Script to plot Figure 5, Simulation

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 script plots the two panels on the right side of figure 5. It simulates the response of an ECoG electrode in early visual cortex to periodic visual inputs.

ECoG signal model

This script

Asynchronous broadband signals are the principal source of the BOLD response in human visual cortex.

Winawer J, Kay KN, Foster BL, Parvizi J, Wandell BA

Code by Jonathan Winawer, 2012

Overview

The simulation assumes that there are ON periods (containing periodic stimuli) and OFF periods (no inputs). The responses are the sum of these components:

```
Response(ON)  = Evoked + Induced + Spontaneous
Response(OFF) = Spontaneous
```

Each component is generated by a subroutine that takes as input (a) one or more time varying spike rates and (b) one or more distrbitions of synaptic weights, and produces as output a single time varying voltage. For the ON responses, the voltages from each component (Evoked, Induced, Spontaneous) are summed.

The subroutine that converts rates to voltage (**ecogSimulate**) is adapted from equations 3 and 4 in Miller et al, 2009 (PLoS Computational Biology).

Path to save the eps figures

```
savepth   = fullfile(ecogPRFrootPath, 'scratch');
```

Set the random seed so that the simulation is reproducible

```
s = RandStream('mt19937ar','Seed',1);
RandStream.setGlobalStream(s);
```

Experiment parameters

```
% Choose these to match the experimental data sets
stimFrequency = 15; % number of contrast reversals per second during ON phase
nTrials       = 72; % number of trials
trialDur      = 1;  % seconds
```

Circuit parameters

```
% Values for alpha and tau from: Miller et al, 2009 (PLOS Comp Biology)
alpha   = 0.100;    % time constant of neural integrator (s)
tau     = 0.0023;   % time constant of current response function induced by PSP (s

evokedRate      = 15;  % 15 spikes per s per synapse.
inducedRate     = 30;  % 30 spikes per s per synapse.
spontaneousRate = 10;  % 10 spikes per s per synapse.
```

Analysis parameters

```
calcPower = true;  % Plot spectra as power (squared amplitude) or amplitude
useHann   = false; % Use a Hanning window for spectral analysis
dt        = .001;  % temporal sampling rate (s)
```

Derived parameters

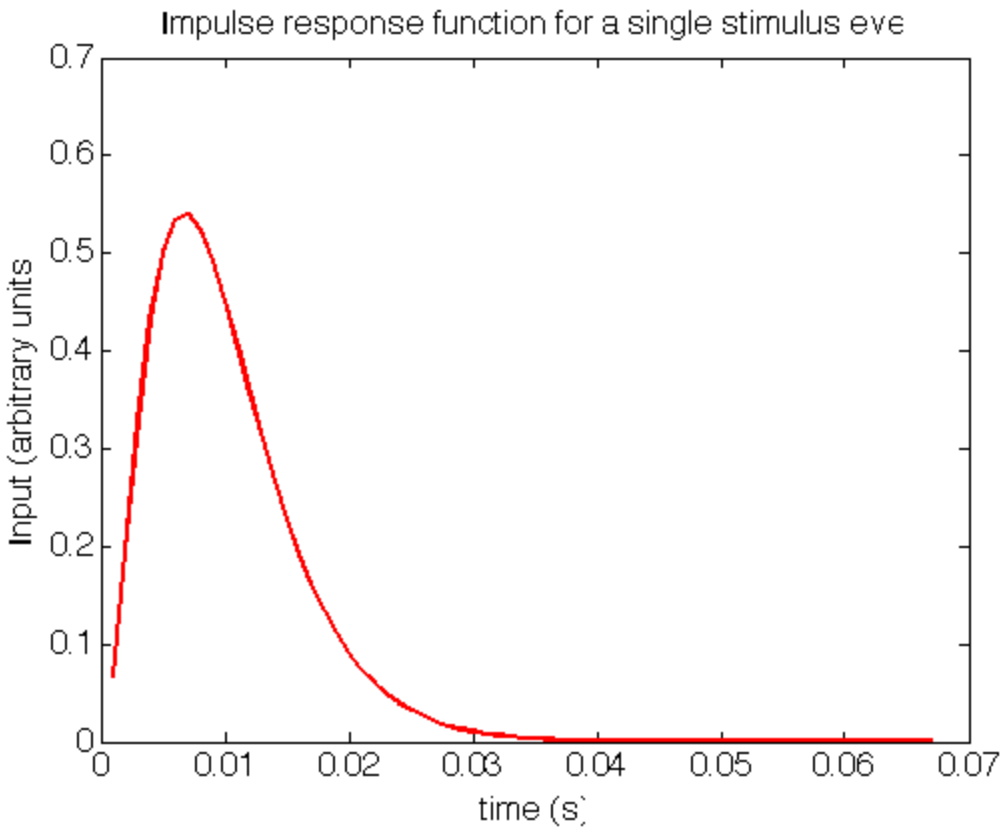
```
t  = dt:dt:trialDur;            % time vector for a single trial (s)
nt = length(t);                 % number of time samples in one trial
ntEvent = round(1/(stimFrequency*dt));  % number of time samples between stimulus
f  = (0:length(t)-1)/max(t);    % temporal frequencies for a single trial

% Pad the time vector so that the circuit has a chance to settle. Signals
% at time values < 0 will be discarded.
tpadded = -1:dt:max(t);
ntpadded = length(tpadded); % number of samples in padded time vector
```

Evoked response function

```
% Make a stimulus vector (ones for stimulus events, zeros elsewhere)
stimEvents = false(1,ntpadded);
stimEvents(1:ntEvent:ntpadded) = 1;

% Impulse response function (IRF) for transient (evoked) signals. This will
% be used to make two time-varying rates as inputs to the simulator, one
% input for the excitatory pool, and the same input (but slightly delayed)
% for the inhibitory pool. We use a gamma function for the IRF.
params = [1.5 3];
irfSupport = (1:ntEvent)/ntEvent*30;
irf = ( (irfSupport./params(1)).^(params(2)-1) ).*exp(-irfSupport./params(1));

% Convolve the stimulus events with impulse response function
stimOn  = conv2(double(stimEvents), irf, 'full');
stimOn  = stimOn(1:ntpadded);

% OFF inputs are the same as ON, but delayed by 25 ms
shift = round(.025/dt);
inds = 1+ mod((1:length(stimOn))-shift, length(stimOn));
stimOff = stimOn(inds);
```

Plot the impulse response function (input for the evoked response)

```
figure; set(gcf, 'Color', 'w'); set(gca, 'FontSize', 16)
plot(dt*(1:length(irfSupport)), irf, 'r-', 'LineWidth', 2);
xlabel('time (s)'); ylabel('Input (arbitrary units)');
title('Impulse response function for a single stimulus even')
snapnow;
```



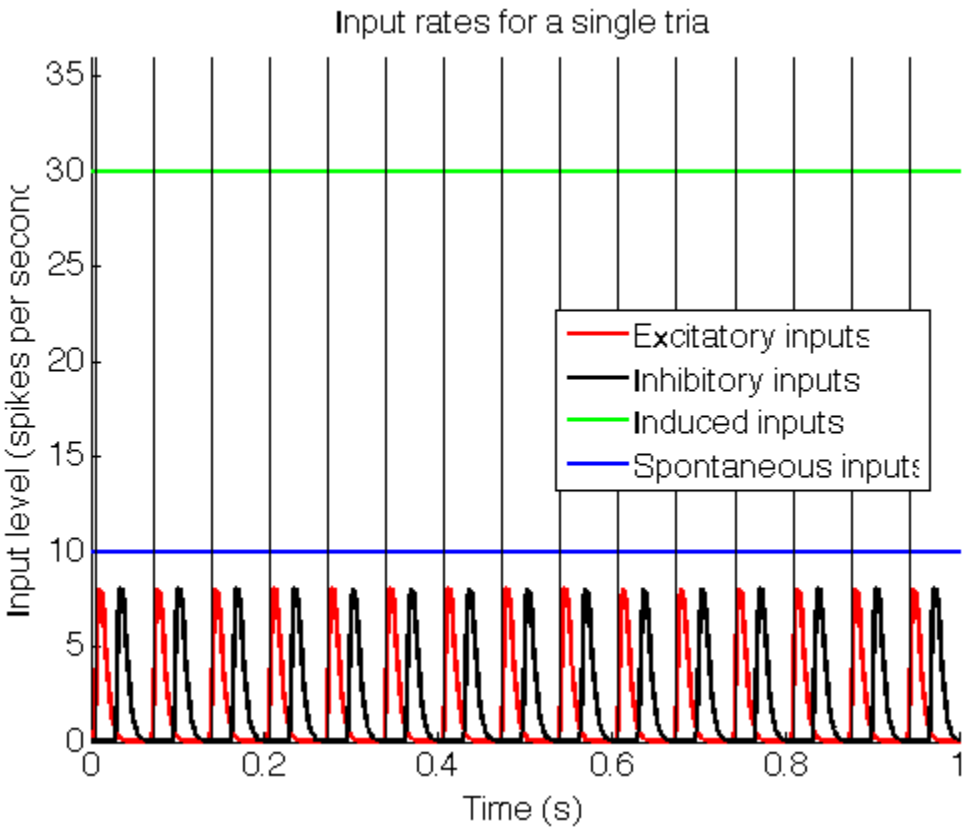
Plot the inputs to the model

Plot lines include:

- the stimulus-evoked input rates to the excitatory pool
- the stimulus-evoked input rates to the inhibitory pool
- the induced signal input rate (to a mixed E/I pool)
- the spontaneous input rate (to a mixed E/I pool)
- grid lines to indicate stimulus events

```
figure; set(gcf, 'Color', 'w');
set(gca, 'FontSize', 16, 'Xlim', [0 max(t)]); hold on;
%

plot(t, stimOn(tpadded>0) * evokedRate,'r-', ...
    t, stimOff(tpadded>0)*evokedRate, 'k-', ...
    t, ones(size(t))*inducedRate, 'g-', ...
    t, ones(size(t))*spontaneousRate, 'b-', ...
    'LineWidth', 2);

% scale the y-axis appropriately
yl = [0 1.2*max([evokedRate inducedRate spontaneousRate])]; ylim(yl)
```

```
% plot the stimulus events
plot([1; 1] * tpadded(stimEvents), repmat(yl, sum(stimEvents),1)', 'k-')
xlabel('Time (s)'); ylabel('Input level (spikes per second)')
title('Input rates for a single trial')
legend({'Excitatory inputs', 'Inhibitory inputs',...
    'Induced inputs', 'Spontaneous inputs'}, 'Location', 'Best');
```



Calibrate the noise generator

The simualtor takes in a spike rate and outputs a voltage. The scaling depends on the spike rate, as well as some fairly arbitrary things like the number of synapses, the ampltidue of the synaptic currents and so forth. We would like to amplify the output to get it into the range of the observed signals, which is on the order of sd = 40 µV for the off condition ('spontanous activity'). We choose a spike rate for the spontanous activity of 10 Hz. So we calculate what scale factor we need such that a spike rate input of 10 Hz generates a time seris with sd = 40 µV.

```
% set the spontaneous level (spikes / s)
rate = 10;

% number of calibration trials
n = 100;

% initialize a vector to store the sd of the time series of many iterations
% of spontaneous activty
sd = zeros(1,n);
```

```
% synaptic distribution is unifrom random on [-1 1], centered at exactly 0
synapseFunc = @(x) zeromean(2*rand(x,1));

% 100 calibration trials
for ii = 1:n
    sd(ii) = std(ecogSimulate(tpadded, rate, synapseFunc,[],[],alpha,tau));
end

% this is the scale factor we need so get, on average, sd = 40 given input
% rate = 10
noiseAmp  = 40/mean(sd);
```

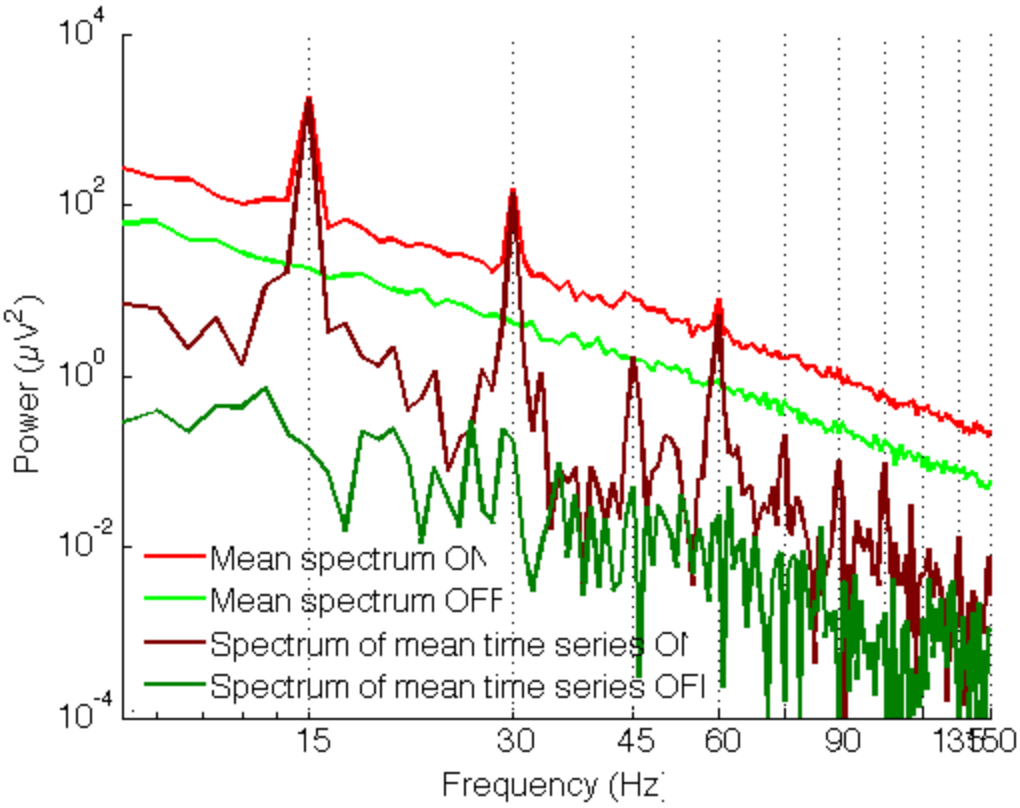
Run the simualtion

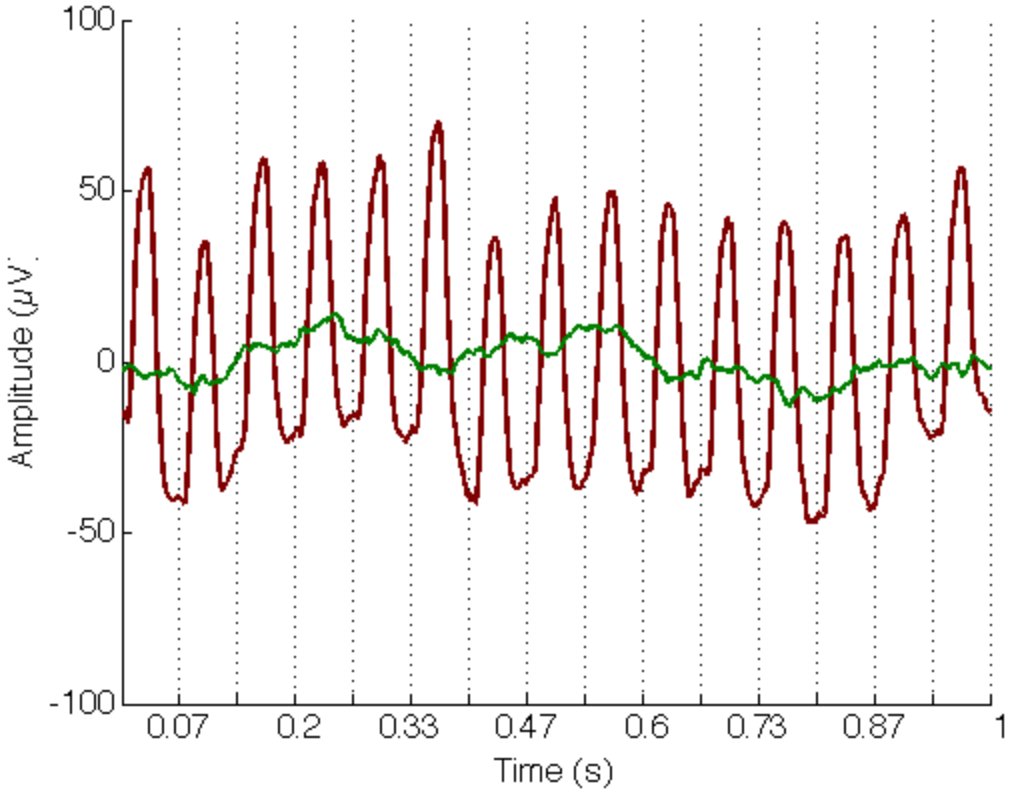
```
on.signal  = zeros(nt, nTrials);
off.signal = zeros(nt, nTrials);

for trial = 1:nTrials

    % --- OFF: Spontanoues only ----------------------------------------
    Spontaneous = ecogSimulate(tpadded, spontaneousRate, [], [],[],alpha, tau);
    off.signal(:, trial) = Spontaneous*noiseAmp;
    % ------------------------------------------------------------------


    % --- ON: Spontaneous + Evoked + Induced  --------------------------

    % Evoked response. Two populations of synapses, all positive or all
    % negative. We define the synapse distribution as all positive. The
    % simulator will assume that if there are two rates, the synapse
    % distribition for the second rate is the additive inverse of the first
    % rate.
    Evoked = ecogSimulate(tpadded, stimOn * evokedRate, @(x) rand(x, 1), ...
        stimOff * evokedRate, [], alpha, tau);

    % Induced and spontaneous response (when stimulus is ON). Each is
    % derived from a single population of synapes, distributed equally
    % about zero (half excitatory, and half inhibitory).
    Induced     = ecogSimulate(tpadded, inducedRate,     [], [],[],alpha, tau);
    Spontaneous = ecogSimulate(tpadded, spontaneousRate, [], [],[],alpha, tau);

    % Combined Evoked and Induced and Spontaneous
    on.signal(:, trial)  = (Evoked + Induced + Spontaneous)*noiseAmp;
    % ------------------------------------------------------------------

end
```

Summarize spectra and time series

```
% compute means across trials
[on, off] = ecogCalcOnOffSpectra(on, off, useHann, calcPower);
```

```
% Plot time series and spectra
fH = ecogPlotOnOffSpectra(on, off, t, stimFrequency, calcPower);
%
%
% %% Save
%
% hgexport(fH(1), fullfile(savepth, 'Figure5_Simulation_TimeSeries.eps'));
% hgexport(fH(2), fullfile(savepth, 'Figure5_Simulation_Spectra.eps'));
```





Published with MATLAB® 8.0