实验九 电话拨号音的合成与识别

1 实验目的

本实验基于对电话通信系统中拨号音合成与识别的仿真实现，主要涉及到电话拨号音合成的基本原理及识别的主要方法，利用 MATLAB 软件以及 FFT 算法实现对电话通信系统中拨号音的合成与识别。并进一步利用 MATLAB 中的图形用户界面 GUI 制作简单直观的模拟界面。使其对电话通信系统拨号音的合成与识别有个基本的了解。

能够利用矩阵不同的基频合成 0 — 9 不同按键的拨号音，并能够对不同的拨号音加以正确的识别，实现由拨号音解析出电话号码的过程。进一步利用 GUI 做出简单的图形操作界面。要求界面清楚，画面简洁，易于理解，操作简单。从而实现对电话拨号音系统的简单的实验仿真。

2 实验原理

双音多频 DTMF（Dual Tone Multi-Frequency）信号，是用两个特定的单音频率信号的组合来代表数字或功能。在 DTMF 电话机中有 16 个按键，其中 10 个数字键 0 — 9，6 个功能键 * 、 # 、 A 、 B 、 C 、 D。其中 12 个按键是我们比较熟悉的按键，另外由第 4 列确定的按键作为保留，作为功能键留为今后他用。根据 CCITT 建议，国际上采用 697Hz、770Hz、852Hz、941Hz 低频群及 1209Hz、1336Hz、1477Hz、1633Hz 高频群。从低频群和高频群任意各抽出一种频率进行组合，共有 16 种组合，代表 16 种不同的数字键或功能，每个按键唯一地由一组行频和列频组成，如表 1 所示。

表 1 ： DTMF 的组合功能

fH(Hz) / fL(Hz)	1209	1336	1477	1633
697	1	2	3	A
770	4	5	6	B
852	7	8	9	C
941	*	0	#	D

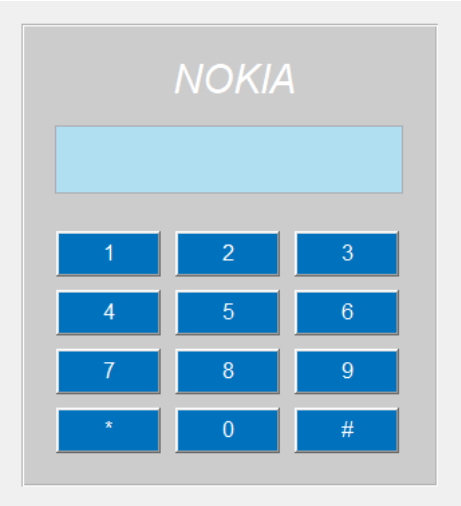
3 实验过程与实验结果

3.1 图形电话拨号面板的制作

利用 GUI 图形用户界面设计工具制作电话拨号面板，把 DTMF 信号和电话机的键盘矩阵对应起来。其中选用我们熟悉的 10 个数字键 0 — 9， 2 个功能键“ * ”、“＃”，另四个键省略。按照表 1 电话机键盘矩阵的排列方式制作四行三列的按键控件。每个按键可用 Push Button 添加。

然后，为了更直观的反映对应的按键号码，可以设置一个编辑框，用于动态的显示拨号号码，模拟实际电话的拨号显示窗口。编辑框可用 Edit Text 添加。另外，为了图形电话拨号面板的简洁美观，可以添加空白区域作为背景，并用静态文本框制作文字信息。背景可用 Frame 添加，静态文本框可用 Static Text 添加。

最终利用 GUI 图形用户界面设计工具生成的图形电话拨号面板用于拨号音的合成产生部分，如下图所示。这里将其保存为 tu1.fig 文件。



3.2 DTMF 信号的产生合成

对上节制作的图形电话拨号面板上的各控件单位的动作和变化进行设置，即对 tu1.m 文件进行编辑。其主要的功能是使对应的按键，按照表 1 的对应关系产生相应的拨号音，完成对应行频及列频的叠加输出。此外，对于图形界面的需要，还要使按键的号码数字显示在拨号显示窗口中。

鉴于 CCITT 对 DTMF 信号规定的指标，这里每个数字信号取 1000 个采样点模拟按键信号，并且每两个数字之间用 100 个 0 来表示间隔来模拟静音，以便区别连续的两个按键信号。间隔的静音信号也是在按键时产生的。

以按键 1 为例，简单介绍拨号音产生的过程：

```
76      % --- Executes on button press in pushbutton1.
77      function varargout = pushbutton1_Callback(h, eventdata, handles, varargin)
78 —      n=[1:1000]; % 每个数字 1000 个采样点表示
79 —      d0=sin(0.5346*n)+sin(0.9273*n); % 对应行频列频叠加
80 —      n0=strcat(get(handles.edit1,'string'),'1'); % 获取数字号码
81 —      set(handles.edit1,'string',n0); % 显示号码
82 —      space=zeros(1,100); %100 个 0 模拟静音信号
83 —      global NUM
84 —      phone=[NUM,d0];
85 —      NUM=[phone,space]; % 存储连续的拨号音信号
86 —      sound(d0,8192); % 产生拨号音
```

程序解释：

NUM 为定义的全局变量，用于存储连续的拨号音（DTMF）信号，包括数字信号音以及静音信号；

d0=sin(0.5346*n)+sin(0.9273*n) 中的行频与列频是由表 1 中 1 键对应的$f_L = 697Hz$，$f_H = 1209Hz$计算得出，已知声音取样频率$f_s = 8192Hz$，则取样后$\omega_L = 2\pi f_L/f_s = 0.5346$，$\omega_H = 2\pi f_H/f_s = 0.9273$。

每个按键计算出的频率如下：

button	FL	FH	ωL	ωH
1	697	1209	0.5346	0.9273
2	697	1336	0.5346	1.0247
3	697	1477	0.5346	1.1328
4	770	1209	0.5906	0.9273
5	770	1336	0.5906	1.0247
6	770	1477	0.5906	1.1328
7	852	1209	0.6535	0.9273
8	852	1336	0.6535	1.0247
9	852	1477	0.6535	1.1328
0	941	1336	0.7217	1.0247
*	941	1209	0.7217	0.9273
#	941	1477	0.7217	1.1328

对于其他的数字按键，按照按键 1 的代码进行参数的修改即可。

对于保留的两个功能键“ * ”“ # ”，按照现行键盘式拨号电话的习惯，将“ * ”作为删除键，“ # ”作为确认键。“ * ”删除键的作用是将前面拨错的号码删除退回，表现为将显示窗口已经显示的错误号码退回一位数字，并且将连续拨号音信号的存储单元 NUM 中退回一位拨号音信号和静音信号。删除可

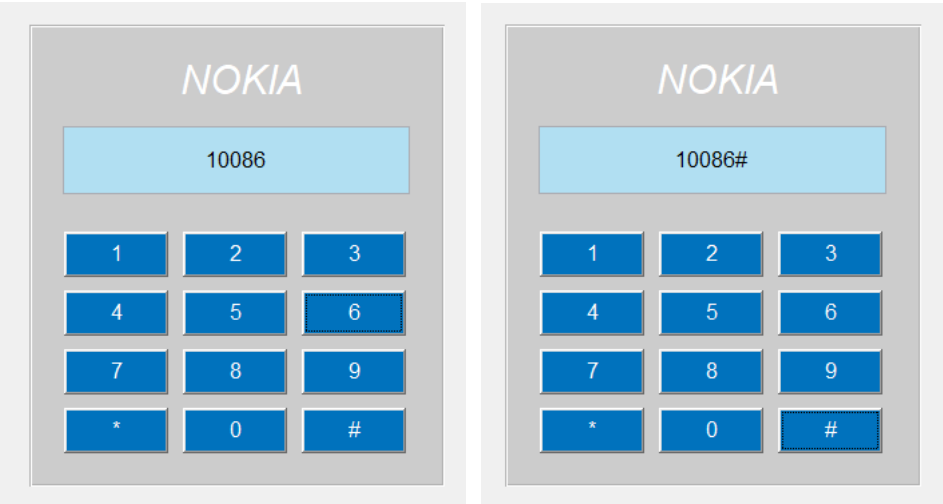
以进行连续的操作。“＃”确认键的作用是将前面拨过的号码进行确认保留，意味着此时连续拨号音信号的存储单元 NUM 中的信号即为最后用于识别的连续拨号音 DTMF 信号，并在显示窗口中显示“＃”号作为标记。

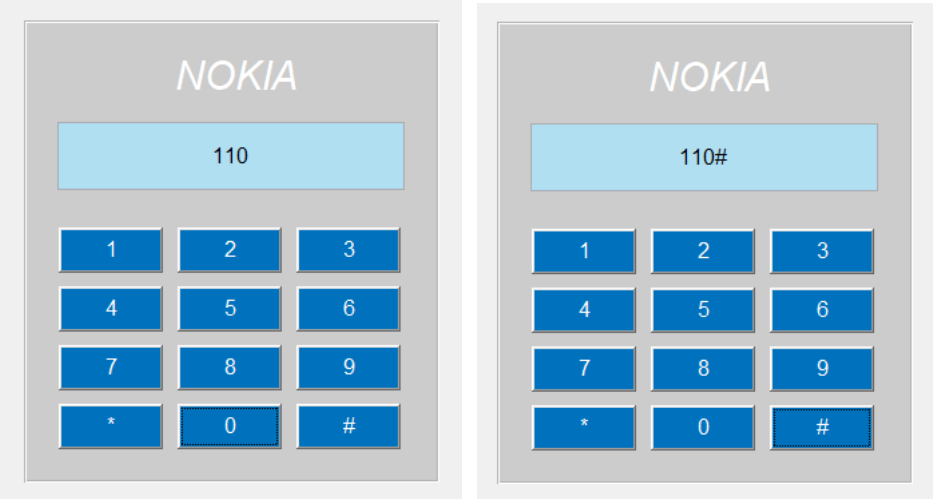
```
192     % 删除键的响应函数
193     function varargout = pushbuttonback_Callback(h, eventdata, handles, varargin)
194 —   n=[1:1000];
195 —   num=get(handles.edit1,'string');
196 —   l=length(num);
197 —   n11=strrep(num,num,num(1:l-1)); %去掉末尾号码在面板上的显示
198 —   d11=sin(0.7217*n)+sin(0.9273*n);
199 —   set(handles.edit1,'string',n11);
200 —   global NUM
201 —   L=length(NUM);
202 —   NUM=NUM(1:L-1100); %删除末尾号码在拨号音信号中的存储
203 —   sound(d11,8192);
```

```
220     % 按键 # 的响应函数
221     function pushbuttonNUM_Callback(h, eventdata, handles, varargin)
222 —   n=[1:1000]; % 每个数字 1000 个采样点表示
223 —   d0=sin(0.7217*n)+sin(1.1328*n); % 对应行频列频叠加
224 —   n0=strcat(get(handles.edit1,'string'),'#'); % 获取数字号码
225 —   set(handles.edit1,'string',n0); % 显示号码
226 —   space=zeros(1,100); %100 个 0 模拟静音信号
227 —   sound(d0,8192); % 产生拨号音
```

为简单起见，先使“#”键的功能仅为在显示窗口中显示“#”号标记号码输入结束，之后再完善相关功能。

运行结果如下：





在按下每一个按键时，都会发出相应的拨号音，按下数字按键后会在 NUM 中存入相应的数字信号，而按下“*”则会删除在最尾端的数字信号，按下“#”仅标记输入结束，不会对 NUM 进行修改。

3.3 DTMF 信号的检测识别

DTMF 信号的检测识别原理如下：

要实现电话拨号音（DTMF）信号的检测识别，可以通过直接计算傅里叶变换得到输入信号的组成频率。这里采用 FFT 算法对信号进行解码分析。首先对接收到的数字信号作 FFT 分析，计算出其幅度谱，进而得到功率谱，组成输入信号的频率必定对应功率谱的峰值。对于连续的双音多频（DTMF）信号，需要把有效的数字拨号信号从静音间隔信号中分割提取出来，然后再用 FFT 算法对信号进行解码分析。

现将检测识别功能的集成在“#”键上，即按下该键，除了标记号码输入结束之外，还会识别输入的信号，并画出频谱图。

代码如下：

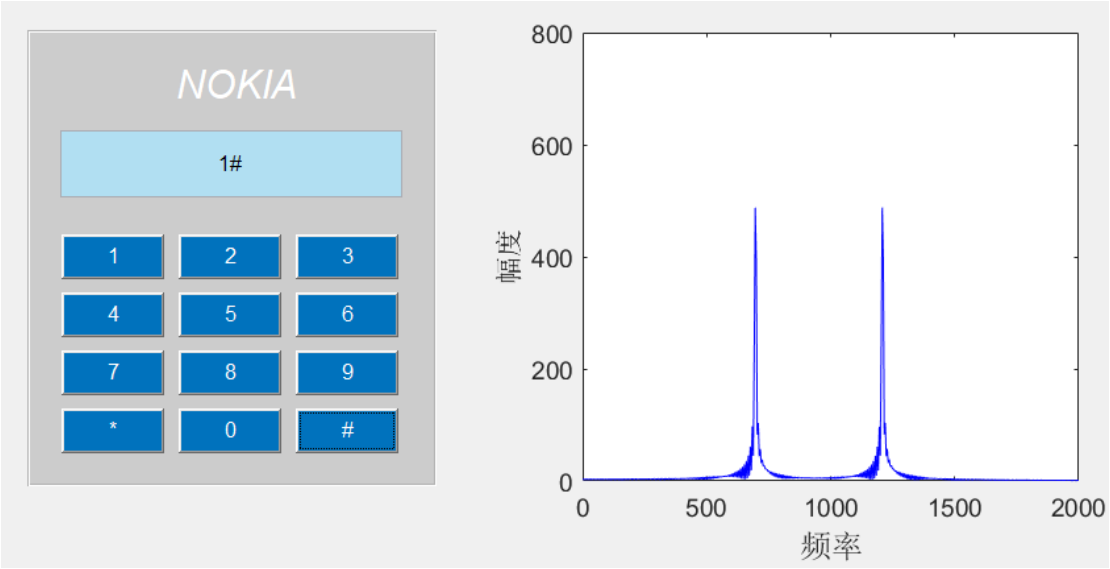
```
220     % 按键 # 的响应函数
221     function pushbuttonNUM_Callback(h, eventdata, handles, varargin)
222 —   n=[1:1000]; % 每个数字 1000 个采样点表示
223 —   d0=sin(0.7217*n)+sin(1.1328*n); % 对应行频列频叠加
224 —   n0=strcat(get(handles.edit1,'string'),'#'); % 获取数字号码
225 —   set(handles.edit1,'string',n0); % 显示号码
226 —   space=zeros(1,100); %100 个 0 模拟静音信号
227 —   sound(d0,8192); % 产生拨号音
228 —   global NUM
229 —   sound(NUM,8192);
230 —   L=length(NUM);
231 —   n=L/1100;
232 —   number='';
```

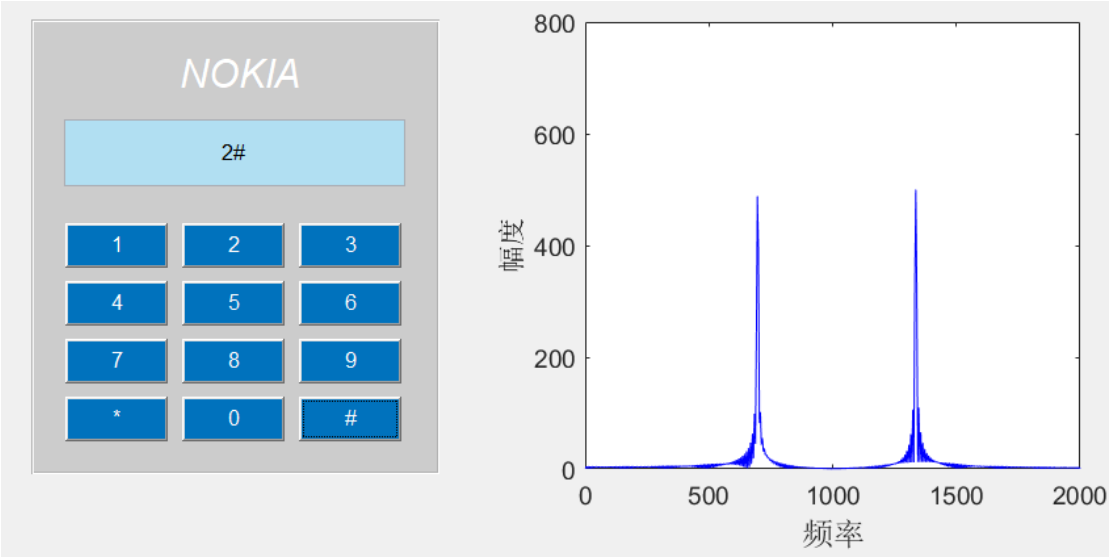
```
for i=1:n
    j=(i-1)*1100+1;
    d=NUM(j:j+999); % 截取出每个数字
    f=fft(d,2048); % 以 N=2048 作 FFT 变换
    a=abs(f);
    p=a.*a/10000; % 计算功率谱
    num(1)=find(p(1:250)==max(p(1:250))); % 找行频
    num(2)=300+find(p(300:380)==max(p(300:380))); % 找列频
    q=(0:2047)*4;
    axes(handles.axes1);
    plot(q,a,'b');
    axis([0,2000,0,800]);
    xlabel('频率','fontsize',12);
    ylabel('幅度');
    if (num(1) < 180)
        row=1; % 确定行数
    elseif (num(1) < 200)
        row=2;
    elseif (num(1) < 220)
        row=3;
    else
        row=4;
    end
    if (num(2) < 320)
        column=1; % 确定列数
    elseif (num(2) < 340)
        column=2;
    else
        column=3;
    end
    z=[row,column]; % 确定数字
    if z==[4,2]
        tel=0;
    elseif z==[1,1]
        tel=1;
    elseif z==[1,2]
        tel=2;
    elseif z==[1,3]
        tel=3;
    elseif z==[2,1]
        tel=4;
    elseif z==[2,2]
        tel=5;
    elseif z==[2,3]
        tel=6;
    elseif z==[3,1]
        tel=7;
    elseif z==[3,2]
        tel=8;
    elseif z==[3,3]
        tel=9;
    end
    t(i)=tel;
    c=strcat(number,int2str(tel));
    number=c;
    i=i+1;
end
set(handles.edit2,'string',number);
```

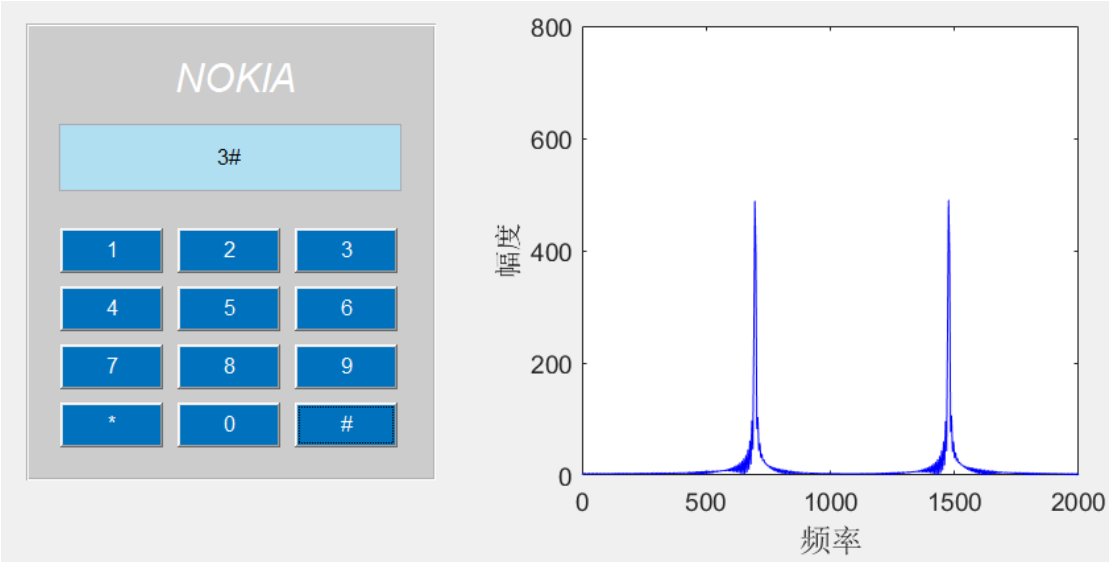
程序解释：

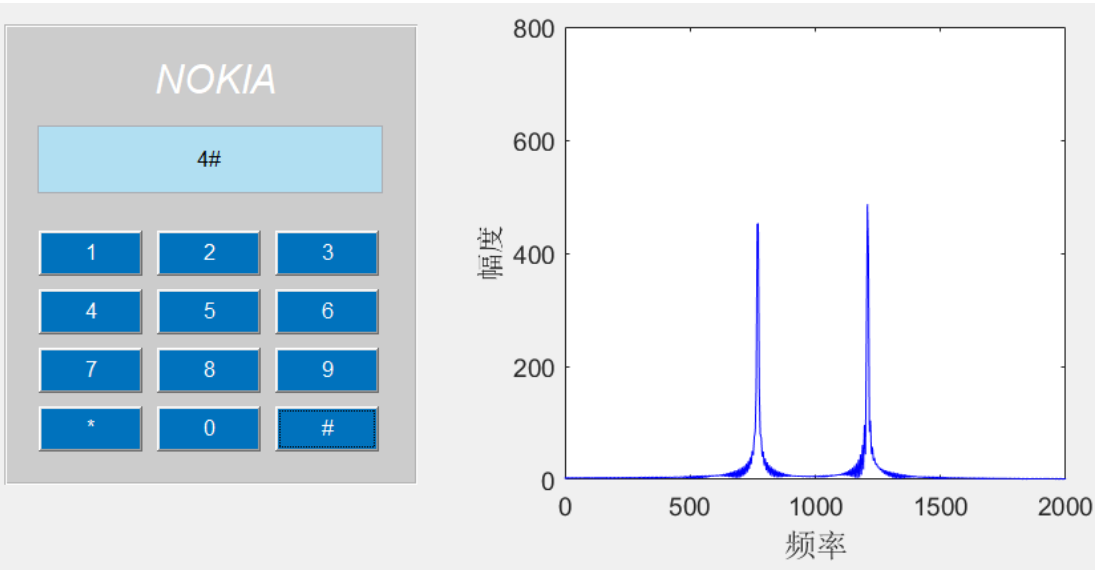
确定行频和列频的数值范围是通过计算得出的：已知输入信号的取样频率$f_s = 8192Hz$，而做 FFT 的 N=2048，则频谱分辨率$F = f_s/N = 4Hz$，由此可算出频谱图上任意点对应的频率$K = f/F$，为真实频率的 1/4。为了使画出的频谱图能直观反映出真实频率，在用 plot 画图前将横坐标 q 乘以 4，即第 241 行的代码 q = (0 : 2047) * 4，这样处理后画出的频谱图的横坐标就对应真实的频率。

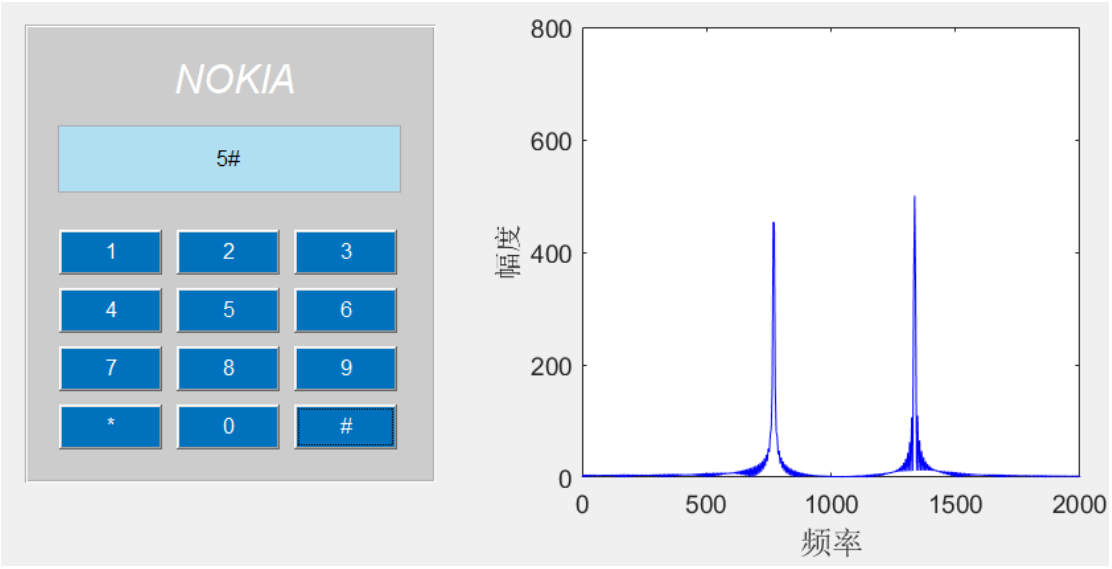
运行结果如下：

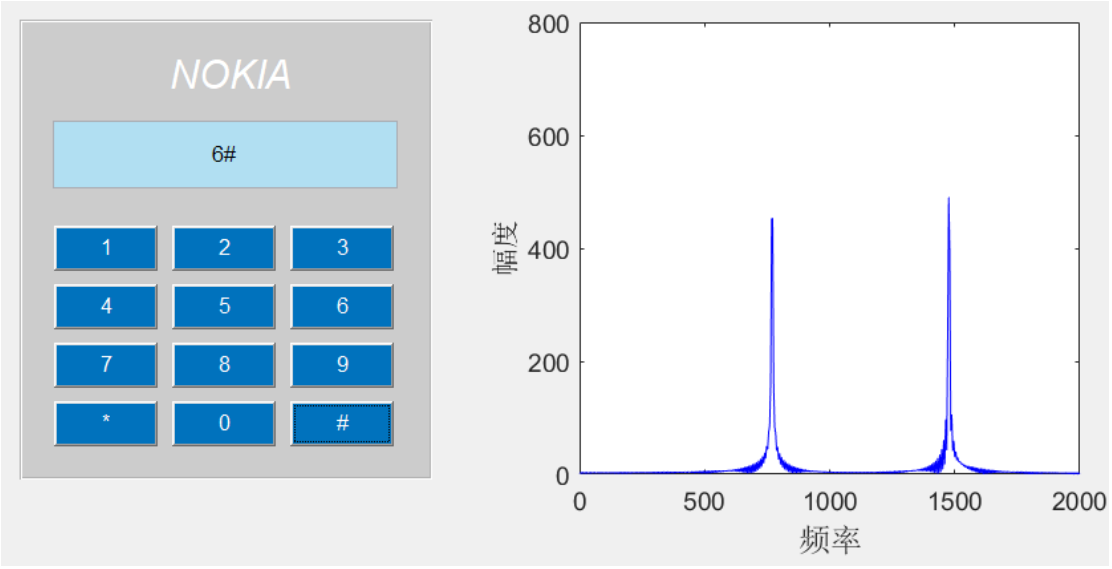

NOKIA
1#
1 2 3
4 5 6
7 8 9
* 0 #
幅度
频率
0 200 400 600 800
0 500 1000 1500 2000

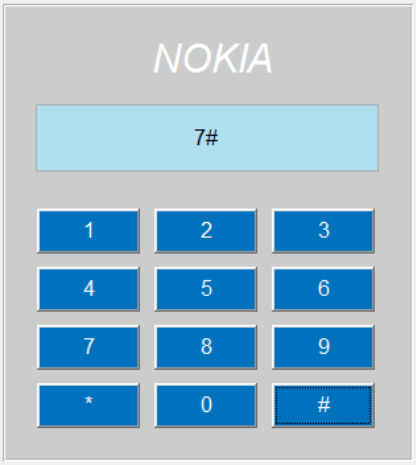

NOKIA
2#
1 2 3
4 5 6
7 8 9
* 0 #
幅度
频率
0 200 400 600 800
0 500 1000 1500 2000

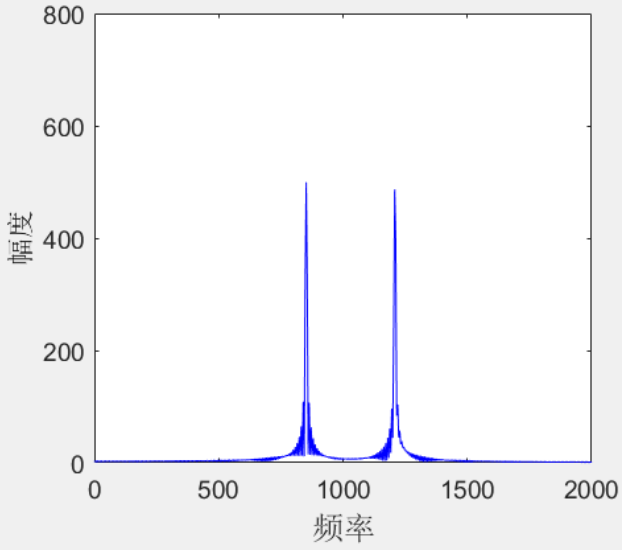

NOKIA
3#
1 2 3
4 5 6
7 8 9
* 0 #
幅度
频率
0 200 400 600 800
0 500 1000 1500 2000

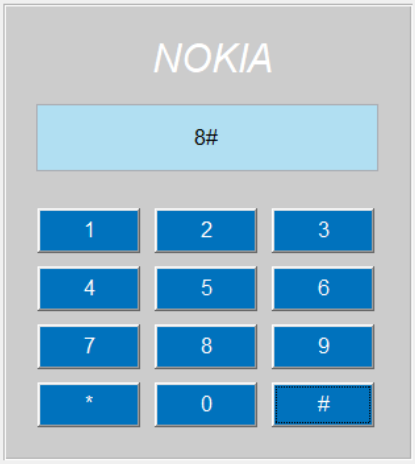

NOKIA
4#
1
2
3
4
5
6
7
8
9
*
0
#
800
600
400
200
0
幅度
0
500
1000
1500
2000
频率

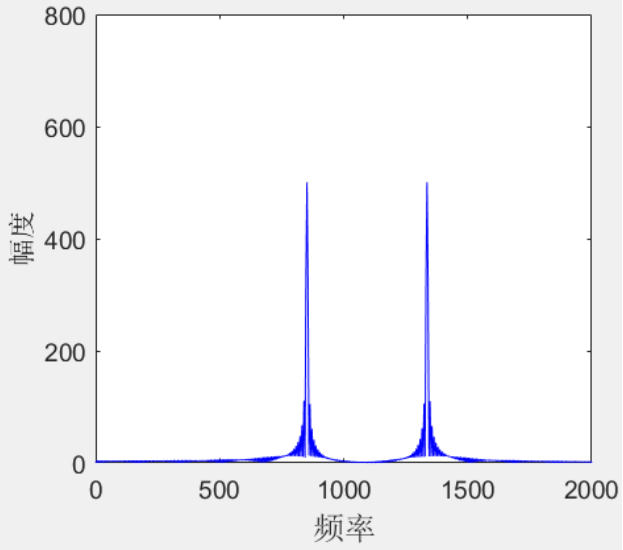

NOKIA
5#
1
2
3
4
5
6
7
8
9
*
0
#
800
600
400
200
0
幅度
0
500
1000
1500
2000
频率

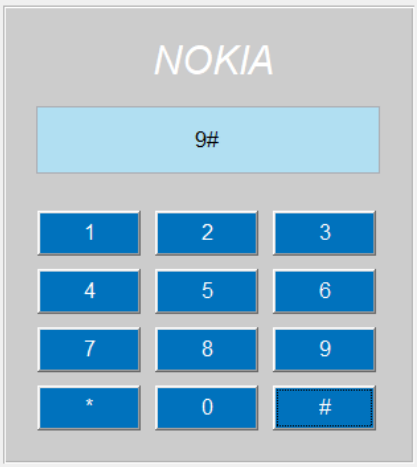

NOKIA
6#
1
2
3
4
5
6
7
8
9
*
0
#
800
600
400
200
0
幅度
0
500
1000
1500
2000
频率

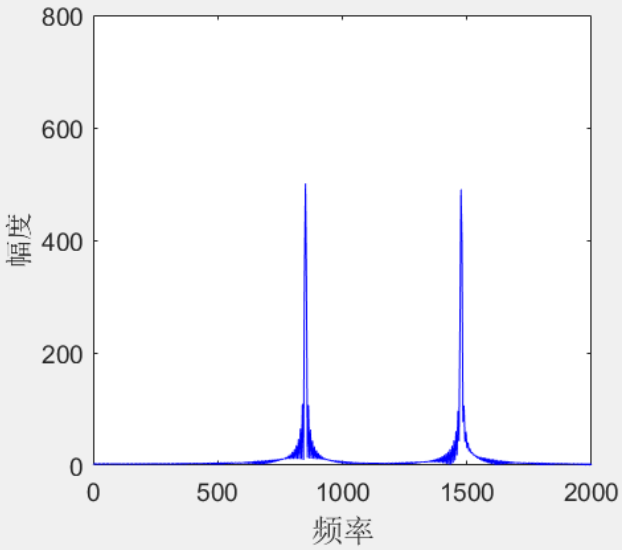

NOKIA
7#
1
2
3
4
5
6
7
8
9
*
0
#

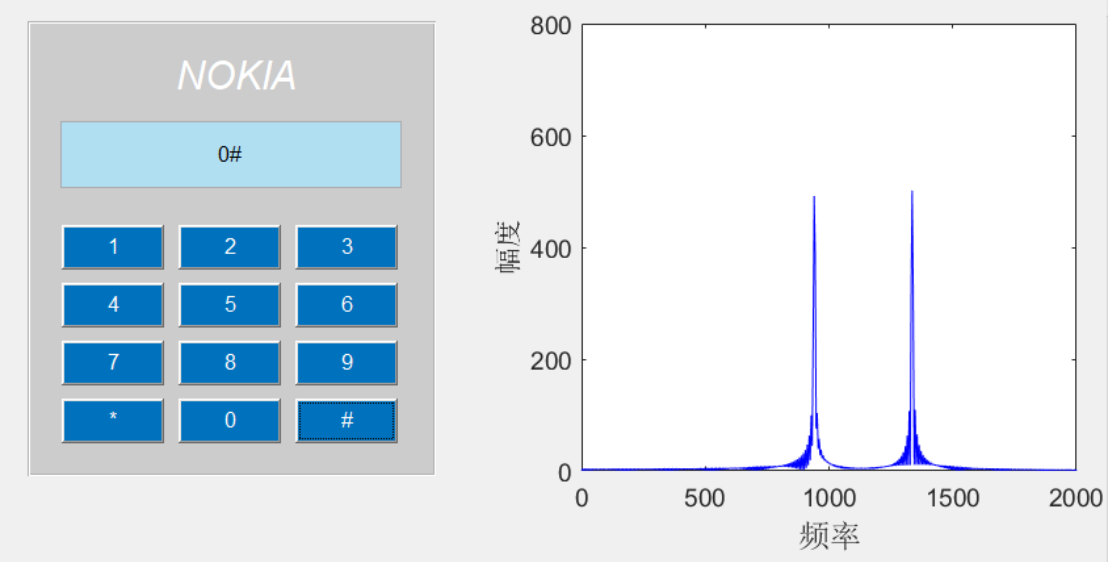

800
600
400
200
0
幅度
0
500
1000
1500
2000
频率


NOKIA
8#
1
2
3
4
5
6
7
8
9
*
0
#


800
600
400
200
0
幅度
0
500
1000
1500
2000
频率


NOKIA
9#
1
2
3
4
5
6
7
8
9
*
0
#


800
600
400
200
0
幅度
0
500
1000
1500
2000
频率



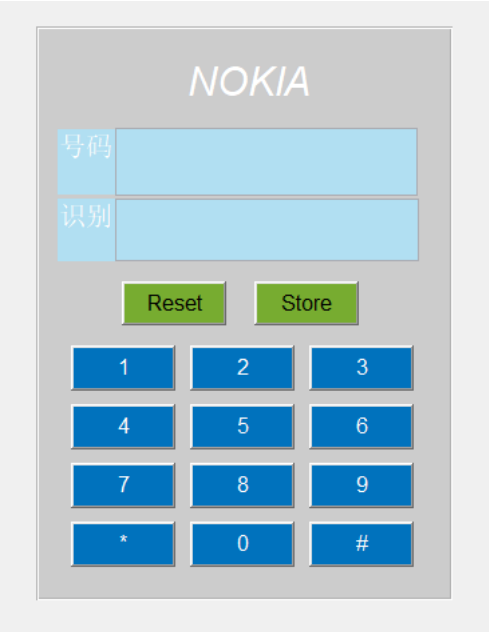
以上是 0~9 号码的频谱图，可以看到结果与表 1 是相对应的（“#”号提示输入结束）。

简单对号码“0”进行分析：通过前述的在画图时把横坐标乘以 4，使得频谱图的频率对应真实频率，方便观察分析；可以看到，号码 0 的频谱图在点 941 和点 1336 处分别出现了一个峰值，这说明号码被成功识别。

4 思考题

添加功能键，实现号码的预存储和来电识别。考虑不同位数号码的存储实现。

4.1 面板的优化设计



添加了复位键 Reset 和存储键 Store，以及一个可编辑文本框。显示窗口中，“号码窗口”动态显示拨号的过程，当按下“#”号键时，对号码进行识别并将识别结果显示在“识别窗口”中。若拨下的号码在预存储的电话簿中，“识别窗口”将显示联系人的名称；若拨下的号码是未知联系人，“识别窗口”的显示结果仅仅是这一串号码。

4.2 号码的预存储程序

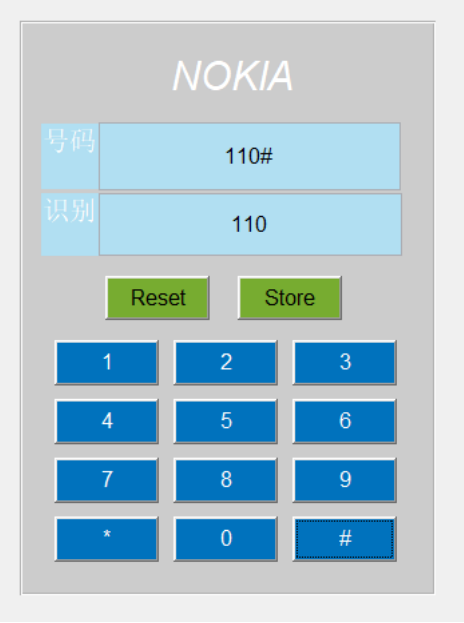
代码如下：

```
291      %预存储程序
292 —    global BOOK;              %定义全局电话簿
293 —    BOOK{1}='10086';          %预存移动、电信、联通的号码
294 —    BOOK{2}='China Mobile';
295 —    BOOK{3}='10000';
296 —    BOOK{4}='China Telecom';
297 —    BOOK{5}='10010';
298 —    BOOK{6}='China Union';
299 —    sum=length(BOOK);
300 —    temp=number;              %识别出的号码
301 —    for j=1:sum-1
302 —        if(strcmp(BOOK{j},number))   %与电话簿中的号码匹配
303 —            temp=BOOK{j+1};
304 —        end
305 —        j=j+2;
306 —    end
307 —    set(handles.edit2,'string',temp);
```

当前存储了三个号码，当输入这三个号码时，会自动进行识别。

4.3 来电识别

识别结果如下：



从运行结果来看：若是预存储在电话簿中的号码，按下“#”确认识别后会显示该电话号码对应的联系人；若是一个未知的号码（不在电话簿中），按下“#”确认识别后显示的是这一串号码。

在需要识别新的号码时，首先要按下 Reset 键清除上一次的识别结果，并清空全局变量 NUM，防止其对下一次识别造成影响。这一功能的代码如下：

```
318      % --- Executes on button press in pushbuttonReset.
319      function pushbuttonReset_Callback(h, eventdata, handles, varargin)
320 —      set(handles.edit1,'string','');
321 —      set(handles.edit2,'string','');
322 —      global NUM;
323 —      NUM=[];
```

对应的是 Reset 按键的回调函数。

4.4 不同位数号码的存储实现

为了使得用户能自行存储常用联系人，开发了存储功能 Store，能实现不同位数号码的存储。

代码如下：

```
310      function pushbuttonStore_Callback(h, eventdata, handles, varargin)
311 —      num=get(handles.edit1,'string');
312 —      name=get(handles.edit2,'string');
313 —      global BOOK;
314 —      sum=length(BOOK);
315 —      BOOK{sum+1}=num;
316 —      BOOK{sum+2}=name;
```

代码的基本思路是在电话簿 BOOK 尾端继续添加号码和联系人姓名。

用户在“号码窗口”输入号码，在“识别窗口”输入联系人姓名，再按下 Store 按键，即可完成存储操作。在下一次拨打这个号码时，“识别窗口”就会显示出对应的联系人。

下面通过几个例子来测试存储功能。

存入 4 个号码：

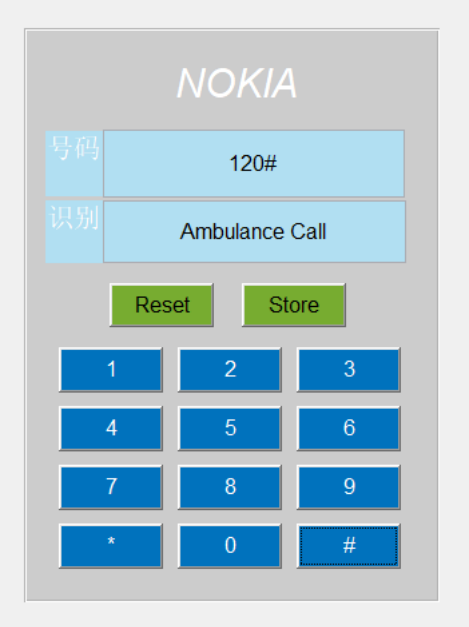
①110 Emergency Call

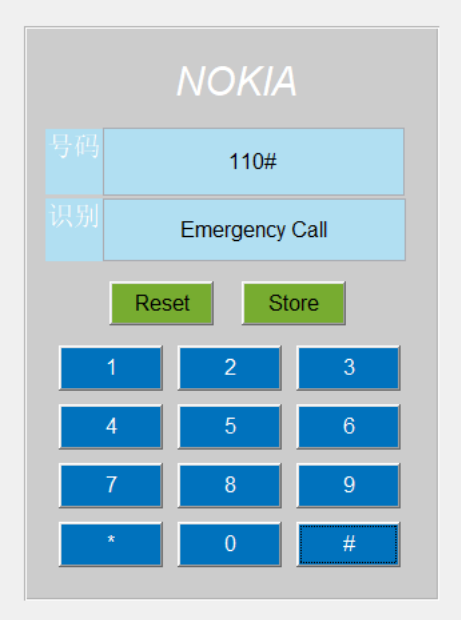
②119 Fire Call

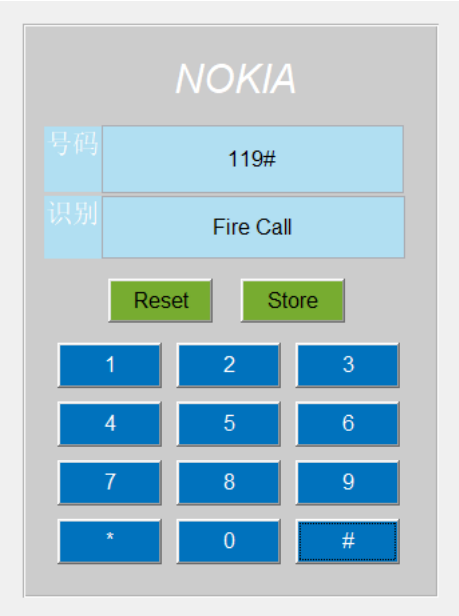
③120 Ambulance Call

④12306 China Railway

经过上述存储操作后，再一次依次输入这些号码并按下“#”确认识别，显示结果如下：







根据以上几个例子的测试，可见达到了预期的效果。

5 实验心得体会

5.1 实验总结

在本次实验中，实现了使用 MATLAB 来进行电话拨号音的合成与识别，以及电话号码的新增、识别和存储的实现，掌握了一些通讯设备生成和识别拨音信号的基本原理和技术，为以后相关的通讯课程和实践实习打下了良好的基础，并且激发了我进一步学习和掌握相关技术的兴趣。

5.2 遇到的困难和收获

①本次实验是我第一次用GUI进行面板设计，由于没有任何这方面的基础，上手比较慢，查看了一些帮助文档和网上的相关教程，但还是遇到了不少的坑。

②在回调函数和相应控件的对应问题上，我因为一些错误操作导致对应关系混乱，也因为对编辑界面一些选项的盲目修改，导致在设计过程中出现了按键按下没有任何反应的情况，这些问题都通过摸索逐步解决，现在对 GUI 图形用户界面的设计有了一定程度的理解和掌握，能进行简单的设计。

③通过实验发现，回调函数的编写其实比较简单，但要注意全局变量和局部变量的问题，变量的作用域直接影响程序运行的结果。

④因为“*”键一次只能删除一位号码，也为了避免上一次的号码输入对下一次的号码识别产生影响，需要设置 Reset 键，释放 NUM 数组中的内容，并清空窗口显示的内容。Reset 键的设计是在仿真过程中根据需要而添加的，并不是一开始设计时就想到的。

⑤反复检查代码后，认为逻辑上并没有问题，但在存储号码和联系人姓名时，有时需要连续两次按下 Store 键才能存储到电话簿 BOOK 中。这个问题还在困扰着我，算是程序的一个小问题，但不影响功能的正常实现。

⑥通过这次实验，在设计和操作过程中感受到了乐趣，能动手设计这样一个与现实生活相联系的电话拨号界面，让我有一点成就感。

附件

tul.fig

tul.m

测试视频.mp4