REVISÃO DO ALGORÍTMO PTLOC PARA DETERMINAÇÃO DA POSIÇÃO DE UM PONTO EM RELAÇÃO A UM POLÍGONO

J.K. Yamamoto; M.A. Braghin

O algorítmo de Hall (1975) é extremamente utilizado em computação gráfica para verificar se um ponto qualquer está dentro ou fora de um polígono e tem-se mostrado confiável na maioria das aplicações. Basicamente, este algorítmo trabalha com a soma orientada de ângulos entre o ponto de interesse e os vértices do polígono, se esta soma for +/- 360^0 o ponto está dentro e se for zero o ponto está fora. Entretanto, aplicando-se este algorítmo na determinação da posição relativa de um ponto em relação ao domínio de interesse de uma jazida, usualmente definida por uma série de segmentos norte-sul e leste-oeste, verificou-se que para alguns pontos o citado algorítmo não funciona (vide exemplo na Figura 1), devido a uma falha de contagem da variável iec, que conta o número de vezes que a soma de ângulos cruza a linha do equador em $+180^0$ ou -180^0 (soma orientada). Nos casos em como o do polígono citado por exemplo, tem-se que, ao definir uma soma orientada de ângulos positivos, certos ângulos, pela particular posição do ponto em relação aos vértices do polígono, serão negativos fazendo com que o contador iec seja acionado toda vez que cruze o equador tanto na soma (positivos) como na subtração de ângulos (negativos). Verificado isto, apresenta-se neste trabalho uma solução que ao invés de contar o número de vezes que a soma orientada de ângulos cruza o equador, soma-se simplesmente os ângulos, conforme a idéia original do trabalho, não implementada por Hall (1975), para economia de tempo de execução e para não utilizar a função da biblioteca arco-tangente. A única modificação feita então foi a definição dos ângulos parciais e a soma destes ao final da interação com todos os vértices, verificando-se então a soma final dos ângulos ao invés do número de vezes que a soma de ângulos cruzou a linha do equador (número par está fora e número ímpar está dentro). Vide figura 2 mostrando uma aplicação do algorítmo corrigido, este apresentado na figura 3.

REFERÊNCIAS

HALL, J.K. 1975. PTLOC-A FORTRAN subroutine for determining the position of a point relative to a closed boundary. Mathematical Geology., 7(1):75-79.

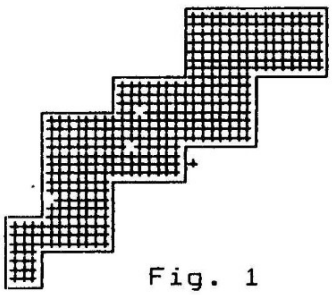

Fig. 1

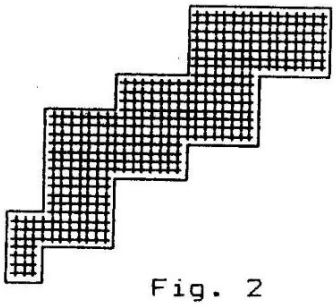

Fig. 2

(1)Desenho mostrando os limites de uma jazida hipotética e os pontos de avaliação pertencentes ao domínio (+). Observe os três pontos que estão no interior do polígono e não foram considerados na procedure original, bem como um ponto exterior que foi considerado como dentro. A figura (2) mostra o mesmo polígono preenchido utilizando-se o algorítmo proposto.

```
procedure ptloc(var np:integer;var x,y:array[1..100] of real;
          var xp,yp:real;var loc,i1,i2,isr:integer);
var
 ip,i:integer;
 tsc,sc,tcc,cc,x1,y1,r1,x2,y2,
 r2,vpl,st,ct:real;
const
 c180=0.99999999984;
begin
i1:=0;
i2:=0;
isr:=0;
tsc:=0.00;
sc:=0.00;
tcc:=1.00;
cc:=1.00;
sarco:=0.00;
ip:=np;
x1:=x[np]-xp;
y1:=y[np]-yp;
r1:=sqrt(sqr(x1)+sqr(y1));
i:=1;
while i <= do
begin
 x2:=x[i]-xp;
 y2:=y[i]-yp;
 r2:=sqrt(sqr(x2)+sqr(y2));
 vpl:=r1*r2;
 if r1 <>0.00
 then begin
  if r2 <>0.00
  then begin
  st:=(x1*y2-x2*y1)/vpl;
  ct:=(x1*x2+y1*y2)/vpl;
  if ct = 0.00
  then if st  0
   then arco:=90.00
   else arco:=270.0
  else if st >=0
   then if ct > 0 then arco:=arctan(st/ct)*180.0/pi
     else arco:=180.0+arctan(st/ct)*180.0/pi
   else if ct > 0 then arco:=360+arctan(st/ct)*180.0/pi
     else arco:=180+arctan(st/ct)*180.0/pi;
    if arco > 180.0 then arco:=arco-360.0;
    sarco:=sarco+arco;
    if (ct+c180) > 0.00
    then begin
     if i < np
     then begin
      tsc:=sc*ct+cc*st;
      tcc:=cc*ct-sc*st;
      sc:=tsc;
      cc:=tcc;
      x1:=x2;
      y1:=y2;
      r1:=r2;
      ip:=i;
     end
     else begin
      if abs(round(sarco)) > 0
      then begin
      isr:=1;
      loc:=1;
      if sc < 0.00 then isr:=-1;
      i:=np;
     end
     else begin
      loc:=-1;
      i:=np;
     end;
     end;
    end
    else begin
     loc:=0;
     i1:=ip;
     i2:=i;
     i:=np;
    end;
    end
    else begin
     loc:=2;
     i1:=i;
     i2:=i;
     if r1 <> 0.00
     then i:=np
     else begin
       i1:=ip;
       i2:=ip;
       i:=np;
      end;
     end;
    end
    else begin
     loc:=2;
     i1:=i;
     i2:=i;
     if r1 <> 0.00
     then i:=np
     else begin
      i1:=ip;
      i2:=ip;
      i:=np;
     end;
     end;
    i:=i+1;
    end;
    end;
```

Figura 3 - Listagem do algorítmo com as modificações propostas, adaptado para TURBO-PASCAL.