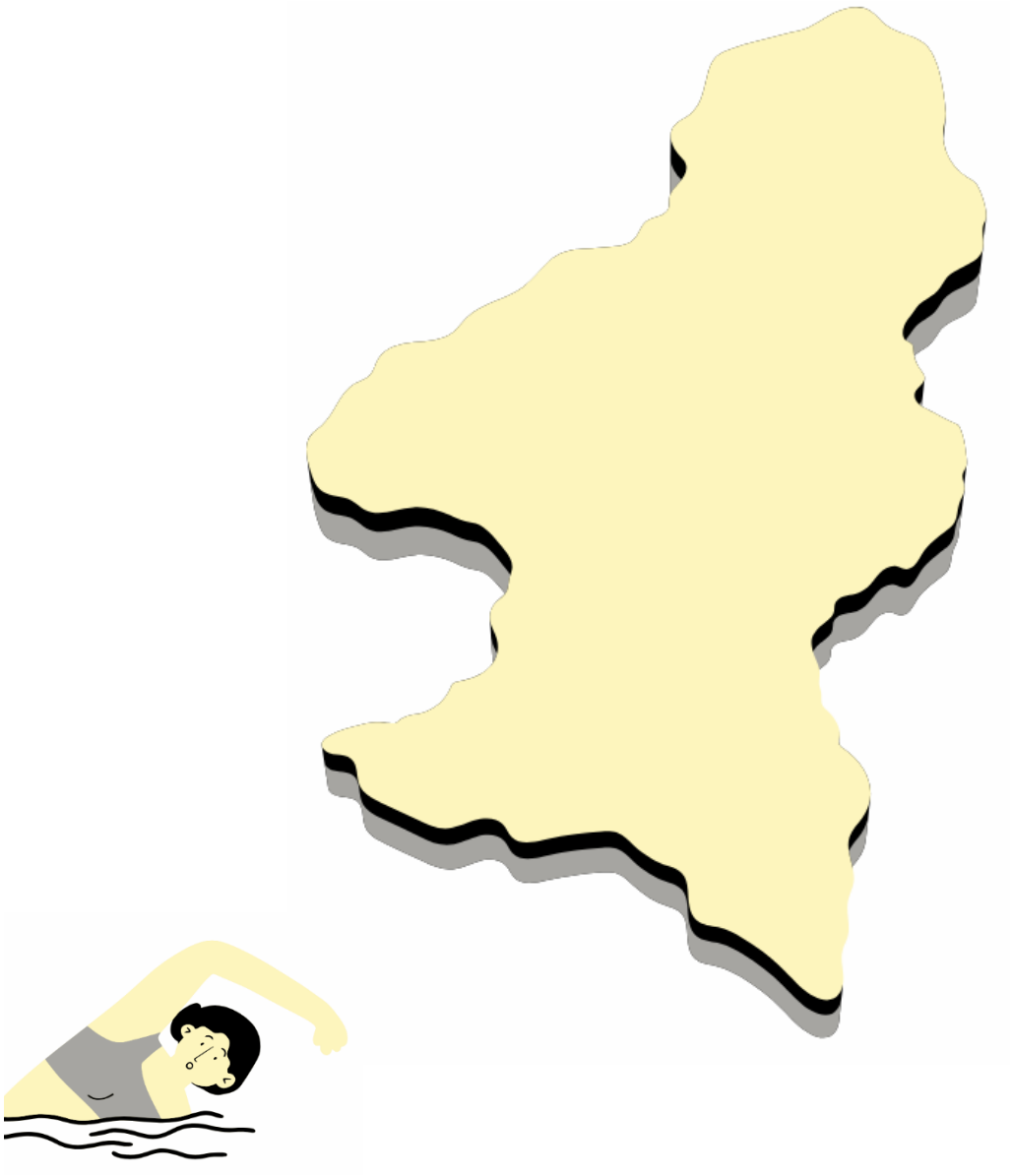
MATh.en.JEANS 2022-2023

Le Tour de l'île à la nage

HEMERY Inès (P08), LEANG Alice (P08), MARTIN Montaine (P05), MICHEL Lélia (T07) et RENARD Dimitri (2ND05)



Établissement : **Lycée Marguerite de Navarre de Bourges**

Enseignants : **BRINAS Olivier, CRECHET Olivier, HERMINIER Nathalie et ROCHE-HERNANDEZ Amélie**.

Chercheurs : **NGUYEN Benjamin, BULTEL Xavier de l'INSA Centre Val de Loire**

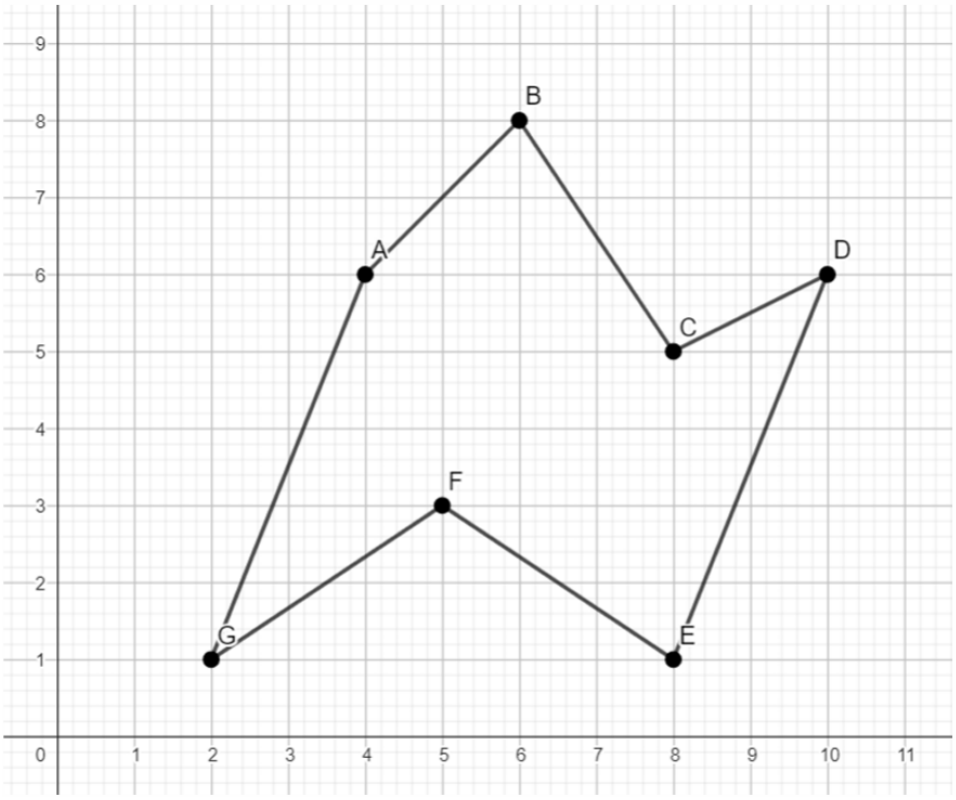
Table des matières

Présentation du sujet

Dans une île de forme polygonale, un nageur doit faire le tour de l'île à la nage, en nageant seulement en ligne droite et le plus rapidement possible. Le sujet consiste donc à déterminer le point de départ et la trajectoire la plus courte possible pour le nageur.

Pour répondre à ce sujet, nous avons exploré plusieurs pistes de recherches que nous développerons dans l'ordre chronologique auxquelles elles nous sont apparues.

Pour commencer nos recherches, nous avons d'abord établi une île type dans un repère orthonormé, sur laquelle nous avons appliqué nos différentes solutions et ainsi visualisé les résultats :



Les sommets de ce polygone (notre île type) ont pour coordonnées :

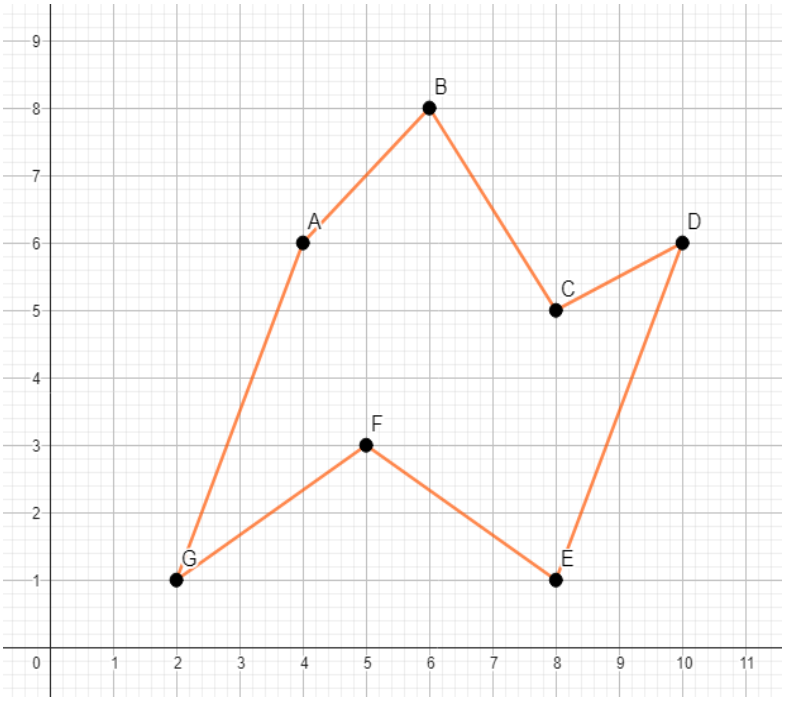
A (4 ; 6)	E (8 ; 1)
B (6 ; 8)	F (5 ; 3)
C (8 ; 5)	G (2 ; 1)
D (10 ; 6)	

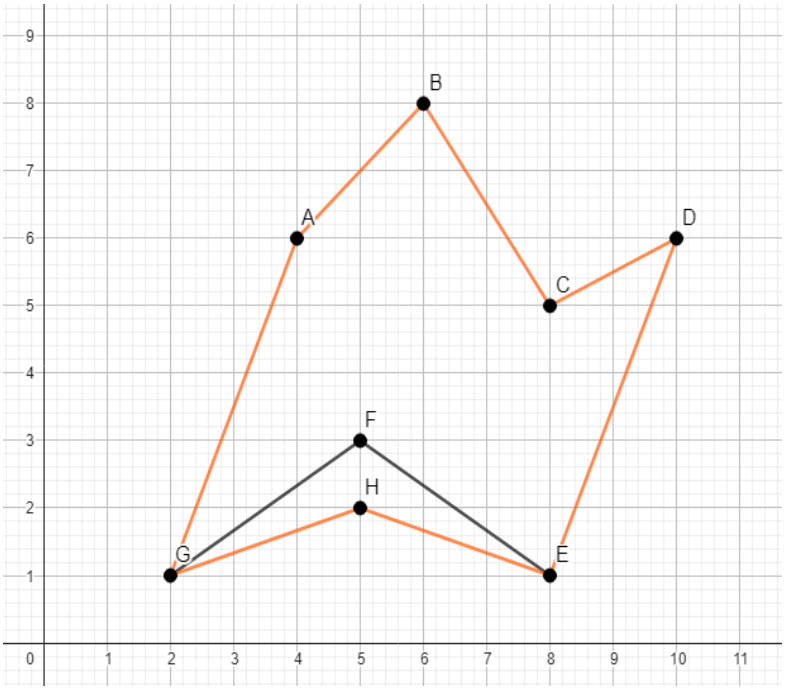
Premiers pas : le périmètre

Nous nous sommes, dans un premier temps, intéressés au périmètre de l'île, l'approche la plus simple pour faire le tour de l'île car il suffit de suivre ses côtés.

La formule pour calculer le périmètre d'une île polygonale correspond à la somme de tous les côtés de la figure.

$P_{polygone}$ = somme de tous ses côtés $\quad P = \sum_{k=1}^{n} C_k \quad$ avec n, nombre entier de côtés





Cependant, cette première approche ne permet pas de déterminer la trajectoire la plus rapide.

En effet, il est facile de trouver une trajectoire plus courte.

On a dans un premier cas, la trajectoire du nageur en orange suivant scrupuleusement le périmètre de notre île, elle correspond à la figure ci-contre.

Ensuite, dans l'image ci-dessous, on a placé un point H(5 ; 2), la trajectoire en orange correspond à une partie du périmètre mais contrairement au premier cas, elle passe par le point H sans passer par le point F .

Pour la première figure :

$GF = \sqrt{(2-5)^2+(1-3)^2} \approx 3{,}61$

$FE = \sqrt{(5-8)^2+(3-1)^2} \approx 3{,}61$

D'où $GF + FE = 3{,}61 + 3{,}61 \approx 7{,}22$.

Pour la deuxième figure :

$GH = \sqrt{(2-5)^2+(1-2)^2} \approx 3{,}16$

$HE = \sqrt{(5-8)^2+(2-1)^2} \approx 3{,}16$

D'où $GH + HE = 3{,}16 + 3{,}16 \approx 6{,}32$.

Ainsi, la trajectoire du deuxième cas est plus courte que la trajectoire formée par le périmètre.

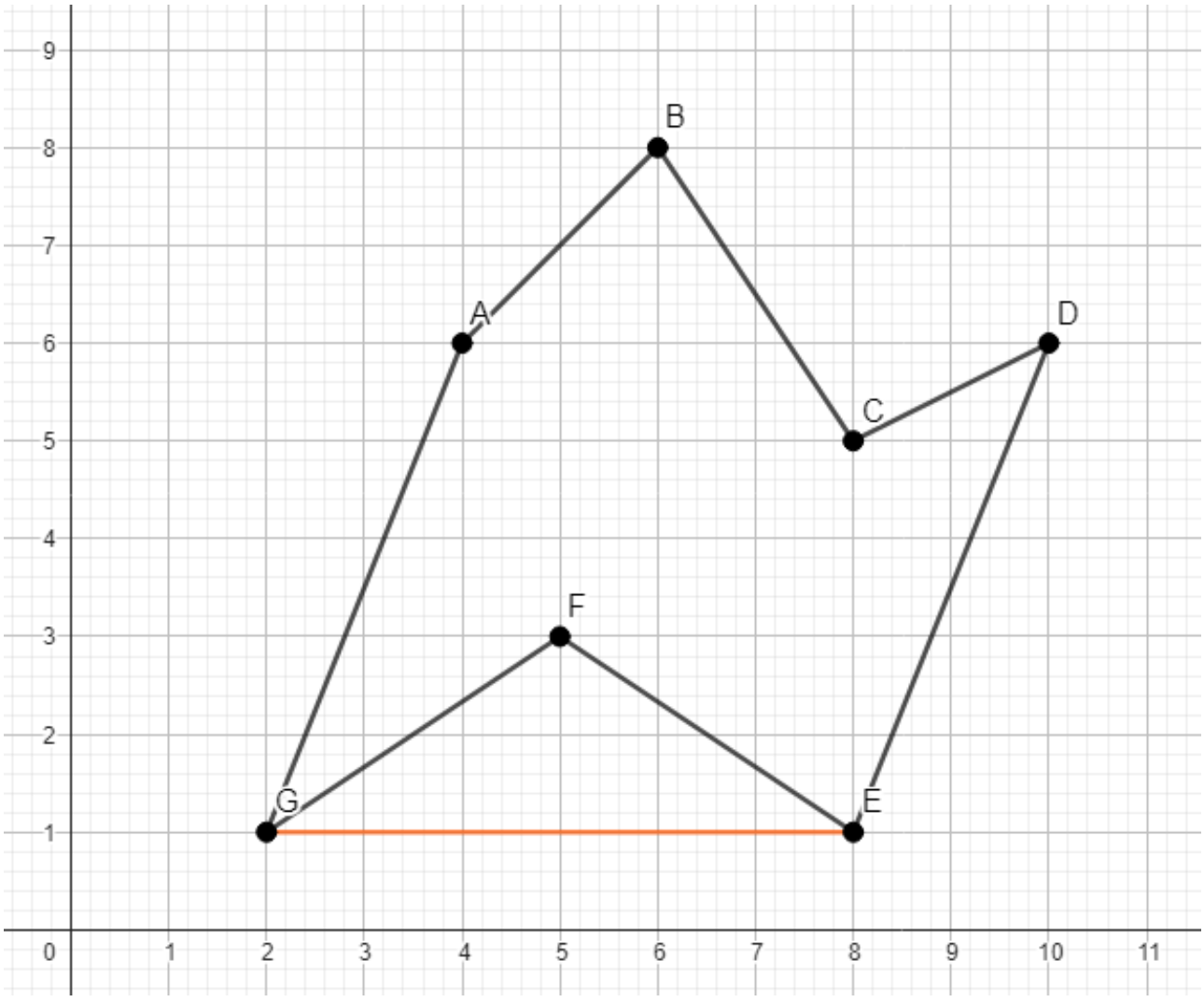
Ce dernier ne nous permettra donc pas de définir la trajectoire du nageur.

Nouveau temps de réflexion

On considère que trois points qui se suivent dans la figure forment un triangle. On sait que la somme des longueurs de deux côtés d'un triangle sera toujours supérieure à la longueur du troisième : c'est l'inégalité triangulaire.

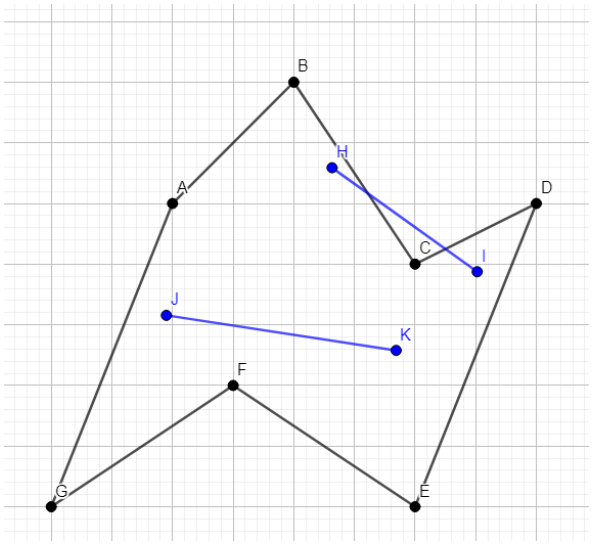
En effet, on voit dans la figure ci-dessous que la troisième longueur GE en orange est visuellement (et mathématiquement) plus courte que la somme des longueurs des segments [GF] et [FE].

Notre nageur devrait donc toujours suivre cette troisième longueur (à condition que la troisième longueur ne coupe pas l'île puisque le nageur ne pourrait donc plus nager). En partant de ce principe, on peut déterminer la trajectoire la plus courte mais on peut aussi remarquer que faire ceci revient à déterminer l'enveloppe convexe de l'île.



Déterminer l'enveloppe convexe

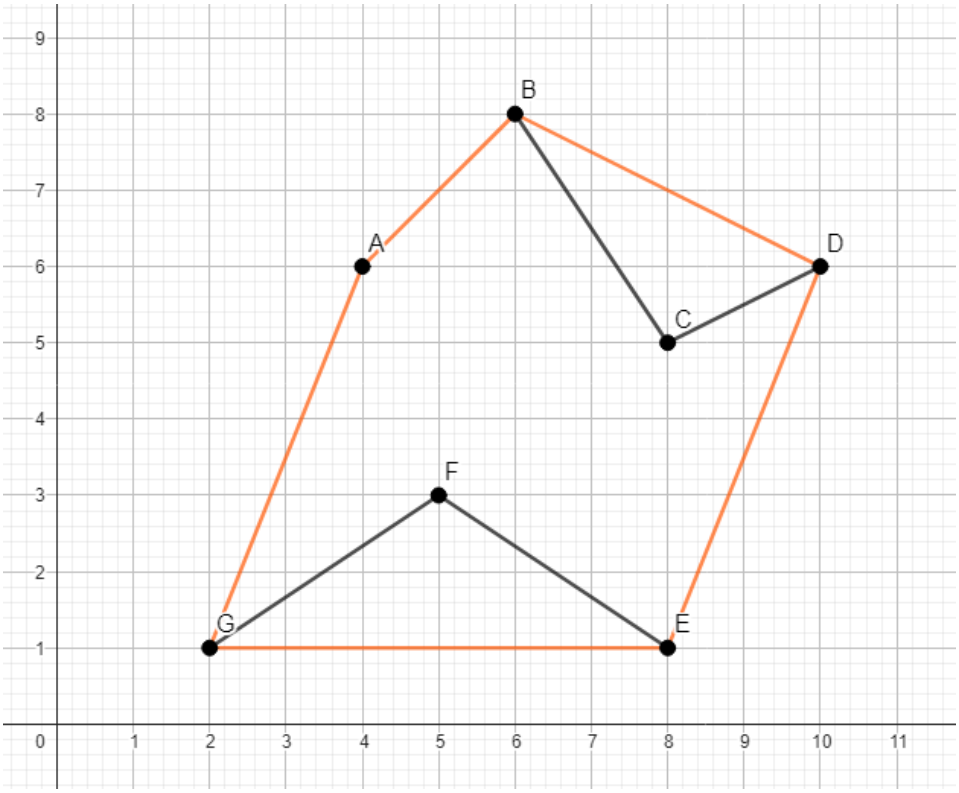
Définition : Un polygone est convexe lorsqu'avec n'importe quel point A et B à l'intérieur, le segment [AB] est entièrement contenu dans le polygone.



Ici, les points H, I, J et K sont à l'intérieur du polygone formé par notre île, le segment [JK] est bien contenu dans l'île, mais pas le segment [HI] donc le polygone n'est pas convexe.

Pour répondre à la problématique, nous avons donc choisi de déterminer l'enveloppe convexe de l'île, c'est-à-dire de dessiner une enveloppe autour de l'île reprenant une partie des mêmes sommets de tel sorte que tous les côtés soient convexes.

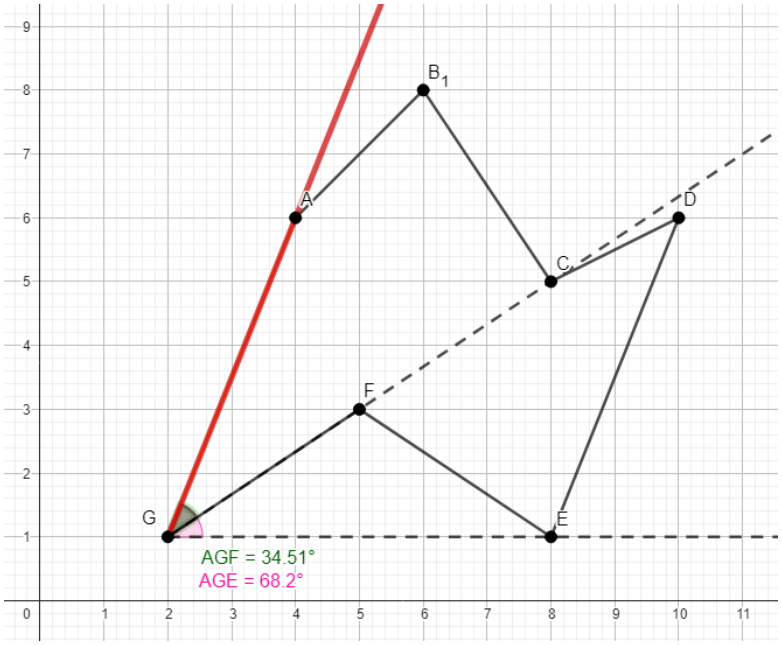
Nous avons commencé par une approche visuelle en modélisant, ici en orange, ce qui nous semblait être une enveloppe convexe :



Il faut ensuite pouvoir déterminer mathématiquement l'enveloppe convexe. Pour ce faire, nous avons établi deux méthodes.

1ère méthode avec les angles

Premier cas de figure : l'île est tracée.



On trace une droite passant par deux sommets du polygone représentant l'île. On mesure ensuite les angles entre la droite et les autres sommets. Le sommet qui forme l'angle le plus grand avec la droite devient un point de l'enveloppe convexe. En répétant la même démarche pour chaque sommet, l'enveloppe convexe se dessinera naturellement et avec elle, la trajectoire du nageur.

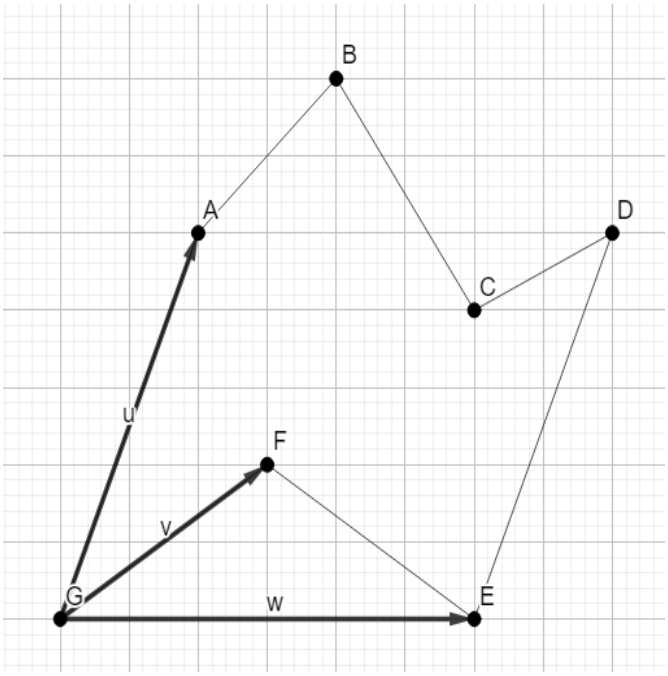
On voit dans la figure ci-contre :

$A\hat{G}E = 68{,}2° > A\hat{G}F = 34.51°$

Donc les segments [AG] et [GE] feront partie de l'enveloppe convexe.

Deuxième cas de figure : seules les coordonnées des points de l'île sont données.

Pour définir l'enveloppe convexe en utilisant toujours les angles mais seulement avec les coordonnées des points du polygone, nous utiliserons le produit scalaire car il permet de calculer des angles à partir de coordonnées de points.



En prenant deux points de l'île, on choisit un vecteur, on détermine ensuite plusieurs vecteurs avec les autres points ayant en commun le point d'origine du vecteur choisi, ici G. On calcule ensuite le produit scalaire de chaque vecteur normé avec le vecteur de départ avec la formule $\vec{u} \cdot \vec{v} =$ xx'+yy' puis on trouve le cosinus de l'angle des deux vecteurs.

$$\vec{u} \cdot \vec{v} = \|\vec{u}\| \cdot \|\vec{v}\| \times cos\,(A\hat{G}E)$$

$$\frac{\vec{u} \cdot \vec{v}}{\|\vec{u}\| \cdot \|\vec{v}\|} = \cos\,(A\hat{G}E)$$

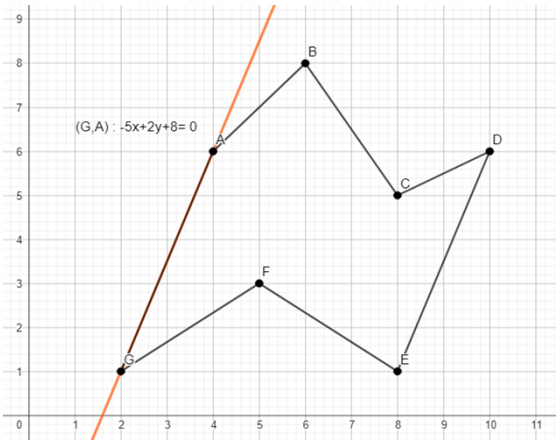
On sait que plus un cosinus est petit, plus la valeur de l'angle sera grande, ainsi les segments formés par les vecteurs ayant le cosinus le plus petit feront partie de l'enveloppe convexe.

Dans le cas ci-dessous, ce sont les vecteur $\vec{u} = \overrightarrow{GA}$ et $\vec{w} = \overrightarrow{GE}$. On répète ensuite la même démarche en changeant à chaque fois le point d'origine, et l'enveloppe convexe se déterminera naturellement.

2ème méthode avec les équations de droites

La deuxième méthode que nous avons établie consiste à calculer des équations de droites.

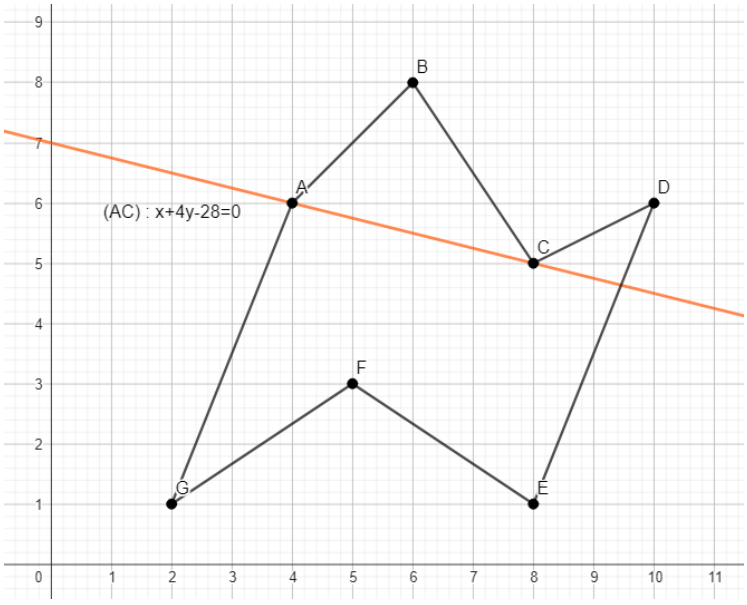
Mais, il faut d'abord visualiser pourquoi est-ce qu'on peut utiliser les équations de droites. On trace une droite passant par deux points de l'île et cette droite partage le plan en deux :



Si les points de l'île sont tous d'un côté de la droite, alors le segment qui appartient à la droite appartient aussi à l'enveloppe convexe.

Les points sont du même côté s'ils vérifient tous :
soit $-5x+2y+8 > 0$
soit $-5x+2y+8 < 0$
soit $-5x+2y+8 = 0$
Ici, ils vérifient tous $-5x + 2y + 8 < 0$ et on peut voir visuellement que tous les points sont sur ou à droite de la droite.

Donc [AG] appartient à l'enveloppe convexe.



A l'inverse, s'il y a des points de chaque côté de la droite, alors celle-ci ne forme pas une partie de l'enveloppe convexe.

Déterminer l'équation de la droite

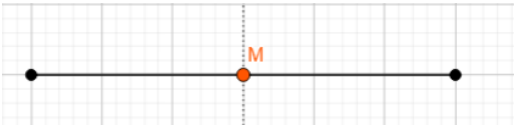
Pour exécuter cette méthode, il faut donc calculer l'équation cartésienne (de forme ax + by + c = 0) de la droite passant par le point de départ et le point suivant. Si les résultats des calculs de ax + by + c avec les coordonnées des autres points sont tous de même signe, le segment formé par les deux points fait partie de l'enveloppe convexe, sinon on répète le même processus avec le point suivant jusqu'à revenir au point de départ. L'enveloppe convexe sera ainsi complètement déterminée. Mais le faire à la main serait trop long et fastidieux, c'est pourquoi un programme python a été codé.

```
26  def equation_de_la_droite(pointLoin, pointA):
27      if pointLoin[0]!=pointA[0]:
28          a = (pointLoin[1]-pointA[1])/(pointLoin[0]-pointA[0])
29          c=1
30          b = pointA[1]-a*pointA[0]
31      else:   #pour traiter le cas de la droite verticale
32          a=1
33          c=0
34          b = -pointA[0]
35      equationDroite=[a,b,c]
36      return equationDroite
```

Déterminer le point de départ avec le barycentre

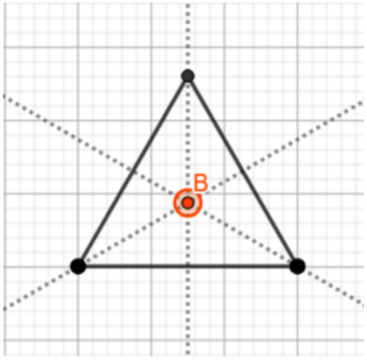
D'autre part, il faut déterminer le point de départ du nageur, et il nous a paru évident que le meilleur point de départ serait le point le plus éloigné du barycentre (centre de la figure) qui fera d'ailleurs toujours partie de l'enveloppe convexe. Il faut donc tout d'abord calculer le barycentre.

Pour trouver le milieu de deux points, il faut faire les demi-sommes de chacune des coordonnées des deux points.



$$M = \left(\frac{x_1 + x_2}{2}; \frac{y_1 + y_2}{2}\right)$$

Le calcul pour trouver le barycentre d'une figure est similaire : c'est la somme de chacune des coordonnées de tous les points du polygone divisée par le nombre de points de la figure qui donne ainsi les coordonnées du barycentre.



$$B = \left(\frac{x_1 + x_2 + x_3}{3}; \frac{y_1 + y_2 + y_3}{3}\right)$$

Formule générale : $B = \left(\frac{\Sigma x}{nx}; \frac{\Sigma y}{ny}\right)$

```
3   def barycentre (L):
4       X=0
5       Y=0
6       for i in range (len(L)):
7           X=X+L[i][0]
8           Y=Y+L[i][1]
9       b = [X/len(L),Y/len(L)]
10      return b
```

Ci-contre, le morceau de code calculant le barycentre :

Pour trouver le point le plus éloigné du barycentre, on calcule la distance entre le barycentre et chaque point de notre île avec les vecteurs. On compare les distances entre elles et le point qui forme la plus grande distance avec le barycentre sera le point de départ.

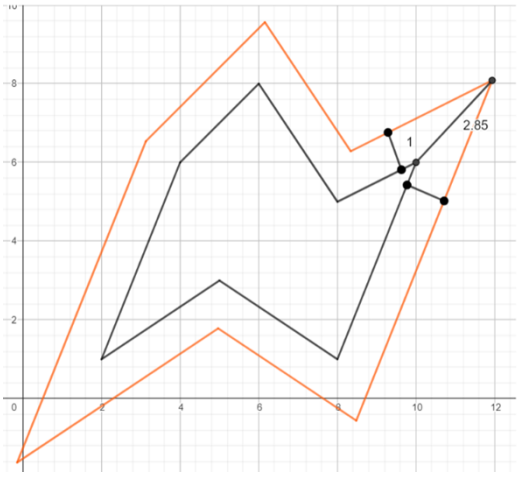
En ayant déterminé l'enveloppe convexe de l'île qui est la trajectoire la plus courte possible pour le nageur ainsi que le point de départ, nous avons donc répondu à notre sujet.

```
16  def point_le_plus_loin(b,L):
17      pointLoin = 0
18      distanceLoin = distance(b,L[0])
19      for i in range(1,len(L)):
20          distanceTest = distance(b,L[i])
21          if distanceTest > distanceLoin :
22              pointLoin = i
23              distanceLoin = distanceTest
24      return pointLoin
```

Et s'il y avait d'autres possibilités ? D'autres facteurs ?

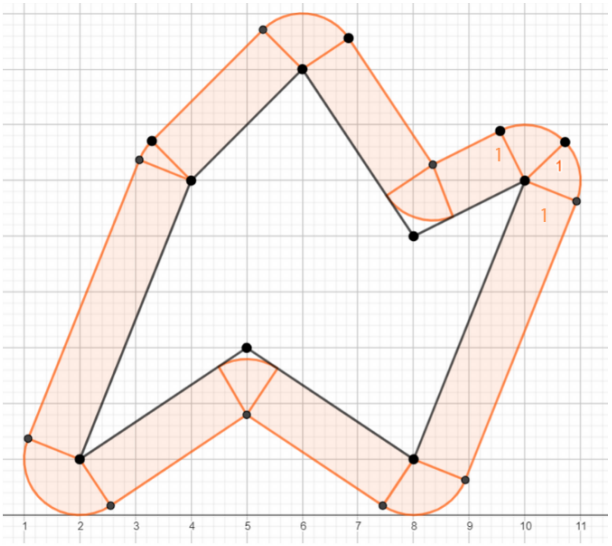
Notre réponse n’est pas applicable dans des conditions réelles, outre le fait qu’il est peu probable qu’un humain soit capable de faire le tour complet d’une île à la nage de manière constante et sans dévier de la trajectoire déterminée, s’il devait vraiment nager, il ne serait pas comme l’implique l’enveloppe convexe, collé au rivage.

Nous avons donc représenté graphiquement, en orange, la trajectoire du nageur s’il nageait à un mètre des bords de l’île.

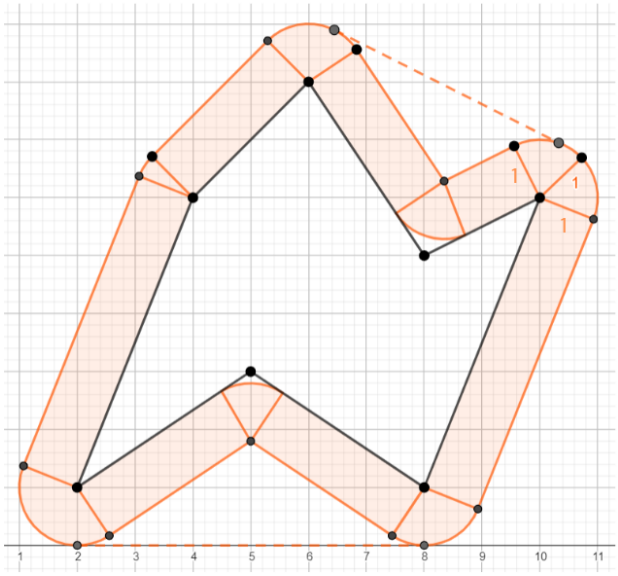


Mais on peut voir que si le nageur nage à un mètre des côtes, il lui arrive de dépasser cette distance lorsqu’il arrive aux extrémités de l’île.

Si l’on veut que notre nageur nage à exactement 1 mètre du bord en tout point, il faudrait tracer des arcs de cercles aux sommets du polygone de cette manière :



Et l’enveloppe convexe sera naturellement différente de celle de départ :



Cependant, il faut rappeler que le nageur doit seulement nager en ligne droite, cette nouvelle trajectoire ne correspond donc plus au sujet mais pourrait être une suite à nos recherches.

De même, nous pourrions nous interroger sur la réalisation de ces méthodes dans l’espace et appliquer nos recherches à d’autres problématiques comme l’emballage de cadeaux en optimisant la consommation de papier.

Programme Python

```python
from math import*

def barycentre (L):   # en faisant la moyenne des coordonnées on obtient les coordonnées du barycentre
    X=0
    Y=0
    for i in range (len(L)):
        X=X+L[i][0]              #PROBLEME   DE TYPE!!!!!
        Y=Y+L[i][1]
    b = [X/len(L),Y/len(L)]
    return b

def distance(point1,point2) :    # fonction qui permet de calculer une distance entre deux points
    d= sqrt((point1[0]-point2[0])**2+(point1[1]-point2[1])**2)
    return d

def point_le_plus_loin(b,L):    #on cherche le point le plus loin du barycentre, s'il y en a plusieurs, il prend le premier
    pointLoin = 0
    distanceLoin = distance(b,L[0])
    for i in range(1,len(L)):
        distanceTest = distance(b,L[i])
        if distanceTest > distanceLoin :
            pointLoin = i
            distanceLoin = distanceTest
    return pointLoin      #retourne l'indice du point le plus loin dans la liste

def equation_de_la_droite(pointLoin, pointA):          # calcule l'équation d'une droite passant par deux points
    if pointLoin[0]!=pointA[0]:
        a = (pointLoin[1]-pointA[1])/(pointLoin[0]-pointA[0])   #on part sur une équation de la forme "cy=ax+b" où c=1 si elle est réduite et c=0 si droite verticale
        c=1
        b = pointA[1]-a*pointA[0]
    else:   #pour traiter le cas de la droite verticale
        a=1
        c=0
        b = -pointA[0]
    equationDroite=[a,b,c]
    return equationDroite             # "cy=ax+b" retourne en liste le coefficient directeur a et l'ordonnée à l'origine b de l'équation de la droite et c devant le y

```

```
38
39 def enveloppe(L,pointLoin,decalage):           #données: la liste de p
   oints, le point loin et un paramètre décalage
40     depart=(pointLoin+decalage)%len(L)
41     print(depart,pointLoin)
42                          # on prend le point après le poinloin
43     test=True
44     eq=equation_de_la_droite(L[pointLoin],L[depart])
45     print(eq)
46                 #retourne a et b de la droite passant par point loin et
   un point situé après.
47     A = list(range(len(L))) # on crée une liste momentanée avec les ind
   ices des points sauf le point Loin et le point de départ. !!!!
48     A.remove(pointLoin)
49     A.remove(depart)
50     print(A)
51     signe=eq[2]*L[A[0]][1]-(eq[0]*L[A[0]][0]+eq[1])    #on calcule cy -
   (ax+b) avec le 1er point qui n'est pas sur la droite !!!!!!
52
53     compteur=1
54     while test and compteur<=len(L)-3:
55         calcul=  eq[2]*L[A[compteur]][1]- (eq[0]*L[A[compteur]][0]+eq[1
   ])   # on calcule la valeur de cy - (ax+b) avec le point suivant de la
    liste A !!!!!
56
57         if calcul*signe<0:    #s'ils n'ont pas le même signe alors le t
   est a échoué sinon on continue
58             test=False
59         compteur+=1
60
61     return(test)  #dans tous les cas à la fin on retourne le test qui r
   estera true si pas de souci et sera passé à false si incohérence des si
   gnes
62
63 def processus():  #on lance le processus jusqu'à ce qu'une droite soit
    trouvée
64     decalage=1
65     while enveloppe(L,pointLoin,decalage)==False:
66         decalage+=1
67     return(decalage)  #retourne l'indice qui a fonctionné soit la posit
   ion du point à relier à pointloin pour assurer la convexité!
68
69
70 # ensuite il faut itérer le procédé pour tracer TOUTE L'ENVELOPPE CONVE
   XE
71
72 #le nouveau pointloin est celui du décalage et on recommence
73
74
75
76
77
```

```
78 #PARTIE GRAPHIQUE DU PROGRAMME
79 import pygame,sys
80 from pygame.locals import*
81
82 pygame.init()
83 largeur=500
84 longueur=500
85 fenetre=pygame.display.set_mode((longueur,largeur),0,32)
86 fenetre.fill((255,255,255))
87
88 L=[]
89 i=0
90
91 blue=(0,0,255)
92 white=(255,255,255)
93 red=(255,0,0)
94 basicFont=pygame.font.SysFont(None,20)
95
96 text1=basicFont.render('Polygone',True,blue,white)
97 text1Rect=text1.get_rect()
98 text1Rect.centerx=455
99 text1Rect.centery=115
100
101 rect1=pygame.draw.rect(fenetre,blue,(text1Rect.left-5,text1Rect.top-10,
    text1Rect.width+10,text1Rect.height+20 ),2)
102
103 fenetre.blit(text1,text1Rect)
104
105
106 text2=basicFont.render('Enveloppe',True,red,white)
107 text2Rect=text1.get_rect()
108 text2Rect.centerx=455
109 text2Rect.centery=180
110
111 rect2=pygame.draw.rect(fenetre,red,(text2Rect.left-5,text2Rect.top-10,t
    ext2Rect.width+10,text2Rect.height+20 ),2)
112
113 fenetre.blit(text2,text2Rect)
114
115
```

```
while True:
    for event in pygame.event.get():
        if event.type==QUIT:
            pygame.quit()
            sys.exit()

        if event . type == MOUSEBUTTONDOWN  and event.button ==1 and ev
    ent.button ==1 and not rect1.collidepoint(event.pos) and not rect2.coll
    idepoint(event.pos) :
            nouveau_point=pygame.mouse.get_pos()
            L.append([nouveau_point[0],nouveau_point[1]])
            if len(L)!=1:
                pygame.draw.line(fenetre,(0,255,0),L[i],L[i+1])
                i+=1
        if event . type == MOUSEBUTTONDOWN  and event.button ==1 and re
    ct1.collidepoint(event.pos):
            pygame.draw.polygon(fenetre,(0,255,0),L,0)
            print("La liste des points est", L)
        if event . type == MOUSEBUTTONDOWN  and event.button ==1 and re
    ct2.collidepoint(event.pos):
            b=barycentre(L)
            print("Le barycentre est",b)
            pointLoin=point_le_plus_loin(b,L)  #indice du point Loin
            print("Le point Loin a pour coordonnée",pointLoin)
            enveloppe_finale=[L[pointLoin]]  #on initialise l'enveloppe
    avec le premier point qui est le point Loin
            while enveloppe_finale.count(L[pointLoin])==1:   #on attend
    que le point Loin soit choisi à nouveau dans notre enveloppe pour dire
     qu'on a bouclé et
                #qu'on a fini l'enveloppe
                p=processus()
                print("le décalage appliquée est",p)
                enveloppe_finale.append(L[(pointLoin+p)%len(L)])
                pointLoin=(pointLoin+p)%len(L)
            print("enveloppe",enveloppe_finale)
            pygame.draw.polygon(fenetre,(255,0,0),enveloppe_finale,2)

    pygame.display.update()

```