一、地震记录合成

1、实验目的

（1）熟悉合成一维地震数据过程
（2）熟悉从实际测井曲线中合成地震记录

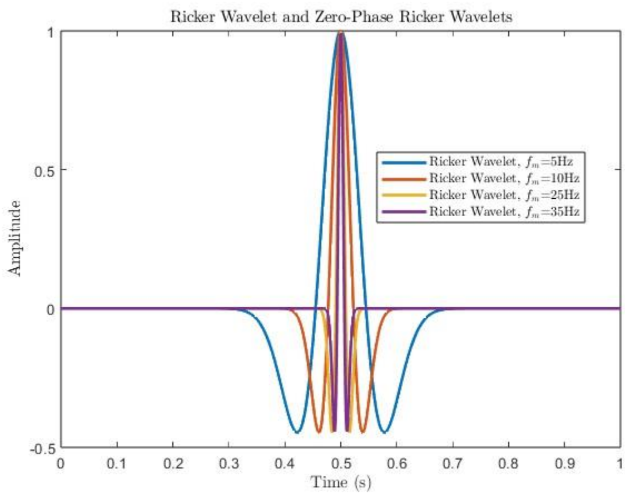
2、实验内容

（1）编制显示地震雷克子波的 Matlab 程序，并在同一张图上绘制主频 5Hz、15Hz、25Hz、35Hz 的零相位地震雷克子波图形；
（2）编制出上述不同频率子波的振幅谱，并绘制出图形；
（3）给定一个速度和密度模型，根据波阻抗公式计算出反射系数，并绘制出其与雷克子波褶积结果；
（4）利用给定的测井声波时差曲线和密度曲线，计算反射系数并合成地震记录。

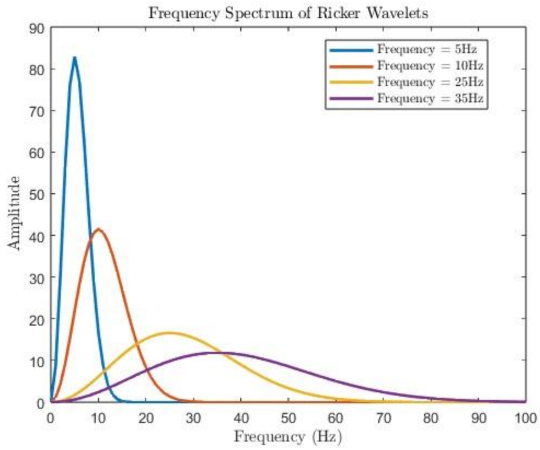
3、实验过程

根据给定的雷克子波公式，带入主频和时间序列可得：

$$r(t)=e^{-(\pi f_m t)^2}\left(1-2\left(\pi f_m t\right)^2\right)$$



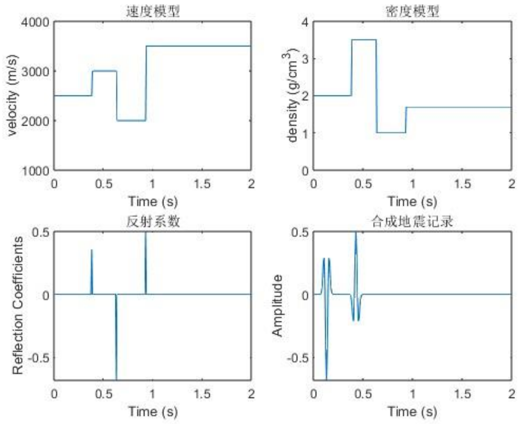
其中f_m为主频，主频越大，波越窄，对其进行傅里叶变化，映射至频率域:



观察不同主频的时间/频率域特性,根据给定的速度和密度模型，通过反射系数计算公式可得：

$$R(n)=\frac{\rho_{i+1}V_{i+1}-\rho_i V_i}{\rho_{i+1}V_{i+1}+\rho_i V_i}\qquad i=1,2,\cdots,M\text{-}1$$

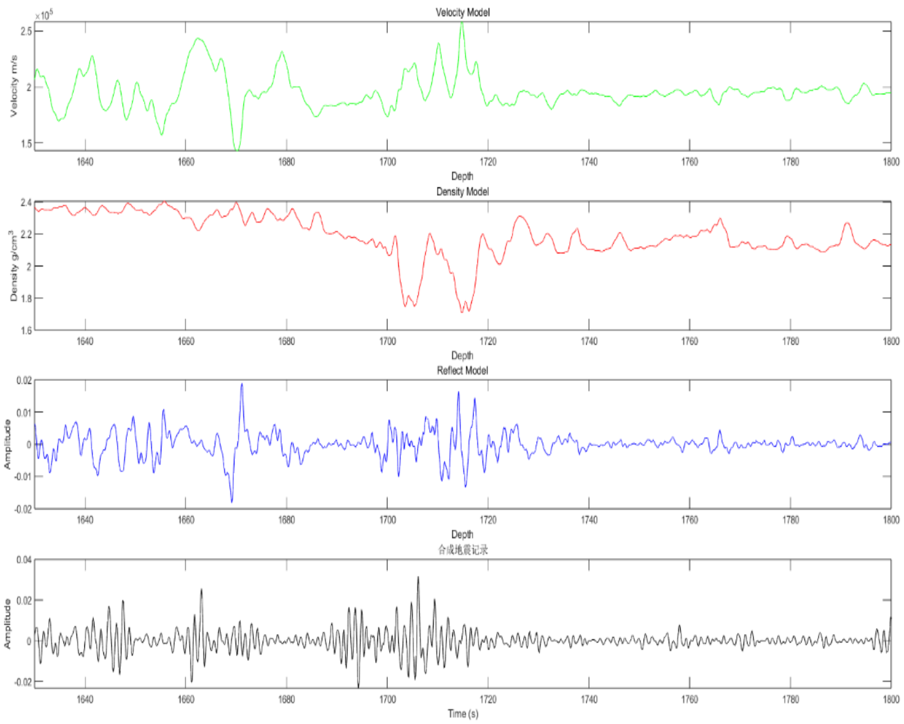
得到界面的反射序列，而后与雷克子波进行卷积操作，则可以合成地震记录：



最后，对于真实测井曲线进行读取，其数据格式为5000×3的二维矩阵，其中第一列为测井深度$Depth$、第二列为声波时差$time$、第三列为密度，通过以下的换算关系可以得到速度$velocity$随深度的关系：

$$velocity=\frac{Depth}{time}$$

得到速度和密度模型后，再次使用反射系数计算公式即可得到反射序列，最后使用主频为 15Hz 的峰值在 0.5s 的雷克子波与得到反射系数褶积即可得到下图：



Code:

```
clear, clc, close all;

t = linspace(0, 1, 1000);
frequencies = [5, 10, 25, 35];
bias = 0.5;

%% (1)绘制雷克子波不同频率波形
figure;
for f = frequencies
    ricker = ricker_wavelet(t - bias, f);
    plot(t, ricker, 'LineWidth', 2, 'DisplayName', sprintf('Ricker Wavelet, $f_m$=%dHz', f));
    hold on;
end

title('Ricker Wavelet and Zero-Phase Ricker Wavelets', 'Interpreter', 'latex');
xlabel('Time (s)', 'Interpreter', 'latex');
ylabel('Amplitude', 'Interpreter', 'latex');
legend('Location', 'Best', 'Interpreter', 'latex');
hold off;
```

```matlab
%% (2)频谱分析
figure;
for f = frequencies
    ricker = ricker_wavelet(t - bias, f);
    spectrum = fft(ricker);
    freq = linspace(0, 1/(2*(t(2)-t(1))), length(t)/2);
    plot(freq, abs(spectrum(1:length(t)/2)), 'LineWidth', 2,
'DisplayName', sprintf('Frequency = %dHz', f));
    hold on;
end

title('Frequency Spectrum of Ricker Wavelets', 'Interpreter',
'latex');
xlabel('Frequency (Hz)', 'Interpreter', 'latex');
ylabel('Amplitude', 'Interpreter', 'latex');
legend('Location', 'Best', 'Interpreter', 'latex');
xlim([0, 100]);
hold off;

%% (3)波阻抗
figure;

% 速度模型
% velocity = 2500 * (time > 0 & time < 0.5) + ...
%            3000 * (time >= 0.5 & time < 0.75) + ...
%            2000 * (time >= 0.75 & time < 1.3) + ...
%            3500 * (time >= 1.3);

% 读取速度数据
load('velocity.mat');
% 读取密度数据
load('density.mat');

time = linspace(0, 2, length(velocity));

subplot(2, 2, 1);
plot(time, velocity);
ylim([1000, 4000]);
xlabel("Time (s)");
ylabel("velocity (m/s)");
title("速度模型");


% density = 1.5 * (time > 0 & time < 0.5) + ...
```

```
%           1.8 * (time >= 0.5 & time < 0.75) + ...
%           1.3 * (time >= 0.75 & time < 1.3) + ...
%           1.4 * (time >= 1.3);
subplot(2, 2, 2);
plot(time, density);
ylim([0, 4]);
xlabel("Time (s)");
ylabel("density (g/cm^3)");
title("密度模型");

R = zeros(size(velocity));
for i = 1: length(velocity) - 1
    R(i) = (density(i+1)*velocity(i+1) - density(i)*velocity(i))
/ ...
           (density(i+1)*velocity(i+1) + density(i)*velocity(i));
end

subplot(2, 2, 3);
plot(time, R);
% ylim([-0.5, 0.5]);
xlabel("Time (s)");
ylabel("Reflection Coefficients");
title("反射系数");

% 对反射系数和雷克子波进行卷积
conv_trace = conv(R, ricker_wavelet(time - bias, 15), 'same');
subplot(2, 2, 4);
plot(time, conv_trace);
xlabel('Time (s)')
ylabel('Amplitude')
title("合成地震记录");




%% (4) 互相关
% figure;
% xcorr_trace = xcorr(conv_trace, ricker_wavelet(time - bias, 15),
'coeff');
% recovered_reflection_xcorr = xcorr_trace(numel(conv_trace):end);
% plot(time, recovered_reflection_xcorr);
% xlabel('Time (s)')
% ylabel('Amplitude')
% title("Xcorr of Reflection Coefficients and Ricker Wavelet");
```

```matlab
%% 读取数据
n1 = 5000;
n2 = 3;

filename1 = './well.dat';
filename2 =  './well2.dat';

fid1 = fopen(filename1, 'r');
fid2 = fopen(filename2, 'r');

well1 = fread(fid1, [n1, n2], 'float');
well2 = fread(fid2, [n1, n2], 'float');

fclose(fid1);
fclose(fid2);

% 分解矩阵为向量
Depth = well2(:, 1);
velocity_model = well2(:, 1) ./ well2(:, 2);
density_model = well2(:, 3);
% reflection_coefficient_model = well(:, 3);

% 绘制速度模型
figure;
subplot(4, 1, 1);
plot(Depth, velocity_model ,'g');
title('Velocity Model');
xlabel('Depth');
xlim([1630, 1800]);
ylabel('Velocity m/s');

% 绘制密度模型
subplot(4, 1, 2);
plot(Depth, density_model, 'r');
title("Density Model");
xlabel("Depth");
xlim([1630, 1800]);
ylabel("Density g/cm^3");

% 绘制反射系数模型
Rflect = zeros(size(velocity_model));
for i = 1: length(velocity_model) - 1
    Rflect(i) = (density_model(i+1)*velocity_model(i+1) - 
density_model(i)*velocity_model(i)) / ...
```

```
            (density_model(i+1)*velocity_model(i+1) +
density_model(i)*velocity_model(i));
end

subplot(4, 1, 3);
plot(Depth, Rflect, 'blue');
title("Reflect Model");
xlabel("Depth");
ylabel('Amplitude')
xlim([1630, 1800]);


% 对反射系数和雷克子波进行卷积
conv_trace_my = conv(Rflect, ricker_wavelet(time - bias, 15),
'same');
subplot(4, 1, 4);
plot(Depth, conv_trace_my, 'black');
xlim([1630, 1800]);
xlabel('Time (s)')
ylabel('Amplitude')
title("合成地震记录");

%% 定义雷克子波函数
function r = ricker_wavelet(t, f_m)
    r = exp(-1 * pi^2 * f_m^2 * t.^2) .* (1 - 2 * (pi * f_m * t).^2);
end
```

二、速度分析与动校正

1、实验目的

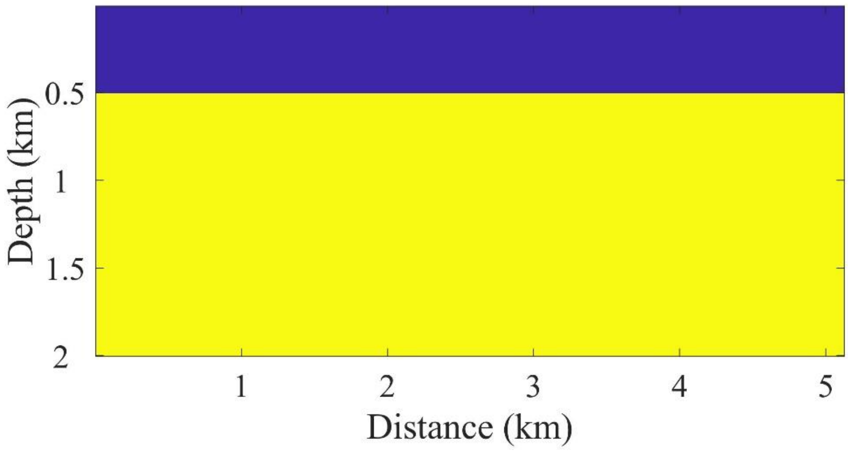
（1）熟悉地震速度建模过程
（2）熟悉动校正和速度分析过程

2、实验内容

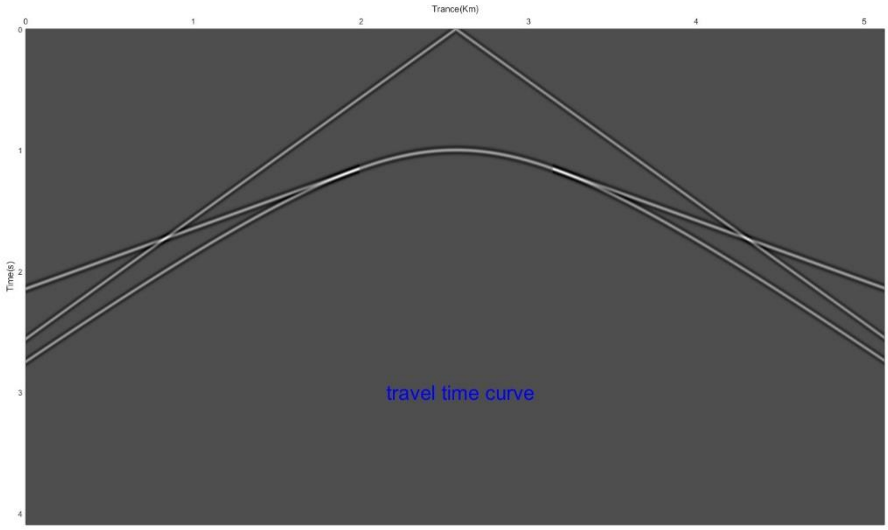
（1）给定二层均匀层状速度模型，根据时距曲线公式计算对应的记录（直达波、反射波、折射波）；
（2）对该反射波进行速度分析，得到相应的速度（此处可以只模拟反射波）；
（3）利用速度分析结果对地震记录进行动校正；
（4）根据动校正结果观测动校拉伸现象及特点。

3、实验过程

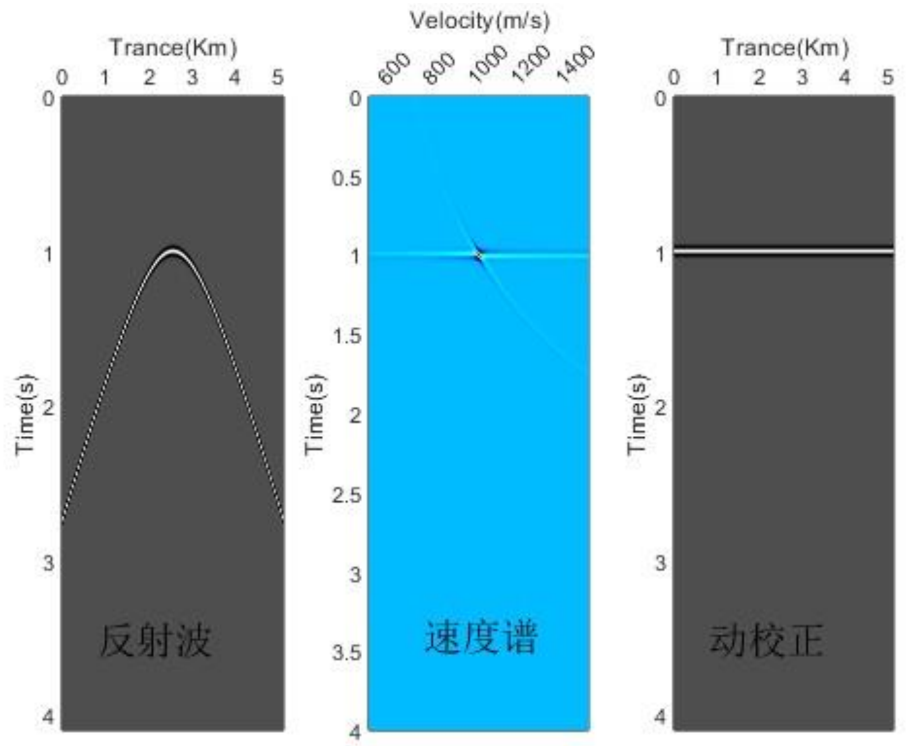
解：
模型参数： 网格大小 dx=dz=10 m，模型大小200 × 512第一层速度 v1=1000 m/s, 第二层速度 v2= 2000 m/s，第一层厚度 h1=500 m， 时间采样间隔 4 ms，时间记录长度 1024 个点 4.096 s， 检波器数量 512。

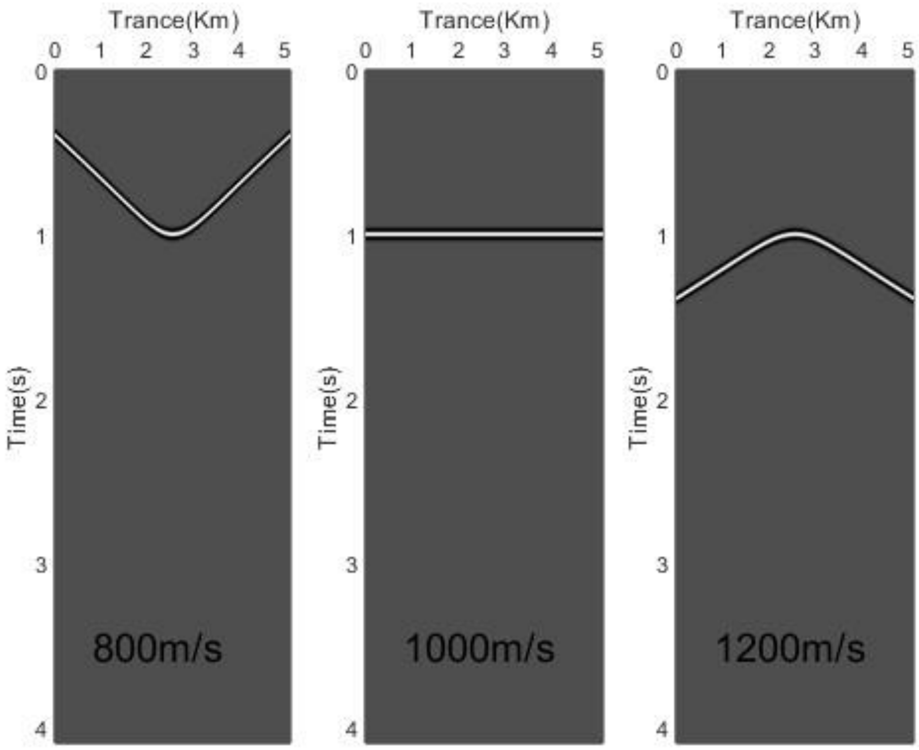


首先，需要初始化长宽为速度模型的零矩阵，并作如下判断：如果深度小于 0.5km，则该网格为 1km/s；若深度大于 0.5km，则该网格为 2km/s，至此构建了与题目相符的速度模型。
其次计算直达波、反射波和折射波的到时曲线，并使用 Ricker 波形模型生成不同类型地震波的子波，并生成地震波的传播曲线。如下图所示：



最后，枚举速度，并计算其动校正量，并进行叠加，记录到数组中，并寻找使得能量最大对应的速度，在该模型中，经过循环寻找，校正的最佳速度为1km/s，并带入反射波时距曲线中，使得时间减去代入最佳速度的动校正量，即可把曲线校平，这里使用了三种速度分别为：800m/s,1000m/s,1200m/s，观察其校正效果。





Code:

```
clear;clc, close all;

% 参数列表
% Parameters
V1 = 1000;                  % Velocity of the first layer (m/s)
V2 = 2000;                  % Velocity of the second layer (m/s)
h = 500;                    % Depth of the model (m)
nx = 512;                   % Number of grid points in the
horizontal direction
nz = 200;                   % Number of grid points in the vertical
direction (m)
nt = 1024;                  % Number of time samples
dx = 10;                    % Grid spacing in the horizontal
direction (m)
dz = 10;                    % Grid spacing in the vertical direction
(m)
dt = 0.004;                 % Time sampling interval (s)
len_x = dx * nx * 10^-3;    % Length of the model in the horizontal
direction (km)
len_z = dz * nz * 10^-3;    % Length of the model in the vertical
direction (km)
len_t = dt * nt;            % Length of the time series (s)
```

```
[X, Z] = meshgrid(linspace(0, len_x, nx + 1), linspace(0, len_z, nz
+ 1));  % Grid of Z and X

%% (1)velocity model
% Generate velocity model
V = zeros(nz + 1, nx + 1);
for i = 1:nz + 1
    for j = 1:nx + 1
        if i * dz <= h
            V(i, j) = V1;  % Velocity in the first layer
        else
            V(i, j) = V2;  % Velocity in the second layer
        end
    end
end

% Plot velocity model
figure(1)
pcolor(X, Z, V)          % Picture of velocity
shading interp           % Clear the grid lines
set(gca, 'YDir', 'reverse', ...
    'xtick', 0:1:5, 'ytick', 0:0.5:2)
set(gca, 'XAxisLocation', 'top')
xlabel('Distance (Km)')
ylabel('Depth (Km)')
text(2, 1.5, 'Velocity Model', 'fontsize', 25);

%% （2）travel time curve
% Grid of time and X
[X, T] = meshgrid(linspace(0, len_x, nx), linspace(0, len_t, nt));
% Generate x values
x = 0:dx * 10^-3:len_x;
% Travel time of direct wave (unit: μs)
dir_t = abs(((x - len_x/2) / V1) * 1000);
% Travel time of reflect wave (μs)
refl_t = abs(1 / (V1 / 1000) * sqrt(((x - len_x/2).^2) + 4 * (h /
1000)^2));
% Refract angle
seta = asind(V1 / V2);
% Across distance
x_cross = 2 * (h / 1000) * tand(seta);
% Initialize array for travel time of refract wave
ref_t = zeros();
```

```
for i=1:length(x)
if abs(x(i)-len_x/2)<x_cross
    ref_t(i)=inf;
else
    ref_t(i)=abs(x(i)-
len_x/2)/(V2/1000)+2*(h/1000)*cosd(seta)/(V1/1000);      %travel
time of refract wave us
end
end

fm=15;
R_direct=zeros();R_reflect=zeros();R_refract=zeros();
t=linspace(0,len_t,nt);

for i=1:length(x)-1
    for j=1:1024
        ricker_direct_wave=exp(-(pi*fm*(t(j)-dir_t(i))).^2) .* (1-
2*(pi*fm*(t(j)-dir_t(i))).^2);      % direct_wave_ricker_wave
        R_direct(j,i)=ricker_direct_wave;

        ricker_reflect_wave=exp(-(pi*fm*(t(j)-refl_t(i))).^2) .* (1-
2*(pi*fm*(t(j)-refl_t(i))).^2);   % reflect_wave_ricker_wave
        R_reflect(j,i)=ricker_reflect_wave;

        ricker_refract_wave=exp(-(pi*fm*(t(j)-ref_t(i))).^2) .* (1-
2*(pi*fm*(t(j)-ref_t(i))).^2);     %refract_wave_ricker_wave
        R_refract(j,i)=ricker_refract_wave;
    end
end

R=R_direct+R_reflect+R_refract;
figure(2)
pcolor(X,T,R)
set(gca,'YDir','reverse',...
     'xtick',0:1:5,'ytick',0:1:4)
set(gca,'XAxisLocation','top')
xlabel('Trance(Km)')
ylabel('Time(s)')
text(2.15, 3, 'travel time curve', 'fontsize',25, 'Color', 'blue')
shading interp
colormap(gray)


%% (3)dynamic correction
```

```
figure(3)
subplot(1,3,1)
sub1=subplot(131);
pcolor(X,T,R_reflect) %picture of reflection curve
set(gca,'YDir','reverse',...
     'xtick',0:1:5,'ytick',0:1:4)
set(gca,'XAxisLocation','top')
xlabel('Trance(Km)')
ylabel('Time(s)')
text(0.8,3.5,'反射波','fontsize',15)
shading interp
colormap(sub1,gray)

subplot(1,3,2)
sub2=subplot(132);
t0=2*h/V1;
dv=10;
v=500:dv:1500;

[vv,tt]=meshgrid(v,t);
R_reflect1=zeros();
Velo=zeros(nt,length(v));

for i=1:length(v)
    delta_t=sqrt(t0*t0+abs(x-len_x/2).*abs(x-
len_x/2)/((v(i)/1000)*(v(i)/1000)))-t0;
    for j=1:length(x)-1
        for k=1:1024
            ricker_reflect_wave2=exp(-(pi*fm*(t(k)-
refl_t(j)+delta_t(j))).^2) .* (1-2*(pi*fm*(t(k)-
refl_t(j)+delta_t(j))).^2);
            R_reflect1(k,j)=ricker_reflect_wave2;
        end
    end
    Velo(:,i)=sum(R_reflect1,2)/size(R_reflect1,1);
end
pcolor(vv,tt,Velo)
axis([500 1500 0 4])
shading interp
set(gca,'YDir','reverse',...
     'xtick',600:200:1400,'ytick',0:0.5:4)
set(gca,'XAxisLocation','top')
xlabel('Velocity(m/s)')
ylabel('Time(s)')
```

```
text(750,3.4,'速度谱','fontsize',15)
colormap(sub2,jet)

subplot(1,3,3)
sub3=subplot(133);
R_reflect1=zeros();
R_reflect2=zeros();
R_reflect3=zeros();
t0=2*h/V1;
[Vel_1,V]=max(max(Velo));
[Vel_,P]=max(Velo(:,V));

if Vel_1==Vel_
    V_suit=v(V);
    fprintf("the velocity is: %f\n", V_suit);
end

v1 = 800;
v3 = 1200;
derta_t1=sqrt(t0*t0+abs(x-len_x/2).*abs(x-
len_x/2)/((v1/1000)*(v1/1000)))-t0;
derta_t2=sqrt(t0*t0+abs(x-len_x/2).*abs(x-
len_x/2)/((v(V)/1000)*(v(V)/1000)))-t0;
derta_t3=sqrt(t0*t0+abs(x-len_x/2).*abs(x-
len_x/2)/((v3/1000)*(v3/1000)))-t0;

for i=1:length(x)-1
    for j=1:1024
        % v1
        ricker_reflect_wave1=exp(-(pi*fm*(t(j)-
refl_t(i)+derta_t1(i))).^2) ...
            .* (1-2*(pi*fm*(t(j)-refl_t(i)+derta_t1(i))).^2);   %
reflect_wave_ricker_wave

        % v2
        ricker_reflect_wave2=exp(-(pi*fm*(t(j)-
refl_t(i)+derta_t2(i))).^2) ...
            .* (1-2*(pi*fm*(t(j)-refl_t(i)+derta_t2(i))).^2);   %
reflect_wave_ricker_wave

        % v3
        ricker_reflect_wave3=exp(-(pi*fm*(t(j)-
refl_t(i)+derta_t3(i))).^2) ...
```

```
            .* (1-2*(pi*fm*(t(j)-refl_t(i)+derta_t3(i))).^2);   %
reflect_wave_ricker_wave

        R_reflect1(j,i)=ricker_reflect_wave1;
        R_reflect2(j,i)=ricker_reflect_wave2;
        R_reflect3(j,i)=ricker_reflect_wave3;
    end
end
pcolor(X,T,R_reflect2)
set(gca,'YDir','reverse',...
     'xtick',0:1:5,'ytick',0:1:4)
set(gca,'XAxisLocation','top')
xlabel('Trance(Km)')
ylabel('Time(s)')
text(0.8,3.5,'动校正','fontsize',15)
shading interp
colormap(sub3,gray)

%% （4） 不同速度的动校正结果
figure(4);
subplot(1, 3, 1);
sub1 = subplot(131);
pcolor(X,T,R_reflect1)
set(gca,'YDir','reverse',...
     'xtick',0:1:5,'ytick',0:1:4)
set(gca,'XAxisLocation','top')
xlabel('Trance(Km)')
ylabel('Time(s)')
text(0.8,3.5,'800m/s','fontsize',15)
shading interp
colormap(sub1,gray)

subplot(1, 3, 2);
sub2 = subplot(132);
pcolor(X,T,R_reflect2)
set(gca,'YDir','reverse',...
     'xtick',0:1:5,'ytick',0:1:4)
set(gca,'XAxisLocation','top')
xlabel('Trance(Km)')
ylabel('Time(s)')
text(0.8,3.5,'1000m/s','fontsize',15)
shading interp
colormap(sub2,gray)
```

```
subplot(1, 3, 3);
sub3 = subplot(133);
pcolor(X,T,R_reflect3)
set(gca,'YDir','reverse',...
     'xtick',0:1:5,'ytick',0:1:4)
set(gca,'XAxisLocation','top')
xlabel('Trance(Km)')
ylabel('Time(s)')
text(0.8,3.5,'1200m/s','fontsize',15)
shading interp
colormap(sub3,gray)
```