# Pertenencia de un punto del plano al interior o a la frontera de una región limitada por un polígono

## Ownership of a Point of the Plane to the Interior or to the Border of a Region Limited by a Polygon

**Arístides Alejandro Legrá Lobaina**[1]
**Andrés Silva Pineda**[2]

[1] Doctor en Ciencias Técnicas. Profesor Asistente. Investigador de Geoestadística aplicada a yacimientos lateríticos y desarrollo de algoritmos para cartografía digital. Departamento de Matemáticas, Facultad de Geología y Minería, Instituto Superior Minero Metalúrgico.

[2] Profesor Instructor, Departamento de Matemáticas. Investigador de la creación y estudio de algoritmos computacionales. Unidad Docente Baracoa. Instituto Superior Pedagógico de Guantánamo.

**Resumen**

En este artículo se describen dos algoritmos para determinar la pertenencia de un punto del plano al interior o a la frontera de una región limitada por un polígono. El segundo de ellos es recomendado por su rapidez. Al final se explica, como anexo a este trabajo, un programa en lenguaje Pascal para la implementación del segundo algoritmo.

**Palabras claves:** Polígono, convexo, pertenencia, algoritmo.

**Abstract**

In these article two algorithms are described in order to determine the ownership of a point of the plan to the interior or to the border of a region limited by a polygon. The second of them is recommended for their rapidity. At final is described, like annex to this work, a program in Pascal language for the implementation of the second algorithm.

**Key Words:** Polygon, Convex, Ownership, Algorithm.

## INTRODUCCIÓN

Es usual definir una región A del plano $R^2$ mediante un conjunto de n puntos $Q_1,...,Q_n$, de coordenadas $(x_i,y_i)$ i=1,...,n en el sistema O X Y, ordenados, en general en sentido positivo (contrario a las manecillas del reloj) y según la secuencia del dibujo, que forman un polígono al cual se considera un modelo de la frontera de dicha región.

El problema que se plantea es determinar si un punto P de coordenadas (x,y) está en el exterior o en el interior y frontera de A (Gómez, 1995; Bustillo, 1996; Bustillo. 1997; ITGE, 1997). Los casos más conocidos que se presentan relacionados con la geología y la minería son los de la densificación de una red en una región plana limitada por un polígono, la búsqueda de la posición de un punto con respecto a una región en un mapa digitalizado y, en la minería del Ni en Cuba, la determinación de la cantidad y calidad del mineral extraído a partir de mediciones topográficas realizadas; en este caso, se obtiene un polígono no convexo que enmarca la zona minada y es necesario saber cuales de los puntos de la red plana de control de la base de datos están en el interior o en la frontera de este polígono de manera que a partir de la diferencia de cotas en estos puntos, se realizarán los cálculos. Un caso menos conocido donde se plantea este problema es el relacionado con la fórmula de G. Pick (Shaskin, 1989) que plantea que si dividimos el plano en cuadrados de área 1 y denominamos al conjunto de vértices de todos los cuadrados *retículo puntual* y a dichos vértice *nodos* del retículo y además tenemos un polígono tal que todos sus vértices pertenezcan al retículo (a este polígono se le llama *polígono reticular*), entonces el área de la región limitada por el polígono está dada por Ap = i-1+b/2 donde i es el número de nodos que se encuentran dentro del polígono y b el número de nodos que contiene la frontera. Estos resultados son usados en varias ramas teóricas de la Matemática.

Se conocen varios algoritmos para resolver el problema planteado. Entre ellos son mas conocidos el de la *suma de ángulos subtendidos*, el de *saltos de una línea a un punto exterior*, el de *segmentos de la frontera a la derecha de punto* y el de Davis-David (Davis and David, 1979). Todos presentan dificultades relacionadas con una gran complejidad o con un considerable volumen de cálculo.

El objetivo de este trabajo es resolver el problema planteado mediante la presentación de:

a. Una variación del algoritmo de suma de ángulos subtendidos donde se reduce el volumen de cálculo y su complejidad.
b. Un algoritmo iterativo que combina fronteras y subfronteras.

## DESARROLLO

Sea un polígono O arbitrario de n vértices $Q_k$ ordenados positivamente. En el vértice $Q_k$, kÎ{1,...,n} se tendrá el ángulo interior $a_k$ (definido por tres vértices consecutivos) y se puede definir en este punto el ángulo exterior $b_k$=p-$a_k$. Se conoce que:

$$\sum_{k=1}^{n} \alpha_k = (n-2)\pi \qquad \sum_{k=1}^{n} \beta_k = 2\pi$$

Un polígono se dice *convexo* si todos los ángulos interiores $a_k$ son menores o iguales que p.

El área de un polígono se calcula (Bronstein y Semendiaev 1973) por la fórmula clásica:

$$A_{PC} = [(x_1-x_2)*(y_1+y_2) + ... + (x_{n-1}-x_n)*(y_{n-1}+y_n) + (x_n-x_1)*(y_n+y_1]/2$$

En particular el área de un triángulo se puede calcular por:

$$A_T = [(x_1-x_2)*(y_1+y_2) + (x_2-x_3)*(y_2+y_3) + (x_3-x_1)*(y_3+y_1]/2$$

### Criterio *PertenecePC*:

Para determinar si un punto P pertenece al exterior o al interior y frontera de la región A limitada por un polígono convexo PC hallamos $A_{PC}$ y también hallamos las áreas $A_{Ti}$ de los n triángulos que forma el punto P al unirlo con cada pareja de vértices consecutivos de PC. Si $A_{PC}$ es igual a la suma de los valores $A_{Ti}$ entonces se puede afirmar que P pertenece al interior o a la frontera de A. Si además alguno de los valores $A_{Ti}$ es 0 entonces P pertenece estrictamente a la frontera de A.

Si el polígono O que limita a la región A no es convexo (PNC) entonces proponemos el siguiente algoritmo:

1. Buscar el polígono convexo PC (que limita a una región B), formado por el subconjunto de puntos del PNC tal que todos los puntos de PNC pertenezcan al interior o a la frontera de la región B. A la región B se le llama *cápsula convexa* de los puntos del PNC. El polígono PC puede obtenerse mediante el *algoritmo del ángulo mínimo* (Legra y Puente, 1996).
2. Si el punto P está en el exterior de B entonces también estará en el exterior de A y finaliza este algoritmo. En caso de que P esté en el interior o en la frontera de B, entonces continuamos ejecutando el paso 3.
3. Agregamos un nuevo punto $Q_{n+1}$ al conjunto $Q_1$, $Q_2$, ... ,$Q_n$ tal que todos sus componentes son iguales a las del punto $Q_1$. Podemos asumir en lo que sigue que n=n+1.
4. Trasladamos todos los puntos a un nuevo sistema de coordenadas con centro en P. Se obtienen los puntos $Q1_1$, $Q1_2$,..., $Q1_n$ en el sistema de coordenadas O X1 Y1 y se asume que la *suma* de los ángulos con vértice en el punto P y subtendido a dos lados consecutivos del polígono es 0.
5. Para j igual 1,2,3, ... , n-1, hacer lo siguiente:
   A. Hallar g, ángulo que forma el segmento $OQ1_j$ con el eje OX1.
   B. Rotar el punto $Q1_{j+1}$ el ángulo g y se obtiene el punto $Q2_{j+1}$ en el sistema O X2 Y2.
   C. Hallar a, ángulo entre $Q2_{j+1}$ y OX2, aÎ[-p,p].
   D. Hallar *suma*=*suma*+a.
6. Si *suma*=0° entonces el punto está fuera del polígono y si SUMA=360° entonces está dentro.

   Es obvio que el algoritmo solo se ejecuta totalmente para los puntos interiores o fronteras de B por lo que el volumen de cálculo se hace menor.

   Asimismo afirmamos que la complejidad del algoritmo y de los cálculos es pequeña puesto que solo intervienen operaciones sencillas y análisis elementales.

   Una segunda forma de resolver este problema es mediante el siguiente algoritmo al que hemos llamado *algoritmo frontera-subfronteras*:

   Sea el polígono O (convexo o no) que tiene n vértices ordenados positivamente y que limita a la región A. Agregamos un nuevo punto $Q_{n+1}$ al conjunto $Q_1$, $Q_2$, ..., $Q_n$ de forma tal que todos sus componentes sean iguales a las del punto $Q_1$ y asumimos que n=n+1.

Los n puntos Q de O están ordenados según cierto índice *i* = 1, ... ,n.

1. Se busca el polígono PC (de puntos P) que limita a la región B donde se cumple P(1)=P(k) y además hemos conservado los índices i.
2. Determinamos t subconjuntos Dj de O que forman las *entradas no convexas*. Para ello verificamos en PC cuando se produce salto en el índice.
3. Dado un punto P, se tiene que P pertenece al interior o frontera de A sí se cumple:
   a. P pertenece a PC. Aplicamos el criterio Pertenece PC.
   b. Para todo j=1,...,t, se cumple que P no pertenece al interior de $D_j$. Puesto que Dj puede ser convexo o no convexo se aplica este mismo algoritmo de manera recursiva.

   Ejemplo:

   Sea O = {Q1,Q2,....,Q14,Q1} y se determina PC = {Q1,Q2,Q5,Q6,Q7,Q8,Q13,Q14,Q1}. Es evidente que se tienen las *entradas no convexas*:

   $D_1$ = {Q2,Q3,Q4,Q5,Q2}
   $D_2$ = {Q8,Q9,Q10,Q11;Q12,Q13,Q8}

   Para determinar si P pertenece a O verificamos que:
   a. P pertenezca a PC.
   b. P no pertenezca al interior de $D_1$.
   c. P no pertenezca al interior de $D_2$.

A modo de ilustración para este ejemplo, veamos el siguiente Gráfico 1:

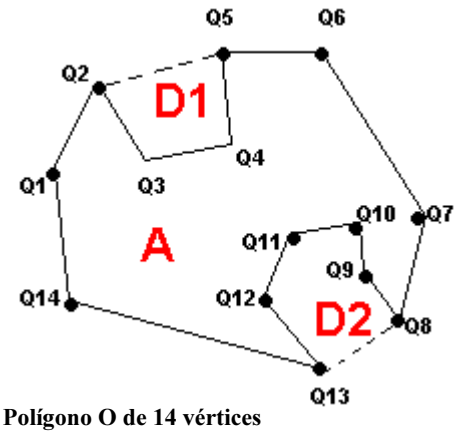


**GRÁFICO1. Polígono O de 14 vértices**

En el Anexo 1 se presenta un programa para computadoras desarrollado en lenguaje Pascal, nótese que a pesar de que en él se determina la pertenencia de 10 puntos al polígono del ejemplo anterior, las funciones *PuntoDentroPC* y *EstaDentro* son generales y se pueden utilizar en cualquiera de nuestros programas en lenguaje Pascal. Para la obtención de los polígonos convexos puede verse Legrá y Puente, 1996.

El algoritmo desarrollado con el criterio *PerteneecePC* ha sido aplicado con éxito en la determinación del escombro y mineral extraído en la Subdirección de Minas de la Empresa Ernesto "Che Guevara" de Moa, lográndose gran rapidez en los cálculos para levantamientos topográficos con grandes volúmenes de mediciones y considerando áreas extensas.

## CONCLUSIONES

Los algoritmos presentados, que resuelven el problema planteado en el objetivo declarado en la introducción, han sido programados y en la práctica se ha comprobado que son eficientes cuando se analiza la pertenencia de un gran número de puntos al exterior o al interior y frontera de una región limitada por un polígono arbitrario ordenado que tiene también un alto número de puntos. Vale recordar que durante la programación se tuvieron en cuenta los diferentes errores de redondeo, operacionales, etc., que pueden provocar tomas de decisiones equivocadas.

## BIBLIOGRAFÍA


BRONSTEIN, I. y K. SEMENDIAEV: *Manual de Matemáticas para ingenieros y estudiantes,* Editorial MIR, Moscú, 1973.

BUSTILLO REVUELTA, M. y C. LÓPEZ JIMENO: *Manual de evaluación y diseño de explotación minera,* Entorno Gráfico, S.L., Madrid,1997.

BUSTILLO REVUELTA, M. y C. LÓPEZ JIMENO: *Recursos minerales: tipología, prospección, evaluación, explotación, mineralurgia e impacto ambiental,* Gráficas Arias Montalvo S.A., Madrid,1996.

DAVIS, M. y M. DAVID: *An Algorithm for Finding the Position of a Point Relative to a Fixed Polygonal Boundary,* Plenum Publishing Corporation, Montreal, Canada, 1979.

GÓMEZ DE LAS HERAS, JESÚS; CARLOS LÓPEZ JIMENO; EMILIO LÓPEZ JIMENO y otros: *Manual de Arranque, carga y transporte en minería a cielo abierto,* Madrid, 1995.

INOSTROSA, PATRICIO y MARÍA CECILIA RIVARA: "Triangulación interactiva de polígonos". Solución de problemas en ingeniería con métodos numéricos, Sociedad Venezolana de Métodos Numéricos en Ingeniería, Caracas, 1994.

INSTITUTO TECNOLÓGICO GEOMINERO DE ESPAÑA (ITGE): "Manual de Evaluación Técnica-Económica de Proyectos Mineros de Inversión", Madrid, 1997.

LEGRÁ LOBAINA, A. y ALFONSO PUENTE MARTÍN: "Determinación de los Volúmenes de Sólidos Minerales Irregulares usando Spline Cúbico Natural", en *Minería y Geología*, no 3, vol. 13, Instituto Superior Minero Metalúrgico de Moa, 1996.

SHASKIN, YU: "Característica Euleriana", en *Lecciones Populares de Matemática*, Editorial MIR, Moscú, 1989.


## ANEXO

### Programa básico fuente en Lenguaje Pascal del algoritmo frontera-subfronteras


*Elaborado para Turbo Pascal Versión 7.0 de Borland International, In.*

```
$M 65000,0,655360
Program AlgFSF;
Uses crt;
const Max = 100;
type
   Punto      = array[1..2] of real;
   Polígono = record
 Q : array[1..Max] of Punto; {Coordenadas}
                I : array[1..Max] of byte; {Indices}
                  n : word;  {Número de Puntos }
              end;
var O                 : Polígono;
    Int               : Boolean;
    s,CualFrontera    : Byte;
    Prueba            : array[1..10] of punto;
```

(Aquí se supone que se tiene un algoritmo para hallar la frontera convexa del polígono O. En este caso

```
se sigue el ejemplo visto en el desarrollo. Los índices de X se heredan de O)
        Procedure FronteraConvexa(O:polígono; Var X:Polígono);
          begin
           case CualFrontera of
             1:with X do
                begin
                  n:=9;
                    Q[1,1]:=O.Q[ 1,1];   Q[ 1,2]:=O.Q[ 1,2]; I[1]:=O.I[1]; Q[ 2,1]:=O.Q[ 2,1];
                      Q[2,2]:=O.Q[ 2,2];  I[2]:=O.I[2]; Q[ 3,1]:=O.Q[ 5,1];   Q[ 3,2]:=O.Q[ 5,2];
                      I[3]:=O.I[5]; Q[ 4,1]:=O.Q[ 6,1];   Q[ 4,2]:=O.Q[ 6,2];  I[4]:=O.I[6];
                    Q[5,1]:=O.Q[ 7,1];   Q[ 5,2]:=O.Q[ 7,2]; I[5]:=O.I[7]; Q[ 6,1]:=O.Q[ 8,1];
                      Q[6,2]:=O.Q[ 8,2];  I[6]:=O.I[8]; Q[ 7,1]:=O.Q[13,1];   Q[ 7,2]:=O.Q[13,2];
                     I[7]:=O.I[13]; Q[ 8,1]:=O.Q[14,1];   Q[ 8,2]:=O.Q[14,2];  I[8]:=O.I[14];
                    Q[9,1]:=O.Q[ 1,1];   Q[ 9,2]:=O.Q[ 1,2]; I[9]:=O.I[15];
                end;
             2:with X do
                begin
                  n:=5;
                    Q[1,1]:=O.Q[ 1,1];   Q[ 1,2]:=O.Q[ 1,2]; I[1]:=O.I[1]; Q[ 2,1]:=O.Q[ 2,1];
                      Q[2,2]:=O.Q[ 2,2];  I[2]:=O.I[2]; Q[ 3,1]:=O.Q[ 3,1];   Q[ 3,2]:=O.Q[ 3,2];
                     I[3]:=O.I[3]; Q[ 4,1]:=O.Q[ 4,1];   Q[ 4,2]:=O.Q[ 4,2];  I[4]:=O.I[4];
                    Q[5,1]:=O.Q[ 5,1];   Q[ 5,2]:=O.Q[ 5,2]; I[5]:=O.I[5];
                end;
             3:with X do
                begin
                  n:=6;
                    Q[1,1]:=O.Q[ 1,1];   Q[ 1,2]:=O.Q[ 1,2]; I[1]:=O.I[1]; Q[ 2,1]:=O.Q[ 3,1];
                      Q[2,2]:=O.Q[ 3,2];  I[2]:=O.I[3]; Q[ 3,1]:=O.Q[ 4,1];   Q[ 3,2]:=O.Q[ 4,2];
                     I[3]:=O.I[4]; Q[ 4,1]:=O.Q[ 5,1];   Q[ 4,2]:=O.Q[ 5,2];  I[4]:=O.I[5];
                    Q[5,1]:=O.Q[ 6,1];   Q[ 5,2]:=O.Q[ 6,2]; I[5]:=O.I[6];
                    Q[6,1]:=O.Q[ 7,1];   Q[ 6,2]:=O.Q[ 7,2]; I[6]:=O.I[7];
                end;
             4:with X do
                begin
                  n:=4;
                    Q[1,1]:=O.Q[ 1,1];   Q[ 1,2]:=O.Q[ 1,2]; I[1]:=O.I[1]; Q[ 2,1]:=O.Q[ 2,1];
                      Q[2,2]:=O.Q[ 2,2];  I[2]:=O.I[2]; Q[ 3,1]:=O.Q[ 3,1];   Q[ 3,2]:=O.Q[ 3,2];
                     I[3]:=O.I[3]; Q[ 4,1]:=O.Q[ 4,1];   Q[ 4,2]:=O.Q[ 4,2];  I[4]:=O.I[4];
                end;
           end;
          end;
        Function PuntoDentroPC(P:Punto;PC:Poligono;var EnInterior:boolean):boolean;
          var AreaP,AreaT,SumaAt: real;
              j                 : byte;
          begin
           (Área del Polígono)
           ÁreaP:=0;
                    for j:=1 to PC.n-1 do areaP:=AreaP+(PC.Q[j,1]-C.Q[j+1,1])*(PC.Q[j,2]+PC.Q[j+1,2]);
           AreaP:=abs(Areap/2);
           (Áreas de los Triángulos}
           EnInterior:=true;   SumaAT:=0;
           for j:=1 to PC.n-1 do
            begin
           ÁreaT:=((P[1]-PC.Q[j,1])*(P[2]+PC.Q[ j,2])+
                          (PC.Q[j,1]-PC.Q[j+1,1])*(PC.Q[ j,2]+PC.Q[j+1,2])+
                                                  (PC.Q[j+1,1]-P[1])*(PC.Q[j+1,2]+P[2]))/2;
                SumaAT:=SumaAt+abs(AreaT);
              (Se descarta el lado que no pertenece a O)
              if (j<>PC.n-1) and (abs(AreaT)<0.0000001) then EnInterior:=false;
            end;
            if Abs(AreaP-SumaAT)<0.0000001 then PuntoDentroPC:=true
            else PuntoDentroPC:=false;
          end;
        Function EstaDentro(P:Punto;O:Poligono;var EnInterior:boolean):boolean;
          var PC            : Poligono;
              D             : array[1..10] of Poligono;
              t,j,k         : byte;
               Ok,Oa,Ob : boolean;
          begin
           inc(CualFrontera);
           FronteraConvexa(O,PC);
           (Algoritmo para Obtener las Entradas No Convexas de O)
           (Analizando la 'continuidad' de los índices de PC)
           t:=0;
           for j:=1 to PC.n-1 do
            begin
              if PC.I[j+1]-PC.I[j]>1 then
              begin
                inc(t);
                (número de datos)
                   D[t].n:=PC.I[j+1]-PC.I[j]+2;
                (Indices: Nuevos desde 1)
                  for k:=1 to D[t].n do D[t].I[k]:=k;
                 for k:=1 to D[t].n-1 do
                  begin
```

```
                    D[t].Q[k,1]:=O.Q[O.I[k+PC.i[j]-1],1];
D[t].Q[k,2]:=O.Q[O.I[k+PC.I[j]-1],2];
                  end;
                    D[t].Q[D[t].n,1]:=D[t].Q[1,1];
D[t].Q[D[t].n,2]:=D[t].Q[1,2];
              end;
            end;
          ( Comprobación ¿Interior, Frontera o Fuera?)
          Ok:=PuntoDentroPC(P,PC,EnInterior);
          if Ok then
            for j:=1 to t do
             begin
               Oa:=EstaDentro(P,D[j],Ob);
              ( Dentro de Dj y solo en su Interior:  significa que está fuera del polígono O)
               if (Oa) and (Ob) then
                 begin
                   Ok:=false; break;
                 end;
             end;
           estadentro:=Ok;
         end;
        Procedure IniciaDatos;
          var k: byte;
          begin
            CualFrontera:=0;
            with O do
             begin
               n:=15;
                  Q[1,1]:= 0; Q[ 1,2]:= 6;     Q[2,1]:= 1;
Q[2,2]:= 9;   Q[3,1]:= 3; Q[3,2]:= 7;
                 Q[4,1]:= 4; Q[ 4,2]:= 8;     Q[5,1]:= 4;
Q[5,2]:=10;     Q[6,1]:= 6; Q[6,2]:=10;
                 Q[7,1]:=10; Q[ 7,2]:= 7;     Q[8,1]:= 9;
Q[8,2]:= 2;     Q[9,1]:= 8; Q[9,2]:= 3;
                 Q[10,1]:= 8; Q[10,2]:= 5;    Q[11,1]:= 5;
Q[11,2]:= 5;    Q[12,1]:= 5; Q[12,2]:= 2;
                 Q[13,1]:= 7; Q[13,2]:= 0;    Q[14,1]:= 1;
Q[14,2]:= 3;    Q[15,1]:= 0; Q[15,2]:= 6;
               for k:=1 to n do I[k]:=k;
             end;
          end;
        (Parte Principal del Programa)
        begin
          Prueba[1][1]:= 3;   Prueba[1][2]:= 5;  Prue-
ba[2][1]:= 0;   Prueba[2][2]:= 0;
          Prueba[3][1]:=10;   Prueba[3][2]:=10;  Prue-
ba[4][1]:= 9;   Prueba[4][2]:= 6;
          Prueba[5][1]:= 8;   Prueba[5][2]:= 1;  Prue-
ba[6][1]:= 0;   Prueba[6][2]:= 6;
          Prueba[7][1]:= 5;   Prueba[7][2]:=10;  Prue-
ba[8][1]:= 3;   Prueba[8][2]:= 9;
          Prueba[9][1]:= 6;   Prueba[9][2]:= 4;  Prue-
ba[10][1]:= 5;   Prueba[10][2]:= 4;
          clrscr;
          writeln('Demostración del Algoritmo Frontera - Subfronteras para conocer ');
          writeln('la pertenencia de un punto al interior de una región poligonal.'); writeln;
          for s:=1 to 10 do
           begin
                                                IniciaDatos;
write('(',Prueba[s][1]:6:2,',',Prueba[s][2]:6:2,')');
             if EstaDentro(Prueba[s],O,Int) then  writeln(' Está Dentro')
               else  writeln(' No Está Dentro');
           end;
          writeln;
          write('Oprima ENTER...'); readln;
        end.
```