% 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
clear all

%% Part (a)

global pp
% Given values for c(z)
% Given values for c(z)
z = [0:500:4000, 5000:1000:12000]';
c = [5042 4995 4948 4887 4867 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')
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 
',[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')
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
[out, 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-xl)+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';
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
\[ c'(z) = a_1 + 2a_2(x - x_1) + 3a_3(x - x_3)^2 \]

\[ v_p = (k - 1)c(index, 1); \]

\[
\text{for } i = 2:k-1 \\
\quad v_p = xs(:,).*v_p + (k-i)*c(index, i); \\
\text{end}
\]

\[ v = \text{reshape}(v, d, lx); \]
\[ v_p = \text{reshape}(v, d, lx); \]
\[
\text{if } \text{ixexist}, v(:,ix) = v; \text{ end} \\
\]
\[ v = \text{reshape}(v, [d, size(xxx)]); \]
\[ v_p = \text{reshape}(v, [d, size(xxx)]); \]

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

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

Part (a):

<table>
<thead>
<tr>
<th>Theta</th>
<th>$z(\hat{x}) - 3000$</th>
</tr>
</thead>
<tbody>
<tr>
<td>-10</td>
<td>2137.563</td>
</tr>
<tr>
<td>-9</td>
<td>639.249</td>
</tr>
<tr>
<td>-8</td>
<td>-17.127</td>
</tr>
<tr>
<td>-7</td>
<td>-1627.385</td>
</tr>
<tr>
<td>-6</td>
<td>-1324.147</td>
</tr>
<tr>
<td>-5</td>
<td>-1377.331</td>
</tr>
<tr>
<td>-4</td>
<td>515.244</td>
</tr>
<tr>
<td>-3</td>
<td>169.843</td>
</tr>
<tr>
<td>-2</td>
<td>199.218</td>
</tr>
<tr>
<td>-1</td>
<td>112.127</td>
</tr>
<tr>
<td>0</td>
<td>269.458</td>
</tr>
<tr>
<td>1</td>
<td>418.436</td>
</tr>
<tr>
<td>2</td>
<td>652.632</td>
</tr>
<tr>
<td>3</td>
<td>928.727</td>
</tr>
<tr>
<td>4</td>
<td>-918.275</td>
</tr>
<tr>
<td>5</td>
<td>-495.288</td>
</tr>
<tr>
<td>6</td>
<td>-495.288</td>
</tr>
<tr>
<td>7</td>
<td>-1692.916</td>
</tr>
<tr>
<td>8</td>
<td>-1820.215</td>
</tr>
<tr>
<td>9</td>
<td>-1005.843</td>
</tr>
</tbody>
</table>

Part (b):
Part (c):

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