```matlab
% CMSC/AMSC 460 Fall 2007
% Homework 7
%
% Purpose: Practice in different computational methods like: ODE,
%          spline interpolation, and solution of non-linear system
% We want to trace a sound ray in ocean water, z(x) is the depth of
% the ray when it is a horizontal distance x from the source.
%
% Sima Taheri, 4 Dec. 2007
%
% Input:
%       c: The speed of sound (in ft/sec) at some depth values z.
%       z: Given depth values
%       z0: Initial depth of the ray
%       Theta0 : Initial angle between the tangent to z(x) and the
%       horizontal axis.
%                        tan(theta) = dz/dx
%       a: Constant value in Snell's Law
%                        a = cos(theta0)/c(z0)
%
% A sound source at a depth of z0=2000 ft transmits to
% a receiver xhat=24 miles away, at a depth of 3000 ft
% We want to have,
%
% Output
%      Part(a): Plot of z(x) for x\in[0,24mi] when theta0=5.4 degree
%      Part(b): A table of values of z(xhat)-3000 for theta0 in the range
%               -10 to 10 degrees when xhat=24mi.
%      Part(c): 4 rays with angles between -10 and 10 degrees that pass
%               through the receiver at xhat=24mi.
%
% Matlab Functions: ODE45, fzero
clc
clear all

%% Part (a)

global pp
% Given values for c(z)
z = [0:500:4000, 5000:1000:12000]';
c = [5042 4995 4948 4887 4868 4863 4865 4869 4875 ...
     4875 4887 4905 4918 4933 4949 4973 4991]';
% Spline coefficients
pp = spline(z,c);

% Radian = pi*Degree/180;
Theta0 = pi*5.4/180;
% to have the second order derivative of z we define the
% initial condition [z0 ; dz/dx(0)=tan(Theta0)]
[xout,zout] = ode45(@zdoublep,[0,24*6076],[2000;tan(Theta0)]);
plot (xout,zout(:,1))
grid
title ('z(x)')
xlabel ('x (feet), horizontal distance to the sound source')
ylabel('z (feet), depth under the ocean surface')
```

```matlab
%% Part (b)
k=1;
out = zeros(1,21);
for theta = -10:10
    out(k) = depth(theta);
    k = k+1;
end;
% Table
theta = (-10:10);
disp ('Theta        z(xhat)-3000');
disp ('-------------------------');
disp(sprintf(' %2d       %5.3f \n',[theta;out]))

%% Part (c)
% Rays that pass through the receiver have z(xhat)-3000=0
% So, we want to find 4 values for theta for which depth function
% gives zero output
%
% Find appropriate starting values for fzero.
% These starting values correspond to zero crossings of out
temp = out>0;
init = theta(temp(1:end-1)-temp(2:end)~=0);

Theta = zeros(size(init));
for i=1:length(init)
    Theta(i) = fzero(@depth,init(i));
end

% Plot those sound rays
for i=1:length(init)
    [xout,zout] = ode45(@zdoublep,[0,24*6076],[2000;tan(pi*Theta(i)/180)]);
    plot (xout,zout(:,1));
    hold on
end
Theta
grid
title ('z(x)')
xlabel ('x (feet), horizontal distance to the sound source')
ylabel('z (feet), depth under the ocean surface')
```

```matlab
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function out = depth(theta)
% out = depth(theta)
% This function traces the sound ray transmitted from a
% sound source at a depth of z0 = 2000 ft to a receiver xhat =24 miles away,
% with the initial angle theta
% then returns the value out = z(xhat)-3000.

xhat = 24*6076; %feet
[xout,zout] = ode45(@zdoublep,[0,xhat],[2000;tan(pi*theta/180)]);
out = zout(end,1)-3000;



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function out = zdoublep (x,y)
% out = zdoublep (x,y)
% To have the second order derivative of z we define
% a new variable y = [z;dz/dx];
% Therefore the output will be out=[dz/dx,d2z/dx2]
% Matlab Functions: Spline, Myppval
global pp

z = y(1);
dzdx = y(2);

% Constant a
a = (cos(pi*5.4/180)/4868);

% Evaluate c(z) and c'(z)
% To have the spline interpolation of c'(z), we use the coefficient of
% cubic spline to build the quadratic polynomial of c'(z)
% In each interval [xl,xu], the piecewise cubic spline interpolation
% computes the coefficients [a0,a1,a2,a3] of
%                   c(z)=a0+a1(x-x1)+a2(x-xl)^2+a3(x-x3)^3
% So we can compute c'(z) as
%                   c'(z)=a1+2*a2(x-xl)+3*a3(x-x3)^2
% we modify the Matlab ppval function
% to return both function value and derivative.

[cz,czp] = Myppval(pp,z);

% Output
out(1) = dzdx;
out(2) = -czp./(a^2*cz^3);
% output must be a vector
out = out';
```

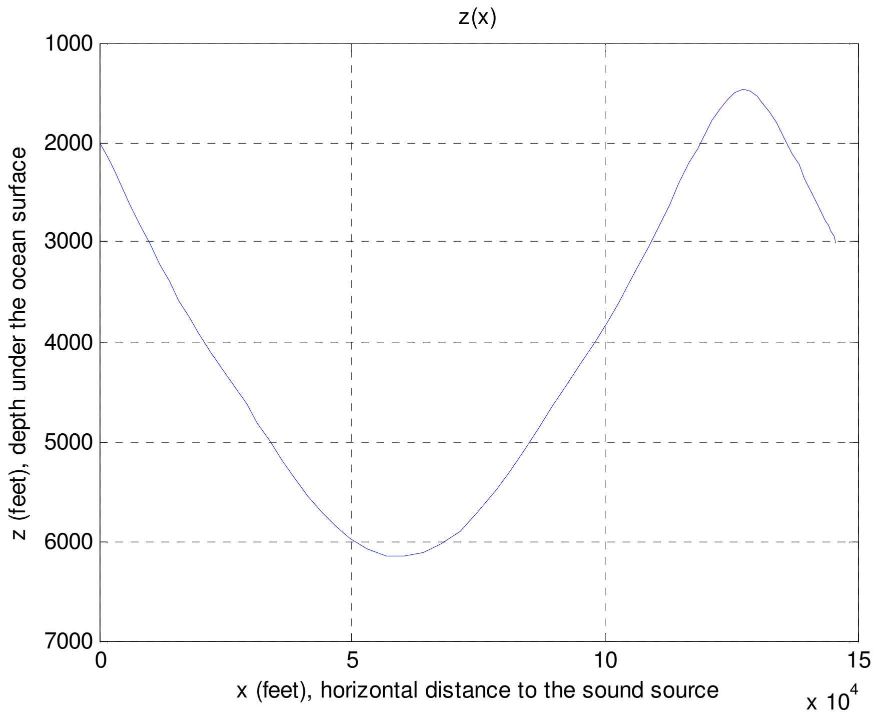
```matlab
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [v,vp]=Myppval(pp,xx)
% Modifications have been separated by stars

if isstruct(xx) % we assume that ppval(xx,pp) was used
   temp = xx; xx = pp; pp = temp;
end


ndimsxx = ndims(xx);
isvectorxx = isvector(xx) && ~isscalar(xx);
%  obtain the row vector xs equivalent to XX
sizexx = size(xx); lx = numel(xx); xs = reshape(xx,1,lx);
%  if XX is row vector, suppress its first dimension
if length(sizexx)==2&&sizexx(1)==1, sizexx(1) = []; end

% if necessary, sort xs
ixexist = false;
if any(diff(xs)<0)
    [xs,ix] = sort(xs);
    ixexist = true;
end

% take apart PP
[b,c,l,k,dd]=unmkpp(pp);

% for each data point, compute its breakpoint interval
[ignored,index] = sort([b(1:l) xs]);
index = reshape(find(index>l),1,lx)-(1:lx);
index(index<1) = 1;

% now go to local coordinates ...
xs = xs-b(index);

d = prod(dd);
if d>1 % ... replicate xs and index in case PP is vector-valued ...
   xs = reshape(xs(ones(d,1),:),1,d*lx);
   index = d*index; temp = (-d:-1).';
   index = reshape(1+index(ones(d,1),:)+temp(:,ones(1,lx)), d*lx, 1 );
else
   if length(sizexx)>1, dd = []; else dd = 1; end
end

% ... and apply nested multiplication:
v = c(index,1);
for i=2:k
   v = xs(:).*v + c(index,i);
end
```

```
%****************************
%****************************
% c'(z)=a1+2*a2(x-xl)+3*a3(x-x3)^2
vp =(k-1)*c(index,1);
for i=2:k-1
   vp = xs(:).*vp + (k-i)*c(index,i);
end
%****************************
%****************************

v = reshape(v,d,lx);
vp = reshape(vp,d,lx);
if ixexist, v(:,ix) = v; end
v = reshape(v,[dd,sizexx]);
vp = reshape(vp,[dd,sizexx]);

if isfield(pp,'orient') && strcmp(pp.orient,'first')
    % spline orientation is returns size(yi) == [d1 ... dk m1 ... mj]
    % but the interp1 usage prefers size(yi) == [m1 ... mj d1 ... dk]
    if ~(isempty(dd) || (isscalar(dd) && dd == 1))
        % The function is non-scalar valued
        if isvectorxx
            permVec = [ndims(v) 1:(ndims(v)-1)];
        else
            permVec = [(ndims(v)-ndimsxx+1) : ndims(v) 1:(ndims(v)-ndimsxx)];
        end
        v = permute(v,permVec);
    end
end
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% An alternative to modify ppval for computing c'(z) is to use
% Matlab's function unmkpp and mkpp to give the coefficients
% so that we can construct the derivative.
pp = spline(z,c);
[breaks,coefs] = unmkpp(pp);
Ncoefs(:,1) = 3*coefs(:,1);
Ncoefs(:,2) = 2*coefs(:,2);
Ncoefs(:,3) = coefs(:,3);
Npp = mkpp(breaks,Ncoefs);
czp = ppval(Npp,z);
```

Results:

Part (a):



Part (b):

Theta	z(xhat)-3000
-10	2137.563
-9	639.249
-8	-17.127
-7	-1627.385
-6	-1324.147
-5	-1377.331
-4	515.244
-3	169.843
-2	199.218
-1	112.127
0	269.458
1	418.436
2	652.632
3	928.727
4	-918.275
5	-495.288
6	469.313
7	385.961
8	-1692.916
9	-1820.215
10	-1005.843

Part (c):

Theta (degree) = [-8.0150 -4.2067 3.7734 5.3911 7.2190]