United States
Department of
Agriculture

Forest Service

**Southern
Research Station**

General Technical
Report SRS–67

A Simulation Algorithm to Approximate the Area of Mapped Forest Inventory Plots

William A. Bechtold, Naser E. Heravi, and Matthew E. Kinkenon

The Authors

William A. Bechtold, Research Forester, USDA Forest Service, Southern Research Station, Asheville, NC 28802; **Naser E. Heravi**, Professor, Community College of Southern Nevada, Henderson, NV 89015; and **Matthew E. Kinkenon**, Database System Consultant, University of Nevada Las Vegas, Harry Reid Center, Las Vegas, NV 89154-4009.

July 2003

Southern Research Station
P.O. Box 2680
Asheville, NC 28802

A Simulation Algorithm to Approximate the Area of Mapped Forest Inventory Plots

William A. Bechtold, Naser E. Heravi, and Matthew E. Kinkenon

Abstract

Calculating the area of polygons associated with mapped forest inventory plots can be mathematically cumbersome, especially when computing change between inventories. We developed a simulation technique that utilizes a computer-generated dot grid and geometry to estimate the area of mapped polygons within any size circle. The technique also yields a matrix of change in mapped-plot area between two points in time.

Keywords: Area transition matrix, condition class, forest inventory, mapped plots, plot area calculation.

Introduction

The U.S. Department of Agriculture, Forest Service, Forest Inventory and Analysis (FIA) Program utilizes a mapped, fixed-plot design as part of their national core sampling protocols (Hahn and others 1995). Each ground plot contains a cluster of four points spaced 120 feet apart. Each point is surrounded by a 24-foot fixed-radius subplot where trees 5.0-inches diameter at breast height (d.b.h.) and larger are measured. Data are also gathered about the area, or setting in which the trees are located. To enhance analysis, it is important that the tree data recorded on these plots are properly associated with the area classifications. To accomplish this, plots are mapped by "condition class" (U.S. Forest Service 2000).[1] Field crews assign an arbitrary number, usually 1, to the first condition class encountered on a plot. This number is then characterized by a series of predetermined variables attached to it (e.g., forest type, stand size, and stand age). Additional condition classes are identified if there is a distinct change in any of the condition-class variables on the plot.

Sometimes a plot straddles two or more distinct conditions. Boundaries usually occur between the subplots, but occasionally bisect the subplots, in which case they are mapped. When mapping a subplot, field crews first specify and define (if not previously defined) the condition class at subplot center. If a subplot straddles two or more conditions, they specify the condition class that contrasts with the condition at subplot center. Standing at subplot center and facing the contrasting condition, they then record the two azimuths where the boundary crosses the subplot perimeter. A third azimuth, with a distance, is permissible if the boundary contains a sharp curve or a corner. All trees tallied are then assigned to the condition class in which they occur. Figure 1 shows the data elements recorded for each subplot boundary: center condition number, contrasting condition number, left azimuth, right azimuth, and corner azimuth and distance. By convention, left and right azimuths are always assigned from the crew's perspective at subplot center (i.e., left to right is always clockwise on the subplot perimeter), the maximum angle between left and right is 180^{o} (unless a corner point has been identified), and boundary lines are not permitted to cross. Corner azimuths and distances are optional.

Given the available boundary data, the area in each resulting polygon can be computed mathematically (Scott and Bechtold 1995). However, when subplot maps from two different points in time are overlaid for the purpose of computing change in area among forest conditions (i.e., transition matrices), the mathematics can become unwieldy. This paper describes a simulation technique that approximates the area of all polygons mapped within any size circle, as well as the transition matrix between two points in time.

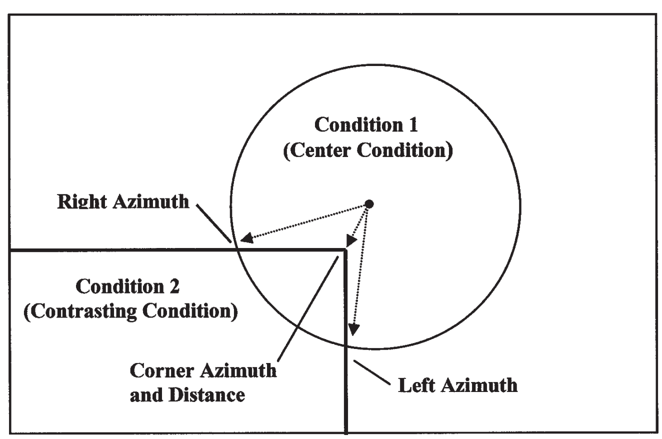

Figure 1—Using azimuths to reference a boundary between condition 1 and condition 2 to subplot center.

[1] U.S. Department of Agriculture, Forest Service. 2000. Forest inventory and analysis national core field guide, volume 1: field data collection procedures for phase 2 plots, vers. 1.4. Washington Office. Internal report. On file with: U.S. Department of Agriculture, Forest Service, Forest Inventory and Analysis, 201 14th St., Washington, D.C. 20250. htttp://fia.fs.fed.us/library.htm#manuals. [Date accessed: April 22, 2003].

Methods

A. A uniform dot grid of a specified number of points (e.g., 100,000) is superimposed on a unit circle. The density of grid points depends on the precision and computer speed required by the user.

B. All boundary data recorded in the field are converted to Polar and Cartesian coordinates on a unit circle.

C. Each contrasting condition class (i.e., mapped polygon) is associated with the boundary segments that surround it (e.g., its arc on the perimeter plus one boundary line if it has no corner; or its arc on the perimeter plus two boundary lines if it has a corner).

D. Rays are projected horizontally in one direction from each simulated grid point to the perimeter of the unit circle. Logic is used to assign each grid point to a condition class based on the number of times its projected ray crosses a boundary segment.

E. The number of grid points associated with each condition is then tallied, yielding percentages by condition class after all grid points are counted.

Generating the Dot Grid

The dot grid is generated by first calculating the distance (D) between grid points on a unit-circle with a radius of 1:

$$D = \sqrt{\frac{\pi}{n}} \tag{1}$$

where

$n =$ the number of grid points to be generated.

Grid points (G_p) can then be created two to four at a time using symmetry in a loop as in the fragment of code shown below:

$G_1 = (0,0)$; * assign grid-point 1 to origin;
$p = 2$; * initialize grid-point counter;
$x = 0$; $y = 0$; * initialize x and y coordinates;

Loop from 1 to n;
  $x = x + D$;
  if $x^2 + y^2 > 1$ then do; * check for edge of circle, restart grid on y-axis;
    $y = y + D$; $x = 0$;
  end;
  if $y > 1$ then end loop; * check for end of loop at top of y-axis;
  if $y = 0$ then do; * generate grid points 2 at a time on x-axis;
    $G_p = (x, y); p = p + 1;$
    $G_p = (-x, y); p = p + 1;$
  end;
  if $x = 0$ then do; * generate grid points 2 at a time on y-axis;
    $G_p = (x, y); p = p + 1;$
    $G_p = (x, -y); p = p + 1;$
  end;
  if (x $\neq 0$) and (y $\neq 0$) then do; * generate grid points 4 at a time off axes;
    $G_p = (x, y); p = p + 1;$
    $G_p = (x, -y); p = p + 1;$
    $G_p = (-x, y); p = p + 1;$
    $G_p = (-x, -y); p = p + 1;$
  end;
End Loop;

$p = p - 1$; * final grid-point count.

Note that grid points only have to be generated once. After a suitable grid density is established, the same array of grid points can be stored and used for boundary checks on any subplot.

Boundary Coordinates

The azimuths used to map FIA plots do not conform to conventional geometry. FIA boundaries are referenced to azimuth readings where 0º is due north, with angles of increasing degrees proceeding in a clockwise direction. In the geometric polar system the 0º mark is at due east, with angles of increasing degrees proceeding in a counter-clockwise direction (fig. 2).

In order to apply conventional geometry techniques to FIA boundaries, FIA azimuths are converted to the geometric polar system. The following equation can be used to accomplish the conversion:

$$\theta = 360 - \theta_F + 90 - Q \tag{2}$$

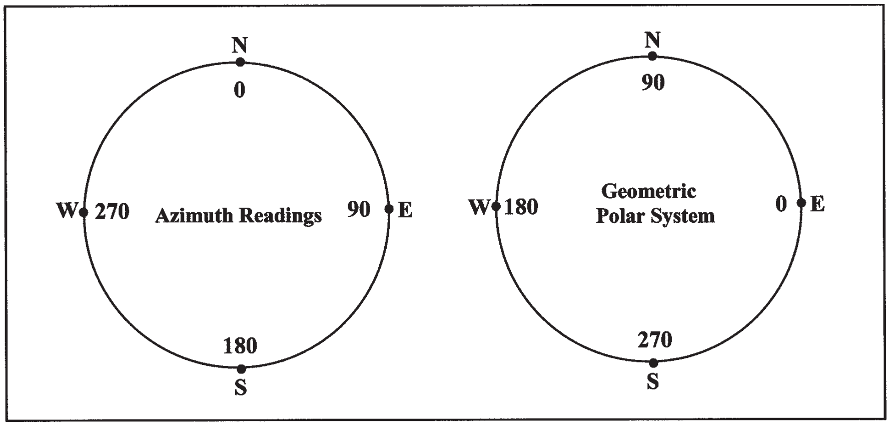

Figure 2—FIA azimuth readings and the geometric polar system.

where

θ = the polar angle,
θ_F = the FIA azimuth, and
$Q = 360$ if $\theta_F \leq 90$; 0 otherwise.

The points at which the left and right boundary azimuths cross the arc of the subplot are thus defined as the polar coordinates (R, θ_L) and (R, θ_R), and the location of the boundary corner point, if any, is defined as (C, θ_C).

where

θ_L = the polar left angle,
θ_R = the polar right angle,
θ_C = the polar corner angle,
R = the subplot radius, and
C = the corner distance.

Cartesian (x, y) coordinates are easily obtained from the polar coordinates

$x = R(\cos\theta)$, and
$y = R(\sin\theta)$.

Note that most computer trigonometric functions require the argument to be specified in radians, so θ may need to be converted to radians before applying sin and cos computer functions (i.e., multiply by $\pi/180$).

When placed on a unit-circle basis, the Cartesian coordinates of FIA boundary points on the perimeter of the circle are

x = 1 $(\cos\theta)$, and
y = 1 $(\sin\theta)$.

The Cartesian coordinates of boundary points associated with a corner point are

$x = C'(\cos\theta)$, and
$y = C'(\sin\theta)$,

where

$C' = C/R$.

All discussion hereafter is based on polar and Cartesian coordinates in terms of a unit circle.

Establishing Boundary Segments

Each contrasting condition class is associated with the two or three boundary segments that border it: its arc on the perimeter plus one boundary line (if no corner present), or its arc on the perimeter plus two boundary lines (if corner present).

The boundary arc—The range of the boundary arc on the subplot perimeter of a unit circle includes all coordinate points between (θ_L) and (θ_R). For contrasting condition 2 in figure 3, the boundary arc includes all points on the arc $\theta_{L2}\theta_{R2}$.

Boundary line without corner—The range of a boundary line with no corner point includes all points that lie along the line segment with endpoints (θ_L) and (θ_R). For contrasting condition 2 in figure 3, the boundary line includes all points on the line segment $\theta_{L2}\theta_{R2}$.

Boundary line with corner—A boundary with a corner point implies two boundary lines. The range of each is computed in the same manner as the single boundary (with no corner) as described above, except that both are truncated at the point of intersection (i.e., the corner):

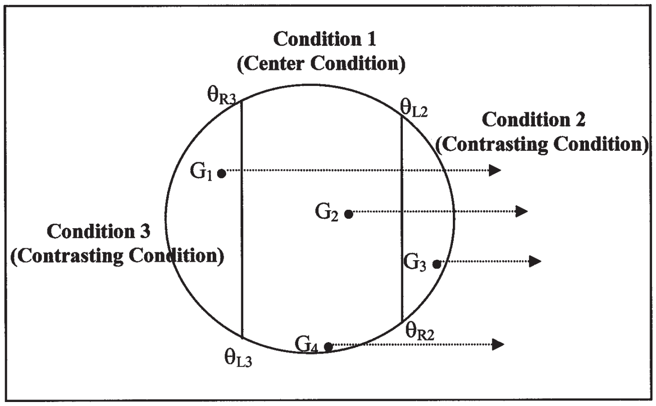

Figure 3—Projecting grid-point rays across a center condition bordered by two contrasting conditions.

Segment 1 has end points (θ_L) and (C',θ_C).

Segment 2 has end points (θ_R) and (C',θ_C).

For contrasting condition 2 in figure 4, segment 1 includes all points on line segment $\theta_{L2}(C'_2,\theta_{C2})$, and segment 2 includes all points on line segment $\theta_{R2}(C'_2,\theta_{C2})$.

Projecting Rays

After the range of each boundary segment associated with a contrasting condition class has been established, rays are projected horizontally in one direction (to the right) from each superimposed grid point to the circle perimeter. The following logic is then applied:

- If the ray projected from a grid point crosses the boundary segments (line or arc) of a contrasting condition class an odd number of times, then that point must be located within the contrasting condition.
- If the ray crosses the boundary segments an even number of times (or not at all), then that point cannot be located within that contrasting condition class.
- If the point is not located within any contrasting condition class, it is associated with the center condition class.

Figure 3 shows a center condition class bounded on either side by two contrasting condition classes. This situation is common when a road crosses a subplot. The boundary segments associated with contrasting condition 2 are line segment $\theta_{L2}\theta_{R2}$ and arc $\theta_{L2}\theta_{R2}$. The boundary segments associated with contrasting condition 3 are line segment $\theta_{L3}\theta_{R3}$ and arc $\theta_{L3}\theta_{R3}$. The horizontal ray projected from grid-point G_1 crosses boundary segments of condition 3 once (line segment $\theta_{L3}\theta_{R3}$), so grid-point G_1 is assigned to condition 3. The horizontal ray projected from grid-point G_2 crosses boundary segments of condition 2 twice (line segment $\theta_{L2}\theta_{R2}$ and arc $\theta_{L2}\theta_{R2}$), so grid-point G_2 is not assigned to condition 2 (or any other contrasting condition). Grid-point G_2 is thus assigned to the center condition class (condition 1) by default. Grid-point G_4 is also assigned to the center condition class because its projected ray crosses no contrasting-condition boundary segments. The horizontal ray projected from grid-point G_3 crosses boundary segments of condition 2 once (arc $\theta_{L2}\theta_{R2}$), so grid-point G_3 is assigned to condition 2.

The boundary segments associated with contrasting condition 2 in figure 4 are line segment $\theta_{L2}(C'_2,\theta_{C2})$, line segment $\theta_{R2}(C'_2,\theta_{C2})$, and arc $\theta_{L2}\theta_{R2}$. The horizontal ray projected from grid-point G_1 crosses boundary segments of condition 2 once (line segment $\theta_{R2}(C'_2,\theta_{C2})$), so grid-point G_1 is assigned to condition 2. The horizontal ray projected from grid-point G_2 crosses boundary segments of condition 2 twice (line segments $\theta_{L2}(C'_2,\theta_{C2})$, and $\theta_{R2}(C'_2,\theta_{C2})$), so grid-point G_2 is not assigned to condition 2 or any other contrasting condition. Grid-point G_2 is therefore assigned to center condition 1 by default. Grid-point G_3 is also assigned to the center condition because its projected ray crosses no contrasting-condition boundary segments.

Rather than assigning grid points to the center condition by default, the center condition could be actively defined by the lines and arcs that encompass it. The same logic used to assign grid points to contrasting conditions would then apply to center conditions. This provides some measure of safety by enabling checks for programming errors resulting in the failure to assign a grid point to any polygon. However, because center conditions are not actively defined in the field, identification of the lines and arcs that contain them is difficult, which increases the complexity of the algorithm and adds another potential source of error.

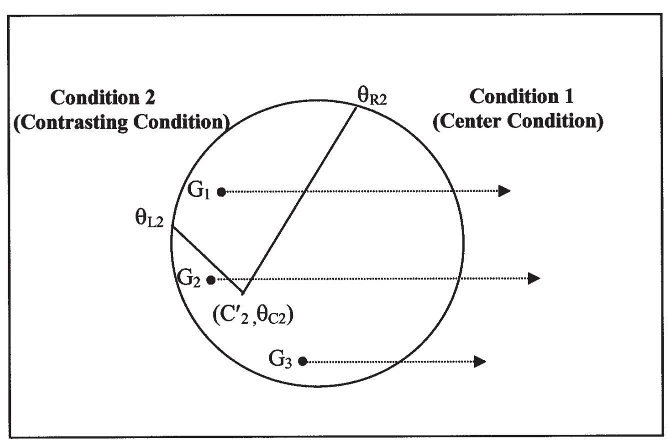

Figure 4—Projecting grid-point rays across a contrasting condition with a corner point.

Nested Boundary Segments

It is possible for one contrasting condition to be completely nested within another. Figure 5 shows a case where contrasting condition 2 (bounded by line segment $\theta_{L2}\theta_{R2}$ and arc $\theta_{L2}\theta_{R2}$) contains contrasting condition 3 (bounded by line segment $\theta_{L3}(C'_3,\theta_{C3})$, line segment $\theta_{R3}(C'_3,\theta_{C3})$, and arc $\theta_{L3}\theta_{R3}$)). When contrasting conditions are nested, rays projected from a grid point inside the nested condition result in assignment of that point to both the nested condition and the larger condition surrounding it. A check must therefore be made to determine if contrasting conditions are nested. If so, then any grid points located in both are assigned *only* to the smaller, nested condition.

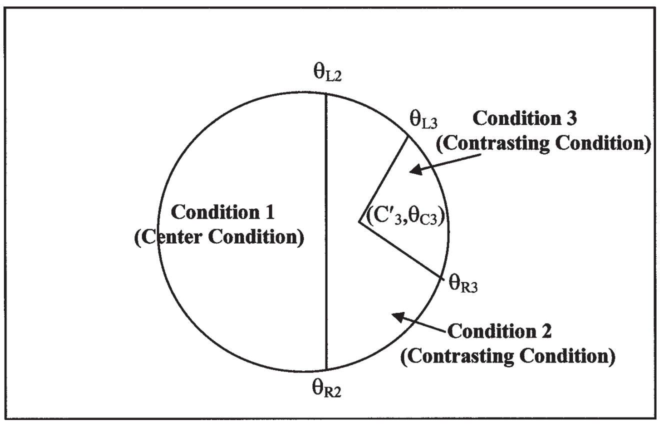

Figure 5—Example of a nested contrasting condition class (condition 3 within condition 2).

Transition Matrix

Change matrices are necessary to quantify changes in area by condition class, and to enable the partitioning of growth, removals and mortality by condition-class parameters at either the initial or terminal inventory of a measurement cycle. Visually, a condition-class transition matrix is produced by overlaying a map of each subplot at time *t* with a similar map of the same subplot at time *t+1* (fig. 6). Actually, the area associated with the intersection of all combinations of condition classes is calculated by simultaneously solving each simulated grid point for time *t* and time *t+1*. For each cell of the intersection matrix, an observation is created that includes area percent, as well as all of the condition-class variables at both time *t* and time *t + 1*. Note that it is not necessary for field crews to retain specific condition-class numbers over time. Numbers assigned to conditions remain arbitrary and are defined by the series of condition-class variables attached to them.

Algorithm Details

The algorithm used to perform the required logic checks at a given point in time consists of three main parts:

A. Determine if a ray projected from a grid point intersects a boundary arc.

B. Determine if a ray projected from a grid point intersects a boundary line segment.

C. Determine if a contrasting condition class is nested within another contrasting condition class.

There are a variety of ways these checks can be accomplished with polar coordinates and/or Cartesian coordinates. One method for each check is explained below.

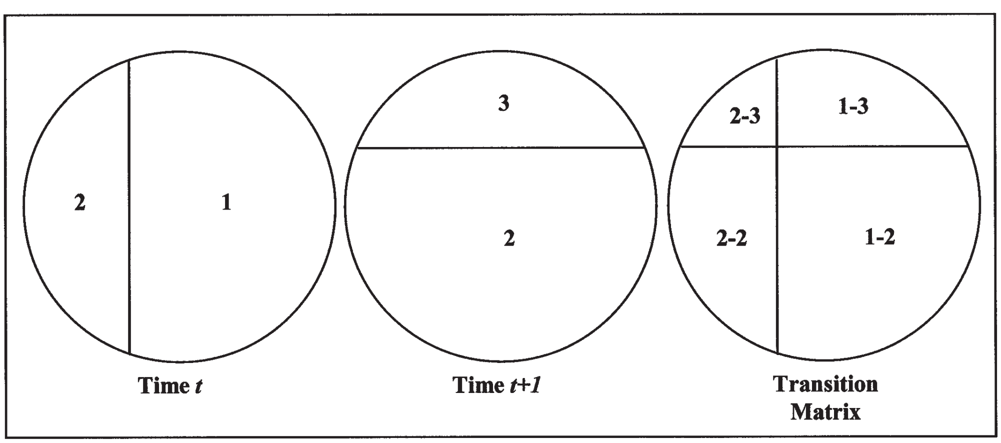

Figure 6—Example of condition classes at time *t*, time *t* + 1, and associated transition matrix.

Keep in mind that the left and right boundary points on the circle perimeter are always recorded from left to right in a clockwise direction, the maximum arc between two boundary points on the circle perimeter is 180º (unless associated with a corner point) and boundary line segments are not permitted to cross.

Determine if a projected ray intersects a boundary arc—

Given:

$\theta_L =$ the left polar angle of the contrasting condition class.

$\theta_R =$ the right polar angle of the contrasting condition class.

$\theta_C =$ the corner polar angle of the contrasting condition class, if any.

$(X_L, Y_L) =$ the Cartesian coordinates of the left boundary point on a unit circle.

$(X_R, Y_R) =$ the Cartesian coordinates of the right boundary point on a unit circle.

$(X_C, Y_C) =$ the Cartesian coordinates of the corner boundary point on a unit circle.

$(X_G, Y_G) =$ the Cartesian coordinates of the grid point to be checked.

Problem:

Determine if a ray (with 0 slope) projected to the right of grid point (X_G, Y_G) intersects the arc on a unit circle between boundary points (X_L, Y_L) and (X_R, Y_R).

Solution:

A. Determine if the arc $\theta_L\theta_R \le 180^o$.
Field protocols require that the arc between θ_L and θ_R must be $\le 180^o$ unless a corner point is involved. Additional checks are necessary if the arc exceeds 180^o. Arc $\theta_L\theta_R$ can be checked as follows:

1. arc $\theta_L\theta_R = \theta_L - \theta_R$.
2. if arc $\theta_L\theta_R < 0$ then arc $\theta_L\theta_R =$ arc $\theta_L\theta_R + 360$ (to correct for quadrant).
3. if arc $\theta_L\theta_R > 180$, go to step C.

B. Check for grid-point ray and arc intersection if arc $\theta_L\theta_R \le 180$.

If $X_L < 0$ and $X_R < 0$ the arc originates and terminates on the left side of the circle, and the projected ray cannot cross the arc. However, if $X_L \ge 0$ or $X_R \ge 0$, the following additional checks must be made.

A ray with 0 slope projected to the right of grid point (X_G, Y_G) intersects the circle at the point on the y axis where y = Y_G. To determine if Y_G is contained within the boundary arc, Y_{max} and Y_{min} must be identified,

where

Y_{max} = the maximum y-value of the boundary arc on the right half of the circle.
Y_{min} = the minimum y-value of the boundary arc on the right half of the circle.

The following logic is used to establish Y_{max} and Y_{min} and check for intersection:

1. if $X_L \le 0$ and $X_R \ge 0$ (i.e., boundary arc originates on the left side and terminates on the right side of the circle) then

   $Y_{max} = 1$ (i.e., the arc includes the top of the circle);

   and

   $Y_{min} = Y_R$.
2. if $X_L \ge 0$ (i.e., boundary arc originates on right side of circle) then $Y_{max} = Y_L$; and

   a. if $X_R \le 0$ then $Y_{min} = -1$ (i.e., the arc includes the bottom of the circle); or

   b. if $X_R \ge 0$ then $Y_{min} = Y_R$.
3. After Y_{max} and Y_{min} are established, the following logic completes the check: if $Y_{min} < Y_G < Y_{max}$ then the grid-point ray intersects a boundary arc on the right half of the circle; otherwise it does not.

C. Check for grid-point ray and arc intersection if arc $\theta_L\theta_R > 180$.

If a boundary arc exceeds 180º (due to a corner point), it is possible for a contrasting condition to have two separate arcs on the right side of the circle. For example, arc $\theta_{L2}\theta_{R2}$ in figure 7 is greater than 180º, resulting in arcs $\theta_{R2}90$ and $\theta_{L2}270$ on the right side.

1. If $X_L \ge 0$ and $X_R \ge 0$ (i.e., boundary arc originates and terminates on the right side of the circle), then there are two arcs associated with the condition. Both arcs must then be checked as described in step B3 above, after establishing their maximum and minimum y values:

   $Y_{max1} = Y_L$;

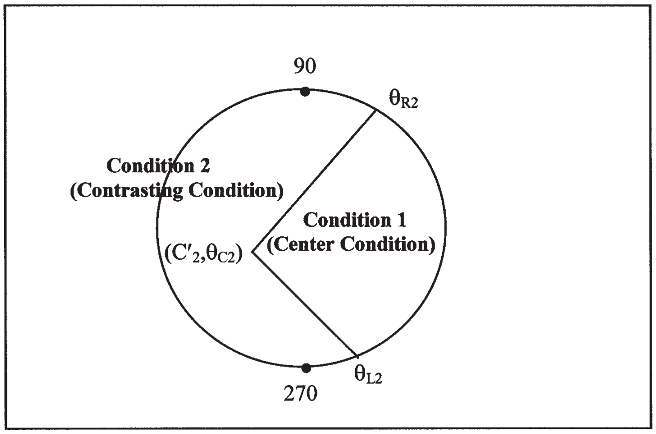

Figure 7—Example of a contrasting condition class where arc $\theta_{L2}\theta_{R2}$ > 180º, resulting in two arcs on the right side.

Y_{min1} = -1 (i.e., the arc includes the bottom of the circle);

Y_{max2} = 1 (i.e., the arc includes the top of the circle); and

$Y_{min2} = Y_R$.

2. If $X_L \geq 0$ and $X_R \leq 0$ (i.e. boundary arc originates on the right side of the circle and terminates on the left), only one arc needs to be checked, where

   $Y_{max} = Y_L$; and
   Y_{min} = -1 (i.e., the arc includes the bottom of the circle).

3. If $X_L \leq 0$ and $X_R \leq 0$ (i.e., boundary arc originates on the left side of the circle and terminates on the right), only one arc needs to be checked, where

   Y_{max} = 1 (i.e., the arc includes the top of the circle); and
   $Y_{min} = Y_R$.

4. If $X_L \leq 0$ and $X_R \leq 0$ (i.e., the arc originates and terminates on the left side of the circle), the boundary arc encompasses the entire right side of the circle, so

   Y_{max} = 1; and
   Y_{min} = -1.

5. The same logic specified in step B3 above is used to check for ray and arc intersection. If a boundary results in two arcs, both must be checked.

Determine if a projected ray intersects a boundary line segment—

Given:

(X_L, Y_L) = the Cartesian coordinates of the left boundary point on a unit circle.

(X_R, Y_R) = the Cartesian coordinates of the right boundary point on a unit circle.

(X_C, Y_C) = the Cartesian coordinates of the corner point, if any, on a unit circle.

(X_G, Y_G) = the Cartesian coordinates of the grid point to be checked.

Problem:

Determine if a horizontal ray projected to the right of grid point (X_G, Y_G) intersects the boundary line segment between boundary (X_L, Y_L) and (X_R, Y_R).

Note: if the boundary line is a segment involving a corner point, then (X_C, Y_C) would be substituted for (X_L, Y_L) and (X_R, Y_R) as appropriate.

Solution:

The equation of a line with slope m and intercept k is $y = mx + k$. In terms of Cartesian coordinates, the slope of a boundary line segment is

$$m = \frac{Y_L - Y_R}{X_L - X_R} \tag{3}$$

Once m is obtained, the intercept can be found by solving k for either pair of (x, y) coordinates

$$k = Y_L - mX_L \tag{4}$$

The x-coordinate of the intersection of any two lines (X_i) is defined by the equation

$$X_i = \frac{k_2 - k_1}{m_1 - m_2} \tag{5}$$

where

k_1 = the intercept of line 1,
m_1 = the slope of line 1,
k_2 = the intercept of line 2, and
m_2 = the slope of line 2.

The y-coordinate of the point of intersection of two lines $\left(Y_i\right)$ is defined by the equation

$$\mathrm{Y}_i = \frac{m_1 k_2 - m_2 k_1}{m_1 - m_2} \qquad (6)$$

The problem can then be solved as follows:

A. Establish the maximum and minimum y-coordinates for the boundary line segment:

$\mathrm{Y}_{\max}$ = maximum $\left(Y_L, Y_R\right)$

$\mathrm{Y}_{\min}$ = minimum $\left(Y_L, Y_R\right)$.

B. Determine the x-coordinate for the point of intersection $\left(X_i\right)$ between the boundary line and the horizontal line generated from the grid point $\left(X_G, Y_G\right)$. X_i is obtained by substituting the following values into equation (5):

$k_1 =$ the intercept of the boundary line,

$m_1 =$ the slope of the boundary line,

$k_2 = Y_G$ (the intercept of the horizontal line through the grid point), and

$m_2 = 0$ (the slope of the horizontal line through the grid point).

C. Apply the following logic to determine if X_G is to the left of the boundary line, and within the range of the y-values of the boundary line segment:

if $X_G \le X_i$ and $\mathrm{Y}_{\min} \le \mathrm{Y}_{\mathrm{G}} \le \mathrm{Y}_{\max}$, then the ray extending to the right of $\left(X_G, Y_G\right)$ and the boundary line segment intersect. Otherwise, they do not.

Note that equations (5) and (6) can also be used to ensure that two boundary line segments do not intersect within the unit circle:

if $-1 < X_i < 1$ and $-1 < Y_i < 1$, then the boundary line segments intersect inside the circle and there is an error in the field data.

Determine if contrasting condition classes are nested—

Nesting can be identified by checking the arcs of contrasting conditions on the circle perimeter.

Given:

θ_L = the left polar angle of the contrasting condition class with the longer arc on a unit circle.

θ_R = the right polar angle of the contrasting condition class with the longer arc on a unit circle.

θ'_{L} = the left polar angle of the contrasting condition class with the shorter arc on a unit circle.

θ'_{R} = the right polar angle of the contrasting condition class with the shorter arc on a unit circle.

Problem:

Determine if the arc $\theta_L \theta_R$ contains the arc $\theta'_{\mathrm{L}}\theta'_{\mathrm{R}}$.

Solution:

A. if $\theta_L < \theta_R$ then $\theta_L = \theta_L + 360$ (to correct for quadrant).

B. if $\theta'_{\mathrm{L}} < \theta'_R$ then $\theta'_L = \theta'_{\mathrm{L}} + 360$ (to correct for quadrant).

C. if $\theta_{\mathrm{L}} \ge \theta'_{\mathrm{L}} > \theta'_R \ge \theta_R$ then the arc $\theta_L \theta_R$ contains the arc $\theta'_{\mathrm{L}}\theta'_{\mathrm{R}}$,and the conditions are nested. Otherwise, they are not.

Additional considerations—It is possible that a boundary line segment recorded in the field produces a vertical line with an undefined slope (i.e., $X_L = X_R$). This problem can be circumvented by adding 5° to all azimuths recorded for all boundary segments, which has the effect of slightly rotating the subplot without changing the relative area assigned to each condition class. For any given subplot, axes must be rotated similarly at both time *t* and time *t+1* when computing change matrices.

Conclusions

A variety of mathematical and simulation techniques can be used to compute the area of mapped-plot polygons. The advantage of the simulation approach outlined here is that it can be applied to any polygon defined by a series of lines and arcs, without regard to polygon shape. This feature is especially useful when computing change, since an infinite number of shapes are possible when a subplot mapped at time *t* is overlaid with its counterpart at time *t+1*.

Literature Citations

Hahn, J.T.; MacLean, C.D.; Arner, S.L.; Bechtold, W.A. 1995. Procedures to handle inventory cluster plots that straddle two or more conditions. Forest Science Monograph. 31: 12–25.

Scott, C.T.; Bechtold, W.A. 1995. Techniques and computations for mapping plot clusters that straddle stand boundaries. Forest Science Monograph. 31: 46–61.

Bechtold, William A.; Heravi, Naser E.; Kinkenon, Matthew E. 2003. A simulation algorithm to approximate the area of mapped forest inventory plots. Gen. Tech. Rep. SRS–67. Asheville, NC: U.S. Department of Agriculture, Forest Service, Southern Research Station. 8 p.

Calculating the area of polygons associated with mapped forest inventory plots can be mathematically cumbersome, especially when computing change between inventories. We developed a simulation technique that utilizes a computer-generated dot grid and geometry to estimate the area of mapped polygons within any size circle. The technique also yields a matrix of change in mapped-plot area between two points in time.

Keywords: Area transition matrix, condition class, forest inventory, mapped plots, plot area calculation.