ALGORITMI geometrici elementari

Grigore ALBEANU

Pornind de la câteva rezultate simple de geometrie analitică plană, sunt prezentate metode algoritmice pentru rezolvarea problemelor privind intersecția segmentelor de dreaptă și a poligoanelor. În final este abordată problema punctului interior. Pentru toate temele se discută și aspecte relative la complexitatea rezolvării.

1. Preliminarii

1.1. Elemente de geometrie analitică

Fie un *plan P* și *(Ox, Oy)* o pereche ordonată de *axe ortogonale* în planul *P*,

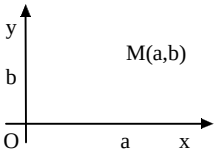

Fig. 1: *Sistem de axe ortogonale în planul P*

cu *originea O* comună și cu sensurile pozitive ale axelor indicate prin săgeți ca în fig. 1. Acest ansamblu este numit *reper cartezian* în planul *P*, este notat cu *xOy*, iar *O* este originea reperului. Pentru orice punct *M* din planul *P* este definită funcția bijectivă $f : P \to R_2$, $f(M)=(a,b)$ care asociază punctului *M* coordonatele carteziene *a* și *b*.

Următoarele rezultate vor fi considerate elementare:

a) *Formula distanței*: Fie $M_1(x_1,y_1)$ și $M_2(x_2,y_2)$ puncte ale planului *P* specificate în reperul cartezian *xOy*. Dacă prin M_1M_2 notăm distanța dintre punctele M_1 și M_2, atunci

$$M_1M_2 = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

b) *Formula pantei unei drepte*: Fie *d* o dreaptă neparalelă cu axa *Oy* pe care fixăm două puncte distincte $M_1(x_1,y_1)$ și $M_2(x_2,y_2)$. Panta (coeficientul unghiular al) dreptei *d* este dată de formula

$$m = \frac{y_2 - y_1}{x_2 - x_1}, x_1 \neq x_2.$$

c) *Caracterizarea dreptelor oblice paralele/perpendiculare*: Dreptele oblice d_1 și d_2, de pante m_1 respectiv m_2 sunt paralele dacă și numai dacă $m_1=m_2$. Ele sunt perpendiculare dacă și numai dacă $m_1m_2=-1$.

d) *Ecuația dreptei oblice*: Fie dreapta *d* definită prin punctul $M_0(x_0,y_0)$ situat pe *d* și prin panta *m*. Atunci punctul $M(x,y)$ aparține dreptei *d* dacă și numai dacă $y-y_0=m(x-x_0)$. Fie dreapta *d* definită prin punctele $M_1(x_1,y_1)$ și $M_2(x_2,y_2)$. Dacă $x_1=x_2$, atunci dreapta *d* este verticală și are ecuația $x=x_1$. Dacă $y_1=y_2$, atunci dreapta *d* este orizontală și are ecuația $y=y_1$. Dacă $x_1 \neq x_2$ atunci dreapta *d* este oblică, iar ecuația acesteia poate fi scrisă în oricare din formele echivalente:

$$y - y_1 = \frac{y_2 - y_1}{x_2 - x_1}(x - x_1)$$

sau

$$\frac{x - x_1}{x_2 - x_1} = \frac{y - y_1}{y_2 - y_1}$$

sau (exprimarea robustă)

$$(x - x_1)(y_2 - y_1) - (y - y_1)(x_2 - x_1) = 0.$$

e) *Condiția de coliniaritate a trei puncte* $M_i(x_i,y_i)$, i =1, 2, 3:

$$(x_3 - x_1)(y_2 - y_1) - (y_3 - y_1)(x_2 - x_1) = 0.$$

f) *Rapoarte*: Fie $M_1(x_1,y_1)$ și $M_2(x_2,y_2)$ două puncte diferite care determină segmentul M_1M_2 si dreapta (M_1M_2). Fie *M* pe dreapta (M_1M_2) diferit de M_2, $M(x,y)$. Numărul *k* definit prin regula:

Dacă M aparține segmentului M_1M_2 *atunci:*

$$k = -\frac{MM_1}{MM_2}$$

$$k = \frac{MM_1}{MM_2}$$

altfel

se numește *raportul în care punctul M împarte segmentul nenul M_1M_2 ($k\neq1$)*. Fiind cunoscut numărul real $k\neq1$, coordonatele (x,y) ale punctului M care împarte segmentul M_1M_2 în raportul k sunt:

$$x = \frac{x_1 - kx_2}{1-k}, y = \frac{y_1 - ky_2}{1-k}.$$ s

g) *Ecuațiile parametrice ale dreptei*: Fie M_1, M_2 ca mai sus. Egalitățile:

$$x = x_1 + (x_2 - x_1)t;\ y = y_1 + (y_2 - y_1)t, t \in \mathrm{R}$$

se numesc ecuațiile parametrice ale dreptei (M_1M_2).

Dacă impunem $t\in[0,1]$ atunci obținem segmentul M_1M_2. Mijlocul segmentului M_1M_2 se obține pentru $t=1/2$.

h) *Centrul de greutate*: Centrul de greutate al unui sistem omogen de n puncte: M_1, M_2, ..., M_n cu $M_i(x_i,y_i)$, $i=1, 2, ..., n$, are drept coordonate: media aritmetică a absciselor, respectiv media aritmetică a ordonatelor.

i) *Distanța de la un punct la o dreaptă:* Formula distanței de la punctul $M_0(x_0,y_0)$ la dreapta $h: ax+by+c=0$ este:

$$d(M_0,h) = \frac{|ax_0 + by_0 + c|}{\sqrt{a^2 + b^2}}.$$

j) *Aria unui triunghi*: Fie $M_i(x_i,y_i)$, $i=1,2,3$, puncte necoliniare. Aria triunghiului $M_1M_2M_3$ este dată prin:

$$\sigma(M_1M_2M_3) = \pm\frac{1}{2}[(x_2 - x_1)(y_3 - y_1) - (x_3 - x_1)(y_2 - y_1)].$$

Semnul se ia + sau - astfel încât aria triunghiului să fie pozitivă. Dacă M_1, M_2, M_3 sunt specificate în ordinea inversă a acelor de ceasornic atunci semnul este +. Deci, condiția ca trei puncte M_1, M_2, M_3, date prin coordonatele carteziene (x_i,y_i), să fie specificate în sens invers acelor de ceasornic se scrie:

$$(x_2 - x_1)(y_3 - y_1) - (x_3 - x_1)(y_2 - y_1) > 0.$$

k) *Semiplane*: O dreaptă d din planul P are proprietate că separă planul P în două submulțimi. Fie $f:R_2\to R$, $f(x,y)=ax+by+c$. Definim $P-$ – mulțimea punctelor pentru care $f(x,y)<0$, iar $P+$ – mulțimea punctelor pentru care $f(x,y)>0$. Evident, d este mulțimea punctelor pentru care $f(x,y)=0$. Mulțimile: $P-$, $P+$ se numesc *semiplane deschise*. Dreapta d se numește *frontiera semiplanelor*. Deoarece semiplanele sunt mulțimi convexe, orice intersecție de semiplane este o mulțime convexă. Pentru aflarea semnului unui semiplan se utilizează metoda sondajului: se alege un punct particular (x_0,y_0) și se vede semnul numărului $f(x_0,y_0)$. Semiplanul opus are semn contrar.

1.2. Câteva funcții utile

În această secțiune sunt prezentate două funcții elementare, utile în secțiunile următoare.

a) Funcția *s*

Fie d o dreaptă determinată de două puncte $A(x_A,y_A)$ și $B(x_B,y_B)$. Fie Q un punct oarecare din planul celor două puncte. Notăm prin $s(Q,A,B)$ funcția semn a punctului Q relativ la dreapta (AB), $s(Q,A,B)=1$ dacă $Q\in P^+$, $s(Q,A,B)=-1$ dacă $Q\in P^-$, respectiv $s(Q,A,B)=0$ dacă $Q\in(AB)$.

Conform celor prezentate mai sus, notând

$E := (x_Q - x_A)(y_B - y_A) - (y_Q - y_A)(x_B - x_A)$,

obținem:

Dacă $E>0$ *atunci* $s(Q,A,B)=1$, *dacă* $E=0$ *atunci* $S(Q,A,B)=0$, *iar dacă* $E<0$ *atunci* $s(Q,A,B)=-1$. Observăm că pentru evaluarea funcției s sunt necesare cinci operații de scădere, două operații de înmulțire și cel mult două comparații cu zero.

b) Funcția *p*

Fie $A(x_A,y_A)$, $B(x_B,y_B)$, $C(x_C,y_C)$ trei puncte din planul xOy. Ne interesează dacă trecerea de la punctul A la punctul C prin intermediul punctului B se realizează în sens trigonometric sau nu (când punctele nu sunt coliniare), respectiv dacă C este între A și B (când punctele sunt coliniare). Definim funcția p astfel (fig. 2): $p(A,B,C)=-1$ dacă trecerea de la A la C prin B se face în sens invers trigonometric (în sensul acelor de ceasornic) sau A este între B și C (în cazul punctelor coliniare); $p(A,B,C)=0$ dacă C este între A și B (numai în cazul punctelor coliniare) și $p(A,B,C)=1$ atunci când deplasarea se face în sens trigonometric (în sens invers acelor de ceasornic) sau când B este între A și C (în cazul punctelor coliniare).

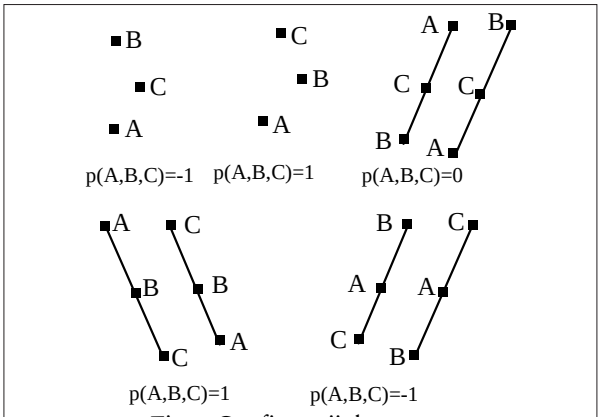

Fig 2. Configurații de puncte

În continuare vom propune câteva metode de evaluare a funcției p.

a) Metoda pantelor

Fie m_1 panta dreptei (AB) și m_2 panta dreptei (AC). Inițial presupunem că $m_1\neq m_2$. Dacă $m_1>m_2$ atunci pentru a ajunge în punctul C plecând din punctul A, trecem în deplasare invers trigonometrică prin B, deci $p(A,B,C)=-1$. Dacă $m_1<m_2$ atunci deplasarea se va face în sens trigonometric, deci $p(A,B,C)=1$. Deoarece $m_1=(y_B-y_A)/(x_B-x_A)=dy_1/dx_1$, $m_2=(y_C-y_A)/(x_C-x_A)=dy_2/dx_2$, testul de comparare între m_1 și m_2 (ce necesită două împărțiri plus o

verificare suplimentară în cazul segmentelor verticale), poate fi transformat astfel:

$m_1>m_2 \Rightarrow dy_1/dx_1>dy_2/dx_2$ Ţ $dy_1dx_2>dx_1dy_2$ Ţ $p(A,B,C)=-1$;

$m_1<m_2 \Rightarrow dy_1dx_2<dx_1dy_2 \Rightarrow p(A,B,C)=1$.

Dacă $m_1=m_2$ se va face şi o analiză a lungimii segmentelor AB şi AC. Prin urmare: $p(A,B,C)=1$ dacă $dx_1dy_2 - dy_1dx_2>0$ sau punctele sunt coliniare şi $AB<AC$; $p(A,B,C)=0$ dacă punctele sunt coliniare şi $AC<AB$, $p(A,B,C)=-1$ dacă $dx_1dy_2 - dy_1dx_2<0$ sau punctele sunt coliniare şi, fie $dx_1dx_2<0$, fie $dy_1dy_2<0$. Deoarece se poate evita calculul radicalului prin compararea pătratelor lungimilor segmentelor, pentru evaluarea funcţiei p sunt necesare maxim şapte adunări/scăderi, maxim patru înmulţiri, maxim două comparaţii şi patru operaţii de atribuire.

b) Metode trigonometrice

Se pot utiliza formule trigonometrice pentru determinarea unghiurilor dreptelor (AB) şi (AC), dar utilizarea acestora necesită timp mare de calcul. Lăsăm cititorului plăcerea de a face un studiu în această direcţie.

2. Intersecţia a două segmente de dreaptă

Formularea problemei:

Fie AB şi CD două segmente cu extremităţile specificate prin coordonate carteziene: $A(x_A,y_A)$, $B(x_B,y_B)$, $c(x_C,y_C)$, $D(x_D,y_D)$. Decideţi dacă segmentele AB şi CD au intersecţia nevidă.

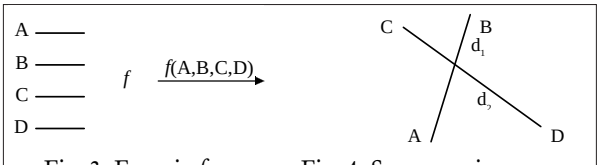

Fig. 3. Funcţia f

Fig. 4. Segmente intersectate

Fie f funcţia cu valori logice ce ia valoarea *true* dacă segmentele AB şi CD au intersecţia nevidă şi valoarea *false* altfel (fig. 3).

2.1. Metoda analitică

Pornim de la următoarea observaţie (fig. 4): Segmentele AB şi CD au intersecţia nevidă dacă şi numai dacă

i) A şi B sunt separate de dreapta (CD), notată d_2 şi

ii) C şi D sunt separate de dreapta suport (AB), notată d_1.

Pentru a stabili relaţiile analitice ce exprimă condiţiile i) şi ii), utilizăm ecuaţiile dreptelor d_1 şi d_2 precum şi proprietatea de separare în semiplane. Avem

$$(d_1): ux + vy + w = 0, u = y_B - y_A, v = x_A - x_B, w = y_A x_B - x_A y_B.$$

Atunci C şi D sunt separate de dreapta (d_1) dacă şi numai dacă

$$(ux_C + vy_C + w).(ux_D + vy_D + w)<0 \qquad (1a)$$

De asemenea :

$$(d_2): u' x + v' y + w' = 0, u' = y_D - y_C, v' = x_C - x_D, w' = y_C x_D - x_C y_D,$$

iar A şi B sunt separate de dreapta (d_2) dacă şi numai dacă

$$(u'x_A + v'y_A + w').(u'x_B + v'y_B + w')<0. \qquad (1b)$$

Deci pentru metoda analitică obţinem: $f(A,B,C,D)$ = *true* dacă relaţiile (1a) şi (1b) au loc simultan, şi $f(A,B,C,D)$ = *false* în caz contrar. Numărul de operaţii necesare pentru evaluarea funcţiei f este prezentat în tabelul 1.

Observaţie:

Această metodă nu determină şi coordonatele punctului de intersecţie (în caz că există).

2.2. Metoda vectorială

Vom folosi noţiunile de vector, produs scalar a doi vectori şi produs vectorial. Vom considera planul xOy şi vectorii AC, AD, BC şi BD (fig. 5). Produsele vectoriale $AC \times AD$ şi $BC \times BD$ sunt vectori perpendiculari pe planul xOy. Dacă aceştia sunt de sens opus atunci produsul lor scalar este negativ. Dacă au acelaşi sens, produsul lor scalar este pozitiv.

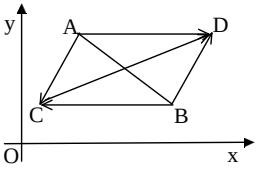

Fig. 5. Vectorii AC, AD, BC şi BD

Putem formula următoarea observaţie: *Intersecţia segmentelor AB şi CD este nevidă dacă şi numai dacă, fie produsul scalar al vectorilor $AC \times AD$ şi $BC \times BD$ este negativ, fie produsul scalar al vectorilor $CA \times CB$ şi $DA \times DB$ este negativ.*

Se ştie că într-un spaţiu ortonormat $Oxyz$ cu vectorii unitari i, j, k, produsul vectorial $AC \times AD$ se scrie astfel:

$$AC \times AD = ((x_C - x_A)(y_D - y_A) - (x_D - x_A)(y_C - y_A))k.$$

Similar

$$BC \times BD = ((x_C - x_B)(y_D - y_B) - (x_D - x_B)(y_C - y_B))k.$$

Atunci produsul scalar $(AC \times AD) \cdot (BC \times BD)$, notat cu p, este egal cu:

$$p := ((x_C - x_A)(y_D - y_A) - (x_D - x_A)(y_C - y_A))$$
$$((x_C - x_B)(y_D - y_B) - (x_D - x_B)(y_C - y_B)).$$

Analog, produsul scalar $(CA \times CB).(DA \times DB)$, notat cu q, este egal cu:

$$q := ((x_A - x_C)(y_B - y_C) - (x_B - x_C)(y_A - y_C)).$$
$$. ((x_A - x_D)(y_B - y_D) - (x_B - x_D)(y_A - y_D)).$$

Obţinem că $f(A,B,C,D)$ = *true* dacă şi numai dacă $p<0$ şi $q<0$. În caz contrar, $f(A,B,C,D)$ = *false*. După prelucrarea expresiilor lui p şi q, identificarea factorilor/termenilor ce se repetă în procesul de evaluare se obţine necesarul de operaţii aritmetico-logice din tabelul 1.

2.3. Metoda parametrică

În continuare vor fi utilizate ecuaţiile parametrice ale unei drepte ce trece prin două puncte (vezi secţiunea 1.1, g):

$$(d_1)\ x = x_A + (x_B - x_A)u;\ y = y_A + (y_B - y_A)u, u \in R;$$

(d_2) $x = x_C + (x_D - x_C)v;\ y = y_C + (y_D - y_C)v, v \in R.$

Atunci segmentul *AB* (respectiv *CD*) se obține pentru $u \in [0,1]$ (respectiv $v \in [0,1]$). Dacă cele două segmente se intersectează, atunci dreptele suport d_1 și d_2 au intersecția nevidă. Deci, sistemul cu două ecuații liniare în necunoscutele u și v are soluțiile:

$$u = \frac{(x_C - x_A)(y_C - y_D) - (x_C - x_D)(y_C - y_A)}{(x_B - x_A)(y_C - y_D) - (x_C - x_D)(y_B - y_A)},$$

$$v = \frac{(x_B - x_A)(y_C - y_A) - (x_C - x_A)(y_B - y_A)}{(x_B - x_A)(y_C - y_D) - (x_C - x_D)(y_B - y_A)}$$

Obținem că: *f(A,B,C,D)=true* dacă și numai dacă $u \in [0,1]$ și $v \in [0,1]$ simultan. Altfel *f(A,B,C,D)=false*. Complexitatea evaluării funcției *f* este prezentată în tabelul 1.

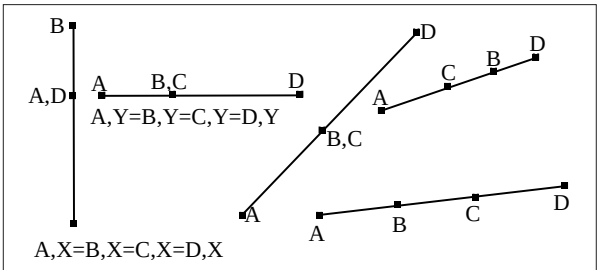

Fig. 6. Cazuri particulare

Observații:

1. Este necesar un test preliminar privind paralelismul dreptelor suport. Aceasta echivalează cu verificarea egalității:

$$(x_C - x_D)(y_B - y_A) = (x_B - x_A)(y_C - y_D).$$

Totuși, dacă punctele sunt coliniare, iar segmentele au cel puțin un punct comun, acest aspect trebuie tratat separat (fig. 6).

2. Dacă segmentele se intersectează atunci se obțin foarte ușor și coordonatele punctului de intersecție.

2.4. Metoda Min-Max

Dacă dreptele suport *(AB)* și *(CD)* se intersectează, atunci segmentele *AB* și *CD* au un punct comun dacă punctul de intersecție se află în interiorul intersecției dreptunghiurilor de diagonale *AB* și *CD* (fig. 7). Folosim notațiile prezentate în cadrul metodei analitice (secțiunea 2.1). Dacă dreptele suport au intersecția nevidă atunci coordonatele punctului de intersecție sunt:

$$x_I = \frac{vw' - v'w}{uv' - u'v}, y_I = \frac{wu' - w'u}{uv' - u'v}$$

Atunci cele două segmente se intersectează (într-un singur punct) dacă și numai dacă:

$$max(min(x_A,x_B), min(x_C,x_D)) \le x_I \le min(max(x_A,x_B), max(x_C,x_D)) \ (x\_int)$$
$$\text{și } max(min(y_A,y_B), min(y_C,y_D)) \le y_I \le min(max(y_A,y_B), max(y_C,y_D)).(y\_int)$$

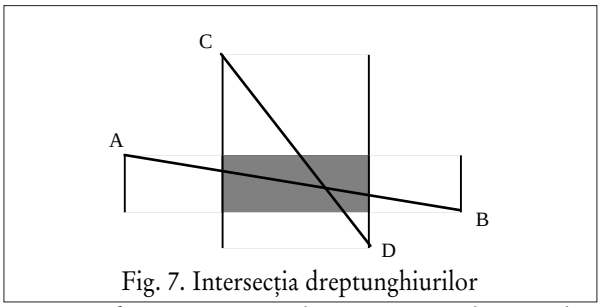

Fig. 7. Intersecția dreptunghiurilor

Deci, *f(A,B,C,D)=true* dacă și numai dacă au loc simultan relațiile *(x_int)* și *(y_int)*. În caz contrar, *f(A,B,C,D)=false*.

Complexitatea evaluării funcției *f* prin această metodă este prezentată în tabelul 1.

2.5. Metoda funcției p

Utilizând funcția *p* (secțiunea 1.2), putem defini funcția *f* astfel: *f(A,B,C,D)=true* dacă și numai dacă

$$P(A,B,C) * p(A,B,D) \le 0 \text{ și}$$
$$p(C,D,A) * p(C,D,B) \le 0 \text{ simultan.}$$

Altfel, considerăm *f(A,B,C,D)=false*. Numărul de operații necesare evaluării funcției *f* prin această metodă este prezentat în tabelul 1. Să observăm, că prin metoda funcției *p* nu se determină punctul de intersecție, dacă acesta există.

Metoda	+ (-)	*	/	<	and sau or	:=	Punctul de inters.
Analitica	14	14	0	2	1	0	NU
Vectoriala	20	10	0	2	1	0	NU
Parametrica	9	5	2	4	1	7	DA!
Min-Max	9	9	1	16	1	19	DA
Funcția p	28	8	0	8	1	16	NU

Tabelul 1. *Numărul de operații necesar pentru evaluarea funcției f prin metodele prezentate.*

3. Intersecția poligoanelor

În general intersecția a două obiecte grafice prezintă un interes special în grafica pe calculator [1,4,5,6]. Algoritmii de decupare (clipping) la limita unei ferestre dreptunghiulare sau unei ferestre poligonale, algoritmii de vizibilitate a obiectelor descrise în spațiu se bazează pe intersecții de poligoane. În prima parte se vor considera algoritmi privind intersecția dintre un poligon convex și un dreptunghi. Se vor considera apoi, algoritmi pentru determinarea intersecției a două poligoane oarecare.

3.1. Algoritmul Liang-Barsky

Fie *P* un poligon oarecare și *D* un dreptunghi cu laturile paralele cu axele sistemului de coordonate, situat în același plan cu poligonul *P*. Algoritmul *Liang-Barsky* folosește reprezentarea parametrică a laturilor poligonului *P* și calculează intersecțiile cu laturile dreptunghiului numai atunci când este necesar. Presupunem că poligonul *P* este dat printr-o secvență de puncte $P_1, P_2, ..., P_n$, ce definesc

laturile poligonului. Latura *i* începe din punctul P_i și este orientată către punctul P_{i+1}. Ecuația parametrică (vezi secțiunea 1.1) a laturii P_iP_{i+1} este:

$$R.X(t)=(1-t)\,P_i.X + tP_{i+1}.X,\ R.Y(t)=$$
$$(1-t)P_i.Y + tP_{i+1}.Y,\ 0<t\leq 1.$$

Observăm că nu se ia în considerare punctul de start al laturii. Astfel, fiecare vârf al poligonului *P* aparține unei singure laturi din cele două adiacente în acel vârf. Notăm cu x_i dreapta determinată de punctele P_i și P_{i+1}. Aceasta intersectează (dacă nu e paralelă cu laturile dreptunghiului) cele patru drepte suport pentru laturile dreptunghiului *D*. Fiecare astfel de dreaptă împarte planul celor două obiecte grafice în două semiplane. Vom numi semiplan interior semiplanul ce conține dreptunghiul *D*. Astfel planul este împărțit în nouă regiuni (fig. 8). Regiunile sunt notate în funcție de numărul de semiplane interioare în care sunt situate.

2	3	2
3	4	3
2	3	2

Fig. 8. Divizarea planului în regiuni

O latură care intră cu extremitatea finală într-o regiune de tip 2 determină includerea vârfului dreptunghiului în lista de ieșire a punctelor intersecției celor două obiecte. Acest vârf este numit punct de întoarcere. Totuși, dacă următoarea latură care intersectează dreptunghiul are punctul de intersecție situat pe aceeași latură care a determinat un punct de întoarcere, atunci acest vârf de întoarcere este incorect și nu trebuie inserat în lista de ieșire. O dreptă x_i diagonală intersectează laturile dreptunghiului *D* în puncte ce corespund valorilor t_{I1}, t_{I2}, t_{E1}, t_{E2}. Deoarece o latură diagonală x_i începe și se termină într-o regiune de tip 2, t_{I1} și t_{E2} reprezintă minimul, respectiv maximul celor patru valori. Celelalte două valori determină dacă x_i intersectează dreptunghiul sau nu. În cazul în care $t_{E1}<0$ și $t_{E2} \leq 1$, latura interesectează dreptunghiul și segmentul vizibil trebuie adăugat în lista de ieșire. Dacă latura nu intersectează dreptunghiul, dreapta care o conține începe în, trece prin și se termină într-o regiune de tip 2 (fig. 9).

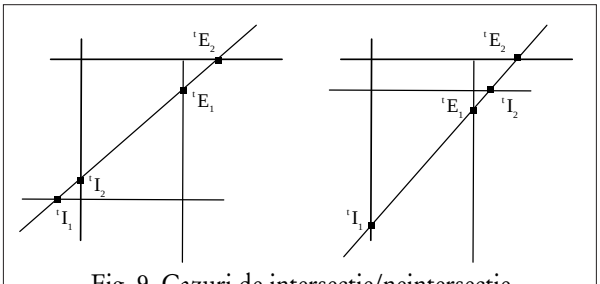

Fig. 9. Cazuri de intersecție/neintersecție

Netratând cazul particular al laturilor poligonului *P*, paralele cu laturile dreptunghiului *D*, ideile principale ale algoritmului *Liang-Barsky* sunt [7,8]:

Pentru fiecare latură x_i a poligonului P se efectuează următoarele operații:

1. Determină direcția laturii pentru a afla care este prima dreaptă a dreptunghiului intersectată de dreapta pe care se află latura x_i.
2. Determină valorile *t* pentru punctele de ieșire: t_{E1} și t_{E2}.
3. Dacă $t_{E2} > 0$ atunci determină t_{I1} și t_{I2}.
4. Dacă $t_{I2} > t_{E1}$ atunci
   Dacă $0 < t_{E1} \leq 1$ atunci pune vârful de întoarcere în lista de ieșire.
5. Dacă $t_{I2} \leq t_{E1}$ atunci există o parte comună:
   Dacă $0 < t_{E1}$ și $t_{I2} \leq 1$ atunci
   Dacă $t_{I2} \geq 0$ atunci pune, în lista de ieșire, punctul de intersecție;
   Dacă $t_{I2} < 0$ atunci pune, la ieșire, punctul de start;
   Dacă $t_{E1} \leq 0$ atunci pune, la ieșire, punctul de intersecție;
   Dacă $t_{E1} > 1$ atunci pune, la ieșire, punctul final.
   Dacă $0 < t_{E2} \leq 1$ atunci pune, în lista de ieșire, punctul de întoarcere.

Observații:

1. Algoritmul are complexitate liniară, numărul de etape necesare fiind $4*n$, iar pentru fiecare etapă este necesar un număr constant de operații aritmetice.

2. Generalizarea algoritmului este dificilă, datorită mulțimii de valori t_I, t_E ce apar.

3.2. Algoritmul Sutherland-Hodgman

Fie *P* și *Q* două poligoane convexe situate în același plan. Algoritmul *Sutherland-Hodgman* [10], acceptă ca date de intrare lista vârfurilor poligonului *P* ce definesc laturile acestuia, precum și laturile poligonului *Q*. Pentru a determina intersecția celor două poligoane, pentru fiecare latură a poligonului *Q* se parcurge lista vârfurilor poligonului *P*, examinându-se relația dintre oricare două vârfuri succesive ale poligonului *P* și latura poligonului *Q*, obținându-se zero, unul sau două vârfuri (într-o listă de ieșire *L*). Dreapta determinată de latura curentă a poligonului *Q* împarte planul celor două poligoane în două semiplane. Semiplanul care conține poligonul *Q* va fi numit semiplanul interior.

Fie *A* și *B* două vârfuri succesive ale poligonului *P*. Presupunem că punctul *A* a fost analizat în pasul anterior. Sunt posibile patru cazuri (fig. 10). În primul caz vârfurile *A* și *B* sunt în semiplanul interior, algoritmul adaugă vârful *B* în lista *L*. În cazul al doilea, latura *AB* intersectează latura curntă în punctul *I*, Vârful *A* fiind în semiplanul interior, iar *B* în cel exterior. Algoritmul adaugă punctul de intersecție *I* în lista de ieșire. În cazul al treilea vârfurile *A* și *B* sunt în semiplanul exterior, caz în care algoritmul nu adaugă nimic în lista *L*. În cazul al patrulea, latura *AB* intersectează latura poligonului *Q* în punctul *I*, vârful *B* fiind în semiplanul interior, iar *A* în cel exterior. Algoritmul adaugă punctele *I* și *B* în lista de ieșire.

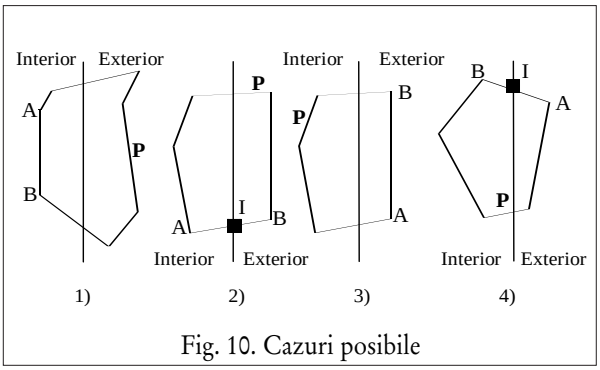

Fig. 10. Cazuri posibile

Procedura `Decupare_latura` acceptă la intrare:

`VP` — vectorul vârfurilor poligonului *P*,

`NP` — numărul vârfurilor poligonului *P*,

`x` — latura curentă a poligonului *Q*.

La ieșire se obțin:

`L` — vectorul vârfurilor obținute conform descrierii de mai sus,

`NL` — numărul vărfurilor depuse în lista `L`.

Presupunem că dispunem de o procedură eficientă, numită `intersectie` care determină punctul de intersecție dintre o latură a poligonului *P* și latura *x* precum și de funcții pentru clasificarea semiplanelor.

Pașii procedurii `Decupare_latura` sunt următorii:

```
1. NL:=0; s:=VP(NP);
2. for j:=1 to NP do
   begin
   p:=VP(j);
   if în_semiplan_interior(p)
   then
     if în_semiplan_interior(s)
     then L(NL):=p; Inc(NL)
     else i:=intersectie(s,p,x);
               L(NL):=i; Inc(NL); L(NL):=p;
Inc(NL)
   else
     if în_semiplan_exterior(s)
     then i:=intersectie(s,p,x);
          L(NL):=i; Inc(NL);
   s:=p
end;
```

Această procedură se apelează de atâtea ori câte laturi are poligonul *Q*. Pentru a descrie algoritmul pentru determinarea vârfurilor intersecției poligoanelor *P* și *Q*, notăm cu x_i latura *i* a poligonului *Q*, `V1` și `V2`, vectorii ce conțin date inițiale, intermediare sau finale privind vârfurile intersecției, cu `NV1`, respectiv `NV2` elemente.

Pașii algoritmului *Sutherland-Hodgman* sunt:

1. `V2:=VP; NV2:=NVP;`
2. **Pentru** *fiecare latură* x_i *a poligonului Q* **execută**
   2.1 `V1:=V2; NV1:=NV2;`
   2.2 `Decupare_latura(V1,NV1,`x_i`,V2,NV2);`

Vârfurile intersecției celor două poligoane se vor afla în tabloul *V2*, după ce s-a parcurs și ultima latură a poligonului *Q*.

Observații:

1. Dacă poligonul *Q* este un dreptunghi, atunci algoritmul *Sutherland-Hodgman* este considerat a fi un algoritm de decupare pentru vizualizarea informației grafice la nivelul unei ferestre de vizualizare.

2. Dacă poligonul *P* este concav pot rezulta laturi suplimentare pe conturul poligonului *Q*; aceste laturi false unesc poligoanele care s-ar obține în urma intersecției celor două poligoane. De aceea este necesară o postprelucrare în urma căreia sunt eliminate laturile false.

3.3. Algoritmul Weiler-Atherton

Acest algoritm [11], prelucrează lista de vârfuri furnizată de algoritmul *Sutherland-Hodgman*, după intersecția cu fiecare latură a poligonului *Q*. Algoritmul începe prin divizarea poligonului *P* față de una din laturile poligonului *Q*. Rezultatul este o listă de poligoane *LP*. Fiecare poligon din această listă este apoi divizat față de următoarea latură a poligonului *Q*, ș.a.m.d. Această prelucrare este redată în secvența următoare:

Pentru fiecare latură x a poligonului Q, pentru fiecare poligon P din lista LP execută Divide(x,P,LP). Procedura *Divide* sparge un poligon *P* din lista inițială *LP* față de latura *x* a poligonului *Q*. Dacă *P* este divizat în două sau mai multe poligoane atunci acestea sunt inserate în listă, iar poligonul inițial este eliminat.

4. Problema punctului interior

Fie *F* o figură plană a cărei frontieră este compusă din segmente. Considerăm definiția naturală a interiorului unei figuri plane. Dacă *Q* este un punct situat în același plan cu figura *F*, problema punctului interior constă în stabilirea valorii de adevăr a propoziției: *Q este un punct interior figurii F.*

În această secțiune se vor considera următoarele cazuri pentru figura F: triunghi, patrulater, poligon convex, poligon oarecare. Pentru alte aspecte se poate consulta [9].

4.1. Punct interior unui triunghi

Fie triunghiul *ABC* ale cărui vârfuri sunt $A(x_A,y_A)$, $B(x_B,y_B)$ și $C(x_C,y_C)$. Fie $M(x_M,y_M)$ un punct din planul triunghiului. Fie `pint3(A,B,C,M)` funcția care ia valoarea `true` dacă *M* este interior triunghiului *ABC* și `false`, în caz contrar. Funcția `pint3` poate fi evaluată în mai multe moduri. Prezentăm două abordări: prima soluție utilizează funcția *p*, iar a doua soluție folosește funcția *s*. Pentru fiecare variantă se face studiul complexității.

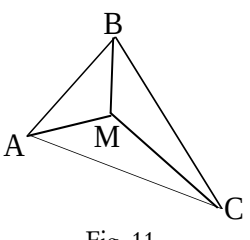

Fig. 11.

a) Metoda funcției p

În acest caz (fig. 11), funcția `pint3` poate fi definită astfel:

`pint3(A,B,C,M)=true` dacă $p(A,M,B)=1$ și $p(B,M,C)=1$ și $p(C,M,A)=1$ simultan, în caz contrar `pint3(A,B,C,M)=false`.

Luând în considerare complexitatea evaluării funcției *p* rezultă că pentru evaluarea funcției `pint3` prin această metodă sunt necesare 21 de operații de adunare/scădere, 12 înmulțirii, 9 teste relaționale, 2 operații logice și 12 operații de atribuire.

Observăm că punctul *M* aparține unei laturi a triunghiului dacă $p(A,B,M)=0$ sau $p(B,C,M)=0$ sau $p(C,A,M)=0$ cu aceeași complexitate de evaluare precum a funcției `pint3`.

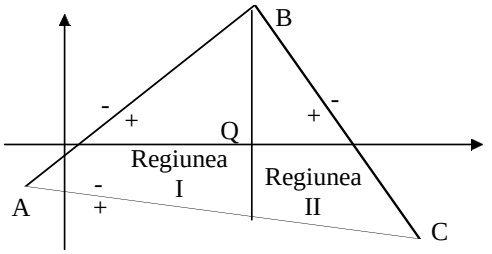

Fig. 12.

b) Metoda semnului

Vom presupune, cu păstrarea generalității, că $x_A \le x_B \le x_C$. Fie *Q* proiecția punctului *B* pe axa *Ox* (fig. 12). Dacă $x_M < x_B$ atunci analizăm regiunea I altfel analizăm regiunea II. Prezentăm în continuare textul pseudo-Pascal al funcției `pint3` prin această metodă [3]:

```
function pint3(A,B,C,M:point):Boolean;
var s1,s2:Real;
begin
  if (xM<xA) or (xM<xC)
  then pint3:=false
  else
    begin
      s2:=(xM-xA)*(yC-yA)-(yM-yA)*(xC-xA);
      if (xM<xB) or (xB=xC)
      then s1:=(xM-xA)*(yB-yA)-(yM-yA)*(xB-xA)
      else s1:=(yM-yC)*(xB-xC)-(xM-xC)*(yB-yC);

      pint3:=((s1≥) and (s2≤0)) or ((s1≤0)
                    and (s2≥0))
    end
end;
```

Prin această metodă sunt necesare 10 adunări/scăderi, 4 înmulțiri, 8 comparații, 4 operații logice **and/or** și maxim 3 operații de atribuire. Se observă că această metodă este mai rapidă decât metoda bazată pe utilizarea *funcției p*.

4.2. Punct interior unui patrulater simplu

Se consideră patrulaterul simplu *ABCD* (fără autointersectări) ale cărui vârfuri sunt date prin coordonate carteziene: $A(x_A,y_A)$, $B(x_B,y_B)$, $C(x_C,y_C)$, $D(x_D,y_D)$ și $M(x_M,y_M)$ un punct din planul *xOy* (fig. 13).

Fie `pint4(A,B,C,D,M)` funcția ce ia valoarea `true` dacă *M* este punct interior patrulaterului *ABCD* și `false` în caz contrar. O metodă de evaluare a funcției `pint4` presupune utilizarea funcției *p*, dar putem utiliza și funcția `pint3`, dacă împărțim patrulaterul în două triunghiuri prin intermediul unei diagonale.

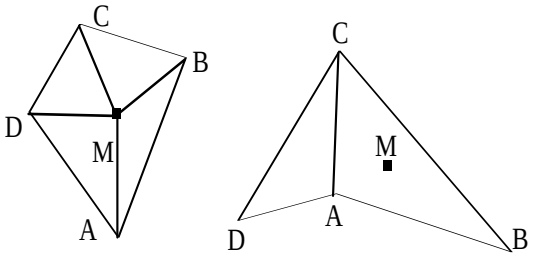

Fig. 13. Patrulatere simple
a: metoda *p*, b:triangularizare

a) Metoda funcției p

În acest caz (fig. 13/a), funcția `pint4` poate fi definită astfel:

`pint4(A,B,C,D,M)=true` dacă simultan au loc egalitățile:

$p(A,M,D)=1$, $p(D,M,C)=1$, $p(C,M,B)=1$
și $p(B,M,A)=1$.

În caz contar `pint4(A,B,C,D,M)=false`.

Deci evaluarea funcției `pint4` necesită 28 de operații de adunare/ scădere, 16 operații de înmulțire, 12 comparații, 3 oparații logice și 16 atribuiri; ceea ce înseamnă un efort de calcul apreciabil.

b) Metoda triangularizării

Condsiderând diagonala *AC*, patrulaterul *ABCD* se împarte în două triunghiuri *(ABC* și *DAC)*. Deci:

`pint4(A,B,C,D,M)=pint3(A,B,C,M)`
or `pint3(D,A,C,M)`,

fiind necesare 20 adunări/scăderi, 8 înmulțiri, 16 comparații, 9 operații logice și cel mult 6 atribuiri.

4.3. Punct interior unui poligon convex

Pentru evaluarea funcției `pint(poligon,M)` se poate utiliza fie funcția *p*, fie metoda triangularizării (ducând diagonale sau considerând centrul de greutate și formând triunghiuri). Dacă evaluarea funcției `pint` este solicitată de mai multe ori pentru același poligon, atunci se justifică utilizarea următorului algoritm eficient[9]:

Se creează un arbore de căutare cu vărfurile poligonului convex, având drept criteriu de ordonare unghiul pe care dreapta care unește centrul de greutate și un vârf, îl

face cu axa Ox. Astfel, dacă patrulaterul are n vârfuri, se evidențiază n sectoare. Mai întâi, prin căutare în arbore, se determină sectorul în care se găsește punctul M, iar apoi se vede dacă M este în interior sau în exterior folosind, de exemplu, funcția `p` *sau funcția* `pint3` *apelată o singură dată.*

4.4. Punct interior unui poligon oarecare

Fie ***pol*** un poligon ale cărui vârfuri sunt ***pol(i)*** pentru i=1, 2, ..., n, iar laturile lui sunt segmente cu extremitățile vârfuri consecutive (presupunem că ***pol***(0)=***pol***(n) și ***pol*** (n+1) =***pol***(1)). Fie M un punct din planul poligonului. Trasăm o semidreaptă cu originea în M și numărăm intersecțiile acesteia cu poligonul (fig. 14). Dacă numărul de intersecții este impar atunci M este punct interior, iar dacă numărul intersecțiilor este par, atunci punctul M este unul exterior poligonului. Totuși, apar complicații atunci când semidreapta are mai mult de un punct comun cu cel puțin una din laturile poligonului.

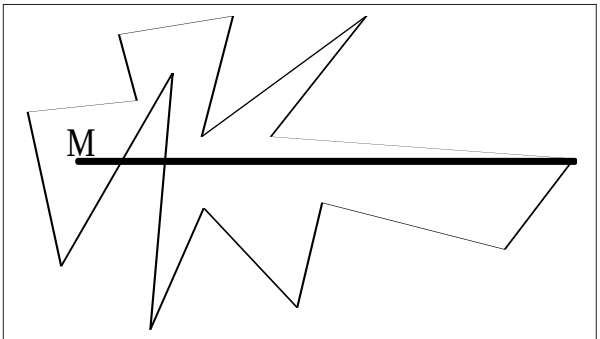

Fig. 14.

Următoarea descriere pseudo-Pascal a funcției `pint(pol,M)` utilizează o semidreaptă de test orizontală, presupune că ***pol***(1) este punctul cu cea mai mică abscisă și cea mai mică ordonată pentru a ne asigura că dacă ***pol***(1) este pe semidreapta de test, ***pol***(0) nu poate fi pe această semidreaptă.

```
function pint(pol:poligon;M:point):Boolean;
var contor,index_curent,index_ancora:Integer;
    semidreapta,latura:line;
begin
  contor:=0; index_ancora:=0;
  pol(0):=pol(n);
  pol(n+1):=pol(1);
  semidreapta.init:=M;
  semidreapta.final:=M;
  semidreapta.final.abscisa:=maxint;
  for index_curent:=1 to n do
  begin
    latura.init:=pol(i);
    latura.final:=pol(i);
    if not f(latura,semidreapta)
    then
      begin
        latura.final:=pol(index_ancora);
        index_ancora:=index_curent;
        if    f(latura,semidreapta)    then
        Inc(contor)
      end
  end;
  pint:=(contor mod 2)=1
end;
```

În descrierea precedentă s-au utilizat următoarele tipuri de date:

```
type point=record
              x,y:tip_coordonate
            end;
     linie=record
              int,final:point;
     poligon=array(0..maxim) of point,
```

iar funcția `f` corespunde unei implementări a problemei intersecției a două segmente (vezi secțiunea 2.2). Se poate observa ușor că algoritmul are complexitate liniară.

Bibliografie:

I.C. Braid — *Designing with volumes*. Cambridge, 1974, 81-85.

F. Feito, J,C, Torres, L.A. Ureña — *Orientation, Simplicity and Inclusion Test for Planar Polygons*, Computer & Graphics, 19(4), 1995

M. Garcia, R. Segura, L.A. Ureña — *Un nuevo algorithmo de Inclusion de puntos en Triangulos*. Departamento de Informatica, Universidad de Jaen, 1997

W.K. Giloi — *Interactive Computer Graphics: Data Structures, Algorithms, Languages,* Prentice-Hall, 1978

E. Haines — *Point in Polygon Strategies*, Graphic GEMs, Academic Press, 1994

S.R. Levine — *Interactive 3-D Motion Graphics with Large Data Bases*, SLAC Report 192, Mass, 1976, 64-65

Y.D. Liang, B. Barsky — *An Analysis and Algorithm for Polygon Clipping*, CACM 26, 1983, 868-877

Y.D. Liang, B. Barsky — *A new Concept and Method for Line clipping*, ACM TOG, 3(1), 1984, 1-22

F.P. Preparata, M. I. Shamos — *Computational Geometry: An introduction*, Springer-Verlag, 1985

I.E. Sutherland, G.W. Hodgman — *Reentrant Polygon Clipping,* CACM, 17, 1974, 32-42

K. Weiler, P. Atherton — *Hidden Surface Removal Using Polygon Area Sorting*, Computer Graphics, 11(2), 1977, 214-222

Dl. Grigore Albeanu este profesor la Catedra de Informatică din cadrul Facultății de Matematică a Universității București.

E-mail: albeanu@math.math.unibuc.ro