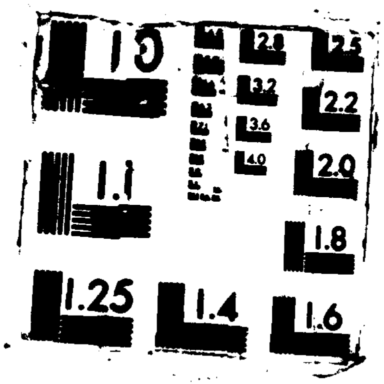
AD-A183 444 CALCULATION OF THE MOMENTS OF POLYGONS(U) DEFENCE RESEARCH ESTABLISHMENT SUFFIELD RALSTON (ALBERTA) D HALLY JUN 87 DREA-TM-87/209 1/1

UNCLASSIFIED F/G 12/2 NL


1.0
2.8
2.5
3.2
2.2
3.6
4.0
2.0
1.1
1.8
1.25
1.4
1.6

UNLIMITED DISTRIBUTION

(1)

National Defence
Research and Development Branch

Défense nationale
Bureau de recherche et développement

TECHNICAL MEMORANDUM 87/209

JUNE 1987

CALCULATION OF THE MOMENTS OF POLYGONS

David Hally

Defence Research Establishment Atlantic

Centre de Recherches pour la Défense Atlantique

DEFENSIO PER SCIENTIAM

Canada

DEFENCE RESEARCH ESTABLISHMENT ATLANTIC
9 GROVE STREET
P O BOX 1012
DARTMOUTH N S
B2Y 3Z7

TELEPHONE
(902) 426 3100

CENTRE DE RECHERCHES POUR LA DÉFENSE ATLANTIQUE
9 GROVE STREET
C P 1012
DARTMOUTH N E
B2Y 3Z7

UNLIMITED DISTRIBUTION

National Defence Research and Development Branch

Défense nationale Bureau de recherche et développement

CALCULATION OF THE MOMENTS OF POLYGONS

David Hally

June 1987

Approved by B.F. Peters A/Director/Technology Division

DISTRIBUTION APPROVED BY

A/D/TD

TECHNICAL MEMORANDUM 87/209

Defence Research Establishment Atlantic

Centre de Recherches pour la Défense Atlantique

Canada

Abstract

Methods for calculating the moments of an arbitrary polygon and for determining whether a point lies within a polygon are derived and discussed. The methods are efficient, robust, concise, and easily programmed in any computer language. A FORTRAN 77 subroutine which calculates the first three moments of an arbitrary polygon is also included, as is a subroutine which determines whether a point lies in an arbitrary polygon.

Résumé

On traite des méthodes permettant de calculer les moments d'un polygone arbitraire et de déterminer si un point est situé à l'intérieur d'un polygone. Les méthodes sont efficaces, solides, concises et faciles à programmer dans n'importe quel langage informatique. On présente aussi un sous-programme en FORTRAN 77 qui calcule les trois premiers moments d'un polygone arbitraire, de même qu'un sous-programme qui détermine si un point est situé à l'intérieur d'un polygone arbitraire.

Contents

List of Figures

Notation

d_k: The distance of the origin to the k^{th} side of the polygon.

$\mathbf{I}^{(n)}$: The n^{th} moment tensor for the polygon.

$I_x^{(1)}, I_y^{(1)}$: Components of the first moment of the polygon.

$I_{xx}^{(2)}, I_{xy}^{(2)}, I_{yy}^{(2)}$: Components of the second moment of the polygon.

$[n]$: The largest integer which does not exceed n.

$\hat{n}$: Outward pointing unit normal.

N: The number of sides of the polygon.

$P_{m,n}(\mathbf{x},\mathbf{y})$: Tensor permutation function defined in Section 2.1.

$\mathrm{sgn}(x)$: Function defined in equation (3.1).

$\mathbf{x}$: Position vector.

$\mathbf{x}^n$: The tensor dyadic $\overbrace{\mathbf{xxx}\ldots\mathbf{x}}^{n\ \text{times}}$.

s: Arclength around the polygon perimeter.

t: Variable used to parameterize a side of the polygon.

x, y: Coordinates.

$\hat{x}, \hat{y}, \hat{z}$: Unit vectors in the coordinate directions.

x_k, y_k: Coordinates of the k^{th} vertex of a polygon.

$\delta(x)$: Dirac delta function.

Bold face characters are reserved for vectors and tensors.

1 Introduction

A problem that arises in many engineering applications is to find some moment (e.g the area or the centroid) of an arbitrary polygon. At DREA the problem has arisen in the context of the calculation of potential flows via panel methods[1]. At large distances, the potential due to a panel is determined most efficiently by a multipole expansion in which the moments of the polygonal panel appear. In this memorandum an efficient, robust, and easily programmed method for calculating the moments of a polygon is derived. Although it seems likely that the method has been derived previously, it does not appear to be widely known. Indeed, the problem was considered suitable (though it was not discussed) for a seminar at the Mathematics Dept. of Dalhousie University in which problems of *unknown* solution were to be tackled[2]. Further evidence comes from the potential flow program EN967[3] which calculates the moments of quadrilateral panels using a method which is less efficient, less robust, and less general than the one presented.

A simple and efficient solution to a related problem is also derived in this memorandum: how does one determine whether a given point lies inside or outside an arbitrary polygon? It, too, has arisen in the context of potential flow panel methods; it is sometimes necessary to know whether a certain point lies within a panel. This problem has also arisen in modelling of a "cycle of perception" for a computational vision problem by the Computer Aided Detection Group at DREA[4].

Appendix A contains a FORTRAN subroutine which calculates the first three moments of an arbitrary polygon using the method discussed in this memorandum. Appendix B contains a FORTRAN subroutine which determines whether a point lies inside or outside an arbitrary polygon.

2 Calculation of the Moments of Polygons

In this section the method of calculation of the moments of an arbitrary polygon is derived and discussed.

2.1 Analytical Formulae for the Moments

Let (x, y) be a coordinate system with unit vectors $\hat{x}$ and $\hat{y}$ along its axes. Bold face characters will be used to denote vectors and tensors. Thus,

$$\mathbf{x} \equiv x\hat{x} + y\hat{y} \tag{2.1}$$

The notation $\mathbf{x}^n$ will be used to denote the tensor dyadic

$$\mathbf{x}^n \equiv \overbrace{\mathbf{xxx}\ldots\mathbf{x}}^{n \text{ times}} \tag{2.2}$$

The problem to be solved may be stated as follows:

Problem 1: *If $\mathbf{x}_k, k = 1, \ldots, N$ are the vertices of a polygon in order as one proceeds around its perimeter counterclockwise, calculate the n^{th} moment of the polygon,*

$$\mathbf{I}^{(n)} \equiv \int_{\Box} \mathbf{x}^n dx dy \tag{2.3}$$

where the notation $\Box$ denotes integration over the surface of the polygon.

$\mathbf{I}^{(0)}$ is the area of the polygon and $\mathbf{I}^{(1)}$ is its centroid times its area. If the polygon has uniform density, the second order tensor $\mathbf{I}^{(2)}$ is proportional to its moment of inertia.

The essence of the method is to express the $\mathbf{x}^n$ as the divergence of a tensor, so that the divergence theorem can be used to express the moment as a line integral around the perimeter of the polygon. The contribution to the line integral from each side is calculated easily. By using tensor notation, one can obtain a single expression for any moment of the polygon.

First, note that

$$\begin{aligned}
\nabla \cdot \mathbf{x}^n &= \frac{\partial(x\mathbf{x}^{n-1})}{\partial x} + \frac{\partial(y\mathbf{x}^{n-1})}{\partial y} \\
&= 2\mathbf{x}^{n-1} + x\frac{\partial \mathbf{x}^{n-1}}{\partial x} + y\frac{\partial \mathbf{x}^{n-1}}{\partial y} \\
&= 2\mathbf{x}^{n-1} + \mathbf{x} \cdot \nabla \mathbf{x}^{n-1}
\end{aligned} \tag{2.4}$$

Now, since $\mathbf{x} \cdot \nabla \mathbf{x} = \mathbf{x}$, one has $\mathbf{x} \cdot \nabla \mathbf{x}^n = n\mathbf{x}^{n-1}$, whence from equation (2.4)

$$\nabla \cdot \mathbf{x}^n = (n+1)\mathbf{x}^{n-1} \tag{2.5}$$

Therefore, using equation (2.5), the divergence theorem, and the definition of equation (2.3), one obtains

$$\mathbf{I}^{(n)} = \frac{1}{n+2} \int_{\square} \nabla \cdot \mathbf{x}^{n+1} dx dy = \frac{1}{n+2} \int_{\partial \square} \hat{n} \cdot \mathbf{x}^{n+1} ds \tag{2.6}$$

where $\partial \square$ denotes the perimeter of the polygon, $\hat{n}$ is an outward pointing unit normal, and ds is an increment of arclength.

The k^{th} side of the panel may be parameterized by $\mathbf{x} = [(\mathbf{x}_{k+1} + \mathbf{x}_k) + t(\mathbf{x}_{k+1} - \mathbf{x}_k)]/2$, $t \in [-1, 1]$. The increment of arclength is then $ds = |\mathbf{x}_{k+1} - \mathbf{x}_k| dt/2$. The outward pointing normal is parallel to $(\mathbf{x}_{k+1} - \mathbf{x}_k) \times \hat{z}$ where $\hat{z}$ is a unit vector perpendicular to the plane of the polygon and such that $\hat{x}$, $\hat{y}$, and $\hat{z}$ define a right handed coordinate system. Thus, $\hat{n} ds = (\mathbf{x}_{k+1} - \mathbf{x}_k) \times \hat{z}\, dt/2$ so that

$$\begin{aligned}
\mathbf{I}^{(n)} &= \frac{1}{n+2} \sum_{k=1}^{N} \int_{-1}^{1} \frac{[(\mathbf{x}_{k+1} - \mathbf{x}_k) \times \hat{z}]}{2} \cdot \left[\frac{(\mathbf{x}_{k+1} + \mathbf{x}_k) + t(\mathbf{x}_{k+1} - \mathbf{x}_k)}{2} \right]^{n+1} dt \\
&= \frac{1}{(n+2)2^{n+2}} \sum_{k=1}^{N} \int_{-1}^{1} [(\mathbf{x}_{k+1} - \mathbf{x}_k) \times \hat{z}] \cdot [(\mathbf{x}_{k+1} + \mathbf{x}_k) + t(\mathbf{x}_{k+1} - \mathbf{x}_k)]^{n+1} dt \quad (2.7)
\end{aligned}$$

In these expressions a subscript of $N+1$ is equivalent to the subscript 1: that is, $\mathbf{x}_{N+1} \equiv \mathbf{x}_1$. Now,

$$\begin{aligned}
[(\mathbf{x}_{k+1} - \mathbf{x}_k) \times \hat{z}] \cdot [(\mathbf{x}_{k+1} + \mathbf{x}_k) + t(\mathbf{x}_{k+1} - \mathbf{x}_k)] &= [(\mathbf{x}_{k+1} - \mathbf{x}_k) \times \hat{z}] \cdot (\mathbf{x}_{k+1} + \mathbf{x}_k) \\
&= [(\mathbf{x}_{k+1} + \mathbf{x}_k) \times (\mathbf{x}_{k+1} - \mathbf{x}_k)] \cdot \hat{z} \\
&= 2(\mathbf{x}_k \times \mathbf{x}_{k+1}) \cdot \hat{z} \\
&= 2(x_k y_{k+1} - x_{k+1} y_k) \quad (2.8)
\end{aligned}$$

and therefore,

$$\mathbf{I}^{(n)} = \sum_{k=1}^{N} \frac{(x_k y_{k+1} - x_{k+1} y_k)}{(n+2)2^{n+1}} \int_{-1}^{1} [(\mathbf{x}_{k+1} + \mathbf{x}_k) + t(\mathbf{x}_{k+1} - \mathbf{x}_k)]^n dt \tag{2.9}$$

The tensor dyadic $(\mathbf{x} + \mathbf{y})^n$ can be expanded, but one must be careful not to use the binomial theorem which assumes commutativity of $\mathbf{x}$ and $\mathbf{y}$ in the terms: for example, $\mathbf{xy} \neq \mathbf{yx}$. Rather,

$$(\mathbf{x} + \mathbf{y})^n = \sum_{m=0}^{n} P_{n-m,m}(\mathbf{x}, \mathbf{y}) \tag{2.10}$$

where $P_{n,m}(\mathbf{x}, \mathbf{y})$ is the sum of all terms which are permutations of n copies of $\mathbf{x}$ and m copies of $\mathbf{y}$: for example,

$$P_{2,1}(\mathbf{x}, \mathbf{y}) = \mathbf{xxy} + \mathbf{xyx} + \mathbf{yxx} \tag{2.11}$$

The definition for $P_{n,m}$ is extended to the case $P_{0,0}$ by defining $P_{0,0} = 1$.

Substitution of equation (2.10) into equation (2.9) yields

$$
\begin{aligned}
\mathbf{I}^{(n)} &= \sum_{k=1}^{N} \frac{(x_k y_{k+1} - x_{k+1} y_k)}{(n+2)2^{n+1}} \int_{-1}^{1} \sum_{m=0}^{n} P_{n-m,m}(\mathbf{x}_{k+1} + \mathbf{x}_k, \mathbf{x}_{k+1} - \mathbf{x}_k) t^m \, dt \\
&= \sum_{k=1}^{N} \frac{(x_k y_{k+1} - x_{k+1} y_k)}{(n+2)2^{n}} \sum_{m=0}^{[n/2]} \frac{P_{n-2m,2m}(\mathbf{x}_{k+1} + \mathbf{x}_k, \mathbf{x}_{k+1} - \mathbf{x}_k)}{2m+1} \qquad (2.12)
\end{aligned}
$$

where $[n/2]$ denotes the largest integer not exceeding $n/2$.

The expression in equation (2.12), though seemingly complicated, yields simple expressions for the first few moments. In particular,

$$
\mathbf{I}^{(0)} = \frac{1}{2} \sum_{k=1}^{N} (x_k y_{k+1} - x_{k+1} y_k) \qquad (2.13)
$$

$$
\mathbf{I}^{(1)} = \frac{1}{6} \sum_{k=1}^{N} (x_k y_{k+1} - x_{k+1} y_k)(\mathbf{x}_{k+1} + \mathbf{x}_k) \qquad (2.14)
$$

$$
\begin{aligned}
\mathbf{I}^{(2)} &= \frac{1}{16} \sum_{k=1}^{N} (x_k y_{k+1} - x_{k+1} y_k)[(\mathbf{x}_{k+1} + \mathbf{x}_k)^2 + (\mathbf{x}_{k+1} - \mathbf{x}_k)^2/3] \\
&= \frac{1}{24} \sum_{k=1}^{N} (x_k y_{k+1} - x_{k+1} y_k)[2\mathbf{x}_{k+1}^2 + \mathbf{x}_{k+1}\mathbf{x}_k + \mathbf{x}_k \mathbf{x}_{k+1} + 2\mathbf{x}_k^2] \qquad (2.15)
\end{aligned}
$$

Alternatively, the components of the centroid and the second moment of area can be written

$$
I_x^{(1)} = \frac{1}{6} \sum_{k=1}^{N} (x_k y_{k+1} - x_{k+1} y_k)(x_{k+1} + x_k) \qquad (2.16)
$$

$$
I_y^{(1)} = \frac{1}{6} \sum_{k=1}^{N} (x_k y_{k+1} - x_{k+1} y_k)(y_{k+1} + y_k) \qquad (2.17)
$$

$$
I_{xx}^{(2)} = \frac{1}{12} \sum_{k=1}^{N} (x_k y_{k+1} - x_{k+1} y_k)[x_{k+1}^2 + x_{k+1} x_k + x_k^2] \qquad (2.18)
$$

$$
I_{xy}^{(2)} = \frac{1}{24} \sum_{k=1}^{N} (x_k y_{k+1} - x_{k+1} y_k)[2x_{k+1} y_{k+1} + x_{k+1} y_k + x_k y_{k+1} + 2x_k y_k] \qquad (2.19)
$$

$$
I_{yy}^{(2)} = \frac{1}{12} \sum_{k=1}^{N} (x_k y_{k+1} - x_{k+1} y_k)[y_{k+1}^2 + y_{k+1} y_k + y_k^2] \qquad (2.20)
$$

These formulae are quite general, correctly calculating the moments of polygons with arbitrary connectivity. For example, Figure 1 indicates a correct ordering of vertices for calculating the moments of two disconnected squares while Figure 2 indicates a correct ordering of vertices to be used to calculate the moments of a square containing a square hole. Interior holes must be traversed clockwise.

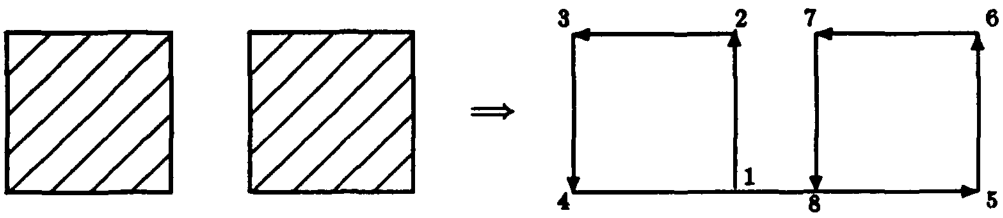


Figure 1: Order of vertices for disconnected polygons.

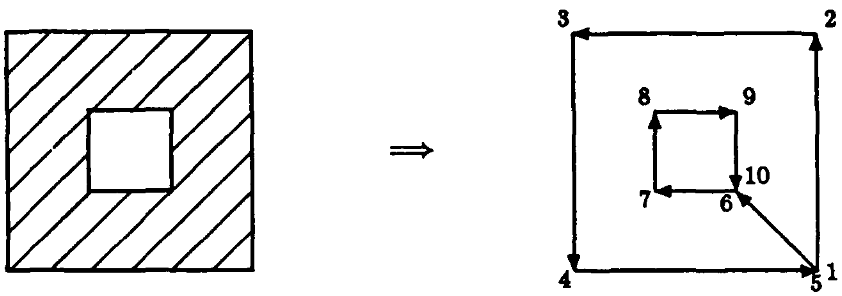


Figure 2: Order of vertices for polygons with holes.

2.2 Avoiding Round-off Errors

When using the formulae derived in the previous section for numerical calculations, care must be taken to avoid round-off error in the term $(x_k y_{k+1} - x_{k+1} y_k)$ when the origin of the coordinate system is many mean polygon diameters from the centroid of the panel. In this case $x_k y_{k+1} \approx x_{k+1} y_k$ and the term $(x_k y_{k+1} - x_{k+1} y_k)$ is the difference of two large, nearly equal numbers. Given any point $\mathbf{x}_0$ which is close to the polygon, these round-off errors can be avoided in two ways:

1. by using the expression

$$x_k y_{k+1} - x_{k+1} y_k = (x_k - x_0)(y_{k+1} - y_0) - (x_{k+1} - x_0)(y_k - y_0) + x_0(y_{k+1} - y_k) - y_0(x_{k+1} - x_k) \tag{2.21}$$

   whose right hand side does not contain the product of two large numbers; or

2. by first shifting the coordinate origin to $\mathbf{x}_0$, calculating the moments, then using the formulae

$$\begin{aligned}
\mathbf{I}^{(0)} &= \mathbf{I}_0^{(0)} && (2.22)\\
\mathbf{I}^{(1)} &= \int_\square \mathbf{x}\,dxdy = \int_\square (\mathbf{x} - \mathbf{x}_0)\,dxdy + \int_\square \mathbf{x}_0\,dxdy = \mathbf{I}_0^{(1)} + \mathbf{x}_0 \mathbf{I}^{(0)} && (2.23)\\
\mathbf{I}^{(2)} &= \int_\square \mathbf{x}^2\,dxdy \\
&= \int_\square (\mathbf{x} - \mathbf{x}_0)^2\,dxdy + \int_\square \mathbf{x}\mathbf{x}_0\,dxdy + \int_\square \mathbf{x}_0\mathbf{x}\,dxdy - \int_\square \mathbf{x}_0^2\,dxdy \\
&= \mathbf{I}_0^{(2)} + \mathbf{x}_0 \mathbf{I}^{(1)} + \mathbf{I}^{(1)}\mathbf{x}_0 - \mathbf{x}_0^2 \mathbf{I}^{(0)} && (2.24)
\end{aligned}$$

   where the subscript 0 denotes a moment about $\mathbf{x}_0$.

Since the second method requires fewer arithmetic operations, it was used in the code provided in Appendix A. The first vertex $\mathbf{x}_1$ was chosen for $\mathbf{x}_0$.

Timing comparisons on the DREA DEC-20/60 computer suggest that the penalty in run time paid for avoiding round-off errors is between 10% and 15% for polygons with small number of vertices. If speed is of the essence and it is known that the origin of the coordinate system will always lie near the panel, it may be best not to worry about round-off errors. On the other hand, the following example (run on the DREA DEC-20/60) serves to illustrate the need in general.

With $N = 4$, $x_1 = (0,0)$, $x_2 = (1,0)$, $x_1 = (1,1)$, and $x_1 = (0,1)$, the code given in Appendix A and similar code ignoring round-off errors both returned the following values for the moments (correct to 7 significant figures):

$$I^{(0)} = 1.0000000 \quad I_x^{(1)} = 0.5000000 \quad I_y^{(1)} = 0.5000000 \tag{2.25}$$

$$I_{xx}^{(2)} = 0.3333333 \quad I_{xy}^{(2)} = 0.2500000 \quad I_{yy}^{(2)} = 0.3333333 \tag{2.26}$$

However, when each vertex was displaced by adding $(10^5, 10^5)$ to it, the code of Appendix A returned

$$I^{(0)} = 1.000000 \quad I_x^{(1)} = 1.000005 \times 10^5 \quad I_y^{(1)} = 1.000005 \times 10^{10} \tag{2.27}$$

$$I_{xx}^{(2)} = I_{xy}^{(2)} = I_{yy}^{(2)} = 1.000010 \times 10^{10} \tag{2.28}$$

which are again correct to 7 significant figures. However, the code ignoring round-off errors returned

$$I^{(0)} = 128.00000 \quad I_x^{(1)} = I_y^{(1)} = 8.566699 \times 10^6 \tag{2.29}$$

$$I_{xx}^{(2)} = I_{xy}^{(2)} = I_{yy}^{(2)} = 6.450141 \times 10^{11} \tag{2.30}$$

2.3 Comparison with the Method of EN967

As mentioned in the introduction, the development of the above formulae for the moments of a polygon was spurred by potential flow calculations using panel methods. For some years DREA has used the program EN967 to calculate potential flows. The expressions given above for the panel moments improve upon the EN967 expressions in the following ways.

1. The code is more efficient than that used by EN967. Timing comparisons indicate that the new method calculates the first three panel moments (the ones required by the potential flow calculations) in approximately 50% of the time taken by EN967.

2. The new method is more robust than that used by EN967. The analytic expressions derived above, and hence the code derived from them, are completely free of singularities. The EN967 code relied upon manipulations of the slopes of the panel sides to calculate the moments of the panel. When the slopes were very large or very small (but non-zero), very large relative errors could occur in the values of the moments. As an example, both methods were used to calculate the moments of the panel having corner points $(x, y) = (1.0, 0.0), (-1.0, 0.5), (-1.0, -1.0)$, and $(0.0, 0.0)$. Both methods returned the correct values of the moments $I_{xx}^{(2)} = 0.41667, I_{xy}^{(2)} = 0.08333, I_{yy}^{(2)} = 0.10417$. However, when the third point was changed to $(-1.0000001, 0.5)$, the new method returned the correct values (which are as before to five significant figures) while EN967 gave $I_{xy}^{(2)} = -327,680$.

3. The new method is more general allowing arbitrary polygons. The method of EN967 allowed only quadrilaterals.

3 Determining if a Point is Inside a Polygon

The second problem to be addressed is whether a given point lies inside or outside an arbitrary polygon. It is sufficient to consider the given point to be the origin, since it can always be made to be so by a simple coordinate translation.

Problem 2: *If* $\mathbf{x}_k, k = 1, \ldots, N$ *are the vertices of a polygon in order as one proceeds around its perimeter counterclockwise, is the origin inside or outside the polygon?*

A simple and efficient solution to Problem 2 can be obtained by using the divergence theorem again. First define the function sgn(x) by

$$\begin{aligned} \mathrm{sgn}(x) &= 1 \quad \text{if } x > 0 \\ &= 0 \quad \text{if } x = 0 \\ &= -1 \ \text{if } x < 0 \end{aligned} \tag{3.1}$$

It has the properties

$$\mathrm{sgn}(x) = -\mathrm{sgn}(-x) \tag{3.2}$$

$$\mathrm{sgn}(xy) = \mathrm{sgn}(x)\mathrm{sgn}(y) \tag{3.3}$$

$$\mathrm{sgn}\left(\frac{1}{x}\right) = \mathrm{sgn}(x) \ \text{ if } x \neq 0 \tag{3.4}$$

$$\frac{d}{dx}\mathrm{sgn}(x) = 2\delta(x) \tag{3.5}$$

where $\delta(x)$ is the Dirac delta function. It is straightforward to show that

$$\int_a^b \delta(x) f(x) dx = f(0)[\mathrm{sgn}(b) - \mathrm{sgn}(a)]/2 \tag{3.6}$$

and hence that

$$\begin{aligned} \int_\square \delta(x)\delta(y)dxdy &= 1 \quad\ \text{if (0,0) is inside the polygon} \\ &= 1/2 \ \text{if (0,0) is on an edge of the polygon} \\ &= 1/4 \ \text{if (0,0) is on a vertex of the polygon} \\ &= 0 \quad\ \text{if (0,0) is outside the polygon} \end{aligned} \tag{3.7}$$

Using equation (3.5) and the divergence theorem one has

$$\begin{aligned}\int_{\Box} \delta(x)\delta(y)\,dx\,dy &= \int_{\Box} \nabla \cdot [\delta(x)\mathrm{sgn}(y)\hat{y}]\,dx\,dy \\ &= \frac{1}{2}\int_{\partial\Box} n_y \delta(x)\mathrm{sgn}(y)\,ds \end{aligned} \tag{3.8}$$

This time the k^{th} side will be parameterized with respect to the x coordinate:

$$y = y_k + \left(\frac{y_{k+1} - y_k}{x_{k+1} - x_k}\right)(x - x_k) \tag{3.9}$$

Then, $n_y ds = -dx$ and, using the properties listed in equations (3.2)–(3.6), one has

$$\begin{aligned}
&\int_{\Box} \delta(x)\delta(y)\,dx\,dy \\
&= -\frac{1}{2}\sum_{k=1}^{N} \begin{cases} \int_{x_k}^{x_{k+1}} \delta(x)\mathrm{sgn}\left[y_k + \left(\frac{y_{k+1}-y_k}{x_{k+1}-x_k}\right)(x - x_k)\right] dx & \text{if } x_k \neq x_{k+1} \\ 0 & \text{if } x_k = x_{k+1} \end{cases} \\
&= -\frac{1}{4}\sum_{k=1}^{N} \begin{cases} [\mathrm{sgn}(x_{k+1}) - \mathrm{sgn}(x_k)]\mathrm{sgn}\left[y_k - \left(\frac{y_{k+1}-y_k}{x_{k+1}-x_k}\right)x_k\right] & \text{if } x_k \neq x_{k+1} \\ 0 & \text{if } x_k = x_{k+1} \end{cases} \\
&= -\frac{1}{4}\sum_{k=1}^{N} \begin{cases} [\mathrm{sgn}(x_{k+1}) - \mathrm{sgn}(x_k)]\mathrm{sgn}\left(\frac{x_{k+1}y_k - x_k y_{k+1}}{x_{k+1}-x_k}\right) & \text{if } x_k \neq x_{k+1} \\ 0 & \text{if } x_k = x_{k+1} \end{cases} \\
&= -\frac{1}{4}\sum_{k=1}^{N} \begin{cases} [\mathrm{sgn}(x_{k+1}) - \mathrm{sgn}(x_k)]\mathrm{sgn}(x_{k+1} - x_k)\mathrm{sgn}(x_{k+1}y_k - x_k y_{k+1}) & \text{if } x_k \neq x_{k+1} \\ 0 & \text{if } x_k = x_{k+1} \end{cases} \\
&= -\frac{1}{4}\sum_{k=1}^{N}[\mathrm{sgn}(x_{k+1}) - \mathrm{sgn}(x_k)]\mathrm{sgn}(x_{k+1} - x_k)\mathrm{sgn}(x_{k+1}y_k - x_k y_{k+1}) \\
&= \frac{1}{4}\sum_{k=1}^{N} |\mathrm{sgn}(x_{k+1}) - \mathrm{sgn}(x_k)|\mathrm{sgn}(x_k y_{k+1} - x_{k+1}y_k)
\end{aligned} \tag{3.10}$$

The expression of equation (3.10) is very simple to calculate. Note, however, that its efficiency can be enhanced by avoiding unnecessary multiplications and additions; this is done in computer code by using appropriate IF THEN ELSE blocks or CASE statements. Thus, an efficient way to calculate

$$\sum_{k=1}^{N} |\mathrm{sgn}(x_{k+1}) - \mathrm{sgn}(x_k)|\mathrm{sgn}(x_k y_{k+1} - x_{k+1}y_k)$$

is by the following algorithm.

```
sum := 0
for k := 1 to N do
   if x_k < 0 then
      if x_{k+1} > 0 then
         sum := sum + 2sgn(x_k y_{k+1} - x_{k+1} y_k)
      else if x_{k+1} = 0 then
         sum := sum + sgn(x_k y_{k+1} - x_{k+1} y_k)
      end if
   else if x_k > 0 then
      if x_{k+1} < 0 then
         sum := sum + 2sgn(x_k y_{k+1} - x_{k+1} y_k)
      else if x_{k+1} = 0 then
         sum := sum + sgn(x_k y_{k+1} - x_{k+1} y_k)
      end if
   else if x_{k+1} ≠ 0 then
      sum := sum + sgn(x_k y_{k+1} - x_{k+1} y_k)
   end if
end do
```

Timing comparisons on the DREA DEC-20/60 indicate that this algorithm is about 30–50% more efficient than using equation (3.10) directly. This algorithm is used in the FORTRAN 77 subroutine INPOLY given in Appendix B.

Round-off errors associated with the term $(x_k y_{k+1} - x_{k+1} y_k)$ are not as critical in this problem as in Problem 1. In the critical case when both x's and y's are large, the term $|\mathrm{sgn}(x_{k+1}) - \mathrm{sgn}(x_k)|$ is zero, so that there is no contribution to the sum. However, round-off errors could be important for points lying very close to an edge, shifting them just enough so that they no longer lie inside (or outside) the polygon. There is no simple means of correcting for this, but a possible solution is to determine the minimum distance of the origin to the perimeter, thus allowing the user to decide when the origin is too close to an edge. The distance of the origin to the k^{th} side is

$$d_k = \frac{|x_k y_{k+1} - x_{k+1} y_k|}{\sqrt{(x_{k+1} - x_k)^2 + (y_{k+1} - y_k)^2}} \tag{3.11}$$

The distance to the perimeter is not calculated by the subroutine INPOLY in Appendix B because it decreases the efficiency of the subroutine, and is not necessary for all uses. Modification of INPOLY to calculate the distance to the perimeter is straightforward.

4 Concluding Remarks

Analytic expressions for the moments of an arbitrary polygon have been derived and used to develop a FORTRAN 77 subroutine which calculates the first three moments of an arbitrary polygon. A similar expression (with corresponding subroutine) for determining whether a point lies within a polygon has also been derived. These expressions have the following properties.

1. They are simple and concise and therefore easily programmed in a computer language (see Appendices A and B).
2. They are computationally efficient.
3. The code derived from the expressions is robust provided care is taken to avoid round-off errors in the terms $(x_k y_{k+1} - x_{k+1} y_k)$.
4. They are general, providing correct expressions for polygons with any number of sides or any degree of connectivity.

Appendix A FORTRAN 77 Subroutine PLYMOM

The FORTRAN 77 subroutine PLYMOM calculates the first three moments of an arbitrary polygon using the methods discussed in Sections 2.1 and 2.2.

```
      SUBROUTINE PLYMOM(N,VERTEX,AREA,ACENT,SECMOM)
C
C PLYMOM calculates the first three moments of an arbitrary polygon.
C It assumes counter-clockwise panel corner point order.
C
C Subroutine PLYMOM was developed by the Canadian Department of
C National Defence.
C
C Author: David Hally, 14/1/87
C
C INPUT:
C
C N      = The number of vertices of the polygon.
C VERTEX = A 2 x N array containing the vertices of the polygon.
C          VERTEX(1,I) is the x-component of the Ith vertex.
C          VERTEX(2,I) is the y-component of the Ith vertex.
C
C OUTPUT:
C
C AREA   = Area of the polygon
C ACENT  = First moments of the panel (centroid times area)
C SECMOM = 2nd moments of area of the panel. SECMOM(1)=Ixx,
C          SECMOM(2)=Ixy=Iyx, SECMOM(3)=Iyy
C
      INTEGER M, MP1, N
      REAL AREA, ACENT(2), SECMOM(3), VERTEX(2,N), YKXK, VSUMX, VSUMY,
     + XKMO, XKP1MO, YKMO, YKP1MO

C Calculate moments about [VERTEX(1,1),VERTEX(2,1)]
      AREA=0.0
      ACENT(1)=0.0
      ACENT(2)=0.0
```

```
      SECMOM(1)=0.0
      SECMOM(2)=0.0
      SECMOM(3)=0.0
      XKMO=0.0
      YKMO=0.0
      DO 10 M=1,N
        MP1=M+1
        IF (M.EQ.N) MP1=1
        XKP1MO=VERTEX(1,MP1)-VERTEX(1,1)
        YKP1MO=VERTEX(2,MP1)-VERTEX(2,1)
        VSUMX=XKP1MO+XKMO
        VSUMY=YKP1MO+YKMO

C Calculate AREA
        YKXK=YKP1MO*XKMO-YKMO*XKP1MO
        AREA=AREA+YKXK

C Calculate ACENT
        ACENT(1)=ACENT(1)+VSUMX*YKXK
        ACENT(2)=ACENT(2)+VSUMY*YKXK

C Calculate SECMOM
        SECMOM(1)=SECMOM(1)+YKXK*(XKP1MO*VSUMX+XKMO**2)
        SECMOM(2)=SECMOM(2)+YKXK*(XKP1MO*YKP1MO+XKMO*YKMO+VSUMX*VSUMY)
        SECMOM(3)=SECMOM(3)+YKXK*(YKP1MO*VSUMY+YKMO**2)

        XKMO=XKP1MO
        YKMO=YKP1MO
   10 CONTINUE

C Calculate moments about (0,0)
      AREA=AREA*0.5
      ACENT(1)=ACENT(1)/6.0+VERTEX(1,1)*AREA
      ACENT(2)=ACENT(2)/6.0+VERTEX(2,1)*AREA
      SECMOM(1)=SECMOM(1)/12.0+VERTEX(1,1)*(2.0*ACENT(1)-
     +          VERTEX(1,1)*AREA)
      SECMOM(2)=SECMOM(2)/24.0+VERTEX(1,1)*ACENT(2)+
     +          VERTEX(2,1)*(ACENT(1)-VERTEX(1,1)*AREA)
      SECMOM(3)=SECMOM(3)/12.0+VERTEX(2,1)*(2.0*ACENT(2)-
     +          VERTEX(2,1)*AREA)
      RETURN
      END
```

Appendix B FORTRAN 77 Subroutine INPOLY

The FORTRAN 77 subroutine INPOLY function determines whether a point is inside an arbitrary polygon using the method discussed in Section 3.

```
      SUBROUTINE INPOLY(N,VERTEX,X,IFLAG)
C
C INPOLY determines whether the point X lies inside an arbitrary
C polygon. It assumes counter-clockwise panel corner point order.
C
C Subroutine INPOLY was developed by the Canadian Department of
C National Defence.
C
C Author: David Hally, 14/1/87
C
C INPUT:
C
C N      = The number of vertices of the polygon.
C VERTEX = A 2 x N array containing the vertices of the polygon.
C          VERTEX(1,I) is the x-component of the Ith vertex.
C          VERTEX(2,I) is the y-component of the Ith vertex.
C X      = An array of length 2 containing the point which is to be
C          checked.
C          X(1) is the x-component of the point.
C          X(1) is the y-component of the point.
C
C OUTPUT:
C
C IFLAG = 4, if X is inside the polygon
C       = 2, if X is on a side of the polygon
C       = 1, if X is on a vertex of the polygon
C       = 0, if X is outside the polygon
C
      INTEGER IFLAG, M, MP1, N, SGN
      REAL VERTEX(2,N), X(2), XKMO, XKP1MO, YKMO, YKP1MO

      IFLAG=0
```

```
      XKMO=VERTEX(1,1)-X(1)
      YKMO=VERTEX(2,1)-X(2)
      DO 10 M=1,N
         MP1=M+1
         IF (M.EQ.N) MP1=1
         XKP1MO=VERTEX(1,MP1)-X(1)
         YKP1MO=VERTEX(2,MP1)-X(2)
         IF (XKMO.LT.0) THEN
            IF (XKP1MO.GT.0) THEN
               IFLAG=IFLAG+2*SGN(XKMO*YKP1MO-XKP1MO*YKMO)
            ELSE IF (XKP1MO.EQ.0) THEN
               IFLAG=IFLAG+SGN(XKMO*YKP1MO-XKP1MO*YKMO)
            END IF
         ELSE IF (XKMO.GT.0) THEN
            IF (XKP1MO.LT.0) THEN
               IFLAG=IFLAG+2*SGN(XKMO*YKP1MO-XKP1MO*YKMO)
            ELSE IF (XKP1MO.EQ.0) THEN
               IFLAG=IFLAG+SGN(XKMO*YKP1MO-XKP1MO*YKMO)
            END IF
         ELSE IF (XKP1MO.NE.0) THEN
            IFLAG=IFLAG+SGN(XKMO*YKP1MO-XKP1MO*YKMO)
         END IF
         XKMO=XKP1MO
         YKMO=YKP1MO
10    CONTINUE
      RETURN
      END

      INTEGER FUNCTION SGN(X)
C Calculates sgn(X)
      REAL X
      IF (X.GT.0.0) THEN
         SGN=1
      ELSE IF (X.LT.0.0) THEN
         SGN=-1
      ELSE
         SGN=0
      END IF
      RETURN
      END
```

References

[1] D. Hally, "POTFLO: A Suite of Programs for Calculating Potential Flow about Ship Hulls," DREA Technical Memorandum, in preparation.

[2] *A Mathematics and Statistics Workshop with Science, Government and Industry*, Dalhousie University (April 1986).

[3] M. Mackay, "Program EN967 – A revised User's Guide," DREA Technical Communication 86/305, 1986.

[4] C. Ann Dent, "User's Manual for VIEWMAP: A Simple Vision System," DREA Technical Communication, in review.

AD-A183 444 ION

Unclassified

Security Classification

DOCUMENT CONTROL DATA - R & D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall document is classified)

1. ORIGINATING ACTIVITY Defence Research Establishment Atlantic	2a. DOCUMENT SECURITY CLASSIFICATION Unclassified 2b. GROUP

3. DOCUMENT TITLE

CALCULATION OF THE MOMENTS OF POLYGONS

4. DESCRIPTIVE NOTES (Type of report and inclusive dates)
Technical Memorandum

5. AUTHOR(S) (Last name, first name, middle initial)

Hally, David

6. DOCUMENT DATE JUNE 1987	7a. TOTAL NO. OF PAGES 23	7b. NO. OF REFS 4

8a. PROJECT OR GRANT NO.	9a. ORIGINATOR'S DOCUMENT NUMBER(S) DREA TECHNICAL MEMORANDUM 87/209
8b. CONTRACT NO.	9b. OTHER DOCUMENT NO.(S) (Any other numbers that may be assigned this document)

10. DISTRIBUTION STATEMENT

UNLIMITED DISTRIBUTION

11. SUPPLEMENTARY NOTES	12. SPONSORING ACTIVITY

13. ABSTRACT

Methods for calculating the moments of an arbitrary polygon and for determining whether a point lies within a polygon are discussed. The methods are efficient, robust, concise, and easily programmed in any computer language. A FORTRAN 77 subroutine which calculates the first three moments of an arbitrary polygon is also included, as is a subroutine which determines whether a point lies in an arbitrary polygon.

DSIS
78-078

Unclassified

Security Classification

KEY WORDS

Polygons

Computer Programs

Panel Methods

Canada

INSTRUCTIONS

1. ORIGINATING ACTIVITY: Enter the name and address of the organization issuing the document.

2a. DOCUMENT SECURITY CLASSIFICATION: Enter the overall security classification of the document including special warning terms whenever applicable.

2b. GROUP: Enter security reclassification group number. The three groups are defined in Appendix 'M' of the DRB Security Regulations.

3. DOCUMENT TITLE: Enter the complete document title in all capital letters. Titles in all cases should be unclassified. If a sufficiently descriptive title cannot be selected without classification, show title classification with the usual one-capital-letter abbreviation in parentheses immediately following the title.

4. DESCRIPTIVE NOTES: Enter the category of document, e.g. technical report, technical note or technical letter. If appropriate, enter the type of document, e.g. interim, progress, summary, annual or final. Give the inclusive dates when a specific reporting period is covered.

5. AUTHOR(S): Enter the name(s) of author(s) as shown on or in the document. Enter last name, first name, middle initial. If military, show rank. The name of the principal author is an absolute minimum requirement.

6. DOCUMENT DATE: Enter the date (month, year) of Establishment approval for publication of the document.

7a. TOTAL NUMBER OF PAGES: The total page count should follow normal pagination procedures, i.e., enter the number of pages containing information.

7b. NUMBER OF REFERENCES: Enter the total number of references cited in the document.

8a. PROJECT OR GRANT NUMBER: If appropriate, enter the applicable research and development project or grant number under which the document was written.

8b. CONTRACT NUMBER: If appropriate, enter the applicable number under which the document was written.

9a. ORIGINATOR'S DOCUMENT NUMBER(S): Enter the official document number by which the document will be identified and controlled by the originating activity. This number must be unique to this document.

9b. OTHER DOCUMENT NUMBER(S): If the document has been assigned any other document numbers (either by the originator or by the sponsor), also enter this number(s).

10. DISTRIBUTION STATEMENT: Enter any limitations on further dissemination of the document, other than those imposed by security classification, using standard statements such as:

(1) "Qualified requesters may obtain copies of this document from their defence documentation center."

(2) "Announcement and dissemination of this document is not authorized without prior approval from originating activity."

11. SUPPLEMENTARY NOTES: Use for additional explanatory notes.

12. SPONSORING ACTIVITY: Enter the name of the departmental project office or laboratory sponsoring the research and development. Include address.

13. ABSTRACT: Enter an abstract giving a brief and factual summary of the document, even though it may also appear elsewhere in the body of the document itself. It is highly desirable that the abstract of classified documents be unclassified. Each paragraph of the abstract shall end with an indication of the security classification of the information in the paragraph (unless the document itself is unclassified) represented as (TS), (S), (C), (R), or (U).

The length of the abstract should be limited to 20 single-spaced standard typewritten lines; 7½ inches long.

14. KEY WORDS: Key words are technically meaningful terms or short phrases that characterize a document and could be helpful in cataloging the document. Key words should be selected so that no security classification is required. Identifiers, such as equipment model designation, trade name, military project code name, geographic location, may be used as key words but will be followed by an indication of technical context.

END
9-87
DTIC