INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES ROUEN

Rapport au 27/09
PFE

PFE : Reconstruction d'un champ de vents

Étudiant :
Henri LEROY

Enseignant-responsable du projet :
A. TONNOIR

Institut National des Sciences Appliquées de Rouen
Département Sciences et Techniques Pour l'Ingénieur
685 Avenue de l'Université BP 08 - 76801 Saint-Étienne-du-Rouvray
Tél : +33 2 32 95 66 21 - Fax : +33 2 32 95 66 31

Résumé

Dans le contexte de l'éolien, la modélisation des champs de vent est un enjeu important pour savoir notamment où placer les éoliennes afin d'optimiser leur productivité. Une problématique commune est alors la reconstruction du champ de vent sur une zone étant donné des mesures ponctuelles de ce dernier. Plusieurs approches sont possibles pour aborder ce problème. Une idée simple consiste à exploiter des méthodes d'interpolation multi-variée. Une autre approche revient à résoudre un problème de minimisation sous contrainte, permettant de prendre en compte les propriétés physiques. Le but de ce PFE est de s'initier aux méthodes numériques pour cette problématique.

Table des matières

Première partie

Récupération et visualisation des données

Dans un premier temps, l'objectif sera de récupérer des données sur les vents afin d'avoir une base de travail. Nous essayerons également de visualiser ses données afin de se familiariser avec le sujet.

1 Choix de la base de données

1.1 Variables à considérer

Plusieurs options sont disponibles pour accéder à des données sur les vents. Avant de commencer notre travail, nous devons donc choisir un moyen d'obtenir les données.

Tout d'abord, il est intéressant de réfléchir aux variables à avoir sur les vents pour notre travail. Dans notre cas, 2 valeurs seront nécessaires :

1. La **direction** du vent.
2. La **vitesse** du vent.

Avec ces 2 valeurs, nous pourrons représenter le champ de vecteurs et commencer notre travail d'interpolation. On doit donc trouver une base de données permettant d'accéder à ces champs. De plus, on doit penser au pas de temps et d'espace utilisé. En effet, l'espace sera considéré comme une grille en 2D avec des points espacés d'une certaine longitude et latitude. Un maillage assez fin permettra d'obtenir une simulation plus réaliste. Enfin, les données obtenues devront être espacées d'une certaine durée. Afin d'obtenir l'évolution des vents sur une journée, un mois ou une année, nous devrons avoir l'évolution de ces mesures du vent dans le temps. Bien que cette option ne soit pas indispensable, elle pourrait être intéressante à observer dans le but de trouver le point optimal pour placer une éolienne. Il faudrait considérer l'évolution du vent au cours du temps pour optimiser la position de l'éolienne.

1.2 Choix d'une base de données

Avant d'interpoler nos données, nous devons trouver une source permettant d'obtenir des données réelles ayant les caractéristiques présentées précédemment.

- Dans un premier temps, j'ai essayé d'utiliser la base de données WRDB (Wind Resource Database) https://wrdb.nrel.gov. Sur cette base de données, on peut obtenir la vitesse et la direction du vent à plusieurs altitudes (de 10m à 160m de hauteur). Ces informations pourront être utiles pour effectuer une interpolation en 3D. On pourracommencer par fixer une altitude avant d'améliorer notre projet en ajoutant une dimension. Pour la localisation géographique, plusieurs pays sont disponibles. Pour commencer, j'ai choisi de travailler sur la zone Ukraine. Cette zone m'a semblé assez pertinente dans la mesure où les données sont disponibles de 2000 à 2022 soit sur une large période de temps. De plus, la résolution patiale (pas dans l'espace) est de 2km et la résolution temporelle (pas de temps) est de 5min ce qui me paraît assez bas comme valeur. La simulation obtenue pourra donc être réaliste. De plus, la zone étudiée (en bleu sur la figure ci-après) possède un profil topologique varié avec des chaînes montagneuses et la mer Noire ce qui pourrait permettre, d'après moi, une étude plus intéressante des vents.

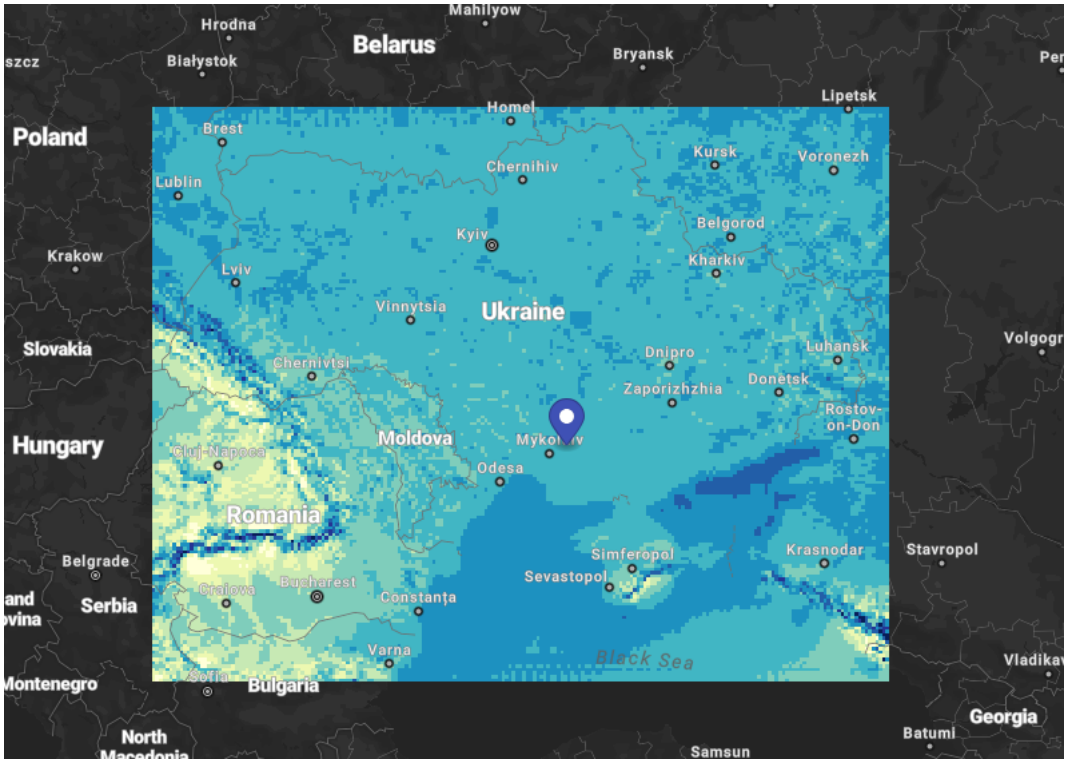

FIGURE 1 – Définition de la zone géographique étudiée (en bleu)

Cependant, le principal inconvénient de cette base de données réside dans la manière de récupérer les données. En effet, on doit indiquer une latitude et une longitude avant d'obtenir les données en rapport avec ce point. De plus, il faut indiquer une adresse mail ce qui pourrait rendre plus compliqué l'obtention automatique des données. La création du jeu de données entier serait donc assez contraignant ce qui m'a poussé à m'intéresser à d'autres bases de données disponibles. J'ai donc décidé de choisir une autre base de données plus adaptée.

- Par la suite, j'ai donc essayé d'utiliser la base de données de Global Wind Atlas https://globalwindatlas.info/en/.Après avoir réussi à recevoir le fichier de données, j'ai dû le convertir pour passer d'un fichier .tif à un jeu de données compréhensible. Malheureusement, après avoir réussi à convertir ce fichier, je me suis rendu compte que la base de données ne contenait aucune information sur la direction du vent. J'ai donc pu observer la vitesse du vent partout en France pour plusieurs altitudes, sans avoir d'indication sur sa direction. Par conséquent, j'ai dû trouver une autre base de données à exploiter.

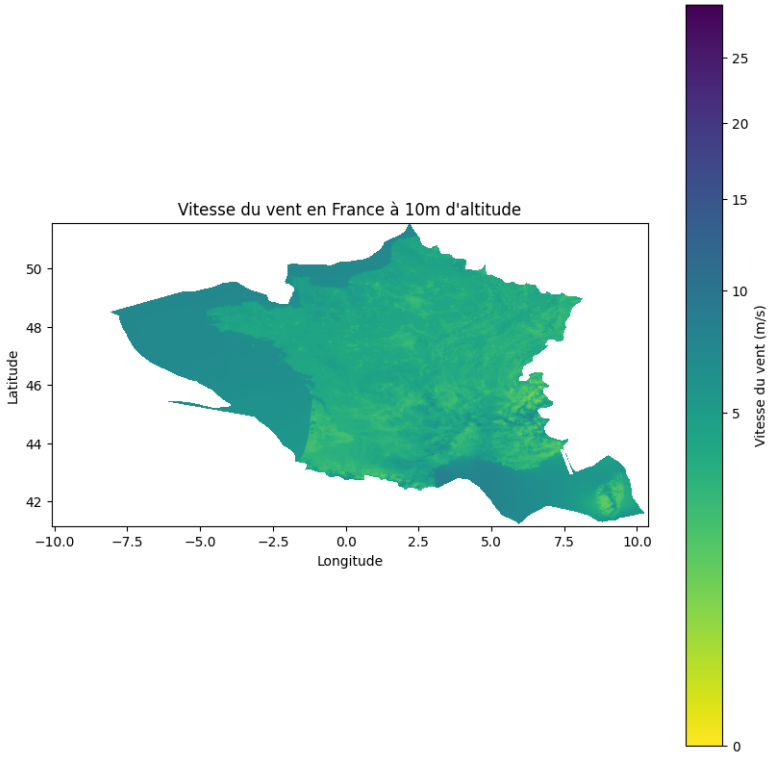

FIGURE 2 – Projection des vitesses de vents en France via Global Wind Atlas

Remarque. Seule la vitesse des vents est indiquée, on ne peut pas avoir sa direction ce qui limite son utilité dans notre projet.

- Après avoir tenté d'utiliser 2 bases de données différentes, en vain, j'ai décidé d'utiliser la base de données stormglass.io https://stormglass.io/wind-api/ possédant plusieurs informations climatiques utiles. Cette API est bien documentée et complète ce qui devrait nous permettre de l'utiliser. L'inconvénient majeur de cette api est le nombre de requêtes disponibles par jour. Dans sa version gratuite, seules 10 requêtes sont autorisées par jour. De plus, cette base de données n'accepte que les requêtes à une latitude et une longitude donnée. Il faudrait donc effectuer plusieurs requêtes pour avoir une vision globale d'un terrain. Cette contrainte nous empêche d'avoir facilement des données. J'ai donc dû poursuivre ma recherche d'une base de données idéale.

- J'ai finalement trouvé une base de données exploitable pour notre projet sur le site https://open-meteo.com/en/docs/historical-weather-api en exploitant l'API « Historical Weather ». Cette API me permet d'obtenir la direction et la vitesse du vent (en km/h) à 10m et à 100m d'altitude mais aussi bien d'autres variables (température, humidité...). De plus, on peut obtenir ces informations pour une liste de coordonnées spécifiées ce qui nous permet de quadriller une zone en connaissant sa latitude et sa longitude. On a également une variable temporelle : on obtient les informations demandées pour chaque heure de la journée sur une plage temporelle donnée. On peut également passer une liste de coordonnées en entrée pour obtenir les variables étudiées sur plusieurs points. J'ai donc choisi cette base de données pour ce projet. Toutefois, le nombre de requêtes par jour est limité à 600. Je ne pense pas que cette valeur soit limitante mais il faudra éviter d'effectuer des requêtes en boucle pour éviter tout désagrément.

2 Première étude du jeu de données.

Après avoir choisi le jeu de données qui sera étudié dans la première partie de ce projet, nous allons effectuer une première étude afin de se familiariser avec lui.

Pour commencer, j'ai essayé d'obtenir le champ de vent autour de la Corse. Ce territoire m'a semblé assez pertinent à étudier. On y trouve du relief et la Méditerranée ce qui permet d'obtenir des vents assez forts sur certaines parties de l'île (*voir partie Climat* https://fr.wikipedia.org/wiki/Corse).

Le code discuté ci-dessous est fourni. Je vais expliquer les parties les plus importantes du code permettant d'afficher la carte des vents de la Corse.

2.1 Explication du code

1. **Choix du maillage.** La zone étudiée sera de 90km de largeur et de 200km de longueur. On va diviser cette zone en 11 segments pour la largeur (soit un pas de ˜8km) et en 16 segments pour la longueur (soit un pas de 12.5km). On va donc de 41.5° à 43° par pas de 0.1° pour la latitude et de 8.5° à 9.5° par pas de 0.1. On fera donc la requête de 176 vitesses et 176 directions de vent. Voici le code associé à cette partie.

```
latitudes = np.arange(41.5, 43.02, 0.1)
longitudes = np.arange(8.5, 9.52, 0.1)
```

2. **Création d'une liste de coordonnées.** Pour cette partie, j'ai simplement utilisé les fonctions repeat et tile de la bibliothèque numpy. Celles-ci me permettant d'obtenir les 176 couples (latitude, longitude) pour obtenir le maillage souhaité.

```
l_latitudes = np.repeat(latitudes_list, len(longitudes)).tolist()
l_longitudes = np.tile(longitudes_list, len(latitudes)).tolist()
```

3. **Utilisation de l'API et obtention du jeu de données.** L'API étant très bien documenté, j'ai pu facilement utiliser celle-ci pour obtenir mon jeu de données. Une partie du code étant fournie sur le site, j'ai simplement dû modifier les champs pour la latitude et la longitude afin de donner la liste de latitudes & longitudes. J'obtiens les vitesses et les directions des vents sur chaque point pour chaque heure de la journée du 09/09/2024 (choisi arbitrairement).

```
import requests_cache
import pandas as pd
from retry_requests import retry
# Setup the Open-Meteo API client with cache and
#retry on error cache_session = requests_cache.CachedSession('.cache',
    expire_after = -1)
retry_session = retry(cache_session, retries = 5, backoff_factor = 0.2)
openmeteo = openmeteo_requests.Client(session = retry_session)
# Make sure all required weather variables are listed here
# The order of variables in hourly or daily is important to assign them correctly
url = "https://archive-api.open-meteo.com/v1/archive"
params = {
        "latitude": l_latitudes,
        "longitude": l_longitudes,
        "start_date": "2024-09-09",
        "end_date": "2024-09-09",
        "hourly": ["wind_speed_10m", "wind_direction_10m"] }
responses = openmeteo.weather_api(url, params=params)
```

4. **Extraction des données à 0h00.** Après avoir obtenu toutes les données, j'ai décidé d'essayer d'afficher le champ de vent pour la première heure du jeu de données soit 0h00. Pour chaque couple (latitude, longitude), j'ai extrait la direction et la vitesse du vent. Je suis ensuite passé des coordonnées polaires aux coordonnées cartésiennes en convertissant la vitesse et la direction en coordonnées d'un vecteur de cette manière :

```
wind_direction_radians = np.deg2rad(wind_dir_tmp)
wind_data.at[i, 'u'] = wind_spd_tmp * np.cos(wind_direction_radians)
wind_data.at[i, 'v'] = wind_spd_tmp * np.sin(wind_direction_radians)
```

5. **Création de la carte.** Enfin, après avoir obtenu nos données, on les affiche en utilisant les librairies cartopy et matplotlib. J'affiche les vecteurs sous la forme de quiver (flèche représentant les vecteurs). J'obtiens la figure di-dessous via le code suivant :

```
# Creation de la carte
plt.figure(figsize=(10, 8))
ax = plt.axes(projection=ccrs.Mercator())
# Tracer les contours de la Corse
ax.coastlines()
ax.set_extent([8.25, 9.6, 41.2, 43.1], crs=ccrs.PlateCarree())
ax.set_title("Direction et vitesse du vent a l'heure 0")
ax.quiver(wind_data['lon'], wind_data['lat'], wind_data['u'], wind_data['v'],
    scale=100, transform=ccrs.PlateCarree(), color='r')
plt.show()
```

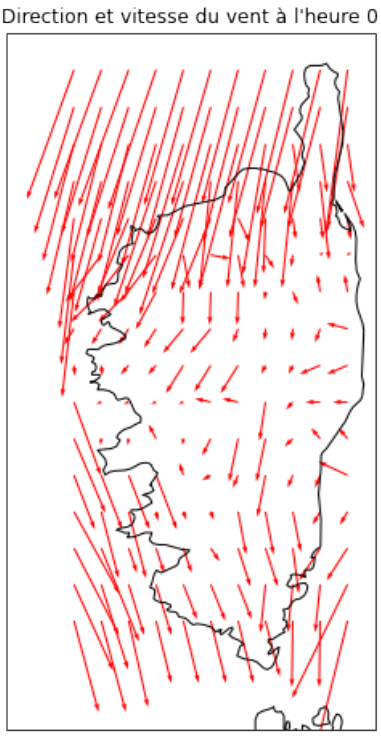

FIGURE 3 – Champ de vents en Corse le 09/09/2024 à 0h00.

3 Production de données

Afin d'obtenir un jeu de données exploitable, on pourrait également faire le choix de créer nos données plutôt que d'aller les chercher sur une base de données extérieure. Cette approche aura l'avantage de pouvoir être utilisée en local. Grâce aux modèles de mécanique des fluides, on pourrait simuler un champ de vecteurs réaliste pouvant être exploité.

Compte rendu Réunion 25/09. Points à aborder :

- Comment interpoler de 2D (coordonnées (x, y)) vers 2D (vecteur vent (u, v)) ? On va d'abord chercher à interpoler u pour tout point (x, y) de la grille en fixant un v. On fera ensuite la même chose avec v en fixant u. On pourra enfin reformer les vecteurs de vents interpolés.
- Comment utiliser un système de mécanique des fluides pour créer des champs de vents réalistes ? 2 méthodes :

1. Les *écoulements potentiels.* La manière la plus simple de modéliser les vents. Dans ce modèle, le champ de vents $\overrightarrow{v}$ respectent 2 propriétés : la *divergence nulle* soit $div(\overrightarrow{v}) = \overrightarrow{\nabla} \cdot \overrightarrow{v} = 0$ (cohérent car l'air est un fluide incompressible) et l'*irrotationnalité* soit $\nabla \wedge v = 0$ (il n'y a pas de tourbillons).
   Rappel. $\nabla = (\frac{\partial}{\partial x}, \frac{\partial}{\partial y}, \frac{\partial}{\partial z})$. En 3D, $\overrightarrow{\nabla} \cdot \overrightarrow{v} = \frac{\partial v_x}{\partial x} + \frac{\partial v_y}{\partial y} + \frac{\partial v_z}{\partial z}$ et $\nabla \wedge v = (\frac{\partial v_z}{\partial y} - \frac{\partial v_y}{\partial z}, \frac{\partial v_x}{\partial z} - \frac{\partial v_z}{\partial x}, \frac{\partial v_y}{\partial x} - \frac{\partial v_x}{\partial y})$.
   Dans $\mathbb{R}^3$ et sous les hypothèses du théorème de Helmholtz-Hodge [1], si un champ $\overrightarrow{v}$ est *irrotationnel*, alors il existe un champ scalaire ψ tel que $\overrightarrow{v} = -\overrightarrow{\nabla}\psi$. On a donc, pour un champ à *divergence nulle*,

$$div(\overrightarrow{v}) = div(-\overrightarrow{\nabla}\psi) = \Delta\psi = 0$$

   Dans un domaine Ω, trouver le champ de vents revient à trouver ψ tel que

$$\begin{cases} \Delta\psi = 0 & \text{dans } \Omega \\ \nabla\psi \cdot \mathbf{n} = \overrightarrow{v} \cdot \mathbf{n} & \text{sur } \partial\Omega \end{cases}$$

   Cette formulation permet une infinité de solutions en ajoutant une constante. Pour fixer, une solution, on imposera donc $\psi = 0$ dans Ω On essayera de résoudre cette EDP en utilisant la librairie python Fipy [2]. On pourra par la suite comparer les valeurs obtenues avec les données réelles (obtenables à partir de l'API). Cette comparaison nous indiquera si, en situation réelle, nos hypothèses de base sont correctes.
   À vérifier. Il me semble que si on a des points à l'intérieur du domaine, on pourra considérer le problème suivant :
   Trouver $\psi^* = argmin_\psi(\sum_{i=1}^{N}(\nabla\psi_i - v_i)^2$ avec $\Delta\psi = 0$

2. Les *écoulements de Stokes.* Si on retire l'hypothèse d'irrationnalité, on peut essayer de résoudre le système en considérant l'air comme un fluide newtonien incompressible en régime permanent et à faible nombre de Reynolds [3] :

$$\begin{cases} -\Delta v + \mu\overrightarrow{\nabla}p = \rho\overrightarrow{f} \\ \Delta v = 0 \end{cases}$$

   avec p la pression dans le fluide, μ la viscosité de l'air, ρ la masse volumique du fluide et $\overrightarrow{f}$ une force massique s'exerçant dans le fluide. En considérant $\rho\overrightarrow{f}$, on a donc le système suivant :

$$\begin{cases} -\Delta v + \mu\overrightarrow{\nabla}p = 0 \\ \Delta v = 0 \end{cases}$$

   Dans ce contexte, on a besoin de connaître v sur $\partial\Omega$. S'il y a des obstacles sur dans l'espace, on devra rajouter les contraintes suivantes : $\overrightarrow{n} = 0$ sur les bords de l'obstacle.

3. Les *écoulements de Navier-Stokes.* On pourra également utiliser l'équation de Navier-Stokes pour un fluide[4][5] :

$$\rho(\frac{\partial v}{\partial t} + v \cdot \nabla v) = \underbrace{-\nabla p}_{Fpression} + \underbrace{\mu \Delta v}_{Fvisqueuse}$$

Avec v le champ de vent, p la pression, rho la masse volumique du fluide et mu sa viscosité. En ajoutant la condition $\nabla v = 0$ et des conditions limites (**à préciser**) on peut simuler un champ de vent grâce aux principes de la thermodynamique.

La suite.

- Faire des interpolations en utilisant les méthodes de Lagrande et les splines.
- Etudier la génération de vents via les modèles de mécanique des fluides.
- Comparer avec les données réelles obtenables via l'API. Conclure sur la pertinence de chaque méthode.
- Etudier la structure des fichiers vtk utilisables pour le logiciel Paraview afin de visualiser les champs de vents
- Utiliser les outils d'interpolation proposés par le logiciel
- Etudier les méthodes de résolution d'EDP via des réseaux de neurones (mentionné en méta-apprentissage)(voir cette étude) [6]

4 Interpolation des données

4.1 Interpolation : Librairie Scipy

4.1.1 Classe scipy.interpolate.interp2d

Dans un premier temps, j'ai essayé d'interpoler les données avec une librairie déjà existante : la classe interp2d de scipy.interpolate. Celle-ci permet d'interpoler une grille en 2D ce qui m'a paru assez adapté pour notre cas. Cette classe utilise la méthode des splines pour interpoler en prenant en entrée deux tableaux de coordonnées et un tableau de valeurs à interpoler. Après avoir testé avec succès cette classe, j'ai obtenu des résultats assez convaincants avec des splines de degré 1.

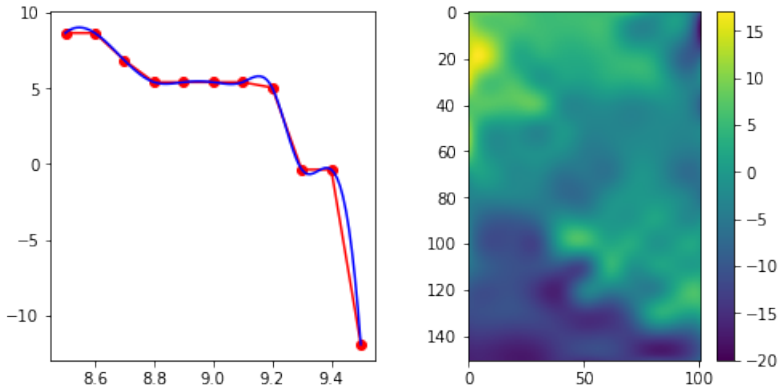

FIGURE 4 – Coordonnées u du champ de vents interpolé (horizontal)

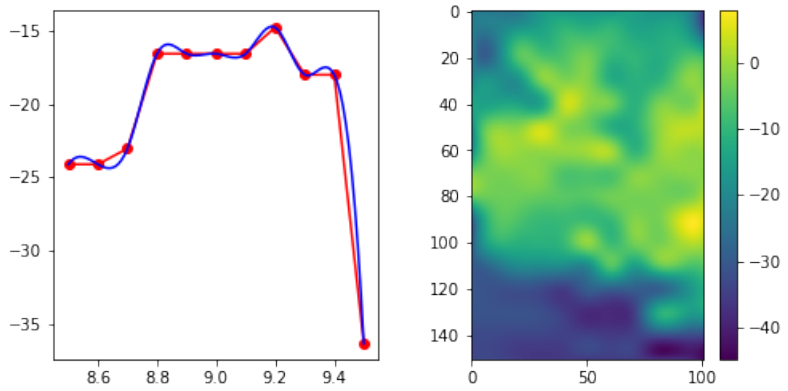

FIGURE 5 – Coordonnées v du champ de vents interpol

Cependant, avec avoir lu la documentation de la classe (`https://docs.scipy.org/doc/scipy-0.14.0/reference/generated/scipy.interpolate.interp2d.html#scipy.interpolate.interp2d`, je me suis rendu compte que celle-ci n'était plus d'actualité. En effet, elle ne sera plus disponible à partir de la version Scipy 1.14.0. Par conséquent, j'ai cherché d'autres alternatives à cette classe.

4.1.2 Classe scipy.interpolate.RectBivariateSpline

J'ai donc décidé d'utiliser une autre classe de scipy.interpolate : RectBivariateSpline `https://docs.scipy.org/doc/scipy/reference/generated/scipy.interpolate.RectBivariateSpline.html#scipy.interpolate.RectBivariateSpline` qui utilise la méthode des splines pour interpoler sur des grilles rectangulaires régulières. De base, les splines sont de degré 3. Pour commencer, on a utilisé la classe RectBivariateSpline avec des splines de degré 3. On obtient les résultats suivants.

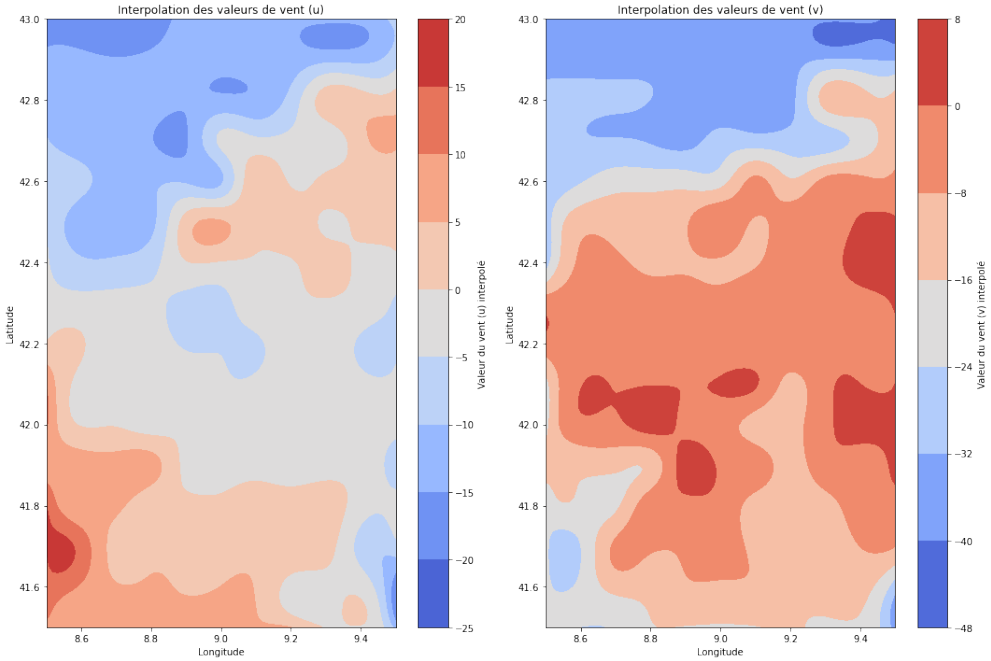

FIGURE 6 – Interpolation en utilisant la classe RectBivariateSpline de scipy.interpolate

Pour rappel, la carte des vents en Corse est la suivante :

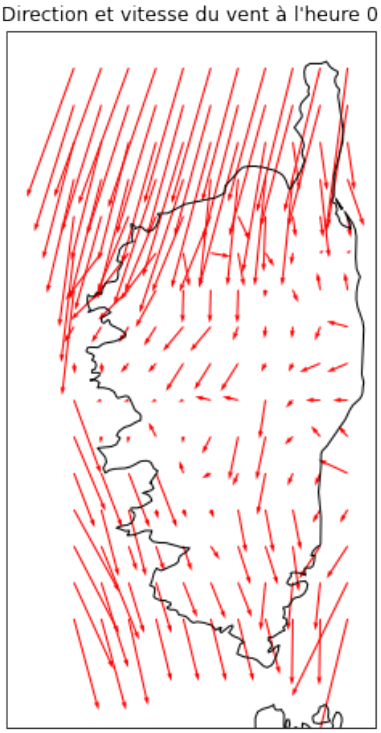

FIGURE 7 – Champ de vents en Corse le 09/09/2024 à 0h00.

Commentaire La méthode des splines utilisée par la classe RectBivariateSpline donne des résultats cohérents. On va pouvoir se baser sur ces résultats pour comparer les valeurs interpolées aux valeurs réelles dans la section 5 comparer les résultats obtenus avec les données réelles en réutilisant l'API.

Remarque Pour des points non répartis sur une grille rectangulaire régulière, on aurait pu utiliser la classe LinearNBInterpolator `https://docs.scipy.org/doc/scipy/reference/generated/scipy.interpolate.LinearNDInterpolator.html#scipy.interpolate.LinearNDInterpolator`.

4.2 Résolution d'EDP : Librairie Fipy[2]

4.3 Physics-informed Machine Learning

5 Comparaison des résultats

Méthode de comparaison Afin de comparer les données reconstruites et les données réelles, nous avons réutilisé l'API "Historical Weather" sur des points choisis aléatoirement dans l'espace. Soit i points choisis aléatoirement dans l'espace. Soit $\hat{y}_i = (\hat{u}_i, \hat{v}_i)$ la prédiction et $y_i = (u_i, v_i)$ la valeur du vecteur vent au point i. Nous souhaitons calculer la somme suivante :

$$\sum_i \|\hat{y}_i - y_i\|_2 = \sum_i \|(\hat{u}_i, \hat{v}_i) - (u_i, v_i)\|_2$$

Ce qui revient au code suivant (pour la méthode d'interpolation)

```
error_L1 = 0
error_L2 = 0
for i in range(nb_points_test):
    u_interpolated_test = test_u_interp(latitudes_test[i], longitudes_test[i])
    v_interpolated_test = test_v_interp(latitudes_test[i], longitudes_test[i])
    error_L1 += np.abs(u_interpolated_test - wind_data_test['u'][i]) + np.abs(v_interpolate
    error_L2 += (u_interpolated_test - wind_data_test['u'][i])**2 + (v_interpolated_test -
```

Résultats Pour 100 points répartis aléatoirement dans l'espace, on obtient :

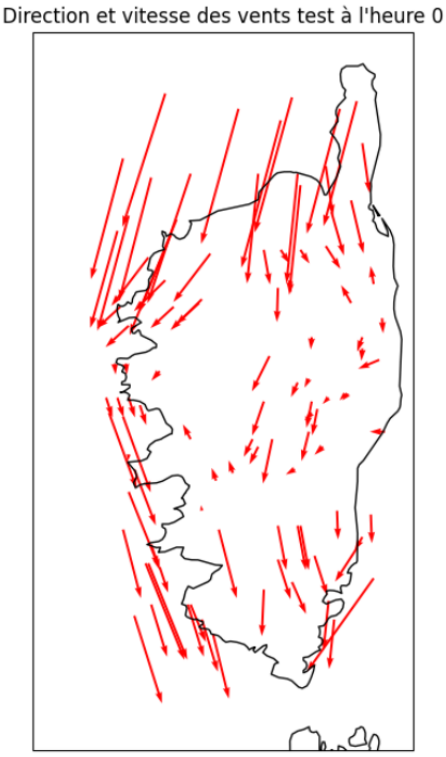

FIGURE 8 – Champ de vents test en Corse le 09/09/2024 à 0h00.

- *Interpolation avec la méthode des splines*

	lat	lon	u	v	u_interp	v_interp
0	42.90	9.34	-11.520060	-32.039974	-11.483510	-31.731617
1	42.45	8.72	-9.359989	-6.840014	-11.300405	-9.265398
2	41.83	9.45	0.359993	-7.560000	-3.600710	-1.546595
3	42.23	8.54	0.359994	-2.880001	0.165331	-1.897971
4	42.13	9.24	-1.800007	-6.839998	-1.422587	-6.636454
5	42.01	9.05	-2.880002	-4.319998	-3.475993	-7.196134
6	41.95	9.18	-2.519999	-1.080001	-3.594362	-11.510033
7	41.84	9.33	0.359993	-7.560000	0.716768	-6.358724
8	42.53	9.20	2.879995	-3.240005	2.409226	-5.824153
9	42.18	9.19	-2.159997	-3.600002	-2.030621	-3.580868
10	41.55	8.85	5.400014	-13.319994	5.286881	-12.861096
11	41.71	9.12	4.320002	-10.079999	4.477885	-11.044705
12	41.84	8.85	-0.360001	1.440000	1.553498	-2.552740
13	41.59	8.81	5.039981	-9.360009	5.443115	-13.476020
14	42.01	9.28	-2.519997	-1.800004	-2.548830	-3.785352
15	42.94	8.72	-14.760008	-33.839996	-16.534495	-35.302150
16	42.14	8.59	2.160001	-5.039999	-0.068825	-1.930623
17	42.30	9.24	-0.359994	-3.240000	-0.271073	-3.187076
18	42.73	9.05	-3.959952	-28.800007	-5.746638	-32.601955
19	42.58	8.55	-9.360030	-24.839987	-7.898067	-25.862985

FIGURE 9 – Comparaison des u,v et u,v interpolés via la méthode des splines.

Erreur en norme 1 = 458.06 - Erreur en norme 2 = 2690.64

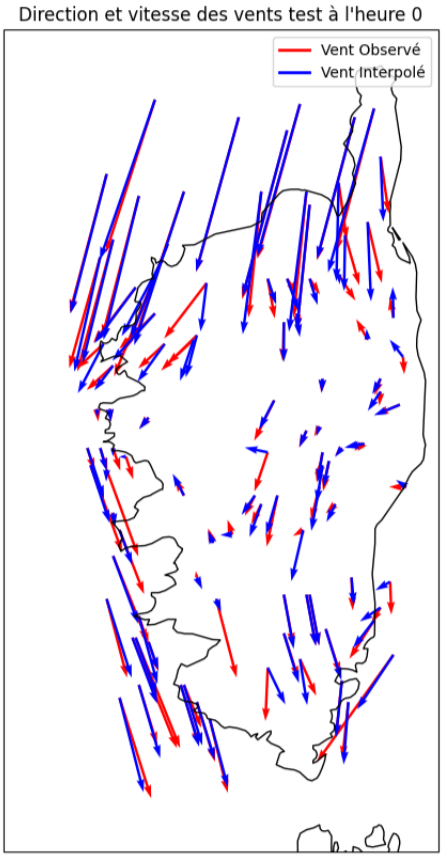

FIGURE 10 – Champ de vents réel et interpolé

Commentaire On remarque que, même si l'interpolation donnait l'impression de donner des résultats pertinentes, elle reste très éloignée de la réalité. On remarque que certaines interpolations sont assez mauvaises, notamment pour les vents faibles. Certains vents interpolés ne sont même pas dans la bonne direction. Selon peut être dû à plusieurs causes, notamment le nombre de points pris pour l'interpolation (seulement 176 points pour une zone de $18000km^2$) ou encore le degré des splines. On pourra tenter de faire varier ces paramètres.

Références

1. *Théorème de Helmholtz-Hodge* `http://fr.wikipedia.org/w/index.php?title=Th%C3%A9or%C3%A8me_de_Helmholtz-Hodge&oldid=218432116`. (Consulté le :27.09.2024).
2. Guyer, J. E., Wheeler, D. & Warren, J. A. FiPy : Partial Differential Equations with Python. *Computing in Science Engineering* **11,** 6-15. `http://www.ctcms.nist.gov/fipy` (2009).
3. *Écoulement de Stokes* `http://fr.wikipedia.org/w/index.php?title=%C3%89coulement_de_Stokes&oldid=207021958`. (Consulté le :27.09.2024).
4. *Écoulement de Navier-Stokse, Science étonnante* `https://scienceetonnante.com/2014/03/03/la-mysterieuse-equation-de-navier-stokes/`. (Consulté le :27.09.2024).
5. *Écoulement de Navier-Stokse, Techno-Science* `https://www.techno-science.net/definition/5797.html`. (Consulté le :27.09.2024).
6. Zhang, J. & Zhao, X. Three-dimensional spatiotemporal wind field reconstruction based on physics-informed deep learning. *Applied Energy* **300,** 117390. issn : 0306-2619. `https://www.sciencedirect.com/science/article/pii/S0306261921007911` (2021).