



西北工业大学

课程设计报告

课程名称: 飞行器信息系统课程设计

课程编号: UCAP21005

班 级: CA021901

学 号: _____

姓 名: _____

成 绩: _____

2022 年 5 月

目录

I 基于 SDR 的飞机通信导航监视系统信号仿真设计	1
一、 基本要求	1
二、 基本原理	1
三、 任务实施	2
3.1 基于 simulink 的 AM 通信系统仿真	2
3.2 基于 GUI 的 AM 通信系统仿真	7
3.2.1 需求分析与设计	7
3.2.2 任务结果与分析	8
3.3 基于 GUI 的 VOR 系统测向仿真	10
3.3.1 需求分析与设计	10
3.3.2 任务结果与分析	12
3.4 基于 GUI 的 ILS 系统偏离测量仿真	16
3.4.1 需求分析与设计	16
3.4.2 任务结果与分析	17
四、 完整代码	19
II 语音信号采集处理设计	31
五、 基本要求	31
六、 基本原理	31
七、 任务实施	32
7.1 思路分析	32
7.2 基于间断分析的域分析	33
7.3 基于整体分析的域分析	34

7.4 基于 GUI 的域分析	36
7.4.1 需求分析与设计.....	36
7.4.2 任务结果与分析.....	37
八、 完整代码.....	39
九、 课程设计总结.....	44
9.1 结论与特色	44
9.2 不足与展望	45
9.3 收获与建议	46
参考文献	47

Part I

基于 SDR 的飞机通信导航监视系统信号仿 真设计

一、 基本要求

1. 熟悉 Matlab/LabVIEW 操作环境；
2. 实现调幅或单边带的工作方式进行双向通话的功能仿真，理解高频/甚高频通信系统的基本原理。
3. 实现模拟机载与地面 VOR 系统收发信号模拟，完成测向功能仿真；
4. 模拟 ILS 系统机载与地面系统互通，完成航向和下滑道偏离测量功能仿真（选做）；
5. 实现模拟机载应答机与地面二次雷达系统互通，实现询问和应答脉冲的识别（选做）

二、 基本原理

基于基本要求，分别对原理进行阐述。

1. 由于前置课程实验平台多为 *matlab*，使用较为熟练，因此本次课程设计使用 *matlab*，可以选择的具体平台包括 *simulink* 仿真环境和 *GUI* 图形界面等。
2. 高频通信系统和甚高频通信系统的基本原理可以简要概括为“调幅发射”和“接收解调”，载波频率分别在十兆和百兆级别，如果仅通过 *m* 文件设计，所需时间较长，无法满足这一频率，因此如果要切实实现这一频段的仿真，需要通过 *simulink* 仿真。参考文献^[1] 提供了基于 *RTL – SDR* 硬件的 *simulink* 仿真方法，可以在有 *SDR* 接收机的条件下搭建 *simulink* 模块^[2]，实现实时高频信号或本地已调制的话音信号的接收解调。

但上述分析只考虑了“接收解调”这一过程，对于“调幅发射”暂时没有好的办法。原因在于现在大多数无线电设备均不采用 *AM* 的调制方法（调制效率低，消耗功率大），*SSB* 发射机由于所需的接收设备对同步要求高的原因难以找到；对于简单易得的无线电发射设备，只有对讲机，但目前市面上的对讲机只采用 *FM* 方式，因此综合考虑，对于“调幅发射”这一部分的仿真，采用另一种方式 *GUI* 实现。由于 *GUI* 方便灵活的特点，也可以顺便加入“接

收解调”的过程。

在任务介绍环节，老师提及可以采用 PartII 产生的话音信号作为调制信号，因此最终对 *GUI* 的设计也包括计算机声卡采集信号的起始过程。由于采用频率存在上限，一般为 48000Hz，无法达到高频/甚高频的工作频率，因此初始设置载波频率为 10kHz。

3. *VOR* 的工作原理较为简单，即塔台持续发射基准相位和可变相位两个信号，后者自磁北方向开始顺时针旋转，相位随 *VOR* 台的径向方位而变，通过比较两者的相位差即可得到飞机此时相对 *VOR* 地面台的位置。

在任务实施中，考虑在地图中分别标志塔台和飞机的位置，基于两个位置坐标计算得到实际的飞机的 *VOR* 方位角。在标志位置的同时产生上述两个信号，分别完成副载波调频、调幅等过程后作为发射信号。对这一信号进行解调(对应超外差接收机)，通过 30Hz 低通滤波得到可变相位信号，通过副载波带通滤波、鉴频、包络检波及滤波得到基准相位信号。在相位比较方法上选取较为准确的相关分析法^[3]，得到这一测量 *VOR* 方位角后可与前者比较判断系统准确性。

4. *ILS* 的工作原理与 *AM* 相似，只是发射信号调制频率不同、强度可能不同且分开一定角度。接收信号解调后就行傅里叶变换得到最大值即可比较判决。

三、任务实施

基于基本原理，对于基本要求 2 实现思路是采用 *simulink* 搭建通信系统模型完成高频/甚高频通信系统接收解调仿真，采用 *GUI* 完成完整通信系统仿真。

3.1 基于 *simulink* 的 *AM* 通信系统仿真

实施过程是创建一个新的 Simulink 模型，放置与 RTL-SDR 接口的 Simulink 库中的组件，并实现解调器。将接收机设计为接收 AM-DSB-TC 音频信号，并将解调的音频信息输出到计算机的扬声器或耳机。如果没有 RTL-SDR，或者无法发送可以接收的 AM-DSB-TC 信号，则仍然可以通过将 RTL-SDR 接收器块替换为 RTL-SDR 数据块。具体实施过程^[4] 如下。

- 初始准备。下载 RTL-SDR 支持包^[1]，接好硬件后，创建新 Simulink 模型，打开 Simulink Library Browser，放置 RTL-SDR 接收器块。
- RTL-SDR 接收器设置。从 RTL-SDR 收音机的通信系统工具箱支持包放置 RTL-SDR 接收器并打开其参数窗口，将“中心频率”和“调谐器增益”的

“源”更改为“输入端口”。在“采样率”字段中输入“240e3”，将 RTL-SDR 设置为以 240kHz 的速率进行采样。将“输出数据类型”下拉菜单设置为“单”，并在“每帧样本”框中输入“4096”。重命名 RTL-SDR 接收器为 o/p fs = 240kHz。

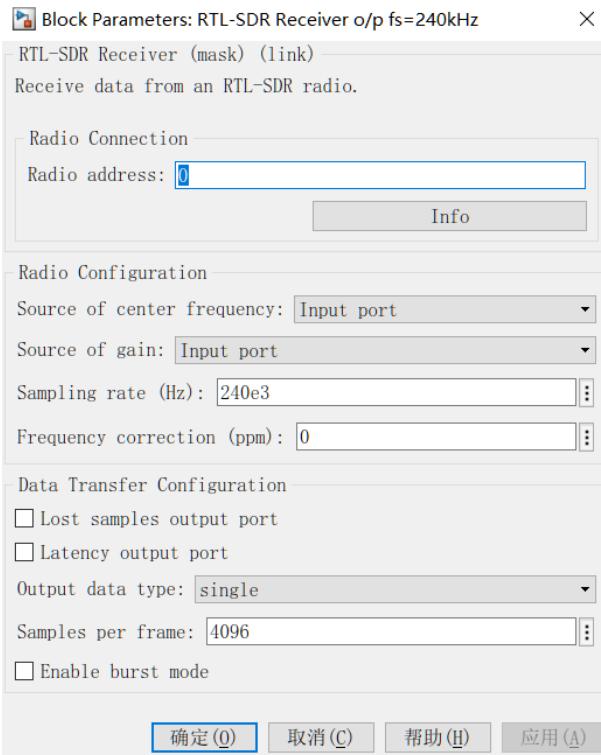


图 1 RTL-SDR 接收器设置

添加三个常量块，分别修改为 AM 信号频率 (Hz)，偏移频率 (Hz) 和调谐器增益 (dB)。将 AM 信号频率的“常数值”设置为要接收的 AM 信号的中心频率，例如 433.9MHz 的“433.9e6”。输入“-40e3”作为偏移频率的值，将偏移量设置为 40kHz。将调谐器增益块设置为默认“30”。将这些模块连接如图2所示。

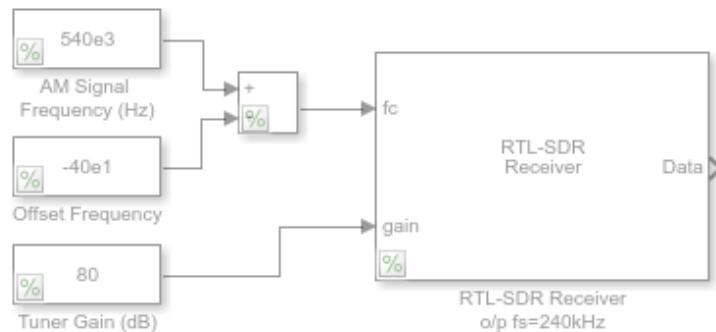


图 2 RTL-SDR 接收器模块连接示意图

- 导入 RTL-SDR 数据块。如果没有 RTL-SDR 或无法传输 AM-DSB-TC 信号，导航到 > RTL-SDR 书库 > 其他工具，并放入导入 RTL-SDR 数据块。更改“文

件名”参数以引用 *am_dsb_tc.mat*。将“输出框架尺寸”设置为“4096”，应用更改并关闭参数窗口。如果此过程成功，则该块应显示右下角记录信号的采样频率 (240kHz)。从该块输出的信号应等于由上述 RTL-SDR 配置输出的信号。当记录该信号时，RTL-SDR 调谐到 433.96MHz，偏移频率设置为 40kHz。

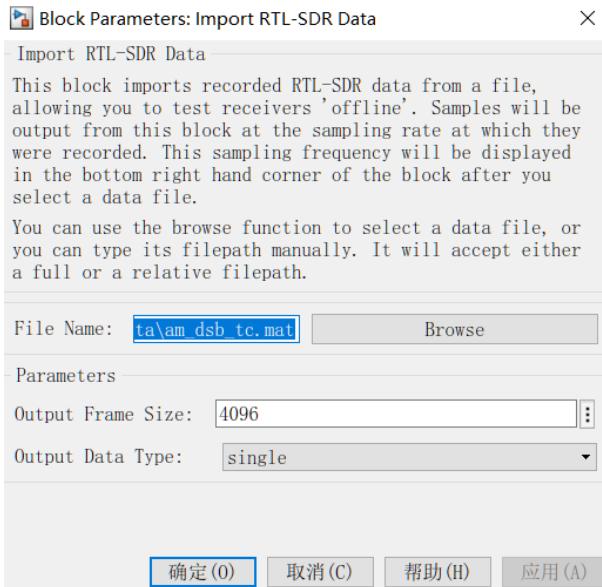


图3 RTL-SDR 数据块设置

- 放置和配置实现带通滤波器。从 DSP 系统工具箱 > 过滤 > 过滤器设计放置带通滤波器。打开其参数窗口并将设置更改，如图4所示。



图4 带通滤波器设置

这将滤波器设置为通过 25kHz 至 55kHz 之间的频率，这意味着仅允许 IF AM-DSB-TC 信号通过。重新命名该通道滤波器 $f_{pass} = 40\text{kHz}$ 。

- 解调器设置。引入限幅器和 *FIR Decimation* 模块。该块对应用于其输入端口的数据进行抽取，可以将采样频率降低整数因子。它还对数据进行低通滤波，以确保不发生混叠。将其中一个放在模型中并打开其参数窗口。将“FIR 滤波器系数”和“抽取因子”进行更改。

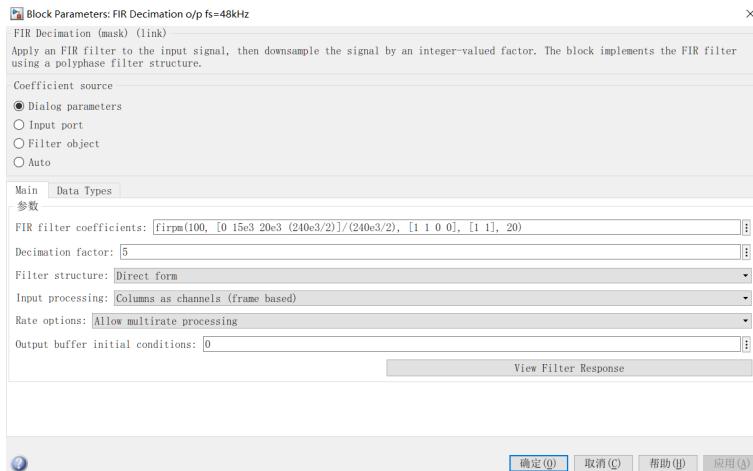


图 5 解调器设置

这将抽取因子配置为 5（即从 240kHz 到 48kHz 的速率变化），并将频率传递到 15kHz。在“value”选项下拉菜单中选择“允许多速率处理”，然后应用更改。重命名此块 FIR 抽取 o/p $fs = 48\text{kHz}$ 。

- 添加范围和音频输出模块。导航到 > DSP 系统工具箱 > 接收器。在模型中放置两个频谱分析仪块，重新命名一个频谱分析仪调制和另一个频谱分析仪解调。接下来放置时间范围和音频设备块。打开 > Simulink > Math Operations，找到 Matrix Concatenate 块。将其放在模型中，然后将所有这些附加到框图中，在每个与 Matrix Concatenate 块的输入连接上，并给出有意义的信号名称。

- 连接模块，如图6所示

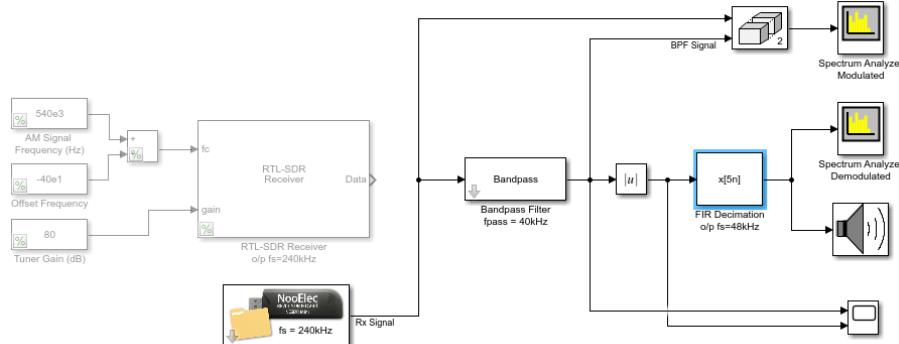


图 6 完整 simulink 仿真连接图

- 配置属性。打开“时间”选项卡，将“时间跨度”更改为“512 / 240e3”。这将限制仅显示 512 个单独样本的范围。应用更改然后关闭窗口。
- 仿真结果。以 RTL-SDR 数据块作为输入信号的情况为例，各路信号时域显示如图7所示，频域显示如图8和图9所示，此外可以听到断断续续的语音信号（歌声）。

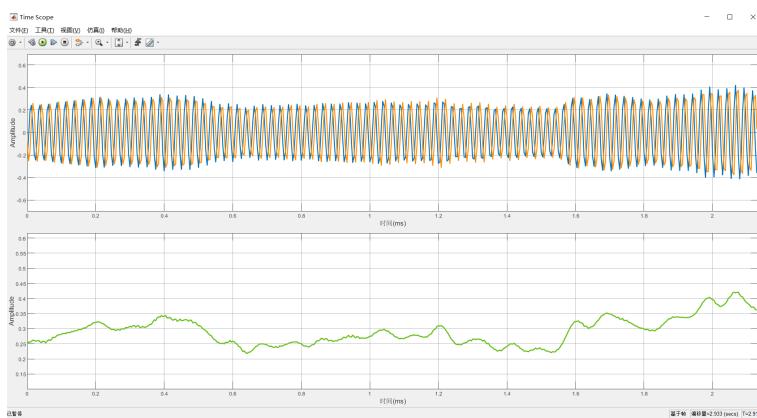


图 7 信号时域显示

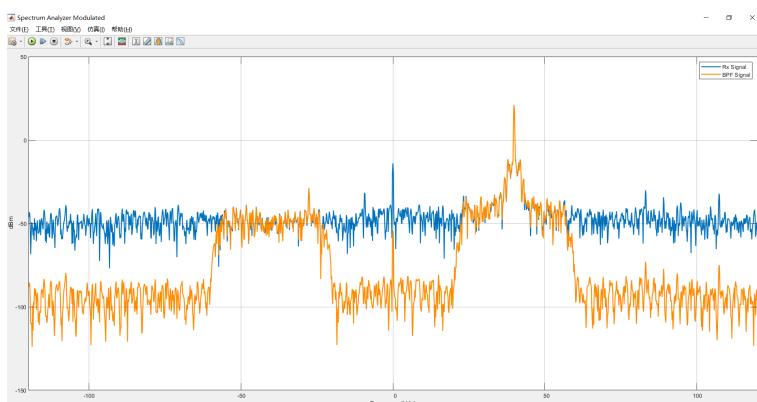


图 8 已调信号频域显示

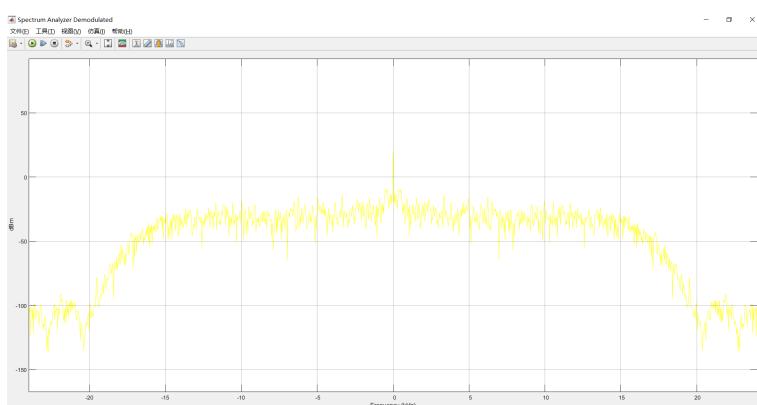


图 9 解调信号频域显示

输出的三张结果图与图6中线路连接相对应。

结果的简单分析：

图8中，蓝线表示从 RTL-SDR 接收的信号的频谱，橙色表示带通滤波后的信号。AM 信号以大约 40kHz 调制，这应该是足够高的频率，使包络检测器正常工作。图9显示了经过包络检测后解调信号的频谱。在这里信息已成功转移回基带。

如果采用 RTL-SDR 接收机，解调的音频质量很差，使用耳只能听到音频信号为“嘶嘶声”，原因可能是 FIR Decimator 中的低通滤波器没有被去除；

如果采用 RTL-SDR 数据块作为输入信号，解调后可以接收到信号音频，声音相对清晰，可以判别出音调，缺点是速度很慢，卡顿严重，即时采用加速仿真也是如此。

3.2 基于 GUI 的 AM 通信系统仿真

3.2.1 需求分析与设计

GUI 实现是课程设计基本要求 2 的主要部分，这一部分的实现过程包括两部分，一是 *fig* 的绘制，二是子函数设计。

图形的绘制：新建 *fig*，绘制标题、画布、按钮和可编辑文本等，如图所示。两个画布分别用来绘制时域图和实时频域图。按钮设计包括开始、停止、播放、调制、传输和解调等功能，可编辑文本用来设置载波频率。

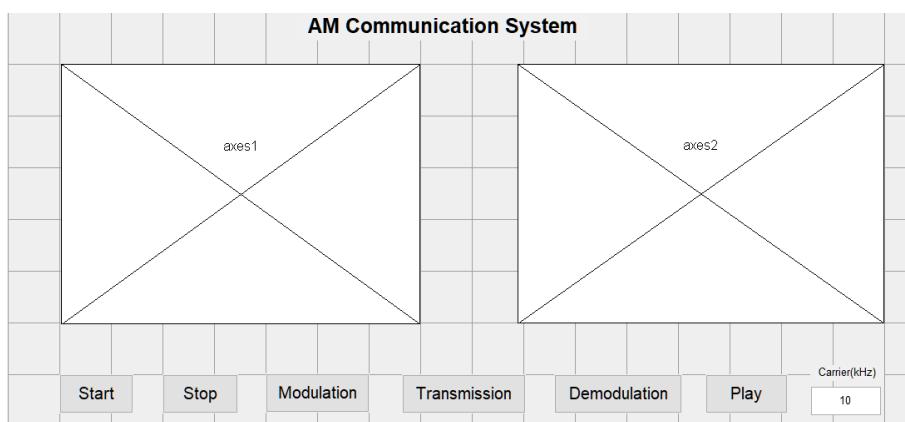


图 10 GUI 界面设计图

子函数设计：

- 录音器初始化：采用 *audiorecorder* 函数，初始化参数为采样率 48000Hz。设

置定时器和其他全局变量。

- 获取原始录音数据；
- 开始按钮：用 *record* 函数即可实现；
- 结束按钮：用 *stop* 函数即可实现；
- 播放按钮：用 *audioplayer* 函数获取数据，*play* 函数播放；*guidata* 函数更新数据；设置播放选项，用于识别播放信号为调制信号和解调信号。
- 实时显示：*getaudiodata* 获取数据，时域图和频域图绘制代码与思路一相同，注意需要用 *drawnow* 更新。绘图前需确定画布句柄；
- 可编辑文本设置：用 *get* 和 *str2double* 等函数实现输入转换，并用 *guidata* 函数更新数据；
- 信号调制：获取更新后或者初始化的载波参数，完成调制任务、绘制时域图和频域图，并将已调信号数据导出到基础工作区；
- 信号传输：采用加性高斯白噪声信道，对已调信号加噪处理，绘制时域图和频域图，并将已调信号数据导出到基础工作区；
- 信号解调：采用相干解调，对信号解调后绘制时域图和频域图，并将解调信号数据导出到基础工作区；

子函数的完整代码见章节四，其与设计的 fig 文件相结合即可得到下一节的结果。

3.2.2 任务结果与分析

以某短时语音作为输入为例，完整的时域图和频域图如图11所示。播放已录语音，发现较为清晰。

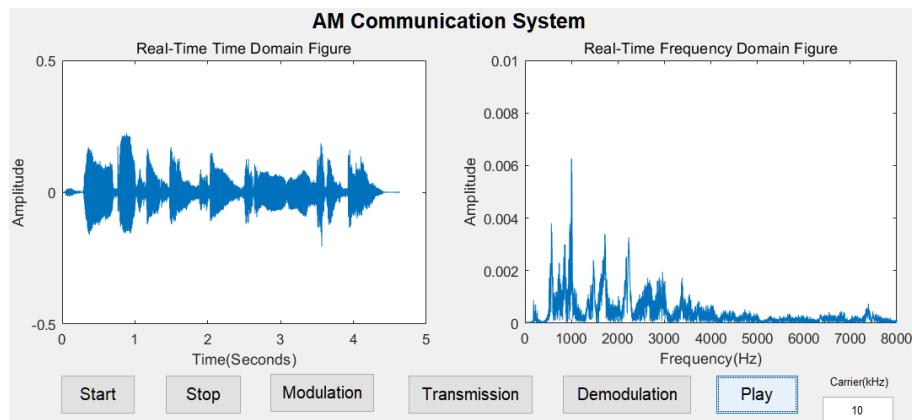


图 11 调制信号时频域示意图

初始化载波频率不变，对信号进行调制，调制后得到的信号时域图和频域

图如图12所示。

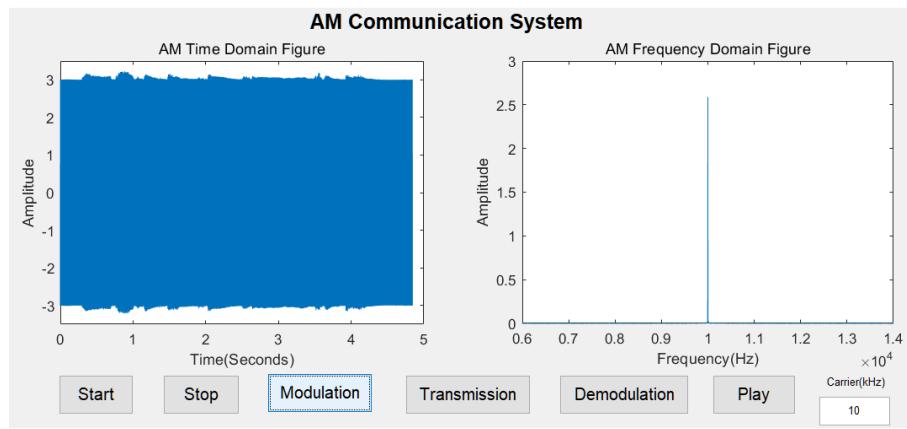


图 12 已调信号时频域示意图

对信号进行信道传输，可以认为接收器得到的信号时域图和频域图如图13所示。

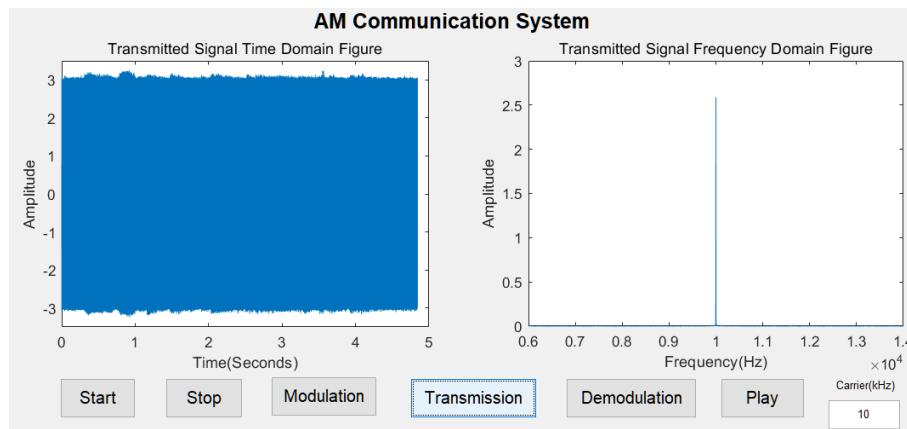


图 13 接收信号时频域示意图

对信号进行相干解调，解调输出的信号时域图和频域图如图14所示。播放解调语音，发现存在嘶嘶声。

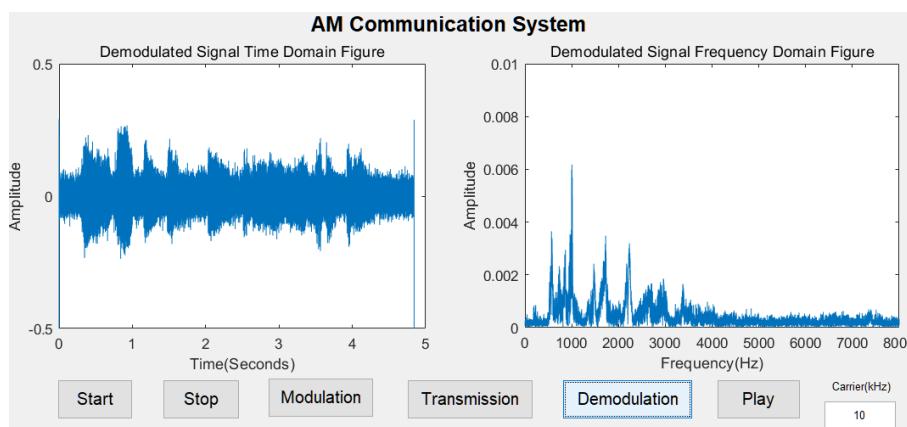


图 14 解调信号时频域示意图

对于图14和图11可以发现，解调效果较好。

本次任务的结论和分析如下

- GUI 实现了信号的产生、调制、传输和解调等完成过程，从原理上模拟了高频/甚高频通话系统。虽然 GUI 展示的效果是单向的，但实际上只要重复这一过程就是双向的；
- GUI 实现基本要求 2 的优点在于其灵活可调，运行速度快，对于 10s 以内的音频，可以做到瞬时处理和解调；
- 任务完成的主要不足之处在于解调后没有进一步进行循环处理，导致最终得到的语音信号嘶嘶声音较大，这一点可以采用自适应滤波的办法解决。

3.3 基于 GUI 的 VOR 系统测向仿真

3.3.1 需求分析与设计

对于基本要求 3，GUI 实现包括两部分，一是 *fig* 的绘制，二是主子函数设计，三是负责计算的子函数。

图形的绘制^[5]：新建 *fig* 制标题、画布、按钮、滑块和可编辑文本等，如图15所示。两个画布分别用来绘制地图和罗盘。按钮设计包括加载图片/初始化、绘点等功能，滑块和部分可编辑文本用来设置所处位置。此外，地图可以做到交互式访问，结合可编辑文本在鼠标点击的情况下显示所处位置。最终结果体现在其他三个可编辑文本上，即实际 VOR 方位角、测量 VOR 方位角和误差。

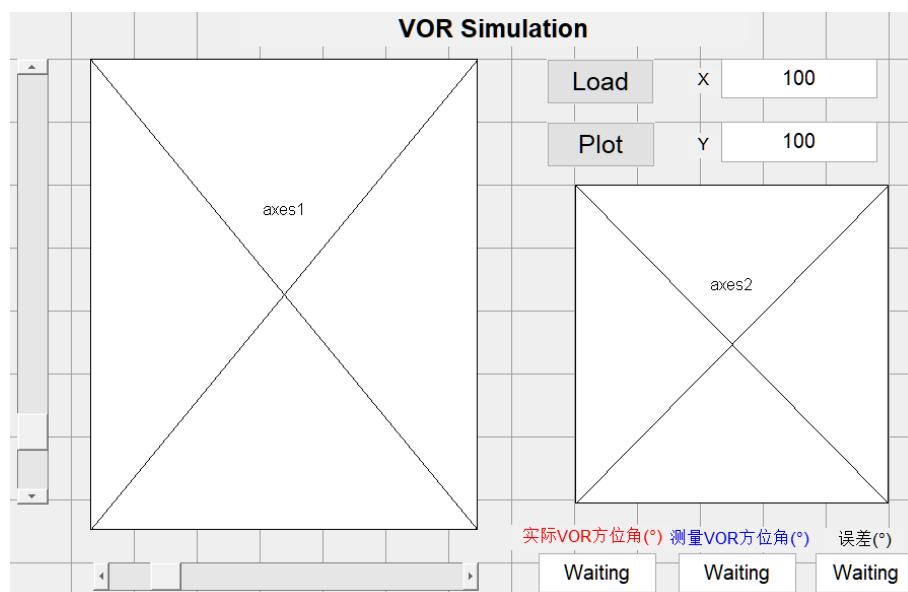


图 15 GUI 界面设计图

主子函数设计：

- 参数初始化：包括初始飞机坐标为(100,100)，初始VOR方位角为303.69度等；
- 滑块设置：交互式设置，拖动滑块后改变的坐标值可以体现在可编辑文本上；
- 可编辑文本：设置两个关于飞机坐标的可编辑文本，做到输入转换；设置三个关于飞机VOR方位角的可编辑文本；
- 加载图片/初始化：在两个图窗中分别加载两张图片，设置塔台位置，默认为(400,300)，并在图中标明；调用鼠标交互函数；
- 鼠标交互函数：获取鼠标所点击的坐标，并在图中相应位置标注；之后分别调用两个计算子函数得到实际方位角和测量方位角，并做到罗盘指针显示和可编辑文本数值显示。为该函数所绘制的所有图形分别定义全局变量，每次调用前进行更新；
- 绘点函数：获取初始化飞机位置、滑块移动后的飞机位置或者手动输入的飞机坐标，在图中相应位置标注；之后分别调用两个计算子函数得到实际方位角和测量方位角，并做到罗盘指针显示和可编辑文本数值显示。此外，为该函数所绘制的所有图形分别定义全局变量，这些全局变量要和加载图片/初始化的全局变量一致，不仅可以做到每次调用自身前进行更新，还可以做到互相更新。

计算子函数设计：

- 实际VOR方位角计算。

默认塔台位置为(400,300)，结合输入的飞机当前坐标通过向量法及分类讨论计算得到基于坐标位置的VOR地面塔台相对于飞机实际的VOR方位角；

- 测量VOR方位角计算。

模拟实际VOR测向原理，基于《民机通信导航与雷达》理论课知识，即产生已调制的基准相位信号和可变相位信号，之后进行解调。

基准相位信号的调频副载波的表达式为

$$U(t) = U_m \cos \left(\Omega_s t + \frac{\Delta\Omega_s}{\Omega} \cos \Omega t \right) = U_m \cos (\Omega_s t + m_f \cos \Omega t) \quad (1)$$

再对载波调幅后的表达式为

$$U_R(t) = U_{Rm} [1 + m \cos (\Omega_s t + m_f \cos \Omega t)] \cos \omega t \quad (2)$$

产生的可变相位信号的表达式为

$$U_v(t) = U_{vm} \cos(\Omega t - \theta) \cos \omega t \quad (3)$$

解调思路包括载波解调、低通滤波和带通滤波、鉴频检波四步，前两步可以得到基可变相位信号，后两步可以得到基准相位信号。

1. 载波解调：较为简单，采用相干解调的办法即可，低通滤波的截止频率选取 1500Hz 即可，不影响后续调幅波的鉴频过程
2. 低通滤波：针对载波的相干解调后，通过低通滤波（实际上是带通滤波，25 ~ 35Hz）得到可变相位信号；
3. 带通滤波：通过带通滤波得到调频的基准相位信号，带通滤波的参数选取与调频系数有关，产生信号时设置的频率偏移大小决定了此时的带通滤波的上限频率和下限频率。
4. 鉴频检波：通过带通滤波得到调频的基准相位信号，鉴频后再经过带通滤波（25 ~ 40Hz）得到恒定相位信号。

解调后需要进行两信号的相位差计算。通过网上搜索得到的办法包括自乘、傅里叶变换、相关分析法等方法，其中效果最好的是相关分析法。此外通过观察解调后的完整波形，可以发现开始和结束部分均存在失真现象，为了计算精确，需要选取解调后信号的中间部分进行相位比较。

通过相关分析法计算基础角度，基于实际坐标情况分类讨论结果可以得到最终输出的测量 VOR 角度。

测量 VOR 方位角计算函数是基础要求 3 的核心所在，信号处理中采用参数和方法在课本基础上进行了一些调整，主要包括：

- 载波频率：甚高频变为 10kHz；
- 副载波频率：9960Hz 变为 996Hz，调频系数和带通滤波指标也随之而变；特别地，仿真过程中的调频指数为 1.6，带通滤波的上下限频率分别为 948Hz 和 1064Hz；
- 低通滤波变为带通滤波：在原有低通滤波之后进行截止频率为 25Hz 的高通滤波，避免直流分量对相位比较的影响。

此外，基准相位信号的调幅指数为 0.3，与课本保持一致。

主子函数和计算子函数的完整代码见章节四，其与设计的 fig 文件相结合即可得到下一节的结果。

3.3.2 任务结果与分析

以西北工业大学长安校区为例，飞机塔台位置在西工大工地南门，飞行任务中各个航路点分别为校外河流与公路交汇处、云天苑操场、长安校区图书馆、星天苑操场、航海楼和高冠峪等，基本涵盖了塔台的所有方向。

采用鼠标点击、滑块拖动或者可编辑文本输入的方式，依次经过上述航路点，得到的结果如图16所示。

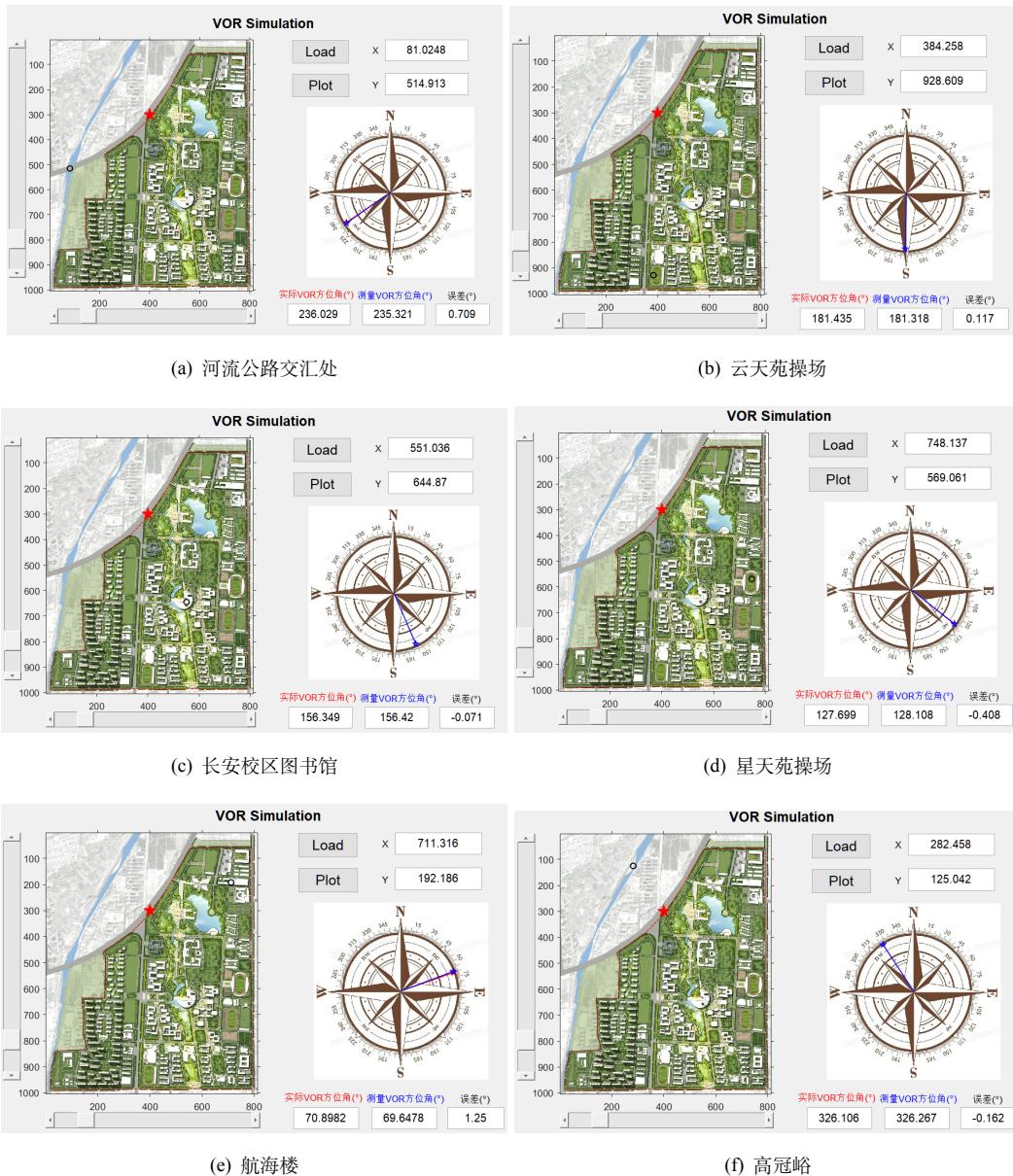


图 16 各航路点 VOR 方位角计算情况

以长安校区图书馆这一航路点为例，其坐标为(551,645)，编辑测量VOR方位角子函数，加入时域和频域图绘制代码，对信号的产生和处理过程逐步进行绘制，以验证结论的可靠性。

Step 1 产生的基准信号和可变信号的时域图和频域图，如图17和图18所示。

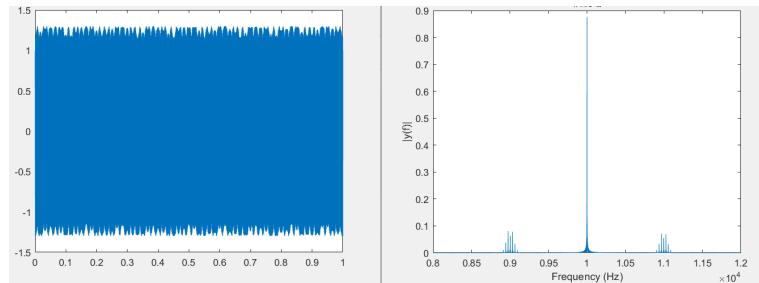


图 17 基准信号时域图和频域图

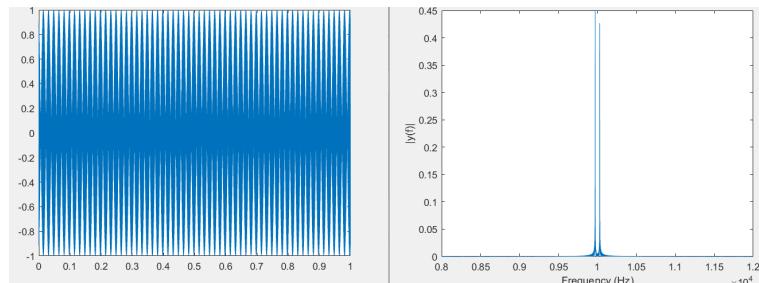


图 18 可变信号时域图和频域图

Step 2 对载波解调后的时域图和频域图，如图19所示

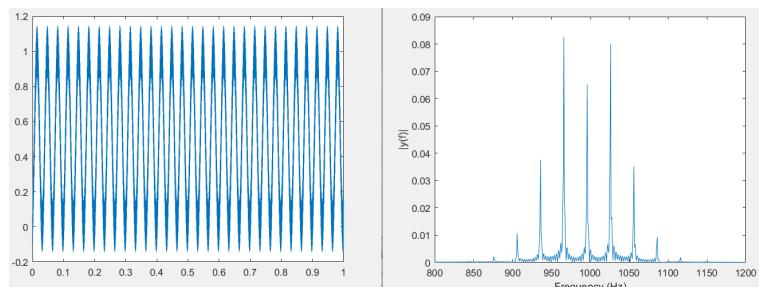


图 19 载波解调后的时域图和频域图

Step3 对 Step2 得到的信号进行低通(带通)滤波，得到的可变相位信号的时域图和频域图，如图20所示

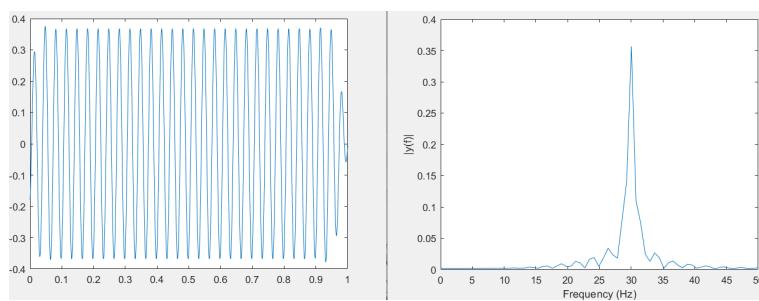


图 20 滤波后得到的可变相位信号的时域图和频域图

Step 4 对 Step2 得到的信号进行带通滤波得到信号的时域图和频域图，如图21所示

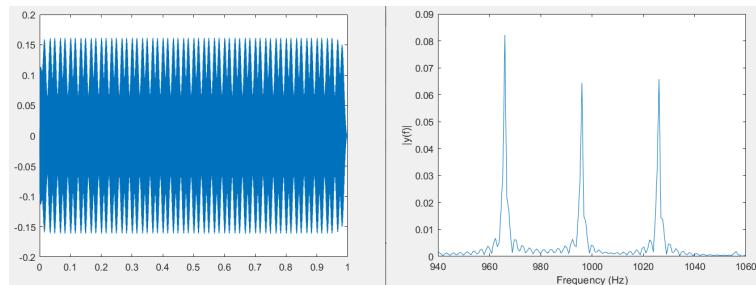


图 21 带通滤波得到信号的时域图和频域图

Step5 对上一步得到的信号进行差分后得到的信号的时域图和频域图，如图22所示

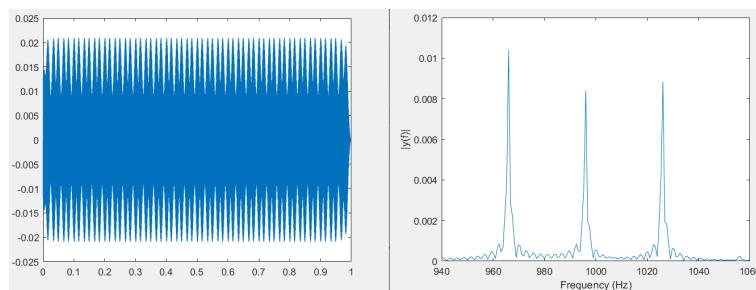


图 22 差分后得到的信号的时域图和频域图

Step6 对上一步得到的信号进行包络检波后得到的信号的时域图和频域图，如图23所示

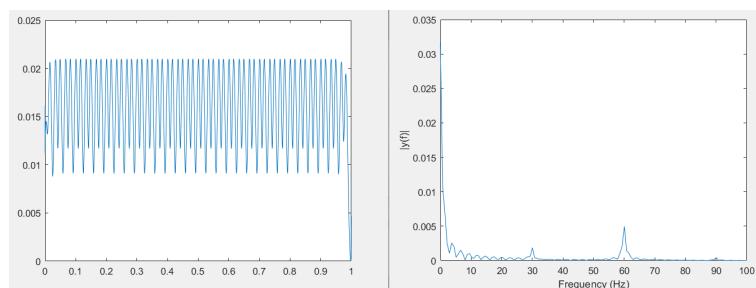


图 23 包络检波后得到的信号的时域图和频域图

Step7 对上一步得到的信号进行低通(带通滤波)放大，得到的基准相位信号的时域图和频域图，如图所示

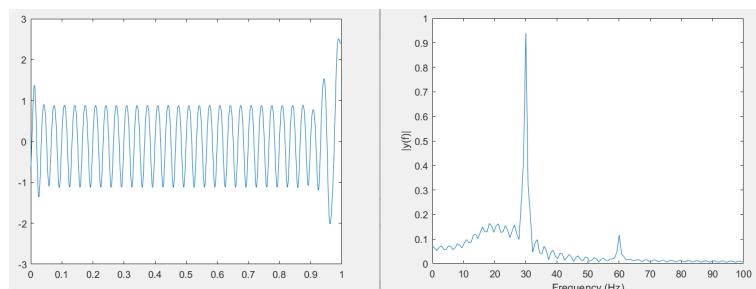


图 24 低通放大后得到的基准相位信号的时域图和频域图

本次任务的结论和分析如下：

- 只有信号产生和处理十分标准才能使得最终得到的误差角度很小。这需要不断对信号的各处理过程进行频域分析。
- 课本上的参数指标大多都是值得借鉴的，但部分信号处理方法在仿真上还需要结合实际调整。
- VOR 仿真得到的结果绝大多数在 1 度之内，十分准确。两个指针基本完全重合也体现了这一点。除了信号的处理到位外，还在于选择了科学的相位比较方法相关分析法。
- 子函数的封装设计具有层次性，也方便局部调试。
- 需要改进的地方在于没有对信号传输进行仿真，需要再加入噪声或者衰减，此外针对部分误差较大的区域（如接近 90 度的情况）没有进一步深入探索，可能原因和处理办法需要结合信号处理的过程进一步摸索。

3.4 基于 GUI 的 ILS 系统偏离测量仿真

3.4.1 需求分析与设计

对于基本要求 3，GUI 实现包括两部分，一是 *fig* 的绘制，二是主子函数设计，三是负责计算的子函数。

图形的绘制：新建 *fig* 制标题、画布、按钮和可编辑文本等，如图25所示。两个画布分别用来绘制示意图和指示器。按钮设计包括加载图片/初始化、绘点等功能，部分可编辑文本用来设置所处位置，最终结果体现在其他两个可编辑文本上，即 LOC 判决和 GS 判决。

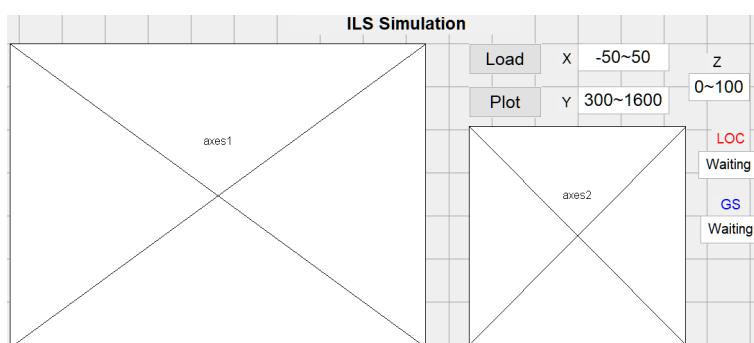


图 25 GUI 界面设计图

主子函数设计

- 参数初始化：包括默认绘点坐标为 (0,1400,76.1538) 等；

- 可编辑文本：设置三个关于飞机坐标的可编辑文本，做到输入转换；设置两个关于飞机航道偏离指示的可编辑文本；
- 加载图片/初始化：在两个图窗中分别加载降落示意图和航道偏离指示器，设置飞机位置 (0,1600,85) 并在图中标明；
- 绘点函数：获取初始化绘点位置或者手动输入的飞机坐标，在图中相应位置标注；之后调用计算子函数得到航道偏离指示字符串和 DDM，并做到指针显示和可编辑文本显示。此外，为该函数所绘制的所有图形分别定义全局变量，这些全局变量要和加载图片/初始化的全局变量一致，不仅可以做到每次调用自身前进行更新，还可以做到互相更新。

计算子函数设计

- 信号发射：设置采样频率、两信号张开角度等参数，输入参数为飞机相对于航向面或下滑面的角度；
- 信号接收：信号经过载波解调和滤波处理后即可得到 150Hz 和 90Hz 信号；
- 计算判决：对信号中间截取主要部分后进行傅里叶变换并计算幅度最值作为调幅指数；设定判决门限确定 CDI 指示；计算 DDM 并进行调整，使得输入大小为 10 度时 DDM 值为 0.155，与课本参数对应。

3.4.2 任务结果与分析

采用可编辑文本输入坐标的方式，依次经过部分位置，对示意图按照显示情况进行旋转，得到的结果如图26所示。

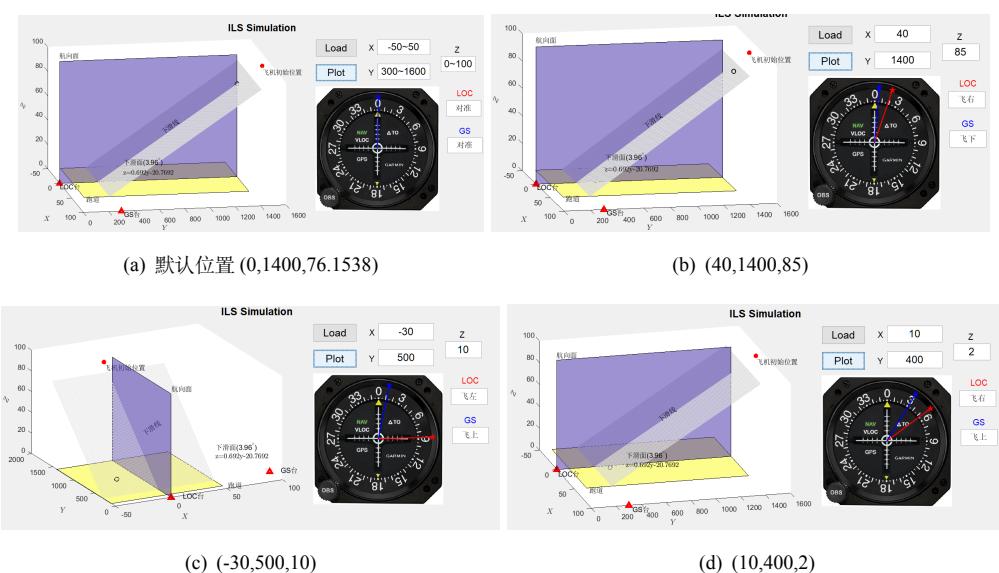


图 26 各位置坐标计算情况

以 LOC 为例, 设置输入角度为 10 度, 编辑测量子函数, 加入时域和频域图绘制代码, 对信号的产生和主要结果进行绘制, 以验证结论的可靠性。

信号产生: 150Hz 与 90Hz 的调制信号叠加后的发射信号时域图和频域图, 如图27所示。

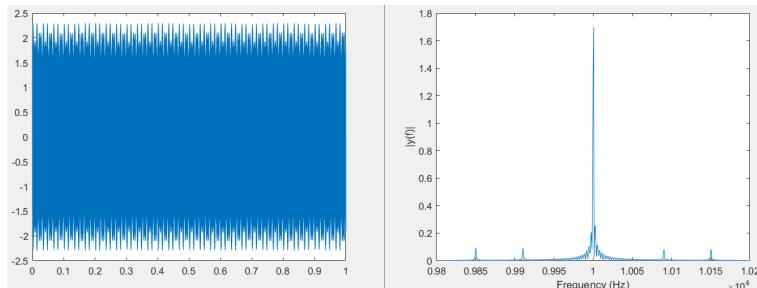


图 27 发射信号时域图和频域图

信号重建: 解调得到的 90Hz 信号和 150Hz 信号的时域图和频域图, 如图28和图29所示。

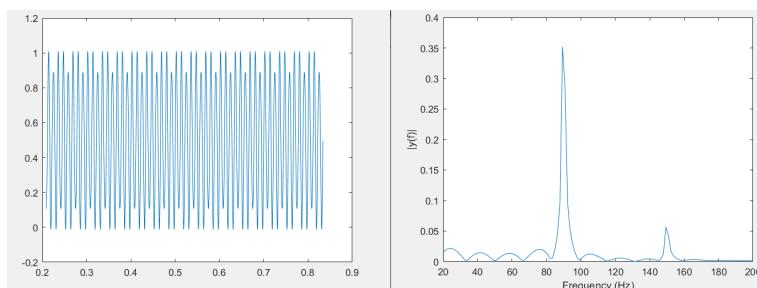


图 28 90Hz 信号时域图和频域图

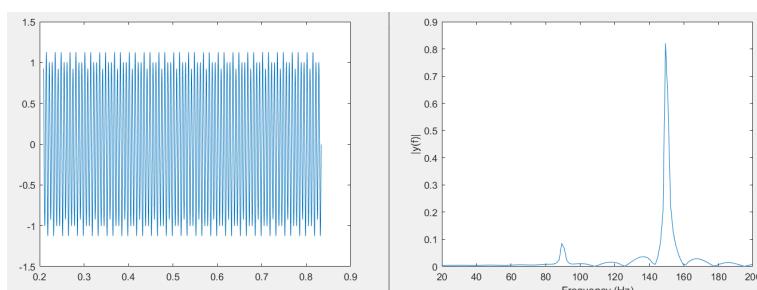


图 29 150Hz 信号时域图和频域图

本次任务的结论和分析如下:

- ILS 信号处理过程不如 VOR 复杂, 但部分参数, 如 $f(\theta)$ 课本没有, 需要自己调整;

- 当偏离角度很小时，计算函数只能给出“对准”指示，这需要合适的“噪声容限”（“判决门限”）设置；
- 子函数的封装设计具有层次性，也方便局部调试。
- 需要改进的地方在于，需要再加入噪声或者衰减，此外解调后出现的信号时域赋值并不稳定，可能原因是滤波设置参数不合适，处理办法需要结合信号处理的过程进一步摸索。

四、完整代码

这一部分的代码顺序如下：

1. 基本要求 2：基于 GUI 的 AM 通信系统仿真
2. 基本要求 3：基于 GUI 的 VOR 系统测向仿真-主子函数；
3. 基本要求 3：基于 GUI 的 VOR 系统测向仿真-实际 VOR 方位角计算子函数；
4. 基本要求 3：基于 GUI 的 VOR 系统测向仿真-测量 VOR 方位角计算子函数；
5. 基本要求 4：基于 GUI 的 ILS 系统偏离测量仿真-主子函数；
6. 基本要求 4：基于 GUI 的 ILS 系统偏离测量仿真-计算子函数；
7. 基本要求 4：基于 GUI 的 ILS 系统偏离测量仿真-平面绘制子函数。

基本要求 2：基于 GUI 的 AM 通信系统仿真

```
1 function varargout = SPEECH_AM(varargin)
2 gui_Singleton = 1;
3 gui_State = struct('gui_Name',     mfilename, ...
4                     'gui_Singleton',  gui_Singleton, ...
5                     'gui_OpeningFcn', @SPEECH_AM_OpeningFcn, ...
6                     'gui_OutputFcn',  @SPEECH_AM_OutputFcn, ...
7                     'gui_LayoutFcn', [], ...
8                     'gui_Callback',  []);
9 if nargin && ischar(varargin{1})
10    gui_State.gui_Callback = str2func(varargin{1});
11 end
12 if nargout
13    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
14 else
15    gui_mainfcn(gui_State, varargin{:});
16 end
17 function SPEECH_AM_OpeningFcn(hObject, eventdata, handles, varargin)
18 handles.output = hObject;
19 handles.recObj = audiorecorder(48000, 16, 1);
20 handles.recObj.TimerFcn={@RecDisplay,handles};
21 handles.recObj.TimerPeriod=0.25;
22 handles.carrier = 10e3;
23 handles.A_0    = 3 ;
```

```
24 handles.playselect = 0;
25 guidata(hObject, handles);
26 function varargout = SPEECH_AM_OutputFcn(hObject, eventdata, handles)
27 varargout{1} = handles.output;
28 function pushbutton1_Callback(hObject, eventdata, handles)
29 record(handles.recObj);
30 function pushbutton2_Callback(hObject, eventdata, handles)
31 stop(handles.recObj)
32 handles.playselect = 0;
33 guidata(hObject, handles);
34
35 function pushbutton4_Callback(hObject, eventdata, handles)
36 handles.myRecording = getaudiodata(handles.recObj);
37 t = (1:length(handles.myRecording))/handles.recObj.SampleRate;
38 assignin('base','t',t)
39 % 调制
40 fc = handles.carrier;
41 A_0 = handles.A_0 ;
42 x_am = (A_0+(handles.myRecording)).*cos(2*pi*fc*t);
43 plot(handles.axes1,t,x_am )
44 title(handles.axes1,'AM Time Domain Figure')
45 ylim(handles.axes1,[-0.5-handles.A_0 0.5+handles.A_0 ])
46 xlabel(handles.axes1,'Time(Seconds)')
47 ylabel(handles.axes1,'Amplitude')
48 drawnow;
49 assignin('base','x_am',x_am)
50 Fs = 48000;
51 L = length(x_am);
52 NFFT = 2^nextpow2(L); %确定FFT变换的长度
53 y = fft(x_am, NFFT)/L;
54 f = Fs/2*linspace(0,1,NFFT/2+1); %频率向量
55 plot(handles.axes2, f,2*abs(y(1:NFFT/2+1))); %绘制频域图像
56 title(handles.axes2,'AM Frequency Domain Figure')
57 xlim(handles.axes2,[handles.carrier - 4000 handles.carrier + 4000]);
58 xlabel(handles.axes2,'Frequency(Hz)'), ylabel(handles.axes2,'Amplitude')
59 drawnow;
60 function pushbutton5_Callback(hObject, eventdata, handles)
61 % 信号传输：加性高斯白噪声信道
62 snr = 30; %10
63 x_am = evalin('base','x_am') ;
64 t = evalin('base','t') ;
65 x_am_snr = awgn(x_am,snr);
66 assignin('base','x_am_snr',x_am_snr)
67 plot(handles.axes1,t,x_am_snr)
68 title(handles.axes1,'Transmitted Signal Time Domain Figure')
69 ylim(handles.axes1,[-0.5-handles.A_0 0.5+handles.A_0 ])
70 xlabel(handles.axes1,'Time(Seconds)'), ylabel(handles.axes1,'Amplitude')
71 drawnow;
72 Fs = 48000;
73 L = length(x_am_snr);
74 NFFT = 2^nextpow2(L); %确定FFT变换的长度
75 y = fft(x_am_snr, NFFT)/L;
76 f = Fs/2*linspace(0,1,NFFT/2+1); %频率向量
77 plot(handles.axes2, f,2*abs(y(1:NFFT/2+1))); %绘制频域图像
78 title(handles.axes2,'Transmitted Signal Frequency Domain Figure')
79 xlim(handles.axes2,[handles.carrier - 4000 handles.carrier + 4000])
80 xlabel(handles.axes2,'Frequency(Hz)'), ylabel(handles.axes2,'Amplitude')
```

```
79 drawnow;
80 function pushbutton6_Callback(hObject, eventdata, handles)
81 % 信号解调
82 x_am_snr = evalin('base','x_am_snr') ;
83 t = evalin('base','t') ;
84 fc = handles.carrier;           %载波频率
85 x_de_pre = x_am_snr.*cos(2*pi*fc*t);
86 x_de = 2.*lowpass(x_de_pre,8000,48000)- handles.A_0 ;
87 assignin('base', 'x_de',x_de)
88 plot(handles.axes1,t,x_de)
89 title (handles.axes1,'Demodulated Signal Time Domain Figure')
90 ylim(handles.axes1,[-0.5 0.5]), xlabel(handles.axes1,'Time(Seconds)'), ylabel(handles.axes1,'Amplitude')
91 drawnow;
92 Fs = 48000;
93 L = length(x_de);
94 NFFT = 2^nextpow2(L);          %确定FFT变换的长度
95 y = fft(x_de, NFFT)/L;
96 f = Fs/2*linspace(0,1,NFFT/2+1); %频率向量
97 plot(handles.axes2, 2*abs(y(1:NFFT/2+1))); %绘制频域图像
98 title (handles.axes2,'Demodulated Signal Frequency Domain Figure'), ylim(handles.axes2,[0 0.01])
99 xlim(handles.axes2,[0 8000]), xlabel(handles.axes2,'Frequency(Hz)'), ylabel(handles.axes2,'Amplitude')
100 drawnow;
101 handles.playselect = 1;
102 guidata(hObject, handles);
103 function pushbutton3_Callback(hObject, eventdata, handles)
104 if handles.playselect == 0
105     handles.myRecording = getaudiodata(handles.recObj);
106     handles.playObj = audioplayer(handles.myRecording,handles.recObj.SampleRate);
107     play(handles.playObj);
108 end
109 if handles.playselect == 1
110     x_de = evalin('base','x_de') ;
111     Fs = 48000;
112     sound(x_de,Fs);
113 end
114 guidata(hObject,handles);
115 function RecDisplay(hObject, eventdata,handles)
116 handles.myRecording = getaudiodata(handles.recObj);
117 plot(handles.axes1,(1:length(handles.myRecording))/handles.recObj.SampleRate,handles.myRecording)
118 title (handles.axes1,'Real-Time Time Domain Figure')
119 ylim(handles.axes1,[-0.5 0.5]), xlabel(handles.axes1,'Time(Seconds)'), ylabel(handles.axes1,'Amplitude')
120 drawnow;
121 Fs = 48000;
122 L = length(handles.myRecording);
123 NFFT = 2^nextpow2(L);          %确定FFT变换的长度
124 y = fft(handles.myRecording, NFFT)/L;
125 f = Fs/2*linspace(0,1,NFFT/2+1); %频率向量
126 plot(handles.axes2, 2*abs(y(1:NFFT/2+1))); %绘制频域图像
127 title (handles.axes2,'Real-Time Frequency Domain Figure')
128 ylim(handles.axes2,[0 0.01]);
129 xlim(handles.axes2,[0 8000]), xlabel(handles.axes2,'Frequency(Hz)'), ylabel(handles.axes2,'Amplitude')
130 drawnow;
131 function edit1_Callback(hObject, eventdata, handles)
132 handles.carrier = str2double(get(hObject,'String')).*1000 ;
133 guidata(hObject,handles)
134 function edit1_CreateFcn(hObject, eventdata, handles)
```

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

基本要求3：基于 GUI 的 VOR 系统测向仿真-主子函数

```
1 function varargout = VOR(varargin)
2 gui_Singleton = 1;
3 gui_State = struct('gui_Name',    mfilename, ...
4                     'gui_Singleton', gui_Singleton, ...
5                     'gui_OpeningFcn', @VOR_OpeningFcn, ...
6                     'gui_OutputFcn', @VOR_OutputFcn, ...
7                     'gui_LayoutFcn', [], ...
8                     'gui_Callback', []);
9 if nargin && ischar(varargin{1})
10     gui_State.gui_Callback = str2func(varargin{1});
11 end
12
13 if nargout
14     [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
15 else
16     gui_mainfcn(gui_State, varargin{:});
17 end
18
19 function VOR_OpeningFcn(hObject, eventdata, handles, varargin)
20 handles.output = hObject;
21 handles.X = 100;
22 handles.Y = 100;
23 handles.theta = 303.69;
24 guidata(hObject, handles);
25
26 function varargout = VOR_OutputFcn(hObject, eventdata, handles)
27 varargout{1} = handles.output;
28
29 function slider1_Callback(hObject, eventdata, handles)
30 handles.X = get(handles.slider1,'value');
31 set(handles.edit1,'string',handles.X);
32 guidata(hObject,handles)
33
34 function slider1_CreateFcn(hObject, eventdata, handles)
35 if isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
36     set(hObject,'BackgroundColor',[.9 .9 .9]);
37 end
38
39 function slider2_Callback(hObject, eventdata, handles)
40 handles.Y = get(handles.slider2,'value');
41 set(handles.edit2,'string',handles.Y);
42 guidata(hObject,handles)
43
44 function slider2_CreateFcn(hObject, eventdata, handles)
45 if isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
46     set(hObject,'BackgroundColor',[.9 .9 .9]);
47 end
```

```
49 function edit1_Callback(hObject, eventdata, handles)
50 handles.X = str2double(get(hObject,'String')) ;
51 guidata(hObject,handles)
52
53 function edit1_CreateFcn(hObject, eventdata, handles)
54 if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
55     set(hObject,'BackgroundColor','white');
56 end
57
58 function edit2_Callback(hObject, eventdata, handles)
59 handles.Y = str2double(get(hObject,'String')) ;
60 guidata(hObject,handles)
61
62 function edit2_CreateFcn(hObject, eventdata, handles)
63 if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
64     set(hObject,'BackgroundColor','white');
65 end
66
67 function pushbutton1_Callback(hObject, eventdata, handles)
68 axes(handles.axes1);
69 pic = imread('background_s.jpg');
70 handles.im = imshow(pic);
71 axis on
72 hold on
73 set(handles.im,'Pickableparts','none');
74 set(handles.axes1,'ButtonDownFcn',@mycallbackfcn);
75 plot(handles.axes1,400,300,'p','LineWidth',3,'Color','red')
76
77 pic2 = imread('Compass.jif');
78 axes(handles.axes2);
79 imagesc([-390 390], [390 -390], flip(pic2,1));
80 axis off
81 hold on
82
83 function mycallbackfcn(hObject,eventdata,handles)
84 global p1 p2 p3 p4 p5
85 if ~isempty(p1)
86     delete(p1);
87 end
88 if ~isempty(p2)
89     delete(p2);
90 end
91 if ~isempty(p3)
92     delete(p3);
93 end
94 if ~isempty(p4)
95     delete(p4);
96 end
97 if ~isempty(p5)
98     delete(p5);
99 end
100 handles = guidata(hObject);
101 pos = get(hObject,'Currentpoint');
102 set(handles.edit1,'string',pos(1,1));
103 set(handles.edit2,'string',pos(1,2));
104 p1 = plot(handles.axes1,pos(1,1),pos(1,2),'o','Color','black','LineWidth',1.2);
```

```
105 drawnow;
106 Theory = theta_js(pos(1,1), pos(1,2));
107 handles.theta = Theory;
108 set(handles.edit3,'string',Theory);
109 % 实际方位角绘制P2
110 Theory = 360 - Theory;
111 Theory = 90 + Theory ;
112 % plot([x1,x2],[y1,y2])
113 p2 =
    plot(handles.axes2,[0,250*cos(2*pi*Theory/360)],[0,-250*sin(2*pi*Theory/360)],'LineWidth',1,'Color','red');
114 p4 =
    plot(handles.axes2,250*cos(2*pi*Theory/360),-250*sin(2*pi*Theory/360),'p','LineWidth',1,'Color','red');
115 axes(handles.axes2);
116 % 测量方位角绘制P3
117 theta = handles.theta;
118 ph = VOR_signal(theta);
119 set(handles.edit4,'string',ph);
120 Fact = 360 - ph;
121 Fact = 90 + Fact ;
122 p3 = plot(handles.axes2,[0,250*cos(2*pi*Fact/360)],[0,-250*sin(2*pi*Fact/360)],'LineWidth',1,'Color','blue');
123 p5 = plot(handles.axes2,250*cos(2*pi*Fact/360),-250*sin(2*pi*Fact/360),'p','LineWidth',1,'Color','blue');
124 % 误差
125 error = roundn(handles.theta - ph,-3) ;
126 set(handles.edit5,'string',error);
127 guidata(hObject,handles);
128 drawnow;
129
130 function pushbutton2_Callback(hObject, eventdata, handles)
131 global p1 p2 p3 p4 p5
132 if ~isempty(p1)
133     delete(p1);
134 end
135 if ~isempty(p2)
136     delete(p2);
137 end
138 if ~isempty(p3)
139     delete(p3);
140 end
141 if ~isempty(p4)
142     delete(p4);
143 end
144 if ~isempty(p5)
145     delete(p5);
146 end
147 % 实际
148 p1 = plot(handles.axes1,handles.X,handles.Y,'o','Color','black','LineWidth',1.2);
149 Theory = theta_js(handles.X, handles.Y);
150 set(handles.edit3,'string',Theory);
151 handles.theta = Theory;
152 Theory = 360 - Theory;
153 Theory = 90 + Theory ;
154 p2 =
    plot(handles.axes2,[0,250*cos(2*pi*Theory/360)],[0,-250*sin(2*pi*Theory/360)],'LineWidth',1,'Color','red');
155 p4 =
    plot(handles.axes2,250*cos(2*pi*Theory/360),-250*sin(2*pi*Theory/360),'p','LineWidth',1,'Color','red');
156 axes(handles.axes2);
```

```
157 % 测量
158 theta = handles.theta;
159 ph = VOR_signal(theta);
160 set(handles.edit4,'string',ph);
161 Fact = 360 - ph;
162 Fact = 90 + Fact ;
163 p3 = plot(handles.axes2,[0,250*cos(2*pi*Fact/360)],[0,-250*sin(2*pi*Fact/360)],'LineWidth',1,'Color','blue');
164 p5 = plot(handles.axes2,250*cos(2*pi*Fact/360),-250*sin(2*pi*Fact/360),'p','LineWidth',1,'Color','blue');
165 % 误差
166 error = roundn(handles.theta - ph,-3) ;
167 set(handles.edit5,'string',error);
168
169 guidata(hObject,handles);
170 drawnow;
171
172 function edit3_Callback(hObject, eventdata, handles)
173
174 function edit3_CreateFcn(hObject, eventdata, handles)
175
176 if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
177     set(hObject,'BackgroundColor','white');
178 end
179
180 function edit4_Callback(hObject, eventdata, handles)
181
182 % ---- Executes during object creation, after setting all properties.
183 function edit4_CreateFcn(hObject, eventdata, handles)
184
185 if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
186     set(hObject,'BackgroundColor','white');
187 end
188
189 function edit5_Callback(hObject, eventdata, handles)
190
191 % ---- Executes during object creation, after setting all properties.
192 function edit5_CreateFcn(hObject, eventdata, handles)
193 if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
194     set(hObject,'BackgroundColor','white');
195 end
```

基本要求3：基于GUI的VOR系统测向仿真-实际VOR方位角计算子函数

```
1 function Theory = theta_js(X,Y)
2 A = [0,1];
3 B = [400 - X , 300 - Y ];
4 Theory = acos(dot(A,B)/(norm(A)*norm(B))) * (180/pi);
5 if B(1) > 0
6     Theory = 360 - Theory;
7 elseif (B(1) == 0)&&(B(2)<0)
8     Theory = 180;
9 elseif (B(1) == 0)&&(B(2)>=0)
10    Theory = 0;
11 end
12 end
```

基本要求3：基于GUI的VOR系统测向仿真-测量VOR方位角计算子函数

```
1 function ph = VOR_signal(theta)
2 t = linspace(0,1,48000);
3 w = 1e4;
4 U_base = cos(2*pi*996*t + 48/30*cos(2*pi*30*t));
5 U_base = (1 + 0.3*U_base).* cos(2*pi*w*t);
6 U_change = cos(2*pi*30*t - pi*theta/180).*cos(2*pi*w*t);
7 U = U_base + U_change;
8 U_filter = U.*cos(2*pi*w*t);
9 [B,A] = butter(4,1500/(48400/2),'low');
10 U_filter = filtfilt (B,A,U_filter); % 超外差
11 [B,A] = butter(4,35/(48400/2),'low');
12 U_30_change = filtfilt(B,A,U_filter);
13 [B,A] = butter(4,20/(48400/2),'high');
14 U_30_change = filtfilt(B,A,U_30_change);% 可变相位
15 [B,A] = butter(4,[948/(48400/2) 1044/(48400/2)]);
16 U_30_base = filtfilt(B,A,U_filter); %带通滤波
17 U_30_base =[0 diff(U_30_base)];
18 U_30_base = abs(hilbert(U_30_base));
19 [B,A] = butter(4,20/(48400/2),'high');
20 U_30_base = filtfilt(B,A,U_30_base);
21 [B,A] = butter(4,40/(48400/2),'low');
22 U_30_base = 666*filtfilt(B,A,U_30_base);% 恒定相位
23 t = t(9600:38400);
24 U_30_base = U_30_base(9600:38400);
25 U_30_change = U_30_change(9600:38400);
26 % 计算相位差
27 N = length(t);
28 IX = sum(U_30_base.*U_30_base)/N ;
29 IY = sum(U_30_change.*U_30_change)/N;
30 IXY = sum(U_30_change.*U_30_base)/N;
31 c = 180*acos(2*IXY/(4*IX*IY)^0.5)/pi ;
32 if theta < 90
33     c = 90 - c;
34 elseif theta <= 180
35     c = c + 90 ;
36 elseif theta <= 270
37     c = c + 90 ;
38 elseif theta <= 360
39     c = 450- c ;
40 end
41 ph = c ;
42 end
43 function [ ] = DrawFFT( x, Fs )
44 L = length(x);
45 NFFT = 2^nextpow2(L);          %确定FFT变换的长度
46 y = fft(x, NFFT)/L;
47 f = Fs/2*linspace(0,1,NFFT/2+1); %频率向量
48 figure(2)
49 plot(f, 2*abs(y(1:NFFT/2+1))); %绘制频域图像
50 xlim([0 100])
51 xlabel('Frequency (Hz)');
52 ylabel('|y(f)|');
53 end
```

基本要求 4：基于 GUI 的 ILS 系统偏离测量仿真-主子函数

```
1 function varargout = ILS(varargin)
2 gui_Singleton = 1;
3 gui_State = struct('gui_Name',    mfilename, ...
4                     'gui_Singleton', gui_Singleton, ...
5                     'gui_OpeningFcn', @ILS_OpeningFcn, ...
6                     'gui_OutputFcn', @ILS_OutputFcn, ...
7                     'gui_LayoutFcn', [] , ...
8                     'gui_Callback', []);
9 if nargin && ischar(varargin{1})
10    gui_State.gui_Callback = str2func(varargin{1});
11 end
12
13 if nargout
14    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
15 else
16    gui_mainfcn(gui_State, varargin{:});
17 end
18 function ILS_OpeningFcn(hObject, eventdata, handles, varargin)
19 handles.output = hObject;
20 handles.X = 0;
21 handles.Y = 1400;
22 %a=0;b=0.9/13;c=-300*b;
23 handles.Z = handles.Y*0.9/13-300*0.9/13;
24 guidata(hObject, handles);
25 function varargout = ILS_OutputFcn(hObject, eventdata, handles)
26 varargout{1} = handles.output;
27 function pushbutton1_Callback(hObject, eventdata, handles)
28 global p1
29 if ~isempty(p1)
30    delete(p1);
31 end
32 pic2 = imread('HSI.png');
33 axes(handles.axes2);
34 imagesc([-300 300], [300 -300], flip(pic2,1));
35 axis off
36 hold on
37 axes(handles.axes1);
38 color =[0;0;0;0;0;0];
39 plot_cuboid([0,0,0],[0,1400,100],color);
40 color =[1;1;1;1;1;1];
41 plot_cuboid([50,100,0],[-50,1400,0],color);
42 text(0,-5,105,'航向面');
43 text(55,105,0,'跑道');
44 text(10,-15,0,'LOC台');
45 text(110,300,0,'GS台');
46 text(0,1600,85,'飞机初始位置');
47 plot(0,0,'Marker','^','LineWidth',3,'Color','r');
48 plot(100,300,'Marker','^','LineWidth',3,'Color','r');
49 plot3(0,1600,90,'Marker','*','LineWidth',3,'Color','r');
50 a=0;b=0.9/13;c=-300*b;%下画面平面方程为z=ax+by+c。
51 x=linspace(-50,50,100);
52 y=linspace(300,1500,100);
53 [x,y]=meshgrid(x,y);
54 z=a*x+b*y+c;
```

```
55 plot3(x,y,z,'Color',[0.86,0.86,0.86]);
56 text(5,800,40,'下滑线','rotation',35);
57 text(55,400,20,'z=0.692y-20.7692','interpreter','latex');
58 text(55,400,30,'下滑面(3.96^{\circ});\atan(0.9/13)*180/3.1415 3.9604
59 txt = xlabel('$X$');
60 set(txt, 'Interpreter', 'latex');
61 txt = ylabel('$Y$');
62 set(txt, 'Interpreter', 'latex');
63 txt = zlabel('$Z$');
64 set(txt,'rotation',60, 'Interpreter', 'latex');
65 hold on
66 axis on
67 rotate3d on
68 function pushbutton2_Callback(hObject, eventdata, handles)
69 global p1 p2 p3 p4 p5
70 if ~isempty(p1)
71     delete(p1);
72 end
73 if ~isempty(p2)
74     delete(p2);
75 end
76 if ~isempty(p3)
77     delete(p3);
78 end
79 if ~isempty(p4)
80     delete(p4);
81 end
82 if ~isempty(p5)
83     delete(p5);
84 end
85 axes(handles.axes1);
86 p1 = plot3(handles.axes1,handles.X ,handles.Y ,handles.Z,'o','Color','black','LineWidth',1.2);
87 drawnow
88 rotate3d on
89 theta = atan((handles.X)/(handles.Y-300));
90 [CDI,DDM] = LOS_signal(theta);
91 set(handles.edit4,'string',CDI);
92 DDM = 100/0.155*DDM;
93 Fact = 360 - DDM;
94 Fact = 90 + Fact ;
95 p2 = plot(handles.axes2,[0,250*cos(2*pi*Fact/360)],[0,-250*sin(2*pi*Fact/360)],'LineWidth',1.5,'Color','r');
96 p3 = plot(handles.axes2,250*cos(2*pi*Fact/360),-250*sin(2*pi*Fact/360),'p','LineWidth',1.5,'Color','r');
97 drawnow
98 theta = atan((handles.Z)/(handles.Y-300));
99 [CDI,DDM] = GS_signal(theta-3.96*pi/180);
100 set(handles.edit5,'string',CDI);
101 DDM = 100/0.155*DDM;
102 Fact = 360 - DDM;
103 Fact = 90 + Fact ;
104 p4 = plot(handles.axes2,[0,250*cos(2*pi*Fact/360)],[0,-250*sin(2*pi*Fact/360)],'LineWidth',1.5,'Color','b');
105 p5 = plot(handles.axes2,250*cos(2*pi*Fact/360),-250*sin(2*pi*Fact/360),'p','LineWidth',1.5,'Color','b');
106 drawnow
107 function edit1_Callback(hObject, eventdata, handles)
108 handles.X = str2double(get(hObject,'String')) ;
109 guidata(hObject,handles)
110 function edit1_CreateFcn(hObject, eventdata, handles)
```

```
111 if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
112     set(hObject,'BackgroundColor','white');
113 end
114 function edit2_Callback(hObject, eventdata, handles)
115 handles.Y = str2double(get(hObject,'String')) ;
116 guidata(hObject,handles);
117 function edit2_CreateFcn(hObject, eventdata, handles)
118 if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
119     set(hObject,'BackgroundColor','white');
120 end
121 function edit3_Callback(hObject, eventdata, handles)
122 handles.Z = str2double(get(hObject,'String')) ;
123 guidata(hObject,handles);
124 function edit3_CreateFcn(hObject, eventdata, handles)
125 if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
126     set(hObject,'BackgroundColor','white');
127 end
128 function edit4_Callback(hObject, eventdata, handles)
129 function edit4_CreateFcn(hObject, eventdata, handles)
130 if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
131     set(hObject,'BackgroundColor','white');
132 end
133 function edit5_Callback(hObject, eventdata, handles)
134 function edit5_CreateFcn(hObject, eventdata, handles)
135 if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
136     set(hObject,'BackgroundColor','white');
137 end
```

基本要求 4：基于 GUI 的 ILS 系统偏离测量仿真-计算子函数

```
1 function [CDI,DDM] = LOS_signal(theta)
2 t = linspace(0,1,48000);%信号发射
3 w = 1e4;
4 f1 = power(cos((theta-5)*pi/180),8);
5 f2 = power(cos((theta+5)*pi/180),8);
6 u1 = f1*(1+0.2*sin(2*pi*90*t)).*sin(2*pi*w*t);
7 u2 = f2*(1+0.2*sin(2*pi*150*t)).*sin(2*pi*w*t);
8 u = u1 + u2;
9 u = u.* sin(2*pi*w*t);%信号接收
10 [B,A] = butter(4,1000/(48000/2),'low');
11 u_filter = filtfilt (B,A,u); % 超外差
12 u_filter = u_filter - (f1+f2);
13 [B,A] = butter(4,120/(48000/2),'low');
14 u1 = 5.26* filtfilt (B,A,u_filter)+5.6; % 90
15 [B,A] = butter(4,120/(48000/2),'high');
16 u2 = 12.5* filtfilt (B,A,u_filter); % 150
17 u1 = u1(10000:40000);
18 u2 = u2(10000:40000);
19 u1 = max(abs(fftshift(fft(u1))))/48000-0.0170;%计算
20 u2 = max(abs(fftshift(fft(u2))))/48000;
21 if(abs(u1-u2)>=2.942e-5) %判决 2.941822266788741e-05
22     if(u1-u2>=2.942e-5)
23         CDI = "飞右";
24     elseif (u2-u1>=2.942e-5)
```

```
25     CDI = "飞左";
26     end
27 else
28     CDI = "对准";
29 end
30 umax = max(u1,u2);
31 DDM = u1/umax-u2/umax;
32 if DDM > 0
33     DDM = (0.155/0.002573)*DDM;
34 elseif DDM <= 0
35     DDM = (0.155/0.001677)*DDM;
36 end
37 DDM = abs(DDM);
38 end
39 function [ ] = DrawFFT( x, Fs )
40 L = length(x);
41 NFFT = 2^nextpow2(L);          %确定FFT变换的长度
42 y = fft(x, NFFT)/L;
43 f = Fs/2*linspace(0,1,NFFT/2+1); %频率向量
44 figure()
45 plot(f, 2*abs(y(1:NFFT/2+1))); %绘制频域图像
46 xlim([20 200])
47 xlabel('Frequency (Hz)');
48 ylabel('|y(f)|');
49 end
```

基本要求 4：基于 GUI 的 ILS 系统偏离测量仿真-平面绘制子函数

```
1 function plot_cuboid(start_point,final_point,color)
2 vertexIndex=[0 0 0;0 0 1;0 1 0;0 1 1;1 0 0;1 0 1;1 1 0;1 1 1];
3 cuboidSize=final_point-start_point;           %方向向量
4 vertex= repmat(start_point,8,1)+vertexIndex.* repmat(cuboidSize,8,1);
5 facet=[1 2 4 3;1 2 6 5;1 3 7 5;2 4 8 6;3 4 8 7;5 6 8 7];
6 patch('Vertices',vertex,'Faces',facet,'FaceVertexCData',color,'FaceColor','interp','FaceAlpha',0.5);
7 view(80,20);
8 hold on
9 axis on
10 end
```

Part II

语音信号采集处理设计

五、基本要求

1. 用计算机自带声卡采集语音信号；
2. 显示语音信号的时域图和频域图；
3. 加入数字滤波器提高语音信号质量；
4. 对语音信号进行时频域分析（语谱图）。

六、基本原理

基于基本要求，分别对原理进行阐述。

1. 采集语音信号实际上就是打开电脑的麦克风，在 *matlab* 中可以用 *audioDeviceReader*、*dsp* 工具箱的 *audioRecorder*、*audioRecorder* 等函数实现。
三个函数的共同点是参数设置，可以规定采样率、声道和间隔时间等。
三个函数的不同点如下：
 - *audioDeviceReader* 的单次采样时间有限，不能超过 1min，因此需要循环调用该函数，即不断地开启和关闭麦克风才能完成较长时间地语言信号搜集。
 - *dsp* 工具箱的 *audioRecorder* 函数可以使麦克风保持一直打开地状态，有利于采集信号和处理信号地同时进行，但缺点在于 2019 版本后地 *matlab* 函数已经将其删除，并建议改用 *audioRecorder* 函数。
 - *audioRecorder* 函数单次执行时间远超过 *audioDeviceReader* 函数，但执行期间一直占用编译器，无法做到采集信号和处理信号地同时进行。
2. 显示语音信号的时域图只需要利用 *plot(t,x)* 实现即可，频域图绘制可以在老师例程之上进行优化¹，也可以自己编写并封装 DrawFFT 函数调用之；
3. 在数字滤波器设计上，老师给出了调用方法，由于是选用试验，因此这里可不对各类滤波器进行比较。
以巴特沃斯滤波器为例，其特性是可以使通带内的幅度响应最大限度地平坦，但会损失截止频率处的下降斜度，使幅度响应衰减较慢。*matlab* 官方网站给

¹例程存在一些小问题，如对于变换后的幅值没有归一化等

出了低通滤波器的应用方法^[6]，*Filtfilt* 函数可以矫正滤波后信号的相位，弥补 *butter* 函数的不足。

截止频率 f_c 、采样频率 f_s 和归一化截止频率 W_c 的关系是

$$W_c = \frac{f_c}{\frac{f_s}{2}} \quad (4)$$

巴特沃斯滤波器设计应用例程

```

1 Wc = fc/(fs/2)
2 [b, a] = butter(4, Wc,'low');
3 FilterData = filtfilt (b,a,ReceiveData);

```

4. 对语音信号进行时频域分析（语谱图）。

语谱图是对信号语音短时傅里叶变换后时域、频域和能量结合的可视化表达，类型有二维图也有三维图。*matlab* 中 *spectrogram* 给出了绘制的办法。参数包括基于帧长的窗口函数、帧移、DFT 的点数和采样频率等。

为了统一起见，任务实施中采用的参数为 256 帧长、汉明窗、128 帧移。

七、任务实施

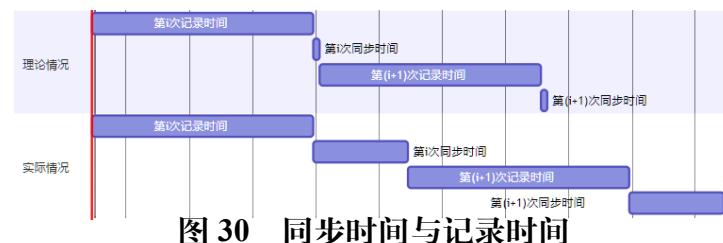
按照具体实施情况，这里将任务实施流程分成三步，实际上是探索问题、逐步优化的三种思路。

7.1 思路分析

首先对三种思路进行过程分析。

- 思路一：基于间断分析的域分析；

这一思路主要是对例程的优化，例程思路是循环调用 *audioDeviceReader* 函数，麦克风每次从打开到关闭的时间只有 0.25s。但信号处理程序（包括打开和关闭麦克风的时间）的单次执行时间（同步时间）对于这 0.25s 是不可忽略的，也就是说，语言信号必然在两次麦克风记录中存在明显丢失信号的现象，如图30所示。



麦克风的间断调用导致明显的信号内容丢失，为了缓解这一现象，容易想到可以增加麦克风单次使用时间，直到可以忽略信号处理的时间。但矛盾之处在于，随着采用信号长度的增加，信号处理时间也会上升，并且 *audioDeviceReader* 函数的单次打开时间是有限的。

这一思路的所谓间断，指的是在录音记录之后进行域分析等信号处理操作，老师在演示例程的时候只采用了较为简单的 FFT 运算，因此没有明显感觉到“卡顿”现象，但如果把滤波、语谱图等程序也加入，则问题就会变得较为严重。

解决这一问题的最好方法是希望麦克风能够一直打开，也就是做到记录信号和处理信号同时进行 dsp 的 *audioRecorder* 函数可以考虑，但在较高版本的 *matlab* 上已经用 *audioRecorder* 代替，但这一函数在记录过程中也是如 *audioDeviceReader* 一般占用编译器资源，同时执行之后的信号处理程序。

- 思路二：基于整体分析的域分析；

在思路一中，当切实想要增加到较长时间的录音记录时，不如直接记录所有信号数据再进行处理。此外，基本要求涉及到“语音信号质量”的提高，采用播放滤波后判断的方法才能确定究竟语言信号有无提高。

综上，思路二的流程为完整录音后进行播放和域分析，之后进行滤波，滤波后再进行播放和域分析，对两次播放和域分析的结果进行对比。

- 思路三：基于 GUI 的域分析；

上述两种思路各有优势，思路 1 的优势在于录音记录的“同时”进行域分析，思路二的优势在于录音一直打开事实上“减少”了同步时间。这里考虑对这两种优势进行结合实现，也就是保证录音一直打开，在录音记录的同时进行信号处理。

声卡打开依然采用 *audiorecorder* 函数，并保持其一直运行，这样也就保证了录音一直打开。为了避免其单一地占用编译器资源，加入定时器同时执行耗时较少的频域分析程序。在完整录音后执行滤波、结果比对以及语谱图分析等耗时较长的程序。为了更好的实现这一程序，在 *matlab* 中采用 GUI 的方法进行设计。

7.2 基于间断分析的域分析

audioDeviceReader 的初始化参数分别为：采样频率 48000，间隔时间 1s，执行时间 20s，录音器参数设置为一个声道。

单次循环的步骤包括

1. 录音机打开，数据存储；
2. 录音数据快速傅里叶变换，并绘制时域图和频域图；
3. 录音数据滤波处理，并绘制时域图和频域图；
4. 录音数据绘制二维和三维语谱图。

完整代码如章节所示。以某次程序运行为例，设置归一化截止频率为 0.1，第 2s 的运行结果图如图31所示。

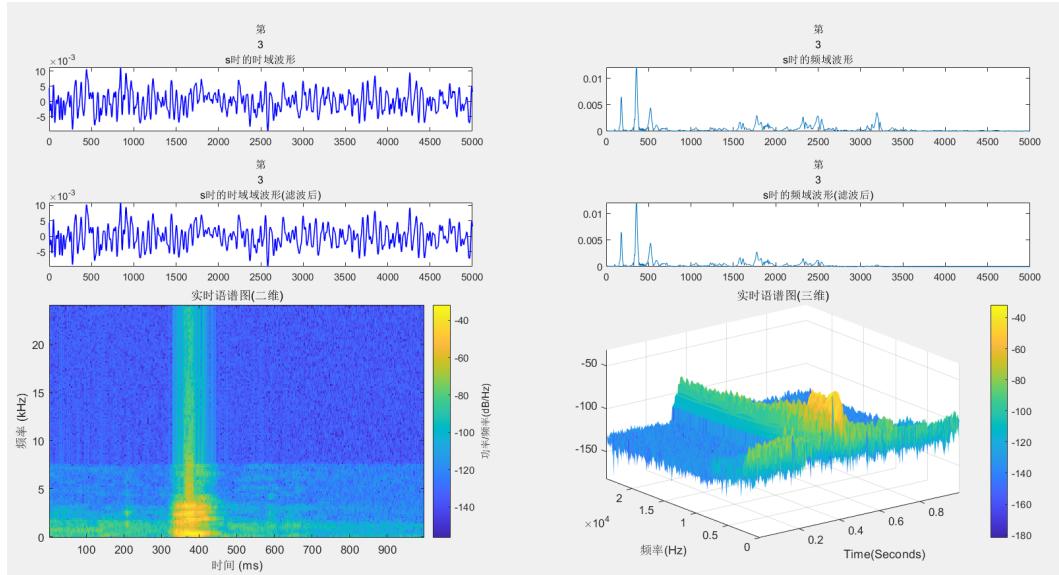


图 31 第 2s 的运行结果图

从图中可以发现，

- 语音信号的频率成分主要以 400Hz 为主，时域图混杂不堪。
- 滤波后的时域图没有明显改善，但频域图高频部分均消失。
- 从原始信号的语谱图可以看出 350-390Hz 频段的语音能量越强^[7]。此外可以发现低频部分的横条纹比较直，而高频部分的条纹变弯。

7.3 基于整体分析的域分析

audiorecorder 函数设置录音时间为 5s，其他初始化参数与思路一保持一致。对绘制语谱图的函数封装表达，完整代码如章节八、所示。

思路二与思路一的步骤相同，这里不再赘述。以某次程序运行的结果为例，设置归一化截止频率为 0.1，运行结果如图32(a)~33(d)所示。其中前四张为原始数据的域分析图，后四张为滤波后的域分析图。

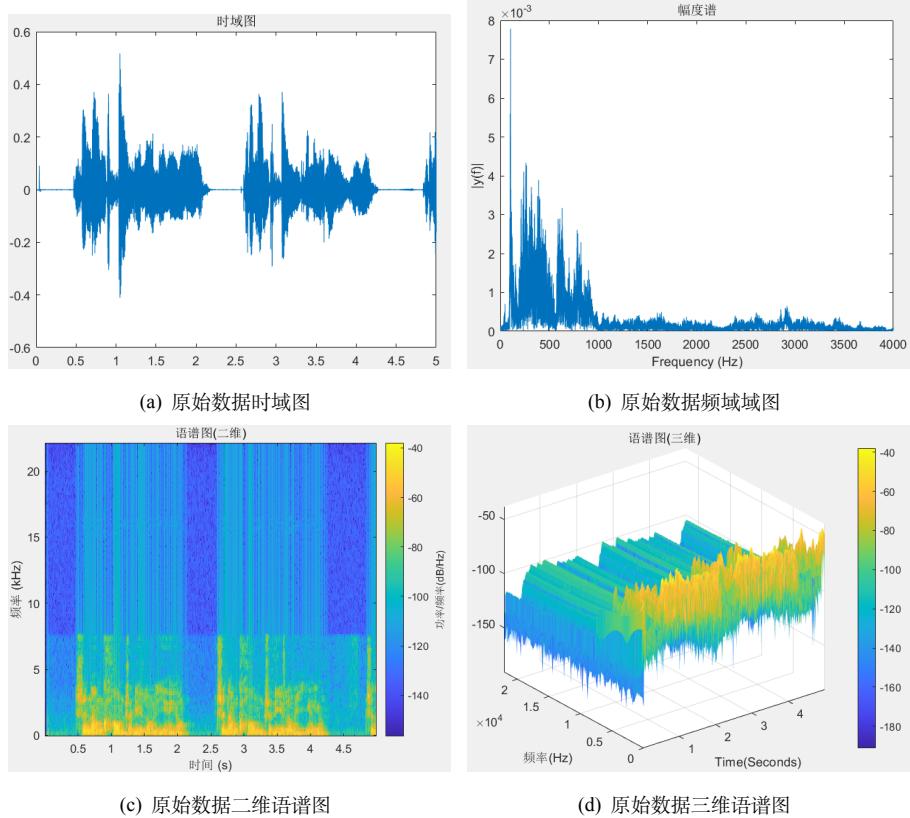


图 32 原始数据域分析图

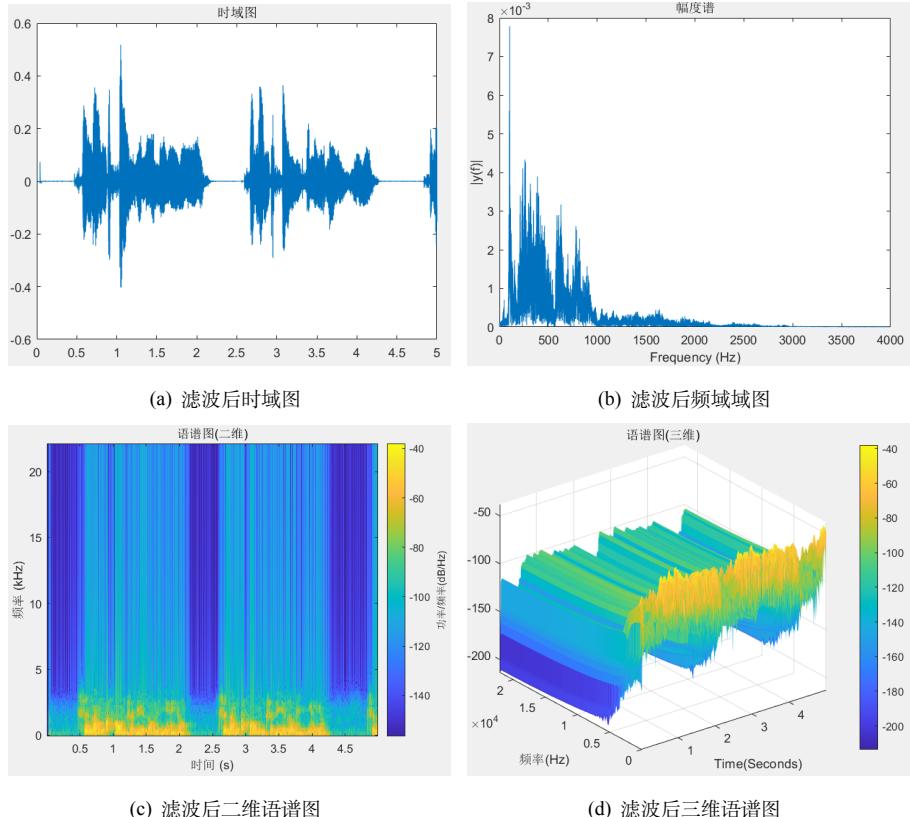


图 33 滤波后域分析图

滤波前后对比可以发现，

- 时域图变化较小；
- 频域图高频部分消失；
- 从语谱图上看，二维语谱图的横条减少，竖条拉长，说明滤波降低了信号的频率分辨率。

7.4 基于 GUI 的域分析

7.4.1 需求分析与设计

思路三是课程设计任务的主要部分，这一部分的实现过程包括两部分，一是 *fig* 图形的绘制，二是子函数设计。

图形的绘制：新建 *fig*，绘制标题、画布、按钮和可编辑文本^[8]等，如图34所示。两个画布分别用来绘制时域图和实时频域图。按钮设计包括开始、停止、播放、保存、滤波和语谱图绘制等功能，可编辑文本用来设置截止频率和阶数等滤波参数。

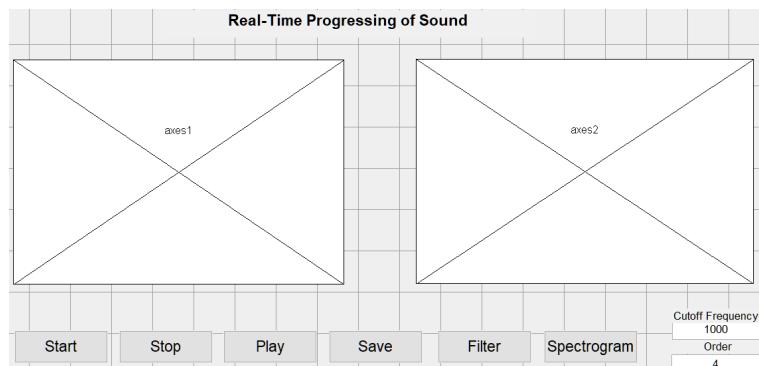


图 34 GUI 界面设计图

子函数设计：

- 录音器初始化：采用 *audiorecorder* 函数，初始化参数与前两个思路保持一致。设置定时器和其他全局变量。
- 获取原始录音数据；
- 开始按钮：用 *record* 函数即可实现；
- 结束按钮：用 *stop* 函数即可实现；
- 播放按钮：用 *audioplayer* 函数获取数据，*play* 函数播放；*guidata* 函数更新数据；设置播放选项，用于识别播放信号为调制信号和解调信号；

- 保存按钮：用 `uiputfile` 函数确定路径，`audiowrite` 函数导出文件；
- 实时显示：`getaudiodata` 获取数据，时域图和频域图绘制代码与思路一相同，注意需要用 `drawnow` 更新。绘图前需确定画布句柄；
- 可编辑文本设置：用 `get` 和 `str2double` 等函数实现输入转换，并用 `guidata` 函数更新数据；
- 滤波处理和显示：获取更新后或者初始化的滤波器参数，滤波处理和绘制图形与思路一相同，完成绘图任务。注意使用 `drawnow` 更新图片，做到对同一各信号采用不同滤波处理和播放，方便比较；
- 语谱图显示：获取输入数据，绘图过程与思路一一致。

7.4.2 任务结果与分析

以某首歌播放作为输入，完整的时域图和频域图如图35所示。

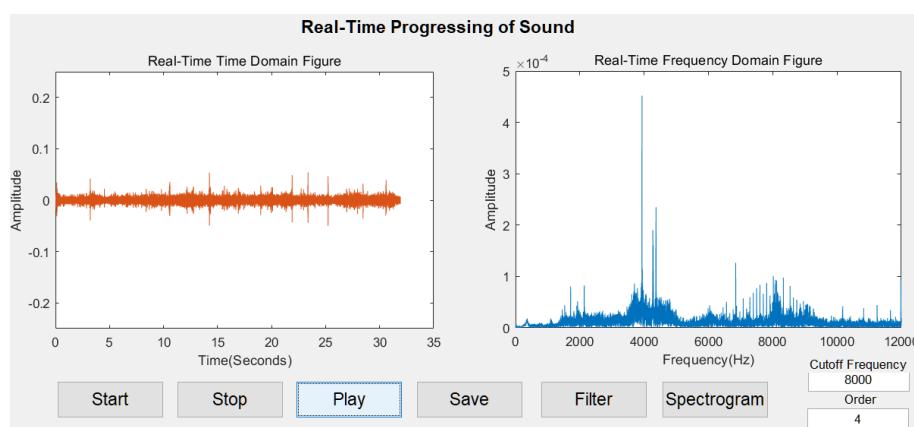


图 35 原始信号的时域图和频域图

信号峰值在 4000Hz 出现，以 8000Hz 作为截止频率，滤波后播放，得到的时域图和频域图如图36所示。

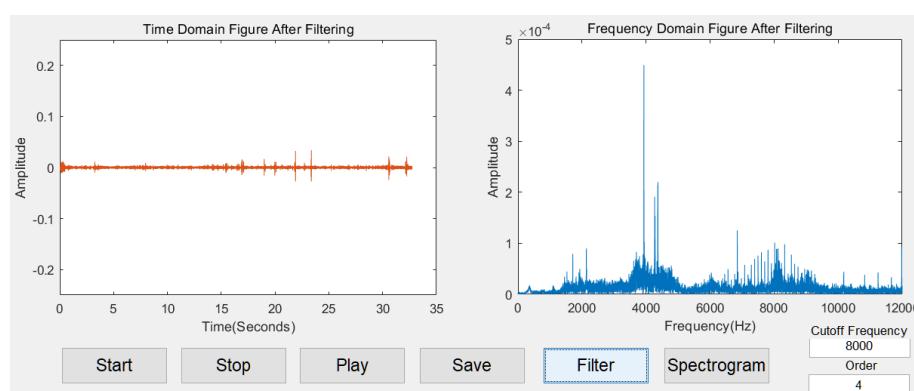


图 36 滤波后的时域图和频域图 (8000Hz,4order)

以 10000Hz 作为截止频率，滤波后播放，得到的时域图和频域图如图37所示。

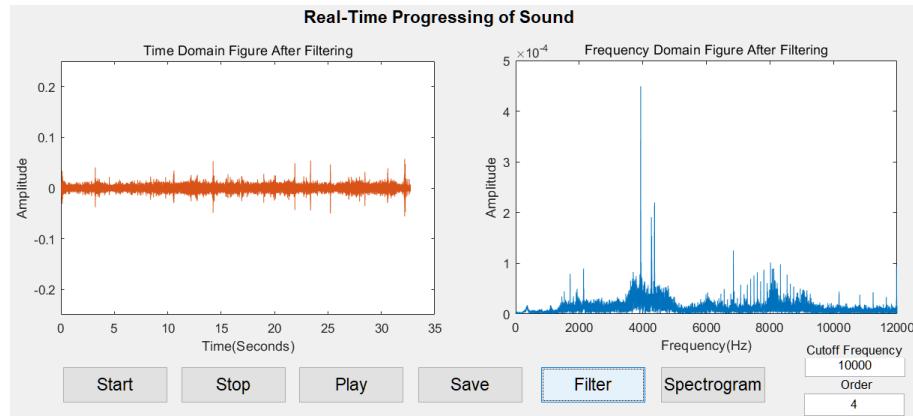


图 37 滤波后的时域图和频域图 (10000Hz,4order)

以 12000Hz 作为截止频率，滤波后播放，得到的时域图和频域图如图38所示。

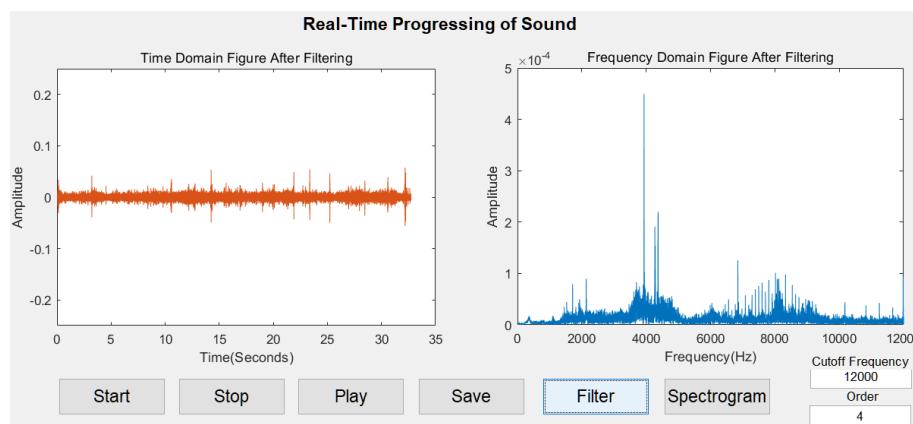


图 38 滤波后的时域图和频域图 (12000Hz,4order)

以 10000Hz 作为截止频率，提高阶数，滤波得到的时域图和频域图如图39所示。

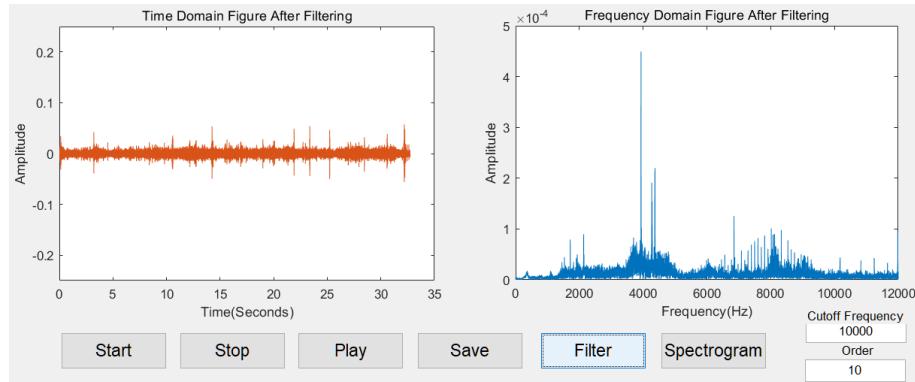


图 39 滤波后的时域图和频域图 (10000Hz,10order)

原始语音信号的语谱图如图40所示。

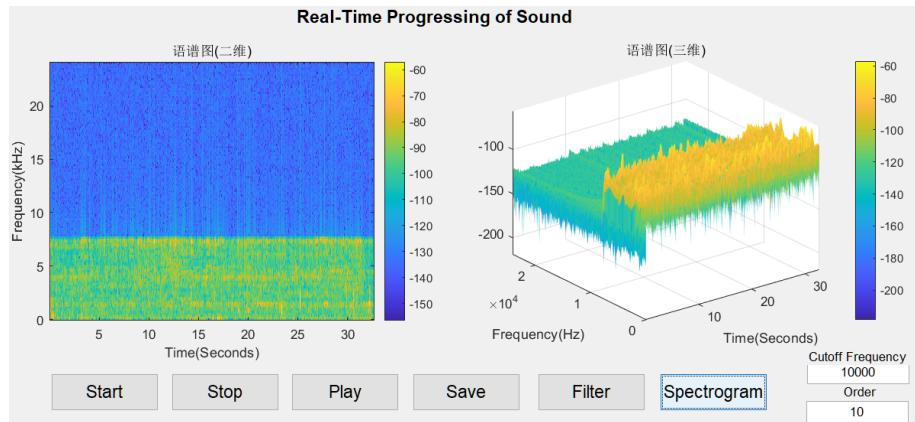


图 40 原始语音信号的语谱图

综上所述，得到的结论有

- 最合适的滤波截止频率为 10000Hz, 80000Hz 杂音依然明显存在, 12000Hz 的效果则与 10000Hz 差不多。
- 阶数对滤波效果的影响较小，即时升到 10 阶，滤波效果也基本相同。
- 从语谱图中可以看到能量主要分布在 8000Hz、4000H 和 2000Hz 以及低频部分，可以认为具有谐波特征。9000Hz 以上几乎没有能量分布，这也与截止频率的选取是对应的。

八、完整代码

这一部分的代码顺序如下：

1. 思路一：基于间断分析的域分析代码实现；

2. 思路二：基于整体分析的域分析代码实现；

3. 思路三：基于 GUI 的域分析代码实现。

思路一：基于间断分析的域分析代码实现

```

1 %% 执行程序1
2 clear all
3 Fs = 48000      %采样频率
4 Duration = 1    %间隔时间
5 N = Fs * Duration %采样点数
6 ExcuteTime = 20 %执行时间
7 i = 0
8 Recorder = audioDeviceReader('NumChannels',1,'SampleRate',Fs,'SamplesPerFrame',N)
9 %录音器参数设置
10 while i < (ExcuteTime/Duration)
11     i = i + 1;
12     %将录音数据存入ReceiveData中 step (Recorder):打开录音机
13     [ReceiveData, nOverrun] = step(Recorder)
14     %对录音数据进行快速傅里叶变换
15     y = abs(fftshift(fft(ReceiveData,N)/N));
16     drawnow
17     subplot(4,2,1)
18     plot(ReceiveData,'Color','b','LineWidth',1)
19     xlim([0 5000])
20     title({'第',num2str(i*Duration),'s时的时域波形'});
21     subplot(4,2,2)
22     w= linspace(-Fs/2,Fs/2,N)
23     plot(w,y)
24     title({'第',num2str(i*Duration),'s时的频域波形'});
25     xlim([0 5000])
26
27 %滤波处理
28 Wc = 0.1
29 [b, a] = butter(4, Wc,'low');
30 FilterData = filtfilt(b,a,ReceiveData);
31 subplot(4,2,3)
32 plot(FilterData,'Color','b','LineWidth',1)
33 title({'第',num2str(i*Duration),'s时的时域波形(滤波后)'});
34 xlim([0 5000]) % ylim([-0.5 0.5])
35 y_filter = abs(fftshift(fft(FilterData,N)/N));
36 subplot(4,2,4)
37 plot(w,y_filter)
38 title({'第',num2str(i*Duration),'s时的频域波形(滤波后)'});
39 xlim([0 5000])
40 %语谱图
41 windowLength = 256;%帧长
42 win = hamming(windowLength,'periodic');%窗口函数（汉明窗）
43 overlap = 128; %帧移（一般为帧长的一半）
44 fTLength = windowLength; %做DFT的点数，一般和帧长一样
45 subplot(4,2,[5 7])
46 spectrogram(ReceiveData,win,overlap,fTLength,Fs,'yaxis');
47 title('实时语谱图(二维)')
48 %title({'第',num2str(i*Duration),'s时的语谱图'})
49 subplot(4,2,[6 8])

```

```

50      [S,F,T,P] = spectrogram(ReceiveData,win,overlap,ffTLength,Fs);
51      surf(T,F,10*log10(P),'edgecolor','none'); axis tight;
52      %view(0,90); % 时谱图采用角度
53      xlabel('Time(Seconds)'); ylabel('频率(Hz)');colorbar
54      title ('实时语谱图(三维)')
55      %title({['第',num2str(i*Duration), 's时的语谱图(三维)']})
56      %pause(0.2) %视觉暂留
57  end

```

思路二：基于整体分析的域分析代码实现

```

1 %% 执行程序2
2 clear all
3 Fs = 44100, nBits = 16, NumChannels = 1,Duration = 5
4 recObj = audiorecorder(Fs,nBits,NumChannels)
5 disp('Start speaking.')
6 recordblocking(recObj,Duration);
7 disp('End of Recording.');
8 data = getaudiodata(recObj)
9 t = [1/Fs:1/Fs:Duration]
10 sound(data,Fs)
11 plot(t,data),title ('时域图')
12 DrawFFT(data,Fs)
13 spectrogram_two(data,Fs)
14 spectrogram_three(data,Fs)
15 % 滤波处理
16 Wc = 0.1
17 [b, a] = butter(4, Wc,'low');
18 Data = filtfilt (b,a,data);
19 sound(Data,Fs)
20 plot(t,Data),title('时域图')
21 DrawFFT(Data,Fs)
22 spectrogram_two(Data,Fs)
23 spectrogram_three(Data,Fs)
24 %% 执行程序3
25 function [ ] = DrawFFT( x, Fs )
26 L = length(x);
27 NFFT = 2^nextpow2(L);          %确定FFT变换的长度
28 y = fft(x, NFFT)/L;
29 f = Fs/2*linspace(0,1,NFFT/2+1); %频率向量
30 figure()
31 plot(f, 2*abs(y(1:NFFT/2+1))); %绘制频域图像
32 xlim([0 4000])
33 title ('幅度谱');
34 xlabel('Frequency (Hz)');
35 ylabel('|y(f)|');
36 end
37 function [ ] = spectrogram_two(ReceiveData,Fs)
38 windowLength = 256;%帧长
39 win = hamming(windowLength,'periodic');%窗口函数 (汉明窗)
40 overlap = 128; %帧移 (一般为帧长的一半)
41 ffTLength = windowLength; %做DFT的点数, 一般和帧长一样
42 figure()
43 spectrogram(ReceiveData,win,overlap,ffTLength,Fs,'yaxis');

```

```

44 title ('语谱图(二维)')
45 end
46 function [ ] = spectrogram_three(ReceiveData,Fs)
47 figure()
48 windowLength = 256;%帧长
49 win = hamming(windowLength,'periodic');%窗口函数 (汉明窗)
50 overlap = 128; %帧移 (一般为帧长的一半)
51 ffTLength = windowLength; %做DFT的点数, 一般和帧长一样
52 [S,F,T,P] = spectrogram(ReceiveData,win,overlap,ffTLength,Fs);
53 surf(T,F,10*log10(P),'edgecolor','none'); axis tight;
54 xlabel('Time(Seconds)'); ylabel('频率(Hz)');colorbar
55 title ('语谱图(三维)')
56 end

```

思路三：基于 GUI 的域分析代码实现

```

1 function varargout = SoundRecorderDemo(varargin)
2 gui_Singleton = 1;
3 gui_State = struct('gui_Name',    mfilename, ...
4                     'gui_Singleton', gui_Singleton, ...
5                     'gui_OpeningFcn', @SoundRecorderDemo_OpeningFcn, ...
6                     'gui_OutputFcn', @SoundRecorderDemo_OutputFcn, ...
7                     'gui_LayoutFcn', [], ...
8                     'gui_Callback', []);
9 if nargin && ischar(varargin{1})
10     gui_State.gui_Callback = str2func(varargin{1});
11 end
12 if nargout
13     [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
14 else
15     gui_mainfcn(gui_State, varargin{:});
16 end
17 function SoundRecorderDemo_OpeningFcn(hObject, eventdata, handles, varargin)
18 handles.output = hObject;
19 handles.recObj = audiorecorder(48000, 16, 2,-1);
20 handles.recObj.TimerFcn={@RecDisplay,handles};
21 handles.recObj.TimerPeriod=0.25;
22 handles.playSpeed=1;
23 handles.order = 4;
24 handles.filter = 1e3;
25 guidata(hObject, handles);
26 function varargout = SoundRecorderDemo_OutputFcn(hObject, eventdata, handles)
27 varargout{1} = handles.output;
28 function pushbutton1_Callback(hObject, eventdata, handles)
29 record(handles.recObj);
30 function pushbutton2_Callback(hObject, eventdata, handles)
31 stop(handles.recObj)
32 function pushbutton3_Callback(hObject, eventdata, handles)
33 handles.myRecording = getaudiodata(handles.recObj);
34 handles.playObj = audioplayer(handles.myRecording,handles.playSpeed*handles.recObj.SampleRate);
35 play(handles.playObj);
36 guidata(hObject, handles);
37 function pushbutton4_Callback(hObject, eventdata, handles)
38 [ file ,path] = uiputfile(['soundDemo' num2str(handles.playSpeed) '.wav'],'Save recorded sound');

```

```
39 if file
40     audiowrite([path '\\' file ], handles.myRecording,handles.playSpeed*hObject.SampleRate)
41 end
42 function RecDisplay(hObject, eventdata,handles)
43 handles.myRecording = getaudiodata(handles.recObj);
44 plot(handles.axes1,(1:length(handles.myRecording))/handles.recObj.SampleRate,handles.myRecording)
45 title (handles.axes1,'Real-Time Time Domain Figure')
46 ylim(handles.axes1,[-0.25 0.25]), xlabel(handles.axes1,'Time(Seconds)'), ylabel(handles.axes1,'Amplitude')
47 drawnow;
48 Fs = 48000;
49 L = length(handles.myRecording);
50 NFFT = 2^nextpow2(L);          %确定FFT变换的长度
51 y = fft(handles.myRecording, NFFT)/L;
52 f = Fs/2*linspace(0,1,NFFT/2+1); %频率向量
53 plot(handles.axes2, 2*abs(y(1:NFFT/2+1))); %绘制频域图像
54 title (handles.axes2,'Real-Time Frequency Domain Figure')
55 xlim(handles.axes2,[0 12000]), xlabel(handles.axes2,'Frequency(Hz)'), ylabel(handles.axes2,'Amplitude')
56 drawnow;
57 % % 参数设置2
58 function edit1_Callback(hObject, eventdata, handles)
59 handles.order = str2double(get(hObject,'String'));
60 guidata(hObject,handles)
61 function edit1_CreateFcn(hObject, eventdata, handles)
62 if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
63 set(hObject,'BackgroundColor','white');
64 end
65 function pushbutton5_Callback(hObject, eventdata, handles)
66 Fs = 48000;
67 fc = handles.filter
68 Wc = fc/(Fs/2)
69 n = handles.order
70 [b, a] = butter(n, Wc,'low');
71 handles.myRecording = getaudiodata(handles.recObj);
72 Data = filtfilt (b,a,handles.myRecording);
73 %绘制时域图像
74 plot(handles.axes1,(1:length(Data))/handles.recObj.SampleRate,Data)
75 ylim(handles.axes1,[-0.25 0.25]), xlabel(handles.axes1,'Time(Seconds)'), ylabel(handles.axes1,'Amplitude')
76 title (handles.axes1,'Time Domain Figure After Filtering')
77 drawnow;
78 %绘制频域图像
79 L = length(Data);
80 NFFT = 2^nextpow2(L);          %确定FFT变换的长度
81 y = fft(Data, NFFT)/L;
82 f = Fs/2*linspace(0,1,NFFT/2+1); %频率向量
83 plot(handles.axes2, 2*abs(y(1:NFFT/2+1)));
84 xlim(handles.axes2,[0 12000]), xlabel(handles.axes2,'Frequency(Hz)'), ylabel(handles.axes2,'Amplitude')
85 title (handles.axes2,'Frequency Domain Figure After Filtering')
86 drawnow;
87 sound(Data,Fs)
88
89 function pushbutton6_Callback(hObject, eventdata, handles)
90 ReceiveData = getaudiodata(handles.recObj);
91 Fs = 48000;
92 windowLength = 256;%帧长
93 win = hamming(windowLength,'periodic');%窗口函数（汉明窗）
94 overlap = 128; %帧移（一般为帧长的一半）
```

```

95 fftLength = windowLength; %做DFT的点数，一般和帧长一样
96 [S,F,T,P] = spectrogram(ReceiveData(:,1),win,overlap,fftLength,Fs);
97 axes(handles.axes1)
98 spectrogram(ReceiveData(:,1),win,overlap,fftLength,Fs,'yaxis');
99 xlabel('Time(Seconds)'); ylabel('Frequency(kHz)');colorbar
100 title ('语谱图(二维)')
101 axes(handles.axes2)
102 surf(T,F,10*log10(P),'edgecolor','none'); axis tight;
103 xlabel('Time(Seconds)'); ylabel('Frequency(Hz)');colorbar
104 title ('语谱图(三维)')
105
106 % 参数设置1
107 function edit2_Callback(hObject, eventdata, handles)
108 handles.filter = str2double(get(hObject,'String'));
109 guidata(hObject,handles)
110
111 function edit2_CreateFcn(hObject, eventdata, handles)
112 if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
113     set(hObject,'BackgroundColor','white');
114 end

```

九、课程设计总结

9.1 结论与特色

仿真设计是飞行器信息系统设计的关键方法，本文完成了基于 SDR 的飞机通信导航监视系统信号仿真设计和语音信号采集处理设计等任务，主要过程和结果如下：

1. 阐述了高频/甚高频通信系统的基本原理，对不同的仿真实现方法进行了分析；阐述了 VOR 工作的基本原理，对任务实施过程进行了规划分析；
2. 基于 RTL-SDR 硬件和支持包，搭建了 AM 通信系统 simulink 仿真模型，阐述了所用模块的参数意义和设置方案，对 RTL-SDR 数据块作为输入信号的情况进行了仿真实现和结果分析；
3. 基于 GUI 设计界面，对通信系统的每一个过程进行了设计说明和实现，建立了完整的 AM 通信系统，实现了可控载波下的实时语音信号时频域分析和调制解调；
4. 基于 GUI 设计界面，绘制了 VOR 仿真平台，对主子函数进行了详细的说明和设置，编写了基于坐标计算方位角和基于信号处理计算方位角的计算子函数，实现了飞行任务中快速获得正确方位并给出罗盘指针方向的效果；对于某一航路点也展开了信号产生和处理的详细分析，具有有效性和可信度；

5. 基于 GUI 设计界面，绘制了 ILS 仿真平台，对具体信号调制解调进行了分析，实现了飞行在降落过程中基于所处位置快速获得航道偏离指示并给出偏离角度的效果；
6. 在语音信号采集处理设计中，阐述比较了不同录音函数的特点，对之后的滤波处理和语谱图分析也进行了简要说明；
7. 在老师所给文件基础上进行优化，形成了基于间断分析的域分析，又克服了录音的同步问题，建立了基于整体分析的域分析办法，但均存在较大漏洞；
8. 基于 GUI 设计界面，引入定时器，使得录音和信号处理同时进行，克服了同步问题，也做到了实时处理。以某首歌播放为例，基于信号峰值分析了滤波参数的选取，对原始信号的语谱图能量进行了简要分析。

本次课程设计的特色优势之处总结如下：

1. 在 AM 通信系统过程中，分别采用 simulink 和 GUI 进行实现，从接收信号到完整通信过程，同时也大大提高了运行速度。
2. 在 VOR 系统测向仿真过程中，采用 GUI 界面，灵活性和可交互性强。对信号产生和处理过程进行了较多考虑分析，最终得到的结果误差小，运算速度快。
3. 在语音信号采集处理设计中采用了多种思路，具有层次性和逻辑性，突出了探索问题和解决问题的过程，并在最终达到了较好的效果。

9.2 不足与展望

本次课程设计的主要不足之处总结如下：

1. 在 AM 通信系统过程中，没有真正做到信号发射和双向通信，simulink 和 GUI 的方法各有特点，没有很好的将实时发送和运算速度快结合起来；
2. 在 VOR 系统测向仿真过程中，没有对误差大（5 度及以上）的区域进行进一步分析，也没有考虑信号在传输过程中的衰减；
3. 在语音信号采集处理设计中对语谱图的局部分析不到位，滤波上没有考虑现代滤波，导致信号最终依然存在一定噪声。

针对上述不足可采用的办法分别对应如下：

1. 在 AM 通信系统过程中，将 RTL-SDR 接口引入 GUI 中，可以做到实时发送，但快速处理暂时没有好的办法；
2. 在 VOR 系统测向仿真过程中，考虑“天线扫掠”，即对误差较大的区域进行

进一步细化，结合频域图和相关分析法进一步优化，对确实无法降低的区域进行最终结果的微调；

3. 在语音信号采集处理设计中引入针对语谱图的可调参数或者引入其他窗口进行窄带分析。

9.3 收获与建议

本次课程涉及跨度时间长、涉及内容多、面临困难也多，因此完成任务的过程也就是克服一个个困难的过程，其中主要收获有：

1. 从搜集资料角度讲，首先找不到适配的 SDR 相关的教程，对于任务中高频和 AM 相结合的情况无从下手。这一问题解决主要是先通过淘宝购置 SDR 接收机，制作了简易 FM 收音机^[9]，从而克服了对 SDR 的恐惧。其次是 VOR 仿真的资料很少，几乎只能通过课本理论部分分析，但好在课本较为详细，只要自己不断根据实际信号不断调整，最终也能得到可观的效果；
2. 从学习知识角度讲，本次课程设计主要包括语音信号处理学习、通信系统建模仿真与应用以及 VOR 工作原理的理解三部分，因此在不断学习过程中大大加强了与《数字信号处理》和《通信导航与雷达课程》的联系，也对语音信号处理领域有了初步了解。
3. 从实践学习的角度讲，仿真过程的不少参数都是实践中的来的，仅仅通过原理分析很难完全达到任务要求，这体现了经验公式的重要性。

对课程的建议和反馈主要包括：

1. 从课程进度角度讲，本次课程设计跨度时间很，导致经常忘记具体任务所在和进度，除了对学生自身学习进一步安排外，这需要老师不断提醒和催促，并对任务安排进行合理划分。此外可以效仿上学期的《飞行器控制系统设计》，每隔两周把学生召集讨论，有利于课程设计的推进；
2. 从实践学习的角度讲，本次课程设计的学习面广泛，初始实践过程中存在不少困难，但好在语音信号采集处理设计老师已经提供了例程，虽然存在不少问题，但值得摸索和实践，毕竟提出问题才能解决问题。因此希望老师能够多多提供相关思路或者资料，指导学生完成任务；
3. 从课程交流和创新的角度讲，本次课程设计命题在 2019 级之前的课程设计中并不存在，因此一切都需要自己摸索，尤其是飞机无线电设备的工作仿真。在这一过程中，单单依靠自身的力量很容易误入歧途，如 simulink 的滤波器存在缺陷，无法做到实时零相位滤波，这样会给 VOR 信号处理带来极大困

难，因此只能采用 m 函数实现。

参考文献

- [1] [https://ww2.mathworks.cn/campaigns/offers/
download-rtl-sdr-ebook.html](https://ww2.mathworks.cn/campaigns/offers/download-rtl-sdr-ebook.html)
- [2] 王明慧. 通信仿真案例设计 [J]. 通信电源技术,2021,38(02):117-119.
- [3] <https://www.docin.com/p-1978518949.html>
- [4] [https://wenku.baidu.com/view/0f7ee7e3aff8941ea76e58fafab069dc502247c3.
html](https://wenku.baidu.com/view/0f7ee7e3aff8941ea76e58fafab069dc502247c3.html)
- [5] http://www.luizmonteiro.com/Learning_VOR_Sim.aspx
- [6] <https://ww2.mathworks.cn/help/signal/ref/butter.html>
- [7] <https://blog.csdn.net/lzrtutu/article/details/78882715>
- [8] <https://www.bilibili.com/video/av21914026/>
- [9] <https://www.bilibili.com/video/BV1Pb4y1x72i>