```
% gda12_12
% Supports Figure 12.16
%
% model Mars rover Mosbauer spectra using both
% Gaussian and Lorentzian curves, fit via nonlinear
% least squares (Newton's Method).
%
% Note: The code requires you to click on the bottom of the
% peaks, and when you're done, to click to the left of the
% x-axis. This process defines the number of peaks and their
% starting positions and amplitudes.

clear all;

% load data
D=load('../data/mars_soil.txt');
v=D(:,1);
d=D(:,2);
d=d/max(d); % normalize
N=length(d);

% delete negative velocities
i=find(v>=0,1);
v=v(i:N);
d=d(i:N);
N=length(v);

% plot data
figure(1);
clf;
set(gca,'LineWidth',3);
set(gca,'FontSize',14);
title('Click at the bottom of each peak, then left of axes');
hold on;
axis( [0, 12, min(d), max(d)] );
plot(v,d,'r-','LineWidth',2);
plot(v,d,'ro','LineWidth',3);
xlabel('velocity, mm/s');
ylabel('counts');

dolorentzian=0; % toggles between lorentzian curves and Normal curves

% lorentzian curve of peak amplitude a, center velocity v0 and width c
% f(v)   = a c^2 / ( (v-v0)^2 + c^2) )
% df/da  = c^2 / ( (v-v0)^2 + c^2) )
% df/dv0 = 2 a c^2 (v-v0) / ( (v-v0)^2 + c^2) )^2
% df/dc  = 2 a c / ( (v-v0)^2 + c^2) ) - 2 a c^3 / ( (v-v0)^2 + c^2) )^2
% 3 model parameters per lorentzian, (a, v0, c)

% Normal curve of peak amplitude a, center velocity v0 and width c
% f(v)   = ( a / ( sqrt(2 pi) c )) exp( -0.5 * (v-v0)^2 c^-2 )
% df/da  = ( 1 / ( sqrt(2 pi) c )) exp( -0.5 * (v-v0)^2 c^-2 )
% df/dv0 = ( a / ( sqrt(2 pi) c )) ((v - v0 )/(c^2)) exp( -0.5 * (v-v0)^2 c^-2 )
% df/dc =  ( a / ( sqrt(2 pi) c^2 )) (((v - v0)^2/(c^2))-1) exp( -0.5 * (v-v0)^2 c^-2 )
% 3 model parameters per Normal curve, (a, v0, c)

% estimate of backgroind level
A = max(d);
```

```
rtp = sqrt(2*pi);

% number of peaks determined by clicking on graph
disp('click on the bottom each peak');
```

```
click on the bottom each peak
```

```
disp('    click to the left of zero when done');
```

```
    click to the left of zero when done
```

```
disp(' ');
```

```


% input peaks
MAXPEAKS=100;
a = zeros(MAXPEAKS,1);
v0 = zeros(MAXPEAKS,1);
c = zeros(MAXPEAKS,1);
K=0;
for k = [1:20]
    p = ginput(1);
    if( p(1) < 0 )
        break;
    end
    K=K+1;
    a(K) = p(2)-A;
    v0(K)=p(1);
    c(K)=0.1;
end

% truncate vectors
a = a(1:K);
v0 = v0(1:K);
c = c(1:K);

% model parameters
M=K*3;
m = [a', v0', c']';

for iter=[1:10]

    % predicted data
    dpre = A*ones(N,1);
    for i = [1:K]
        temp = ((v-m(K+i)).^2+m(2*K+i)^2);
        dpre = dpre + m(i)*(m(2*K+i)^2)./temp;
    end

    % data kernel
    G=zeros(N,M);
    for i = [1:K]
        temp = ((v-m(K+i)).^2+m(2*K+i)^2);
        G(:,i) = (m(2*K+i)^2)./temp; % d/da
        G(:,K+i) = 2*m(i)*(m(2*K+i)^2)*(v-m(K+i))./(temp.^2);  % d/dv0
        G(:,2*K+i) = 2*m(i)*m(2*K+i)./temp - 2*m(i)*m(2*K+i)^3./(temp.^2);  % d/dc
    end
```

```
    % deviations in data
    dd = d - dpre;
    E = dd'*dd;

    % updated model
    epsilon=0.001;
    dm = (G'*G+epsilon*eye(M))\(G'*dd);
    mold=m;
    m = m+dm;

    % output parameters, mostly for debugging purposes
    verbose=0;
    if( verbose )
        fprintf('iteration %d     A    v0    c\n', iter);
        for i=[1:K]
            fprintf('%d %f %f %f\n', i, m(i), m(K+i), m(2*K+i));
        end
        disp(sprintf('total error %f', E));
        fprintf('\n');
    end

end

% plot
dpre2 = A*ones(N,1);
for i = [1:K]
    temp = ((v-m(K+i)).^2+m(2*K+i)^2);
    dpre2 = dpre2 + m(i)*(m(2*K+i)^2)./temp;
end
dd = d - dpre2;
E = dd'*dd;
mlorentzian=m;
Elorentzian=E;
nulorentzian = (N-M);
plot(v,dpre2,'b-','LineWidth',3);

% output lorentzian results for debugginf prurposes
fprintf('Lorentzian parameters\n', iter);
```

```
Lorentzian parameters
```

```
for i=[1:K]
    fprintf('%d %f %f %f\n', i, m(i), m(K+i), m(2*K+i));
end
```

```
1 -0.102641 1.758991 0.406972
2 -0.295403 3.104721 0.114714
3 -0.110400 3.617974 0.194357
4 -0.310498 4.274461 0.129157
5 -0.234739 5.454255 0.132738
6 -0.190866 6.343239 0.115598
7 -0.099214 6.845976 0.156691
8 -0.312464 7.541285 0.126499
9 -0.335230 8.733409 0.157522
10 -0.124273 10.480350 0.332173
```

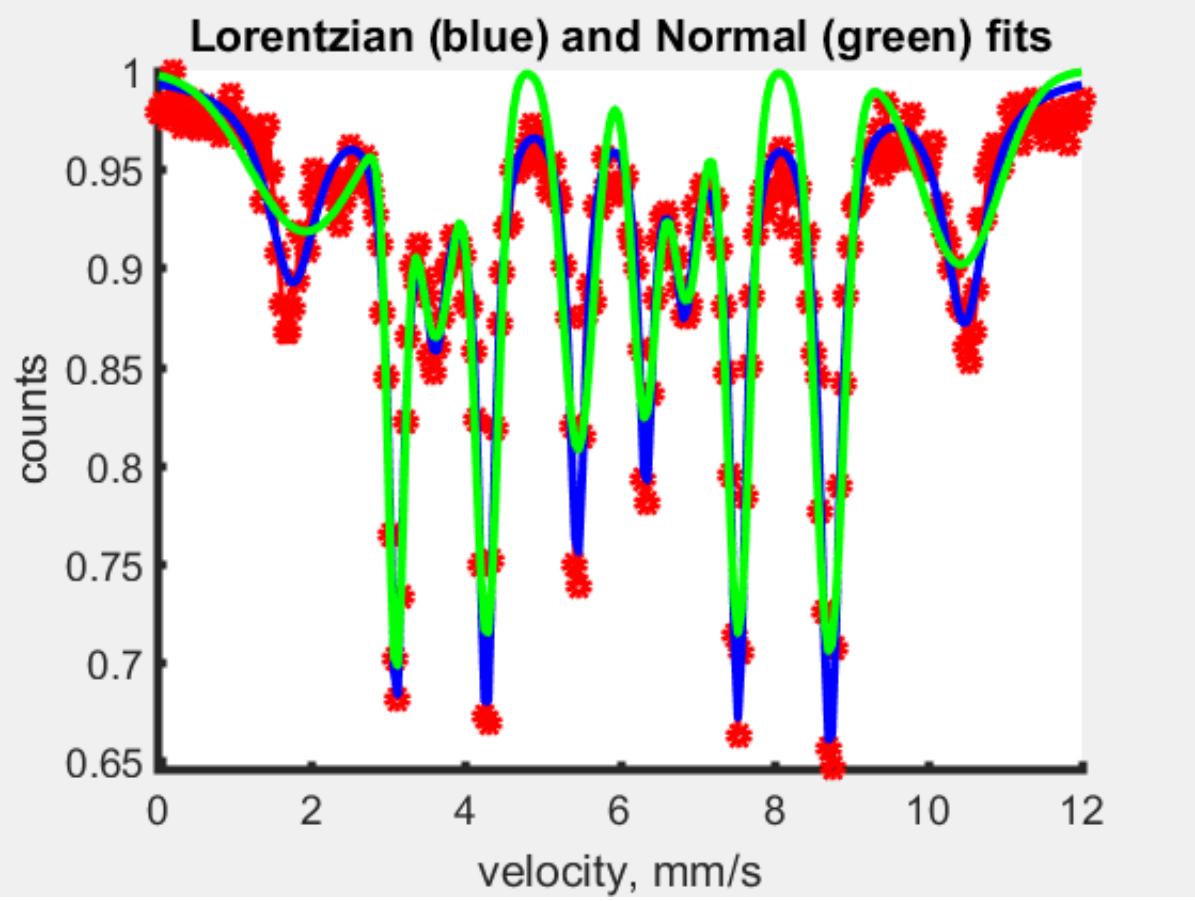
```
disp(sprintf('total error %f', E));
```

```
total error 0.038502
```

```
fprintf('\n');
```

```
% Normal Curve

% model parameters
M=K*3;
m = [a'.*(rtp.*c'), v0', c']';

for iter=[1:10]

    % predicted data
    dpre = A*ones(N,1);
    for i = [1:K]
        temp = exp(-0.5*(v-m(K+i)).^2/(m(2*K+i)^2));
        dpre = dpre + (m(i)/(rtp*m(2*K+i)))*temp;
    end

    % data kernel
    G=zeros(N,M);
    for i = [1:K]
        temp = exp(-0.5*((v-m(K+i)).^2)/(m(2*K+i)^2));
        G(:,i) = (1/(rtp*m(K*2+i)))*temp; % d/da
        G(:,K+i) = (m(i)/(rtp*m(2*K+i))) .* ((v-m(K+i))/(m(2*K+i)^2)) .* temp;  % d/dv0
        G(:,2*K+i) =  (m(i)/(rtp*m(2*K+i)^2)) .* (((v-m(K+i)).^2/(m(2*K+i)^2))-1) .* temp;  %
    end

    % deviations in data
    dd = d - dpre;
    E = dd'*dd;

    % updated model via damped least squares
    epsilon=0.001;
    dm = (G'*G+epsilon*eye(M))\(G'*dd);
    mold=m;
    m = m+dm;

    % output parameters, mostly for debugging purposes
    verbose=0;
    if( verbose )
        fprintf('iteration %d     A    v0    c\n', iter);
        for i=[1:K]
            fprintf('%d %f %f %f\n', i, m(i), m(K+i), m(2*K+i));
        end
        disp(sprintf('total error %f', E));
        fprintf('\n');
    end

end % end iterations

% plot results
dpre2 = A*ones(N,1);
for i = [1:K]
    temp = exp(-0.5*(v-m(K+i)).^2/(m(2*K+i)^2));
    dpre2 = dpre2 + m(i)/(rtp*m(2*K+i))*temp;
```

```
end
dd = d - dpre2;
E = dd'*dd;
mnormal=m;
Enormal=E;
nunormal = (N-M);
title('Lorentzian (blue) and Normal (green) fits');
plot(v,dpre2,'g-','LineWidth',3);
```




```
% Figure 12.16 Example of fitting the sum of 10 Lorentzian (blue) or Normal (green) curves
% to Mossbauer spectroscopic data (red). The Lorentzian curves are better able to fit the
% shape of the curve, with the ratio of estimated variances being about 4. An F-test indicates
% that a null hypothesis that any difference between the two fits can be ascribed to random
% variation can be rejected to better than 99.99%. Data courtesy of NASA and the University
% of Mainz. MatLab script gda12_12.

% output normal results
fprintf('Normal parameters\n', iter);
```

```
Normal parameters
```

```
for i=[1:K]
    fprintf('%d %f %f %f\n', i, m(i), m(K+i), m(2*K+i));
end
```

```
1 -0.148066 1.928948 0.723157
2 -0.071843 3.101362 0.104889
3 -0.077212 3.624236 0.236312
4 -0.106924 4.279725 0.149943
```

```
5 -0.097931 5.462225 0.202804
6 -0.067920 6.326441 0.154575
7 -0.054530 6.870756 0.185764
8 -0.107801 7.543076 0.150618
9 -0.140024 8.725692 0.189404
10 -0.124185 10.431323 0.503627
```

```
disp(sprintf('total error %f', E));
```

```
total error 0.162947
```

```
fprintf('\n');
```

```
F = (Enormal/nunormal)/(Elorentzian/nulorentzian);
disp(sprintf('Fest = E_normal/E_lorentzian: %f', F));
```

```
Fest = E_normal/E_lorentzian: 4.232136
```

```
if( F<1 )
    F=1/F;
end
p = 1-(fcdf(F,nunormal,nulorentzian)-fcdf(1/F,nunormal,nulorentzian));
disp(sprintf('P(F<=1/Fest||F>=Fest) = %f', p));
```

```
P(F<=1/Fest||F>=Fest) = 0.000000
```