Script to plot Figure 1d

Table of Contents

Winawer, Kay, Foster, Parvizi, Wandell **Asynchronous broadband signals are the principal source of the BOLD response in human visual cortex** *Current Biology, 2013*

This figure shows example spectra from an On-Off experiment with a flickering large-field contrast pattern. The flicker was 7.5 Hz square wave (contrast reversals 15 times per second). The subject was S1 and the channel 104 (V1/V2 periphery). Data used for plotting the spectra includes the data plotted in Figure 1D as well as data from two other runs from the same subject and the same channel.

Set up paths and parameters

```
savepth   = fullfile(ecogPRFrootPath, 'scratch');
calcPower = true; % plot squared amplitude rather than amplitude
fmax      = 150;  % plot spectral power up to this frequency (Hz)
useHann   = true; % apply a Hanning window before computing spectrum
```

Load the data

```
% This includes
%   t:          time vector (seconds), 1x3 cell for 3 runs
%   ts:         raw time series (microvolts), 1x3 cell for 3 runs
%   onsets:     epoch onsets in temporal samples (not seconds), 1x3 cell
%   sampleRate: ECoG sampling rate, in Hz
%   T:          epoch length (in seconds)
%   subjnum:    subject number (corresponds with numbering in paper)
%   runType:    indicates that this data comes from OnOff expts
%   dataType:   indicates that data was referenced to common average
%
%   Note that each run consisted of 6 'on' epochs, followed by 6
%   'off' epochs, repeated 4 times (i.e., 4 on-off blocks of duration 12*T
%   seconds each)

load(fullfile(ecogPRFrootPath, 'data', 'figure1Data'));
```

Compute spectra of each epoch

```
% We have 3 on-off runs
```

```
numruns = numel(ts);

% We  store the power spectrum for each epoch in Spectra. Initially we put
% the spectra in a 1x3 cell array (spec) with the 3 cells corresponding to
% the 3 runs. Then we concatentate spec into the matrix Spectra.
spec = cell(1,numruns);
for run = 1:numruns
    tsmatrix = ecogTSeriesVector2TSeriesMatrix(ts{run}, onsets{run});
    spec{run} = ecogGetSpectralData(tsmatrix, T, fmax, useHann);
end
% square the spectra to plot power rather than amplitude
Spectra.all = catcell(1, spec).^2;

% frequencies are computed as multiples of 1/(epoch length)
f = [];
f.all = (0:fmax)/T;

% These are the indices to ON (1s) and OFF (0s) epochs
onEpochs = repmat([1 1 1 1 1 1 0 0 0 0 0 0], [1 4]);
% repeat the vector to account for the 3 runs (44 epochs x 3 = 148 epochs)
onEpochs = logical(repmat(onEpochs, 1, numruns));

% Spectral power from ON and OFF epochs
Spectra.on  = Spectra.all(onEpochs, :);
Spectra.off = Spectra.all(~onEpochs,  :);
```

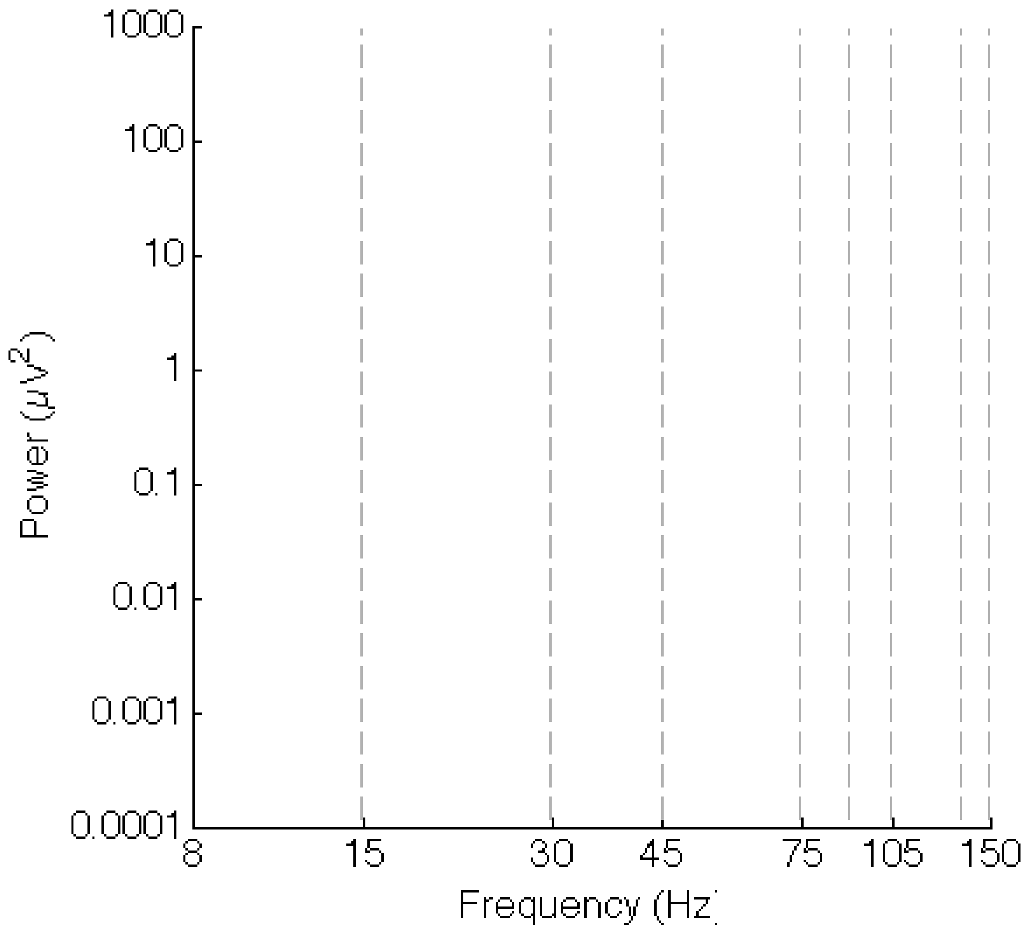
Fit lines to the spectra in log-log space

```
% Get the stimulus-locked (sl_f), asynchronous broadband (ab_f), and keep
% (keep_f) frequencies and their indices (sl_i, ab_i, keep_i) into f
% (frequency vector). Broadband frequencies are all frequencies between min
% and max excluding harmonics of the stimulus-locked and line noise
% frequencies. Keep frequencies are all frequencies between min and max.
f = ecogGetSLandABfrequencies(f.all, T);

% Normalize the power by taking the log. We do this because (a) the noise
% in the spectral measurements are approximately equal acrorss frequencies
% in the log domain, and (b) we will fit the spectrum as a straight line in
% log-log space (log power / log frequency)
Spectra.onLog  = log10(Spectra.on);
Spectra.offLog = log10(Spectra.off);

% fit the spectral data with a line in log-log space using only the
% broadband frequencies (i.e., excluding frequencies near multiples of the
% steady state flicker and line noise)
Spectra.onLogAB  = Spectra.onLog(:,f.ab_i);
Spectra.offLogAB = Spectra.offLog(:,f.ab_i);
f.logAB          = repmat(log10(f.ab), size(Spectra.onLogAB, 1), 1);

% fitted polynomial parameters (slope and intercept) in log-log space
Spectra.polyOn  = polyfit(f.logAB(:), Spectra.onLogAB(:), 1);
Spectra.polyOff = polyfit(f.logAB(:), Spectra.offLogAB(:), 1);
```

```
% predicted broadband responses
Spectra.predOn  = f.logAB*Spectra.polyOn(1)  + Spectra.polyOn(2);
Spectra.predOff = f.logAB*Spectra.polyOff(1) + Spectra.polyOff(2);
```

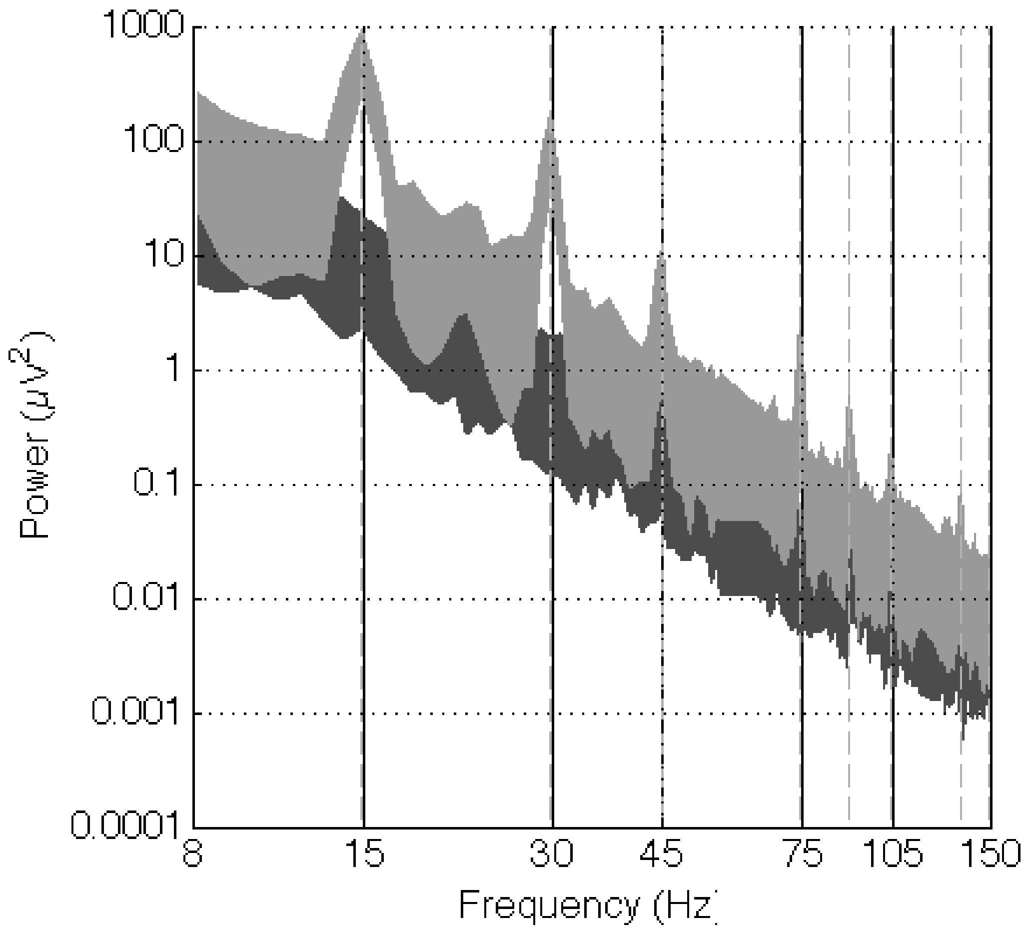
Set up a figure

```
% Figure window
fH = figure; clf; p = get(gcf, 'Position');
set(gcf, 'Color', 'w', 'Position', [p(1) p(2), 480 420])

% Define some colors
darkGray  = [0.3 0.3 0.3];
midGray   = [0.6 0.6 0.6];
lightGray = [0.8 0.8 0.8];

% Ticks and axis limits
yl = [-4 3];  yt = -4:3;
xl = [8 150]; xt = [8 15 30 45 75 105 150];

% Adjust figure properties to look nice
set(gca, 'YLim', yl, 'YTick', yt, 'YTickLabel', 10.^(yt), ...
    'XLim', xl, 'XTick', xt, 'XTickLabelMode', 'auto', 'XScale', 'log', ...
    'FontSize', 16 );
hold all; axis square
xlabel('Frequency (Hz)'); ylabel('Power (μV^2)');

% Grid lines at even harmonics of stimulus-locked frequency (except for 60,
% 120 Hz)
for ii = [1:3 5:7 9:10], plot([ii ii]*f.sl, yl, '--', ...
        'Color', [1 1 1]*.7); end
```



Shading to indicate range of spectral responses across epochs

```
% Shading of +/- 1 standard deviation for OFF epochs.
sd = std(Spectra.offLog(:, f.keep_i));
mn = mean(Spectra.offLog(:, f.keep_i));
shadedplot(f.keep, mn+sd, mn-sd,  darkGray); hold on

% Shading of +/- 1 standard deviation for ON epochs
sd = std(Spectra.onLog(:, f.keep_i));
mn = mean(Spectra.onLog(:, f.keep_i));
shadedplot(f.keep, mn+sd, mn-sd,  midGray); hold on
```



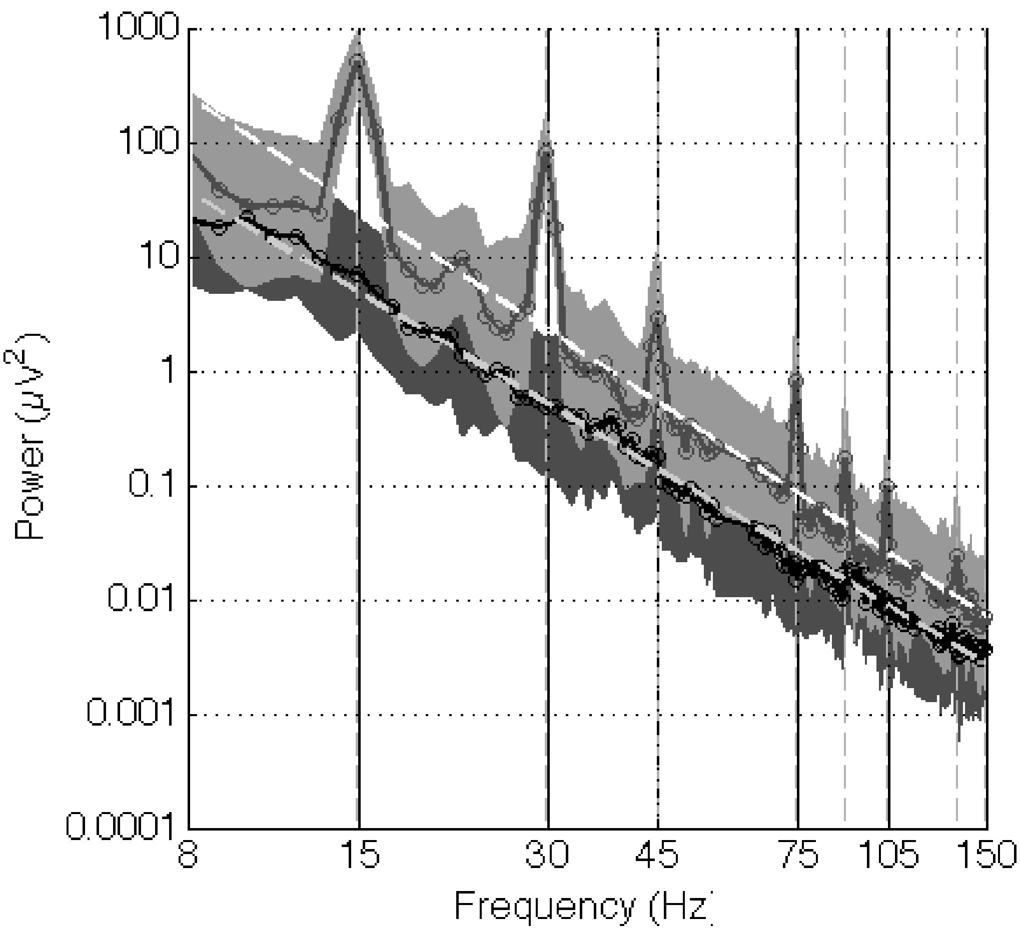
Plot mean data values and fitted lines

```
figure(fH)

% ---- Plot OFF spectrum ------------------------------------

% All data points (line)
plot(f.keep, mean(Spectra.offLog(:, f.keep_i)), '-k', 'LineWidth', 2)

% Just the data points used for the AB linear fit (circles)
plot(f.ab, mean(Spectra.offLog(:,f.ab_i)), 'ok', 'MarkerSize', 7)

% The broadband linear fit
plot(f.ab, Spectra.predOff, '--', 'Color', lightGray, 'LineWidth', 2)

% ---- Plot ON spectrum ------------------------------------

% All data points (line)
plot(f.keep, mean(Spectra.onLog(:, f.keep_i)), '-', 'Color', darkGray, 'LineWidth'

% Just the data points used for the AB linear fit (circles)
```

```
plot(f.ab, mean(Spectra.onLog(:,f.ab_i)), 'o', 'Color', darkGray, 'MarkerSize', 7)

% The broadband linear fit
plot(f.ab, Spectra.predOn, 'w--','LineWidth', 2)
```

SAVE

hgexport(fH, fullfile(savepth, 'Figure1D_onOffSpectra.eps'));



Published with MATLAB® 8.0