摘 要

本文是初步 matlab 程序设计的验收进度报告。

为了实现了：

（1）1-9 键双音多频信号的产生，同时设计 GUI 界面来输入响应，在按下键盘的同时播放输出的信号。

（2）根据具体按键，生成的时域和频域的双音多频信号，同时展现在 GUI 界面之中。

（3）根据时域和频域的双音多频信号曲线解码 1-9

（4）设计出 DTMF 发生和接收界面

关键词：DTMF GUI MATLAB 双音多频

一、 设计目标

（1）1-9 键双音多频信号的产生，同时设计 GUI 界面来输入响应，在按下键盘的同时播放输出的信号。

（2）根据具体按键，生成的时域和频域的双音多频信号，同时展现在 GUI 界面之中。

（3）根据时域和频域的双音多频信号曲线解码 1-9

（4）设计出 DTMF 发生和接收界面

二、 设计思路和设计步骤

2.1 双音多频（DTMF）信号的发生

2.1.1 任务要求

学习并实现 DTMF 信号的产生，并绘制时域和频域的图线。

2.1.2 原理简述

DTMF 简介

双音多频 DTMF（Dual Tone Multi Frequency），双音多频，由高频群和低频群组成，高低频群各包含 4 个频率。一个高频信号和一个低频信号叠加组成一个组合信号，代表一个数字。DTMF 信号有 16 个编码。利用 DTMF 信令可选择呼叫相应的对讲机。

高频群 Hz 功能 低频群 Hz	1209	1336	1477	1633
697	1	2	3	A
770	4	5	6	B
852	7	8	9	C
941	*	0	#	D

DTMF 的技术指标 s

对于连续信号的数字采样，在电信应用中的采样频率普遍采用 8khz 的大小，根据奈奎斯特取样定理，因为 8khz 的一半是 4khz，我们所用到的最高频率是 1633hz，小于 4khz，完全足够我们完成对信号的频域分析。[1]

DTMF 信号在传输时，每秒传输 10 个号码，或者每个号码 100ms。每个号码传送过程中，信号存在时间至少 45ms，且不多于 55ms，50ms 的其余时间是静音。 [1]

在每个频率点上允许有**不超过±1.5%的频率误差。任何超过给定频率±3.5%的信号，均被认为是无效的，拒绝接收。** [1]

2.1.3 实现步骤

时间连续的 DTMF 信号为$x(t) = sin(2\pi \mathrm{t} f_{\mathrm{r}}) + sin(2\pi \mathrm{t} f_c)$。

对信号进行取样就得到 $\mathrm{x}(n) = \sin\left(2\pi n \frac{f_{\mathrm{r}}}{8k}\right) + \sin\left(2\pi n \frac{f_c}{8k}\right)$

f_{r}为行的频率，f_{c}为列的频率

根据电信应用普遍参数设计为采样频率 8khz，采样时长 45-55 ms 之间，可以获得最大范围点为 $(55-45)*8000*0.001=80$ $45*\frac{8000}{1000}=360$ 一共是 440 个点

时域数字信号生成的实现步骤：

（1）生成时域自变量 x，x = [0:1/f : T];（f 是采样频率,T 是采样时长）

（2）生成时域值 y , sin(2*pi*x*rowF(i))+sin(2*pi*x*columnF(j))（$\mathrm{i}=1..4, \mathrm{j}=1..4$）（rowF 和 columnF 分别是行列的频率值）

时域信号转换成频谱实现步骤：

（1）用 fft 函数对数字信号的 y 向量做快速傅里叶变换，再用 fftshift 对得到的结果进行调整，得到 0 频率分量在中央的函数图像 fft_y。fftshift(abs(fft(y))./(Len/2);

（2）计算 fft_y 对应的频率下标，Len 代表取样的点数。

设 N 为取样点数，根据第 n 点对应频率$f = n \times \frac{f_s}{N}$。

得(-Len/2:Len/2-1)*(fs/Len);

2.2 双音多频（DTMF）信号的单音解码 FFT 算法

2.2.1 任务要求

由低频和高频时域信号通过 FFT 运算得到频域谱线，通过谱线中对应的信号进行单音解码，得到解码后的信号频率。推算出输入数字。

2.2.2 原理简述

对我们生成的信号序列进行快速傅里叶变换，由原来的时域波形转化为频域谱线，原信号对应的频率会在频谱上分离出来。通过分离的谱线呈现出的高峰来确定这个信号所含的频率分量。

我的算法在对测算频率高峰的时候，通过快速傅里叶变换得到的频率会与定义的标准值有一定误差，通过观察发现误差都基本在 15 以内，所以我设置一个算法过滤后，得到 FFT 变换后的低频和高频另个分量。

我们对数字信号 fft 得到的结果与 FFT 的设置的点数有关，不具有普适性，对幅度大小进行归一化，根据相关知识，对 FFT 的过程进行改造得到

```
y=fftshift(abs(fft(y))./(T/2)./(T/2);  T=400 为采样长度
```

对于不同的采样点数，幅值的大小也可以保持一致，为代码复用打下基础。

2.2.3 实现步骤

在获得时域与频域的数据后，有两个目的，一个是实现频点的检测，找出频域中高能频率分量，一个为低频和一个高频的频率；另一个是根据频率的获取得到他们与标准频率的误差和在满足最小误差后判定为匹配成功输出结果。

获取高能频率的具体步骤：

（1）在这里我们发现我们只需要取最大值幅度（y 坐标代表幅度），对应的频率（x 坐标代表频率）就是我们 FFT 变换后得到的低频和高频。

由对称关系，我们得到其中一半的频率，同时取最大幅度的频率 index1 和第二大幅度的频率 index2，判断 abs(index1)与 abs(index2)的大小，使得返回值低频在前，高频在后。

代码如下：

```
function [fLen,freqs] = getTargetFreqs(~,y)


    [~,index1]=max(y(1:length(y)/2));
    y(index1)=0;
    [~,index2]=max(y(1:length(y)/2));
    x_tmp=-3999:20:4000;
    if abs(x_tmp(index1))>abs(x_tmp(index2))
        tmp=x_tmp(index1);
        x_tmp(index1)=x_tmp(index2);
        x_tmp(index2)=tmp;
    end
    freqs=[abs(x_tmp(index1)),abs(x_tmp(index2))];
    fLen = length(freqs);

end
```

获取频率所对应按键的具体步骤：

（1） 由上一个函数得到 2 个值，如果过程出错则，输出没有找到 isFind=0

（2） 在 getFreqs 中存储着 1-16 个数的一对标准频率。

（3） 低频与在标准频率误差小与等于 15 和高频与标准频率误差小于等于 15 的同时，向 target 变量输出找到的数字。

```
function [isFind,target,error1,error2] = findTargetBtn(app,fLen,freqs)

    isFind = 0;
    target = '';
    error1 = inf;
    error2 = inf;
    num=1;
    if fLen==2
        for i=1:4
            for j=1:4
                [low,high]=getFreqs(app,num);

                if abs(freqs(1)-low)<=15 && abs(freqs(2)-high)<=15
                    if num<10
```

```
                target=num2str(num);
            else
                switch num
                case 10
                    target=  '0';
                case 11
                    target=  '*';
                case 12
                    target=  '#';
                case 13
                    target=  'A';
                case 14
                    target = 'B';
                case 15
                    target = 'C';
                case 16
                    target = 'D';
                otherwise
                    isFind=0;
                end
            end
        else
        isFind=0;
        end
        num=num+1;
    end
end
error1=abs(freqs(1)-low)/low;
error2=abs(freqs(2)-high)/high;
end

end
```

2.3 双音多频（DTMF）信号的单音解码 Goertzel 算法

2.3.1 任务要求

利用傅里叶算法的简化算法——Goertzel 算法，对单音解码算法进行设计。

2.3.2 原理简述

Goertzel 算法原理

对于直接计算离散傅里叶变换 DFT，计算公式是

$$X(\mathrm{k}) = \sum_{n=0}^{N-1} x(n) W_N^{kn} \ (\mathrm{k} = 0,1,\ldots,\mathrm{N}-1)$$

这个式子在输入整个序列后才可以进行运算，而一个复数乘法需要四次乘法和两次加法，计算 X(k)的运算量是 4N 次乘法和 2N+2(N-1)=4N-2 次实数加法。可以说相比 Goertzel 算法运算量要大得多。

Goertzel 利用了$\mathrm{W_N}$的旋转周期性，因为$\mathrm{W_N^{-kN}} = 1$，在 X(k)左右分别乘上这个系数计算公式仍然成立：

$$X(k) = W_N^{-kN} \sum_{n=0}^{N-1} x(n) W_N^{kn} = \sum_{n=0}^{N-1} x(n) W_N^{-k(N-n)} \ (\mathrm{k} = 0,1,\ldots,\mathrm{N}-1) \quad (*)$$

由这个公式我们可以得到两个信号的序列卷积和，于是设

$$y_k(n) = \sum_{r=0}^{r=N-1} x(r) W_N^{-k(n-r)} u(n-r)$$

当 N=n 时，

$$y_k(n) = \sum_{r=0}^{r=N-1} x(r) W_N^{-k(n-r)} u(n-r)$$

接着可以对这个离散卷积和形式的式子做 z 变换，得出频率响应

$$H(z) = \frac{1}{1 - W_N^{-k} z^{-1}}$$

由

$$H(z) = \frac{Y(z)}{X(z)}$$

可得

$$Y(z) - W_N^{-k} z^{-1} Y(z) = X(z)$$

可以得到差分方程的形式

$$y_k(n) = W_N^{-k} y_k(n-1) + x(n)$$

此时 Goertzel 算法如上述公式所示，比起 FFT 而言最大的优势是可以在输入的同时进行运算。

实现步骤

（1） 枚举识别所需的八种频率

（2） 对每种频率进行递推方程的计算

（3） 得到每种频率对应的 Goertzel 计算结果

（4） 再进行挑选就得到一对标准频率

（5） 再查表就可以得到具体输入的数字

代码如下：

```
f = [697 770 852 941 1209 1336 1477 1633];
freq_indices = round(f/8000*400) + 1;
dft_data = goertzel(app.y_sequence,freq_indices);

[index1,index2]=getMaxTowNum(app,abs(dft_data));
freqs=[f(index1),f(index2)];
[~,target,~,~]=findTargetBtn(app,2,freqs);
```

2.4 双音多频（DTMF）信号序列的生成

2.4.1 任务要求

用 matlab 生成 wav 文件或者文件，并写入计算机的硬盘中。

2.4.2 原理简述

采用一定规定的数据形式来对拨号信进行拼接。

根据采样率为 8000，一个音的长度为 100ms,其中有声部分是 45-55ms，我们设置为 50ms，这样我们就得到 400 个点，在序列生成前后各加上 200 个点，将这 800 个点依次拼接其来，再将每个信号拼接起来，我们就得到有声部分和静默部分的一个总合集。

使用`audiowrite('file1.wav',app.y_sequence,8000)`可以输出wav文件，来保存波形数据。其中name是波形文件的名称，y是波形时域的数据，fs是音频采样率。

其次在我研究matlab 的读写函数后，我也设计成可以保存为文件格式，其中的数据排列。

2.4.3 实现步骤

生成序列的具体步骤：

（1） 根据拨号的个数创建数组

（2） 对每个拨号音构造出静默 200 点，有声 400 点，再静默 200 点的数据

（3） 拼接构造的数据

生成一段拨号序列

```
function generateOneSequence(app,y_TimeDome)
        y_TimeDome=[zeros(1,200),y_TimeDome,zeros(1,200)];
        app.y_sequence=[app.y_sequence,y_TimeDome];

    end
```

2.5 双音多频（DTMF）信号序列的端点切割

2.5.1 任务要求

基于生成的序列进行端点切割。

2.5.2 原理简述

由我们生成的信号序列的规律可以进行端点切割，先将数据分成按拨号数量的数据组，其次是我们知道一个数据是由 200+400+200 点集的组合，所以我们可以在截取 400 点集的集合再调用我们之前分装的 FFT 和 Goertzel 函数来分个检测出拨号数字并添加到数组中。

2.5.3 实现步骤

端点切割的具体步骤：

（1） 确定生成的信号序列的排列

（2） 通过计算总信号序列个数除以采样点个数得到拨号数量

（3） For 循环进行切割，得到处理的数据

代码如下：

```
    app.num_of_all=ceil(length(app.y_sequence_All)/800);
    tic
    for i=1:app.num_of_all
        tmp=app.y_sequence_All(1+800*(i-1):800*i);
        app.y_sequence=tmp(201:600);
        app.y_sequence=FFT(app,app.y_sequence);
        [fLen,freqs]=getTargetFreqs(app,app.y_sequence);
        [~,target,~,~]=findTargetBtn(app,fLen,freqs);

    app.DecodingOutPut_str=sprintf('%s%s',app.DecodingOutPut_str,
    target);
    end
```

2.6 双音多频（DTMF）信号序列的解码

2.6.1 任务要求

对序列切割出的区间进行端点解码。

2.6.2 原理简述

对信号序列的解码思路和之前的单音解码方式一样，在 FFT 变换后，或者 Goertzel 算法后，求出频域能量最大的两个点，得到一对解码后频率。对比标准频率后，即可求出拨号数字。

2.6.3 实现步骤

FFT 算法的解码具体步骤：

（1） 得到每个拨号音区间

（2） 进行 FFT 变换

（3） 提取频谱中范围在误差以内的频率，得到拨号数字。

```
function FFTButtonValueChanged(app, event)
          app.DecodingOutPut_str='';
          app.FFT_time.Text='计算时间:';
          app.num_of_all=ceil(length(app.y_sequence_All)/800);
          tic
          for i=1:app.num_of_all
              tmp=app.y_sequence_All(1+800*(i-1):800*i);
              app.y_sequence=tmp(201:600);
              app.y_sequence=FFT(app,app.y_sequence);
              [fLen,freqs]=getTargetFreqs(app,app.y_sequence);
              [~,target,~,~]=findTargetBtn(app,fLen,freqs);

          app.DecodingOutPut_str=sprintf('%s%s',app.DecodingOutPut_str,
          target);

          end
          t1=toc;
          app.FFT_time.Text=strcat(app.FFT_time.Text,num2str(t1));
          app.FFT_decoded_num.Text=app.DecodingOutPut_str;

      end
```

2.7 双音多频（DTMF）应用的界面设计

2.7.1 任务要求

设计 DTMF 发生和接收的界面，有良好的交互性，使得符合输入输出交互模式。

2.7.2 原理简述

利用 AppDesigner 进行 UI 的拖拽设计。

用面向对象的设计思路进行回调函数的编程。

2.7.3 实现步骤

设计上主要分为输入数字区和展示区，左侧是输入数字，与用户交互拨号，就像是用户正常拨号界面，解码输出的是接收端接受用户的输入的数字。在播放数字时，也有声音进行反馈。其次我在设计的时候参考手机拨号输出界面。设计了情况和退格，我在考虑到清空和退格应该也有音效进行反馈，比较跟手。右侧是渲染图标。

三、 运行结果及分析

3.1 单音的产生和识别

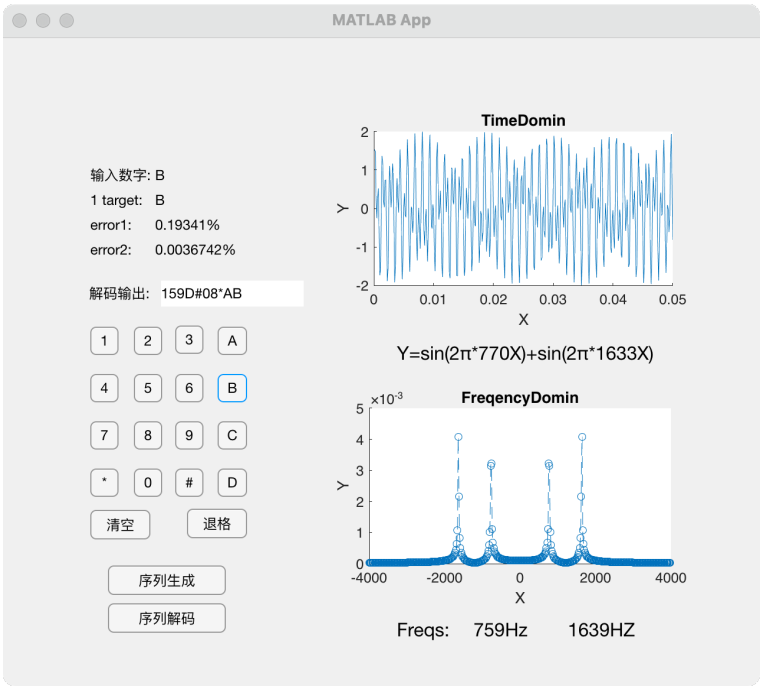

图 1 按键"159D#08*AB"的识别结果和误差

3.2 信号序列的生成

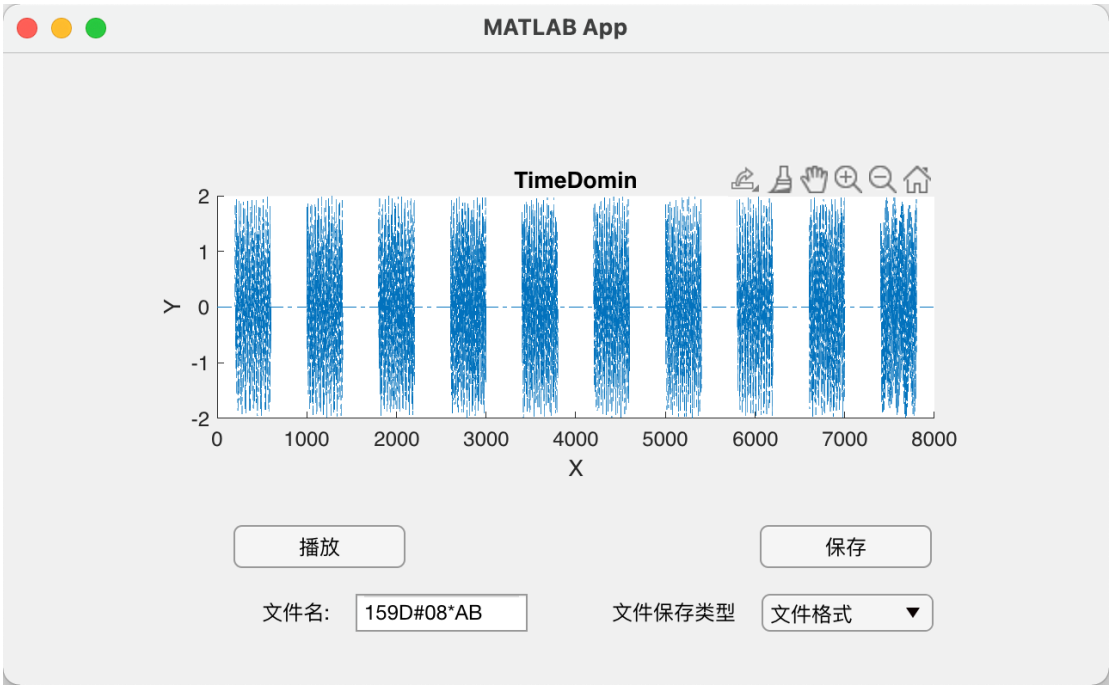

图 2 发生器

这里的文件名可以更改，如果选择 wav 格式，文件名也会随之改变，默认的解码文件名也会随着定义生成信号的文件名的改变而改变。

发生器会获取主界面输入的信号序列，并以时域波形的形式展现出来，当我们在主界面更改信号序列的时候，该界面也会随着改变。播放可以播放当前波形图。

3.3 信号序列的解码

因为是默认获取刚刚生成的文件，文件名也是自己刚刚设置的文件名，点击读取就可以通过文件获得这份信号序列，点击播放就可以听到该波形图。点击 FFT 或者 G 解码就可以得到解码后的结果以及他们的计算时间。

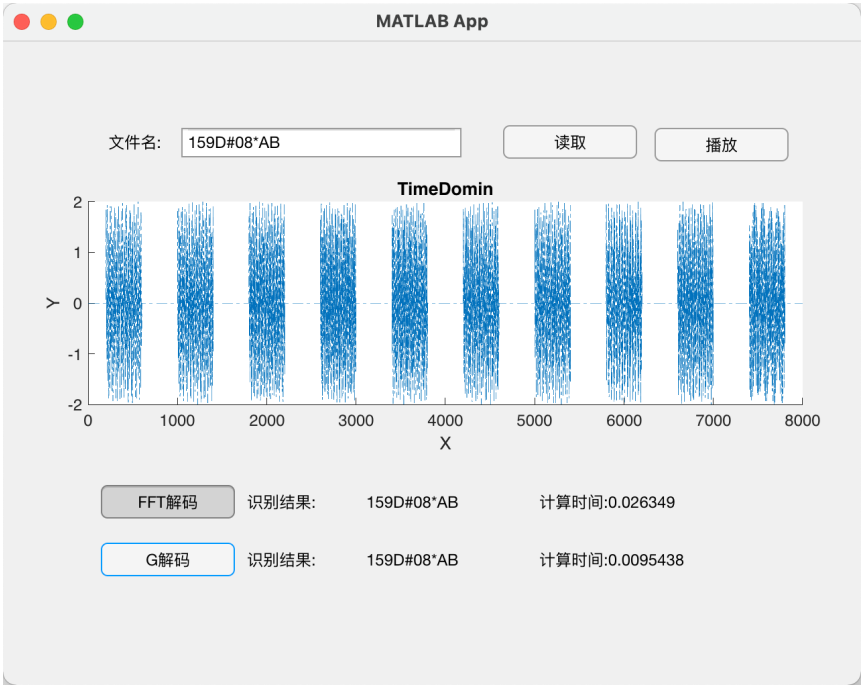

图 5 接收器

3.4 部分 GUI 界面设计图

图 6 拨号器

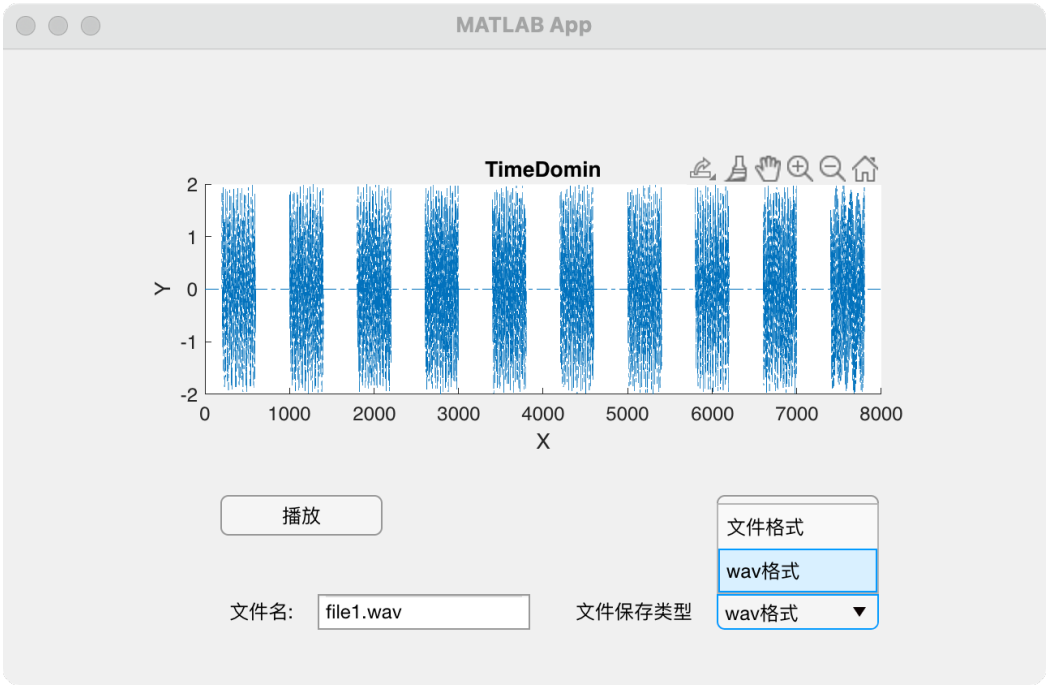

图 7 wav 格式选择

四、 参考资料

[1]张雅琪,基于 Matlab 的双音多频信号仿真[J],DataBase 数据库

[2]徐阿勇,基于 MATLAB 的 DTMF 技术计算机模拟[J],温州师范学院学报(自然科学版),2005

[3] DTMF 百度定义
(https://baike.baidu.com/item/DTMF/3106215?fr=aladdin)

[4]MATLABA 官网查询函数使用
(https://ww2.mathworks.cn/?s_tid=gn_logo)