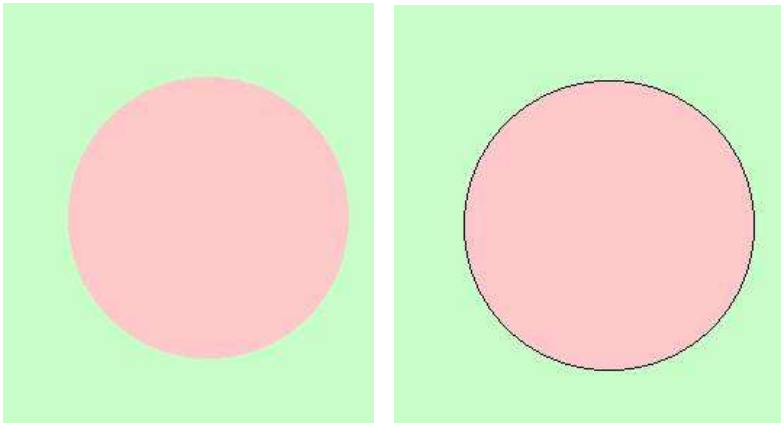
Tracé de contours et de courbes (en deux dimensions)

Ce problème se pose dans de nombreux contextes, mais il s'agit toujours de trouver la frontière entre deux zones. Cela peut être la frontière entre deux zones coloriées différemment. Il peut aussi s'agir de lignes de niveau, ou d'équipotentielles. Enfin c'est un moyen de construire des courbes dont on connaît l'équation. Dans le cas le plus général, avec un repère *Oxy*, cette équation est de la forme $f(x, y) = 0$, c'est-à-dire une relation entre x et y : si l'on se donne une valeur de x, il lui correspond une ou plusieurs valeurs de y, ou aucune. Les solutions de l'équation (à deux inconnues x et y) sont des points (x, y) qui forment une courbe dans un repère. Par exemple l'équation $x^2 + y^2 - 1 = 0$ est celle d'un cercle de centre l'origine et de rayon unité. Une telle courbe est aussi une frontière entre deux zones, celle où l'on a $f(x, y) > 0$ et celle où $f(x, y) < 0$.

Dans tous ces cas, l'objectif est de tracer une ligne de démarcation. Pour cela on va faire de la couture, ou des points de suture, comme si l'on enroulait un fil autour de la courbe que l'on va ainsi construire.

Cas d'une surface fermée

C'est le cas le plus simple : on a une surface fermée coloriée d'une certaine couleur, l'extérieur étant colorié d'une autre couleur.



Sur le dessin ci-contre (*à gauche*), on a un cercle dont l'intérieur est en rose, et l'extérieur en vert. L'objectif est de tracer le contour du disque. On le voit en noir sur le dessin de droite. Comment s'y prend-on ?

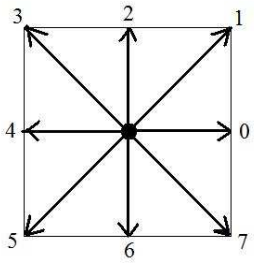
On commence par choisir un point sur la bordure de gauche de l'écran, soit (*xbord*, *ybord*), puis on avance horizontalement tant qu'on reste dans la zone extérieure. On s'arrête dès qu'on tombe sur un point intérieur (un point rose dans l'exemple). On a déjà ce morceau de programme :

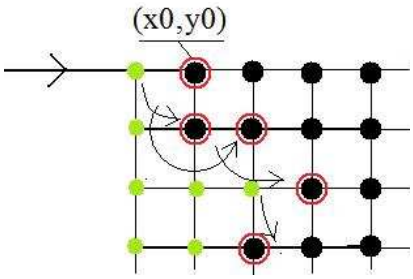
```
xbord=0; ybord=300;   x=xbord;y=ybord;
while(exterieur(x,y)==OUI) x++;
```

Avec la fonction *exterieur*() :

```
int exterieur(int a, int b)    /* point de coordonnées (a, b)  */
{   if (getpixel(a,b)==couleurexterieur) return OUI;
    return NON;
}
```

Le point où l'on est arrivé est noté (*x0*, *y0*). C'est le premier point du contour.





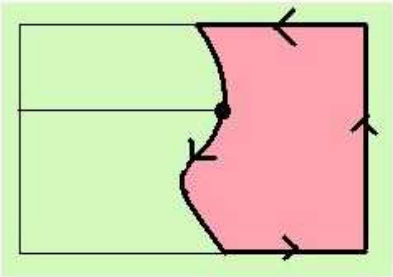
Pour ce point, comme pour ceux qui vont suivre, on définit les huit voisins du point, avec les directions notées de 0 à 7. Le point extérieur qui précède le point intérieur (*x0*, *y0*) est dans la direction 4. On note *interdit* = 4, et l'on va tourner sur les voisins du point dans le sens inverse des aiguilles d'une montre à partir de la direction *interdit* + 1, ramenée modulo 8, jusqu'à ce que l'on retombe sur un point intérieur, qui est à son tour sur le contour. On est alors dans une certaine direction *j*, et pour le nouveau point obtenu, le point intérieur précédent est dans la direction *interdit* = *j* + 4 ramenée modulo 8. On recommence à tourner à partir de la direction *interdit* + 1, jusqu'à retomber sur un point intérieur. Et l'on continue ainsi à passer d'un point intérieur au suivant en tournant sur les voisins du point, jusqu'à ce que l'on revienne au point de départ (*x*0, *y*0).

D'où la suite et fin du programme :

```
interdit=4; x0=x;y0=y;     /*  (x,y) est le point courant sur le contour */
do
{  putpixel(x,y,noir);
    vx[1]=(vx[0]=(vx[7]=x+1));vx[3]=(vx[4]=(vx[5]=x-1));vx[2]=(vx[6]=x);
    vy[3]=(vy[2]=(vy[1]=y-1));vy[5]=(vy[6]=(vy[7]=y+1));vy[4]=(vy[0]=y);
    j=(interdit+1)%8;
   while(exterieur(vx[j%8],vy[j%8])==OUI) j++;
   x=vx[j%8];y=vy[j%8];interdit=(j+4)%8;
}
while(x!=x0 || y!=y0);
```

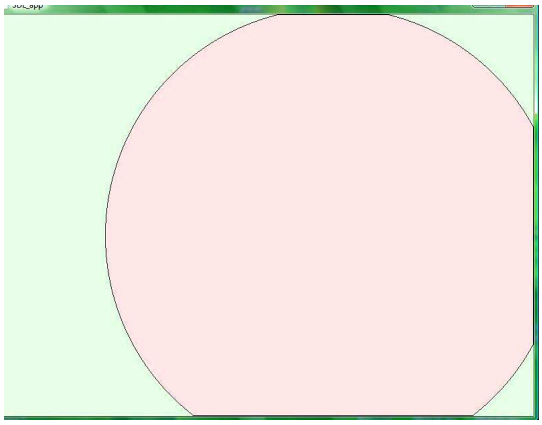
Le problème de la bordure

Mais il peut arriver que la surface empiète sur la bordure. Dans ce cas le programme précédent ne marche plus car jamais on ne reviendra au point de départ. On va alors considérer comme extérieurs non seulement les points extérieurs à la surface, mais aussi les points extérieurs au rectangle de l'écran, ce qui fait qu'on va longer la bordure de l'écran jusqu'à ce qu'on retombe sur le contour véritable, pour se retrouver finalement au point de départ. Il suffit d'ajouter une ligne à la fonction *exterieur*().



```
int exterieur(int a, int b)     /* il s'agit du point de coordonnées (a, b) */
{  if ( a<0 || a>799 || b<0|| b>599) return OUI;     /* écran de 800 sur 600 */
   if (getpixel(a,b)==couleurexterieur) return OUI;
   return NON;
}
```

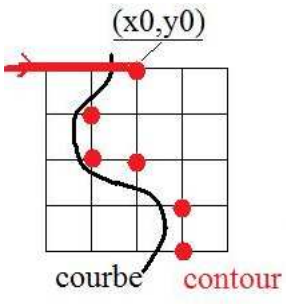
Dans ces conditions le programme précédent trace le contour de la surface morceau par morceau, tout en longeant la bordure de l'écran en cas de besoin pour assurer la continutié.



tracé en noir du contour, tout en longeant la bordure de l'écran

Cas général d'ne courbe

Prenons maintenant une courbe d'équation $f(x, y) = 0$. Il n'y a plus d'intérieur ni d'extérieur, mais des points d'un côté de la courbe avec $f(x, y) > 0$ et des points de l'autre côté avec $f(x, y) < 0$. Cela va demander quelques aménagements par rapport au programme précédent.



D'abord la courbe a une équation avec x et y qui sont éventuellement des nombres à virgule. Mais que ses points soient à coordonnées entières ou pas est sans importance. On va déterminer un contour rasant.

Comme auparavant on part d'un point de la bordure de l'écran, par exemple celle verticale de gauche, soit le point

(*xbord*, *ybord*). On commence par déterminer le signe de *f*(*xbord*, *ybord*), que l'on appelle *signebord*. Puis on avance horizontalement tant que les points obtenus ont le même signe que celui du bord. On s'arrête au premier changement de signe, ce qui donne le premier point (*x0*, *y0*) du contour. A partir de lui, on fait la tournée des voisins tant qu'on tombe sur le même signe que le bord. On s'arrête au premier voisin qui a l'autre signe. C'est aussi un point du contour. Et l'on continue ainsi jusqu'à revenir au point de départ (*x0*, *y0*), quitte à longer la bordure de l'écran en certains endroits.

Le programme principal est le suivant :

```
xbord=0; ybord=200;   x=xbord;y=ybord;
if (signef(x,y)==PLUS)  signebord=PLUS; else signebord=MOINS;
while(memesignequebord(++x,y,signebord)==OUI)
     { putpixel(x,y,rouge); if (x>799)break;}
interdit=4; x0=x;y0=y;
do
{  putpixel(x,y,c[icouleur]);
   vx[1]=(vx[0]=(vx[7]=x+1));vx[3]=(vx[4]=(vx[5]=x-1));vx[2]=(vx[6]=x);
   vy[3]=(vy[2]=(vy[1]=y-1));vy[5]=(vy[6]=(vy[7]=y+1));vy[4]=(vy[0]=y);
   j=(interdit+1)%8;
  while(memesignequebord(vx[j%8],vy[j%8],signebord)==1) j++;
  x=vx[j%8];y=vy[j%8];interdit=(j+4)%8;
}
while(x!=x0 || y!=y0);
```

avec les fonctions :

- *int signef*(*int a*,*int b*) qui selon la fonction *f* ramène le signe de *f*(*a*, *b*)

- ```
  int memesignequebord( int a, int b,int signebord)
    { if ( a>799 || a<0 ||b>599 || b<0)  return OUI;
      if (signef(a,b)==signebord ) return OUI;
      return NON;
    }
  ```

Exemples

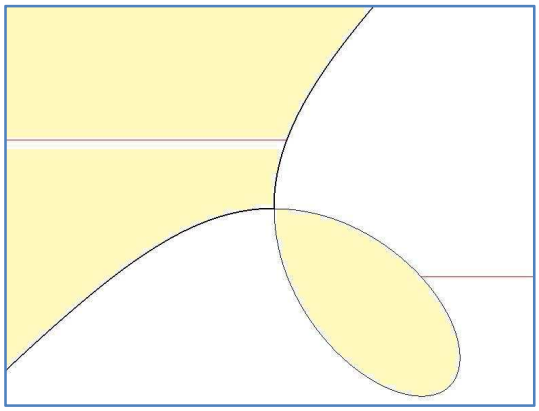
Tout n'est pas encore parfait. Il faudra éventuellement de changer le point de départ sur la bordure verticale gauche de l'écran, au cas où la ligne horizontale lancée à partir de lui ne touche par la courbe. Parfois on devra changer de bordure. Dans certains cas, il conviendra de prendre plusieurs points de départ, lorsque la courbe est complexe, et qu'elle présente des boucles, si on veut l'obtenir en totalité. Dans les exemples qui suivent, on a pris un point de départ à gauche puis un point de départ sur la bordure verticale droite.

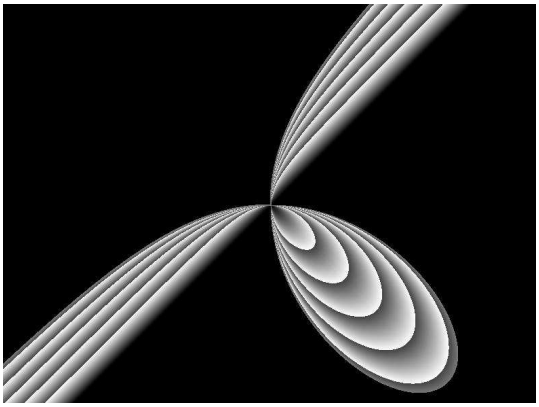
1) Folium de Descartes

Son équation est $x^3 + y^3 - a\,x\,y = 0$. Pour passer à l'échelle de l'écran, on prend $f(x, y) = (x - 400)^3 + (y - 300)^3 - A(x - 400)(y - 300)$; avec $A = 500$ par exemple. D'où la fonction *signef*() qui ramène 1 (PLUS) ou -1 (MOINS) :
  ```

```
int signef(int a,int b)
{ if (  (a-400)*(a-400)*(a-400)+(b-300)*(b-300)*(b-300) - A*(a-400)*(b-300) >0) return 1;
  else return -1;
}
```

Comme la courbe présente une boucle, on doit prendre deux points de départ sur la bordure de l'écran. On trouvera ci-dessous la courbe pour A = 520 et à droite une série de courbes avec A variant de 0 à 525.

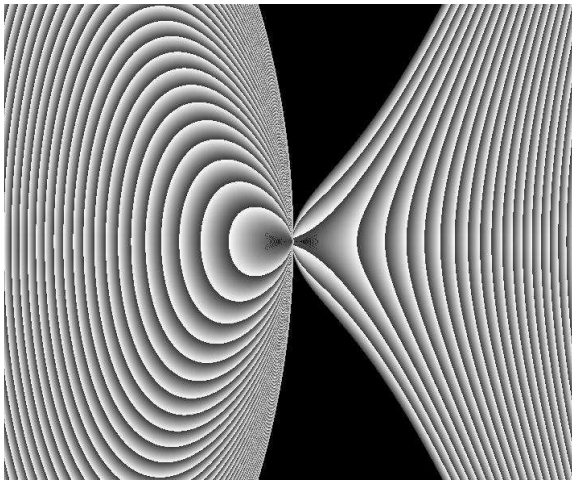


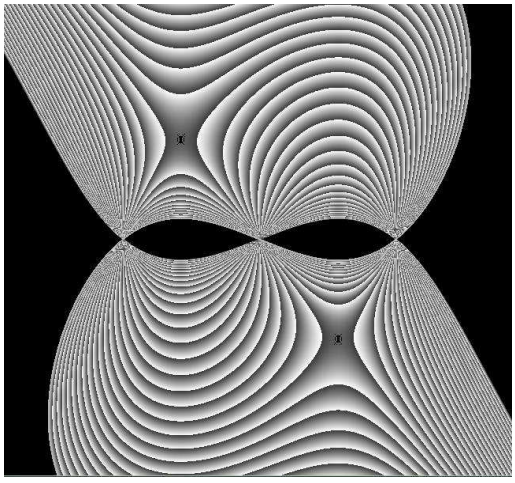


2) Courbes elliptiques

On a pris les équations :

- $y^2 - x^3 + ax - b = 0$
- $2x(x^2 - b) - y(y^2 - a) = 0$.





3) Hyperboles

On se donne une droite D et un point F, ainsi qu'un nombre e supérieur à 1. On sait que les points M tels que MF / MH = e (avec H projection de M sur D) décrivent une hyperbole à deux branches, de foyer $F(xf, yf)$ et de directrice D. On considère que la droite D passe par un point $A(xa, ya)$ et a pour pente m. Son équation est $y - ya = m(x - xa)$. A partir du point M (x, y), on a $MF^2 = (x - xf)^2 + (y - yf)^2$ et $MH^2 = (m(x - xa) - y +$

ya)^2 /(1 + m^2). L'équation de l'hyperbole s'écrit $MF^2 - e^2\ MH^2 = 0$. En faisant varier m de 0,1 à 2,5, on obtient une succession d'hyperboles donnant le dessin suivant.