EPHEC

Louvain-la-neuve

Traitement de signal

Projet - Accordeur de guitare

Rapport de projet

3TL1

Herrier Lucie Juckler Christian Musuvaho Grace Nyssens Sylvain

15 décembre 2014

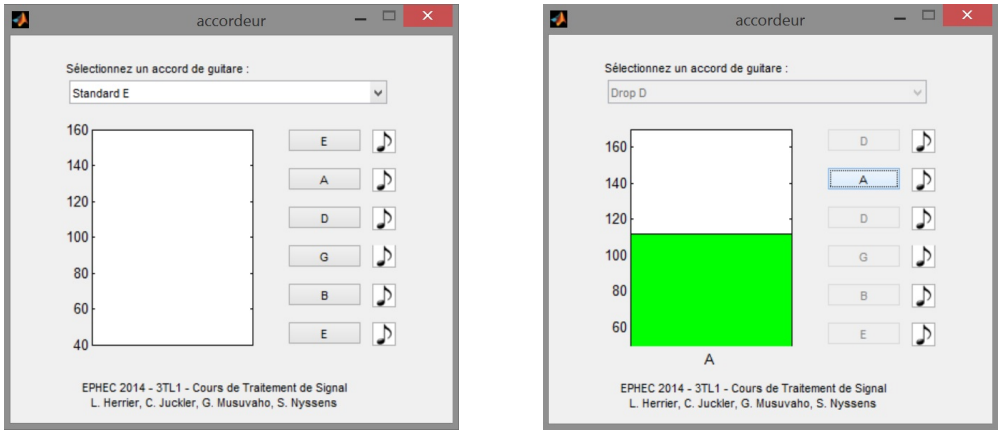
Table des matières

1 Introduction

Dans le cadre du cours de travaux pratiques de traitement de signal, nous avons dû réaliser un projet mettant en application nos acquis. Notre groupe a choisi de créer un accordeur de guitare. Celui-ci a été codé à l'aide de Matlab, l'outil utilisé lors des exercices à réaliser durant le cours. Ce rapport présente tout d'abord en quelques lignes en quoi consistait le projet. Nous expliquons par la suite la méthodologie que nous avons utilisée, et nous apportons des idées d'améliorations possibles pour notre accordeur. Enfin, chaque membre du groupe apporte sa conclusion personnelles concernant ce projet.

2 Présentation du projet

Le projet que nous avons réalisé est un accordeur de guitare. Celui-ci a été codé à l'aide du logiciel utilisé lors des TPs, Matlab. Nous avons choisi d'implémenter une interface graphique, plus intuitive à utiliser. Celle-ci propose d'accorder sa guitare selon plusieurs accords fréquemment utilisés. La figure 1 montre l'interface utilisateur de l'accordeur.



(a) Accordeur au démarrage (b) Accordeur en train d'accorder

FIGURE 1 – Interface utilisateur

On peut voir sur les images ci-dessus la manière dont nous avons composé notre accordeur. Dans le haut de la fenêtre se trouve le menu de sélection des accords. Nous proposons divers accords fréquents pour la guitare :

- L'accordage classique en *E*
- Le *Drop D*
- L'*Open D*
- L'*Open G*
- L'accordage en *quartes*

Les boutons de droite représentent les notes composant l'accord, ainsi que la possibilité d'écouter la note. Celui du dessus correspond à la corde du dessus lorsqu'on tient une guitare, autrement dit la note la plus grave. Enfin, le grand rectangle à gauche sert à visualiser si la corde de la guitare sélectionnée est bien accordée ou non. Le code couleur utilisé est classique : vert si la corde est correctement accordée, à 2Hz près, jaune-orange si la corde est trop haute ou trop basse à 15Hz près, et rouge si elle est hors des limites précédentes.

Lorsqu'une note est sélectionnée pour être accordée, un timer de 5 secondes s'enclenche. Il est possible pendant ce temps de jouer la corde pour l'accorder. De plus, les autres notes sont bloquées, ainsi que la sélection des accords, afin d'éviter les conflits.

3 Méthodologie utilisée

Lors de la réalisation de notre accordeur, nous sommes passés par plusieurs étapes avant d'arriver au produit actuel. Cette partie du rapport explique ces différentes phases de développement, ainsi que les difficultés rencontrées au cours de celles-ci.

En premier lieu, nous avons tout d'abord cherché à enregistrer un son dans Matlab. En effet, nous voulions pouvoir visualiser le spectre fréquentiel de l'enregistrement du son produit par une corde de guitare sur une durée déterminée. Après plusieurs tentatives, nous y sommes arrivé. Nous nous sommes vite rendu compte qu'il y avait un problème dans notre analyse. Nous avions sous-estimé la puissance des harmoniques. Lors de nos test, nous travaillions avec un logiciel capable de simuler un diapason et une guitare. Avec le diapason, nous n'avions aucun doute sur notre approche. En effet, les fréquences émises étant unique, nous avions un spectre fréquentiel net avec la fréquence jouée se démarquant nettement. Toutefois, avec la guitare, nous avons remarqué que les harmoniques étaient plus puissante que la fréquence fondamentale. Nous nous retrouvions donc face à un problème.

Dans un deuxième temps, nous donc avons cherché à éliminer les harmoniques. En effet, les harmoniques sont des fréquences indésirables et parasites empêchant un accordage efficace. Pour ce faire, nous avons analysé les différents accords existants ainsi que la fréquence maximale à détecter pour la corde la plus aiguë. Il en a résulté que nous pouvions nous passer des fréquences au-delà de 500 Hz. Nous avons ainsi cherché à limiter la représentation graphique autour de la fréquence voulue. Chaque note possédant sa propre fréquence, il n'est pas compliqué de borner les fréquences représentées sur le graphique autour de celle recherchée. Nous nous sommes rendus compte que pour éviter toute harmonique indésirable sur la représentation spectrale, il suffisait d'interdire les fréquences supérieures à celle recherchée, plus 50Hz. La fréquence enregistrée et voulue devenait donc la plus puissante, donc plus facile à récupérer. Évidemment, cela a également apporté un problème, de sorte que si l'on joue un La à 220Hz alors qu'on veut accorder un Mi à 83Hz, la fréquence jouée n'est plus détectable. Nous avons alors décidé que ce n'était pas de la plus grande importance, une corde de guitare étant rarement 50Hz plus haut ou plus bas lorsqu'elle doit être accordée.

Par la suite, nous avons réfléchi à une façon de stocker les différents accords existants en associant les notes avec leur fréquence respective. Pour ce faire, nous sommes partis d'un La 440 Hz afin de générer la gamme de la 3ème octave. À partir de cette gamme, nous avons pu calculer les fréquences des notes pour les différents accords. Nous avons tout d'abord généré un accordage classique en E, puis nous sommes partis vers d'autres accord également fréquemment rencontrés, tels que le Drop D ou encore l'Open D.

Parallèlement, nous avons commencé une interface graphique, afin de rendre l'accordeur intuitif pour l'utilisateur. Cette première interface affichait une zone d'accordage rectangulaire avec un graphique s'actualisant à chaque enregistrement, pour une fréquence fixée. Avec cette interface, nous pouvions analyser une note précise durant un laps de temps. À la fin de ce laps de temps, nous affichions la fréquence réelle de la note jouée. Selon l'écart avec la valeur théorique, nous pouvions visualiser si l'accordage était correct, trop grave ou trop aigu. Après avoir limité l'enregistrement à un laps de temps, nous sommes passés au temps réel. Nous lancions l'enregistrement, et à l'aide d'une boucle while, nous analysions continuellement le son enregistré.

Finalement, nous avons crée une interface graphique plus évoluée à l'aide de l'outil GUIDE. Nous avons ensuite assemblé l'interface et la fonction de génération des fréquences des accords et d'enregistrement pour arriver à un programme fonctionnel. Nous avons dû changer les paramètres de la boucle while avec cette nouvelle interface. Au départ, nous pensions lancer l'enregistrement, et la boucle, au clic d'un bouton, et le terminer au clic sur une autre note, sauf que nous n'y

sommes pas arrivés. Nous avons donc décidé de définir un timer de 5 secondes lors d'un clic sur une note, pendant lequel l'accordage serait possible. L'interface proposée aujourd'hui pour notre accordeur permet de changer d'accord, de jouer les notes, et d'accorder, bien évidemment.

4 Améliorations possibles

Notre projet d'accordeur pourrait être amélioré de plusieurs manières. Tout d'abord, nous pourrions rajouter plusieurs autres accords à la liste de ceux déjà proposés. Cela permettrait à l'accordeur d'être encore plus efficace pour les guitaristes aimant jouer plusieurs styles musicaux différents.

Nous avions également pensé, à la base, à laisser l'accordage de la corde actif jusqu'à ce qu'une autre note soit sélectionnée. N'ayant pas réussi à mettre cette technique en place, nous avons opté pour l'implémentation d'un timer. Un amélioration possible serait donc de revenir à notre idée initiale d'accordage en continu pour une corde. Un clic sur la note lancerait le processus d'accordage, et un clic sur une autre note stopperait le premier et lancerait le nouveau.

Enfin, nous aurions pu réaliser une interface permettant, une fois l'accord sélectionné, d'accorder n'importe quelle corde. C'est-à-dire qu'au lieu des notes sur la droite et d'un graphique à gauche, nous aurions eu 6 graphiques (un par note). Nous n'avons pas pu mettre en place un tel système à cause du problème des harmoniques mentionné plus haut. Cependant, une amélioration de l'accordeur pour parvenir à une telle version serait intéressante. En effet, plus besoin de choisir la note. Le guitariste jouerait la corde qu'il veut, et il lui suffirait de regarder le graphique correspondant pour l'accorder.

5 Conclusions personnelles

5.1 Herrier Lucie

A la fin de ce projet, j'ai l'impression de mieux maîtriser l'outil Matlab. J'ai découvert d'autres fonctionnalités que celles vues en classe. C'était un projet intéressant et agréable à réaliser, car la musique fait partie de ma vie depuis que je suis toute petite. J'ai parfois eu un peu du mal avec le code, notamment lors de la réalisation de l'interface graphique. Par ailleurs, je suis contente que notre groupe ait réussi à réaliser un accordeur de guitare suffisamment efficace. Je pense que nous pouvons tous être fier de ce que nous avons produit. Toutefois, une amélioration de l'accordeur le rendrait encore plus efficace.

5.2 Juckler Christian

Ce projet m'a intéressé sur plusieurs aspects. D'abord sur un aspect technique, l'analyse spectrale du son m'a toujours intéressée, mais je n'avais jamais pu y consacrer du temps. C'est chose faite avec ce projet. Ensuite sur un aspect musical, mon groupe m'a appris quelques bases en musique. Ces bases m'ont permis de mieux comprendre le principe de l'accordeur, vu que je n'avais aucune connaissance en musique. Ce projet m'a permis aussi de me rendre compte qu'il n'est pas facile de travailler dans un domaine inconnu. Finalement, je suis satisfait du travail accompli. Notre accordeur fonctionne et peut facilement être amélioré.

5.3 Musuvaho Grace

J'aimais bien le sujet du projet et j'étais super motivée, mais j'ai eu du mal à apporter une réelle contribution au travail. J'avoue que je me suis sentie parfois un peu inutile. J'ai aidé à faire des recherches et essayer de trouver les bugs. Malgré tout, grâce aux recherches, j'ai pu en apprendre d'avantage sur le traitement du son avec Matlab.

5.4 Nyssens Sylvain

Nécessitant de maîtriser différentes notions vue aux cours, ce projet m'a permis de mieux assimiler la théorie vue en cours. Tout cela en illustrant un cas pratique qui m'intéresse particulièrement, étant guitariste. La prise en main de GUIDE, l'outil de création d'interface graphique de MatLab, n'a pas été simple aux premiers abords. Mais celui-ci s'est finalement révélé assez flexible et à permis de créer l'interface désirée sans trop d'encombres. Les interactions au seins du groupe ont été enrichissante et ont, selon moi, donné un beau résultat.

Bibilographie

1. DEWULF A., Syllabus de cours : *Les techniques de traitement de signal*, EPHEC LLN, année 2014-2015.
2. La documentation Matlab, `http://nl.mathworks.com/help/matlab/`, consultée durant tout le projet.
3. How do I display an image on a GUI component (eg. pushbutton) ?, `http://www.mathworks.com/matlabcentral/answers/98593-how-do-i-display-an-image-on-a-gui-component-eg-pushbutton`, consulté le 14 décembre 2014.
4. Fréquences des touches du piano, `http://fr.wikipedia.org/wiki/Fr%C3%A9quences_des_touches_du_piano`, consulté en novembre 2014.
5. Accordage, `http://fr.wikipedia.org/wiki/Accordage`, consulté en novembre 2014.
6. Les open tunings et les accordages particuliers, `http://www.instinctguitare.com/open-tunings-et-accordages-particuliers/`, consulté en novembre 2014.

A Code

```
1  function varargout = accordeur(varargin)
2  % ACCORDEUR MATLAB code for accordeur.fig
3  % ACCORDEUR, by itself, creates a new ACCORDEUR or raises the existing
4  % singleton*.
5  %
6  % H = ACCORDEUR returns the handle to a new ACCORDEUR or the handle to
7  % the existing singleton*.
8  %
9  % ACCORDEUR('CALLBACK',hObject,eventData,handles,...) calls the local
10 % function named CALLBACK in ACCORDEUR.M with the given input arguments.
11 %
12 % ACCORDEUR('Property','Value',...) creates a new ACCORDEUR or raises the
13 % existing singleton*. Starting from the left, property value pairs are
14 % applied to the GUI before accordeur_OpeningFcn gets called. An
15 % unrecognized property name or invalid value makes property application
16 % stop. All inputs are passed to accordeur_OpeningFcn via varargin.
17 %
18 % *See GUI Options on GUIDE's Tools menu. Choose "GUI allows only one
19 % instance to run (singleton)".
20 %
21 % See also: GUIDE, GUIDATA, GUIHANDLES
22
23 % Edit the above text to modify the response to help accordeur
24
25 % Last Modified by GUIDE v2.5 14-Dec-2014 17:00:02
26
27 % Begin initialization code - DO NOT EDIT
28 gui_Singleton = 1;
29 gui_State = struct('gui_Name', mfilename, ...
30                    'gui_Singleton', gui_Singleton, ...
31                    'gui_OpeningFcn', @accordeur_OpeningFcn, ...
32                    'gui_OutputFcn', @accordeur_OutputFcn, ...
33                    'gui_LayoutFcn', [] , ...
34                    'gui_Callback', []);
35 if nargin && ischar(varargin{1})
36     gui_State.gui_Callback = str2func(varargin{1});
37 end
38
39 if nargout
40     [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
41 else
42     gui_mainfcn(gui_State, varargin{:});
43 end
44 % End initialization code - DO NOT EDIT
45
46 % --- Executes just before accordeur is made visible.
47 function accordeur_OpeningFcn(hObject, eventdata, handles, varargin)
48 % This function has no output args, see OutputFcn.
49 % hObject handle to figure
```

```
50  % eventdata reserved - to be defined in a future version of MATLAB
51  % handles structure with handles and user data (see GUIDATA)
52  % varargin command line arguments to accordeur (see VARARGIN)
53
54  % Choose default command line output for accordeur
55  handles.output = hObject;
56
57  % Update handles structure
58  guidata(hObject, handles);
59
60  % This sets up the initial plot - only do when we are invisible
61  % so window can get raised using accordeur.
62  if strcmp(get(hObject,'Visible'),'off')
63      %% Déclaration des variables
64       La = 440;
65       ajoutDemiTon = 2^(1/12);
66       ajoutTon = 2^(1/6);
67       Octave3 = cell(2,12);
68       Octave3(1,:)={'C','C#','D','D#','E','F','F#','G','G#','A','A#','B'};
69
70       %% Génération du tableau de la 3ème octave
71       Octave3{2,10}=La;
72
73       for i = 10:11
74            Octave3{2,i+1}=Octave3{2,i}*ajoutDemiTon;
75       end;
76       for i = 10:(-1):2
77            Octave3{2,i-1}=Octave3{2,i}/ajoutDemiTon;
78       end;
79
80       %% Génération des tableaux des différents accords
81       % Classique E
82       AccordE = cell(2,6);
83       AccordE(1,:)={'E','A','D','G','B','E'};
84       AccordE = initAccordEOpen(AccordE, Octave3);
85
86       % Drop D
87       AccordDropD = AccordE;
88       AccordDropD{1,1}='D';
89       for j= 1:12
90            if (Octave3{1,j}==AccordDropD{1,1})
91                 AccordDropD{2,1} = Octave3{2,j}/4;
92            end;
93       end;
94
95       % Quartes
96       AccordQuartes = AccordE;
97       AccordQuartes{1,5}='C';
98       AccordQuartes{1,6}='F';
99       for i = 5:6
100           for j= 1:12
101                if (Octave3{1,j}==AccordQuartes{1,i})
```

```
102                     AccordQuartes{2,i} = Octave3{2,j};
103                 end;
104             end;
105         end;
106
107         % Open D
108         AccordOpenD = cell(2,6);
109         AccordOpenD(1,:)={'D','A','D','F#','A','D'};
110         AccordOpenD = initAccordEOpen(AccordOpenD, Octave3);
111
112         % Open G
113         AccordOpenG = cell(2,6);
114         AccordOpenG(1,:)={'D','G','D','G','B','D'};
115         AccordOpenG = initAccordEOpen(AccordOpenG, Octave3);
116
117         %% Memoriser les variables pour l'interface graphique
118         handles.AccordE = AccordE;
119         handles.AccordDropD = AccordDropD;
120         handles.AccordQuartes = AccordQuartes;
121         handles.AccordOpenD = AccordOpenD;
122         handles.AccordOpenG = AccordOpenG;
123         handles.Accord = AccordE;
124         handles.Fs = 4000;
125         handles.nBits = 8;
126         handles.nChannel = 1;
127         handles.length = 0.5;
128         handles.L=10000;
129         guidata(hObject, handles);
130
131         %% Barre de départ et bouton de départ avec l'accord
132         accordBar(40,100, '');
133         setBoutonsNotes(AccordE, handles);
134         imageNotes(handles);
135 end
136 % Effacer l'écran
137 clc;
138
139 % UIWAIT makes accordeur wait for user response (see UIRESUME)
140 % uiwait(handles.figure1);
141
142
143 % --- Outputs from this function are returned to the command line.
144 function varargout = accordeur_OutputFcn(hObject, eventdata, handles)
145 % varargout cell array for returning output args (see VARARGOUT);
146 % hObject handle to figure
147 % eventdata reserved - to be defined in a future version of MATLAB
148 % handles structure with handles and user data (see GUIDATA)
149
150 % Get default command line output from handles structure
151 varargout{1} = handles.output;
152
153
```

```
154
155 % --------------------------------------------------------------------
156 function FileMenu_Callback(hObject, eventdata, handles)
157 % hObject handle to FileMenu (see GCBO)
158 % eventdata reserved - to be defined in a future version of MATLAB
159 % handles structure with handles and user data (see GUIDATA)
160
161
162 % --------------------------------------------------------------------
163 function OpenMenuItem_Callback(hObject, eventdata, handles)
164 % hObject handle to OpenMenuItem (see GCBO)
165 % eventdata reserved - to be defined in a future version of MATLAB
166 % handles structure with handles and user data (see GUIDATA)
167 file = uigetfile('*.fig');
168 if ~isequal(file, 0)
169     open(file);
170 end
171
172 % --------------------------------------------------------------------
173 function PrintMenuItem_Callback(hObject, eventdata, handles)
174 % hObject handle to PrintMenuItem (see GCBO)
175 % eventdata reserved - to be defined in a future version of MATLAB
176 % handles structure with handles and user data (see GUIDATA)
177 printdlg(handles.figure1)
178
179 % --------------------------------------------------------------------
180 function CloseMenuItem_Callback(hObject, eventdata, handles)
181 % hObject handle to CloseMenuItem (see GCBO)
182 % eventdata reserved - to be defined in a future version of MATLAB
183 % handles structure with handles and user data (see GUIDATA)
184 selection = questdlg(['Close ' get(handles.figure1,'Name') '?'],...
185                      ['Close ' get(handles.figure1,'Name') '...'],...
186                      'Yes','No','Yes');
187 if strcmp(selection,'No')
188     return;
189 end
190
191 delete(handles.figure1)
192
193
194 % --- Executes on selection change in popupmenu1.
195 function popupmenu1_Callback(hObject, eventdata, handles)
196 % hObject handle to popupmenu1 (see GCBO)
197 % eventdata reserved - to be defined in a future version of MATLAB
198 % handles structure with handles and user data (see GUIDATA)
199
200 % Hints: contents = get(hObject,'String') returns popupmenu1 contents as cell array
201 % contents{get(hObject,'Value')} returns selected item from popupmenu1
202 cla;
203
204 popup_sel_index = get(handles.popupmenu1, 'Value');
205 % Sélection de l'accordage défini par l'utilisateur
```

```
206 switch popup_sel_index
207     case 1
208         Accord = handles.AccordE;
209     case 2
210         Accord = handles.AccordDropD;
211     case 3
212         Accord = handles.AccordOpenD;
213     case 4
214         Accord = handles.AccordOpenG;
215     case 5
216         Accord = handles.AccordQuartes;
217 end
218 jouerAccord(Accord);
219 %Modification des boutons pour les notes
220 setBoutonsNotes(Accord, handles);
221 handles.Accord = Accord;
222 guidata(hObject, handles);
223
224
225
226 % --- Executes during object creation, after setting all properties.
227 function popupmenu1_CreateFcn(hObject, eventdata, handles)
228 % hObject handle to popupmenu1 (see GCBO)
229 % eventdata reserved - to be defined in a future version of MATLAB
230 % handles empty - handles not created until after all CreateFcns called
231
232 % Hint: popupmenu controls usually have a white background on Windows.
233 % See ISPC and COMPUTER.
234 if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
235     set(hObject,'BackgroundColor','white');
236 end
237
238 set(hObject, 'String', {'Standard E', 'Drop D', 'Open D', 'Open G', 'Quartes'});
239
240 % --- Executes on button press in note1.
241 function note1_Callback(hObject, eventdata, handles)
242 activerNote(1, handles);
243
244 % --- Executes on button press in note2.
245 function note2_Callback(hObject, eventdata, handles)
246 activerNote(2, handles);
247
248 % --- Executes on button press in note3.
249 function note3_Callback(hObject, eventdata, handles)
250 activerNote(3, handles);
251
252 % --- Executes on button press in note4.
253 function note4_Callback(hObject, eventdata, handles)
254 activerNote(4, handles);
255
256 % --- Executes on button press in note5.
257 function note5_Callback(hObject, eventdata, handles)
```

```
258 activerNote(5, handles);
259
260 % --- Executes on button press in note6.
261 function note6_Callback(hObject, eventdata, handles)
262 activerNote(6, handles);
263
264
265 % --- Executes on button press in play_note1.
266 function play_note1_Callback(hObject, eventdata, handles)
267 jouerNote(handles.Accord,1);
268
269
270 % --- Executes on button press in play_note2.
271 function play_note2_Callback(hObject, eventdata, handles)
272 jouerNote(handles.Accord,2);
273
274
275 % --- Executes on button press in play_note3.
276 function play_note3_Callback(hObject, eventdata, handles)
277 jouerNote(handles.Accord,3);
278
279 % --- Executes on button press in play_note4.
280 function play_note4_Callback(hObject, eventdata, handles)
281 jouerNote(handles.Accord,4);
282
283 % --- Executes on button press in play_note5.
284 function play_note5_Callback(hObject, eventdata, handles)
285 jouerNote(handles.Accord,5);
286
287 % --- Executes on button press in play_note6.
288 function play_note6_Callback(hObject, eventdata, handles)
289 jouerNote(handles.Accord,6);
290
291
292
293 %% Fonctions personnelles
294 % Fonction d'initialisation des accords de type E ou Open sur base de
295 % l'octave 3.
296 function [Accord] = initAccordEOpen(Accord, Octave3)
297 for i=1:6
298     for j=1:12
299         x=0;
300         if (Octave3{1,j}==Accord{1,i})
301             if (i==6)
302                 x=1;
303             elseif (i<=2)
304                 x=4;
305             else
306                 x=2;
307             end;
308             Accord{2,i} = Octave3{2,j}/x;
309         end;
```

```
310
311         end;
312 end;
313
314 % Fonction permettant de jouer la note en paramètre
315 function jouerNote(Accord,note)
316         tsec=0.5;
317         a=2;
318         F=44100;
319         t = linspace(0,tsec,tsec*F);
320         y=a*sin(2*pi*Accord{2,note}*t);
321         sound(y,F);
322
323 % Fonction permettant de jouer l'accord en paramètre
324 function jouerAccord(Accord)
325         tsec=0.5;
326         a=2;
327         F=44100;
328         t = linspace(0,tsec,tsec*F);
329         y=a*sin(2*pi*Accord{2,1}*t);
330         for i=2 :6
331             y = [y (a*sin(2*pi*Accord{2,i}*t))];
332         end;
333         sound(y,F);
334
335 % Calcul de la couleur de la barre d'accordage qui affiche où on en est par
336 % rapport à l'accord
337 function accordBar(frequence,x, note)
338 if(frequence>=(x+30) || frequence<=(x-30))
339         color = [1 0 0]; % Rouge
340 elseif(frequence>=(x+15) || frequence<=(x-15))
341         color = [1 0.5 0]'; % Orange
342 elseif(frequence>(x+2) || frequence<(x-2))
343         color = [1 1 0]'; % Jaune
344 else
345         color=[0 1 0]; % Vert
346 end
347 bar(frequence,'facecolor', color);
348 xlim([0.6,1.4]);
349 set(gca,'xtick',[]);
350 xlabel(note);
351 ylim([(x-60),(x+60)]);
352
353 % Définition des noms des boutons de notes
354 function setBoutonsNotes(Accord, handles)
355 set(handles.note1,'string',Accord{1,1});
356 set(handles.note2,'string',Accord{1,2});
357 set(handles.note3,'string',Accord{1,3});
358 set(handles.note4,'string',Accord{1,4});
359 set(handles.note5,'string',Accord{1,5});
360 set(handles.note6,'string',Accord{1,6});
361
```

```
362 % Accordage d'une note en particulier pendant 5 secondes
363 function accorderNote(position,handles)
364 clc;
365 Accord = handles.Accord;
366 x = Accord{2,position};
367 % Récupération de variables
368 Fs = handles.Fs;
369 nBits = handles.nBits;
370 nChannel= handles.nChannel;
371 % Enregistrer le son
372 soundRecord = audiorecorder(Fs, nBits, nChannel);
373 record(soundRecord);
374 pause(0.3);
375 compte = 1;
376 % Pendant 5 sec, traiter le son enregistré pour accorder la guitare
377 while (compte < 50)
378     myRecording = getaudiodata(soundRecord);
379     myRecording = xcorr(myRecording);
380     NFFT = Fs;
381     Y = fft(myRecording,NFFT);
382     f = (Fs/2*linspace(0,1,NFFT/2+1));
383     [valeur, frequence]=max(Y(f>0 & f<(x+50)));
384     accordBar(frequence,x, Accord{1,position});
385     pause(0.1);
386     compte = compte+1;
387 end
388
389 % Fonction pour enable et disable les boutons pendant l'accordage d'une
390 % note
391 function boutonOffOn(note, etat, handles)
392 if (note ~= 1) set(handles.note1,'Enable',etat); end
393 if (note ~= 2) set(handles.note2,'Enable',etat); end
394 if (note ~= 3) set(handles.note3,'Enable',etat); end
395 if (note ~= 4) set(handles.note4,'Enable',etat); end
396 if (note ~= 5) set(handles.note5,'Enable',etat); end
397 if (note ~= 6) set(handles.note6,'Enable',etat); end
398 set(handles.popupmenu1,'Enable',etat);
399
400 % Fonctions à exécuter lors de la sélection d'une note
401 function activerNote(pos, handles)
402 boutonOffOn(pos, 'off', handles);
403 accorderNote(pos,handles);
404 boutonOffOn(pos, 'on', handles);
405 xlabel('');
406
407 % Fonction pour l'affichage des boutons style "note"
408 function imageNotes(handles)
409 [x,map]=imread('noteM.jpg');
410 Img=imresize(x, [20 20]);
411 set(handles.play_note1, 'cdata', Img);
412 set(handles.play_note2, 'cdata', Img);
413 set(handles.play_note3, 'cdata', Img);
```

```
414 set(handles.play_note4, 'cdata', Img);
415 set(handles.play_note5, 'cdata', Img);
416 set(handles.play_note6, 'cdata', Img);
```