

$A = \iint dx dy$ $A = \oint ds$
 $x_c A = \iint x dx dy$ $x_c A = \oint$
 $I_{xx} = \iint x^2 dx dy$ $I_{xx} = \oint$

Area, Centroid and Area Moments for Polygonal Objects

Summations shown below are for closed CCW boundary sequences. CW boundary sequences will produce negative values for area and moments. Closed boundaries require $x_{n+1} = x_1$.

The term a_i is twice the signed area of the elementary triangle formed by (x_i, y_i) and (x_{i+1}, y_{i+1}) and the origin.

For improved accuracy, a temporary local origin at the mean of the boundary points should be used.

$a_i = x_i y_{i+1} - x_{i+1} y_i$

$A = \frac{1}{2} \sum_1^n a_i$ units

$x_c = \frac{1}{6A} \sum_1^n a_i (x_i + x_{i+1})$

$y_c = \frac{1}{6A} \sum_1^n a_i (y_i + y_{i+1})$

$I_{xx} = \frac{1}{12} \sum_1^n a_i (y_i^2 + y_i y_{i+1} + y_{i+1}^2)$ units cm^4

$I_{yy} = \frac{1}{12} \sum_1^n a_i (x_i^2 + x_i x_{i+1} + x_{i+1}^2)$

$I_{xy} = \frac{1}{24} \sum_1^n a_i (x_i y_{i+1} + 2x_i y_i + 2x_{i+1} y_{i+1} + x_{i+1} y_i)$

perimeter = $\sum_1^n \sqrt{(x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2}$

centroidal moments

$I_{uu} = I_{xx} - A y_c^2$

$I_{vv} = I_{yy} - A x_c^2$

$I_{uv} = I_{xy} - A x_c y_c$

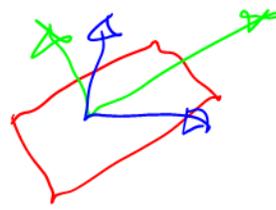
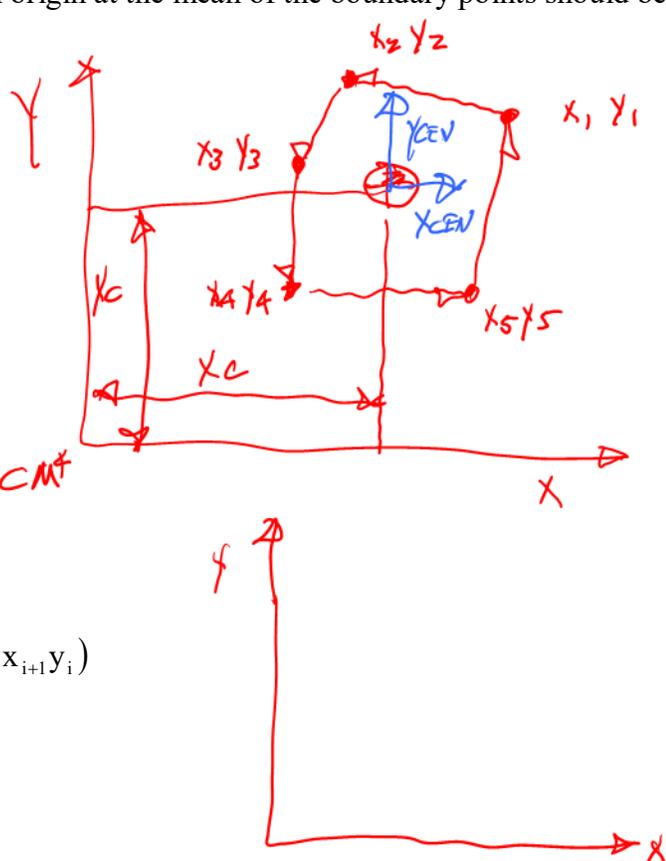
$J = I_{uu} + I_{vv} = I_1 + I_2$

principal moments

$I_1, I_2 = (I_{uu} + I_{vv}) / 2 \pm \sqrt{(I_{uu} - I_{vv})^2 / 4 + I_{uv}^2}$

$\tan 2\theta = 2I_{uv} / (I_{vv} - I_{uu})$

$I_1, I_2 = \text{eig} \begin{bmatrix} I_{uu} & -I_{uv} \\ -I_{uv} & I_{vv} \end{bmatrix}$



Area moments about x and y axes

cm^2, m^2

Object with Holes

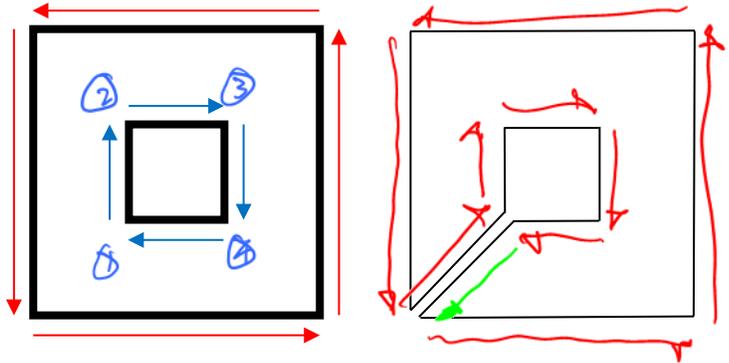
- 1) Digitize the outline of the object in the CCW direction. Be certain to close the outline (i.e. the first and last points must be the same).
- 2) Digitize outlines of holes in the CW direction. Be certain to close the outlines (i.e. the first and last points must be the same).
- 3) Append the data strings. Remember to add the first point in the outline to the end of each string.
- 4) Repeat steps 2) and 3) for multiple holes. For example, an object with three holes will contain a closed CCW outline for the object, followed by three closed CW outlines, one for each hole. Outlines for holes must be separated by the first point for the outline.
- 5) Area, centroid and moment computations will be correct. Perimeter will NOT be correct.

Sample data for figure at right

```
x_outline = [ 0 3 3 0 0 ];
y_outline = [ 0 0 3 3 0 ];
```

```
x_hole = [ 1 1 2 2 1 ];
y_hole = [ 1 2 2 1 1 ];
```

```
x = [ x_outline x_hole x_outline(1) ];
y = [ y_outline y_hole y_outline(1) ];
```



Sample data for three holes

```
x = [ x_outline x_holeA x_outline(1) x_holeB x_outline(1) x_holeC x_outline(1) ];
y = [ y_outline y_holeA y_outline(1) y_holeB y_outline(1) y_holeC y_outline(1) ];
```

```
% test_polygeom.m - test polygeom
% area, centroid, perimeter and area moments of polygonal outline
% H.J. Sommer III - 16.12.09 - tested under MATLAB v9.0
```

```
clear
```

```
% constants
d2r = pi / 180;
```

```
% 3x5 test rectangle with long axis at 30 degrees
% area=15, x_cen=3.415, y_cen=6.549, perimeter=16
% I1=11.249, I2=31.247, J=42.496
x = [ 2.000  0.500  4.830  6.330 ]';
y = [ 4.000  6.598  9.098  6.500 ]';
```

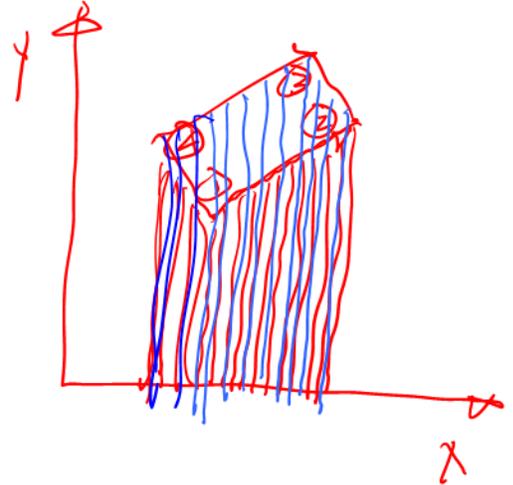
```
% get geometry
[ geom, iner, cpmo ] = polygeom( x, y );
```

```
% show results
area = geom(1);
x_cen = geom(2);
y_cen = geom(3);
perimeter = geom(4);
disp( [ ' ' ] )
disp( [ '3x5 test rectangle with long axis at 30 degrees' ] )
disp( [ ' ' ] )
disp( [ '   area   x_cen   y_cen   perim' ] )
disp( [ area x_cen y_cen perimeter ] )
```

```
I1 = cpmo(1);
angle1 = cpmo(2);
I2 = cpmo(3);
angle2 = cpmo(4);
disp( [ ' ' ] )
disp( [ '   I1       I2' ] )
disp( [ I1 I2 ] )
disp( [ ' angle1   angle2' ] )
disp( [ angle1/d2r angle2/d2r ] )
```

```
% plot outline
xplot = x( [ 1:end 1 ] );
yplot = y( [ 1:end 1 ] );
rad = 10;
x1 = [ x_cen-rad*cos(angle1)  x_cen+rad*cos(angle1) ];
y1 = [ y_cen-rad*sin(angle1)  y_cen+rad*sin(angle1) ];
x2 = [ x_cen-rad*cos(angle2)  x_cen+rad*cos(angle2) ];
y2 = [ y_cen-rad*sin(angle2)  y_cen+rad*sin(angle2) ];
plot( xplot,yplot,'b', x_cen,y_cen,'ro', ...
      x1,y1,'g:', x2,y2,'g:' )
axis( [ 0 rad 0 rad ] )
axis square
```

```
% bottom of test_polygeom.m
```



$$A = \int_{x_1}^{x_2} y \, dx$$

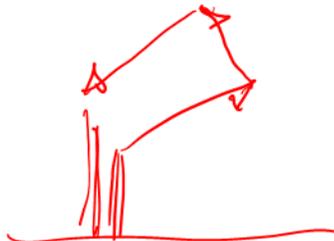
$$A = \int_{x_2}^{x_3} y \, dx \quad dx < 0$$

$$A = \int_{x_3}^{x_4} y \, dx \quad dx < 0$$

$$A = \int_{x_4}^{x_1}$$

$$f_5 = x_1$$

sum = area of object



```

function [ geom, iner, cpmo ] = polygeom( x, y )
%POLYGEOM Geometry of a planar polygon
%
% POLYGEOM( X, Y ) returns area, X centroid,
% Y centroid and perimeter for the planar polygon
% specified by vertices in vectors X and Y.
%
% [ GEOM, INER, CPMO ] = POLYGEOM( X, Y ) returns
% area, centroid, perimeter and area moments of
% inertia for the polygon.
%
% GEOM = [ area X_cen Y_cen perimeter ]
% INER = [ Ixx Iyy Ixy Iuu Ivv Iuv ]
% u,v are centroidal axes parallel to x,y axes.
%
% CPMO = [ I1 ang1 I2 ang2 J ]
% I1,I2 are centroidal principal moments about axes
% at angles ang1,ang2.
% ang1 and ang2 are in radians.
% J is centroidal polar moment. J = I1 + I2 = Iuu + Ivv

% H.J. Sommer III - 16.12.09 - tested under MATLAB v9.0
%
% sample data
% x = [ 2.000 0.500 4.830 6.330 ]';
% y = [ 4.000 6.598 9.098 6.500 ]';
% 3x5 test rectangle with long axis at 30 degrees
% area=15, x_cen=3.415, y_cen=6.549, perimeter=16
% Ixx=659.561, Iyy=201.173, Ixy=344.117
% Iuu=16.249, Ivv=26.247, Iuv=8.660
% I1=11.249, ang1=30deg, I2=31.247, ang2=120deg, J=42.496
%
% H.J. Sommer III, Ph.D., Professor of Mechanical Engineering, 337 Leonhard Bldg
% The Pennsylvania State University, University Park, PA 16802
% (814)863-8997 FAX (814)865-9693 hjs1-at-psu.edu www.mne.psu.edu/sommer/

% begin function POLYGEOM

% check if inputs are same size
if ~isequal( size(x), size(y) ),
    error( 'X and Y must be the same size' );
end

% temporarily shift data to mean of vertices for improved accuracy
xm = mean(x);
ym = mean(y);
x = x - xm;
y = y - ym;

% summations for CCW boundary
xp = x( [2:end 1] );
yp = y( [2:end 1] );
a = x.*yp - xp.*y;

A = sum( a ) / 2;
xc = sum( (x+xp).*a ) / 6/A;
yc = sum( (y+yp).*a ) / 6/A;
Ixx = sum( (y.*y + y.*yp + yp.*yp).*a ) / 12;
Iyy = sum( (x.*x + x.*xp + xp.*xp).*a ) / 12;
Ixy = sum( (x.*yp + 2*x.*y + 2*xp.*yp + xp.*y).*a ) / 24;

dx = xp - x;
dy = yp - y;
P = sum( sqrt( dx.*dx + dy.*dy ) );

% check for CCW versus CW boundary
if A < 0,
    A = -A;
    Ixx = -Ixx;
    Iyy = -Iyy;
    Ixy = -Ixy;
end

```

Area X_CEN Y_CEN
perimeter

MOMENT TERMS
J AREA

cut/paste into
polygeom.m

$M = \rho \times \text{Area}$

$J_{G-MASS} = \rho \times J_{AREA}$

J_{AREA} units cm^4 m^4
 J_{G-MASS} units $\text{kg}\cdot\text{cm}^2$ $\text{kg}\cdot\text{m}^2$

$\rho \times J_{AREA} = \frac{\text{kg}}{\text{cm}^3} \text{cm}^4$
 $= \text{kg}\cdot\text{cm}^2$

```
% centroidal moments
Iuu = Ixx - A*yc*yc;
Ivv = Iyy - A*xc*xc;
Iuv = Ixy - A*xc*yc;
J = Iuu + Ivv;

% replace mean of vertices
x_cen = xc + xm;
y_cen = yc + ym;
Ixx = Iuu + A*y_cen*y_cen;
Iyy = Ivv + A*x_cen*x_cen;
Ixy = Iuv + A*x_cen*y_cen;

% principal moments and orientation
I = [ Iuu -Iuv ;
      -Iuv Ivv ];
[ eig_vec, eig_val ] = eig(I);
I1 = eig_val(1,1);
I2 = eig_val(2,2);
ang1 = atan2( eig_vec(2,1), eig_vec(1,1) );
ang2 = atan2( eig_vec(2,2), eig_vec(1,2) );

% return values
geom = [ A x_cen y_cen P ];
iner = [ Ixx Iyy Ixy Iuu Ivv Iuv ];
cpmo = [ I1 ang1 I2 ang2 J ];

% bottom of polygeom
```