Example

```
Function estimatePi(num)
   Variable num

   Make/O/N=(num+1) xxx,yyy
   xxx=sin(2*pi*x/num)
   yyy=cos(2*pi*x/num)

   Printf "Relative Error=%g\r",(pi-PolygonArea(xxx,yyy))/pi
End
```

See also

areaXY, **faverageXY**, **PolygonOp**

References

O'Rourke, Joseph, *Computational Geometry in C*, 2nd ed., Cambridge University Press, New York, 1998.

PolygonOp

PolygonOp [*flags*] [keyword=*value*]

The PolygonOp operation performs operations on planar polygons. Polygons and polygon paths are represented by XY pairs of waves with NaNs separating closed polygons.

The operation performed is specified using the operation keyword. The resulting polygon is returned as an XY pair of waves as specified by the /DSTX and /DSTY flags.

PolygonOp uses the clipper library by Angus Johnson. It was added in Igor Pro 9.00.

Flags

/ADDR=*addingRule*

addingRule determines how the primary and secondary polygons are added to the primary and secondary paths. The interior of the polygon is defined as the region that would be "filled" using one of the standard fill rules. The even-odd rule applies to scan lines while the other rules are based on winding counts.

addingRule is one of the following:

0:	Even-odd (default). All odd numbered sub-regions are filled, while even numbered sub-regions are not filled.
1:	Non-zero. All non-zero winding subregions are filled.
2:	Positive. All sub-regions with winding counts>0 are filled.
3:	Negative. All sub-regions with winding counts<0 are filled.

/NCFP — Disables closing the resulting polygons by connecting the last vertex to the first vertex of each output path.

/DIST=*dist* — Specifies the distance in scaled units (see the /SM flag) below which vertices are stripped. The default value is approximately sqrt(2).

/DSTX=*destX* — Specifies the X destination wave. The output is corrected for any scaling (see /SM below).

In a user-defined function, if *destX* is a simple wave name, PolygonOp automatically creates a real wave reference for the destination wave. See **Automatic Creation of WAVE References** on page IV-72 for details.

If you omit /DSTX, the X destination wave defaults to polyOpWaveX in the current data folder.

If *destX* is a simple name and you include /FREE, the X destination is created as a free wave.

/DSTY=*destY* Specifies the Y destination wave. The output is corrected for any scaling (see /SM below).

In a user-defined function, if *destY* is a simple wave name, PolygonOp automatically creates a real wave reference for the destination wave. See **Automatic Creation of WAVE References** on page IV-72 for details.

If you omit /DSTY, the Y destination wave defaults to polyOpWaveX in the current data folder.

If *destY* is a simple name and you include /FREE, the Y destination is created as a free wave.

/FA Computes the final area of the resulting polygon and stores it in V_area. This is an algebraic (signed) area in contrast to the area returned by PolygonArea which is always non-negative.

/FREE Creates the output waves specified by /DSTX and /DSTY as free waves.

/JTYP=*jointType* Specifies the joint type used when offsetting a polygon. The following joint types are supported:

0:	Square joint
1:	Round joint
2:	Miter joint

Square and round joints may give rise to additional vertices along the path.

/MLMT=*miterLimit* When the path consists of two edges connected at an acute angle, the offset could give rise to very long spikes. The miter limit sets the maximum distance in multiples of the offset that vertices can be offset from their original positions before a square joint (see /JTYP) is used. The default value for *miterLimit* is 2. The *miterLimit* parameter is scaled by /SM scaleFactor.

/OFST=*offset* Applies an offset to the final polygon. For example, if the polygon represents a rectangle, a positive offset creates an inscribing rectangle while a negative offset creates an inscribed rectangle. If the command includes a binary operation between primary and secondary polygon paths then the offset applies to the result of the binary operation. The *offset* parameter is scaled by /SM *scaleFactor*.

/Q Quiet mode. Do not print any information in the history area of the command window.

/RPRC=*rPrecision* The rounding precision *rPrecision* specifies the maximum distance in scaled units that a line segment can deviate from the exact arc. This applies only when /JTYP=1 and when offseting a polygon. The *rPrecision* parameter is scaled by /SM *scaleFactor*.

/SM={*scaleMethod* [,*scaleFactor*]}

Specifies a scaling method and a scaling factor used when converting the input data into the internal integer representation employed by the algorithm.

scaleMethod is one of the following:

0: No scaling. Floating point vertex locations are converted to integers by simple truncation:

```
Vc = trunc(Vin)
```

1: Multiplication by scaleFactor followed by truncation to integer:

```
Vc = trunc(Vin*scaleFactor)
```

2: Scaling followed by rounding to integer:

```
Vc = round(Vin*scaleFactor)
```

scaleFactor must be a finite positive number and defaults to 1.0. All scaling is reversed on output using multiplication by the reciprocal of scaleFactor. Depending on the range of your data you may be able to use a scale factor which improves the resolution of the calculation. For example, if your data are O(1) and you use a scale factor of 10 or 100 the resulting conversion into integers will be more accurate. On the other hand, if your data are O(2^30) a scale factor greater than 1 is not ideal.

/SPT={*testPointX*, *testPointY*}

Specifies a single point for testing if the point is inside the primary path polygon. The coordinates of point are scaled using the same scaling as the primary waves. The result of the test is stored in V_value as 0 if that the point is outside the polygon, 1 if that the point is inside the polygon, and 2 if the point is on the boundary of the polygon.

/Z Suppresses error reporting. If you use /Z, check the V_Flag output variable to see if the operation succeeded.

Keywords

operation=*op*

You can omit the operation=op pair when you are applying an offset to the primary polygon or when testing for pointsInPoly. Otherwise, *op* is one of the following:

opN is one of the following:

Intersection Computes the intersection of the region defined by the primary path or paths and the region defined by the secondary path or paths.

Union Computes the union between the region defined by the primary path or paths and the region defined by the secondary path or paths.

Difference Computes the difference between the region defined by the primary path or paths and the region defined by the secondary path or paths.

XOR Computes the exclusive OR between the regions defined by the primary path or paths and the regions defined by the secondary path or paths. This includes all regions where either the primary or secondary polygons are filled but not where both are filled.

PointInPoly Tests if the points specified by *pointsWaves* are inside the target polygon. The results are stored in the unsigned byte wave W_inPoly in the current data folder: 0 for a point outside, 1 for a point inside, and 2 for a point on the boundary.

The target polygon is defined by *primaryWaves*. The tested points are internally scaled with the same scaling method (/SM) that applies to primary waves. You can perform this test on a target polygon that is a result of any binary polygon operation if you only specify the *pointsWaves*, for example:

```
PolygonOp operation=union,
primaryWaves={wx,wy}, secondarywaves={sx,sy},
pointsWaves={px,py}
```

Area Computes the polygon area and stores it in V_area when

CleanPolygon Removes vertices from the primary paths that satisfy one of the following conditions:

1. The vertices lie on a co-linear edge. For every three vertices the middle vertex is removed if it lies close enough (see /DIST above) to the edge connecting the vertices on both of its sides.

2. The vertices are situated below a critical distance from an adjacent vertex (see /DIST above).

See also the Simplify operation.

MinkowskiDiff If the primary and secondary paths represent sets of vectors, the Minkowski difference of the two sets is defined as the set formed by subtracting each vector in the primary set from each vector in the secondary set.

MinkowskiSum If the primary and secondary paths represent sets of vectors, the Minkowski sum of the two sets is defined as the set formed by adding each vector in the primary set to each vector in the secondary set.

Simplify Removes self-intersections from the input polygons. In case of touching vertices the resulting polygon is split into two polygons.

Reverse Reverses the polygon paths.

pointsWaves={*pointsXWave, pointsYWave*}