

```

1 function [DCdir midOrdDist midOrdX midOrdY midOrdZ chordAz] = findMidOrdDis
2 %
3 % findMidOrdDistance.m - returns distance between a chord's mid ordinate
4 % and a series of line segments defined by data points.
5 %
6 % Input: search radius, chord and mid-ordinate X & Y coordinates,
7 % easting, northing of a set of ordered data points
8 %
9 % Output: distance between mid ordinate of a chord and the intersection
10 % orthogonal with the chord at the mid ordinate, midOpt coords.
11 %
12 % Syntax:
13 %
14 % *****
15 % Other m-files required: dist2pts.m, midOrd.m, pts2eqn.m,
16 % perp2line.m, intsct2lines.m
17 %
18 % Subfunctions:
19 %
20 % MAT-files required: none
21 %
22 % See also: Route Location and Design, 5th ed. Thomas Hickerson, 1964
23 % chapter 14: "String-Lining Railroad Curves", pp 346-355
24 % *****
25 % Author: Peter J Dailey, inspired by Doug Hull's (Doug.Hull@mathworks.com)
26 % Matlab Video Tutorial: Intersecting a circle with a line series.
27 % email: daileypj@mac.com
28 % Doug's website posting: http://blogs.mathworks.com/videos/2008/02/19/...
29 % practical-example-intersecting-a-circle-with-a-line-series/
30 % Last revision: 11-Aug-2009
31 % *****
32 %
33 % Copyright 2010 by Peter J Dailey
34 %
35 % Permission to use, copy, modify, distribute, and sell this software and i
36 % documentation for any purpose is hereby granted without fee, provided tha
37 % the above copyright notice appear in all copies and that both that
38 % copyright notice and this permission notice appear in supporting
39 % documentation, and that the name of Peter J Dailey not be used in
40 % advertising or publicity pertaining to distribution of the software witho
41 % specific, written prior permission.
42 % Peter J Dailey makes no representations about the
43 % suitability of this software for any purpose. It is provided "as is"
44 % without express or implied warranty.
45 % *****
46
47 % find endpoint of chord, first is station, find second
48 [xChord yChord zChord] = getIntLinesCircle(chordLength,xStation,yStation,zS
49
50 % direction of chord
51 chordAz = az2pts (xStation,yStation,xChord,yChord);
52
53 % middle of chord coordinates
54 % Calculate middle of chord coordinate given end point coordinates
55 dX = abs(xStation - xChord);
56 dY = abs(yStation - yChord);
57 dZ = abs(zStation - zChord);
58 xMidOrd = min(xStation,xChord)+(dX/2);
59 yMidOrd = min(yStation,yChord)+(dY/2);
60 zMidOrd = min(zStation,zChord)+(dZ/2); %elev between chord end points

```

```

61
62 % find closest point to the mid-ordinate
63 [first] = closestPoint(xMidOrd,yMidOrd,zMidOrd,x,y,z);
64
65 % definition of the 2D chord line
66 [aChord bChord cChord] = pts2eqn (xStation,yStation,xChord,yChord);
67
68 % definition of the line orthogonal to the chord passing through the mid or
69 [aPerChord bPerChord cPerChord] = perp2line (xMidOrd,yMidOrd,aChord,bChord,
70
71 %
72 for i = 1:3 % three data points define two lines closest to MidOrdinate
73     % definition of each line
74     point = first-1;
75     xx(i) = x(point+i); % (the point one before closest - 1)
76     yy(i) = y(point+i);
77 end
78
79 [ a(1) b(1) c(1) ] = pts2eqn (xx(1),yy(1),xx(2),yy(2));
80 [ a(2) b(2) c(2) ] = pts2eqn (xx(2),yy(2),xx(3),yy(3));
81
82 % coordinates of the two intersections with the mid ordinate perpendicular
83 [xInt(1) yInt(1)] = intsct2lines (a(1),b(1),c(1),aPerChord,bPerChord,cPerC
84 [xInt(2) yInt(2)] = intsct2lines (a(2),b(2),c(2),aPerChord,bPerChord,cPerC
85
86 % distance from mid ordinate to each intersection
87 d = dist2pts (xMidOrd,yMidOrd,xInt,yInt);
88
89 % find the minimum distance to the two intersections
90 minD = min(d);
91 % index of intersection, use outside of loop
92 foundIdx = find(d == minD);
93 % If they are the same value then find has two index entries.
94 % Use the first one.
95 if numel(foundIdx) > 1
96     index = foundIdx(1);
97 else
98     index = foundIdx;
99 end
100
101 % *****
102 % This this was the original method to determine mid ordinate
103 % Assign the closest intersection to the return value
104 % Return values in midOrdX and midOrdY
105 closeMidOrdDist = d(index);
106 midOrdX = xInt(index);
107 midOrdY = yInt(index);
108 % *****
109
110 % *****
111 % This method determines the mid ordinate from the mean
112 % of the far and near intersections.
113 % Find the coordinates between the two intersections,
114 % Return values in midOrdX and midOrdYs
115 xMidMean = mean(xInt);
116 midOrdX = xMidMean;
117 yMidMean = mean(yInt);
118 midOrdY = yMidMean;
119 meanMidOrdDist = dist2pts(xMidOrd,yMidOrd,xMidMean,yMidMean);
120 % *****

```

```

121 %midOrdDist          = closeMidOrdDist;
122 midOrdDist          = meanMidOrdDist;
123 % *****
124
125 % calculate elevation at intersection
126 dist2Int            = dist2pts (x(first),y(first),xInt(index),yInt(index));
127
128 if index            == 1
129     Xpt1              = x(first-1);
130     Ypt1              = y(first-1);
131     Zpt1              = z(first-1);
132
133     Xpt2              = x(first);
134     Ypt2              = y(first);
135     Zpt2              = z(first);
136 else
137     Xpt1              = x(first);
138     Ypt1              = y(first);
139     Zpt1              = z(first);
140
141     Xpt2              = x(first+1);
142     Ypt2              = y(first+1);
143     Zpt2              = z(first+1);
144 end
145 %
146 midOrdZ              = getStaElev(dist2Int,Xpt1,Ypt1,Zpt1,Xpt2,Ypt2,Zpt2);
147 % *****
148 % find the direction from midordinate to intersection, (+ right) (- left)
149 moAz                 = az2pts(xMidOrd,yMidOrd,xInt(index),yInt(index));
150 DCdir                = round(moAz - chordAz);
151 DCdirStr             = num2str(DCdir);
152 switch DCdirStr
153     % point to the right of the midordinate
154     case '90'
155         direction      = -1;
156     case '-270'
157         direction      = -1;
158
159     % point to the left of the midordinate
160     case '-90'
161         direction      = 1;
162     case '270'
163         direction      = 1;
164 end
165 try
166 midOrdDist = midOrdDist * direction; % adj result left or right
167 catch
168     fprintf('MidOAz: %4.1f DCdir: %4.1f DCdirStr:%s\n',...
169         moAz,DCdir,DCdirStr);
170 end
171 end %function
172
173 % subfunction to find the point before the closest point to the midordinate
174 function [first] = closestPoint(xMidOrd,yMidOrd,zMidOrd,x,y,z)
175 % Input: data set X & Y values; circle center x & y.
176 % Output: index of all points within radius
177 deltaX = x - xMidOrd; % all X coord - circle origin X coord, Cx
178 deltaY = y - yMidOrd; % all Y coord - circle origin Y coord, Cy
179 deltaZ = z - zMidOrd; % all Z coord - circle origin Z coord, Cz
180

```

```
181 % distance = pythagorus from center to point(s)
182 distance          = sqrt(deltaX.^2 + deltaY.^2 + deltaZ.^2);
183
184 % find the closest point
185 closest           = min(distance);
186 % get index of closest point
187 index = find(distance == closest);
188 % return the index before the closest point
189 first = index-1;
190 end
191
```