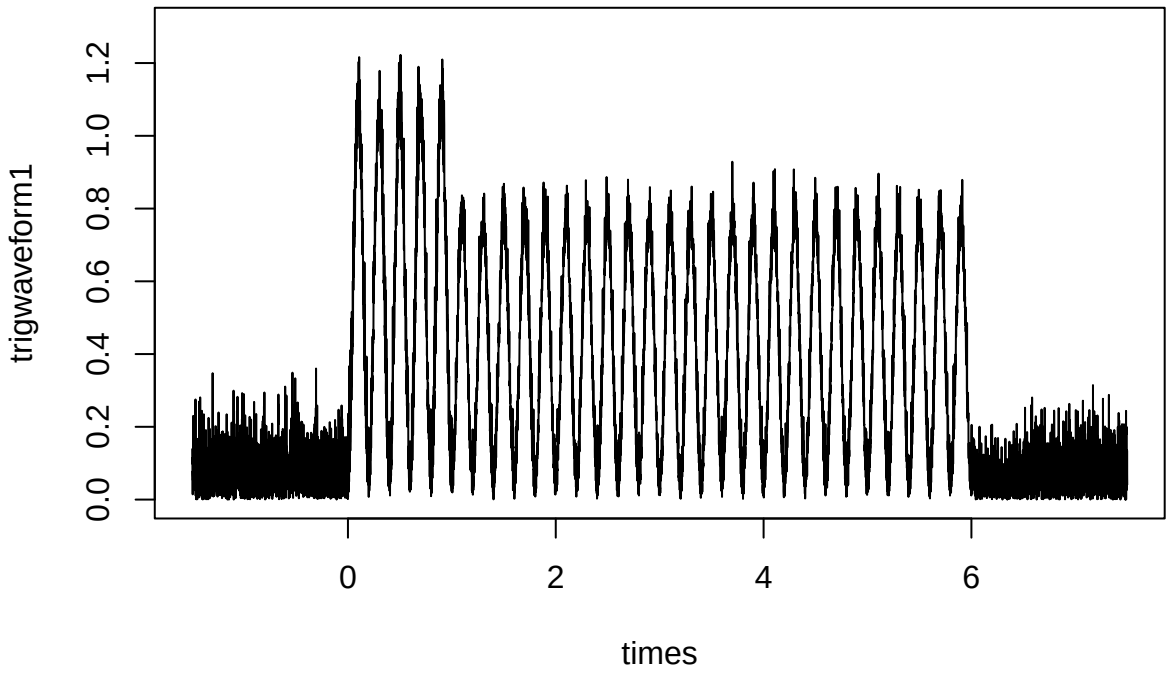
2023-07-31

```
knitr::opts_chunk$set(echo = TRUE)

# set the initial parameters for the simulations
samplerate <- 1000
targetF <- 5
duration <- 6
binwidth <- 1000
tindex <- 1 + targetF*(binwidth/samplerate)
```

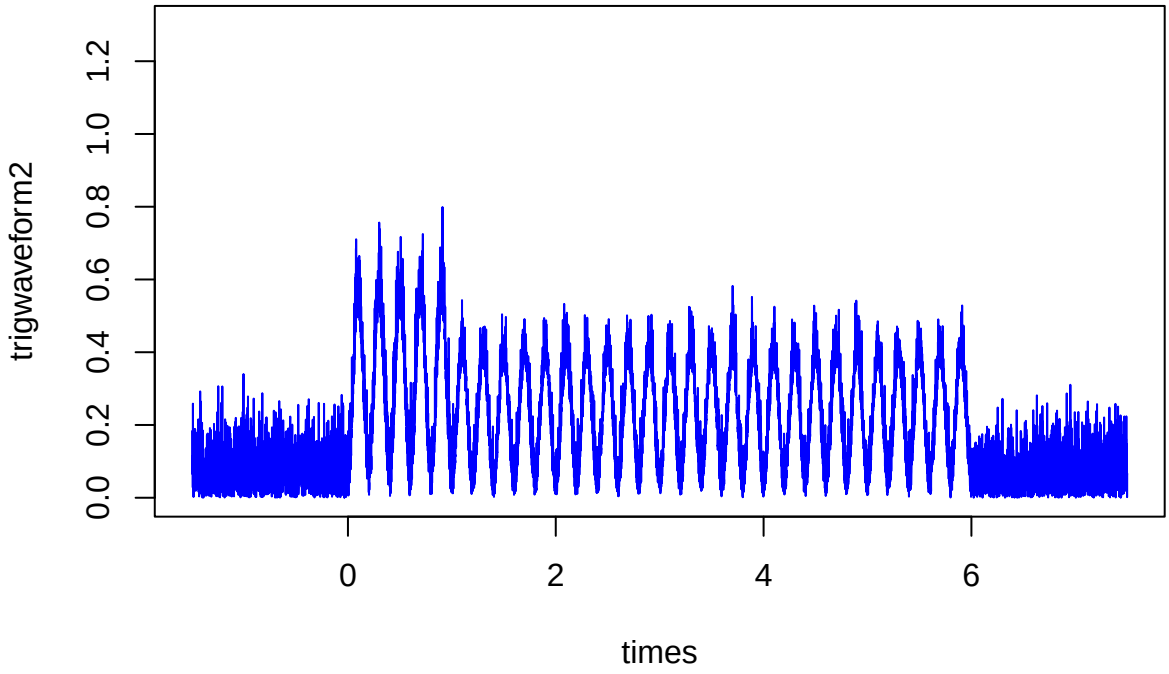
Overview

At the suggestion of a reviewer, we conducted simulations to assess the temporal precision of our analysis methods. In particular, we wanted to check if the use of a 1-second sliding time window placed a limit on our estimates of the change in suppression strength. We began by generating two samples of a sine wave carrier at 5 Hz. To simulate the onset transient often observed in steady-state data, we set the amplitude of the first 1000ms to be 1.4 times higher than the remainder of the waveform. To assess our methods for quantifying suppression, we generated two waveforms with amplitudes that differed by a factor of two. We also added white noise to each signal. The two signals are shown by the black and blue traces in the following plots:

```
times <- seq(1/1000,9,1/1000)-1.5
trigwaveform1 <- -cos(2 * targetF * times * pi)
trigwaveform1[1:1500] <- -1     # set the start to zero
trigwaveform1[7501:9000] <- -1  # set the end to zero

trigwaveform1 <- 0.1*abs(rnorm(9000))+((trigwaveform1 + 1)/2)
trigwaveform1[2501:8000] <- trigwaveform1[2501:8000] / 1.4

trigwaveform2 <- -cos(2 * targetF * times * pi)
trigwaveform2[1:1500] <- -1
trigwaveform2[7501:9000] <- -1

trigwaveform2 <- 0.1*abs(rnorm(9000))+((trigwaveform2 + 1)/4)
trigwaveform2[2501:8000] <- trigwaveform2[2501:8000] / 1.4

plot(times,trigwaveform1,type='l',ylim=c(0,1.3))
```



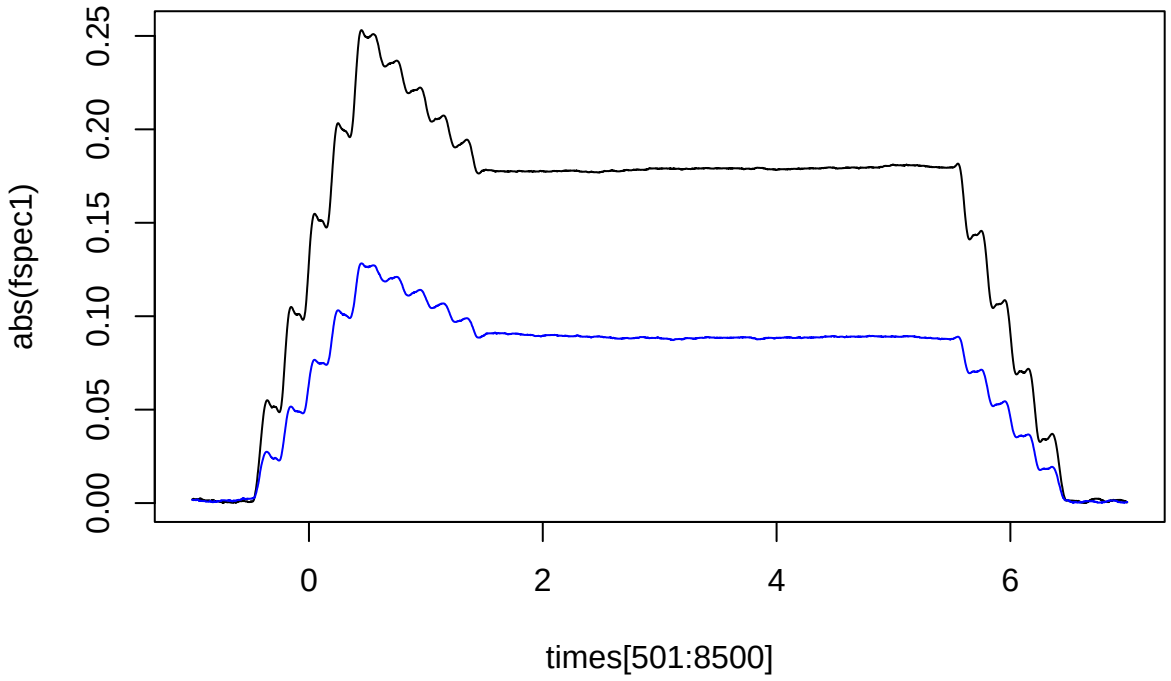

```
plot(times,trigwaveform2,type='l',col='blue',ylim=c(0,1.3))
```



Next we applied our timecourse analysis method in which the Fourier transform of the waveform was calculated in successive 1-second time windows, and the (absolute) amplitude at the carrier frequency (5 Hz) taken as the outcome. These timecourses are then plotted as the black and blue traces in the plot below. The onset transient is apparent, as is the difference in amplitude between the two signals:

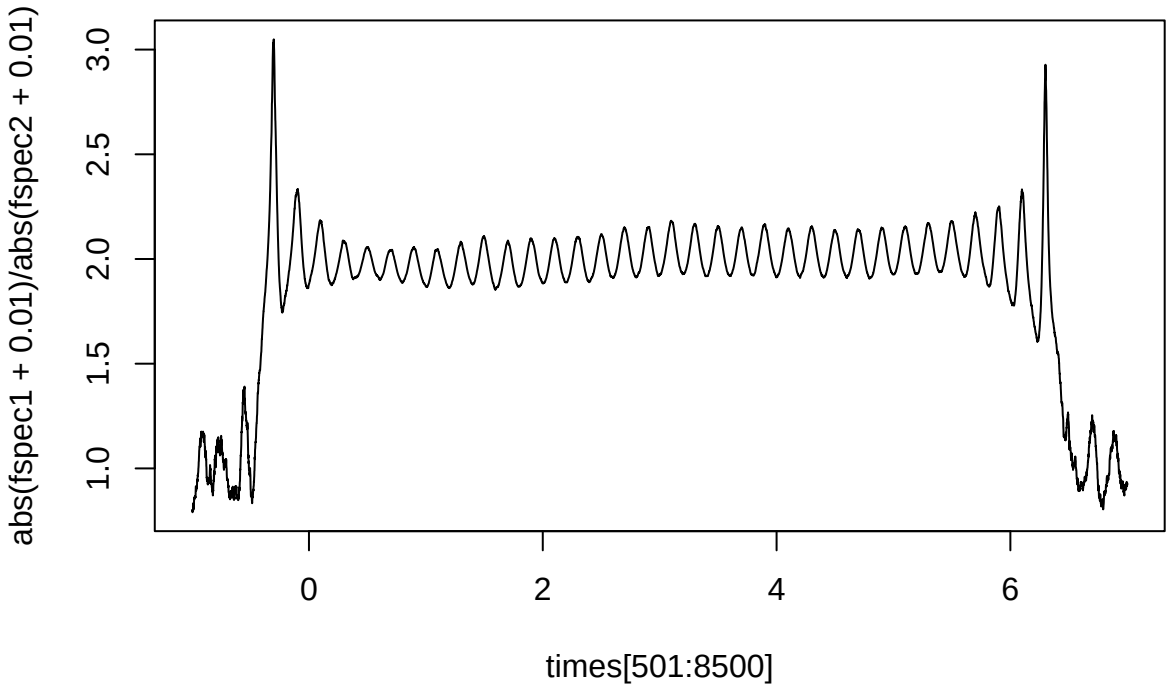
```
fspec1 <- NULL
fspec2 <- NULL
for (t in 1:(9000-binwidth)){
  spec <- fft(trigwaveform1[t:(t+binwidth-1)])/binwidth
  fspec1[t] <- spec[tindex]
```

```
  spec <- fft(trigwaveform2[t:(t+binwidth-1)])/binwidth
  fspec2[t] <- spec[tindex]
}

# adjust times so that the x-values correspond to the centre of the time window
plot(times[501:8500],abs(fspec1),type='l')
lines(times[501:8500],abs(fspec2),col='blue')
```



It is clear that the finite time-window produces a gradually ramped onset over a 1 second period. The final step was to determine the extent to which this affects the timecourse of the suppression ratios, which are defined here as the ratio of the two signals, with a small constant added to each to avoid division by zero during time points when the stimulus amplitude was low. The simulated suppression ratio is plotted below.

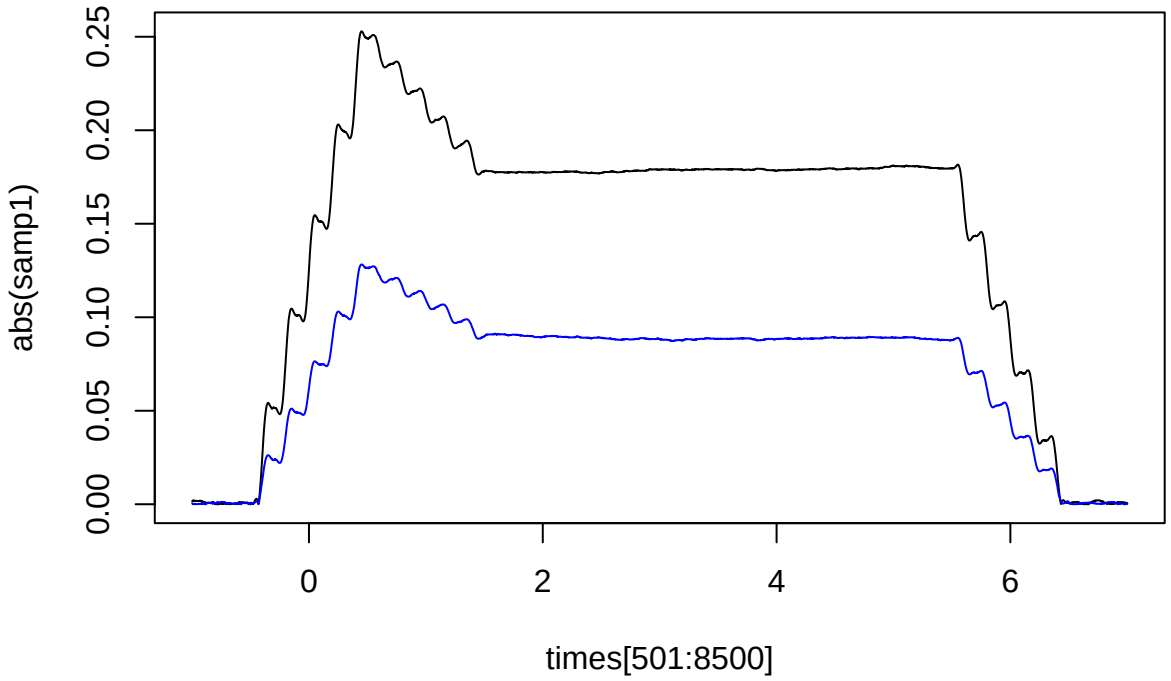
```
plot(times[501:8500],abs(fspec1+0.01)/abs(fspec2+0.01),type='l')
```



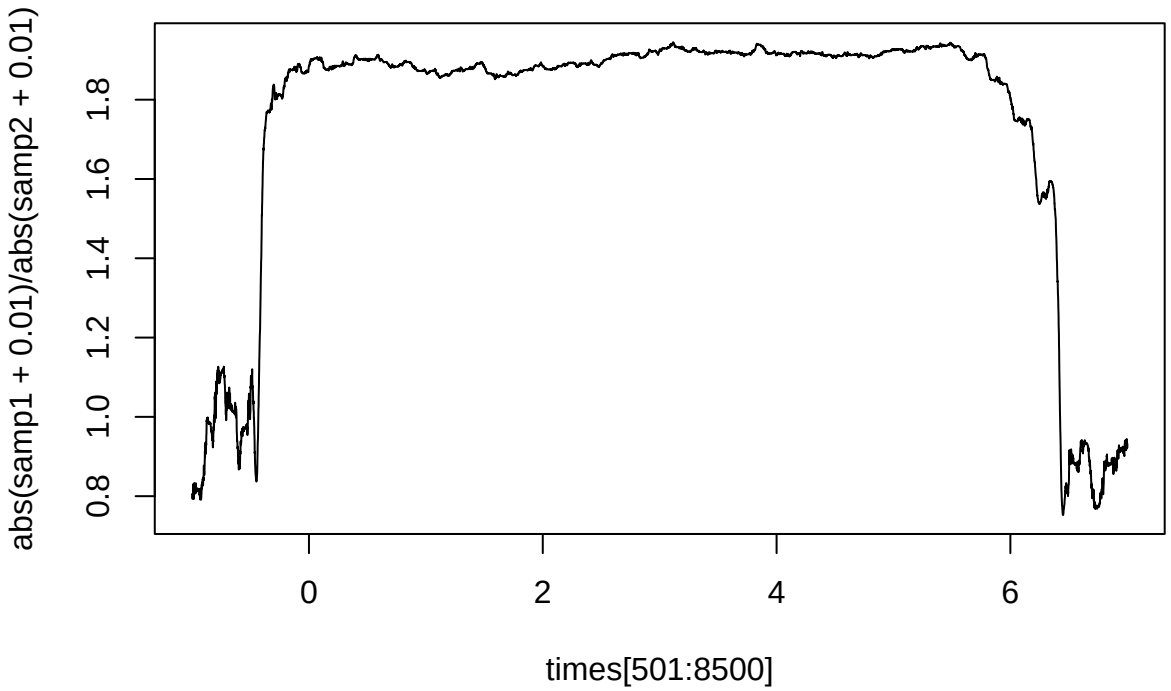
The increase in suppression ratio over time is very abrupt, taking place over the course of 1 carrier cycle. This is much more rapid than the blurring from the sliding time window, most likely because the blur cancels out across the numerator and denominator of the ratio calculation.

Finally, we repeated the analysis using an alternative fixed-phase method. Here the stimulus was multiplied (dot product) with a pure sine wave within each time window (instead of taking the Fourier transform). This has the advantage of removing out-of-phase noise. It generates a similarly precise timecourse, though with a less pronounced 'ripple' artifact from the temporal windowing.

```
# generate a pure sine wave of the correct phase
fullwave <- -cos(2 * targetF * times * pi)

samp1 <- NULL
samp2 <- NULL
for (t in 1:(9000-binwidth)){
  spec <- trigwaveform1[t:(t+binwidth-1)] * fullwave[t:(t+binwidth-1)]
  samp1[t] <- sum(spec)/binwidth
  spec <- trigwaveform2[t:(t+binwidth-1)] * fullwave[t:(t+binwidth-1)]
  samp2[t] <- sum(spec)/binwidth
}

plot(times[501:8500],abs(samp1),type='l')
lines(times[501:8500],abs(samp2),col='blue')
```




```
plot(times[501:8500],abs(samp1+0.01)/abs(samp2+0.01),type='l')
```



Overall, these simulations increase our confidence that we are able to resolve rapid increases in suppression should they exist. In our empirical data we find evidence for more gradual increases, often taking around 2-5 seconds to reach maximum levels of suppression (depending on temporal frequency). These appear not to be an artifact of our analysis method.