Q1

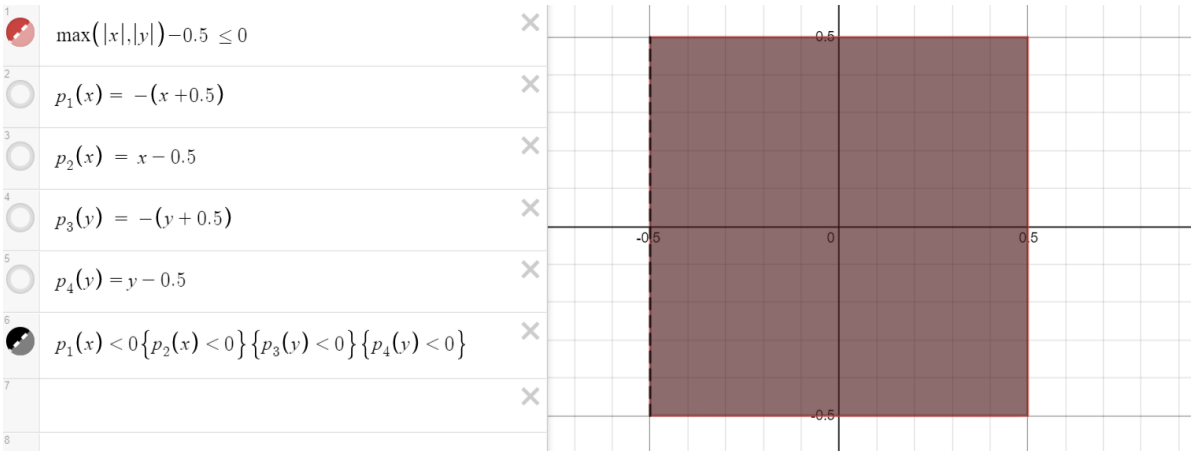
Oui nous pouvons construire un polygone convexe à l'aide de fonctions implicites de demi plans. Il suffit de prendre l'intersection de ces demi plans.

exemple d'un carré de coté centré en $(0, 0)$:

$$f(x, y) = max(|x|, |y|) - 0.5$$

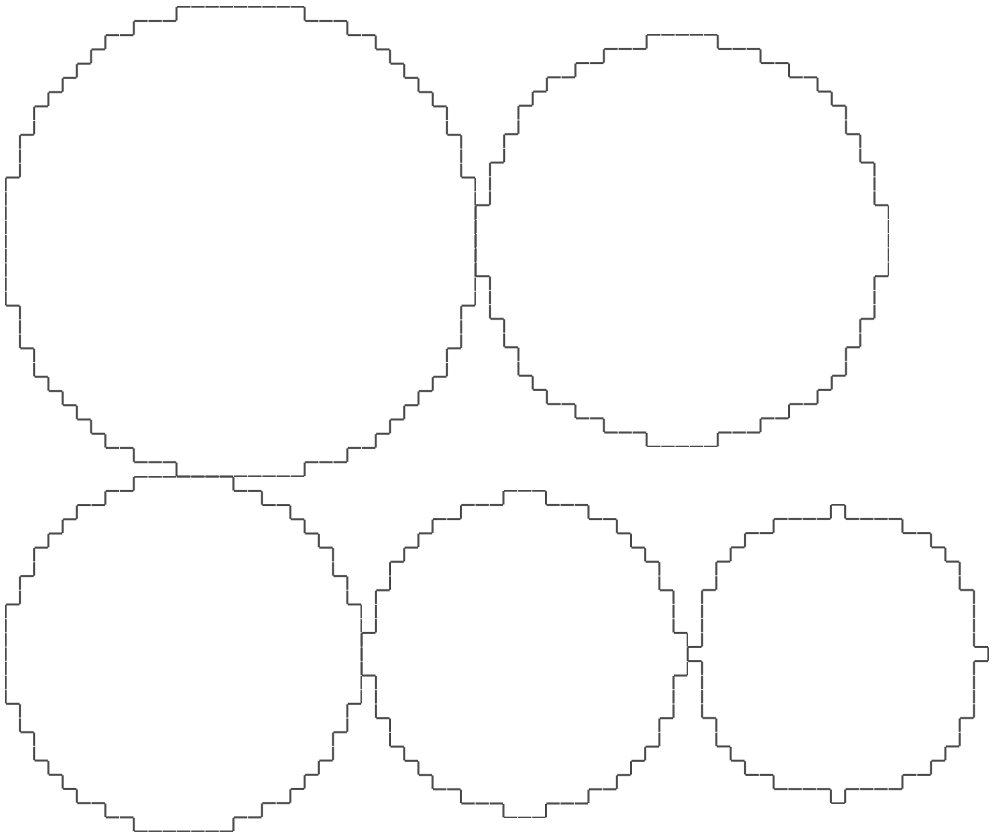
mais cela peux aussi s'écrire comme une intersection de quatre demi plans:
$p_1(x, y) = -(x + 0.5)$
$p_2(x, y) = x - 0.5$
$p_3(x, y) = -(y + 0.5)$
$p_4(x, y) = y - 0.5$

$p_1 < 0$ et $p_2 < 0$ et $p_3 < 0$ et $p_4 < 0$



Q2

Voici un l' inter-pixel boundaries d'un cercle avec différentes résolutions de digitalisation. de $h = 0.6$ à $h = 1.0$



Les questions suivant on été réalisées en prenant en considération un cercle de rayon 10.

real area : $2\pi.10 = 62.83\ m$

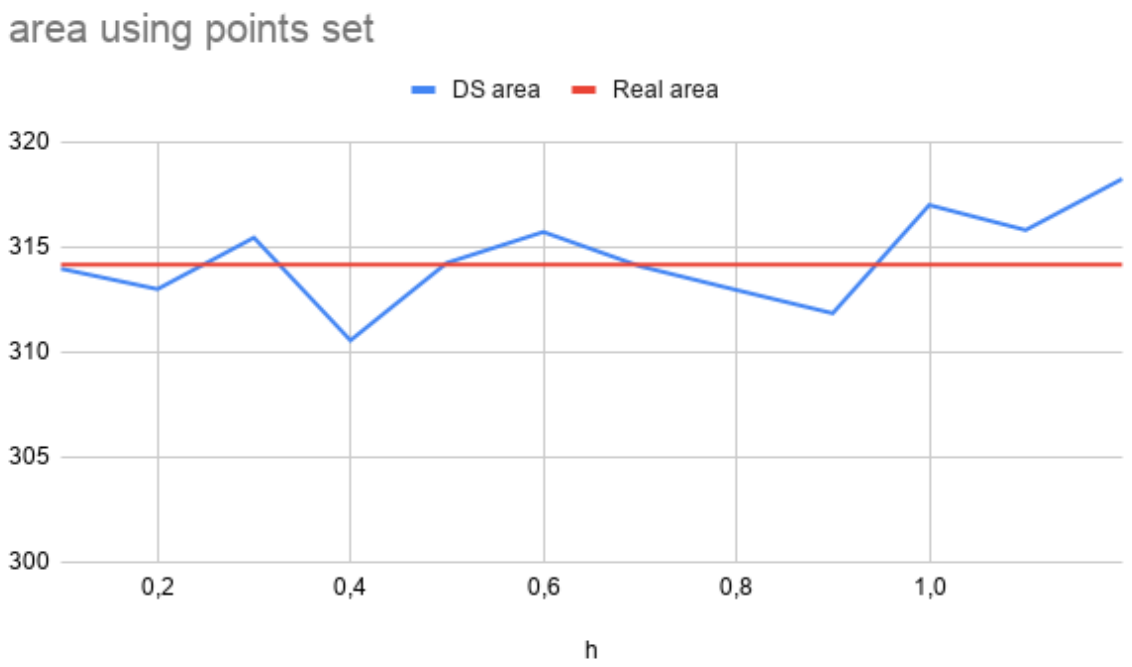
real perimeter: $\pi.10^2 = 314.1595\ m^2$

Q3

Voilà ce dessous les deux graphiques montrant la mesure de respective de l'air et du périmètre en faisant varier la résolution de discrétisation.

On peut constater ici qu'ici la diminution de la résolution de discrétisation h permet améliorer et réduire l'erreur de approximation de l'aire réelle de la forme (ici un cercle de rayon 10).

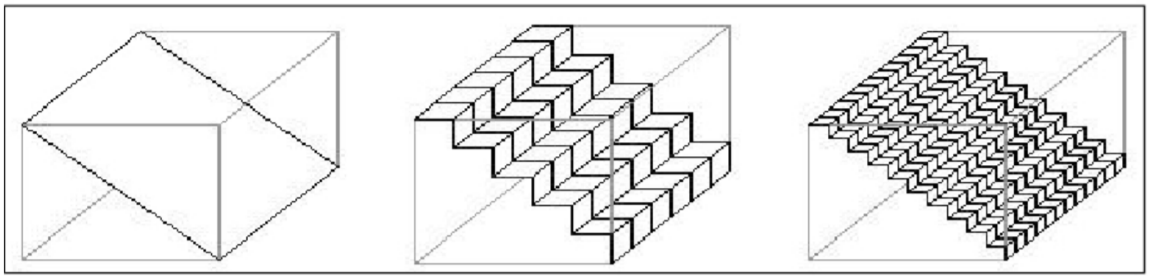
A contrario, le calcul du périmètre à l'aide du nombre de points extérieur reste mauvais. La méthode ne semble pas converger malgré l'amélioration de la résolution h.



C'est donc un bon moyen d'approximer l'aire de la forme mais très mauvais lorsqu'il s'agit approximer le périmètre.

Cela s'explique facilement à l'aide de ce diagramme vu en cours.

On remarque bien ici que l'augmentation de la résolution ne change pas la valeur du périmètre calculé.



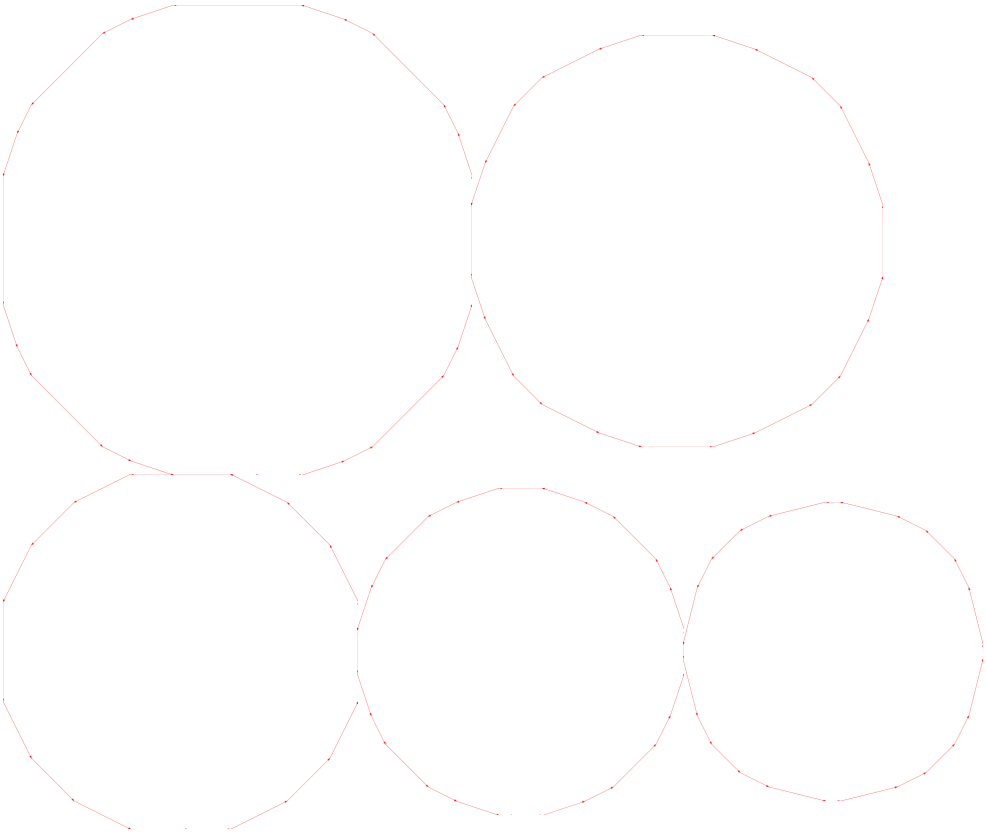
> remarque: nous avons besoin de normaliser par la constante de discrétisation h dans le calcul du périmètre er de l'air.
> Car une 1-cellule mesure h de longueur et qu'une 2-cellule à une aire de h^2.

```
float perimeterDS = static_cast<double>(boundaryPoints.size()-1)*h;
std::cout << "perimeter:" << perimeterDS << std::endl;

float airDS = static_cast<double>(set.size()) * h * h;
```

Q4

Voici un la bordure du converxHull d'un cercle avec différentes résolutions de digitalisation. de $h = 0.6$ à $h = 1.0$



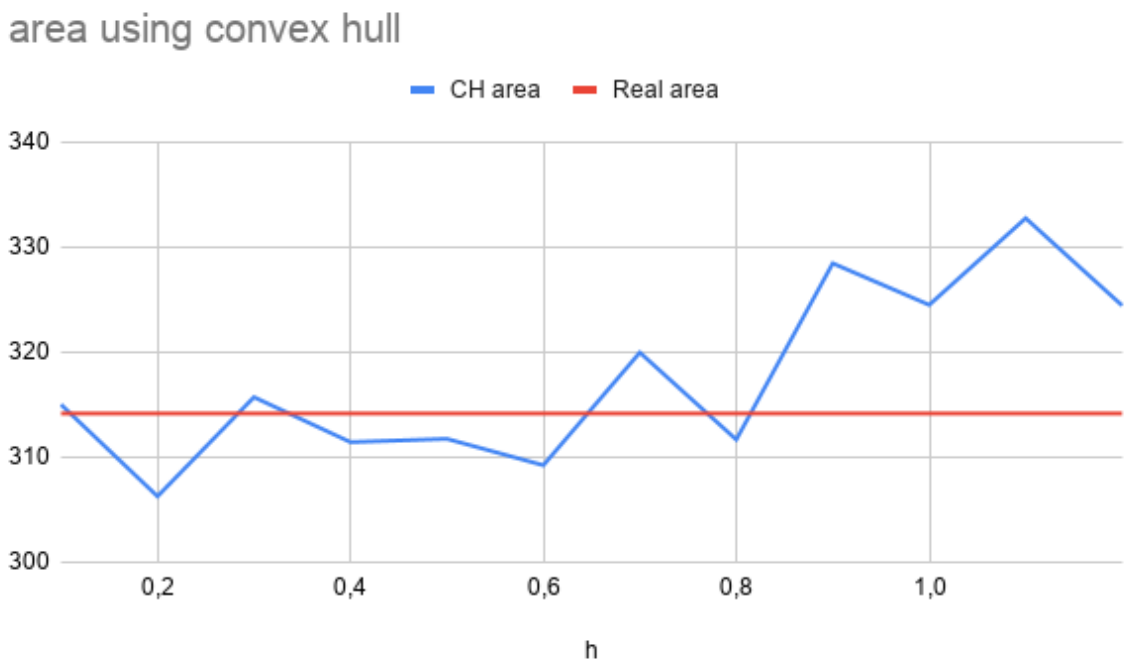
C'est une bonne approximation des formes convexes. Lorsque celle-ci ne le sont pas, elles sont pour ainsi dire comblées et cela fausse le calcul de l'air et du périmètre.

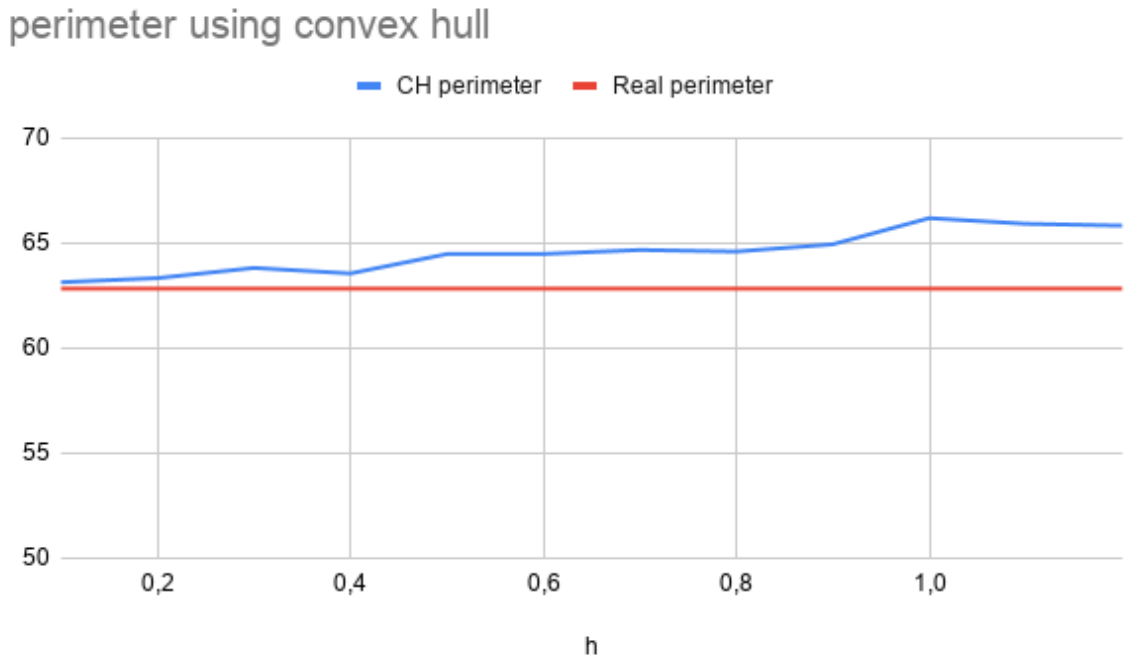
Q5

Voilà ce dessous les deux graphiques montrant la mesure de respective de l'aire et du périmètre en faisant varier la résolution de discrétisation à l'aide cette fois ci du *convex Hull.*

On peut constater ici qu'ici la diminution de la résolution de discrétisation h permet améliorer grandement la précision de approximation du périmètre réel. Cela converge bien vers le périmètre réel.

L'aire cependant reste imprécise et converge mal et de manière chaotique sans que l'on puisse en tirer une réelle tendance.





Cette méthode n'a cependant été testée que sur une forme convexe (un cercle).

Dans ce cadre, cette méthode est très efficace pour le calcul du périmètre. Elle est à éviter dans le cadre de formes non convexes et l'utilisation du set vu à la **Q3** est préférable dans le cadre du calcul de l'aire de la forme.

> Remarques:
>
> De même en utilisant le convex hull il faut normaliser en utilisant la constante de discrétisation h dans les fonctions ci dessous:

```
template <typename T, typename F>
T perimeter(const MelkmanConvexHull<Z2i::Point, F>& cvx, const T h) {

T p = 0.f;
assert(cvx.size() >= 2 && "convex Hull should contain at least two points");

for (size_t i = 0; i < cvx.size()-1; i++) {
p += (cvx[i]-cvx[i+1]).norm();
}
p += (cvx[cvx.size()-1]-cvx[0]).norm();

return p*h;
}
```

```
template <typename T, typename F>
T area(const MelkmanConvexHull<Z2i::Point, F>& cvx, const T h) {

T area = 0.f;
assert(cvx.size() >= 3 && "convex Hull should contain at least three two
points");

for (size_t i = 0; i < cvx.size()-1; ++i) {
area += cvx[i][0]*cvx[i+1][1] - cvx[i][1]*cvx[i+1][0];
}
area += cvx[cvx.size()][0]*cvx[0][1] - cvx[cvx.size()][1]*cvx[0][0];

return abs(area*h*h)/2.0f;
}
```

annexes et données

Données utilisées pour réaliser les diagrammes de ce rapport.

h	0,1	0,2	0,3	0,4	0,5	0,6	0,7	0,8	0,9	1,0	1,1	1,2
DS perimeter	79,5,79	80,1,78	81,5	78,6	80,5	79,2	81,9,83	82,5	80,4			
DS area	313,97,313	315,45	310,56	314,25	315,72	314,09	312,96	311,85,317	315,81	318,24		
CH perimeter	63,1362	63,3334	63,818	63,5561	64,4776	64,4826	64,6723	64,5981	64,9416	66,1871	65,9182	65,8309
CH area	315,01	306,28	315,72	311,44	311,75	309,24	319,97	311,68	328,455	324,5	332,75	324,44