Table of Contents

GRAPHICAL USER INTERFACE

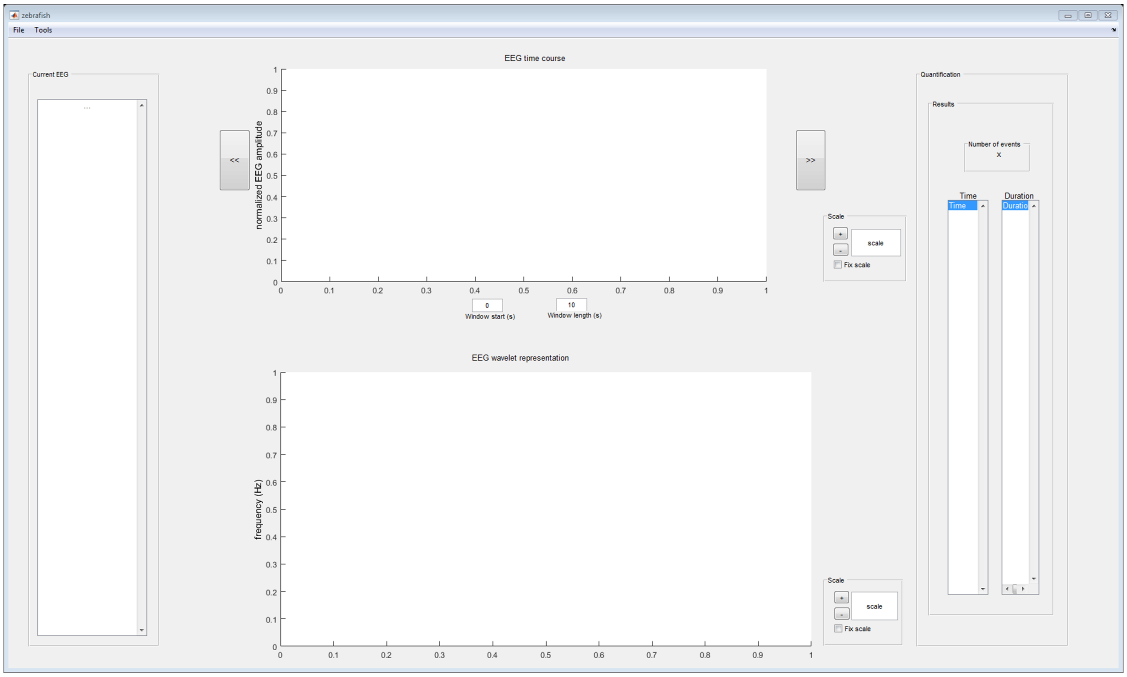
```
% % 1. PURPOSE:
% For visualizing single-channel LFP signals, their time-frequency
% characteristics; and detect epileptic seizures in zebrafish larvae
% Note that the algorithm (feature extraction and classifier)
% was optimized for detecting seizures in a specific genetic and
% chemical model of epilepsy as described in the reference paper.
% For optimal performance on different zebrafish models and signals
% recorded under different conditions, the classifier needs to be
% retrained on data from a similar setting.

% 2. LANGUAGE:
% Matlab 2014b (GUI) and 2016 (Classifier training)

% 3. REFERENCE:
% Borbála Hunyadi, Aleksandra Siekierska, Jo Sourbron, Daniëlle
% Copmans, Peter A.M. de Witte,
% Automated analysis of brain activity for seizure
% detection in zebrafish models of epilepsy,
```

```matlab
% Journal of Neuroscience Methods, Volume 287, 1 August 2017,
% Pages 13-24, ISSN 0165-0270,
% https://doi.org/10.1016/j.jneumeth.2017.05.024.

% 4. OWNER AND DEVELOPER
% Owner: KU Leuven
% Developer: Borbala Hunyadi, Stadius, ESAT, KU Leuven
% Contact person: Borbala Hunyadi, Stadius, ESAT, KU Leuven

%%%%%%%%%%%%%%%%%%%%

% Description of code:
% The functions below initialize the GUI layout and default function
% as created by GUIDE. It also specifies some global parameters
% specific for the application.

% To run the GUI, simply type zebrafish in the Matlab command line and
% press Enter. The GUI will initialize.

function varargout = zebrafish(varargin)
%

% Last Modified by by Borbála Hunyadi 28/06/2017
% STADIUS, Department of Electrical Engineering, KU Leuven

gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
    'gui_Singleton',  gui_Singleton, ...
    'gui_OpeningFcn', @zebrafish_OpeningFcn, ...
    'gui_OutputFcn',  @zebrafish_OutputFcn, ...
    'gui_LayoutFcn',  [] , ...
    'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end

% End initialization code - DO NOT EDIT

% --- Executes just before zebrafish is made visible.
function zebrafish_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to zebrafish (see VARARGIN)

%set(handles.AnalyzeBatchMenuItem, 'Enable', 'on');
```

```
%Initial setting Zebrafish
handles.fs=1000;
handles.window=10*handles.fs; % 10 s long window
handles.start=1;
handles.step=10*handles.fs; %initial length of EEG window is
 10s=10000ms
handles.ylimits1=[];
handles.fixaxes1=0;
handles.fixaxes2=0;
handles.climit=[1 64];
% Choose default command line output for zebrafish
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% This sets up the initial plot - only do when we are invisible
% so window can get raised using zebrafish.
if strcmp(get(hObject,'Visible'),'off')

    axes(handles.axes1);
    ylabel('normalized EEG amplitude')
    axes(handles.axes2);
    ylabel('frequency (Hz)')

end

% UIWAIT makes zebrafish wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.

function axes1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to axes1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
 called

% Hint: place code in OpeningFcn to populate axes1


function varargout = zebrafish_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;
% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
```

```
% handles    structure with handles and user data (see GUIDATA)


% --- Executes when figure1 is resized.
function figure1_ResizeFcn(hObject, eventdata, handles)
% hObject    handle to figure1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --- Executes on selection change in popupmenu1.
function popupmenu1_Callback(hObject, eventdata, handles)

function popupmenu1_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
 get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

set(hObject, 'String',
 {'plot(rand(5))', 'plot(sin(1:0.01:25))', 'bar(1:.5:10)', 'plot(membrane)', 'surf

function popupmenu2_Callback(hObject, eventdata, handles)

function popupmenu2_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
 get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


function listbox3_Callback(hObject, eventdata, handles)

function listbox3_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
 get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function listbox4_Callback(hObject, eventdata, handles)

function listbox4_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
 get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function text3_CreateFcn(hObject, eventdata, handles)
```

```
function text7_Callback(hObject, eventdata, handles)

function text7_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
 get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```



I. FILE MENU

```
% --------------------------------------------------------------------

function FileMenu_Callback(hObject, eventdata, handles)
```

1. Open an EEG file

The expected format is a .txt file with a single column of values, correspoding to each LFP sample. See example.txt for an example EEG/LFP file. WARNINING: The software assumes a sampling rate of 1000Hz.

```
% --------------------------------------------------------------------

function open_file_Callback(hObject, eventdata, handles)

% hObject    handle to open_file (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

[FileName,PathName] = uigetfile('*.txt','Select an EEG
 file','MultiSelect','off');
```

```
if isequal(FileName,0)
    return
end
if isequal(FileName,0)
    return;
end

handles.eegFile=FileName;
eeg=csvread([PathName,FileName]);
handles.eeg=eeg(:,1);
handles.FileName=FileName;
set(handles.text7, 'String', (FileName));
disp('EEG data loaded...')

handles.start=1;
handles.stop=handles.start+handles.window-1;

msg=msgbox('Preprocessing, please wait...');
[handles.eegfilt,handles.wc, handles.eegplot]=processEEG(eeg,10);
handles.wc(isnan(handles.wc))=0;
handles.eegfilt=handles.eegfilt';

plotEEG( handles)
if ishandle(msg)
    delete(msg);
end
guidata(hObject,handles);
%plotting whole signal
%plot(handles.eeg);
```

2. Create a batch file of multiple LFP recordings

A batch file is a list of EEG/LFP files. It allows the user to run a certian analysis on multiple files together. The batch file is saved in .mat format and contains the name of the LFP files

```
% --------------------------------------------------------------------

function create_batch_Callback(hObject, eventdata, handles)

% hObject    handle to create_batch (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
[eegFiles,eegPath] = uigetfile('*.txt','Select EEG files for the
 batch','MultiSelect','on');
if isequal(eegFiles,0)
    return
end
BatchName = uiputfile('.mat','Specify a name for the batch file');
save(BatchName,'eegFiles','eegPath');
```

3. Open a batch file of LFP files, created using this GUI.

```
% --------------------------------------------------------------------

function open_batch_Callback(hObject, eventdata, handles)

[batchFile,batchPath] = uigetfile('*.mat','Select batch
 file','MultiSelect','off');
if isequal(batchFile,0)
    return
end
load([batchPath, batchFile]);
handles.eegFiles=eegFiles;
handles.eegPath=eegPath;
set(handles.text7, 'String', (eegFiles));

axes(handles.axes1);
axes(handles.axes2);

guidata(hObject,handles);
```

4. Save analysis results in a .csv file

```
function save_analysis_Callback(hObject, eventdata, handles)

headers = {'spike time','spike duration'};
cdir = getappdata(0,'currentdir');
[fn p]=uiputfile([cdir,'*.csv'], 'Save to');
if isequal(fn,0) | isequal(p,0)
    return
end

data=[handles.spiketime',handles.duration'];
cd(pwd);
savef = fullfile(p, fn);

fid = fopen(savef, 'w');
fprintf(fid, '%s;', headers{1});fprintf(fid, '%s\n', headers{2});
fclose(fid);
dlmwrite(savef,data,'-
append','roffset',1,'roffset',0,'delimiter',';') ;
```

5. Close GUI

```
% --------------------------------------------------------------------

function CloseMenuItem_Callback(hObject, eventdata, handles)

selection = questdlg(['Close ' get(handles.figure1,'Name') '?'],...
    ['Close ' get(handles.figure1,'Name') '...'],...
    'Yes','No','Yes');
```

```
if strcmp(selection,'No')
    return;
end

delete(handles.figure1)
```

II. TOOLS MENU

```
% --------------------------------------------------------------------

function ToolsMenu_Callback(hObject, eventdata, handles)
```

1. Seizure detection in single file

This option will run the seizure detection on the LFP file which has been loaded previously using the File menu > Open EEG option. The software prompts for 2 values, an energy threshold and a minimum seizure length value. Recommended values are filled in by default, or the user can specify an appropriate value. Energy threshold: the larger it is, the less candidate seizure segments will be selected. Extreme values are between 5-100, a reasonable values are 10-20. Minimum seizure length (in ms): the larger it is, the less candidate seizure segments will be selected. Typical values vary depending on the specific epilepsy model. The software also prompts the user to choose a classifier file (class_svm_genetic or class_svm_chemical) : see the reference paper for details.

```
% --------------------------------------------------------------------

function AnalyzeFileMenuItem_Callback(hObject, eventdata, handles)

prompt = {'Enter energy threshold:','Enter mininum seizure length:'};
dlg_title = 'Input';
num_lines = 1;
def = {'20','60','0.5'};
answer = inputdlg(prompt,dlg_title,num_lines,def);
if isequal(answer,{})
    return
end

nleo_thr=str2double(answer{1});
len_thr=str2double(answer{2});
%prior=str2double(answer{3}); prior=[prior 1-prior];

classifier = uigetfile('*.mat','Select a classifier
 file','MultiSelect','off');
%classifier='class_svm_chemical.mat';
 if isequal(classifier,0)
     return
 end

[handles.spiketime,handles.duration]=detect_spikes_final(handles.eeg,handles.eegFi
no_spikes=size(handles.spiketime,2);
set(handles.text3, 'String', num2str(no_spikes));
spiketime = sprintf('%.2f|',handles.spiketime);
duration = sprintf('%.2f|',handles.duration);
set(handles.listbox3,'String',num2str(spiketime));
set(handles.listbox4, 'String', num2str(duration));
```

```
guidata(hObject,handles);
```

2. Seizure detection in a batch of files

This option will run the seizure detection on all the LFP files specified in the batch file perviously loaded using the File > Open batch file option. The batch file can be created using File > Create batch file option. The software prompts for 2 values, an energy threshold and a minimum seizure length value. Recommended values are filled in by default, or the user can specify an appropriate value. Energy threshold: the larger it is, the less candidate seizure segments will be selected. Extreme values are between 5-100, a reasonable values are 10-20. Minimum seizure length (in ms): the larger it is, the less candidate seizure segments will be selected. Typical values vary depending on the specific epilepsy model. The software also prompts the user to choose a classifier file (class_svm_genetic or class_svm_chemical) : see the reference paper for details. --

```
function AnalyzeBatchMenuItem_Callback(hObject, eventdata, handles)

prompt = {'Enter energy threshold:','Enter mininum seizure length:'};
dlg_title = 'Input';
num_lines = 1;
def = {'20','60'};
answer = inputdlg(prompt,dlg_title,num_lines,def);
if isequal(answer,{})
    return
end

nleo_thr=str2double(answer{1});
len_thr=str2double(answer{2});
%prior=str2double(answer{3}); prior=[prior 1-prior];


classifier = uigetfile('*.mat','Select a classifier
 file','MultiSelect','off');
%classifier='class_svm_chemical.mat';
 if isequal(classifier,0)
     return
 end
%classifier=load('class_svm.mat');
for i=1:numel(handles.eegFiles)
    msg=msgbox(['Processing file ' num2str(i) 'out of '
 num2str(numel(handles.eegFiles)) '...']);
    eeg=csvread([handles.eegPath,handles.eegFiles{i}]);
    handles.eeg=eeg(:,1);



 [spiketime{i},duration{i}]=detect_spikes_final(handles.eeg,handles.eegFiles{i},nl
    no_spikes(i)=size(spiketime,2);

    if ishandle(msg)
        delete(msg);
    end
end
```

```
waitfor(msgbox('Batch analysis is done. Press OK to save
 results...'));
%ResultsName = uiputfile('.mat','Specify a filename to save analysis
 results...');

headers = {'seizure start time','duration'};
cdir = getappdata(0,'currentdir');
[fn p]=uiputfile([cdir,'*.csv'], 'Specify a filename to save batch
 analysis results');
if isequal(fn,0) | isequal(p,0)
    return
end

data=[];
for i=1:numel(handles.eegFiles)
    data=[data; [{'filename'},handles.eegFiles{i}]];
    data=[data; [{'seizure start time'},{'duration'}]];
    data=[data;  num2cell([spiketime{i}',duration{i}'])];
    data=[data; [{''},{''}]];
end
cd(pwd);
savef = fullfile(p, fn);

fid = fopen(savef, 'w');

cell2csv(savef,data,';')
fclose(fid);
```

3. Time-frequency analysis: visualization

This option visualizes the spectrogram of the currently displayed LFP segment.

```
function tf_anal_Callback(hObject, eventdata, handles)

[S] =
 spectrogram(handles.eegfilt(handles.start:handles.stop),100,99,1:1:400,handles.fs
figure;
imagesc(abs(S));
xlabel('time (ms)');
ylabel('frequency (Hz)');
```

4. Time-frequency quantification

This option computes the power spectral density estaimte, using Welch's method, of the full LFP signal (all LFP files, in case a batch was loaded) in a user-defined frequency band. The software will prompt the user to define the lower and higher limits of the frequency band (in Hz). It is possible to define multiple frequency bands. In this case, the lower limits and the higher limits of the frequency bands have to be specified as Matlab arrays. For example, to define the frequency bands 1-20Hz, 21-40Hz, 41-60Hz, the following parameters should be specified: lower frequency of the band: 1:20:41. higher frequency of the band: 20:20:60. Note that the signals are analysed in short windows, therefore, the estimation will not be precise for low frequencies (below 10Hz). The results will be save in a csv file. The user will be prompted to specify a file name.

```
function tf_quant_Callback(hObject, eventdata, handles)

prompt = {'Enter lower frequency of the band:','Enter higher frequency
 of the band:'};
dlg_title = 'Input';
num_lines = 1;
def = {'20','40'};
answer = inputdlg(prompt,dlg_title,num_lines,def);
if isequal(answer,{})
    return
end


f1=eval(answer{1});
f2=eval(answer{2});

try
    singlemode=0;
    handles.eegFiles
catch
    singlemode=1;
handles.eegFiles{1}=handles.eegFile;
end


for i=1:numel(handles.eegFiles)
    msg=msgbox(['Processing file ' num2str(i) 'out of '
 num2str(numel(handles.eegFiles)) '...']);
    if singlemode
        % do nothing
    else
    eeg=csvread([handles.eegPath,handles.eegFiles{i}]);
    handles.eeg=eeg(:,1);
    end

    % freq. anal

    [pW(i,:)] = pwelch(handles.eeg,100,80,512); %% 100 sample windows
 with 80% overlap

    if ishandle(msg)
        delete(msg);
    end
end
f=0:(handles.fs/2)/(size(pW,2)-1):(handles.fs/2);
for j=1:numel(f1)
f1x(j)=find(f>f1(j),1);
f2x(j)=find(f>f2(j),1);
end
f50=find(f<50,1,'last');

figure
for i=1:numel(handles.eegFiles)
semilogy(pW(i,:));
```

```
hold all;
end
legend(handles.eegFiles)

title('Power Spectral Density')
xlabel('Frequency (Hz)')
ylabel('log(Y(f).^2)')
set(gca,'XTick',f50:f50:max(f))
set(gca,'XTickLabel',50:50:500)
xlim([1 numel(f)-1]);


waitfor(msgbox('Batch analysis is done. Press OK to save
 results...'));

cdir = getappdata(0,'currentdir');
[fn p]=uiputfile([cdir,'*.csv'], 'Specify a filename to save batch
 analysis results');
if isequal(fn,0) | isequal(p,0)
    return
end

data=[];
hdrs=[];
for j=1:numel(f1)
hdrs=[hdrs {[num2str(f1(j)) ' - ' num2str(f2(j)) ' Hz band']}]
end
data=[data; [{'filename'},hdrs]];
for i=1:numel(handles.eegFiles)
    bandpow=[];
    for j=1:numel(f1)
        f1x(j)
        f2x(j)
    bandpow=[bandpow num2cell(sum(pW(i,f1x(j):f2x(j))))];
    end
    data=[data; [handles.eegFiles{i} bandpow]];
end
cd(pwd);
savef = fullfile(p, fn);

fid = fopen(savef, 'w');

cell2csv(savef,data,';')
fclose(fid);
```

III. DISPLAY OPTIONS

1. Fix axes scale for time course

```
function checkbox1_Callback(hObject, eventdata, handles)

handles.fixaxes1 = get(hObject, 'Value');
axes(handles.axes1);
```

```
y=get(gca,'ylim');
handles.ylimits1=y;
guidata(hObject,handles);
```

2. Fix color scale for wavelet transformed signal

```
function checkbox2_Callback(hObject, eventdata, handles)

handles.fixaxes2 = get(hObject, 'Value');
axes(handles.axes2);
y=get(gca,'clim');
handles.climit=y;
guidata(hObject,handles);
```

3. Manual adjust window length

```
function edit2_CreateFcn(hObject, eventdata, handles)  % window length

if ispc && isequal(get(hObject,'BackgroundColor'),
 get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit2_Callback(hObject, eventdata, handles) % window length

handles.window=str2double(get(hObject,'String'))*handles.fs;
handles.stop=handles.start+handles.window;
if handles.stop<numel(handles.eegfilt)

    plotEEG(handles)

end
handles.step=handles.window;
guidata(hObject,handles);
```

4. Semi-auto adjust timecourse scale limits: increase or decrease

```
function pushbutton6_Callback(hObject, eventdata, handles) %% adjust
 scale timecourse

if ~handles.fixaxes1
    axes(handles.axes1);
    y=get(gca,'ylim');
    handles.ylimits1=[-max(abs(y))*2 max(abs(y))*2];
    set(handles.axes1,'ylim',handles.ylimits1);
    set(handles.edit3,'String',['['
 num2str(handles.ylimits1(1),'%.1f') ' '
 num2str(handles.ylimits1(2),'%.1f')  ']' ]);
```

```
    guidata(hObject,handles);
end


function pushbutton7_Callback(hObject, eventdata, handles) %% adjust
 scale timecourse

if ~handles.fixaxes1
    axes(handles.axes1);
    y=get(gca,'ylim');
    handles.ylimits1=[-max(abs(y))/2 max(abs(y))/2];
    set(handles.axes1,'ylim',handles.ylimits1);
    set(handles.edit3,'String',['['
 num2str(handles.ylimits1(1),'%.1f') ' '
 num2str(handles.ylimits1(2),'%.1f')  ']' ]);
    guidata(hObject,handles);
end
```

5. Semi-auto adjust wavelet scale

```
function pushbutton8_Callback(hObject, eventdata, handles)

if ~handles.fixaxes2
    axes(handles.axes2);
    y=get(gca,'clim');

    handles.climit=[y(1)/2 y(2)*2];
    set(handles.axes2,'clim',handles.climit);
    set(handles.edit4,'String',['['
 num2str(handles.climit(1),'%.1f') ' '
 num2str(handles.climit(2),'%.1f')  ']' ]);
    guidata(hObject,handles);

end


function pushbutton9_Callback(hObject, eventdata, handles)

if ~handles.fixaxes2
    axes(handles.axes2);
    y=get(gca,'clim');
    try
        handles.climit=[y(1)*2 y(2)/2];
        set(handles.axes2,'clim', handles.climit);
    catch
        handles.climit=[y(1)*2 max(y(2)/2, y(1)*2+1)];
        set(handles.axes2,'clim', handles.climit);
    end
    set(handles.edit4,'String',['['
 num2str(handles.climit(1),'%.1f') ' '
 num2str(handles.climit(2),'%.1f')  ']' ]);
    guidata(hObject,handles);

end
```

6. Manual adjust timecourse scale

```
function edit4_Callback(hObject, eventdata, handles) % manual adjust
 timecourse scale


if ~handles.fixaxes2
    handles.climit = eval(get(hObject,'String'));
    set(handles.axes2,'clim',handles.climit);
    guidata(hObject,handles);
end



function edit4_CreateFcn(hObject, eventdata, handles) % manual adjust
 timecourse scale

% hObject    handle to edit4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
 called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
 get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
if exist('handles.axes1')
    axes(handles.axes1);
    y=get(gca,'ylim');
    set(hObject,'String',['[' num2str(y) ']' ]);
end
```

7. Manual adjust wavelet scale

```
function edit3_Callback(hObject, eventdata, handles) % manual adjust
 wavelet scale

if ~handles.fixaxes1
    handles.ylimit1 = eval(get(hObject,'String'));
    set(handles.axes1,'ylim',handles.ylimit1);
    guidata(hObject,handles);
end

% --- Executes during object creation, after setting all properties.

function edit3_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
 get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

```
if exist('handles.axes2')
    axes(handles.axes2);
    y=get(gca,'clim');
    set(hObject,'String',['[' num2str(y) ']' ]);
end
```

8. Adjust window start time

```
function edit5_Callback(hObject, eventdata, handles)

handles.start = str2double(get(hObject,'String'))*handles.fs;
handles.stop = handles.start+handles.window;
guidata(hObject,handles);
plotEEG(handles);

function edit5_CreateFcn(hObject, eventdata, handles)  % window start
 time

if ispc && isequal(get(hObject,'BackgroundColor'),
 get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

9. Scroll forward and backward

```
function pushbutton4_Callback(hObject, eventdata, handles)

handles.start=handles.start-handles.step;
handles.stop=handles.start+handles.window;
if handles.stop>=handles.window

    plotEEG( handles)
end
guidata(hObject,handles);


function pushbutton5_Callback(hObject, eventdata, handles) %%

handles.start=(handles.start+handles.step);
handles.stop=handles.start+handles.window;
if handles.stop<numel(handles.eegfilt)

    plotEEG( handles)
end

guidata(hObject,handles);
```

10. Data Visualization

Visualizes the signle channel LFP signal (top) and its wavelet transform (bottom)

```
function plotEEG( handles)
```

```
axes(handles.axes1);
plot([handles.start:handles.stop]./
handles.fs,handles.eegplot(handles.start:handles.stop));
if handles.fixaxes1
    set(handles.axes1,'ylim',handles.ylimits1,'xlim',[[handles.start
 handles.stop]./handles.fs])
else
    axis('tight')
end
axes(handles.axes2);
if handles.fixaxes2
    imagesc([handles.start:handles.stop]./handles.fs,
 [1:size(handles.wc,1)], handles.wc(:,[handles.start:handles.stop]),
[ handles.climit]);
else
    maxval=max(max(handles.wc(:,[handles.start:handles.stop])));
    minval=min(min(handles.wc(:,[handles.start:handles.stop])));
    imagesc([handles.start:handles.stop]./handles.fs,
 [1:size(handles.wc,1)], handles.wc(:,[handles.start:handles.stop]),
[minval maxval;]);
end
colorbar('location','EastOutside')
axes(handles.axes1);
y=get(gca,'ylim');
set(handles.edit3,'String',['[' num2str(y,'%.1f') ']' ]);
axes(handles.axes2);
y=get(gca,'clim');
set(handles.edit4,'String',['[' num2str(y,'%.1f') ']' ]);


% Preprocess EEG prior to visualization

function [eeg,wc,eegplot]=processEEG(eeg,cutoff)

fs=1000;


% settings

res=10; % 10 windows per second




% 'From which second is the EEG stable? (detection starts from here):
 ';
tmp=zeros(1,100);
for i=1:100
tmp(i)=max(abs(eeg((i-1)*fs+1:i*fs)));
if tmp(i)<1
    break;
end
end
```

```
eeg(1:(i-1)*fs)=eeg((i-1)*fs+1);

% interpolate large artifacts

%eeg(1)=0;eeg
x=1:numel(eeg);
xi=x(find(abs(eeg)<1));
yi=eeg(find(abs(eeg)<1));
eegi=interp1(xi,yi,x,'linear');
clear xi yi;

% high pass filter again baseline fluctuation
[b,a] = butter(6,cutoff/(fs/2),'high'); %%%select appropriate filter
 curoff
eegfilt=filtfilt(b,a,eegi);
eegfilt=zscore(eegfilt);

[b,a] = butter(6,1/(fs/2),'high'); %%%select appropriate filter curoff
eegplot=filtfilt(b,a,eegi);
eegplot=zscore(eegplot);


eeg=eegfilt;
clear eegi eegfilt;

disp('preprocessing...')


% compute wavelet coefficients

scales = make_scales(2,200,1,'bior1.5',fs);
wc=zeros(length(scales),length(eeg));


wc=cwt(eeg,scales,'bior1.5').^2;
for i=1:length(scales)
    wc(i,:)=zscore(wc(i,:));
end
```

IV. CORE FUNCTIONS

```
% These functions are called by the buttons of the GUI, the user does
 not
% need to interact with them directly.
```

1. Seizure detection

```
function [spike_timing,
 duration]=detect_spikes_final(eeg,fname,nleo_thr,len_thr,classifier)

load(classifier);
if size(SVMModel.X,2)==112
    flag='genetic'
```

```
elseif size(SVMModel.X,2)==101
    flag='chemical';
end


fs=1000;
res=10;
%nleo_thr=10;
%len_thr=50;

% preprocess before feature extraction
         disp('preprocessing, please wait...')
        % 'From which second is the EEG stable? (detection starts from
 here): ';
        tmp=zeros(1,100);
        for j=1:100
            tmp(j)=max(abs(eeg((j-1)*fs+1:j*fs)));
            if tmp(j)<1
                break;
            end
        end
        eeg(1:(j-1)*fs)=eeg((j-1)*fs+1);

        % interpolate large artifacts

        %eeg(1)=0;
        x=1:numel(eeg);
        xi=x(find(abs(eeg)<1));
        yi=eeg(find(abs(eeg)<1));
        eegi=interp1(xi,yi,x,'linear');
        clear xi yi;

        % high pass filter again baseline fluctuation

        if strcmp(flag,'chemical')
        [b,a] = butter(6,10/(fs/2),'high');
        elseif strcmp(flag,'genetic')
        [b,a] = butter(6,1/(fs/2),'high');
        end
        eegfilt=filtfilt(b,a,eegi);
        eegfilt=zscore(eegfilt);

        eeg=eegfilt;
        clear eegi eegfilt;




disp('extracting features, please wait...')

% segmentation: select segment which an NLEO larger than threshold


eegs=zscore(eeg);
```

```
if size(eegs,1)==1; eegs=eegs'; end
sig_kaiser = kaiserVarRT(eegs);
cand=find(sig_kaiser>=nleo_thr);
len=zeros(size(cand));
fin=cand;



% cluster the close high energy samples together
for j=length(cand):-1:2
    if cand(j)-cand(j-1)<200  %% if they are not 200ms apart, they
 belong to the same event
        fin(j-1)=fin(j);
        len(j-1)=fin(j-1)-cand(j-1);
        len(j)=0;
    end
end

% feature extraction
clear selector segment featvec;
c=find(len>len_thr);
for j=1:numel(c)
    candfin=zeros(size(eegs));
    candfin(cand(c(j)):cand(c(j))+len(c(j)))=1;
    selector{j}=unique(ceil(find(candfin)/(fs/res)));
    segment{j}=cand(c(j)):cand(c(j))+len(c(j));



    featvec(j,:)=extract_features_segment(eeg(segment{j}),flag);

end

if(isempty(c))
    featvec=[];
    spike_timing=[];
    duration=[];
else

    disp('classification...')




    detections = find(predict(SVMModel,featvec));

    if(isempty(detections))
        spike_timing=[];
        duration=[];
    else
        for i=1:numel(detections)
            spike_timing(i)=segment{detections(i)}(1)/1000;
            duration(i)=numel(segment{detections(i)});
        end
```

```
    end
end


    disp('done...')
```

2. Feature extaction

```
function featvec=extract_features_segment(eeg,flag)
% flag: specify if feature extraction according to the genetic or
 chemical model should be run (see reference paper for details!)


 fs=1000;
%
%
if strcmp(flag,'chemical')

% compute wavelet coefficients
%
 scales = make_scales(10,100,1,'morl',fs);
 freqs=10:1:100;
 f10=find(freqs==10);
 f20=find(freqs==20);
 f50=find(freqs==50);

%
    wc=cwt(eeg,scales,'morl').^2;

for i=1:length(scales)
    wc(i,:)=wc(i,:)./std(wc(i,:));
end

% mean wavelet coef. in 1-50Hz band
wc20=mean(wc(f10+1:f20,:),1);
wc50=mean(wc(f20+1:f50,:),1);
wc100=mean(wc(f50+1:end,:),1);


    i=1;
% feature 1: time domain
     zamp(i)=max(abs(eeg));
     rms(i)=sqrt(mean(eeg.^2));

% feature 2: the aveage wavelet coef. in 1-50Hz band exceeds the 20x
 standard deviation
   %  mwc10(i)=mean(wc10);
     mwc20(i)=mean(wc20);
     mwc50(i)=mean(wc50);
     mwc100(i)=mean(wc100);
     mwc(:,i)=mean(wc,2);
```

```
% criterion 3: zero crossings, mins, maxes

    [zeronum(i),
 maxnum(i),minnum(i),m_zero_intvl(i),std_zero_intvl(i)]=countzerominmax(zscore(eeg
    zeronum=zeronum/length(eeg);
    maxnum=maxnum/length(eeg);
    minnum=minnum/length(eeg);

%GEN
%featvec=[zamp; rms; mwc10; mwc20; mwc50; mwc100; mwc; wcr10; wcr20;
 zeronum; maxnum; minnum; m_zero_intvl; std_zero_intvl]';
%CHEM
featvec=[zamp; rms; mwc; mwc20; mwc50; mwc100; zeronum; maxnum;
 minnum; m_zero_intvl; std_zero_intvl]';

else

    % compute wavelet coefficients
%
 scales = make_scales(2,100,1,'bior1.5',fs);

    wc=cwt(eeg,scales,'bior1.5').^2;

for i=1:length(scales)
    wc(i,:)=zscore(wc(i,:));
end

% mean wavelet coef. in 1-50Hz band
wc10=mean(wc(1:10,:),1);
wc20=mean(wc(1:20,:),1);
wc50=mean(wc(1:50,:),1);
wc100=mean(wc(1:end,:),1);

    i=1;
% feature 1: time domain
     zamp(i)=max(abs(eeg));
     rms(i)=sqrt(mean(eeg.^2));

% feature 2: the aveage wavelet coef. in 1-50Hz band exceeds the 20x
 standard deviation
     mwc10(i)=mean(wc10);
     mwc20(i)=mean(wc20);
     mwc50(i)=mean(wc50);
     mwc100(i)=mean(wc100);
     mwc(:,i)=mean(wc,2);

% criterion 3: high and low frequency ratio
     wcr10(i)=mean(abs(wc100))/mean(abs(wc10));
     wcr20(i)=mean(abs(wc100))/mean(abs(wc20));

% criterion 4: zero crossings, mins, maxes

    [zeronum(i),
 maxnum(i),minnum(i),m_zero_intvl(i),std_zero_intvl(i)]=countzerominmax(zscore(eeg
```

```
    zeronum=zeronum/length(eeg);
    maxnum=maxnum/length(eeg);
    minnum=minnum/length(eeg);



featvec=[zamp; rms; mwc10; mwc20; mwc50; mwc100; mwc; wcr10; wcr20;
 zeronum; maxnum; minnum; m_zero_intvl; std_zero_intvl]';

end


function [zeronum,
 maxnum,minnum,m_zero_intvl,std_zero_intvl]=countzerominmax(x)
% This function calualates the number zero crossing, minima, maxima,
 and
% the mean and standard deviation of zero-to-zero intervals
zeronum=0;
maxnum=0;
minnum=0;
zero_idx=[];
for i=2:length(x)-1
    if sign(x(i-1))~=sign(x(i))
        zeronum=zeronum+1;
        zero_idx=[zero_idx i];
    end

    zero_intvl=zero_idx(2:end)-zero_idx(1:end-1);

    m_zero_intvl=mean(zero_intvl);
    std_zero_intvl=std(zero_intvl);



    if x(i-1)<x(i) && x(i)>x(i+1)
        maxnum=maxnum+1;
    end

    if x(i-1)>x(i) && x(i)<x(i+1)
        minnum=minnum+1;
    end

end

function sig_kaiser = kaiserVarRT(sig)
%This function calculates the energy of the signal with the teager-
kaiser
%energy operator. This energy is related to the square of the
 immediate
%amplitude and frequency. The operator enhances spike activity
 facilitating
%the segmentation.

[ySize, xSize] = size(sig);
```

```
sig_kaiser = zeros(ySize, xSize);

for c=1:xSize
    sig_kaiser(4:end,c) = (sig(3:end-1,c).*sig(2:end-2,c))-
(sig(4:end,c).*sig(1:end-3,c));
end
sig_kaiser=abs(sig_kaiser);

return;

function scales = make_scales(startFreq,endFreq,freqRes,wl,fs)

% scales = make_scales(startFreq,endFreq,freqRes,wl)
% Function that determines the scales needed as input for a CWT with a
% certain wavelet. The start frequency (in Hz) is the first input
 argument,
% the end frequency (in Hz) the second, the frequency resolution at
 which
% the transformation should be calculated, the third. The final
 argument is
% the name of the wavelet used for the CWT. A sampling of 250Hz of the
 data
% is assumed.
%

centerFreq = centfrq(wl);
counter=1;
for i=startFreq:freqRes:endFreq
    scales(counter) = ((centerFreq*fs)/(i));
    counter=counter+1;
end
```

x. Others

```
function cell2csv(filename,cellArray,delimiter)
% Writes cell array content into a *.csv file.
%
% CELL2CSV(filename,cellArray,delimiter)
%
% filename      = Name of the file to save. [ i.e. 'text.csv' ]
% cellarray    = Name of the Cell Array where the data is in
% delimiter = seperating sign, normally:',' (default)
%
% by Sylvain Fiedler, KA, 2004
% modified by Rob Kohr, Rutgers, 2005 - changed to english and fixed
 delimiter
if nargin<3
    delimiter = ',';
end

datei = fopen(filename,'w');
for z=1:size(cellArray,1)
    for s=1:size(cellArray,2)
```

```
        var = eval(['cellArray{z,s}']);

        if size(var,1) == 0
            var = '';
        end

        if isnumeric(var) == 1
            var = num2str(var);
        end

        fprintf(datei,var);

        if s ~= size(cellArray,2)
            fprintf(datei,[delimiter]);
        end
    end
    fprintf(datei,'\n');
end
fclose(datei);
```

Published with MATLAB® R2016a