Digital planimetry in APLSF

ERIC TURKHEIMER and RONALD A. YEO
University of Texas, Austin, Texas

and

ERIN D. BIGLER
*University of Texas
and Austin Neurological Clinic, Austin, Texas*

The need to compute the area of complex polygons arises in diverse scientific applications. This is usually accomplished by reproducing the polygon on paper of known density and weighing it, by use of a mechanical device known as a polar planimeter, or by various methods of numerical integration. These methods all have obvious drawbacks. Weighing paper is inconvenient, inelegant, and of questionable accuracy; polar planimeters are difficult to find and cumbersome to use; numerical integration is more suited to finding the area under a function than it is to finding the area of complex closed curves, and it is often computationally slow because of the many iterations required for sufficient accuracy.

This paper describes a noniterative APL program that employs the digital algorithm on which the mechanical planimeter is based and computes the exact area of any polygon. The program is most conveniently used in conjunction with a digitizing device such as a Summagraphics Bit Pad, which writes x and y coordinates as a cursor is moved around the perimeter of a figure, but it is easily used with data generated by other means as well.

The algorithm employed by the program was originally described by the Russian mathematician Lopshitz (1963). Referring to Figure 1, notice that the area of the figure ABCD is equal to the sum of the four triangles created when each of the vertices of the polygon is connected with an interior point N. In Figure 2, in which N has been placed outside the polygon, the area of the polygon is equal to the sum of the areas of the triangles NAB, NBC, and NCD, minus the area of the triangle NDA. NDA differs from the other three triangles in that when it is traced in the order described, the pen moves clockwise; for the other triangles, the pen moves counterclockwise. Lopshitz shows that if the vertices of any polygon are connected to any point N, the area of the polygon is equal to the sum of all the counterclockwise triangles, minus the sum of all the clockwise triangles.

The area of each triangle is computed from the x and y coordinates of its vertices by using the formula

$$\frac{1}{2}[(x_2 - x_1)(y_3 - y_1) - (x_3 - x_1)(y_2 - y_1)]. \quad (1)$$

This formula results in a negative value for clockwise triangles and in a positive value for counterclockwise ones; the area of the polygon is therefore equal to the

Figure 1. Illustration of Lopshitz's technique with internal point N.

Figure 2. Illustration of Lopshitz's technique with external point N.

The authors are with the Department of Psychology, University of Texas, Austin, Texas 78712. E. D. Bigler is also at the Austin Neurological Clinic, Austin, Texas 78705.

sum of the areas of the component triangles as computed by Equation 1. The computation is simplified by always selecting the arbitrary point to be (0,0), so that Equation 1 simplifies to

$$\frac{1}{2}(x_2y_3 - x_3y_2). \quad (2)$$

The program is written in APLSF, a version of APL for Digital Equipment Corporation equipment. This version of APL does not use the standard APL character set; instead, it uses a set of ASCII character sequences (Table 1). An equivalent version, using standard APL characters, has also been provided.

The program takes as input an $N \times 2$ array of (x,y) coordinates, with each row of the array describing a vertex of the N-gon. The vertices must be consecutive around the perimeter of the polygon, in either direction. The first line of the program converts the input array to a $2N \times 2$ array, with each line of the input array repeated. This is required because each vertex of the polygon is a component of two triangles. The second line of the program reshapes the new array into an $N \times 2 \times 2$ array. Each element across the first dimension of this array is a 2×2 array of the (x,y) coordinates of one of the polygon's component triangles. The point

Table 1
Listing of Planim in ASCII and Standard APL Characters

```
.DL PLANIM[.BX].DL
.DL Z PLANIM X;V;B
[1] Z_1,RV [1]X[V,V*((.RO V),1),RO V_1O .RO X[1;1;]
[2] Z_((0.5#,.RO Z[1;1]),2,2).RO Z
[3] Z_AB 0.5#/(Z[B;1;1]#Z[B;2;2])~Z[B;2;1]#Z[B_1O (.RO Z)[1;1;2]
.DL
VPLANIM[[]]V
V Z PLANIM X;V;B
[1] Z_1#(1)X[V,V*((pV),1)pV_1O X[1;1;]
[2] Z_((0.5#pZ[1;1]),2,2)pZ
[3] Z_10.5#/(Z[B;1;1]#Z[B;2;2])~Z[B;2;1]#Z[B_1O (pZ)[1;1;2]
V
```

Figure 3. Polygon used in Example 1.

Figure 4. Polygon approximating a circle used in Example 2.

Figure 5. Polygon approximating complex shape used in Example 3.

(0,0) is implicit for each. The last line of the program computes the area of each triangle according to Equation 2, and sums them to obtain the area of the polygon.

The areas calculated by the function are exact to the extent that the figure in question is adequately described by the input polygon: some error is introduced when rounded figures are approximated by polygons. Referring to Table 2, Example 1 calculates the area of the star (each side being of unit length) depicted in Figure 3. The area of each of the 12 equilateral triangles composing the star is .433 square units, so the area of the polygon is 5.196 square units, exactly as computed by the program.

Table 2
Area Approximation

Example 1 (See Figure 3)

```

      TRIANG
0.000 1.000 1
0.866 1.500 1
0.866 0.500 1
1.732 0.000 1
0.866-0.500 1
0.866-1.500 1
0.000-1.000 1
-0.866-1.500 1
-0.866-0.500 1
-1.732 0.000 1
-0.866 0.500 1
-0.866 1.500 1

      PLANIM TRIANG

5.196

```

Example 2 (See Figure 4)

```

      RO CIRC

400 2

      PLANIM CIRC

3.14052

(3.14159-3.14052)%3.14159

0.0003405918659

```

Example 3 (See Figure 5)

```

      RO SHAPE

252 2

      PLANIM SHAPE

.2403638889

```

The unit circle in Figure 4, however, was approximated by a polygon with 400 vertices. The program's result in Example 2 is the exact area of this polygon, which differs from the area of the circle by less than .1%. The complex shape in Figure 5 was digitized to produce the input array SHAPE. The precise area of this figure is of course difficult to validate by other means, but PLANIM is as accurate with complex polygons as it is with simple

ones: Lobshitz's algorithm is independent of the complexity of the figure.

REFERENCE

LOPSHITZ, M., *Computation of the areas of oriented figures*. Boston: Heath, 1963.

(Manuscript accepted for publication May 6, 1983.)