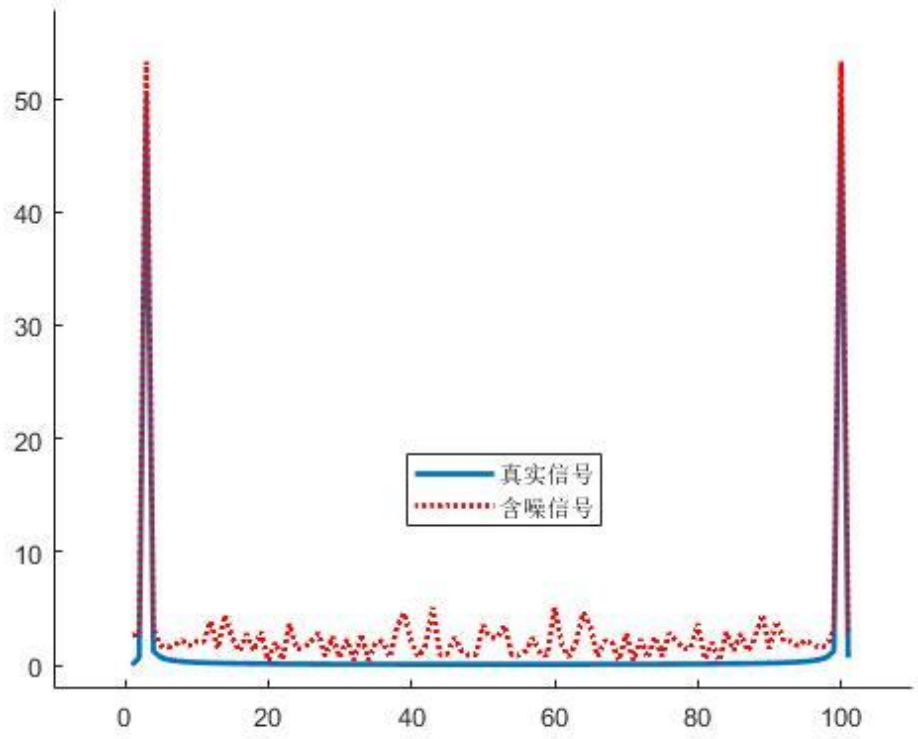
Q1:

```
>> x=0:pi/50:2*pi;
>> rng default
>> f=sin(2*x);
>> y=f+randn(1,101)/5;
% 接下来分别绘制真实信号和含噪信号的离散傅立叶变换
>> clf,hold on
>> plot(abs(fft(f)),'LineWidth',2);
>> plot(abs(fft(y)),'r:','LineWidth',2);
>> legend('真实信号','含噪信号','Location','best');
>> hold off
```




```
% 从图中可以看出真实信号 Y（k）的傅立叶变换仅在 k=3 和 k=100 处有较大的系数模；而含噪信号的傅立叶变换会在中间部分有一些非零的系数。
```

% 五点均值滤波法

```
>> f1= smooth(y);
```

% 去噪信噪比结果

```
>> snr(f,f1'-f)

ans =

    10.0206
```

% 因为噪声是 0 均值的，并且噪声的幅度小于信号本身的强度。故采用简单的均值方法对带噪信号做光滑化处理就可以得到一个近似的恢复信号。

% 傅里叶系数阈值法

```
>> Ff = fft(y);

>> Ff(abs(Ff)<10)=0.0;    % 将绝对值小于 10 的傅里叶系数都设为 0.0

>> f2= ifft(Ff);
```

% 去噪信噪比结果

```
>> snr (f , f2-f)

ans =

    22.7585
```

% 观察上图可知，去除中间部分因噪音信号产生的、绝对值小于 10 的傅立叶系数即可将减少噪音对信号的影响

Q2：

```
%% Matlab《小燕子》
% Coding by 17344011
%% define the basic sound wave
fs = 44100; % Standard sample rate
dt = 1/fs; % Standard sampling time interval
T16 = 1/fs*7200; %To determine the time length of a    1/16 note, suggest as an odd number
t16 = [0:dt:T16];
[temp k] = size(t16);
t1 = linspace(0,16*T16,16*k);% An array with the same length as a full note
t2 = linspace(0,8*T16,8*k);
t4 = linspace(0,4*T16,4*k);
```

```
t4d = linspace(0,6*T16,6*k);%A special array represents a 1/4+1/8 note
t8 = linspace(0,2*T16,2*k);
mod1 = sin(pi*t1/t1(end));% Defining a basic amplitude function (So that the sound won't
suddenly occur or vanish
mod2 = sin(pi*t2/t2(end));
mod4 = sin(pi*t4/t4(end));
mod4d = sin(pi*t4d/t4d(end));
mod8 = sin(pi*t8/t8(end));

%% Frequency and note List
f0 = 2^(1/4)*261.6; % 1 = E^b , which is three half tones higher than C tone
ScaleTable = [1 2^(2/12) 2^(4/12) 2^(5/12) 2^(7/12) 2^(9/12) 2^(11/12)];%Other frequencies
                    % full notes
do2F = mod1.*cos(2*pi*ScaleTable(1)*f0*t1);
re2F = mod1.*cos(2*pi*ScaleTable(2)*f0*t1);
mi2F = mod1.*cos(2*pi*ScaleTable(3)*f0*t1);
fa2F = mod1.*cos(2*pi*ScaleTable(4)*f0*t1);
so2F = mod1.*cos(2*pi*ScaleTable(5)*f0*t1);
la2F = mod1.*cos(2*pi*ScaleTable(6)*f0*t1);
xi2F = mod1.*cos(2*pi*ScaleTable(7)*f0*t1);
blkF = zeros(size(mod1));
                % 1/2 notes
do2h = mod2.*cos(2*pi*ScaleTable(1)*f0*t2);
re2h = mod2.*cos(2*pi*ScaleTable(2)*f0*t2);
mi2h = mod2.*cos(2*pi*ScaleTable(3)*f0*t2);
fa2h = mod2.*cos(2*pi*ScaleTable(4)*f0*t2);
so2h = mod2.*cos(2*pi*ScaleTable(5)*f0*t2);
la2h = mod2.*cos(2*pi*ScaleTable(6)*f0*t2);
xi2h = mod2.*cos(2*pi*ScaleTable(7)*f0*t2);
blkh = zeros(size(mod2));
            % 1/4+1/8 notes
do2fd = mod4d.*cos(2*pi*ScaleTable(1)*f0*t4d);
re2fd = mod4d.*cos(2*pi*ScaleTable(2)*f0*t4d);
mi2fd = mod4d.*cos(2*pi*ScaleTable(3)*f0*t4d);
fa2fd = mod4d.*cos(2*pi*ScaleTable(4)*f0*t4d);
so2fd = mod4d.*cos(2*pi*ScaleTable(5)*f0*t4d);
la2fd = mod4d.*cos(2*pi*ScaleTable(6)*f0*t4d);
xi2fd = mod4d.*cos(2*pi*ScaleTable(7)*f0*t4d);
blkfd = zeros(size(mod4d));
do2fdu = mod4d.*cos(2*pi*(2*ScaleTable(1))*f0*t4d);
            % 1/4 notes
do2f = mod4.*cos(2*pi*ScaleTable(1)*f0*t4);
re2f = mod4.*cos(2*pi*ScaleTable(2)*f0*t4);
mi2f = mod4.*cos(2*pi*ScaleTable(3)*f0*t4);
```

```
fa2f = mod4.*cos(2*pi*ScaleTable(4)*f0*t4);
so2f = mod4.*cos(2*pi*ScaleTable(5)*f0*t4);
la2f = mod4.*cos(2*pi*ScaleTable(6)*f0*t4);
xi2f = mod4.*cos(2*pi*ScaleTable(7)*f0*t4);
blkf = zeros(size(mod4));
do2fu = mod4.*cos(2*pi*(2*ScaleTable(1))*f0*t4);
re2fu = mod4.*cos(2*pi*(2*ScaleTable(2))*f0*t4);
mi2fu = mod4.*cos(2*pi*(2*ScaleTable(3))*f0*t4);
fa2fu = mod4.*cos(2*pi*(2*ScaleTable(4))*f0*t4);
so2fu = mod4.*cos(2*pi*(2*ScaleTable(5))*f0*t4);
la2fu = mod4.*cos(2*pi*(2*ScaleTable(6))*f0*t4);
xi2fu = mod4.*cos(2*pi*(2*ScaleTable(7))*f0*t4);
            % 1/8 notes
do2e = mod8.*cos(2*pi*ScaleTable(1)*f0*t8);
re2e = mod8.*cos(2*pi*ScaleTable(2)*f0*t8);
mi2e = mod8.*cos(2*pi*ScaleTable(3)*f0*t8);
fa2e = mod8.*cos(2*pi*ScaleTable(4)*f0*t8);
so2e = mod8.*cos(2*pi*ScaleTable(5)*f0*t8);
la2e = mod8.*cos(2*pi*ScaleTable(6)*f0*t8);
xi2e = mod8.*cos(2*pi*ScaleTable(7)*f0*t8);
blke = zeros(size(mod8));
do2eu = mod8.*cos(2*pi*(2*ScaleTable(1))*f0*t8);
re2eu = mod8.*cos(2*pi*(2*ScaleTable(2))*f0*t8);
mi2eu = mod8.*cos(2*pi*(2*ScaleTable(3))*f0*t8);

%% Melody
s = [mi2e so2e do2eu la2e so2f blkf mi2e so2e la2e do2eu so2f blkf do2fdu ...
     mi2eu re2fu do2fu re2eu do2eu la2e do2eu so2f blkf mi2fd so2e la2f so2e...
     la2e do2fu re2eu so2e la2f blkf mi2f do2f re2f blkf re2f re2e mi2e so2f...
     so2f do2fu re2e mi2e so2f blkf];
ys=[mi2e so2e do2eu la2e so2f blkf mi2e so2e la2e do2eu so2f blkf do2fdu ...
     mi2eu re2fu do2fu re2eu do2eu la2e do2eu so2f];

%% Processes before play
s = s/max(s); %Balance the amplitude never greater than 1
audiowrite('Swallow_Sample.mp4',ys,fs); %Save the music sample to a file
% audiowrite('Swallow_Sample.mp4',s,fs); %Save the music to a file
sound(s,fs); % Play the sound
```

Q3:

```
>> ub = length(ys)-4410;        % 设置窗口大小为4410,即0.1秒
```

```
>> height = zeros(ub,1);          % 设置一个逐点的音高函数
for i = 1:ub
temp = ys(i:i+4410);      % 对每个时间及以后的4410个采样点截取
tempf = fft(temp);
[maxv, maxp]=max(abs(tempf));        %记录短时傅里叶变换最大模所在位置
height(i) = maxp;
end
>> plot(1:ub,log(height))
>> legend('对数音高','best')
```