Yarding Distance Analysis for Cable Systems Using a Digital Elevation Model

Francis E. Greulich
Associate Professor
Logging Engineering
University of Washington
Seattle, WA

Steward G. Pickford
Professor
Logging Engineering
University of Washington
Seattle, WA

Introduction

In this paper the search for faster as well as better estimates of yarding production and cost for the individual cable setting focuses on the estimation of average yarding distance (AYD). The AYD has a long history of use in the quantitative description of harvest settings (Greulich 1987). Very recent theoretical work (Greulich 1989) suggests that significant gains in estimation precision and speed might be possible under the most general of topographic conditions.

A proof-of-concept program has been developed to accept simulated digital elevation model (DEM) data, perform the necessary classifications, separate the DEM area into slope-aspect polygons, and finally create a data file containing the (x,y,z) coordinates of the vertices of the resulting planar polygons (triangles). This data file is then used as an input file for the average yarding distance algorithm. Within this latter algorithm each polygonal element is evaluated for its horizontal area and mean distance from the landing. This analysis makes use of a newly developed exact formula for the mean distance of a planar polygonal surface from a non-coplanar point. An area weighted mean distance is then calculated using all of the polygonal elements across the setting. This weighted mean distance is an estimate of the AYD for the given setting under the standard developmental assumptions for a cable setting.

A listing of the FORTRAN coded AYD algorithm is given in the appendix.

Fitting Meshed Planes to a DEM

Standard geographic information system (GIS) operations on a DEM easily produce maps of slope and aspect classes. These two themes can be combined into a third with its classes defined by slope-aspect combinations. Several approaches exist to accomplish this task. One common

approach produces a lattice in which the grid points are simply connected together to form a set of interconnected triangles. Another, somewhat similar approach produces a triangulated irregular network wherein the sizes and positions of the resulting triangles reflect the underlying topographic surface.

The average yarding distance algorithm can be applied to most forms of data that have been classified by slope and aspect. However many areas of interest; e.g., settings with long uniform slopes and few breaks, can often be divided into planar polygons much larger than would be obtained from the grid spacing of the USGS 7 1/2 minute DEMs. In such instances a larger polygon may approximate the surface reasonably well while reducing computation time. To this end a program was written to identify large planar areas from DEM data which had been stratified into slope-aspect classes. This DEM-manipulating program was written in C, and its characteristics and general operation are described here.

The DEM program operates as a sequence of functions called from the main routine. These functions are listed in the order in which they are called and are briefly described below.

main( )	control routine
InitGrp( )	initialize graphics
RegisterDrivers( )	. .
RegisterFonts( )	. .
read_data( )	get DEM data into program
slp(i,j,k)	compute slopes using unit vectors derived from DEM
asp(i,j,k)	compute aspects using unit vectors derived from DEM
classify(&n,a,b)	classify slope or aspect data in array a into n classes and store in array b
combine(c,d,e)	create unique slope-aspect classes from slope c and aspect d arrays and store in array e
bndryseg( )	find boundaries between grid points with differenct slope-aspect class values
mkalnods( )	enumerate all possible boundaries
bblsort(m)	sort list of all m boundaries
FindCorners( )	find the corners of the DEM area in the boundary list
FindNodes( )	find points in boundary list where three or more boundaries meet

FindPaths( )	find all paths along boundary segments between nodes and corners, nodes and nodes, and corners and corners
MakePoly( )	combine paths into single polygons (those that do not contain another polygon)
Triangle( )	divide each polygon into triangles and print vertices as output file

Slope and aspect values are computed at each DEM grid point using a unit vector method described by Ritter (1987). This function produces slope and aspect values at each grid point that depend on z-values of the four neighboring grid points. Ritter's routine is modified to accept unequal grid spacings in the x- and y-directions. This modification accommodates computations along the map edge.

Slope data are classified by default into five classes: 0-30%, 30-60%, 60-90%, 90-120% and 120+%. The program will however accept user-specified slope classes. Likewise, aspect data are classified by default into eight classes beginning with North (337.5° to 22.5° azimuth). Once again these classes can be changed by the user. The resulting two data sets are combined to produce unique slope-aspect classes.

Values in the combined slope-aspect array are examined individually, and when differences between neighboring values are found, a boundary is constructed between the two values. Boundaries are also drawn completely around the DEM map area. When all boundaries have been enumerated, the list is sorted into ascending order.

A simple continuous boundary segment has two paths into each end point: one from each neighboring point. Where two or three boundaries meet there are three or four paths respectively into each end point. Therefore, in the enumerated list of boundary segments, points where two or more boundaries meet occur three or four times in succession. The FindNodes( ) function uses this characteristic to identify such points (herein called nodes). These nodes, and the four corners of the DEM area define the polygons surrounding areas with the same slope-aspect class.

The FindPaths( ) function traces the paths between each node or corner and the next neighboring node or corner. Once these paths are known, the MakePoly( ) function traces around these paths to define all the simple polygons which together make up the DEM map area. Finally the Triangle( ) function divides each polygon of four or more sides into triangles and writes the (x,y,z) coordinates of each vertex

to an output data file. This file contains the data used by the average yarding distance algorithm.

Average Yarding Distance Algorithm

The computational algorithm presented in the paragraphs below and given as a FORTRAN listing in the appendix has been described in general terms in a previous paper (Greulich 1989). An effort has been made to maintain notational and structural connectivity with the previous paper so that easy reference to it may assist the reader. In order to facilitate development and presentation of the mathematical model the procedures and notation of linear algebra are employed.

The coordinates of the landing are given by the position vector "S":

$$S = \begin{bmatrix} x_\bullet \\ y_\bullet \\ z_\bullet \end{bmatrix}$$

and the coordinates of the "i"th vertex of the "q"th polygonal element are given by the position vector "$T_{i,q}$":

$$T_{i,q} = \begin{bmatrix} x_i \\ y_i \\ z_i \end{bmatrix}_q$$

The "n_q" vertices of the "q"th polygonal element are all assumed to lie in the same plane. The corresponding vertex position vectors are stored as sequential rows in the matrix "M_q". Vector selection for entry into this matrix proceeds in a counter-clockwise pattern (as viewed from above) around the polygonal element. The arbitrarily selected starting vector is repeated, also being the last entry in this n_q+1 X 3 matrix.

$$M_q = \begin{bmatrix} x_1\ y_1\ z_1 \\ x_2\ y_2\ z_2 \\ \\ x_i\ y_i\ z_i \\ x_j\ y_j\ z_j \\ \\ x_{n_q} y_{n_q} z_{n_q} \\ x_1\ y_1\ z_1 \end{bmatrix}_q$$

With regard to notation it should be noted that j≡i+1. This identity provides an important notational tie to the previously mentioned paper. Additionally, in the development that follows attention is restricted to one

polygonal element and the "q" index has subsequently been dropped for notational simplicity.

The length and bearing of a side "i" of the polygonal element may be obtained from the vector "V_i" which is calculated as:

$$V_i = T_{i+1} - T_i \qquad i=1,n \text{ and, } T_{n+1}=T_1$$

In order to find a unit vector, "U_o", perpendicular to the plane of the polygonal element select from $\{V_i | i=1,n\}$ any two non-collinear vectors "V_u" and "V_v"; then

$$R_o = V_u \times V_v$$

where the bold cross, "X", indicates the vector product operation. The unit perpendicular vector is then given by:

$$U_o = R_o / \|R_o\|$$

This unit perpendicular vector can now be used to find the coordinates of the point on the plane of the polygonal element (a plane denoted X'Y' in the previous publication) that is nearest to the location, "S", of the landing. This position vector, denoted "Ω", is found by arbitrarily selecting a vector "T_i" from "M" and calculating:

$$W = T_i - S$$

and

$$\delta_o = W \bullet U_o$$

where the dot, "•", indicates the scalar product operation; finally then

$$\Omega = S + \delta_o U_o = \begin{bmatrix} x_o \\ y_o \\ z_o \end{bmatrix}$$

For each side "i" (i=1,n) of the polygonal element calculate:

$$R_i = V_i \times U_o$$

$$U_i = R_i / \|R_i\|$$

$$W_i = T_i - \Omega$$

$$\delta_i = W_i \bullet U_i$$

Yielding the position vector:

$$P_i = \Omega + \delta_i U_i = \begin{bmatrix} x_p \\ y_p \\ z_p \end{bmatrix}_i$$

The position vectors "P_i" give the location of the point on the extended side "i" of the polygonal element that is nearest the point "Ω".

It is now possible to proceed to the calculation of the tetrahedral side lengths required by the average yarding distance formula.

$$L_{si} = \|S - T_i\|$$

$$L_{sj} = \|S - T_j\|$$

$$L_{op} = \|\Omega - P_i\|$$

$$L_{so} = \|S - \Omega\|$$

$$L_{pi} = \|P_i - T_i\|$$

$$L_{pj} = \|P_i - T_j\|$$

$$L_{sp} = \|S - P_i\|$$

The average yarding distance (AYD) for the triangle "opi" is calculated as:

$$AYD_{opi} = \left[\frac{L_{si}}{3}\right] + \left[\frac{L_{op}^2 + 3L_{so}^2}{3L_{pi}}\right]\left[\ln\left(\frac{L_{pi} + L_{si}}{L_{sp}}\right)\right]$$

$$- \left[\frac{2L_{so}^3}{3L_{op}L_{pi}}\right]\left[\tan^{-1}\left(\frac{L_{op}L_{pi}}{L_{sp}^2 + L_{so}L_{si}}\right)\right]$$

The AYD for triangle "opj" is calculated using the same formula but with "j" substituted wherever "i" appears.

The horizontal area corresponding to each of the two triangles is given by:

$$A_{opj} = \frac{1}{2}\begin{vmatrix} x_o & y_o & 1 \\ x_p & y_p & 1 \\ x_j & y_j & 1 \end{vmatrix}$$

and

$$A_{oip} = \frac{1}{2}\begin{vmatrix} x_o & y_o & 1 \\ x_i & y_i & 1 \\ x_p & y_p & 1 \end{vmatrix}$$

The horizontal area, A_i, and the average yarding distance, AYD_i, associated with side "i" are then calculated by:

$$A_i = A_{opj} + A_{oip}$$

and

$$AYD_i = \frac{A_{opj}AYD_{opj} + A_{oip}AYD_{opi}}{A_i}$$

After the horizontal area and average yarding distance for each side of the "q"th polygonal element have been calculated then the composite area, A_q, and average yarding distance, AYD_q, can be found:

$$A_q = \sum_{i=1}^{n_q} A_i$$

and

$$AYD_q = \frac{\sum_{i=1}^{n_q} A_i AYD_i}{A_q}$$

After all of the polygonal elements have been evaluated the area and average yarding distance for the setting may be computed:

$$A = \sum_q A_q$$

and

$$AYD = \frac{\sum_q A_q AYD_q}{A}$$

which completes the algorithm.

References

Greulich, Francis E. 1987. "The quantitative description of cable yarder settings - parameters for the triangular setting with apical landing" Forest Science 33(3):603-616.

Greulich, Francis E. 1989. "The calculation of average yarding distance to a centralized landing, given a polygonal mesh approximation of the setting surface" Canadian Journal of Forest Research 19(1):141-144.

Ritter, Paul. 1987. "A vector-based slope and aspect generation algorithm" Photogrammatric Engineering and Remote Sensing 53(8):1109-1111.

APPENDIX A

The following FORTRAN 77 program is written in accordance with ANSI standard X3.9-1978. Program variable names were generally selected to be suggestive of those used in the body of this paper. A serious effort has been made to eliminate programming errors but the prudent user will independently validate program results. Some program efficiency has been sacrificed in the interest of clarity and serious users might consider rewriting the code.

```
      PROGRAM AYD3D
C     *****
C     THIS PROGRAM CALCULATES THE AVERAGE YARDING DISTANCE (AYD)
C     FOR A THREE DIMENSIONAL SURFACE WHICH HAS BEEN APPROXIMATED
C     BY A POLYGONAL MESH WHOSE ELEMENTS CONSIST OF PLANAR POLYGONS
C     THERE IS NO REQUIREMENT THAT TRIANGULAR ELEMENTS BE USED
C     THOUGH THAT IS GENERALLY THE EASIEST PLANAR POLYGON TO FIT
C     THE POINT (LANDING) TO WHICH THE AVERAGE DISTANCE IS TO BE
C     CALCULATED MAY BE FREELY PLACED IN THREE DIMENSIONAL SPACE
C     *****
      DIMENSION N(10),S(3),T(30,3),RO(3),UO(3),W(3),O(3),RI(3),UI(3),
     1      WI(3),P(3),SA(30),SAYD(30),EA(10),EAYD(10),V(30,3)
      REAL LSO,LSI,LSJ,LOP,LPI,LPJ,LSP
      INTEGER  Q, QMAX, H
C     *****
C     THE DESIGNATED FILE FOR UNIT 7 CONTAINS ALL INPUT DATA
C     "QMAX" IS THE NUMBER OF POLYGONAL ELEMENTS TO BE EVALUATED
C     "N(Q)" IS THE NUMBER OF SIDES TO POLYGONAL ELEMENT "Q"
C     "I" IS THE INDEX ON VERTICES, I=1,N(Q)+1; NOTE THAT FOR
C     EACH ELEMENT THE FIRST VERTEX READ IS ALSO THE LAST
C     "S(H)" IS THE POSITION VECTOR FOR THE LANDING LOCATION
C     *****
      OPEN(UNIT=7,FILE='POLY1.DAT')
      READ(7,10)QMAX,(N(Q),Q=1,QMAX)
10    FORMAT(15I5)
      READ(7,15)(S(H),H=1,3)
15    FORMAT(3F5.0)
C     *****
C     LOOP THROUGH THE BASIC PROGRAM FOR EACH POLYGONAL ELEMENT
C     AT THE START OF EACH LOOP READ THE VERTEX POSITION VECTORS
C     "T(I,H)" FOR THE CURRENT POLYGONAL ELEMENT
C     *****
      DO 1000 Q=1,QMAX
      READ(7,15)((T(I,H),H=1,3),I=1,N(Q)+1)
C     *****
C     CALCULATE SEQUENTIAL COUNTER-CLOCKWISE VECTORS AROUND THE
C     CURRENT POLYGONAL ELEMENT
C     *****
      DO 100 I=1,N(Q)
      DO 200 H=1,3
      V(I,H)=T(I+1,H)-T(I,H)
200   CONTINUE
```

```
100   CONTINUE
C     *****
C     CALCULATE THE COORDINATES "O(H)" OF THE POINT ON THE X'Y'
C     PLANE OF THE CURRENT POLYGONAL ELEMENT THAT IS NEAREST TO
C     THE LANDING
C     *****
      I=1
50    CONTINUE
      RO(1)=V(I,2)*V(I+1,3)-V(I+1,2)*V(I,3)
      RO(2)=V(I+1,1)*V(I,3)-V(I,1)*V(I+1,3)
      RO(3)=V(I,1)*V(I+1,2)-V(I+1,1)*V(I,2)
      ENRM=SQRT(RO(1)**2+RO(2)**2+RO(3)**2)
      I=I+1
      IF(ENRM.EQ.0.0)GO TO 50
      UO(1)=RO(1)/ENRM
      UO(2)=RO(2)/ENRM
      UO(3)=RO(3)/ENRM
      W(1)=T(I,1)-S(1)
      W(2)=T(I,2)-S(2)
      W(3)=T(I,3)-S(3)
      DO=W(1)*UO(1)+W(2)*UO(2)+W(3)*UO(3)
      O(1)=S(1)+DO*UO(1)
      O(2)=S(2)+DO*UO(2)
      O(3)=S(3)+DO*UO(3)
C     *****
C     CALCULATE THE DISTANCE "LSO" FROM THE LANDING TO THE NEAREST
C     POINT ON THE PLANE X'Y'
C     *****
      LSO=0.0
      DO 300 H=1,3
      LSO=LSO+(O(H)-S(H))**2
300   CONTINUE
      LSO=SQRT(LSO)
C     *****
C     FOR EACH SIDE OF THE CURRENT POLYGONAL ELEMENT CALCULATE
C     ITS HORIZONTAL AREA AND MEAN DISTANCE FROM THE LANDING
C     *****
      DO 2000 I=1,N(Q)
C     *****
C     CALCULATE THE COORDINATES "P(H)" OF THE POINT ON THE EXTENDED
C     LINE OF THE CURRENT SIDE THAT IS NEAREST THE POINT "O(H)"
C     *****
      RI(1)=UO(3)*V(I,2)-UO(2)*V(I,3)
      RI(2)=UO(1)*V(I,3)-UO(3)*V(I,1)
      RI(3)=UO(2)*V(I,1)-UO(1)*V(I,2)
      ENRM=SQRT(RI(1)**2+RI(2)**2+RI(3)**2)
      UI(1)=RI(1)/ENRM
      UI(2)=RI(2)/ENRM
      UI(3)=RI(3)/ENRM
      WI(1)=T(I,1)-O(1)
      WI(2)=T(I,2)-O(2)
      WI(3)=T(I,3)-O(3)
      DI=UI(1)*WI(1)+UI(2)*WI(2)+UI(3)*WI(3)
      P(1)=O(1)+DI*UI(1)
```

```
      P(2)=O(2)+DI*UI(2)
      P(3)=O(3)+DI*UI(3)
C     *****
C     CALCULATE DISTANCES:
C     "LSI", "LSJ" ARE THE DISTANCES FROM THE LANDING TO THE TWO
C     VERTICES (ORDERED SEQUENTIALLY) OF THE CURRENT SIDE
C     "LPI", "LPJ" ARE THE DISTANCES FROM THE PERPENDICULAR POINT
C     ON THE LINE TO THE TWO VERTICES OF THE CURRENT SIDE
C     "LOP"  DISTANCE FROM POINT "O(H)" TO POINT "P(H)"
C     "LSP"  DISTANCE FROM POINT "S(H)" TO POINT "P(H)"
C     *****
      LSI=0.0
      LSJ=0.0
      LOP=0.0
      LPI=0.0
      LPJ=0.0
      LSP=0.0
      DO 400 H=1,3
      LSI=LSI+(S(H)-T(I,H))**2
      LSJ=LSJ+(S(H)-T(I+1,H))**2
      LOP=LOP+(O(H)-P(H))**2
      LPI=LPI+(T(I,H)-P(H))**2
      LPJ=LPJ+(T(I+1,H)-P(H))**2
      LSP=LSP+(P(H)-S(H))**2
400   CONTINUE
      LSI=SQRT(LSI)
      LSJ=SQRT(LSJ)
      LOP=SQRT(LOP)
      LPI=SQRT(LPI)
      LPJ=SQRT(LPJ)
      LSP=SQRT(LSP)
C     *****
C     CALCULATE THE HORIZONTAL AREA AND MEAN DISTANCE FROM THE
C     LANDING FOR EACH OF THE TWO TRIANGLES "opj" AND "oip" OF
C     THE CURRENT SIDE, VIZ, "AI" "AJ" "AYDI" AND "AYDJ"
C     *****
      IF(LOP.EQ.0.0)GO TO 80
      IF(LPI.EQ.0.0)GO TO 110
      AYDI=(LSI/3.)+((LOP**2+3.*LSO**2)/(3.*LPI))*
     1      ALOG((LPI+LSI)/
     2      LSP)-((2.*LSO**3)/(3.*LOP*LPI))*ATAN((LOP*LPI)/
     3      (LSP**2+LSO*LSI))
      GO TO 120
110   AYDI=0.0
120   IF(LPJ.EQ.0.0)GO TO 130
      AYDJ=(LSJ/3.)+((LOP**2+3.*LSO**2)/(3.*LPJ))*
     1      ALOG((LPJ+LSJ)/
     2      LSP)-((2.*LSO**3)/(3.*LOP*LPJ))*ATAN((LOP*LPJ)/
     3      (LSP**2+LSO*LSJ))
      GO TO 90
130   AYDJ=0.0
      GO TO 90
80    CONTINUE
      AYDI=0.0
```

```
      AYDJ=0.0
90    CONTINUE
      AI=(-.5)*((P(1)-O(1))*(T(I,2)-O(2))-
     1 (P(2)-O(2))*(T(I,1)-O(1)))
      AJ=(+.5)*((P(1)-O(1))*(T(I+1,2)-O(2))-
     1 (P(2)-O(2))*(T(I+1,1)-O(1)))
C     *****
C     COMBINE THE TWO AREAS AND MEAN DISTANCES TO OBTAIN THE
C     HORIZONTAL AREA AND MEAN DISTANCE FOR THE TRIANGLE "oij"
C     OF THE CURRENT SIDE; VIZ, "SA(I)" AND "SAYD(I)"
C     *****
      SA(I)=AI+AJ
      IF(SA(I).EQ.0.0)GO TO 60
      SAYD(I)=(AJ*AYDJ+AI*AYDI)/SA(I)
      GO TO 2000
60    SAYD(I)=0.0
2000  CONTINUE
C     *****
C     CALCULATE THE HORIZONTAL AREA AND MEAN DISTANCE FROM THE
C     LANDING FOR THE CURRENT POLYGONAL ELEMENT
C     *****
      EAYD(Q)=0.0
      EA(Q)=0.0
      DO 500 I=1,N(Q)
      EA(Q)=EA(Q)+SA(I)
      EAYD(Q)=EAYD(Q)+SA(I)*SAYD(I)
500   CONTINUE
      IF(EA(Q).EQ.0.0)GO TO 70
      EAYD(Q)=EAYD(Q)/EA(Q)
      GO TO 1000
70    EAYD(Q)=0.0
1000  CONTINUE
C     *****
C     CALCULATE THE HORIZONTAL AREA AND MEAN DISTANCE FROM THE
C     LANDING FOR THE SETTING
C     *****
      AYD=0.0
      A=0.0
      DO 600 Q=1,QMAX
      A=A+EA(Q)
      AYD=AYD+EA(Q)*EAYD(Q)
600   CONTINUE
      AYD=AYD/A
      WRITE(6,*) A, AYD
      RETURN
      END
```

ISSN 0743-9564

PROCEEDINGS OF IUFRO 1990
S3:04 Subject Area
XIX WORLD CONGRESS

Montreal, PQ, Canada
5–11 August 1990

International Union of Forestry Research Organizations
XIX World Congress

Internationaler Verband Forstlicher Forschungsanstalten
XIX Weltkongreβ

Union Internationale des Instituts de Recherches Forestières
XIX Congrès mondial

Unión International de Organizaciones de Investgación Forestal
XIX Congreso Mundial

Miscellaneous Report 354

January 1991

MAINE AGRICULTURAL EXPERIMENT STATION
University of Maine

Proceedings of IUFRO 1990 S3:04 Subject Area XIX World Congress

Montreal, PQ, Canada
5–11 August 1990

Edited by

Thomas J. Corcoran

Peter E. Linehan

and
Suping Liu

Department of Forest Management
University of Maine
Orono, Maine 04469

IUFRO Subject Area S3:04 scientific responsibilities included (1) the planning and control of forest operations and (2) work study. The 1990 World Congress theme was "Science in Forestry: IUFRO's Second Century."