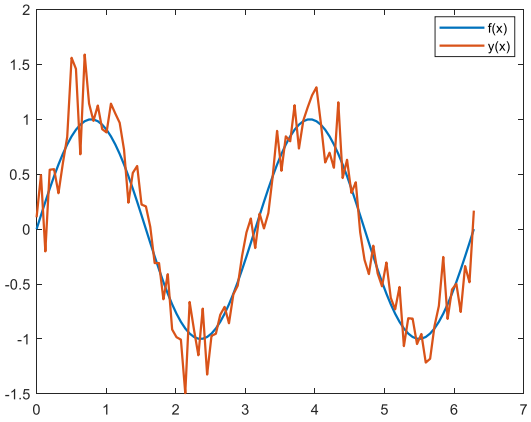
Q1. 设$f(x)=\sin 2x, x\in[0,2\pi]$，假设采样间距为$\frac{\pi}{50}$。类似于今天的例题，设$y(x)=f(x)+\epsilon$，在设定 `rng default` 后，设$\epsilon$为标准差为 0.2 的正态分布（高斯）噪声。图示并分析$f(x)$与$y(x)$的离散傅里叶变换，而后使用均值滤波法、傅里叶系数阈值法进行去噪，无需使用其他方法。（写出代码、图示、去噪信噪比结果以及原因解释）

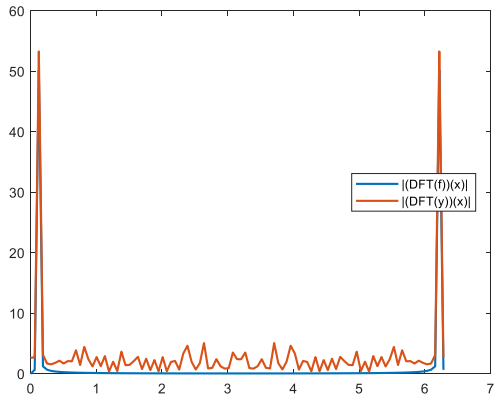
答：（1）函数的生成与计算

```
x = 0:pi/50:2*pi;
l = length(x);
f = sin(2.*x);
rng default
y = f + randn(1,l)*0.2;
plot(x,f,x,y,'LineWidth',1.5);
legend('f(x)','y(x)','Location','best');
```



（2）离散傅里叶变换与模的分析

```
ff = fft(f);
yf = fft(y);
figure;
plot(x,abs(ff),x,abs(yf),'LineWidth',1.5);
legend('|(DFT(f))(x)|','|(DFT(y))(x)|','Location','best');
```



因为函数周期恰为π，故仅有第 3 和 n-2 有大系数，其余系数几乎为 0。之所以仍不等于 0，是因为采样周期较少或采样点数不足时，频率识别在每周期的采样点减少时，傅里叶系数的误差会进一步的增大。故真实的离散傅里叶小系数并不是完全为 0。

（3）两类方法的结果与分析

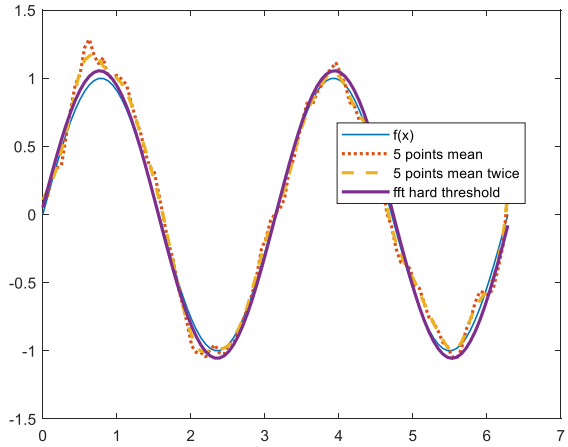
```
snr_noise = snr(f,y-f)
snr_noise =  9.6341

y1 = smooth(y); y1 = y1';
snr_5points_mean = snr(f,y1-f)
snr_5points_mean = 16.6124

y2 = smooth(y1); y2 = y2';
snr_5points_twice_mean = snr(f,y2-f)
snr_5points_twice_mean = 18.6912

Ff = fft(y);
Ff(abs(Ff)<10)=0.0;
y2 = ifft(Ff);
snr_fft_hs = snr(f,y2-f)
snr_fft_hs = 22.7585

plot(x,f,'LineWidth',1);
hold on
plot(x,y1,':',x,y2,'--',x,y3,'LineWidth',2);
legend('f(x)','5 points mean','5 points mean twice','fft hard
threshold','Location','best');
hold off
```



从结果发现傅里叶变换的恢复效果并不如课堂的实例好，主要是因为噪声标准差更大，其次是因为真实函数$\sin 2x$的频率更高，在相同的采样密度下，傅里叶系数在除频率所在点之外，有着更大的误差。因此，虽然傅里叶系数的硬阈值算法效果更佳，却难以获得 30 左右的信噪比。软阈值算法反而可以得到超过 25 的 SNR（阈值设为 5），这个我最初也没想到。

Q2.参照 Maryslamb.m 文件，加入自己的乐谱或音符（乐理知识不足的可以做简单的曲子），完成一段音乐曲目的制作。
答： 本题为开放性试题，这里仅提供一些相应的功能解释。

音乐是由长短等各种音符构成的。按照一秒钟 44100 个时间采样点，在制定了 1/16 音符时长的基础上，首先定义出各种长度音符的时间采样序列 t1,t2,…t16。

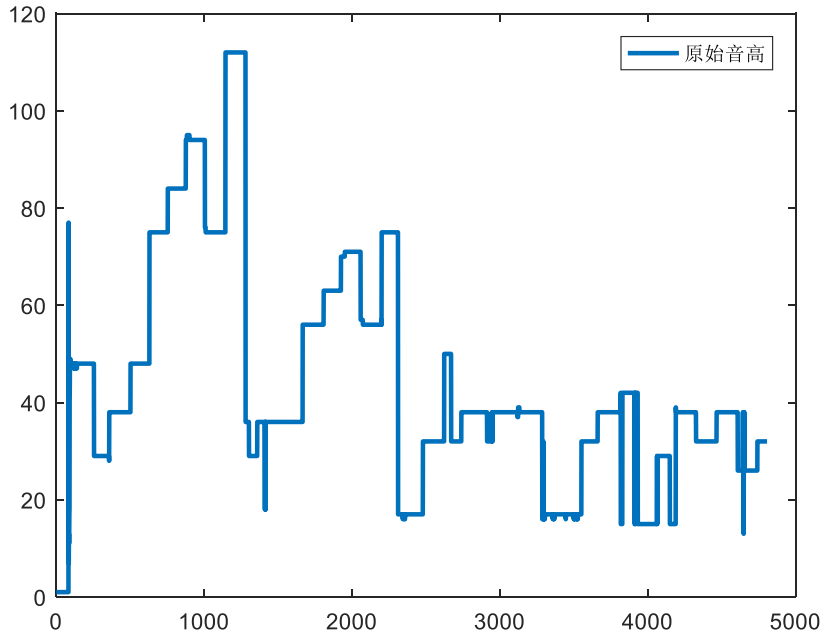
mod1, mod2 等即为设置一个基础的全音符或二分音符的声音包络线，因为每一个音符都包含一种特定频率的振动，设置包络线是因为每一种音符是从无振动到振动加强，最后振动不断减弱。突然出现的恒定频率的振动，容易使不同音符之间的切换显得生硬刺耳，甚至有噪声的感觉。

根据 C 大调基准 261.6Hz,后面的代码设计了 E^b 大调的基准，并为所有出现的音符根据各自不同的频率，设计了具有特定频率的震荡函数。这样声音的频率由音高确定，响度（振幅）由包络线确定。

在乐谱代码中，出现了一处“mi2h+so2h”，表示声音的叠加，此时可以同时听到两种音高同时的奏响。但是超过 1 的函数值在 sound 演奏时会出现噪声，所以在使用 sound 函数播放前需要对声音信号函数进行规范化使函数最大模等于 1。然后音乐就得以成功奏响。

Q3. 选择一首你喜欢的声音片段，或直接使用 Q2 你生成的声音，长度 10 秒左右。尝试使用短时傅里叶变换对音调进行分析。（如果速度很慢除 Ctrl+C 强制中断程序运行外，还可以增大 i 的循环步长到 100 或更大，可以大大减少时间和空间的消耗，人声音乐建议短时窗口设在 6000 以上，纯乐器可稍低些）（图示音调分析结果，并在邮件中粘贴对应的声音片段）
答：详细答案从略，因为只需重复课件内容即可完成，为了提高效率，可以将短时傅里叶变换的采样点大大减少，这里提供一个版本的优化代码：

```
close all; clc; clear;
info = audioinfo('Kevin Kern - Sundial Dreams.mp3');
[y,Fs] = audioread('Kevin Kern - Sundial Dreams.mp3');
ys = y(1:11*Fs,1);
ub = length(ys)-4410;
height = zeros(floor(ub/100),1);
%将短时傅里叶变换计算点间距设置为100个点，大大提高分析效率
for i = 1:100:ub
    temp = ys(i:i+4410);
    tempf = fft(temp);
    [maxv, maxp]=max(abs(tempf));
    height((i-1)/100+1) = maxp;
end
figure;
plot(height,'LineWidth',2);
legend('原始音高');
```



Q4. （选做）根据网盘或作业提供的代码 MarysLamb.m 外，同时参考 16 级同学吴虓杨的 summer 音乐的生成代码，思考不同音色（波形）的设计规律。

（1）可尝试解读并分析这套代码或其他声音信号，解释不同的音色的提取或生成方法。（附加解释与提示：该代码音色系数是用对取样的真实乐器声音短新号段用 cftool 拟合出来的）

(2)题目与答案从略，属于完全开放性的题目

答：（1）吴虓杨的代码 tone1，tone2 和 tone3 提供了三种不同的音色函数，其每一个周期的波形由多个三角函数进行线性组合获取。那么关键就是如何获取线性组合的系数使波形尽可能的具有特点。

为了获取波形，吴虓杨同学通过对 Summer_sample.flac 文件进行单音片段获取，然后截取大约一个周期的波形（事实上可以尝试截取多个周期），并将其代入 cftool 工具，在特定的拟合类型下即可获取对应的线性系数。

此外，这套代码使用的包络线是高斯函数（正态分布的概率密度函数）而不是三角函数，也使得音色会有一些不同的特点。