classdef Software01 < matlab.apps.AppBase

% Properties that correspond to app components

properties (Access = public)

UIFigure matlab.ui.Figure

TabGroup matlab.ui.container.TabGroup

Specifics matlab.ui.container.Tab

Stimul matlab.ui.control.EditField

OverLap matlab.ui.control.NumericEditField

OverLapsEditFieldLabel matlab.ui.control.Label

WindowLength matlab.ui.control.NumericEditField

WindowLengthsEditFieldLabel matlab.ui.control.Label

Estimulo matlab.ui.control.EditField

StimuliLabel matlab.ui.control.Label

ChooseObservation matlab.ui.control.DropDown

ChooseObservationLabel matlab.ui.control.Label

Start matlab.ui.control.Button

SamplingFrequency matlab.ui.control.EditField

SamplingFrequencyEditFieldLabel matlab.ui.control.Label

Channels matlab.ui.control.EditField

ChannelsEditFieldLabel matlab.ui.control.Label

Results matlab.ui.container.Tab

UITable matlab.ui.control.Table

end

properties (Access = private)

VALOR;

TABLE;

end

%

methods (Access = private)

function PM=Sinal(app,C,E,eeg,F,Fs)

K=1;

[canal,sinal,estadio]=size(eeg);

for c=1:canal

figure('Name',['Canal ',num2str(C{c})])

g=1;

for e=1:estadio

S\_C\_E{c,e}=eeg(c,:,e); % Sinal no canal (c) e estadio (e)

subplot(estadio,1,g)

for f=1:length(F)

B\_F=F{1,f}; % Filtro dessa Matriz

[num\_B\_F{1,f},den\_B\_F{1,f}]=sos2tf(B\_F.sosMatrix,B\_F.ScaleValues); % coeficientes dos filtros

S\_C\_E\_B\_F{e,f}=filtfilt(num\_B\_F{1,f},den\_B\_F{1,f},S\_C\_E{c,e}); % sina filtrado

FFT\_S\_C\_E\_B\_F{e,f}=fft(S\_C\_E\_B\_F{e,f})/length(S\_C\_E\_B\_F{e,f}); % Fourrier do Sinal

Fc=ceil(length(S\_C\_E\_B\_F{e,f})/2); % frequencia de corte

F\_S=FFT\_S\_C\_E\_B\_F{e,f}; % Ajuda na simplificaçao

F\_S=F\_S(1:Fc); % Vai ate á frequencia de corte

frequency=(0:length(S\_C\_E\_B\_F{e,f})-1)\*Fs/length(S\_C\_E\_B\_F{e,f}); % Vetor frequencia

Ef{e,f}=Energia(app,S\_C\_E\_B\_F{e,f},Fs); % Energia

%%% Pm{e,f}=Potencia(app,Ef{e,f},(0:length(Ef{e,f})-1)/Fs);% Potencia media

Pm{e,f}=Potencia(app,Ef{e,f},app.WindowLength.Value);% Potencia media

%%% subplot(length(E),length(F),K),stem(Pm{e,f}),title(['Fase ',num2str(E(e)),' ',F{2,f}]),xlabel('Amostras'),ylabel('Potência/(W)')%, ylim([0 1])% Apresenta a potencia do sinal filtrado para cada canal estadio e banda de frequencia

plot(frequency(1:Fc),abs(F\_S)),hold on,title(['Fase ',num2str(E(e))]),xlabel('Frequency/(Hz)'),ylabel('Power') % Apresenta a potencia do sinal filtrado para cada canal estadio e banda de frequencia

K=K+1;

end

g=g+1;

hold off

legend('Filtro Alfa', 'Filtro Teta', 'Filtro Delta', 'Filtro Beta', 'Filtro Gama')

end

% atribui os valores calculados para cada canal de celulas

PM{c}=Pm;

K=1;

end

end

function Eff=Energia(app,X,Fs)

Seg=app.WindowLength.Value;

OLap=app.OverLap.Value;

Segment=Seg\*Fs;

OverLap=OLap\*Fs;

Pos=1;

for k=Segment:OverLap:length(X)

Eff(Pos)=sum(abs(X(k+1-Segment:k)).^2);

Pos=Pos+1;

end

end

function Pm=Potencia(app,E,t)

P=E./t;

P=P(1,2:end);

Pm=mean(P(:,2:end));

end

function Signal=Sin(app,C,E,eeg,F,Fs)

[canal,sinal,estadio]=size(eeg);

for e=1:estadio

for c=1:canal

S\_C\_E{c,e}=eeg(c,:,e); % Sinal no canal (c) e estadio (e)

for f=1:length(F)

B\_F=F{1,f}; % Filtro dessa Matriz

[num\_B\_F{1,f},den\_B\_F{1,f}]=sos2tf(B\_F.sosMatrix,B\_F.ScaleValues); % coeficientes dos filtros

S\_C\_E\_B\_F{c,f}=filtfilt(num\_B\_F{1,f},den\_B\_F{1,f},S\_C\_E{c,e}); % sina filtrado

% FFT\_S\_C\_E\_B\_F{c,f}=fft(S\_C\_E\_B\_F{c,f})/length(S\_C\_E\_B\_F{c,f}); % Fourrier do Sinal

% Fc{c,f}=ceil(length(S\_C\_E\_B\_F{c,f})/2); % frequencia de corte

end

end

Signal{e}=S\_C\_E\_B\_F;

end

end

end

% Callbacks that handle component events

methods (Access = private)

% Button pushed function: Start

function StartPushed(app, event)

% Load data

File=struct2cell(load(uigetfile('Test/Data/\*.mat'))); % que se usa

%para se ir buscar um ficheiro

% File=load(string(app.FileName.Value)); % este apenas é

% preciso colocar o nome do ficheiro

% Caracterisitcas do ficheiro

a=[];

K=1;

ChanLabel=File{1,1}.ChanLabel;

EEGdata=File{1,1}.EEGdata;

Fs=File{1,1}.Fs;

Stage=File{1,1}.Stage;

[canal,sinal,estadio]=size(EEGdata);

Canal=ChanLabel(1:canal);

Estadio=Stage(1:estadio);

Fs=Fs;

EEG=EEGdata;

H=Canal.';

app.VALOR=[];

for p=1:length(H)

app.VALOR=[app.VALOR string(H(p))];

end

for p=1:length(Estadio)

app.VALOR=[app.VALOR num2str(Estadio(p))];

end

app.ChooseObservation.Items=app.VALOR;

for l=1:canal

a=[a string(Canal{K,1})];

K=K+1;

end

% Matriz com os Filtros

F={Filtro\_alfa(Fs) Filtro\_teta(Fs) Filtro\_delta(Fs) Filtro\_beta(Fs) Filtro\_gama(Fs);

'Alfa' 'Teta' 'Delta' 'Beta' 'Gama'}; % acrescentar o filtro notch

% Caracteristicas visuais

app.Channels.Value=cell2mat(strcat(a,','));

app.Stimul.Value=num2str(Estadio);

app.SamplingFrequency.Value=strcat(num2str(Fs),' Hz');

% Potencia media do sinal para cada canal, estado e banda de

% frequencia

PM=Sinal(app,Canal,Estadio,EEG,F,Fs);% Sinal Filtrado para cada banda de frequencia, estadio e canal

Signal=Sin(app,Canal,Estadio,EEG,F,Fs); % Sinal

% permite calculcar a percentagem de potencia que tem cada banda

% para o canal

p=1;

for i=1:canal

pm=cell2mat(PM{1,i});

soma=[];

for u=1:length(F)

soma=[soma sum(pm(:,u))];

end

a=[pm;soma];

b=[Estadio.'; "TOTAL"];

M=[b a];

pert=[];

for h=1:length(F)

pert=[pert (str2double(M(end,h+1))/sum(str2double(M(end,2:end))))\*100];

end

a=[pm;soma;round(pert,2)+"%"];

ESTADIO=[Estadio.';"TOTAL";"PERCENTAGEM"];

ALFA=a(:,1);

TETA=a(:,2);

DELTA=a(:,3);

BETA=a(:,4);

GAMA=a(:,5);

% app.TABLE{k}=table(legenda.',h.',pValue.',banda.');

TABLE1{p}=table(ESTADIO,ALFA,TETA,DELTA,BETA,GAMA);

p=p+1;

end

k=1;

P=1;

for SE=1:estadio

for SC=1:canal-1

for SF=1:length(F)

[h(P),pValue(P)]=gctest(Signal{1,SE}{SC,SF}.',Signal{1,SE}{SC+1,SF}.');

legenda(P)=[string(Canal{SC,1}) + ' & ' + string(Canal{SC+1,1})];

banda(P)=string(F{2,SF});

P=P+1;

end

end

LEGENDA=legenda.';

H=h.';

PVALUE=pValue.';

BANDA=banda.';

% m=[LEGENDA,H,PVALUE,BANDA];

TABLE2{k}=table(LEGENDA,H,PVALUE,BANDA);

% m=[legenda.',h.',pValue.',banda.'];

% % matriz{k}=m;

% TABLE2{k}=table(legenda.',h.',pValue.',banda.');

% writematrix(matriz{k},'dados7@temp.xlsx','Sheet',SE,'WriteMode','replacefile'); % verificar uma forma mais geral de gerar a tabela

% app.TABLE{k}=readtable("dados7@temp.xlsx",'Sheet',SE,'ReadVariableNames',false);

k=k+1;

P=1;

end

app.TABLE=[TABLE1 TABLE2];

% app.TABLE{p}=[table(ESTADIO,ALFA,TETA,DELTA,BETA,GAMA);

% MATRIX{p}=[b a];

% writematrix(MATRIX{p},'dados7@temp.xlsx','Sheet',i,'WriteMode','replacefile');

% app.TABLE{p}=readtable("dados7@temp.xlsx",'Sheet',i,'ReadVariableNames',false);

% p=p+1;

% end

%

% P=1;

% k=1;

% for SE=1:estadio

%

% for SC=1:canal-1

%

% for SF=1:length(F)

%

% [h(P),pValue(P)]=gctest(Signal{1,SE}{SC,SF}.',Signal{1,SE}{SC+1,SF}.');

% legenda(P)=[string(Canal{SC,1}) + ' & ' + string(Canal{SC+1,1})];

% banda(P)=string(F{2,SF});

% P=P+1;

% end

%

% end

% m=[legenda.',h.',pValue.',banda.'];

% matriz{k}=m;

% app.TABLE{k}=table(legenda.',h.',pValue.',banda.');

% % writematrix(matriz{k},'dados7@temp.xlsx','Sheet',SE,'WriteMode','replacefile'); % verificar uma forma mais geral de gerar a tabela

% % app.TABLE{k}=readtable("dados7@temp.xlsx",'Sheet',SE,'ReadVariableNames',false);

% k=k+1;

% P=1;

% end

%

%

% app.VALOR=[];

% for p=1:length(Estadio)

% app.VALOR=[app.VALOR num2str(Estadio(p))];

% end

% app.ChooseObservation.Items=app.VALOR;

end

% Callback function: ChooseObservation, ChooseObservation,

% ...and 1 other component

function ChooseObservationValueChanged(app, event)

value = app.ChooseObservation.Value;

app.Results.Title=value;

position=find(value==app.VALOR);

app.UITable.Data=app.TABLE{1,position};

app.UITable.ColumnName=app.TABLE{1,position}.Properties.VariableNames;

end

end

% Component initialization

methods (Access = private)

% Create UIFigure and components

function createComponents(app)

% Create UIFigure and hide until all components are created

app.UIFigure = uifigure('Visible', 'off');

app.UIFigure.Position = [100 100 640 480];

app.UIFigure.Name = 'MATLAB App';

% Create TabGroup

app.TabGroup = uitabgroup(app.UIFigure);

app.TabGroup.Position = [1 1 640 480];

% Create Specifics

app.Specifics = uitab(app.TabGroup);

app.Specifics.Title = 'Specifics';

% Create ChannelsEditFieldLabel

app.ChannelsEditFieldLabel = uilabel(app.Specifics);

app.ChannelsEditFieldLabel.HorizontalAlignment = 'right';

app.ChannelsEditFieldLabel.Position = [71 366 55 22];

app.ChannelsEditFieldLabel.Text = 'Channels';

% Create Channels

app.Channels = uieditfield(app.Specifics, 'text');

app.Channels.Editable = 'off';

app.Channels.Position = [141 366 150 22];

% Create SamplingFrequencyEditFieldLabel

app.SamplingFrequencyEditFieldLabel = uilabel(app.Specifics);

app.SamplingFrequencyEditFieldLabel.HorizontalAlignment = 'right';

app.SamplingFrequencyEditFieldLabel.Position = [43 272 115 22];

app.SamplingFrequencyEditFieldLabel.Text = 'Sampling Frequency';

% Create SamplingFrequency

app.SamplingFrequency = uieditfield(app.Specifics, 'text');

app.SamplingFrequency.Editable = 'off';

app.SamplingFrequency.Position = [173 272 90 22];

% Create Start

app.Start = uibutton(app.Specifics, 'push');

app.Start.ButtonPushedFcn = createCallbackFcn(app, @StartPushed, true);

app.Start.BackgroundColor = [0.8902 0.8902 0.8902];

app.Start.FontWeight = 'bold';

app.Start.Position = [163 137 100 23];

app.Start.Text = 'Start';

% Create ChooseObservationLabel

app.ChooseObservationLabel = uilabel(app.Specifics);

app.ChooseObservationLabel.HorizontalAlignment = 'right';

app.ChooseObservationLabel.Position = [338 250 118 22];

app.ChooseObservationLabel.Text = 'Choose Observation';

% Create ChooseObservation

app.ChooseObservation = uidropdown(app.Specifics);

app.ChooseObservation.DropDownOpeningFcn = createCallbackFcn(app, @ChooseObservationValueChanged, true);

app.ChooseObservation.ValueChangedFcn = createCallbackFcn(app, @ChooseObservationValueChanged, true);

app.ChooseObservation.ClickedFcn = createCallbackFcn(app, @ChooseObservationValueChanged, true);

app.ChooseObservation.Position = [471 250 100 22];

% Create StimuliLabel

app.StimuliLabel = uilabel(app.Specifics);

app.StimuliLabel.HorizontalAlignment = 'right';

app.StimuliLabel.Position = [41 316 41 22];

app.StimuliLabel.Text = 'Stimuli';

% Create Estimulo

app.Estimulo = uieditfield(app.Specifics, 'text');

app.Estimulo.Editable = 'off';

app.Estimulo.Position = [97 316 166 22];

% Create WindowLengthsEditFieldLabel

app.WindowLengthsEditFieldLabel = uilabel(app.Specifics);

app.WindowLengthsEditFieldLabel.HorizontalAlignment = 'right';

app.WindowLengthsEditFieldLabel.Position = [43 229 105 22];

app.WindowLengthsEditFieldLabel.Text = 'Window Length (s)';

% Create WindowLength

app.WindowLength = uieditfield(app.Specifics, 'numeric');

app.WindowLength.Position = [163 229 100 22];

% Create OverLapsEditFieldLabel

app.OverLapsEditFieldLabel = uilabel(app.Specifics);

app.OverLapsEditFieldLabel.HorizontalAlignment = 'right';

app.OverLapsEditFieldLabel.Position = [43 188 68 22];

app.OverLapsEditFieldLabel.Text = 'OverLap (s)';

% Create OverLap

app.OverLap = uieditfield(app.Specifics, 'numeric');

app.OverLap.Position = [126 188 137 22];

% Create Stimul

app.Stimul = uieditfield(app.Specifics, 'text');

app.Stimul.Editable = 'off';

app.Stimul.Position = [270 316 100 22];

% Create Results

app.Results = uitab(app.TabGroup);

app.Results.Title = 'Results';

% Create UITable

app.UITable = uitable(app.Results);

app.UITable.ColumnName = {'Column 1'; 'Column 2'; 'Column 3'; 'Column 4'};

app.UITable.RowName = {};

app.UITable.Position = [1 1 638 455];

% Show the figure after all components are created

app.UIFigure.Visible = 'on';

end

end

% App creation and deletion

methods (Access = public)

% Construct app

function app = Software01

% Create UIFigure and components

createComponents(app)

% Register the app with App Designer

registerApp(app, app.UIFigure)

if nargout == 0

clear app

end

end

% Code that executes before app deletion

function delete(app)

% Delete UIFigure when app is deleted

delete(app.UIFigure)

end

end

end