```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% This program is written by Erheng Wang. The copy right belongs the
% Dynamic Photo-Mechanics Laboratory.
%
% Code has been modified by Payam Fahr to add reference lines for load area
% based on muzzle configuration. July 2013
%
% This program is for calculating the deformation energy of the gas,
% the momentum and the kinetic energy of the specimen in a shock tube
% experiment.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

clear all;
close all;
clc;
format long;

disp('%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%');
disp('This program is for calculating the deformation energy of the gas,');
disp('the momentum and the kinetic energy of the specimen in a shock tube');
disp('experiment.');
disp(' ');
disp('Please follow the instruction.');
disp('%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%');
disp(' ');

disp('%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%');
disp('First Step: load the reflection pressure profile.');
disp(' ');
disp('The reflection pressure profile should have following form:');
disp('0.00001  124');
disp('0.00002  160');
disp('0.00003  215');
disp('0.00004  260');
disp('0.00005  302');
disp('The first column is time. And second column is pressure.');
disp('You need to input the unit of time and pressure. Please check the unit
carefully.');
disp('Please follow the instruction.');
disp(' ');

eval(['load ref_sp.dat;'])
disp(' ');

disp('We have following time unit:');
disp('1. second');
disp('2. millisecond');
disp('3. microsecond');
unit_judge=true;
time_unit=0; % this number can be any integer except 1, 2 and 3.
while unit_judge==true
    time_unit=input('Please choose the unit you use (input the No. before the
unit):');
    if time_unit==1
        disp(' ');
        disp('The time unit you use is second;');
```

```
        eval(['ref_sp(:,1)=ref_sp(:,1);'])
        unit_judge=false;
    elseif time_unit==2
        disp(' ');
        disp('The time unit you use is millisecond;');
        eval(['ref_sp(:,1)=ref_sp(:,1)./1000;'])
        unit_judge=false;
    elseif time_unit==3
        disp(' ');
        disp('The time unit you use is microsecond;');
        eval(['ref_sp(:,1)=ref_sp(:,1)./1000000;'])
        unit_judge=false;
    else
        disp('Wrong input. Please choose again.');
        unit_judge=true;
    end
end
disp(' ');

disp('We have following pressure unit:');
disp('1. psi');
disp('2. MPa');
disp('3. Pa');
unit_judge=true;
pressure_unit=0; % this number can be any integer except 1, 2 and 3.
while unit_judge==true
    pressure_unit=input('Please choose the unit you use (input the No. before
the unit):');
    if pressure_unit==1
        disp(' ');
        disp('The pressure unit you use is psi;');
        eval(['ref_sp(:,2)=ref_sp(:,2);'])
        unit_judge=false;
    elseif pressure_unit==2
        disp(' ');
        disp('The pressure unit you use is MPa;');
        eval(['ref_sp(:,2)=ref_sp(:,2).*1000000./6894.7;'])
        unit_judge=false;
    elseif pressure_unit==3
        disp(' ');
        disp('The pressure unit you use is Pa;');
        eval(['ref_sp(:,2)=ref_sp(:,2)./6894.7;'])
        unit_judge=false;
    else
        disp('Wrong input. Please choose again.');
        unit_judge=true;
    end
end
disp(' ');

disp('The pressure data has been resaved into variable ref_sp.');
disp('There are two columns in ref_sp. The first column is time and unit is s
(second).');
disp('The second column is pressure and unit is psi.');
disp('First Step end');
disp('%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%');
```

```
disp(' ');

disp('%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%');
disp('Second Step: load the time series of the images.');
disp(' ');

disp('You have three ways to load the time series of the images.');
disp('1. The time between two frames is same.');
disp('   You can input total number of frames and time between two frames.')
disp('   The code will generate the time series automatically.');
disp(' ');

disp('2. The time between two frames is not same.');
disp('   You can input total number of frames and input time between two
frames frame by frame.')
disp(' ');

disp('3. The time between two frames is not same.');
disp('   And you have saved the time series into one data file.')
disp('   Then you can just load that time series data file.');
disp(' ');

time_series_judge=true;
time_series=0; % this number can be any integer except 1, 2 and 3.
while time_series_judge==true
    time_series=input('Please choose which method you want to use (input the
No. before the method):');
    if time_series==1
        frames=input('Please input the total number of frames for
calculating(integer): '); % the number of images for calculating
        frame_time=input('Please input the time between two frames (unit:
microsecond): ')/1000000;
        for i=1:frames
            t_frame(i,1)=(i-1)*frame_time;
        end
        time_series_judge=false;
    elseif time_series==2
        frames=input('Please input the total number of frames for
calculating(integer): '); % the number of images for calculating
        sum_time=0;
        for i=1:frames
            disp('recent frame is')
            i
            disp('frame.')
            disp('Please input 0 when i=1;');
            sum_time=input('Please input the time between this frame and one
frame before(unit: \mus): ')/1000000+sum_time;
            t_frame(i,1)=sum_time;
        end
        time_series_judge=false;
    elseif time_series==3
        time_series_name=input('Please input the filename of the time serise
(without extension):','s')
        time_series_extension=input('Please input the extension of the time
serise:','s')
        eval(['load ',time_series_name,'.',time_series_extension,';'])
```

```
        eval(['t_frame=',time_series_name,';'])
        time_series_judge=false;
    else
        disp('Wrong input. Please choose again.');
        time_series_judge=true;
    end
end
disp(' ');

disp('Second Step end');
disp('%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%');
disp(' ' );


disp('%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%');
disp('Third Step: length calibration.');
disp(' ');
disp('you can choose any image for length calibration.');
disp('On the image, you need to choose two points and the vertical distance
between these two points will be used to calibrate the length');
disp('Therefore, you need to know one real vertical scale in the image.');
disp('For example:');
disp('the span of the supports is 6 inches');
disp('the outer diameter of the shock tube is 5 inches');
disp('');
disp('The process will repeat three times. Thus, totally you will pick six
times');
disp('Please follow the instruction.');
disp(' ');

disp('Please enter image filename for length calibration:');
I=input('(for example: calibration.jpg) ','s');
Judge1='n';
while Judge1=='n'
    % load the jpg file
    imshow(I);
    hold on

    xlabel('Length Calculation')
    title('Please pick first point for calibration');
    [xc(1),yc(1)] = ginput(1);
    title('Please pick second point for calibration');
    [xc(2),yc(2)] = ginput(1);
    title('Please pick third point for calibration');
    [xc(3),yc(3)] = ginput(1);
    title('Please pick fourth point for calibration');
    [xc(4),yc(4)] = ginput(1);
    title('Please pick fifth point for calibration');
    [xc(5),yc(5)] = ginput(1);
    title('Please pick sixth point for calibration');
    [xc(6),yc(6)] = ginput(1);

    title('Please go to the matlab main window and input the real distance');
    % average point between two calibration points
    Y(1) = abs(yc(1)-yc(2));
    Y(2) = abs(yc(3)-yc(4));
```

```
    Y(3) = abs(yc(5)-yc(6));
    measured = mean(Y);

    % determin the middle position of the shock tube
    ym(1)=(yc(1)+yc(2))/2;
    ym(2)=(yc(3)+yc(4))/2;
    ym(3)=(yc(5)+yc(6))/2;
    midy=mean(ym);


    % real distance between two calibration points. unit: m
    true = input('Please input the real distance between two points you
choose (in): ')*0.0254;

    % The transfor from the pixes to distance
    %~pf 7/18/2013
    scale = measured/true;
    innerdia = (.75*0.0254*scale) %Select this for SMALL MUZZLE
    %innerdia = (1.5*0.0254*scale) % Select this for LARGE MUZZLE
     %~pf 7/18/2013

    xlabel('')
    title('Length Calculation End');

    Judge1=input('Is calibration OK? (y/n)','s');

    close all;
end

disp('Third Step end');
disp('%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%');
disp(' ' );

disp('%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%');
disp('Fourth Step: real measurement.');
disp(' ');
disp('you need to measure the deformation shape of front face for every
image.');
disp('For each image, you need to choose seven points on the front face.');
disp('There will be a symmetric line on the image.');
disp('It is better to choose these points symmetric to this line.');
disp('Please follow the instruction.');
disp(' ');

frames = input('Please input the total number of frames for
calculating(integer): '); % the number of pictures for calculating
%Tube_d=0.0762;    % the real scale of the large muzzle diameter of shock
tube
Tube_d=0.0762/2    % the real scale of the large muzzle diameter of shock
tube

% point_number=input('How many points will you choose for face shape fitting?
(integer) ');
```

```
point_number=7;

x=zeros(point_number,frames);
y=zeros(point_number,frames);
for i = 1:frames

    disp(' ');
    if i==1
        disp('Please enter the first image filename for measurement:');
        I=input('(for example: measure_image.jpg) ','s');
    else
        I=input('Please enter next image filename for measurement: ','s');
    end


    % Simulate the Front Surface Shape with Cubic Spline interpolation method
    Judge2='n';
    while Judge2=='n'
        imshow(I);
        hold on;
        xlabel('Simulate the Front Surface');

        %~pf 7/18/2013
        plot ([0;1200],[midy;midy],'r'), hold on
        plot ([0;1200],[midy+innerdia;midy+innerdia],'r'), hold on
        plot ([0;1200],[midy-innerdia;midy-innerdia],'r'), hold on
        % Given the scale based on picked points and length input, this
        % will project horizontal markers on the centerline of specimen and
        % the upper and lower boundaries of the inner diameter of the
        % muzzle depending on which has been selected above (large or
        % small) This will aid in keeping all profile measurements about
        % the loading area
        %~pf 7/18/2013

        title('Please pick the top point for shape calculation');
        [x(1,i),y(1,i)] = ginput(1);
        plot(x(1,i),y(1,i),'go'),hold on;

        title('Please pick the bottom point for shape calculation');
        [x(7,i),y(7,i)] = ginput(1);
        plot(x(7,i),y(7,i),'go'),hold on;

        for j=1:point_number
            yfl(j,1)=(y(1,i)+y(7,i))/2-abs(y(1,i)-y(7,i))/2+(j-1)*abs(y(1,i)-
y(7,i))/(point_number-1);
        end
        x1=linspace(0,1200);
        for j=1:point_number
            clear y1;
            y1=linspace(yfl(j),yfl(j));
            plot(x1,y1,'c'), hold on;
        end
%         if i==1
%         else
%             plot(xx(:,(i-1)),yy(:,(i-1)),'y','linewidth',0.25), hold on;
```

```
%             legend('symmetric line','previous shape');
%         end

        % choose seven points for the surface shape fit

        title('Please pick the second point for shape calculation');
        [x(2,i),y(2,i)] = ginput(1);
        plot(x(2,i),y(2,i),'go'),hold on;

        title('Please pick the third point for shape calculation');
        [x(3,i),y(3,i)] = ginput(1);
        plot(x(3,i),y(3,i),'go'),hold on;

        title('Please pick the fourth point for shape calculation');
        [x(4,i),y(4,i)] = ginput(1);
        plot(x(4,i),y(4,i),'go'),hold on;

        title('Please pick the fifth point for shape calculation');
        [x(5,i),y(5,i)] = ginput(1);
        plot(x(5,i),y(5,i),'go'),hold on;

        title('Please pick the sixth point for shape calculation');
        [x(6,i),y(6,i)] = ginput(1);
        plot(x(6,i),y(6,i),'go'),hold on;

%       d=Tube_d*scale; % the pixes scale of the diameter of shock tube
        dD=abs(y(1,i)-y(7,i))/100;
        dd=Tube_d*scale/100;
        for m=1:101
            yy(m,i)=(y(1,i)+y(7,i))/2-abs(y(1,i)-y(7,i))/2+(m-1)*dD; % the
range of shock applied
            yys(m,i)=midy-(Tube_d*scale/2)+(m-1)*dd; % the range of shock
applied
        end
        xx(:,i)=spline(y(:,i),x(:,i),yy(:,i));   % cubic spline data
interpolation
        xxs(:,i)=spline(y(:,i),x(:,i),yys(:,i));   % cubic spline data
interpolation
        plot(xx(:,i),yy(:,i),'r'), hold on;
        plot(xxs(:,i),yys(:,i),'g'), hold on;

        title('Press any key to continue');
        pause;

        Judge2=input('Is that curve OK? (y/n)','s');

        close all;
    end
end

disp('Fourth Step end');
disp('%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%');
disp(' ' );
```

```
disp('%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%');
disp('Fifth Step: real measurement.');
disp(' ');
disp('The measured position of the specimen will be used to calculate the
force displacement, ');
disp('velocity of the specimen.');
disp('Most parts of this step are automatic.');
disp('Please follow the instruction.');
disp(' ');

% calculate the surface position of every frame
xf=xx(:,1);
yf=yy(:,1);
xfs=xxs(:,1);
yfs=yys(:,1);

specimen_M=input('Please input the total mass of the specimen (g):   ')/1000;
sumimpulse(:,1)=t_frame;
sumyimpulse(:,1)=t_frame;
sumKE(:,1)=t_frame;
for i = 1:frames
    sumimpulse(i,2)=0;
    sumyimpulse(i,2)=0;
    sumKE(i,2)=0;
    for j=1:101
        xd(j,i)=(xx(j,i)-xf(i))/scale;
        yd(j,i)=(yy(j,i)-yf(i))/scale;
        xds(j,i)=(xxs(j,i)-xfs(i))/scale;
        yds(j,i)=(yys(j,i)-yfs(i))/scale;
        if i==1
            xv(j,i)=0;
            yv(j,i)=0;
            average_v(j,i)=0;
        else
            xv(j,i)=((xx(j,i-1)-xx(j,i))/scale)/frame_time;
            yv(j,i)=((yy(j,i-1)-yy(j,i))/scale)/frame_time;
            average_v(j,i)=sqrt(xv(j,i)^2+yv(j,i)^2);
        end
        sumimpulse(i,2)=sumimpulse(i,2)+(specimen_M/101)*xv(j,i);
        sumyimpulse(i,2)=sumyimpulse(i,2)+(specimen_M/101)*yv(j,i);
        sumKE(i,2)=sumKE(i,2)+((specimen_M/101)*average_v(j,i)^2)/2;
    end
end

% calculate the deflection for every points of every frame

for j=1:101;
    for i=1:frames
        xdd(j,i)=abs(xd(j,i)-xd(j,1));
        ydd(j,i)=abs(yd(j,i)-yd(j,1));
        xdds(j,i)=abs(xds(j,i)-xds(j,1));
        ydds(j,i)=abs(yds(j,i)-yds(j,1));
    end
end

% figure(2)
```

```
% for i=1:frames;
%     plot(-xdd(:,i),yy(:,i)/scale,'k'),hold on
%     plot(-xdd(:,2),yy(:,1)/scale,'k--'),hold on
%     plot(-xdd(:,3),yy(:,1)/scale,'r'),hold on
%     plot(-xdd(:,4),yy(:,1)/scale,'r--'),hold on
%     plot(-xdd(:,5),yy(:,1)/scale,'g'),hold on
%     plot(-xdd(:,6),yy(:,1)/scale,'g--'),hold on
%     plot(-xdd(:,7),yy(:,1)/scale,'b'),hold on
%     plot(-xdd(:,8),yy(:,1)/scale,'b--'),hold on
%     plot(-xdd(:,9),yy(:,1)/scale,'m'),hold on
% end
% plot(-xdd(:,10),yy(:,1)/scale,'m--'),hold on
% plot(-xdd(:,11),yy(:,1)/scale,'y'),hold on
% plot(-xdd(:,12),yy(:,1)/scale,'c'),hold on
% plot(-xdd(:,13),yy(:,1)/scale,'c--'),hold on
% xlabel('unit: m');
% ylabel('unit: m');
% title('Deflection Sketch');
% axis tight

% choose the biggest time to normalize data

for i=1:10000
    t(i,1)=(i-1)*2E-6;
    if (t(i,1)>=t_frame(frames,1))
        break;
    end
end

ref=spline(ref_sp(:,1),ref_sp(:,2),t);

% normalize the time for deflection data
for j=1:101
    De(:,j)=spline(t_frame,xdd(j,:),t);
    Des(:,j)=spline(t_frame,xdds(j,:),t);
end

S_x_impulse(:,1)=t;
S_x_impulse(:,2)=spline(sumimpulse(:,1),sumimpulse(:,2),t);


%~pf 7/18/2013
%S_y_impulse(:,1)=t;
%S_y_impulse(:,2)=spline(sumyimpulse(:,2),sumyimpulse(:,2),t);
%This is causing error possibly because there is no deviation in y
%direction if points are selected on the projected horizontal points
%~pf 7/18/2013

S_KE(:,1)=t;
S_KE(:,2)=spline(sumKE(:,1),sumKE(:,2),t);

% calculate the energy increase between every two closed frame
n0=length(t);
egy(1)=0;
delta_d=(Tube_d*0.99)/99;
```

```
for i=2:n0
    A(i)=0;
    for j=1:100
        B(j)=((ref(i)+ref(i-1))*6894.7)*(Des(i,j)-Des(i-
1,j))*delta_d*(sqrt((Tube_d/2)^2-((Tube_d/2)-j*delta_d)^2)+sqrt((Tube_d/2)^2-
((Tube_d/2)-(j-1)*delta_d)^2))/2;
        A(i)=B(j)+A(i);
    end
    egy(i)=A(i);
end

% calculate the energy increase between every frame and initial frame
for i=1:n0
    A1=0;
    for j=1:i
        B1=egy(j);
        A1=A1+B1;
    end
    energy(i,1)=A1;
end

DFLE(:,1)=t;
DFLE(:,2)=energy(:,1);

Center(:,1)=t;
Center(:,2)=De(:,51);

figure(1)
plot(t,De(:,51),'r','linewidth',3),hold on
ylabel('Deflection (m)');
xlabel('Time (s)');
grid on;
title('Maxi Deflection-Time Curve(Middle Point)');

figure(2),plot(t,S_x_impulse(:,2),'r','linewidth',3);
xlabel('Time (s)');
ylabel('Horizontal momentum (kgm/s or Ns)');
% axis tight;
grid on;

%~pf 7/18/2013
%figure(3),plot(t,S_y_impulse(:,2),'r','linewidth',3);
%xlabel('Time (s)');
%ylabel('Vertical momentum (kgm/s or Ns)');
% axis tight;
%grid on;
%~pf 7/18/2013

figure(3),plot(t,S_KE(:,2),'r','linewidth',3);
xlabel('Time (s)');
ylabel('Kinetic Energy (J)');
% axis tight;
grid on;

figure(4),plot(t,energy(:,1),'r','linewidth',3);
```

```
xlabel('Time (s)');
ylabel('Deformation Energy (J)');
% axis tight;
grid on;

disp(' ');
disp('After runing this code, there will be Five dat files with following
name: ');
disp('filename_deformation_E.DAT             Deformation energy');
disp('filename_x_momentum.DAT                Horizontal momentum');
%~pf 7/18/2013
%disp('filename_y_momentum.DAT                Vertical momentum');
%~pf 7/18/2013
disp('filename_Kinetic_E.DAT                 Kinetic energy');
disp('filename_center_displacement.DAT        Center displacement');
disp(' ');
disp('All of other data have been save into the file entitled
filename_specimen.mat.');

filename=input('Now, please input the filename you want to save the final
data into: ','s');

eval(['save ',filename,'_specimen.mat'])
eval(['save ',filename,'_deformation_E.DAT',' DFLE',' /ascii'])
eval(['save ',filename,'_x_momentum.DAT',' S_x_impulse',' /ascii'])
%~pf 7/18/2013
%eval(['save ',filename,'_y_momentum.DAT',' S_y_impulse',' /ascii'])
%~pf 7/18/2013
eval(['save ',filename,'_Kinetic_E.DAT',' S_KE',' /ascii'])
eval(['save ',filename,'_center_displacement.DAT',' Center',' /ascii'])

disp('Fifth Step end');
disp('%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%');
disp(' ' );
```