Analyzing behavior-relevant physiology data

Jianing Yu Feb 7 2020

Table of Contents

[Extract behavioral events 2](#_Toc32745975)

[Extracting data from NS6 3](#_Toc32745976)

[Spike sorting 4](#_Toc32745977)

[Check the quality of spike sorting 5](#_Toc32745978)

[Local field potential: 9](#_Toc32745979)

[Extract event times from MED 10](#_Toc32745980)

[Useful codes: 12](#_Toc32745981)

[DetectSpikes.m 12](#_Toc32745982)

# Extract behavioral events

e.g., **filename** = ‘data001.nev

openNEV(**filename**, 'report', 'read') 🡪 open ‘datafile###.nev’, create “datafile###.mat”

EventOut = DIO\_Events(NEV, trange) 🡪 create

The result is **EventOut.mat**

1. BuildArray.m: this will take care of reading all behavior events.

# Extracting data from NS6

Filename = ‘data001.ns6’

**Run ‘DetectSpikes.m’**

Two outputs: **chdat##.mat, chdat##\_spikes.mat**

Note that data might be separated into segments (see NS6.MetaTags). The time stamp of each segment is defined in NS6.MetaTags.Timestamp

A screenshot of a cell phone

Description automatically generated

With the following codes we extract raw data:

for i =1:length(live\_ch)

data = [];

index =[];

for k =1:length(NS6.Data)

data= [data double(NS6.Data{k}(i, :))]; % data is a necessary vector

index = [index [0:length(double(NS6.Data{k}(i, :)))-1]\*1000/Fs+NS6.MetaTags.Timestamp(k)\*1000/Fs];

end;

[index, unique\_index]=unique(index);

data = data(unique\_index);

savefile = ['chdat' num2str(live\_ch(i)) '.mat']; % name of files

save(savefile, 'data', 'index');

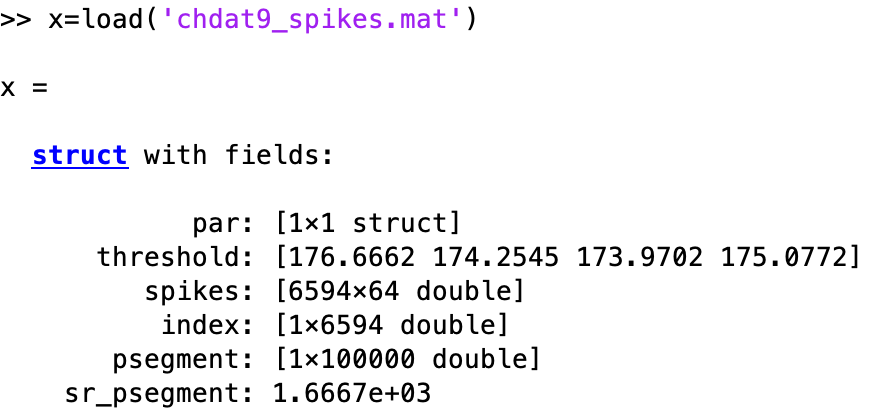
filelist{i}=savefile;

end;

Data and index (in ms) are stored in ‘chdata##.mat’. This format can be read by wave-clus’ *Get\_spikes* Get\_spikes(tosort\_list,'parallel',false,'par',param) which produces a file with spike times: chspikes = load(['chdat' num2str(functional\_channels(i)) '\_spikes.mat']);

Run the attached scripts (**DetectSpikes.m**) to extract data from NS6 file and make several figures to summarize the data (this is before spike sorting).

Important notice: spike index (in ms) from chdata#\_spikes.mat (produced by get\_spikes) does not consider the time stamp of the first raw data point, which might not be 1. This has to be corrected later (after spike sorting).



# Spike sorting

Spike sorting is performed using wave-clus.:

<https://github.com/csn-le/wave_clus>

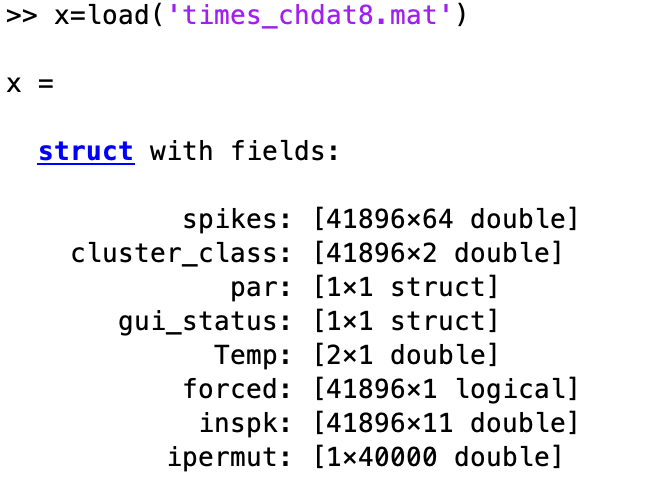
A screenshot of a cell phone

Description automatically generated

The result after “save clusters” is ***times\_chdat#.mat***

Spike times and cluster identify can be found in cluster\_class.

Note that spike time needs to be corrected with reference to timestamp of the first data point collected (in NS6.MetaTags.Timestamps(1) or raw.index(1)).



Rules for single unit (S)

# Check the quality of spike sorting

function CheckWaveClusSorting(chname)

% chname can ben chdat2

% raw data

raw = load(['chdat' chname]); % if chname is 2, then load('chdat2.mat')

% spike detection

spkall = load(['chdat' chname '\_spikes.mat']); % load spike times from detection, including all that pass thredhold

% par: [1√ó1 struct]

% threshold: [151.8461 150.4016 150.6069 150.6656]

% spikes: [11979√ó64 double]

% index: [1√ó11979 double]

% psegment: [1√ó100000 double]

% sr\_psegment: 1.6667e+03

spksort = load(['times\_chdat' chname '.mat'])

% spikes: [953√ó64 double]

% cluster\_class: [953√ó2 double]

% par: [1√ó1 struct]

% gui\_status: [1√ó1 struct]

% Temp: 13

% forced: [1√ó953 logical]

% inspk: [953√ó11 double]

% ipermut: [1√ó953 double]

% spksort.cluster\_class(:, 1) is the class definition, 0 unsorted, 1 sorted

figure(23); clf

set(gcf, 'unit', 'centimeters', 'position',[2 2 20 16], 'paperpositionmode', 'auto' )

ha=axes;

tdur=5;

tmax = floor(raw.index(end)/1000-tdur);

toplot=1;

ha2 = subplot(2, 2, [3]);

set(ha2, 'nextplot', 'add')

% plot sorted spikes

clusters = unique(spksort.cluster\_class(:, 1));

nclusters = length(clusters);

allcolors = varycolor(3);

allcolors = [

0 0 0

0 0 1

1 0 0

0 1 0

allcolors];

for k =1:nclusters-1

if size(spksort.spikes(spksort.cluster\_class(:, 1)== k, :), 1)>20

spkwave\_avg = mean(spksort.spikes(spksort.cluster\_class(:, 1)== k, :), 1);

plot(spkwave\_avg, 'color', allcolors(k+1, :), 'linewidth', 1);

end;

end;

axis tight

while toplot

ha1 = subplot(2, 2, [1 2]);

set(ha1, 'ylim', [-800 800], 'nextplot', 'add')

xlabel('Time (s)')

ha3 = subplot(2, 2, 4);

set(ha3, 'xlim', [0 25], 'nextplot', 'add')

xlabel('Inter-spike interval (ms)')

ylabel ('Counts')

tbeg = randperm(tmax, 1);

tend =[tbeg+tdur];

index\_raw=find(raw.index>=tbeg\*1000 & raw.index<=tend\*1000);

[b\_detect,a\_detect] = ellip(4,0.1,40,[250 10000]\*2/30000); % high pass

data\_plot\_hp = filtfilt(b\_detect, a\_detect, raw.data(index\_raw)); % band pass 2-200 hz

axes(ha1)

plot(raw.index(index\_raw)/1000, data\_plot\_hp, 'k');

% plot detection

index\_detection = find(spkall.index>=tbeg\*1000 & spkall.index<=tend\*1000);

spk\_peaks = min(spkall.spikes(index\_detection, :), [], 2);

plot(spkall.index(index\_detection)/1000, spk\_peaks, 'mo','markersize', 3)

% Either way, wave\_clus generates a file times\_filename.mat, with a variable cluster\_class of two columns: the first column is the class of the spike and the second one is the spike time in ms.

spk\_peaks\_max = prctile(min(spksort.spikes(find(spksort.cluster\_class(:, 1) ~= 0), :),[], 2), 1);

for k =1:nclusters-1

if size(spksort.spikes(spksort.cluster\_class(:, 1)== k, :), 1)>20

axes(ha1)

spk\_time\_sort = (spksort.cluster\_class(find(spksort.cluster\_class(:, 1) == k), 2)); % in ms

spk\_peaks\_sort = prctile(spksort.spikes(find(spksort.cluster\_class(:, 1) == k), :)', 5);

if ~isempty(find(spk\_time\_sort>=tbeg\*1000 & spk\_time\_sort<=tend\*1000))

plot([spk\_time\_sort(spk\_time\_sort>=tbeg\*1000 & spk\_time\_sort<=tend\*1000)/1000 spk\_time\_sort(spk\_time\_sort>=tbeg\*1000 & spk\_time\_sort<=tend\*1000)/1000]', [spk\_peaks\_max-100 spk\_peaks\_max],...

'linewidth', 2, 'color', allcolors(k+1, :));

end;

% ISI histogram

edges = [0:1:25];

[ncounts{k}] = histcounts(diff(spk\_time\_sort), edges);

axes(ha3)

plot(edges(2:end), ncounts{k}, 'color', allcolors(k+1, :), 'linewidth', 1)

end;

end;

set(ha1, 'ylim', [-800 800], 'nextplot', 'add', 'xlim', [tbeg tend])

reply = input('Keep showing Y/N [Y]:', 's');

if isempty(reply) || strcmp(reply, 'Y')|| strcmp(reply, 'y')

toplot =1;

else

toplot=0;

end;

end;

reply = input('Save this pic? Y/N [Y]:', 's');

if isempty(reply) || strcmp(reply, 'Y')|| strcmp(reply, 'y')

print (gcf,'-dpng', ['waveclus\_chdat' chname])

end;

A close up of a device

Description automatically generated

# Extract event times from MED

**GetMedTime.m**

bout=track\_training\_progress\_advanced('2020-01-23\_15h58m\_Subject Lucky.txt');

behout = bout;

save behout behout

**GetBehaviorTimes.m**

% bout=track\_training\_progress\_advanced('2020-01-23\_15h58m\_Subject Lucky.txt');

load('C:\Users\jiani\OneDrive\Work\Physiology\Data\Subjects\Lucky\20200123\B\_LUCKY\_20200123\_155828.mat');

% this yields 'b'

all\_press\_times = b.PressTime\*1000; % in s

all\_press\_FPs = b.FPs;

all\_release\_times = b.ReleaseTime\*1000;

all\_correct\_index = b.Correct;

load('EventOut.mat');

leverpress\_br = EventOut.Onset{5};

%

figure;

plot(all\_press\_times, 5, 'ro');

hold on

plot(leverpress\_br, 5, 'ko')

n\_allpress = length(all\_press\_times);

n\_pressbr = length(leverpress\_br);

dtime = [];

figure;

ha=subplot(2, 1, 1); set(ha, 'nextplot','add')

for i = 1:n\_allpress-n\_pressbr

dtime=[dtime; std(abs(all\_press\_times(i:n\_pressbr+i-1)-all\_press\_times(i)-leverpress\_br'))];

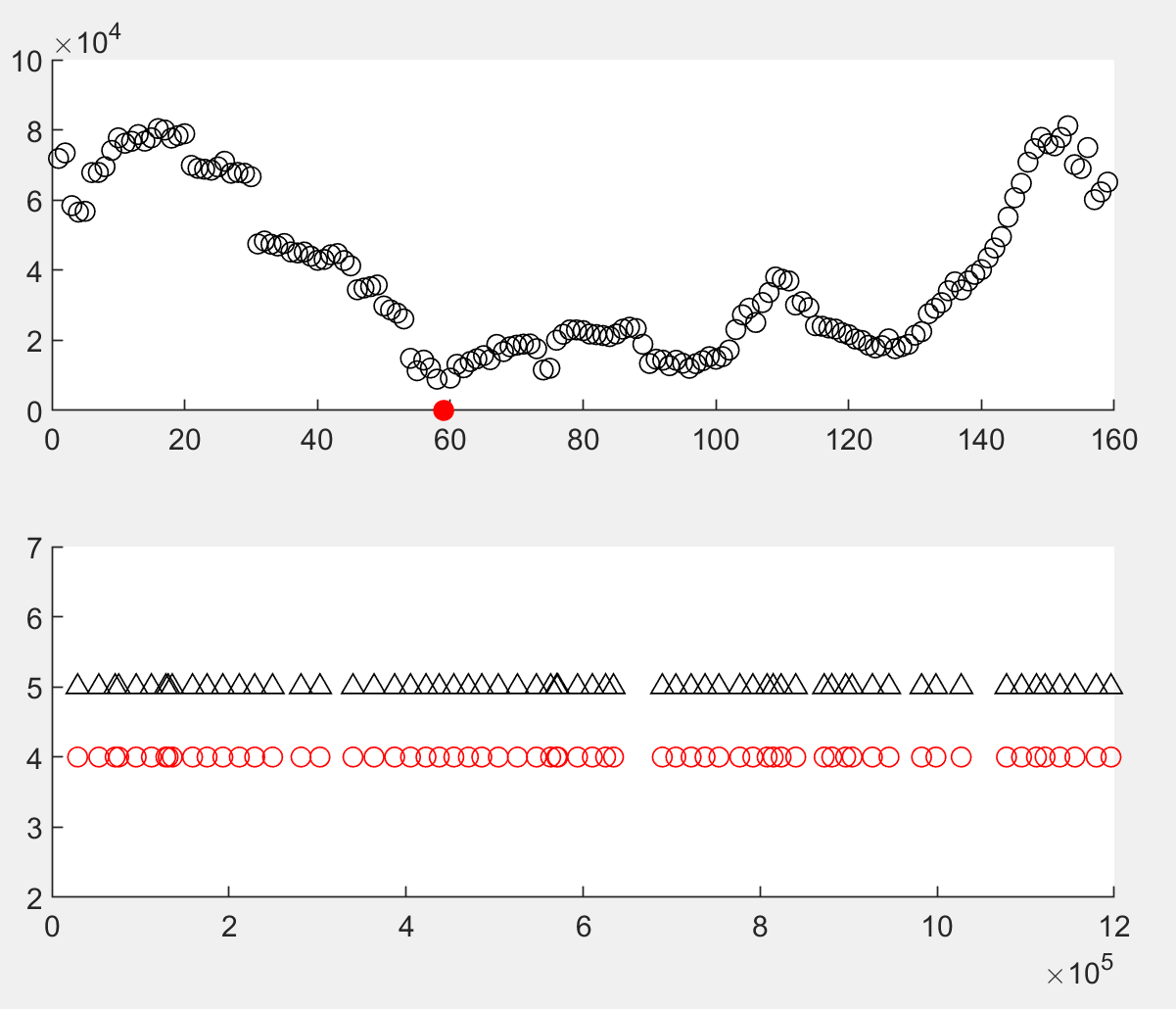
plot(i, dtime(i, :), 'ko')

end;

[~, ind\_offset] = min(dtime);

plot(ind\_offset, dtime(ind\_offset), 'ro', 'markerfacecolor', 'r')

toffset = all\_press\_times(ind\_offset)-leverpress\_br(1); % offset between first press in datafile### and the corresponding time in MED file



ha2=subplot(2, 1, 2);

set(ha2, 'nextplot','add')

plot(leverpress\_br, 4, 'ro')

plot(all\_press\_times(ind\_offset:ind\_offset+n\_pressbr-1)-toffset, 5, 'k^');

set(gca, 'ylim', [2 7]);

all\_tones = bout.TimeTone\*1000-toffset;

tonetime\_br =all\_tones(all\_tones>0 & all\_tones<leverpress\_br(end));

figure;

plot(tonetime\_br, 2, 'ko');

hold on

plot(leverpress\_br,2,'r^');

if isempty(find(strcmp(EventOut.EventsLabels, 'Trigger')))

EventOut.EventsLabels{end+1} = 'Trigger';

EventOut.Onset{end+1} = tonetime\_br;

end;

% Correct, Premature and late index

indpress\_br =ind\_offset:ind\_offset+n\_pressbr-1;

[~, ind\_correct] = intersect(indpress\_br, b.Correct);

[~, ind\_premature] = intersect(indpress\_br, b.Premature);

[~, ind\_late] = intersect(indpress\_br, b.Late);

EventOut.CorrectIndex = ind\_correct;

EventOut.PrematureIndex = ind\_premature;

EventOut.LateIndex= ind\_late;

EventOut.FPs = b.FPs(indpress\_br);

save EventOut EventOut

Build behavior array

**BuildArray.m**

This produces an “r” array

r =

Meta: [1×1 struct]

Behavior: [1×1 struct]

Units: [1×1 struct]

Video: [1×1 struct]

r.Behavior

Labels: {1×10 cell}

LabelMarkers: [1 2 3 4 5 6 7 8 9 10]

CorrectIndex: [41×1 double]

PrematureIndex: [4×1 double]

LateIndex: [12×1 double]

Foreperiods: [64×1 double]

EventTimings: [120727×1 double]

EventMarkers: [120727×1 double]

r.Units

Channels: [1×32 double]

Profile: {9×2 cell}

Definition: {1×3 cell}

SpikeNotes: [12×3 double]

SpikeTimes: [1×12 struct]

r.Video

Files: {1×2 cell}

FrameTimes: {1×2 cell}

FrameIndex: {1×2 cell}

ROI: [1×1 struct]

r.FP

channels: [14×1 double]

samplerate: 1000

data: {1×14 cell}

r.Units

ans =

struct with fields:

Channels: [1×32 double]

Profile: {9×2 cell}

Definition: {1×3 cell}

SpikeNotes: [12×3 double]

SpikeTimes: [1×12 struct]

# Local field potential:

>> load('RTarray.mat')

>> BehaviorFP\_LR(r)

# Useful codes:

### DetectSpikes.m

%% generate NS6 file.

% openNSx('read', 'report')

live\_ch = [1:16 33:48]; % all live channels for 2 16-wire arrays

Fs = NS6.MetaTags.SamplingFreq;

% This is raw data of c13

% chdat = cell(1, length(live\_ch));

pardata = 0;

if pardata ==1

filelist =[];

for i =1:length(live\_ch)

data = [];

index =[];

for k =1:length(NS6.Data)

data= [data double(NS6.Data{k}(i, :))]; % data is a necessary vector

index = [index [0:length(double(NS6.Data{k}(i, :)))-1]\*1000/Fs+NS6.MetaTags.Timestamp(k)\*1000/Fs];

end;

[index, unique\_index]=unique(index);

data = data(unique\_index);

savefile = ['chdat' num2str(live\_ch(i)) '.mat']; % name of files

save(savefile, 'data', 'index');

filelist{i}=savefile;

end;

end;

% functional\_channels = [2 3 4 6 7 8 9 10 11 12 13 14 33 35 37 43 44 45 46];

functional\_channels = [1:16 33:48];

nospike\_channels = [1 ];

pos\_detection =[13]; % for channel 13, use positive detection

plot\_trange = [118 123]; % plot 10 sec of data

figure(20); clf;

set(gcf, 'unit', 'centimeters', 'position',[2 2 35 25], 'paperpositionmode', 'auto' )

ha1 = subplot(1, 2, 1)

set(ha1, 'xlim', [plot\_trange(1)\*1000 plot\_trange(2)\*1000], 'ylim', [-500-16\*300 500], 'nextplot', 'add');

title(['ch#' '1-16, band pass 2-200 Hz'])

line([plot\_trange(1)\*1000 plot\_trange(1)\*1000+2000], [0 0], 'color', 'k', 'linewidth', 3)

text(plot\_trange(1)\*1000+100, 200, '2sec', 'color', 'k', 'fontsize', 12)

axis off

ha2 = subplot(1, 2, 2)

set(ha2, 'xlim', [plot\_trange(1)\*1000 plot\_trange(2)\*1000], 'ylim', [-500-16\*300 500], 'nextplot', 'add');

title(['ch#' '33-48, band pass 2-200 Hz'])

axis off

figure(21); clf;

set(gcf, 'unit', 'centimeters', 'position',[2 2 35 25], 'paperpositionmode', 'auto' )

figure(22); clf;

set(gcf, 'unit', 'centimeters', 'position',[2 2 35 25], 'paperpositionmode', 'auto' )

ha3 = subplot(1, 2, 1)

set(ha3, 'xlim', [plot\_trange(1)\*1000 plot\_trange(2)\*1000], 'ylim', [-100-16\*300 200], 'nextplot', 'add');

title(['ch#' '1-16, high pass 2-200 Hz'])

line([plot\_trange(1)\*1000 plot\_trange(1)\*1000+2000], [0 0], 'color', 'k', 'linewidth', 3)

text(plot\_trange(1)\*1000+100, 200, '2sec', 'color', 'k', 'fontsize', 12)

axis off

ha4 = subplot(1, 2, 2)

set(ha4, 'xlim', [plot\_trange(1)\*1000 plot\_trange(2)\*1000], 'ylim', [-100-16\*300 200], 'nextplot', 'add');

title(['ch#' '33-48, high pass 2-200 Hz'])

axis off

for i =1:length(functional\_channels);

indx = find(live\_ch==functional\_channels(i));

tosort\_list{i, 1} = ['chdat' num2str(functional\_channels(i)), '.mat'];

param.stdmin = 4;

param.sr = 30000;

if ~isempty(find(pos\_detection==functional\_channels(i)))

param.detection = 'pos';

else

param.detection = 'neg';

end;

param.detect\_fmin = 250; % high pass filter for detection

param.detect\_fmax = 5000; % low pass filter for detection (default 1000)

param.detect\_order = 4; % filter order for detection

param.sort\_fmin = 250; % high pass filter for sorting

param.sort\_fmax = 5000; % low pass filter for sorting (default 3000)

Get\_spikes(tosort\_list,'parallel',false,'par',param);

chspikes = load(['chdat' num2str(functional\_channels(i)) '\_spikes.mat']);

% struct with fields:

% par: [1√ó1 struct]

% threshold: 147.9486

% spikes: [584√ó64 double]

% index: [1√ó584 double]

% psegment: [1√ó100000 double]

% sr\_psegment: 1.6667e+03

load(['chdat' num2str(functional\_channels(i)) '.mat']); % index and data

data\_plot = data(index>=plot\_trange(1)\*1000 & index<=plot\_trange(2)\*1000);

index\_plot = index(index>=plot\_trange(1)\*1000 & index<=plot\_trange(2)\*1000);

% remove line noise

d = designfilt('bandstopiir','FilterOrder',2, ...

'HalfPowerFrequency1',48,'HalfPowerFrequency2',52, ...

'DesignMethod','butter','SampleRate',param.sr);

data\_plot2 = filtfilt(d,data\_plot);

[b\_detect,a\_detect] = butter(2, [2 200]\*2/param.sr, 'bandpass'); % field potential

data\_plot\_band = filtfilt(b\_detect, a\_detect, data\_plot2); % band pass 2-200 hz

[b\_detect,a\_detect] = ellip(param.detect\_order,0.1,40,[250 5000]\*2/param.sr); % high pass

data\_plot\_hp = filtfilt(b\_detect, a\_detect, data\_plot); % band pass 2-200 hz

figure(21)

if i<=16

axes(ha1)

addon = -300\*i;

else

axes(ha2)

addon = -300\*(i-16);

end;

plot(index\_plot, data\_plot\_band/5+addon, 'k')

hold on

spikeindex\_trange = find(chspikes.index>=plot\_trange(1)\*1000 & chspikes.index<=plot\_trange(2)\*1000);

if ~isempty(spikeindex\_trange)

line([chspikes.index(spikeindex\_trange); chspikes.index(spikeindex\_trange)], [-50 50]/5+addon, 'color', 'r', 'linewidth', 1.5)

%

end;

text(plot\_trange(2)\*1000-1, addon+50, ['#' num2str(functional\_channels(i))], 'color', 'r', 'fontsize', 12);

xlabel('Time (ms)')

ylabel ('Artificial unit')

figure(22)

if i<=16

axes(ha3)

addon = -300\*i;

else

axes(ha4)

addon = -300\*(i-16);

end;

plot(index\_plot, data\_plot\_hp/2+addon, 'k')

hold on

if ~isempty(spikeindex\_trange)

line([chspikes.index(spikeindex\_trange); chspikes.index(spikeindex\_trange)], [-50 50]/2+addon, 'color', 'r', 'linewidth', 1.5)

%

end;

text(plot\_trange(2)\*1000-1, addon+50, ['#' num2str(functional\_channels(i))], 'color', 'r', 'fontsize', 12);

xlabel('Time (ms)')

ylabel ('Artificial unit')

axis off

figure(21)

subplot(4, 8, indx)

title(['ch#' num2str(functional\_channels(i))])

set(gca, 'xlim', [0 64]/30, 'nextplot', 'add', 'ylim', [-800 600]);

if size(chspikes.spikes, 1)>100

toplot = randperm(size(chspikes.spikes, 1), 100);

else

toplot = [1:100];

end;

waveindex = [1:64]/30;

plot(waveindex, chspikes.spikes(toplot, :), 'r')

xlabel('Time (ms)')

ylabel ('Artificial unit')

axis off

end;

print (20,'-dpng', ['LFP\_spike\_all' ])

print (21,'-dpng', ['spike\_wave\_all'])

print (22,'-dpng', ['HP\_signal\_all'])