



TIPE MP2I :  
SPORT, JEU

# DÉPLACEMENT D'UNE PLANCHE À VOILE EN LIGNE DROITE

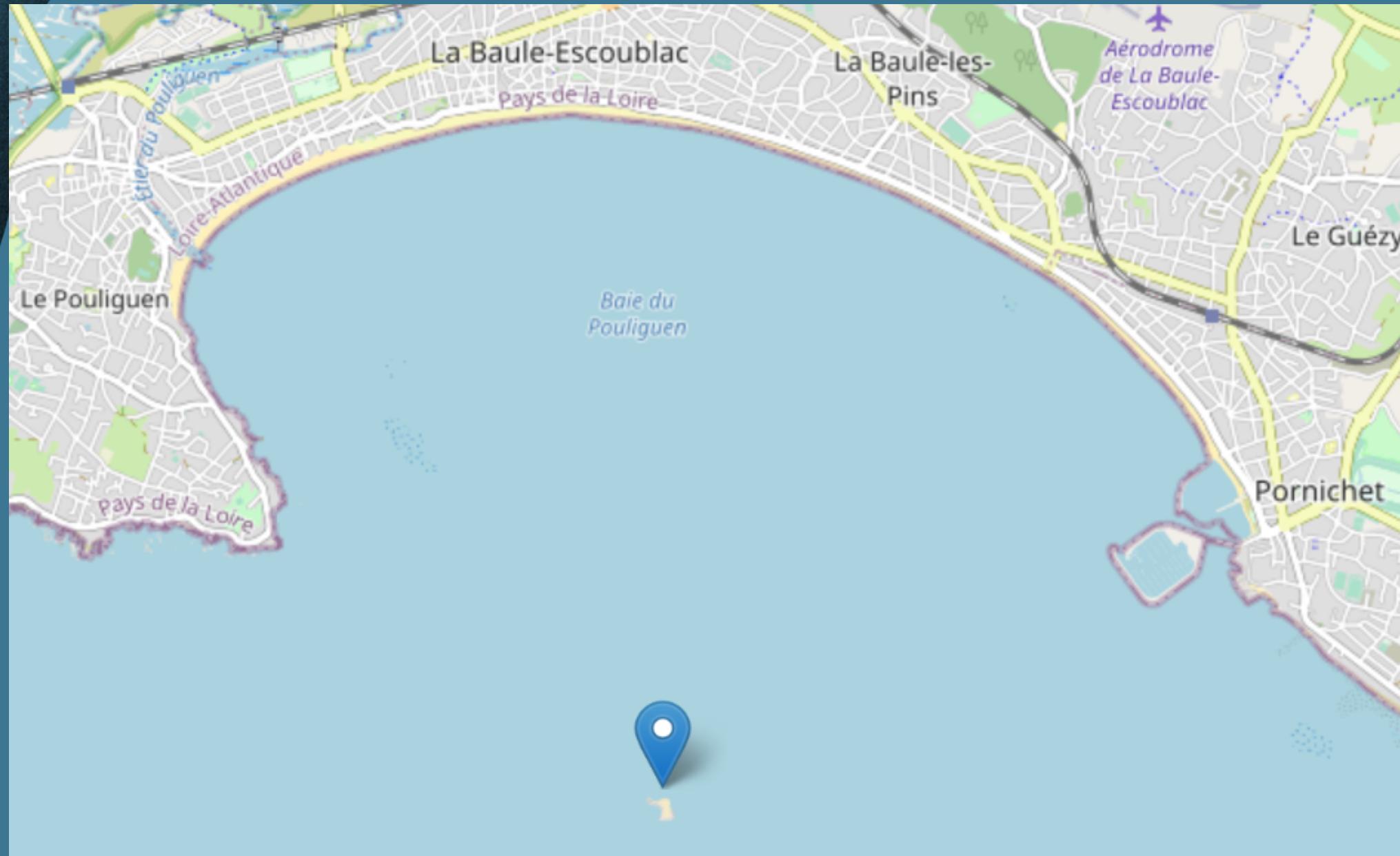
# INTRODUCTION



- 00 Présentation du TIPE
- 01 Organisation des tâches
- 02 Cartographier la baie de la Baule
- 03 Modélisation d'une trajectoire en ligne droite sur une carte
- 04 Traduction de l'équation physique en code informatique
- 05 Conclusion

# 00 PRÉSENTATION DU TIPE

## PROBLÉMATIQUE



### Paramètres

---

**Le poids de la planche et  
du planchiste**

**Le temps en temps réel**

# 01 ORGANISATION DES TÂCHES

**Kevin Kuzu**

**Modélisation de la carte**

**Recherche du meilleur  
trajet**

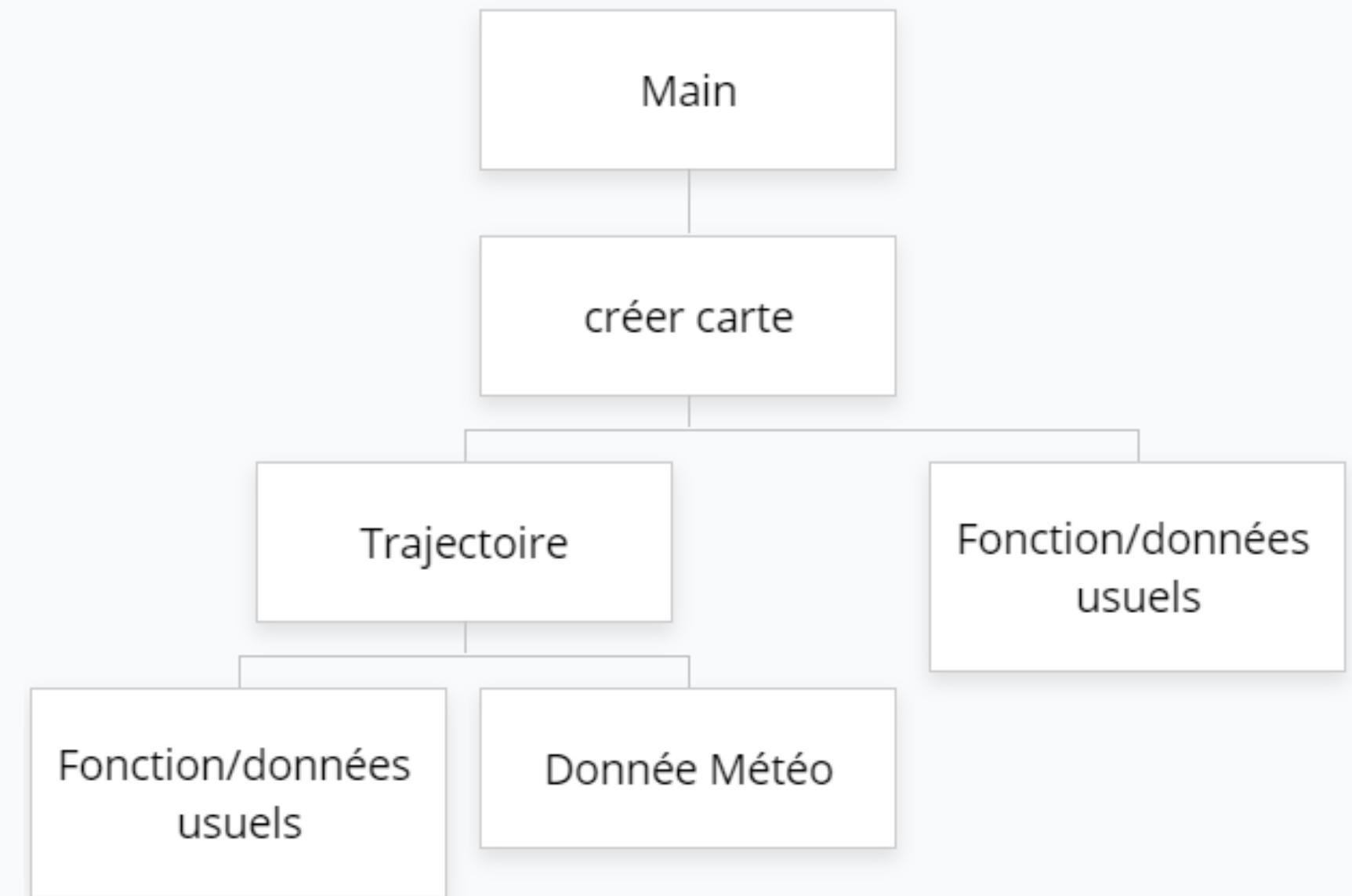
**Paul Le Drogo**

**Modélisation physique**

**Déterminer les temps de  
trajet**

# 01 ORGANISATION DES TÂCHES

## ARBORESCENCE DU CODE



# 02 CARTOGRAPHIER LA BAIE DE LA BAULE

## BASE DE DONNÉES

	Latitude	Longitude	Angle	Distance
	Filtre	Filtre	Filtre	Filtre
1	47.2707281235484	-2.42483443406442	126.375840000085	3.97187229667424
2	47.2707281235484	-2.42383443406442	126.370796817987	3.91851533943555
3	47.2707281235484	-2.42283443406442	126.365753635888	3.86589525023291
4	47.2707281235484	-2.42183443406442	126.36071045379	3.81404252759464
5	47.2707281235484	-2.42083443406442	126.355667271692	3.7629888938234
6	47.2717281235484	-2.42483443406442	126.350624089593	4.05073163453344
7	47.2717281235484	-2.42383443406442	126.345580907495	3.99842814778924
8	47.2717281235484	-2.42283443406442	126.340537725397	3.94687458781507
9	47.2717281235484	-2.42183443406442	126.335494543298	3.89610072403145
10	47.2717281235484	-2.42083443406442	126.3304513612	3.84613743548935
11	47.2727281235484	-2.42483443406442	126.325408179102	4.131078870069
12	47.2727281235484	-2.42383443406442	126.320364997003	4.07980654719797
13	47.2727281235484	-2.42283443406442	126.315321814905	4.02929529323369
14	47.2727281235484	-2.42183443406442	126.310278632807	3.97957408818393
15	47.2727281235484	-2.42083443406442	126.305235450708	3.93067291343787
16	47.2737281235484	-2.42483443406442	126.30019226861	4.21282887241463

## Obtention des données

Latitude Longitude :  
Génération de points

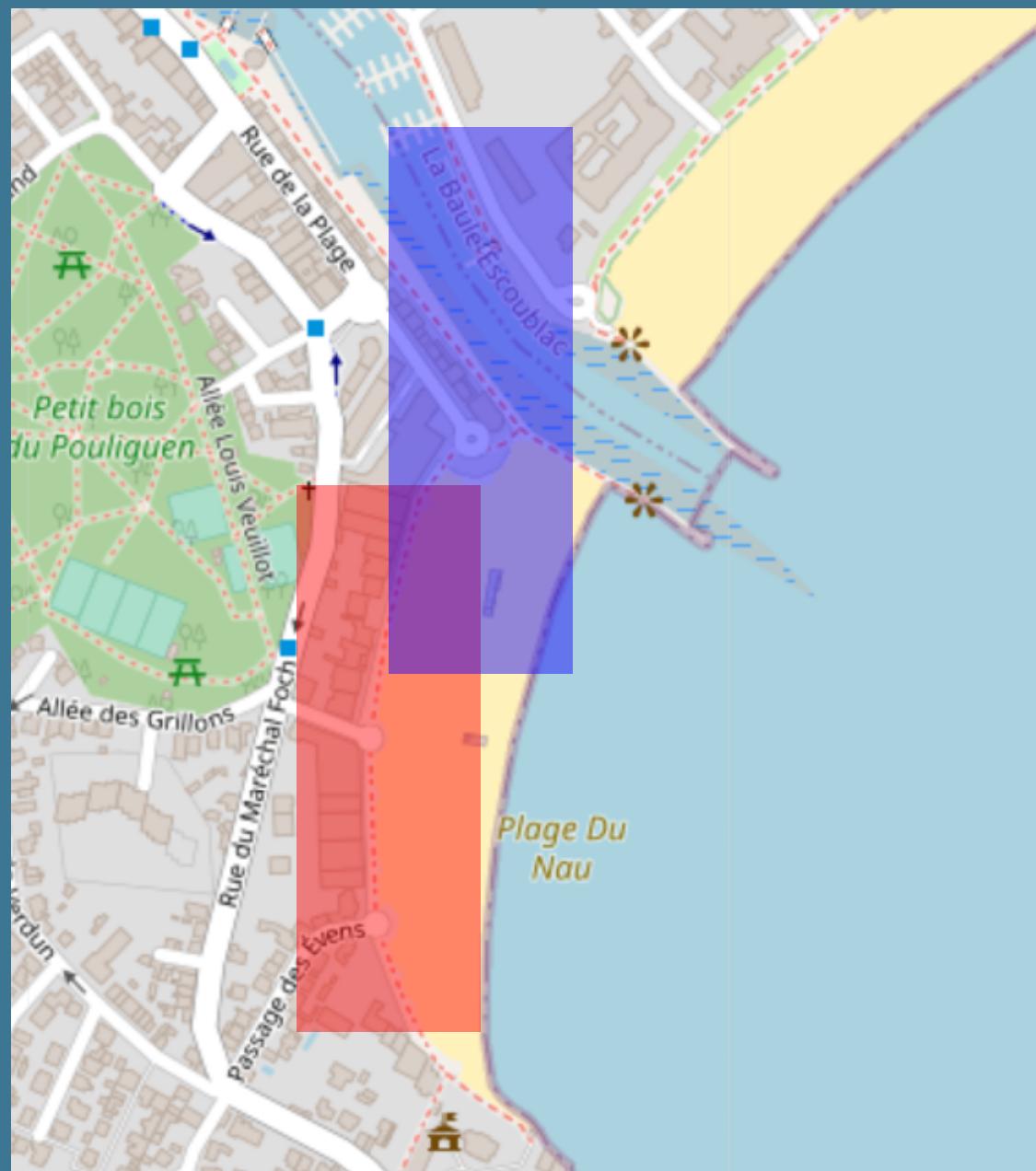
Distance :  
Formule Haversine

Angle

# 02 CARTOGRAPHIER LA BAIE DE LA BAULE

## BASE DE DONNÉES

### Latitude Longitude : Génération de points



```
# Les limites géographiques d'une zone de la Baie de La Baule
lat_max = 47.27599848411349
lat_min = 47.26199364389432
lon_max = -2.3415423853609005
lon_min = -2.3504251728775216

# L'espacement souhaité entre les points
spacing = 0.0001

# Générer les coordonnées latitude/longitude
lats = np.arange(lat_min, lat_max, spacing)
lons = np.arange(lon_min, lon_max, spacing)

# Créer une grille de points en combinant les coordonnées latitude/longitude
points = np.array(np.meshgrid(lats, lons)).T.reshape(-1, 2)

# Convertir les coordonnées en DataFrame pour une utilisation facile
df = pd.DataFrame(points, columns=['Latitude', 'Longitude'])

# Enregistrer les données dans une base de données (par exemple, un fichier CSV)
df.to_csv('baie_de_la_baule_pointsbiq.csv')
```

# 02 CARTOGRAPHIER LA BAIE DE LA BAULE

## BASE DE DONNÉES

Distance :  
Formule Haversine



$$\text{hav}\left(\frac{d}{r}\right) = \text{hav}(\varphi_2 - \varphi_1) + \cos(\varphi_1) \cos(\varphi_2) \text{hav}(\lambda_2 - \lambda_1)$$
$$+ 2r \arcsin\left(\sqrt{\sin^2\left(\frac{\varphi_2 - \varphi_1}{2}\right) + \cos(\varphi_1) \cos(\varphi_2) \sin^2\left(\frac{\lambda_2 - \lambda_1}{2}\right)}\right)$$



```
def distance(lat1, lon1, lat2, lon2):
    # Conversion des degrés en radians
    lat1_rad = radians(lat1)
    lon1_rad = radians(lon1)
    lat2_rad = radians(lat2)
    lon2_rad = radians(lon2)

    # Rayon de la Terre en kilomètres
    radius = 6371.0

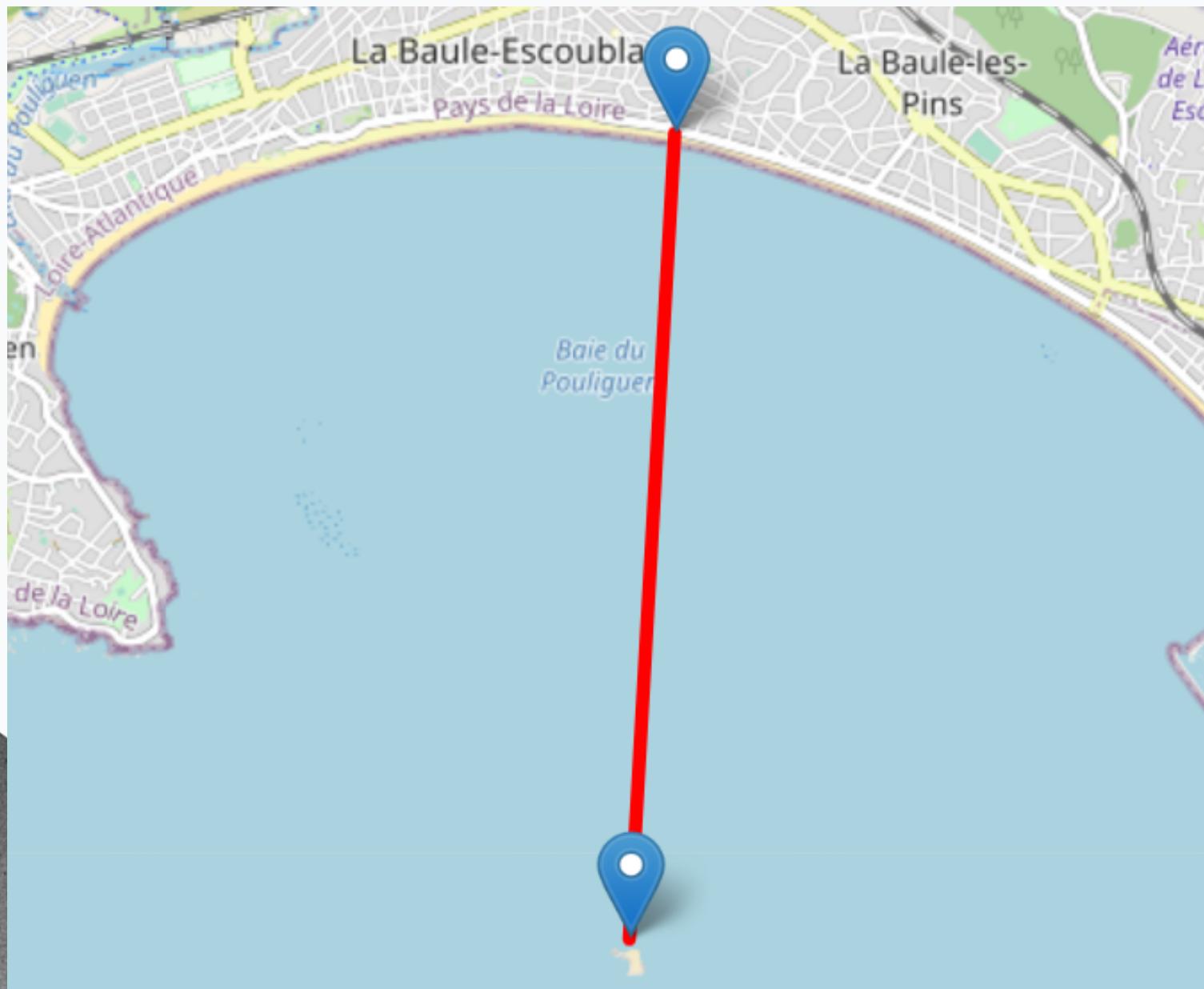
    # Différence des longitudes et des latitudes
    dlon = lon2_rad - lon1_rad
    dlat = lat2_rad - lat1_rad

    # Formule de la distance entre deux points sur une sphère, formule de Haversine
    a = sin(dlat / 2)**2 + cos(lat1_rad) * cos(lat2_rad) * sin(dlon / 2)**2
    c = 2 * asin(sqrt(a))

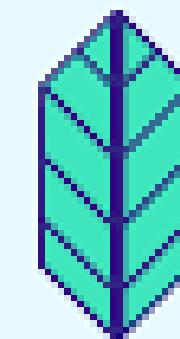
    # Distance en kilomètres
    distance_km = radius * c

    return distance_km
```

# 03 MODÉLISATION D'UNE TRAJECTOIRE EN LIGNE DROITE SUR UNE CARTE



**Bibliothèque python**



**Folium**



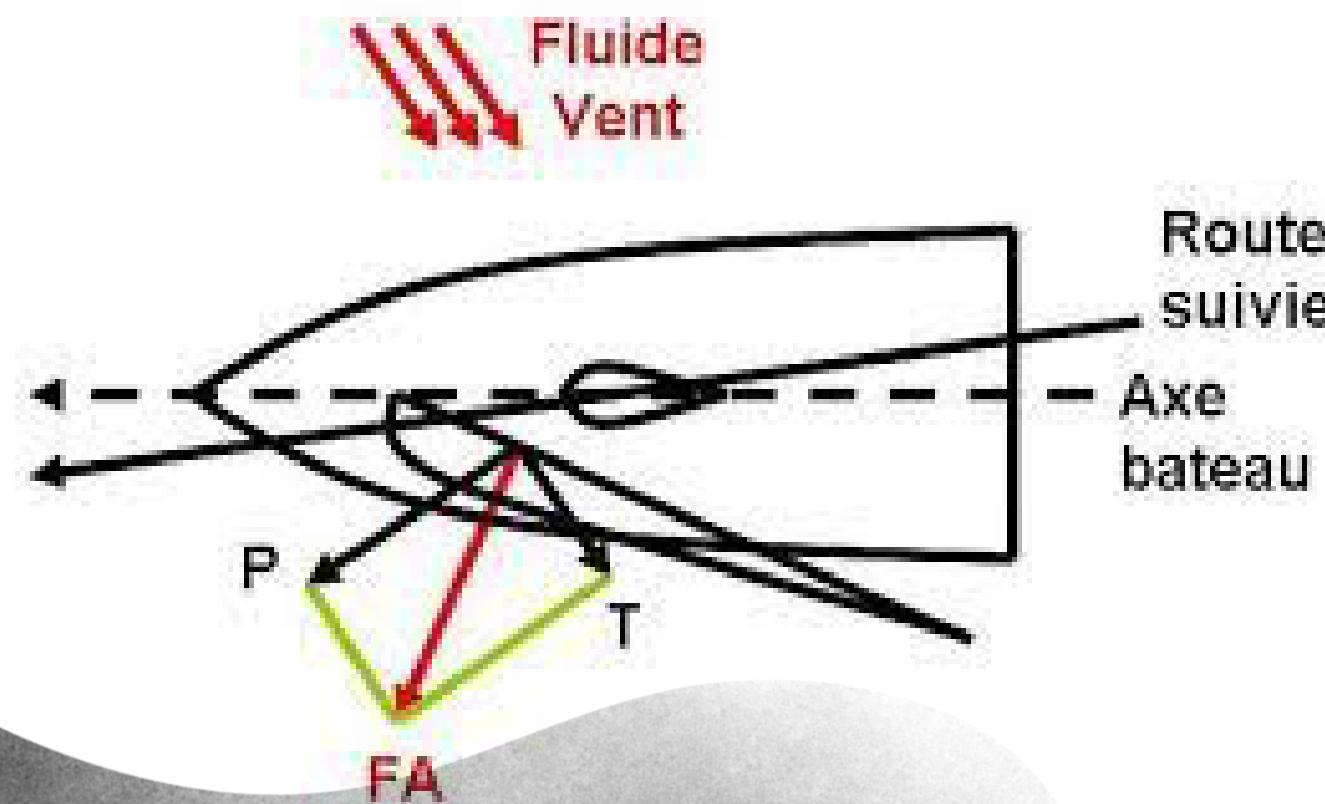
**OpenWeather**

# 04 TRADUCTION DE L'ÉQUATION PHYSIQUE EN CODE INFORMATIQUE

Théorème de la résultante cinétique :

$$m\vec{a} = \vec{F}_P \sin(\alpha) + \vec{F}_T \cos(\alpha)$$

Equation finale :  $\vec{v} = t \frac{1}{2m} \rho S v^2 (C_P \sin(\alpha) + C_T \cos(\alpha))$



$$C_P = \frac{\vec{F}_P}{2\rho v^2 S}$$

$$C_T = \frac{\vec{F}_T}{2\rho v^2 S}$$

Avec :

1.  $F_P$  Force de porté.
2.  $F_T$  Force de trainée
3.  $\rho$  la masse volumique de l'air  $\rho = 1.2 \text{ kg.m}^{-3}$ .
4.  $S$  la surface de la voile.

# 04 TRADUCTION DE L'ÉQUATION PHYSIQUE EN CODE INFORMATIQUE

## BASE DE DONNÉES

	Poids	Vent	Voile1	Voile2
	Filtre	Filtre	Filtre	Filtre
1	60	4	7	9
2	60	5	6	8
3	60	7	5	6.5
4	60	9	4.5	5.5
5	60	11	3.3	4.3
6	60	13	2.8	3.7
7	60	16	2.5	3.5
8	60	19	2.3	3
9	60	19.1	2.7	2.7

Rider weight (kg)	Wind speed range (m/s)									Rider's skill level	
	3-4	4-5	5-7	7-9	9-11	11-13	13-16	16-19	19+	Beginner	Intermediate
Windsurf sail size (m <sup>2</sup> )									Windsurf board volume (liters)		
50-60	7-9	6-8	5-6,5	4,5-5,5	3,3-4,3	2,8-3,7	2,5-3,5	2,3-3	< 2,7	145-160	95-108
60-70	8-10	7-9	6-7,5	5-6,2	4-5	3,3-4	3,1-3,7	2,5-3,5	< 3,0	160-175	108-121
70-80	9-11	7-10	6,5-8	5,5-7	4,5-5,5	3,8-5	3,5-4,2	3-4	< 3,5	175-190	121-134
80-90	10-12	9-11	7,5-9,5	6,5-8	5-6	4,5-5,5	4,2-4,7	3,5-4,3	< 4	190-205	134-147
90+	12+	11+	9+	7,5+	5,5+	4,7+	4+	3,5+	< 4,5	205+	147+

source : <https://kiteforce.ca/fr/content/21-equipement>

# 04 TRADUCTION DE L'ÉQUATION PHYSIQUE EN CODE INFORMATIQUE

## BASE DE DONNÉES



```
# Récolte de données sur l'utilisateur
poid = int(input("Quel est votre poids ?"))
poid = arrondi_dizaine(poid)
if (poid > 90):
    poid = 90.1
if (poid < 60):
    poid = 60
print("Attention au petit poids !")

# Connexion à la base de données
connect = sqlite3.connect('./data/BDD.db')

# Création d'un curseur pour exécuter des requêtes
cur = connect.cursor()

# Recolte de données
cur.execute("SELECT Voile1,Voile2 FROM Planche_a_voile WHERE Poids = '{}' and Vent >= '{}'".format(poid,
int(wind_speed)))
data_voile = cur.fetchall()      # Données sur les voiles en fonction du poid

cur.execute("SELECT * FROM Coord_baie")
element = cur.fetchall()      # Toute les coordonnées 4-uplets (Latitude, Longitude, angle, distance (km))
```

# 04 TRADUCTION DE L'ÉQUATION PHYSIQUE EN CODE INFORMATIQUE



```
def function(t, surface, angle_planche, ct, cp):
    angle = abs(wind_deg - angle_planche) % 360
    return (1.0/(2.0*(poid + 50))) * t * 1.292 * surface*(wind_speed**2)*cp*sin(angle) +
ct*cos(angle)

# On determine la vitesse de la planche à un instant t

def determine_v(t, surface, angle_planche):
    ct, cp = coefficient_porte_traine(wind_deg, angle_planche)
    v = function(t, surface, angle_planche, ct, cp)
    return v
```

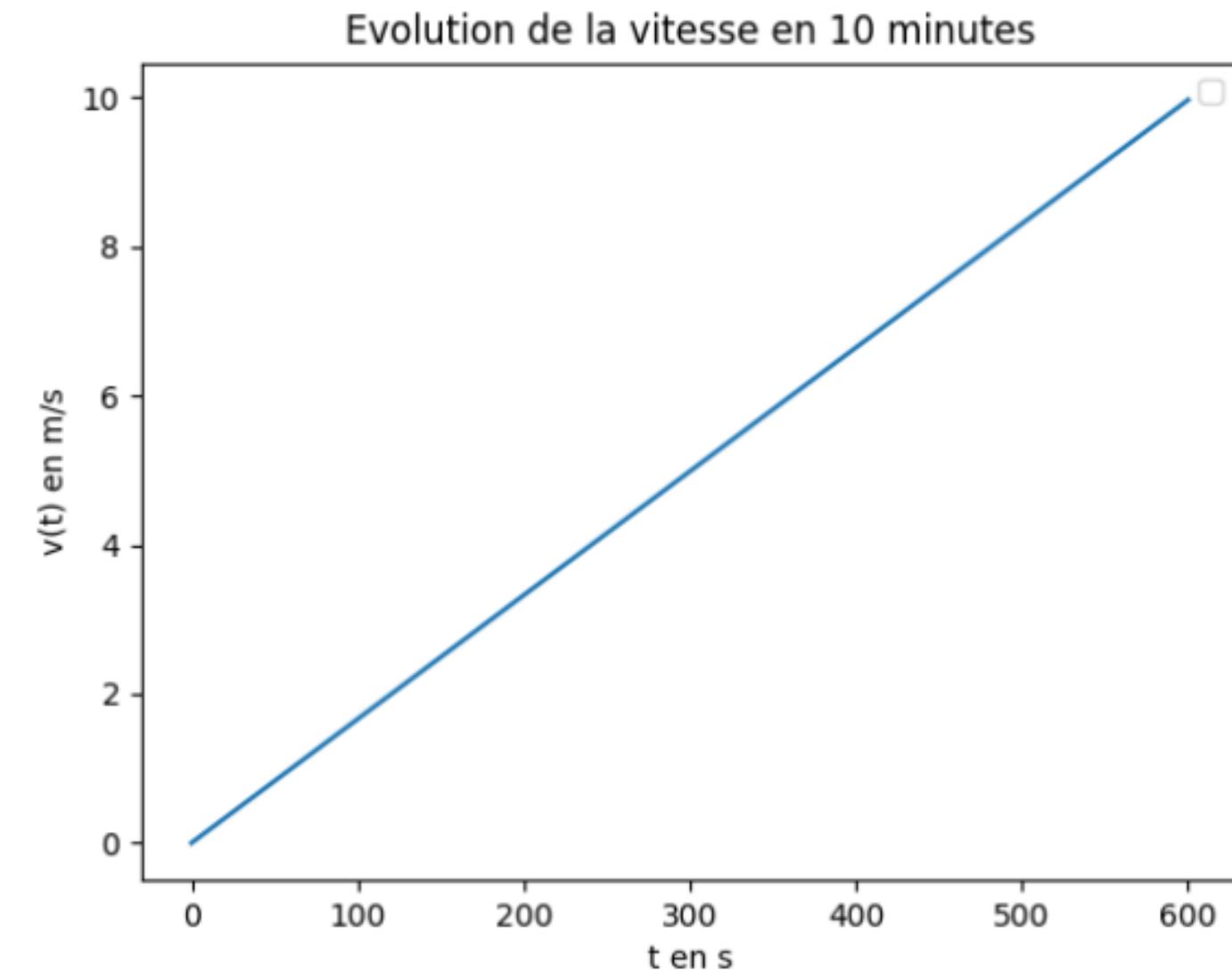
# 04 TRADUCTION DE L'ÉQUATION PHYSIQUE EN CODE INFORMATIQUE

## RECHERCHE DU TEMPS MINIMUM

```
# On fait le rapport de la distance/vitesse de tous les points pour obtenir le temps minimum

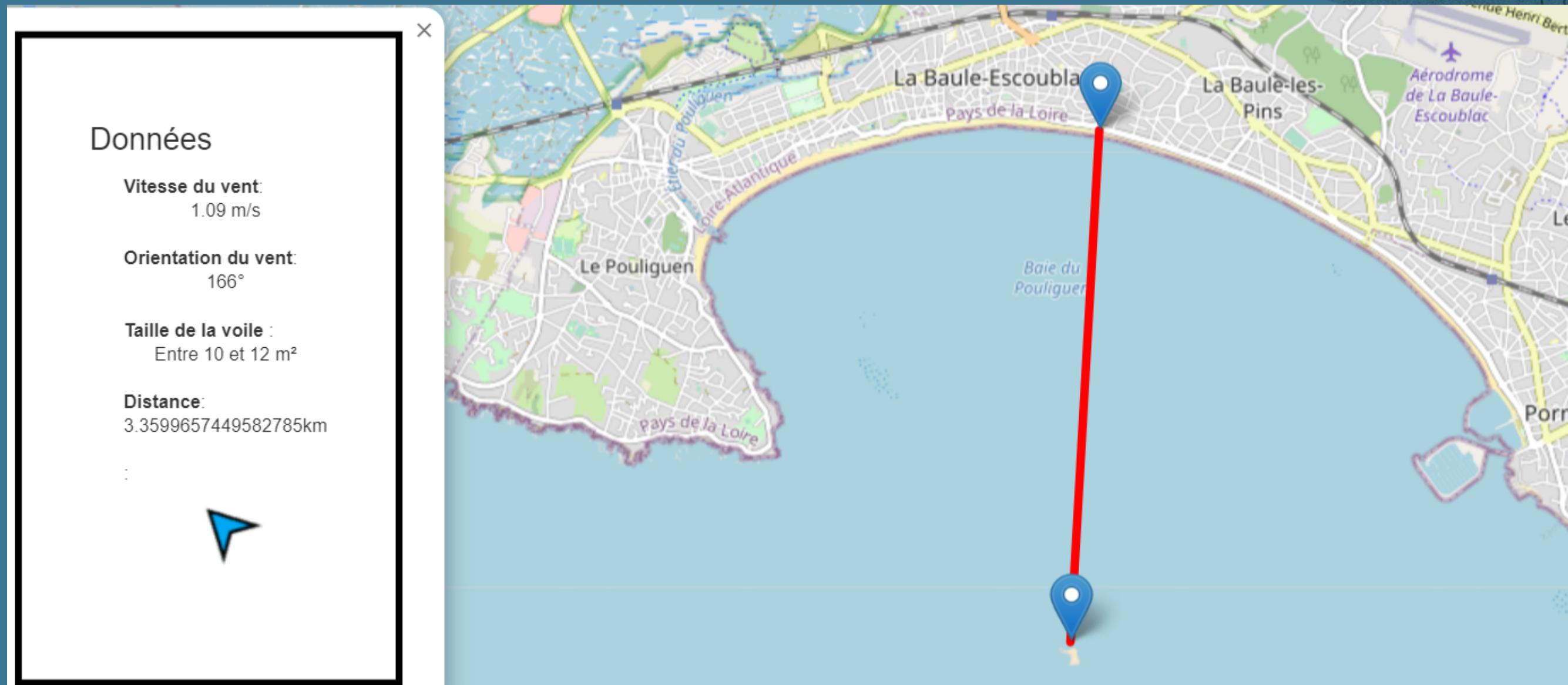
def liste_vitesse():
    vitesses = []
    for i in range (len(element)):
        v = determine_v(600, data_voile[0][0], element[i][2])
        vitesses.append(v)
    return vitesses

def temps_min():
    vitesses = liste_vitesse()
    temps = []
    for i in range (len(vitesses)):
        if(vitesses[i] != 0):
            t = element[i][3] / vitesses[i]
            temps.append(t)
        else :
            temps.append(700)
    indice_position = temps.index(min(temps))
    return indice_position
```



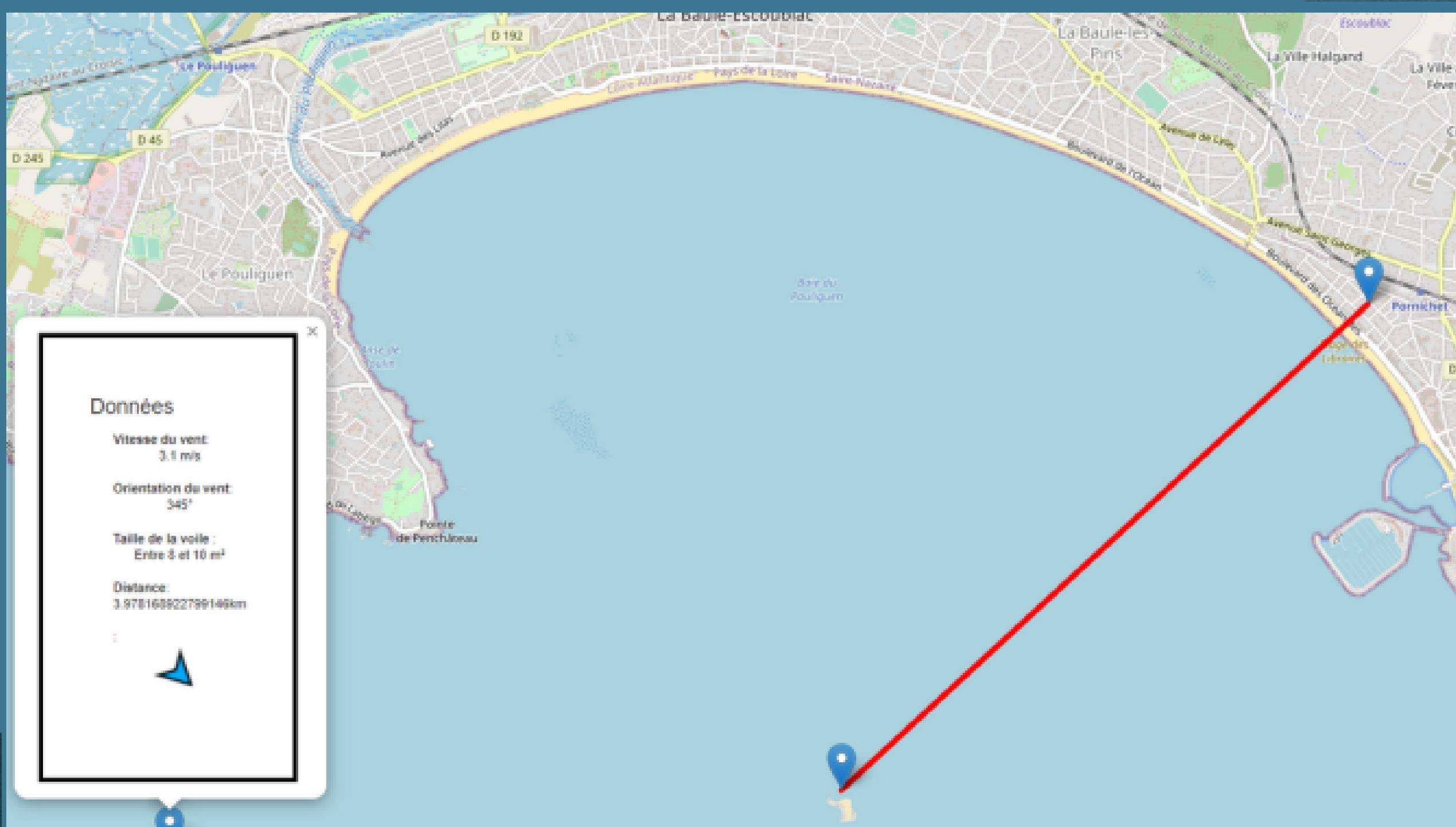
# 05 CONCLUSION

## RESULTAT FINAL



# 05 CONCLUSION

## RESULTAT FINAL



# 05 CONCLUSION

## DIFFICULTÉ

Equation non modélisable précisément

Beaucoup de paramètres négligés :  
Force de frottement  
Vent apparent

Complexité aberrantes

## AMÉLIORATIONS

Prendre en compte chaque facteur

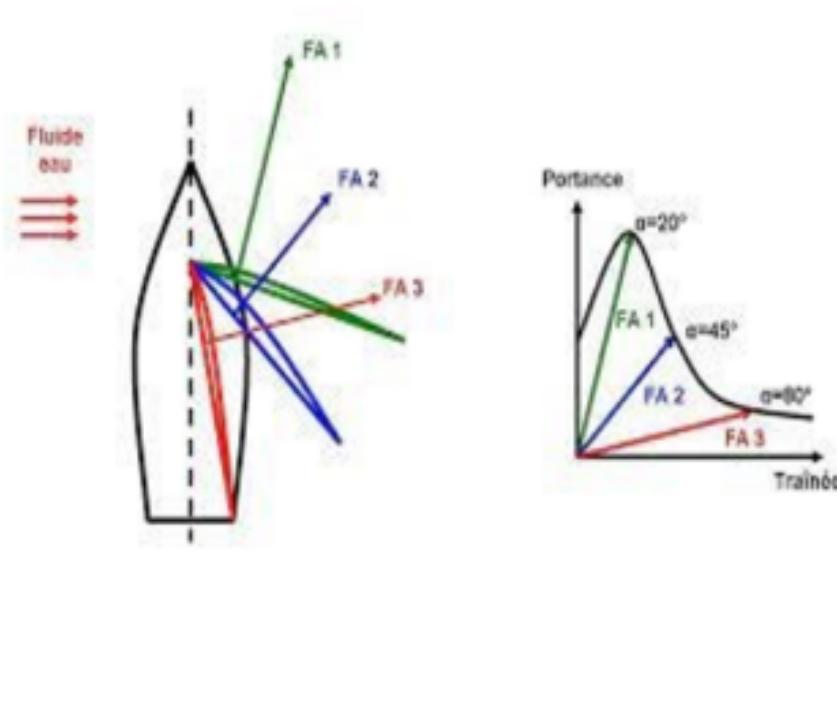
Amélioré la complexité de l'algorithme

MERCI POUR VOTRE  
ATTENTION

# ANNEXES

MODÉLISATION DE L'ÉQUATION / CODE

# MODÉLISATION DE L'ÉQUATION



Avec les coefficients on peut déterminer leurs forces. On note  $C_P$  le coefficient de portance et  $C_T$  le coefficient de trainé. Ainsi :

$$C_P = \frac{\vec{F}_P}{2\rho v^2 S}$$

$$C_T = \frac{\vec{F}_T}{2\rho v^2 S}$$

Avec :

1.  $F_P$  Force de portance.
2.  $F_T$  Force de trainé.
3.  $\rho$  la masse volumique de l'air  $\rho = 1.2 \text{ kg.m}^{-3}$ .
4.  $S$  la surface de la voile.
5.  $v$  la vitesse du vent.

Il suffit d'isoler la force pour pouvoir l'utiliser. Ce qui donne :

$$F_P = C_P 2 \rho v^2 S$$

$$F_T = C_T 2 \rho v^2 S$$

## Théorème de la résultante cinétique (TRC)

Il ne nous reste plus qu'à sommer les forces de portance et de trainé en projetant les deux forces sur le même vecteur que l'accélération. Ce qui nous donne :

$$m\vec{a} = \vec{F}_P + \vec{F}_T$$

$$ma = \frac{1}{2} \rho S v^2 (C_P \sin(\alpha) + C_T \cos(\alpha))$$

$$a = \frac{1}{2m} \rho S v^2 (C_P \sin(\alpha) + C_T \cos(\alpha))$$

On sait que  $\vec{a} = \frac{d\vec{v}}{dt}$ . On va donc sommer de  $t = 0$  à  $t = t$  afin d'avoir la vitesse nécessaire pour aller à l'Île des Evens et ainsi pouvoir comparer les vitesses maximales de chaque point de départ.

On obtient donc :

$$\frac{dv_P}{dt} = \frac{1}{2m} \rho S v^2 (C_P \sin(\alpha) + C_T \cos(\alpha))$$

$$dv_P = \frac{1}{2m} \rho S v^2 (C_P \sin(\alpha) + C_T \cos(\alpha)) dt$$

$$\int_{t_0}^t dv_P = \int_{t_0}^t \frac{1}{2m} \rho S v^2 (C_P \sin(\alpha) + C_T \cos(\alpha)) dt$$

$$v_P = \frac{1}{2m} \rho S v^2 (C_P \sin(\alpha) + C_T \cos(\alpha)) t$$

# CODE / MAIN



```
from creer_carte import*
#####
#/
#/          LE MAIN
#/
#####
# Propose la planche adapté en fonction du vent et du poids

# Créer la carte
make_map(wind_speed, wind_deg)

#####
#/
#/          COURBE DISTANCE DE LA
#/          MEILLEURS TRAJECTOIRE
#/
#####
t = np.linspace(0, 600, 600)
plt.plot(t, vitesse)
plt.show()
```

# CODE / CREER CARTE

```
import folium
from fonction_donnees_usuel import*
from trajectoire import*
from jinja2 import Template

#####
#//          CREATION DE LA CARTE
#//
#####

# Coordonnées de la carte centrée sur La Baule
lat = 47.264
lon = -2.385

def make_map(windspeed, winddeg):

    # Donnée de la voile pour sa taille en fonction du poid et du vent
    voile = "Entre {} et {} m2".format(data_voile[0][0],data_voile[0][1])

    # Donné de la direction du vent en fonction du degré du vent

    image = "<img src= image/{}°.png>".format(rosace(winddeg))

    # Création de la carte
    ma_carte = folium.Map(location=[lat, lon], zoom_start=13)

    # Coordonnées prise en fonctions des paramètres du vent et de son orientation
    point_de_depart = [element[indicebest][0],element[indicebest][1]]
    point_d_arrive = [47.24563801228261, -2.387373826689938]
```

```
# Ajout d'un marqueur sur le point de départ
folium.Marker(location=point_de_depart, tooltip='Le départ !').add_to(ma_carte)

# Ajout d'un marqueur sur l'arrivé
folium.Marker(location=point_d_arrive, tooltip="L'arrivé !").add_to(ma_carte)

# Ajout du chemin entre le départ et l'arrivé
folium.PolyLine(locations=[point_de_depart, point_d_arrive], color='red', weight=5).add_to(ma_carte)

    # Formatage de la chaîne HTML avec les données
template = Template(html_template)
formatted_html = template.render(data=data)

# Ajouter l'encadré à la carte
folium.Marker([47.235536057291405, -2.4741868229542208],
popup=folium.Popup(formatted_html)).add_to(ma_carte)

# Enregistrement de la carte dans un fichier HTML
ma_carte.save('ma_carte.html')
```

# CODE / TRAJECTOIRE



```
from fonction_donnees_usuel import*
from meteo_data import*
import numpy as np
from math import*
import matplotlib.pyplot as plt

#####
# EQUATION PHYSIQUE
#####
def function(t, surface, angle_planche, ct, cp):
    angle = abs(wind_deg - angle_planche) % 360
    return (1.0/(2.0*(poid + 50))) * t * 1.292 * surface * (wind_speed**2) * abs(cp*sin(angle) - ct*cos(angle))

# On determine la vitesse de la planche à un instant t
def determine_v(t, surface, angle_planche):
    ct, cp = coefficient_porte_traine(wind_deg, angle_planche)
    v = function(t, surface, angle_planche, ct, cp)
    return v
```



```
#####
# RECHERCHE TEMPS MINIMUM
#####
# On fait le rapport de la distance/vitesse de tous les points pour obtenir le temps minimum

def liste_vitesse():
    vitesses = []
    for i in range (len(element)):
        v = determine_v(600, data_voile[0][0], element[i][2])
        vitesses.append(v)
    return vitesses

def temps_min():
    vitesses = liste_vitesse()
    temps = []
    for i in range (len(vitesses)):
        if(vitesses[i] != 0):
            t = element[i][3] / vitesses[i]
            temps.append(t)
        else :
            temps.append(700)
    indice_position = temps.index(min(temps))
    return indice_position

indicebest = temps_min()
print(indicebest)

vitesse = []
for t in range (600):
    vitesse.append(determine_v(t, data_voile[0][0], element[indicebest][2]))
```

# CODE / METEO DATA

```
● ○ ●

///////////////////////////////
///                      //
///          METEO DATA   //
///                      //
///////////////////////////////

import requests

city_name = "Baule"
language = "fr"
clef = "e390b048d4f4d1e00adf7da19dd55613"
api_lien = f"https://api.openweathermap.org/data/2.5/weather?q={city_name}&lang={language}&appid={clef}"


json = requests.get(api_lien).json()

# Vitesse du vent
wind_speed = json["wind"]["speed"]

# Orientation du vent
wind_deg = json["wind"]["deg"]


def conversion(data):
    return (data * 3.6)

wind_speedbis = conversion(float(wind_speed))

print(wind_speed, "m/s")
print(wind_speedbis , "km/h")
print("Direction du vent",wind_deg, "°")
```

# BIBLIOGRAPHIE

AUTEURS	TITRE	INFORMATION COMPLEMENTAIRE
Wikipédia	Firmule Haversin	<a href="https://fr.wikipedia.org/wiki/Formule_de_haversine">https://fr.wikipedia.org/wiki/Formule_de_haversine</a>
StackOverflow	Cartographie	Aucune
OpenWeather	Météo	<a href="https://openweathermap.org/current">https://openweathermap.org/current</a>
Kitesurf	Tableau du mètre carré de la planche à voile	<a href="https://kiteforce.ca/fr/content/21-equipement">https://kiteforce.ca/fr/content/21-equipement</a>