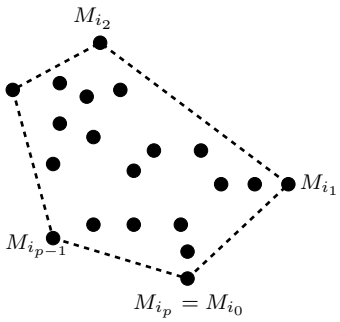
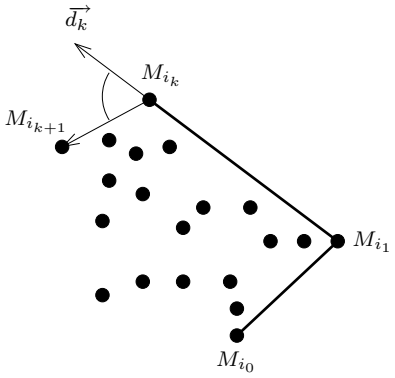
Enveloppe convexe et cercle entourant un nuage de points

1. Algorithme de Jarvis

Soient $M_0, \ldots, M_{n-1}$, des points du plan. L'enveloppe convexe de ces points, $\mathcal{K} = \text{conv}(M_0, \ldots, M_{n-1})$ est la plus petite partie du plan pour l'ordre d'inclusion qui soit convexe et qui contienne tous les points. On se convainc aisément que $\mathcal{K}$ est un polygone dont la frontière est une ligne brisée fermée, $\mathcal{L} = (M_{i_0}, M_{i_1}, \ldots, M_{i_{p-1}}, M_{i_p})$, pour une certaine suite $(i_0, \ldots, i_p)$ dans $[\![0, n-1]\!]$ avec $i_p = i_0$.





Il existe de nombreux algorithmes permettant de déterminer $\mathcal{L}$, l'algorithme suivant est dû à Jarvis et est appelé « algorithme du paquet cadeau » (on entoure les points avec du papier pour faire un emballage cadeau).

0. On suppose les points M_i distincts.

1. Déterminer un point M_{i_0} d'ordonnée minimale, et d'abscisse minimale parmi les points ayant cette ordonnée. M_{i_0} est le point de départ de $\mathcal{L}$. On note $\overrightarrow{d_0}$ la direction horizontale orientée dans le sens des abscisses croissantes.

2. Les points $M_{i_0}, \ldots, M_{i_k}$ et la direction $\overrightarrow{d_k}$ étant connus, déterminer un point $M_{i_{k+1}}$ parmi les points restant et M_{i_0} tel que l'angle $(\overrightarrow{d_k}, \overrightarrow{M_{i_k} M_{i_{k+1}}})$ (mesuré entre 0 et π) soit minimal. Poser $\overrightarrow{d_{k+1}} = \text{direction}(\overrightarrow{M_{i_k} M_{i_{k+1}}})$.

3. Répéter l'étape 2 tant que $M_{i_{k+1}} \neq M_{i_0}$.

Cet algorithme soulève un problème technique : comment déterminer de façon exacte le point $M_{i_{k+1}}$ sans faire de calcul d'angle approché ? En supposant que les points M_i sont définis par des coordonnées exactes, on remarque que l'angle $(\overrightarrow{d_k}, \overrightarrow{M_{i_k} M_a})$ est inférieur ou égal à l'angle $(\overrightarrow{d_k}, \overrightarrow{M_{i_k} M_b})$ si et seulement si le déterminant $\det(\overrightarrow{M_{i_k} M_a}, \overrightarrow{M_{i_k} M_b})$ est positif ou nul (s'en convaincre). Cette condition peut être testée de manière exacte, et a l'avantage de ne pas faire intervenir les directions $\overrightarrow{d_k}$ qu'il est ainsi inutile de calculer.

2. Programmation en Caml

On utilisera les nombres rationnels fournis par la bibliothèque `num` de façon à pouvoir effectuer des calculs exacts sans limitation de capacité. Un nuage de points sera représenté par un vecteur `p` de type `(num * num) vect`, avec la convention : `p.(i)` $= (x_i, y_i)$ où x_i et y_i sont les coordonnées du point M_i. Le code suivant permet de tirer un tableau de points distincts au hasard dans le rectangle $[x_{\min}, x_{\max}[\times [y_{\min}, y_{\max}[$:

```
#open "num";;
let zero = num_of_int(0);;

(* compare deux points *)
let égal (a,b) (c,d) = (a =/ c) && (b =/ d);;

(* tire un tableau de points distincts au hasard *)
let random_points n xmin ymin xmax ymax =
  let p = make_vect n (zero,zero) in
  let i = ref(0)                  in
  while !i < n do
    let a = num_of_int(xmin + random__int(xmax-xmin)) in
    let b = num_of_int(ymin + random__int(ymax-ymin)) in
```

```
    let m = (a,b)  in
    let j = ref(0) in
    while (!j < !i) && not(égal m p.(!j)) do j := !j + 1 done;
    if !j = !i then begin p.(!i) <- m; i := !i + 1 end
  done;
  p
;;
```

Entrer le code ci-dessus et écrire les fonctions suivantes :

– `det` : $(a, b, c) \longmapsto \det(\overrightarrow{ab}, \overrightarrow{ac})$ où a, b, c sont des points (couples de rationnels).

– `ordonnée_min` : prend un tableau p de points et les permute dans p de sorte qu'en sortie le point $p.(0)$ soit le point d'ordonnée minimale défini à l'étape 1.

– `angle_min` : prend un tableau de points p de longueur n et un indice k en paramètres et permute les points $p.(k+1), \ldots, p.(n-1)$ de sorte qu'en sortie le point $p.(k+1)$ soit celui défini à l'étape 2.

– `convexe` : prend un tableau de points p en paramètre et retourne un nouveau tableau q contenant les sommets de l'enveloppe convexe déterminés par l'algorithme de Jarvis (le premier point de q figure aussi en dernière position). On effectuera en premier lieu une copie de p de façon à ne pas modifier le tableau argument.

Visualisation : vous pouvez afficher le nuage de points p et le contour de son enveloppe convexe q avec le code suivant :

```
#open "graphics";;
open_graph "";;
(* arrondi entier *)
let round(x) = int_of_num(round_num x);;

(* affiche des points *)
let plot_points p =
  set_color black;;
  for i=0 to vect_length(p)-1 do
    let (a,b) = p.(i) in
    fill_circle (round a) (round b) 2
  done
;;

(* affiche une ligne brisée *)
let plot_lines p =
  set_color red;;
  let (a,b) = p.(0) in
  moveto (round a) (round b);
  for i=1 to vect_length(p)-1 do
    let (c,d) = p.(i) in
    lineto (round c) (round d)
  done
;;

(* essai *)
let p = random_points 50 150 150 350 350;;
let q = convexe p;;
clear_graph();;
plot_points p;;
plot_lines q;;
```

3. Programmation en Maple

Un point sera représenté par une liste de deux coordonnées, et un nuage de points par un tableau de telles listes. Le code suivant permet de tirer un tableau de points distincts au hasard dans le rectangle $[x_{\min}, x_{\max}] \times [y_{\min}, y_{\max}]$:

```
random_points := proc(n,xmin,ymin,xmax,ymax)
local m,p,i,j,rx,ry;
p := array(1..n);
rx := rand(xmin..xmax);
ry := rand(ymin..ymax);
i := 1;
while i <= n do
  m := [rx(),ry()];
  j := 1;
  while (j < i) and not(egal(m,p[j])) do j := j+1 od;
  if (j = i) then p[i] := m; i := i+1 fi
od;
eval(p)
end;
```

Entrer le code ci-dessus et écrire les fonctions `det`, `ordonnée_min`, `angle_min` et `convexe` décrites dans la section « Programmation en Caml ». On passera en paramètre supplémentaire le nombre n de points du nuage ou on utilisera le code suivant pour déterminer la longueur d'un tableau :

```
longueur := p -> op(2,op([1,2],p));
```

Pour afficher le nuage de points et le contour de son enveloppe convexe on utilisera le code suivant :

```
plot_points  :=  p -> plot(p, style = point, color = black);
plot_polygon :=  p -> plot(p, style = line,  color = red);

p := random_points(50,150,150,350,350);
q := convexe(p,50);
plots[display](plot_points(p),plot_polygon(q));
```

4. Tests d'appartenance

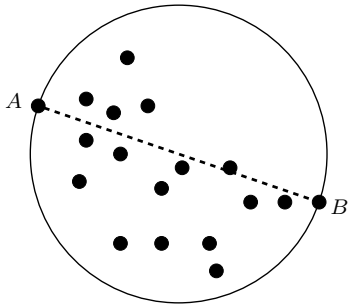
Écrire les fonctions `intérieur` et `frontière` qui prennent en argument un point m et une enveloppe convexe définie par son contour fermé q et déterminent si le point m est intérieur ou sur la frontière de l'enveloppe convexe.

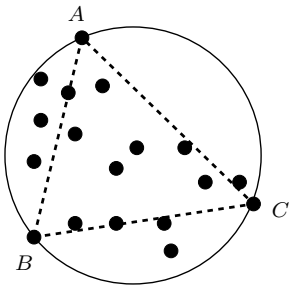
5. Cercle entourant un nuage de points

On démontre avec un argument de compacité et un peu de géométrie qu'il existe un et un seul cercle de rayon minimal entourant les points d'un nuage fini, l'objectif de cette partie est de programmer le calcul de ce cercle $\mathcal{C}$. On utilisera les propriétés suivantes qui caractérisent $\mathcal{C}$:

– *$\mathcal{C}$ entoure les sommets de l'enveloppe convexe du nuage.*

– *il existe deux sommets de l'enveloppe convexe A, B tels que $\mathcal{C}$ est le cercle de diamètre $[A, B]$ ou il existe trois sommets de l'enveloppe convexe non alignés A, B, C tels que le triangle ABC a tous ses angles aigus et $\mathcal{C}$ est le cercle circonscrit à ABC.*





La deuxième propriété implique qu'il y a un nombre fini de cercles à tester, et vu l'unicité de $\mathcal{C}$, le premier cercle vérifiant la première propriété est le bon. Par ailleurs le calcul exact du centre de $\mathcal{C}$ et du *carré* de son rayon est possible, sachant que le centre du cercle circonscrit à un triangle est le point de concours de deux médiatrices.

On programmera les fonctions suivantes :

- `dist2` : donne le carré de la distance entre deux points.
- `milieu` : donne le milieu de deux points.
- `centre` : donne le centre du cercle passant par trois points supposés non alignés.
- `aigu` : dit si trois points sont non alignés et forment trois angles aigus.
- `entoure` : dit si un cercle donné par son centre et le carré de son rayon entoure un nuage de points.
- `cercle` : retourne le cercle de rayon minimal (centre, carré du rayon) entourant l'enveloppe convexe d'un nuage de points. L'argument de `cercle` est le contour de l'enveloppe convexe qui aura été déterminé au préalable.

Visualisation : utiliser le code suivant pour afficher le nuage de points, le contour de son enveloppe convexe et le cercle de rayon minimal entourant le nuage :

En Caml

```
(* affiche un cercle *)
let plot_circle (a,b) r2 =
  set_color blue;;
  draw_circle (round a) (round b) (int_of_float(sqrt(float_of_num r2)))
;;

(* essai *)
let p = random_points 50 150 150 350 350 in
let q = convexe p in
let (c,r2) = cercle(q) in
clear_graph();
plot_points p;
plot_lines q;
plot_circle c r2
;;
```

En Maple :

```
plot_circle  := (c,r2) -> plottools[circle](c, sqrt(r2), color = blue);

p := random_points(50,150,150,350,350);
q := convexe(p,50);
n := op(2,op([1,2],q));
c,r2 := op(cercle(q,n));
plots[display](plot_points(p),plot_polygon(q),plot_circle(c,r2));
```