La modélisation des cyclones dans l'Atlantique Nord

PAUL Maël - N°40532

MP - Spécialité Informatique

Source : francebleu.fr

Contexte

- Le bassin Atlantique Nord est sujet à de nombreux cyclones.
- De nombreux organismes répertorient les caractéristiques de ces cyclones.
- La mise en place d'outils de prévision a permis de minimiser les dégâts matériels et humains.

Problématique

- Dans quelle mesure peut-on prévoir la trajectoire d'un cyclone à l'aide d'un modèle probabiliste simple ?

Plan de la présentation

1. Mise en place du modèle probabiliste
   - Récupération des données existantes
   - Discrétisation de l'espace
   - Présentation du modèle probabiliste
2. Programmation
   - Mise en oeuvre du modèle probabiliste
   - Exploitation des résultats
3. Limites du modèle

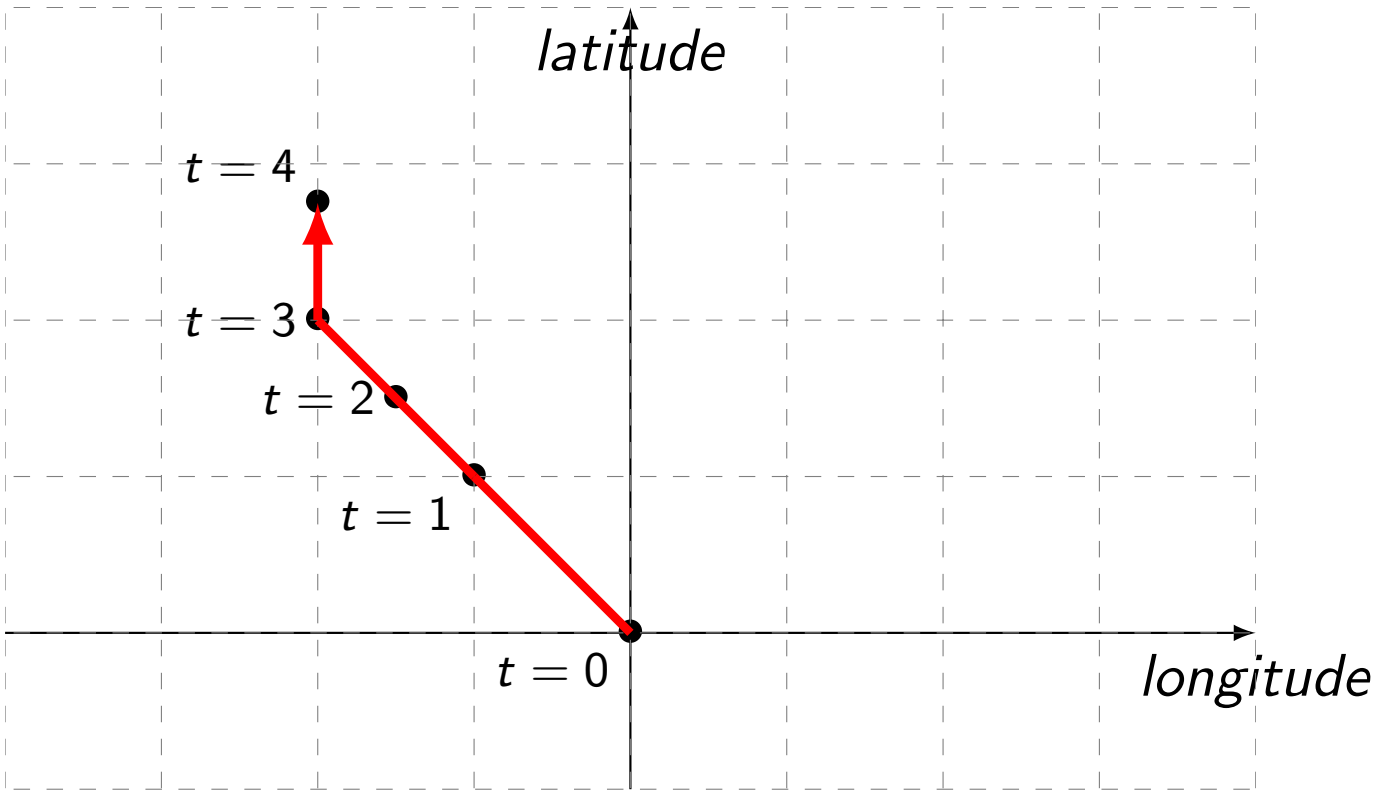
Choix de l'échantillon de données

- Base de données du NHC (National Hurricane Center) 2010-2019


```
AL062012,            FLORENCE,     21,
20120803, 0600,  , LO, 12.2N,  23.1W,  25, 1011,
20120803, 1200,  , LO, 12.5N,  24.5W,  30, 1010,
20120803, 1800,  , TD, 13.0N,  25.9W,  30, 1009,
20120804, 0000,  , TD, 13.5N,  27.3W,  30, 1009,
20120804, 0600,  , TS, 14.1N,  28.6W,  35, 1007,
20120804, 1200,  , TS, 14.7N,  29.9W,  40, 1005,
20120804, 1800,  , TS, 15.3N,  31.2W,  45, 1003,
20120805, 0000,  , TS, 15.8N,  32.5W,  50, 1002,
20120805, 0600,  , TS, 16.1N,  33.8W,  50, 1002,
20120805, 1200,  , TS, 16.2N,  35.0W,  45, 1004,
20120805, 1800,  , TS, 16.2N,  36.2W,  45, 1005,
20120806, 0000,  , TS, 16.2N,  37.3W,  35, 1008,
20120806, 0600,  , TD, 16.3N,  38.4W,  30, 1009,
20120806, 1200,  , LO, 16.4N,  39.6W,  30, 1009,
20120806, 1800,  , LO, 16.5N,  40.9W,  30, 1009,
20120807, 0000,  , LO, 16.7N,  42.5W,  25, 1010,
20120807, 0600,  , LO, 16.9N,  44.3W,  25, 1010,
20120807, 1200,  , LO, 17.2N,  46.3W,  25, 1010,
20120807, 1800,  , LO, 17.6N,  48.5W,  25, 1011,
20120808, 0000,  , LO, 18.0N,  50.7W,  25, 1011,
20120808, 0600,  , LO, 18.4N,  52.9W,  25, 1011,
```

Source : www.nhc.noaa.gov

Trajectoire d'un cyclone



Modélisation

- **Bassin Atlantique Nord :**

$0 \rightarrow 60°$ latitude Nord - $0 \rightarrow 110°$ longitude Ouest

6600	6599	...	...	6492	6491
⋮	⋮			⋮	⋮
220	219	...	...	112	111
110	109	...	...	2	1

- **Relation de localisation :**

$$\boxed{\text{Localisation} = 110 \times \lfloor \text{Latitude} \rfloor + \lfloor \text{Longitude} \rfloor + 1}$$

Réécriture de la base de données

- Transcription de la base en données exploitables :

```
 FLORENCE,     21,
LO, 12.2N,  23.1W,  25, 1011,
LO, 12.5N,  24.5W,  30, 1010,
TD, 13.0N,  25.9W,  30, 1009,
TD, 13.5N,  27.3W,  30, 1009,
TS, 14.1N,  28.6W,  35, 1007,
TS, 14.7N,  29.9W,  40, 1005,
TS, 15.3N,  31.2W,  45, 1003,
TS, 15.8N,  32.5W,  50, 1002,
TS, 16.1N,  33.8W,  50, 1002,
TS, 16.2N,  35.0W,  45, 1004,
```

→

Position	X	Y
1	23,1	12,2
2	24,5	12,5
3	25,9	13
4	27,3	13,5
5	28,6	14,1
6	29,9	14,7
7	31,2	15,3
8	32,5	15,8
9	33,8	16,1
10	35	16,2

→

Localisation
1344
1345
1456
1458
1569
1570
1682
1683
1794
1796

- $\boxed{\text{Localisation} = 110 \times \lfloor Y \rfloor + \lfloor X \rfloor + 1}$

Probabilités de passage d'une localisation à une autre

Cyclone à la localisation l_i

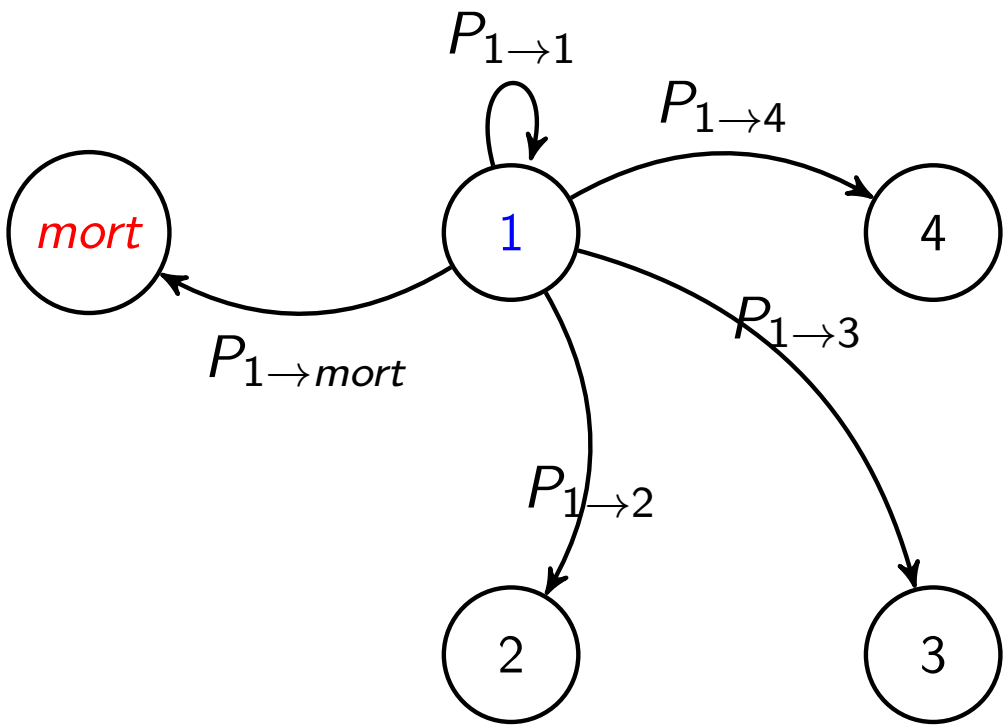
1 - On recense toutes les localisations (m_j) qui suivent l_i dans la base des localisations : $L = [m_1, m_1, ..., m_n]$.

$l_i = 2013$
$L = [2013, 2013, 2014, 2124, 2124]$

2 - On pose : $d = len(L)$. À j fixé, on pose : $c_j =$ nombre d'occurrences de m_j dans L. On a donc : $p_{m_j} = \frac{c_j}{d}$.

$p_{2013} = 2/5 \quad p_{2014} = 1/5 \quad p_{2124} = 2/5$

Probabilités de passage d'une localisation à une autre



- Le modèle utilisé est un modèle de probabilités discrètes.
  $P_{1\to1} + P_{1\to2} + P_{1\to3} + P_{1\to4} + P_{1\to mort} = 1$

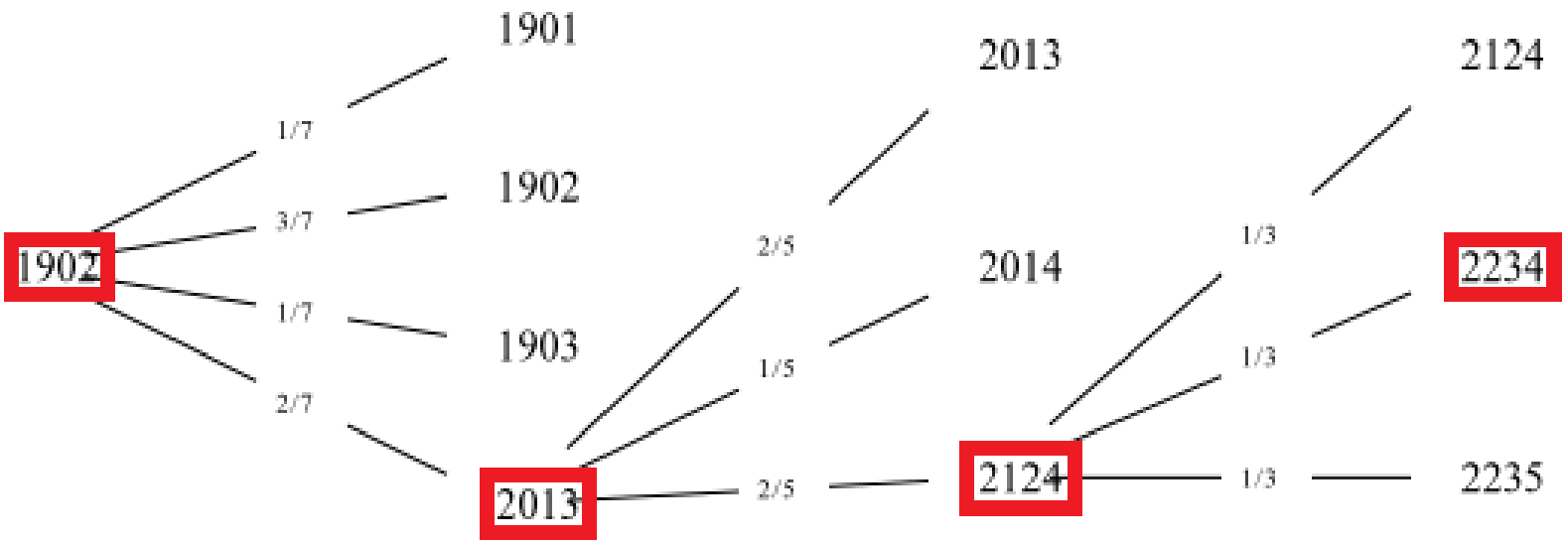
Principe de l'algorithme

- Localisations successives d'un cyclone :

  l_0 = localisation initiale
  $i = 0$
  Tant que $l_i \neq mort$:
    l_{i+1} = position_suivante(l_i)
    i = i+1

  (Les localisations successives sont stockées dans une liste.)

Exemple de trajectoire



2345	2344	2343	2342	2341
2235	2234	2233	2232	2231
2125	2124	2123	2122	2121
2015	2014	2013	2012	2011
1905	1904	1903	1902	1901
1795	1794	1793	1792	1791

Stockage des probabilités

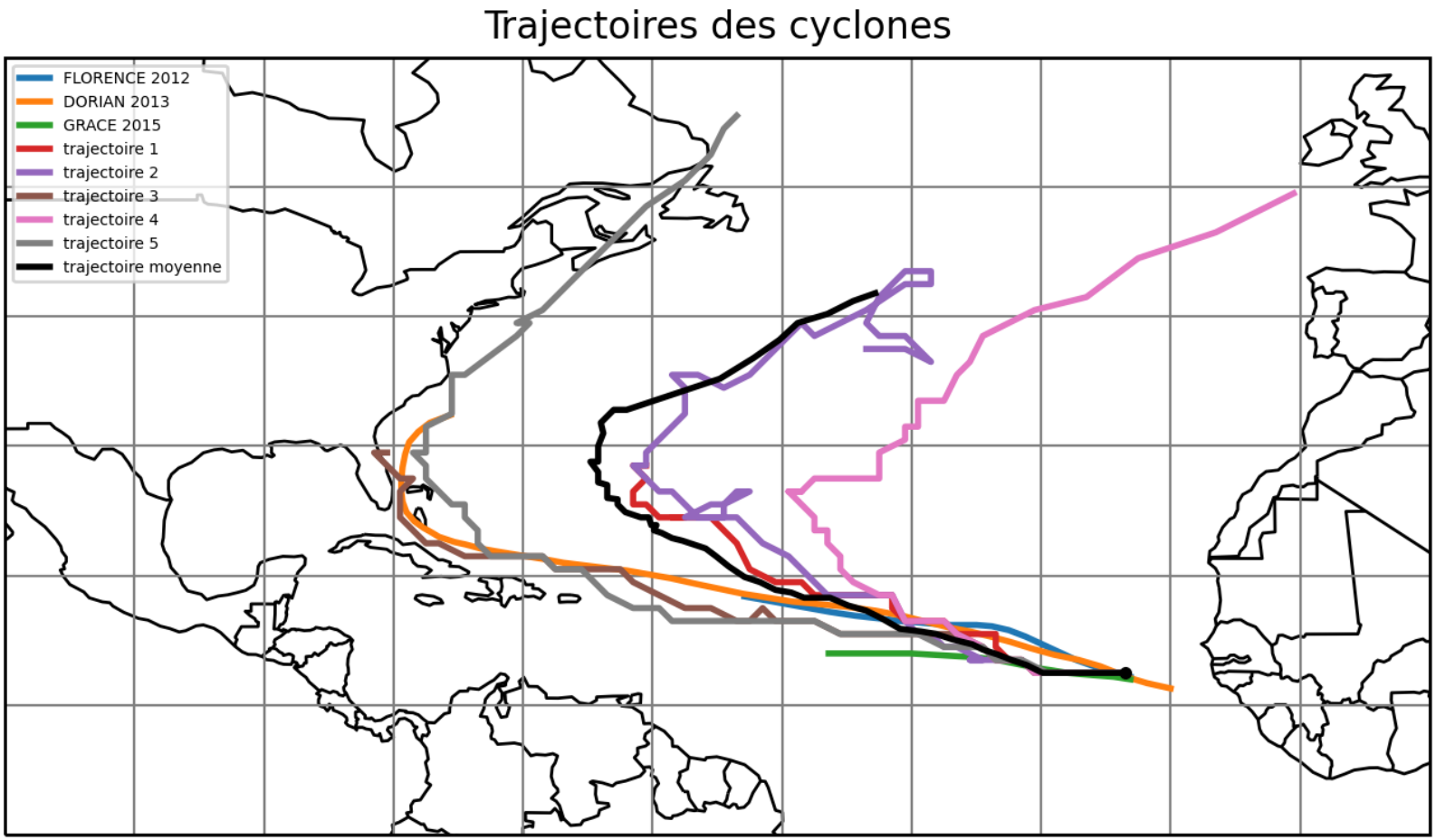
Le calcul des probabilités est stocké de la manière suivante :

```
[1813, 1923, 2033]
[1924, 1925, 2034]
[2036]
[1926]
[1927, 1928]
[2039]
[1929, 1930, 2040]
[1930, 2040]
[1930, 1931, 2040, 2042]
[1823, 1930, 1932, 2042]
[1932, 1933, 2044]
[2043, 2044]
[1935, 2044]
[1935, 1936, 2046]
[1936, 1937, 2046]
[1938, 1939]
[0, 1938, 1939, 2160]
[0, 2049, 2051]
[0]
[0]
[2053]
```

```
[0.3333333333333333, 0.3333333333333333, 0.3333333333333333]
[0.3333333333333333, 0.3333333333333333, 0.3333333333333333]
[1.0]
[1.0]
[0.3333333333333333, 0.6666666666666666]
[1.0]
[0.3333333333333333, 0.3333333333333333, 0.3333333333333333]
[0.5, 0.5]
[0.2857142857142857, 0.42857142857142855, 0.14285714285714285, 0.14285714285714285]
[0.2, 0.2, 0.2, 0.4]
[0.5, 0.25, 0.25]
[0.5, 0.5]
[0.5, 0.5]
[0.4, 0.4, 0.2]
[0.2, 0.2, 0.6]
[0.5, 0.5]
[0.2, 0.2, 0.4, 0.2]
[0.3333333333333333, 0.3333333333333333, 0.3333333333333333]
[1.0]
[1.0]
[1.0]
```

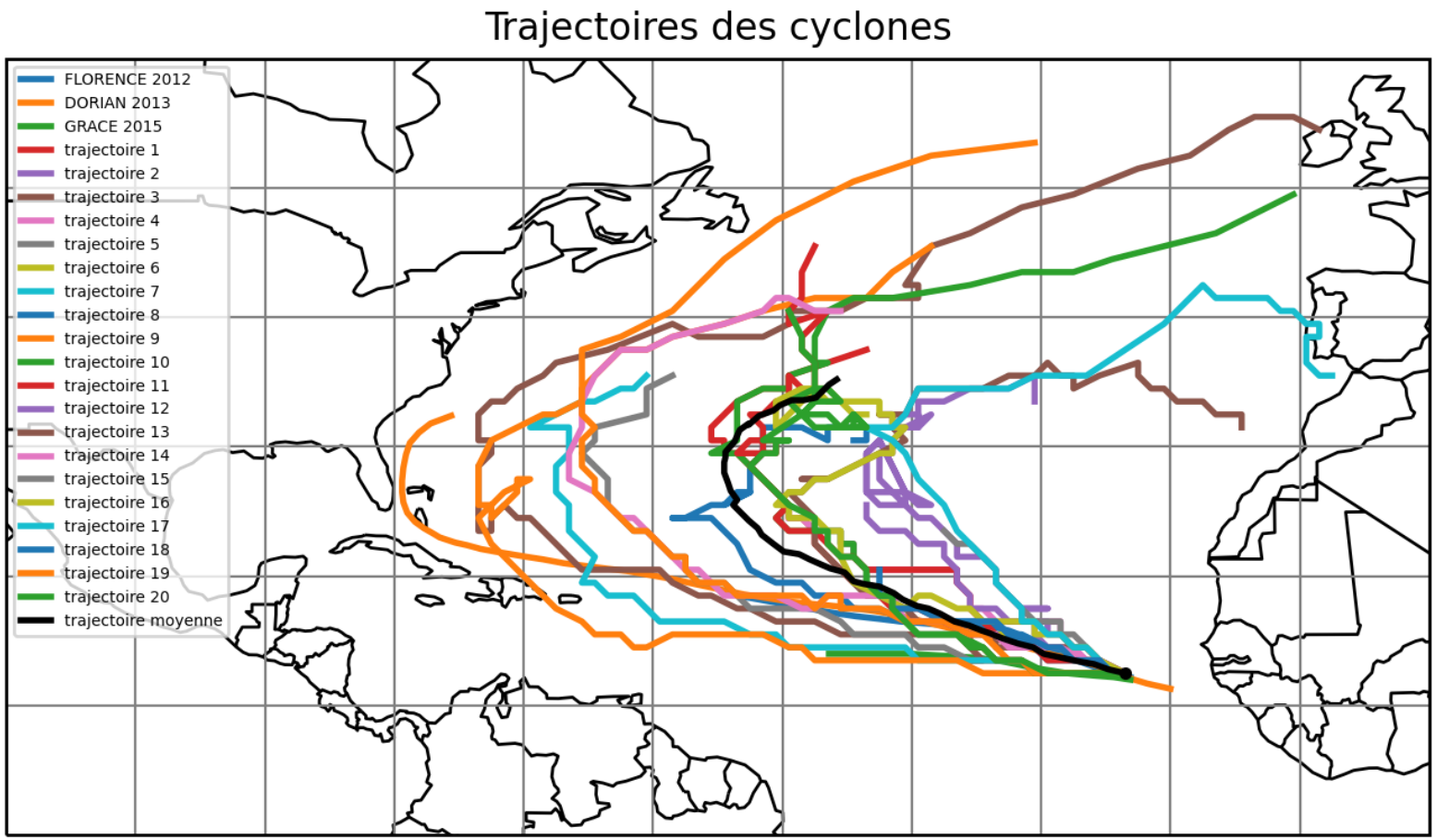
Listes des localisations suivantes possibles et leurs probabilités associées pour les localisations de 1922 à 1942

Tracé des trajectoires



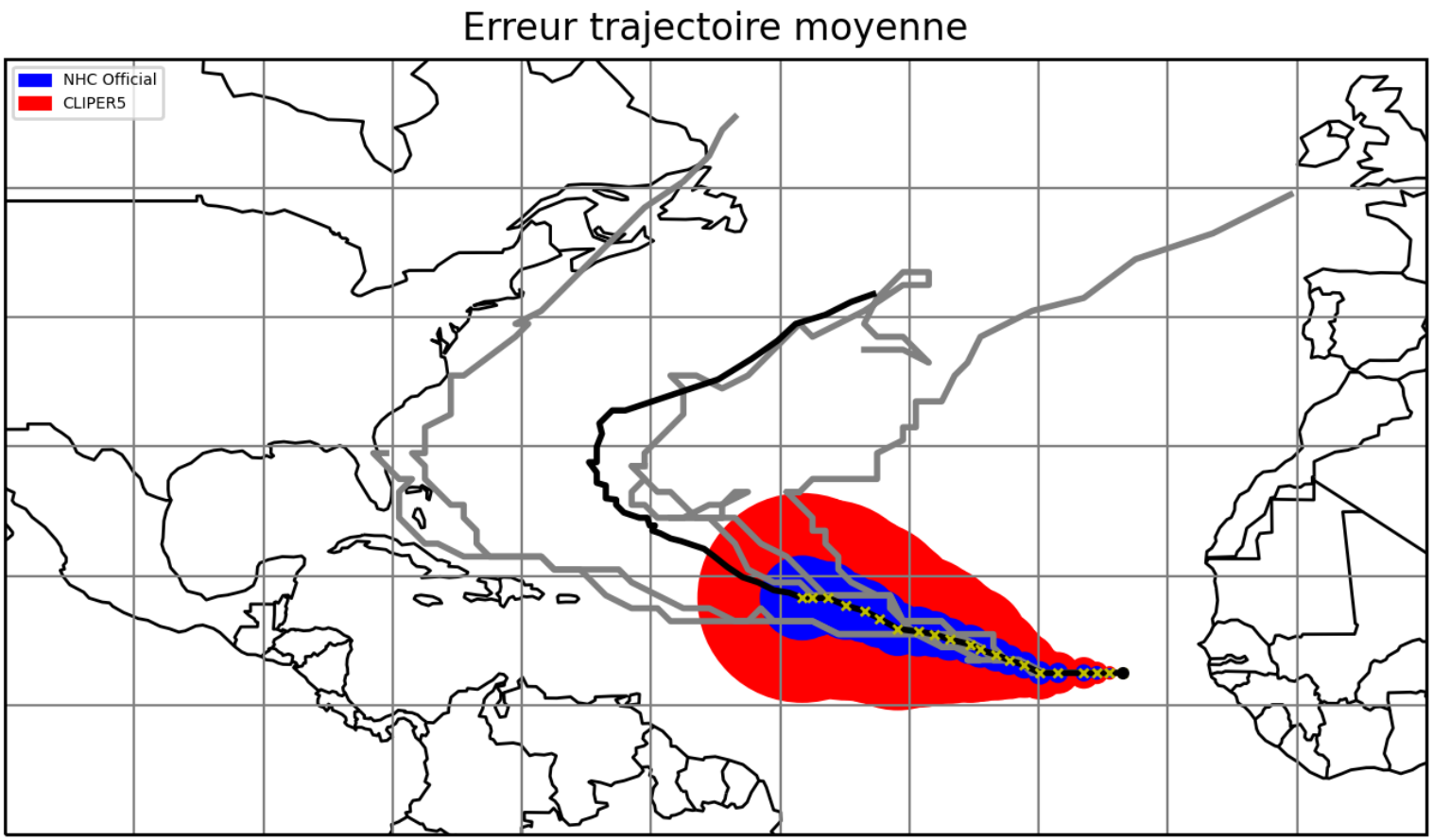
Modélisation par mon programme des trajectoires pour un point de départ donné

Tracé des trajectoires



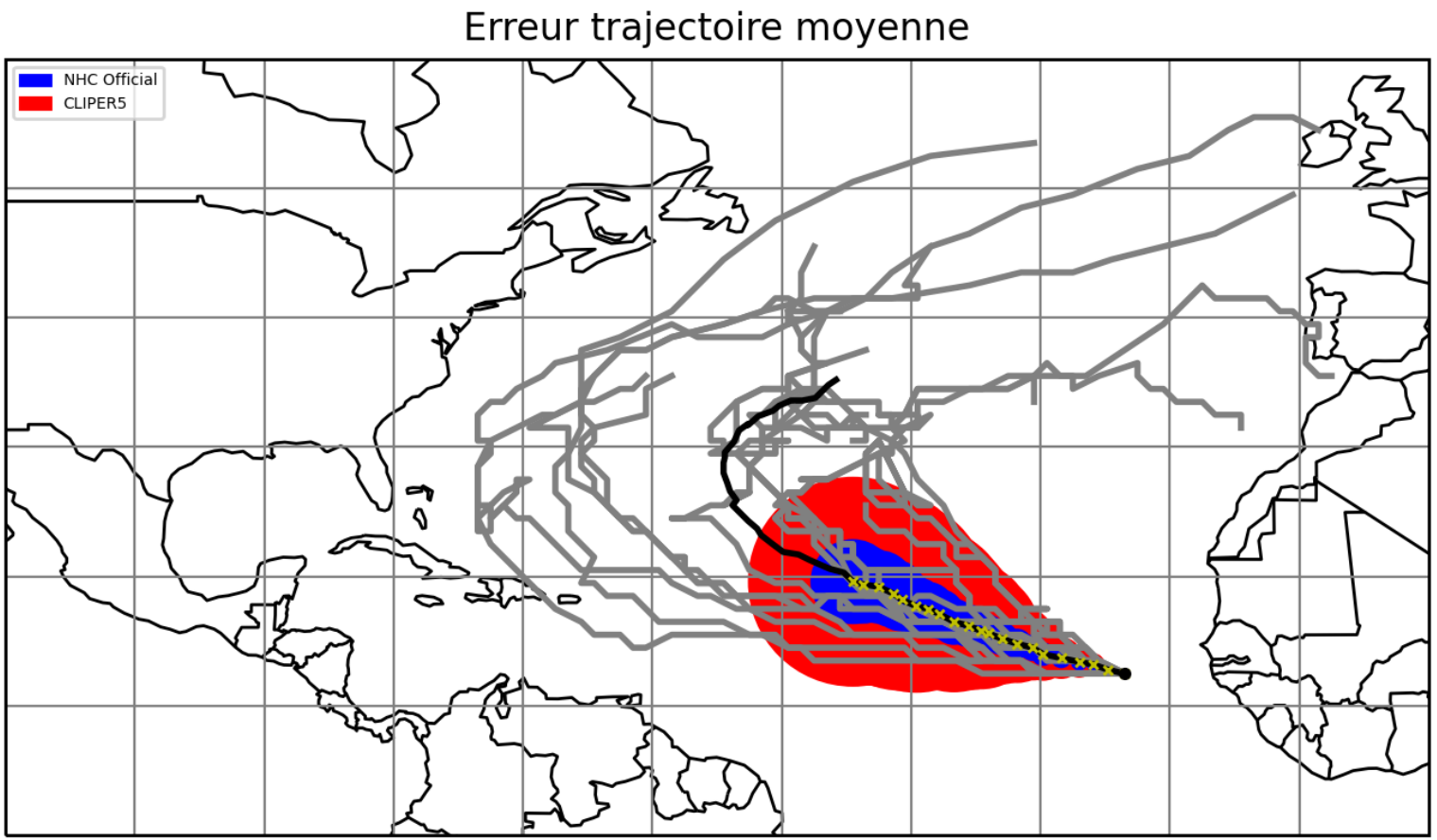
Modélisation par mon programme des trajectoires pour un point de départ donné

Erreur trajectoire



Modélisation par mon programme de l'erreur à partir des données du NHC

Erreur trajectoire



Modélisation par mon programme de l'erreur à partir des données du NHC

Comparaison du modèle avec les modèles existants

- Comparaison de la distance d'erreur simulation/réalité de mon modèle par rapport à ceux existants pour le cyclone Fay (2008) :

Technique de prévision	Erreur en milles nautiques (km)				
	12 heures	24 heures	36 heures	48 heures	72 heures
NHC OFFICIAL	23 (43)	31 (57)	46 (85)	62 (115)	102 (189)
CLIPER5	36 (67)	80 (148)	141 (261)	208 (385)	340 (630)
Mon modèle	108 (200)	190 (351)	291 (539)	378 (700)	457 (847)

Tableau fait à partir des données du NHC

- Influence des données
- Validité du modèle

Calcul de la position suivante

l_i : la localisation d'un cyclone à l'instant t

p_{m_j} : la probabilité pour ce cyclone d'atteindre la localisation m_j à l'instant $t+1$

On pose : $T_{m_j} = \left]\sum_{k=1}^{j-1} p_{m_k}, \sum_{k=1}^{j} p_{m_k}\right]$.

Soit $X \sim U(0,1)$, on a : $P(X \in T_{m_j}) = p_{m_j}$.

On pose : a = random.random() et on trouve $a \in T_{m_h}$.

Cyclone à la localisation $l_{i+1} = m_h$

Tracé des trajectoires

```
from math import *
import random as rd
import numpy as np
import matplotlib.pyplot as plt
import cartopy.crs as ccrs
import cartopy.feature as cfeature
import os #pour la suppression des fichiers créés
import sys #pour tuer le programme quand la localisation initiale est impossible


#longitude=CYCX ; latitude=CYCY


#ligne 48: il faut créer le fichier base avant la première exécution du programme, ensuite mettre cette ligne en commentaire ; lignes 189-90-91: input du nombre de cyclones à tracer (n) et de la localisation initiale (a -> choisir entre loc_init ou input, et mettre la ligne non choisie en commentaire), lignes 17-31 : modifier le chemin d'accès


if os.path.exists('C:\\Users\\maelp\\coord x'): #teste si le fichier existe
    os.remove('C:\\Users\\maelp\\coord x') #supprime le fichier
if os.path.exists('C:\\Users\\maelp\\coord y'):
    os.remove('C:\\Users\\maelp\\coord y')
if os.path.exists('C:\\Users\\maelp\\localisation'):
    os.remove('C:\\Users\\maelp\\localisation')
if os.path.exists('C:\\Users\\maelp\\trajectoires x'):
    os.remove('C:\\Users\\maelp\\trajectoires x')
if os.path.exists('C:\\Users\\maelp\\trajectoires y'):
    os.remove('C:\\Users\\maelp\\trajectoires y')
if os.path.exists('C:\\Users\\maelp\\trajectoire moyenne'):
    os.remove('C:\\Users\\maelp\\trajectoire moyenne')
for i in range(1,1001):
    if os.path.exists('C:\\Users\\maelp\\trajectoire '+str(i)):
        os.remove('C:\\Users\\maelp\\trajectoire '+str(i))


def traitement(t,n): #fonction récupérant les coordonnées pour un cyclone de la base de données (t) et la ligne de début du prochain cyclone (n)
    CX,NCX,CY = [],[],[]
    for i in range (n,len(t)):
        if t[i][0] == "A" : #correspond à un nouveau cyclone à traiter
            return CX,NCX,CY,i
        else:
            w=float(t[i][30]+t[i][31]+t[i][32]+t[i][33]+t[i][34]) #coordonnées ouest
            n=float(t[i][23]+t[i][24]+t[i][25]+t[i][26]) #coordonnées nord
            CX.append(w) #ajoute la nouvelle coordonnée longitude au tableau (positive)
            NCX.append(-w) #ajoute la nouvelle coordonnée longitude au tableau (N = négative pour l'utilisation de cartopy)
            CY.append(n) #ajoute la nouvelle coordonnée latitude au tableau
    return CX,NCX,CY,i
```

Tracé des trajectoires

```
46
47
48  #base = open("base","w") #crée le fichier base dans Ce PC -> C: -> Utilisateurs -> maelp
49
50  base=open("base","r") #ouvre le fichier txt contenant la base de données normalisée à exploiter
51  m=base.readlines() #crée un tableau contenant les lignes de la base en chaines de caractères
52  base.close()
53
54
55  #cartopy
56  latitude_min=0
57  latitude_max=60
58  longitude_min=-110
59  longitude_max=0
60
61  ax = plt.axes(projection=ccrs.PlateCarree(),autoscale_on=False,xlim=(longitude_min,longitude_max),ylim=(latitude_min,latitude_max))
62  ax.add_feature(cfeature.BORDERS)
63  ax.add_feature(cfeature.COASTLINE)
64
65
66  #quadrillage
67  for i in range(10,60,10):
68      plt.plot([-110,0],[i,i],'-',color='grey',linewidth=0.8)
69
70  for j in range(10,110,10):
71      plt.plot([-j,-j],[0,60],'-',color='grey',linewidth=0.8)
72
73
74  CYCX,CYCY=[],[]
75  c=1 #numéro ligne début cyclone
76  while c<len(m):
77      a,x,y,c0 = traitement(m,c) #a <- CX ; x <- NCX ; y <- CY ; c0 <- i = ligne nom cyclone suivant
78      cyclonex=open("coord x","a") #crée un fichier avec les listes des coordonnées x de tous les cyclones
79      cyclonex.write(str(a)+"\n")
80      cyclonex.close()
81      CYCX.append(a)
82      cycloney=open("coord y","a") #crée un fichier avec les listes des coordonnées y de tous les cyclones
83      cycloney.write(str(y)+"\n")
84      cycloney.close()
85      CYCY.append(y)
86      c=c0+1 #numéro ligne début cyclone suivant
87
88
89  l=[] #base des localisations
90
```

Tracé des trajectoires

```
91  for i in range (len(CYCY)): #la=CYCY et lo=CYCX sont de longueur égale
92      L1 = []
93      for j in range (len(CYCY[i])):
94          l1 = 110 * floor(CYCY[i][j]) + 1 #110 pour 110 degrés de longitude ; floor(float(a)) = partie entière de la latitude ; +1 pour la
    formule qui crée 6600 (110x60) cases numérotées de 1 à 6600
95          L1.append(l1)
96      L2 = []
97      c = 0 #indice des éléments de L1
98      for k in range (len(CYCX[i])):
99          l2 = floor(CYCX[i][k]) + L1[c] #floor(float(b)) pour la longitude ; la formule crée le tableau de 6600 cases ; formule : localisation =
    110 x floor(latitude) + floor(longitude) + 1
100         L2.append(l2)
101         c += 1
102     loc=open("localisation","a") #crée un fichier avec la localisation de tous les cyclones
103     loc.write(str(L2)+"\n")
104     loc.close()
105     l.append(L2)
106
107
108 def loc_init(t): #renvoie la localisation initiale du cyclone
109     n = rd.randint(0,len(t)-1) #renvoie un entier aléatoire entre 0 et len(t)-1
110     return t[n][0]
111
112
113 bd1=np.zeros((6601,1),dtype = list) #base de données des localisations suivantes possibles
114 bd2=np.zeros((6601,1),dtype = list) #base de données des probabilités associés
115
```

Tracé des trajectoires

```
116 for a in range(6601):
117     L = [] #localisations
118     for i in range (len(l)):
119         for j in range (len(l[i])):
120             if l[i][j] == a and len(l[i]) > j+1 :
121                 L.append(l[i][j+1])
122             elif l[i][j] == a and j == len(l[i])-1 :
123                 L.append(0) #0 correspond à l'état mort (plus de cyclone)
124             elif i == len(l)-1 and j == len(l[i])-1 and l[i][j] != a and L == [] :
125                 L.append(0)
126     P = [] #probabilités de transition
127     L.sort() #trie la liste L
128     M = [] #liste L sans doublons
129     c = 1 #compteur du nombre de répétitions d'une même localisation
130     for i in range (len(L)):
131         if i != len(L)-1 and L[i] != L[i+1] :
132             P.append(c/len(L)) #c/len(L) = proba
133             M.append(L[i])
134             c = 1
135         elif i == len(L)-1 :
136             P.append(c/len(L))
137             M.append(L[i])
138         else:
139             c = c+1
140     bd1[a,0]=M
141     bd2[a,0]=P
142
143
144 def position_suivante(a): #fonction qui donne la localisation suivante du cyclone ; a = localisation présente
145     M = bd1[a,0]
146     P = bd2[a,0]
147     n = rd.random() #renvoie un réel aléatoire entre 0 et 1
148     s = P[0] #somme des probabilités qui vaut 1 au maximum
149     for i in range (len(P)): #P et M ont la même longueur
150         if n <= s:
151             return M[i]
152         else:
153             s = s + P[i+1]
154
155
156 def trajectoire_cyclone(a): #fonction qui renvoie la liste des localisations successives du cyclone ; a = localisation initiale
157     m = a
158     L = [] #liste des localisations successives du cyclone
159     while m != 0:
160         L.append(m)
161         m = position_suivante(m)
162     return L
```

Tracé des trajectoires

```
163
164
165 def liste_moins(L): #applique un x(-1) à tous les éléments de la liste
166     M=[]
167     for i in range(len(L)):
168         M.append(-L[i])
169     return M
170
171
172 def name(t,n): #fonction récupérant le nom et la date pour un cyclone de la base de données (t), n = (numéro du cyclone)-1 (le -1 permet de
    commencer à 0 au 1er cyclone rencontré pour avoir la correspondance avec les indices d'une liste)
173     a = -1 #compte le nombre de cyclones rencontrés, -1 pour 0 cyclone rencontré (même raison que pour n)
174     name = ""
175     for i in range(len(t)):
176         if t[i][0] == "A": #correspond à un nouveau cyclone à traiter
177             a += 1 #on augmente de 1 le nombre de cyclones rencontrés
178             if a == n:
179                 c = t[i][4]+t[i][5]+t[i][6]+t[i][7] #année du cyclone
180                 j = 9 #premier blanc de la ligne du nom
181                 while t[i][j] != ",": #boucle qui donne le nom du cyclone
182                     if t[i][j] != " ":
183                         name = name + t[i][j]
184                     j+=1
185                 name = name + " " + c
186                 return name
187
188
189 n = int(input("Rentrer le nombre de cyclones à tracer : ")) #nombre de cyclones à tracer
190 #a = int(input("Localisation initiale du cyclone : ")) #la localisation initiale est un nombre compris entre 1 et 6600 obtenu avec la formule
    du calcul de la localisation
191 a = loc_init(l) #l = base des localisations
192
193
194 M = [] #liste des indices de ligne commençant par a
195 for i in range(len(l)):
196     for j in range(len(l[i])):
197         if l[i][j] == a:
198             if (i in M) == False:
199                 M.append(i)
200 if M == []:
201     print("localisation initiale impossible") #la localisation n'est pas présente dans la base de données
202     sys.exit() #tue le programme
203 else:
204     for i in range(len(M)):
205         nom = name(m,M[i]) #m = base de données
206         X = liste_moins(CYCX[M[i]])
207         Y = CYCY[M[i]]
208         plt.plot(X,Y,'-',linewidth=2,label=nom) #trace tous les cyclones ayant commencé à la localisation a
```

Tracé des trajectoires

```
209
210
211 traj=[]
212
213 for k in range(1,n+1):
214     L = trajectoire_cyclone(a)
215     X = []
216     Y = []
217     A = []
218     nom="trajectoire "+str(k)
219     for i in range (len(L)):
220         x = ((L[i]-1)%110) + 0.5 #-1 pour compenser le +1 de la formule ; %110 pour accéder au reste = longitude ; +0.5 pour centrer au milieu
    d'un carré de 1°x1°
221         y = ((L[i]-1)//110) + 0.5 #-1 pour compenser le +1 de la formule ; //110 pour accéder au quotient = latitude ; +0.5 pour centrer
222         X.append(-x) #-x pour l'utilisation de cartopy
223         Y.append(y)
224         A.append(x)
225     traj.append([nom,A,Y])
226     cyclone=open(nom,"a") #crée un fichier avec les listes des coordonnées
227     cyclone.write("X:"+str(traj[k-1][1])+"\n")
228     cyclone.write("Y:"+str(traj[k-1][2])+"\n")
229     cyclone.close()
230     plt.plot(X,Y,'-',linewidth=2,label=nom)
231
232
233 for i in range(1,n+1):
234     t1 = traj[i-1][1]
235     t2 = traj[i-1][2]
236     trajx = open("trajectoires x","a") #crée le fichier avec la liste des abscisses de tous les cyclones modélisés
237     trajx.write(str(t1)+"\n")
238     trajx.close()
239     trajy = open("trajectoires y","a") #crée le fichier avec la liste des ordonnées de tous les cyclones modélisés
240     trajy.write(str(t2)+"\n")
241     trajy.close()
242
243
244 def moyenne(t): #fonction qui calcule la moyenne de la longueur des lignes du tableau t, ici tableau des cyclones modélisés
245     s = 0 #somme des longueurs des lignes
246     for i in range(len(t)):
247         s += len(t[i][1])
248     s = floor(s/len(t)) #floor = partie entière car la longueur d'une liste est un entier
249     return s #renvoie la moyenne de la longueur des lignes du tableau t
250
251
```

Tracé des trajectoires

```
252 def longitude(t): #fonction qui calcule la longitude moyenne (= moyenne des abscisses) des cyclones modélisés
253     mX=[]
254     n = moyenne(t)
255     for j in range(0,n): #on va effetuer le calcul des moyennes des abscisses sur la longueur moyenne des lignes
256         s = 0 #somme des abscisses pour un indice j de liste fixé
257         c = 0 #compteur du nombre de listes ayant une longueur strictement supérieur à j
258         for i in range(0,len(t)): #prend en compte tous les cyclones
259             if j < len(t[i][1]): #il faut que l'indice j soit présent dans la liste
260                 s +=t[i][1][j]
261                 c += 1
262         mX.append(round((s/c),1)) #on arrondi à une décimale la moyenne des abscisses associé à l'indice j pour correspondre avec les autres
    valeurs qui ont seulement un seul chiffre après la virgule
263     trajx = open("trajectoires x","a")
264     trajx.write(str(mX)+"\n") #on ajoute la liste des abscisses de la meilleure trajectoire au fichier contenant toutes la liste des abscisses
    de tous les cyclones
265     trajx.close()
266     mt = open("trajectoire moyenne","a") #ouvre le fichier meilleure trajectoire
267     mt.write("X:"+str(mX)+"\n") #on ajoute la liste des abscisses de la meilleure trajectoire au fichier "meilleure trajectoire"
268     mt.close()
269     return mX #renvoie le tableau contenant les listes des abscisses de tous les cyclones (meilleure trajectoire comprise)
270
271
272 def latitude(t): #fonction qui calcule la latitude moyenne (= moyenne des ordonnées) des cyclones modélisés, note idem fonction longitude en
    remplaçant abscisses par ordonnées et longitude par latitude
273     mY=[]
274     n = moyenne(t)
275     for j in range(0,n):
276         s,c = 0,0
277         for i in range(0,len(t)):
278             if j < len(t[i][2]):
279                 s += t[i][2][j]
280                 c += 1
281         mY.append(round((s/c),1))
282     trajy = open("trajectoires y","a")
283     trajy.write(str(mY)+"\n")
284     trajy.close()
285     mt = open("trajectoire moyenne","a")
286     mt.write("Y:"+str(mY)+"\n")
287     mt.close()
288     return mY #renvoie le tableau contenant les listes des ordonnées de tous les cyclones (meilleure trajectoire comprise)
289
290
291 X = liste_moins(longitude(traj)) #tableau contenant les listes des abscisses de tous les cyclones (meilleure trajectoire comprise)
292 Y = latitude(traj) #tableau contenant les listes des ordonnées de tous les cyclones (meilleure trajectoire comprise)
293
```

Tracé des trajectoires

```
294 plt.plot(X,Y,'-',linewidth=2,label="trajectoire moyenne",color="k")
295
296
297 plt.plot(-traj[0][1][0],traj[0][2][0],marker="o",color="black",markersize=3) #point de départ
298
299 plt.legend(prop={'size':5},loc = 2) #légende avec taille lettres et position
300
301 plt.title('Trajectoires des cyclones')
302
303 ax.set_aspect('equal')
304
305 plt.show()
```

Erreur trajectoire

```
from math import *
import numpy as np
import matplotlib.pyplot as plt
import cartopy.crs as ccrs
import cartopy.feature as cfeature
import matplotlib.patches as mpatches #pour la légende en fonctions des couleurs


#ligne 59 : input du n° de la trajectoire dont on veut connaître l'erreur (0 pour la trajectoire moyenne, n° de la trajectoire modélisée pour les autres)


#cartopy
latitude_min=0
latitude_max=60
longitude_min=-110
longitude_max=0

ax = plt.axes(projection=ccrs.PlateCarree(),autoscale_on=False,xlim=(longitude_min,longitude_max),ylim=(latitude_min,latitude_max))


#quadrillage
for i in range(10,60,10):
    plt.plot([-110,0],[i,i],'-',color='grey',linewidth=0.8)

for j in range(10,110,10):
    plt.plot([-j,-j],[0,60],'-',color='grey',linewidth=0.8)


def liste_moins(L): #applique un x(-1) à tous les éléments de la liste
    M=[]
    for i in range(len(L)):
        M.append(-L[i])
    return M


def poly_L(L,M): #interpolation de Lagrange, L: liste des abscisses, M: liste des ordonnées
    n = len(L)
    s = 0 #somme des produits multipliés par les ordonnées
    X = np.poly1d([1,0]) #polynôme égal à 1X+0=X
    for j in range(0,n): #réalise la somme
        a = 1
        for i in range(0,n): #réalise le produit pour i =/= j
            if i != j:
                a *= (X-L[i])/(L[j]-L[i])
        s += M[j]*a
    return s #retourne le polynôme de Lagrange

```

Erreur trajectoire

```
48
49  trajx = open("trajectoires x","r") #ouvre le fichier contenant la liste des abscisses de tous les cyclones modélisés
50  tx = trajx.readlines() #tableau contenant les listes des abscisses de tous les cyclones
51  trajx.close()
52
53
54  trajy = open("trajectoires y","r") #ouvre le fichier contenant la liste des ordonnées de tous les cyclones modélisés
55  ty = trajy.readlines() #tableau contenant les listes des ordonnées de tous les cyclones
56  trajy.close()
57
58
59  a = int(input("Numéro trajectoire : ")) #numéro de la trajectoire dont on veut connaître l'erreur (0 pour la trajectoire moyenne, n° de la
    trajectoire modélisée pour les autres)
60  X=[]
61  Y=[]
62  if a == 0:
63      nom = 'trajectoire moyenne' #nom du cyclone
64      X = liste_moins(eval(tx[len(tx)-1])) #liste des abscisses du cyclone
65      Y = eval(ty[len(ty)-1]) #liste des ordonnées du cyclone
66  else:
67      nom = 'trajectoire '+str(a) #nom du cyclone
68      X = liste_moins(eval(tx[a-1])) #liste des abscisses du cyclone
69      Y = eval(ty[a-1]) #liste des ordonnées du cyclone
70
71
72  L = [0,12,24,36,48,72,96,120] #abscisses : pris par rapport au graphe de la NHC (National Hurricane Center), en heures
73  M1 = [0,25,40,55,70,100,140,190] #ordonnées : pris par rapport au graphe de la NHC (National Hurricane Center), en miles
74  P1 = poly_L(L,M1) #calcul du polynôme de Lagrange modélisant la fonction du graphe (entre 2010 et 2019)
75  M2 = [0,45,90,145,205,320,415,480] #valeurs pour CLIPER5
76  P2 = poly_L(L,M2)
77
78
79  for i in range(len(X)-1,-1,-1): #dégressif car pour i grand, erreur grande -> permet de voir l'erreur pour les petits i
80      if i <= 20: #en dessous de 20, respect du graphe
81          r = (P2(6*i)*1.852*180)/(6371*np.pi) #6xi car entre chaque position il y 6 heures d'intervalles (en accord avec la base du NHC) x 1.852
    pour la conversion miles vers km x (180/(6371*pi)) pour la conversion en ° (6371 km = rayon de la Terre)
82          circle1 = plt.Circle((X[i], Y[i]), r, color='r')
83          ax.add_artist(circle1)
84
85
86  for i in range(len(X)-1,-1,-1): #dégressif car pour i grand, erreur grande -> permet de voir l'erreur pour les petits i
87      if i <= 20: #en dessous de 20, respect du graphe
88          r = (P1(6*i)*1.852*180)/(6371*np.pi) #6xi car entre chaque position il y 6 heures d'intervalles (en accord avec la base du NHC) x 1.852
    pour la conversion miles vers km x (180/(6371*pi)) pour la conversion en ° (6371 km = rayon de la Terre)
89          circle1 = plt.Circle((X[i], Y[i]), r, color='b')
90          ax.add_artist(circle1)
91
92
```

Erreur trajectoire

```
93  ax.add_feature(cfeature.BORDERS)
94  ax.add_feature(cfeature.COASTLINE)
95
96
97  if a == 0:
98      X1 = []
99      Y1 = []
100     for i in range(len(tx)-1): #trace les autres trajectoires
101         X1 = liste_moins(eval(tx[i]))
102         Y1 = eval(ty[i])
103         plt.plot(X1,Y1,'-',linewidth=2,color="grey")
104     plt.plot(X,Y,'-',linewidth=2,label="trajectoire moyenne",color="k") #trace la meilleure trajectoire
105 else:
106     X1 = []
107     Y1 = []
108     for i in range(len(tx)-1): #trace les autres trajectoires
109         if i != a-1:
110             X1 = liste_moins(eval(tx[i]))
111             Y1 = eval(ty[i])
112             plt.plot(X1,Y1,'-',linewidth=2,color="grey")
113     X2 = liste_moins(eval(tx[len(tx)-1]))
114     Y2 = eval(ty[len(ty)-1])
115     plt.plot(X2,Y2,'-',linewidth=2,label="trajectoire moyenne",color="maroon") #trace la meilleure trajectoire, permet d'avoir la trajectoire
    moyenne en couleur
116     plt.plot(X,Y,'-',linewidth=2,label="trajectoire "+str(a),color="k") #trace la trajectoire n°a
117
118
119 for i in range(21):
120     plt.plot(X[i],Y[i],marker="x",color="y",markersize=3)
121
122
123 #définition de la légende
124 a = mpatches.Patch(color='b', label='NHC Official')
125 b = mpatches.Patch(color='r', label='CLIPER5')
126
127 plt.plot(X[0],Y[0],marker="o",color="black",markersize=3) #point de départ
128
129 plt.legend(handles=[a,b],prop={'size':5},loc = 2) #légende spécifique définie au dessus avec taille lettres et position
130
131 plt.title('Erreur '+nom)
132
133 ax.set_aspect('equal')
134
135 plt.show()
```