```matlab
function plotDOC(stepValue,milePost,degOfCurv,smoothFactor,stationing, ...
    chordLength,profile,x,y,MP,figName,filePath,fileName,outFile,alpha,trackID,mpScaleFactor)
%
% plotDOC.m - plot the degree of curvature and profile vs mile post reference.
%
%   Input: "CSX GMRS-like Format"
%
%     linear mile post reference
%     degree of curvature
%     track elevationation
%     data source reference name
%     figure name/ title
%
%   Output: plot of track section mile post vs. degree of curvature,
%       track profile, plan view.
%
% Syntax:
%
% **********************************************************************
% Other m-files required:
%
% Subfunctions:
%
% MAT-files required: none
%
% See also:
% ******************************************************************
% Author: Peter J Dailey
% email: daileypj@mac.com
% Last revision: 5-July-2009/10-Feb-2010
% ************************************************************************
%
% Copyright 2010 by Peter J Dailey
%
% Permission to use, copy, modify, distribute, and sell this software and its
% documentation for any purpose is hereby granted without fee, provided that
% the above copyright notice appear in all copies and that both that
% copyright notice and this permission notice appear in supporting
% documentation, and that the name of Peter J Dailey not be used in
% advertising or publicity pertaining to distribution of the software without
% specific, written prior permission. This license includes software written
% for Matlab or any other programming language.
% Peter J Dailey makes no representations about the suitability of this
% software for any purpose.  It is provided "as is" without express or
% implied warranty.
% ************************************************************************
%
% Mean and std dev to clip outliers
% Data mean
mu3      = nanmean(degOfCurv);
medDOC   = nanmedian(degOfCurv);
% Data standard deviation
sigma3   = nanstd(degOfCurv);

%% Degree of Curvature vs. mile post
figure('Name',figName,'NumberTitle','off');
subplot(3,3,1:3);
hold on;
scatter(milePost,degOfCurv,'.');
plot(milePost,smooth(degOfCurv,smoothFactor,'rlowess'),'k','LineWidth',2);grid on;

% Clip x axis to mile posts
minMP    = round(min(milePost));
maxMP    = round(max(milePost));
xlim([minMP maxMP]);
% Set X-axis direction
```

```
if stepValue  == 1
     set(gca,'XDir','normal')
else
     set(gca,'XDir','reverse')
end
% Set a minimum span of +/- 2 degrees
minDOC   = (mu3 - 2*sigma3)-1.00;
maxDOC   = (mu3 + 2*sigma3)+1.00;
% Set a minimum span of +/- 2 degrees
if (maxDOC < 1) || (minDOC > -1)
     maxDOC   = 2;
     minDOC   = -2;
end
% Out of bounds test
if (maxDOC > 8) || (minDOC < -8)
     maxDOC   = 8;
     minDOC   = -8;
end
% Set the Y axis limits
ylim([minDOC maxDOC]);

% Create title, use [ ] syntax to preserve white space
title(strcat([figName ' Horizontal Alinement' trackID ', ' num2str(chordLength) ' ft chord']), ...
    'FontSize',14);
% Create xlabel
sfText   = strcat([', scale factor this mile: ' num2str(mpScaleFactor)]);
xlabel(strcat(['Mile Post' sfText]),'FontSize',12);
% Create ylabel
ylabel('D_{c}','FontSize',14);
%legend1 = legend(strcat(num2str(chordLength),' ft chord'),'smoothed');
%set(legend1,'Location','Best');

hold off;
%Local regression using weighted linear least squares and a 1st degree polynomial model
%Savitzky-Golay filter
% A robust version of 'lowess' that assigns lower weight to outliers in the
% regression. The method assigns zero weight to data outside six mean absolute
% deviations.
% Save smooth data set.
% header: figName,
% data: milePost,smooth(degOfCurv,smoothFactor,'rlowess')



%% Elevation vs mile post
subplot(3,3,4:6);
hold on;
scatter(milePost,profile,'.');
plot(milePost,smooth(profile,smoothFactor,'rlowess'),'k','LineWidth',1);grid on;

% Clip Y axis
% Mean and std dev to clip outliers
% Data mean
muPro         = nanmean(profile);
% Data standard deviation
sigmaPro      = nanstd(profile);
% Use axis limits to clip outliers
minEl         = muPro - 3*sigmaPro;
maxEl         = muPro + 3*sigmaPro;
ylim([minEl maxEl])
% Clip X axis to between mile posts
xlim([minMP maxMP]);
% Set X-axis direction
if stepValue  == 1
     set(gca,'XDir','normal')
else
```

```
     set(gca,'XDir','reverse')
end
% Create title
title(strcat([ 'Profile' trackID ', ' num2str(stationing) ' ft stations']),'FontSize',10);
% Create xlabel? MP label on Dc plot
% xlabel(strcat(['Mile Post']));
% Create ylabel
ylabel({'Elevation, ft'});
% Create legend
%legend2            = legend('HiRail');
%,set(legend2,'Location','Best');
hold off;

%% Plan view
hold on;
subplot(3,3,7);
% translate the coordinates
minX     = min(x);
maxX     = max(x);
minY     = min(y);
maxY     = max(y);
spanX    = floor((maxX-minX)/1000)*1000;
spanY    = floor((maxY-minY)/1000)*1000;
aspRatio = maxX / maxY;
minSpan  = min([spanX spanY]);
tranY    = (floor(minY/1000))*1000;
tranX    = (floor(minX/1000))*1000;

xx       = x-tranX;
yy       = y-tranY;

plot(xx,yy,'k','LineWidth',1)

grid on;
axis square;

% Add mile post lables
text(MP(1,1)-tranX,MP(1,2)-tranY,strcat(['+ MP' num2str(MP(1,3))]),'FontSize',7);
text(MP(2,1)-tranX,MP(2,2)-tranY,strcat(['+ MP' num2str(MP(2,3))]),'FontSize',7);

% Create xlabel
xlabel({strcat([ 'Easting ' num2str(tranX) '+'])});

% Create ylabel
ylabel({strcat([ 'Northing ' num2str(tranY) '+' ])});

% Create title
title(strcat(['Plan View']));

hold off;

%% Compute descriptive statistics.
% Read data file from XLS into structured array
%dataFile      = strcat([filePath fileName]);
%track         = csv2struct(dataFile);

% Skip if no stats.
if alpha ~= 0
% Normal parameter estimates
[mu,sigma,muci,sigmaci] = normfit(degOfCurv,alpha);
med           = median(degOfCurv);

%% Draw histogram
% setup location for statistic output

     n        = hist(degOfCurv);
     minX     = min(degOfCurv);
     maxX     = max(degOfCurv);
     xRange   = maxX - minX;
     xx       = minX + (0.0 * xRange);
     minY     = min(n);
     maxY     = max(n);
     yRange   = maxY - minY;
     yy       = maxY - (0.2 * yRange);
end
%% setup to display mean and std dev

% Identify and filter outliers, set to NaN
outliers = (degOfCurv - mu3) > 2*sigma3;
% Copy DOC to modified DOC
degOfCurvM = degOfCurv;
% Add NaN values
degOfCurvM(outliers) = NaN;
idx      = isnan(degOfCurvM);
idx      = not(idx);
dOc      = degOfCurvM(idx);
% Histogram bin counts
bin_counts    = hist(dOc);
% Maximum bin count
N             = max(bin_counts);
%N            = 10;
% plot histogram
subplot(3,3,8:9);
hist(dOc)
hold on;
% plot line for mean
plot([mu3 mu3],[0 N],'r','LineWidth',2) % Mean
% plot lines for std dev
oneSigma      = repmat(mu3 + sigma3*([-1 1]),2,1);
twoSigma      = repmat(mu3 + sigma3*([-2 2]),2,1);
Y             = repmat([0;N],1,2);
% plot std dev
plot(oneSigma,Y,'b','LineWidth',2)
plot(twoSigma,Y,'g','LineWidth',2)
xlim([minDOC maxDOC]);

% Create title, use [ ] syntax to preserve white space
title(strcat([ figName ' D_{c} Histogram'  ]));
% Create xlabel
xlabel('D_{c}','FontSize',14);
% Mu label
text('Interpreter','latex','HorizontalAlignment','center','VerticalAlignment'...
     ,'bottom','BackgroundColor','w','String','$$\mu_{D_{c}}$$','Position'...
     ,[mu3 N],'FontSize',11);
% One sigma label
text('Interpreter','latex','HorizontalAlignment','center','VerticalAlignment'...
     ,'bottom','BackgroundColor','w','String','$$\sigma_{D_{c}}$$','Position'...
     ,[(mu3+sigma3) N],'FontSize',11);
text('Interpreter','latex','HorizontalAlignment','center','VerticalAlignment'...
     ,'bottom','BackgroundColor','w','String','$$-\sigma_{D_{c}}$$','Position'...
     ,[(mu3-sigma3) N],'FontSize',11);
% Two sigma label
text('Interpreter','latex','HorizontalAlignment','center','VerticalAlignment'...
     ,'bottom','BackgroundColor','w','String','$$2\sigma_{D_{c}}$$','Position'...
     ,[(mu3+(sigma3*2)) N],'FontSize',11);
text('Interpreter','latex','HorizontalAlignment','center','VerticalAlignment'...
     ,'bottom','BackgroundColor','w','String','$$-2\sigma_{D_{c}}$$','Position'...
     ,[(mu3-(sigma3*2)) N],'FontSize',11);
% Create ylabel
ylabel('Observations');
h             = findobj(gca,'Type','patch');
set(h,'FaceColor','k','EdgeColor','w');
```

```
hold off

statDisp      = 0;
if statDisp
%% Setup for statistics text
% Descriptive stats with figure
str1(1)       = {strcat(['{\alpha} = ' num2str(alpha)])};
str1(2)       = {strcat(['{\mu} = ' num2str(mu)])};
str1(3)       = {strcat(['median = ' num2str(med)])};
str1(4)       = {strcat(['{\sigma} = ' num2str(sigma)])};
% str1(5)     = {strcat(['{\mu} ci = ' num2str(muci(1)) ' to ' num2str(muci(2))])};
% str1(6)     = {strcat(['{\sigma} ci = ' num2str(sigmaci(1)) ' to ' num2str(sigmaci(2))])};

text(xx,yy,str1,'FontSize',8);
hold off;
end

%% Output to pdf and png
% Setup page
set(gcf, 'PaperUnits', 'inches');
set(gcf, 'PaperType', 'usletter'); %'usletter'= 8.5x11, 'C'= 17x22, 'D'= 22x34
set(gcf, 'PaperOrientation', 'landscape');
set(gcf, 'PaperPositionMode', 'auto'); % options = 'auto' & 'manual
set(gcf, 'PaperPosition', [0.5, 1.0, 9.5, 7.0])
% Print pdf
suf = strcat('_hzal-',num2str(chordLength),'.pdf');
print ('-r150', '-dpdf', strcat([filePath figName suf])); % option '-loose',
% Print png
suf = strcat('_hzal-',num2str(chordLength),'.png');
set(gcf, 'PaperOrientation', 'portrait');
print ('-r300', '-dpng', strcat([filePath figName suf])); % option '-loose',

end

```