

# Modélisation spatio-temporelle du vent

## Rapport de projet

ABEILLE Quentin, LAIGRET Sébastien,  
LIBERAL CAVALCANTI Alexandre, GUEDIRA Oussama,  
ROUX Loïc

31 mars 2021  
ISAE-SUPAERO – PIE COA 08

Tuteurs  
DEBREILLY Franck, PASTOR Philippe, PROTHIN Sébastien

### Résumé

L'aéro-largage de colis équipés de parachutes a un fort enjeu stratégique, à la fois militaire et civil : il faut pouvoir larguer de la plus haute altitude possible avec l'ellipse de dispersion du point d'impact la plus réduite possible. Le vent a toutefois tendance à perturber les trajectoires anticipées de ces parachutes, si bien qu'il est indispensable de pouvoir le modéliser avec une bonne fiabilité. Notre travail permet, à partir de la connaissance d'un bulletin météo et du relief, de proposer un champ spatio-temporel du vent.

Une étude bibliographique permet de mettre en avant l'intérêt des méthodes diagnostiques pour le calcul du vent stationnaire avec une bonne prise en compte du terrain. Un code qui permet d'obtenir, à partir des prédictions météorologiques et de la topographie du terrain, un cube de vent stationnaire avec un pas spatial relativement faible est alors développé à l'aide du logiciel **WindNinja**. Enfin, une série de tests permet finalement de comparer nos données simulées à des relevés expérimentaux et d'y voir un accord tout à fait satisfaisant.

**Mots-clés :** *Modélisation, Vent, WindNinja, Modèles diagnostiques, Turbulence*

## Table des matières

<b>Liste des figures</b>	<b>5</b>
<b>Liste des tables</b>	<b>5</b>
<b>1 Bibliographie</b>	<b>8</b>
1.1 Approche par la mécanique des fluides . . . . .	8
1.1.1 Modèles explicites . . . . .	8
1.1.2 Modèles Diagnostiques . . . . .	10
1.1.3 Méthodes spectrales . . . . .	16
1.1.4 Synthèse . . . . .	18
1.2 Approche par l'étude statistique . . . . .	19
1.2.1 Distributions statistiques . . . . .	19
1.2.2 Méthodes stochastiques . . . . .	20
1.2.3 Séries temporelles . . . . .	22
1.2.4 Réseaux de neurones . . . . .	23
1.2.5 Caractéristiques des données . . . . .	23
1.2.6 Synthèse . . . . .	24
<b>2 Démarche retenue</b>	<b>26</b>
2.1 Extrapolation en altitude . . . . .	26
2.2 Composante temporelle : turbulence . . . . .	27
2.2.1 Analyse préliminaire de données expérimentales . . . . .	27
2.2.2 Incorporation de la turbulence . . . . .	30
2.2.3 Gestion de la cohérence spatiale . . . . .	33
<b>3 Notice du code</b>	<b>35</b>
3.1 Pré-requis . . . . .	35
3.2 Préparation du configFile.json . . . . .	36
3.2.1 Le bloc "def" . . . . .	36
3.2.2 Le bloc "extrapolation" . . . . .	36
3.2.3 Le bloc "windNinjaSimulations" . . . . .	36
3.3 Description de la classe . . . . .	37
3.3.1 Les attributs . . . . .	38
3.3.2 Les méthodes . . . . .	38
3.4 Fonctions utilisant le temps . . . . .	39
3.4.1 Fonctionnement . . . . .	39
3.4.2 Modifications par l'exploitant . . . . .	40
3.4.3 Un exemple pas à pas . . . . .	41
3.5 Fonctions pour l'extrapolation . . . . .	41
3.6 Fonctions pour l'interpolation et l'affichage des résultats . . . . .	42
3.6.1 Remarque importante pour les méthodes d'affichage . . . . .	42
3.7 Cas d'utilisation . . . . .	44
3.7.1 Préparation du calcul . . . . .	45
3.7.2 Lancement du calcul . . . . .	46
3.7.3 Exploitation des résultats . . . . .	47
3.8 Fonctions pour la récupération et la conversion des fichiers GRIB . . . . .	48
3.9 Fonctions pour la conversion des fichiers VTK . . . . .	49

<b>4 Rapport d'essais</b>	<b>50</b>
4.1 Première phase d'essais . . . . .	50
4.1.1 Présentation de la démarche . . . . .	50
4.1.2 Présentation des résultats . . . . .	50
4.1.3 Ajout de la turbulence . . . . .	52
4.2 Seconde phase d'essais . . . . .	54
4.2.1 Présentation de la démarche . . . . .	54
4.2.2 Présentation des résultats . . . . .	55
<b>Conclusion</b>	<b>56</b>
<b>Références</b>	<b>57</b>
<b>Annexes</b>	<b>60</b>

## Table des figures

1	L'intensité du vent est d'autant plus forte que l'on s'élève. Celui-ci est cependant loin d'être constant et les variations peuvent être très significatives, de l'ordre de 30%.	9
2	Les modèles de Lemelin, Surry et Davenport pour modéliser une colline ( <i>hill</i> ), une falaise ( <i>escarpment</i> ) ou une arrête ( <i>ridge</i> ). La pente moyenne $\phi = \frac{H}{L}$ dépend de la hauteur caractéristique $H$ de l'obstacle et de sa longueur moyenne $L$ .	11
3	Ce graphique permet de voir la concordance des deux lois pour le vent à basse altitude : le modèle de Prandtl avec le terme de Coriolis est très proche du modèle en puissance. Ici, la correspondance est faite à 10 m, $z_0 = 0.02$ m, $\bar{U}_{10} = 20$ m/s (et donc $U_* = 1.38$ m/s), $\alpha = 0.128$ et $\phi = 45^\circ$ .	12
4	Propositions de lois pour $\alpha$ en fonction de $St$ (d'après [18])	14
5	Exemple de calcul réalisé par <b>WindNinja</b> à partir de prévisions météo NOMADS-HRRR	15
6	A gauche, le spectre en fréquence dérivée de Von Karman, et à droite la reconstitution d'un vent suivant ces variations [3].	18
7	Histogramme et densité des modèles à la station Haripur, Pakistan	21
8	Résultats [16]	21
9	Résultats [9]	21
10	Vitesse de vent estimée VS observée pour une prédiction de 0h	22
11	Vitesse de vent estimée VS observée pour une prédiction de 1h	22
12	Erreur résiduelle pour une prédiction de 1h	23
13	Variation journalière de la vitesse de la vitesse du vent dans la région de Firouzkoooh (Iran) - [27]	24
14	Rose des vents dans la région de Firouzkoooh (Iran) - [27]	24
15	Extrapolations pour la composante $u$	26
16	Extrapolations pour la composante $v$	26
17	Extrapolations pour la composante $w$	27
18	Extrapolations pour la norme	27
19	Evolution des trois composantes du vent pendant une durée de 10 minutes.	28
20	Comparaison des densités spectrales : mesure, interpolation, théories.	29
21	Densité spectrale de puissance du vent latéral.	29
22	Densité spectrale de puissance du vent vertical.	30
23	Densité spectrale de puissance en fonction de l'altitude mesurée.	30
24	La composante turbulente est d'autant plus forte qu'on est proche du sol puis elle augmente quelque peu avec l'altitude.	31
25	La composante turbulente est d'autant plus forte que la vitesse stationnaire est élevée.	32
26	Vent turbulent généré par les méthodes spectrales dans le repère géographique.	32
27	Vent turbulent généré par les méthodes spectrales dans le repère du vent.	33
28	Densité spectrale de puissance du signal généré.	33
29	Cube de turbulence : à partir d'un vent stationnaire de base, on peut obtenir une possibilité d'évolution turbulente du vent en tout point, qui soit cohérente spatialement et temporellement.	35
30	Rose des vents : la vitesse turbulente est en bleu, la vitesse stationnaire en rouge clair.	41
31	Un exemple de <code>wind_cube</code> .	43

32	Un exemple de <code>wind_cube</code> turbulent : la turbulence est peu perceptible dès que l'on monte en élévation. . . . .	44
33	Un exemple de <code>wind_surface</code> . . . . .	45
34	Préparation du calcul . . . . .	45
35	Lancement du calcul . . . . .	46
36	Calcul en cours . . . . .	46
37	Fichiers de sortie . . . . .	47
38	Commandes d'affichage . . . . .	47
39	Le <code>wind_cube</code> interpolé . . . . .	48
40	Données d'entrée des essais . . . . .	50
41	Données simulées VS réelles pour la composante $u$ . . . . .	51
42	Données simulées VS réelles pour la composante $v$ . . . . .	51
43	Données simulées VS réelles pour la composante $w$ . . . . .	51
44	Données simulées VS réelles pour la norme plane de la vitesse. . . . .	51
45	Données simulées VS réelles pour la norme de la vitesse. . . . .	51
46	Données simulées VS réelles pour toutes les variables. . . . .	51
47	Répartition de l'erreur commise par la simulation. . . . .	52
48	Vitesse simulée en fonction de l'heure d'observation. . . . .	52
49	Vitesse simulée et réelle sur un échantillon (composante $u$ ). . . . .	53
50	Vitesse simulée et réelle sur un échantillon (composante $v$ ). . . . .	53
51	Vitesse simulée et réelle sur un échantillon (composante $w$ ). . . . .	54
52	Données simulées VS réelles pour la vitesse . . . . .	55
53	Données simulées VS réelles pour la direction . . . . .	55

## Liste des tableaux

1	Paramètre de rugosité en fonction du type de terrain. . . . .	10
2	Paramètres topographiques du modèle de Davenport pour une arête ( $S_m = 2.3G$ ) ou une colline ( $S_m = 2.3G \frac{B/L_0}{B/L_0+0.4}$ ) . . . . .	10
3	Paramètres topographiques du modèle de Davenport pour une falaise ( $S_m = 1.3G$ ) . . . . .	10
4	Résumé des options et temps de calcul en fonction du solveur . . . . .	16
5	Modèle de distribution . . . . .	20
6	Récapitulatif des résultats des tests . . . . .	52
7	Récapitulatif des résultats des tests . . . . .	55
8	Fuseaux horaires possibles. . . . .	63

## Notations

$x$	Direction dominante du vent
$y$	Perpendiculaire à $x$ dans le plan horizontal
$z$	Verticale locale
$t$	Temps
$\mathbf{U}(x, y, z, t) = \begin{cases} u(x, y, z, t) \\ v(x, y, z, t) \\ w(x, y, z, t) \end{cases}$	Vitesse du vent selon l'axe $x$ Vitesse du vent selon l'axe $y$ Vitesse du vent selon l'axe $z$
$\bar{U}$	Moyenne temporelle de $\mathbf{U}$ dans la direction dominante du vent
$\tilde{u}$	Composante turbulente de $\mathbf{U}$ dans la direction dominante
$\tilde{v}$	Composante turbulente de $\mathbf{U}$ dans la direction latérale
$\tilde{w}$	Composante turbulente de $\mathbf{U}$ dans la direction verticale
<code>wind</code>	Classe python permettant la création et la gestion de <code>wind_cube</code>
<code>wind_cube</code>	Cube de vent
<code>wind_surface</code>	Surface de vent à altitude constante
<code>wind_profile</code>	Profile de vent en altitude à une position donnée
a.g.l	Above Ground Level, élévation au-dessus du sol
a.s.l	Above Sea Level, altitude au-dessus du niveau de la mer
Statistiques	
$f_{\alpha, \beta}(x)$	Fonction de densité de probabilité d'une loi de paramètres $\alpha$ et $\beta$
$F_{\alpha, \beta}(x)$	Fonction de répartition d'une loi de paramètres $\alpha$ et $\beta$
$\hat{F}_{\alpha, \beta}(x)$	Estimation de $F_{\alpha, \beta}$

## Introduction

Le support de troupes au sol est l'une des missions de l'Armée de l'Air et de l'Espace en France. Pour l'assurer elle doit notamment réaliser le parachutage de charges utiles dans des zones hostiles et géographiquement difficiles d'accès. Pour ces missions, une erreur de 100 m sur le point de chute du colis peut signifier sa perte ce qui oblige actuellement les équipages à voler à très basse altitude pour limiter l'ellipse de dispersion. Cette solution ne satisfait pas la DGA car elle expose les avions aux risques du terrain et d'une attaque ennemie.

Pour cette raison, la création d'un parachute auto-guidé est à l'étude afin de permettre un largage à haute altitude tout en assurant la précision nécessaire à l'atterrissement. Pour une telle solution, une modélisation très précise du vent, prenant en compte les effets de terrain est nécessaire aussi bien pour l'entraînement de la loi de commande du parachute que pour le calcul du point de largage. Néanmoins, aujourd'hui les seules données disponibles sont celles issues des modèles météo méso-échelle ou de stations au sol. Les premières ne sont pas suffisamment précises dans leur maillage de la surface de la Terre pour bien prendre en compte les effets de terrain tandis que les secondes impliquent une installation au sol qui n'est pas compatible avec notre application.

L'objectif de ce projet est donc de réaliser une modélisation du vent à partir des données météorologiques et de terrain qui soit suffisamment précise et rapide à calculer pour le largage de colis auto-guidés. Dans un premier temps, une étude bibliographie permet de discuter les différents moyens actuels de modélisation du vent et de sa turbulence. Celle-ci s'appuie sur la lecture de nombreux articles qui proposent différents types de solution, entre modèles déterministes basés sur la mécanique des fluides, et modèles aléatoires s'appuyant sur des relevés statistiques. Les avantages des modèles diagnostiques, des méthodes spectrales et du logiciel open-source **WindNinja** sont alors mis en avant en vue de leur exploitation. Cette première étude théorique nous a amené à développer une classe **Python** permettant de construire un cube de vent allant jusqu'à 4000 m au-dessus du sol et de simuler la turbulence à partir de résultats de simulation **WindNinja**. La démarche de construction ainsi que le détail des différentes fonctions présentes sont explicités dans les parties 2 et 3. Finalement, cette classe est testée vis-à-vis de données expérimentales ce qui permet de conclure sur la validité de la modélisation choisie.

# 1 Bibliographie

## 1.1 Approche par la mécanique des fluides

Le vent, mouvement de gaz par rapport à la surface de la Terre, a pour origine les gradients de température à sa surface. Ces gradients sont dus à des différences dans le rayonnement solaire, les courants marins, la rotation de la Terre. Il a également été établi que les variations du vent sont plus importantes dans la couche limite de turbulence atmosphérique. Celle-ci peut varier de quelques centaines de mètres à plusieurs kilomètres en fonction du terrain. La vitesse du vent augmente alors, comme le montre la Figure 1 jusqu'à la fin de la couche limite et au-delà, garde une valeur constante le long des isobares.

Les composantes du vent atmosphérique se décomposent en une vitesse moyenne  $\bar{U}$ , orientée selon nos conventions sur l'axe ( $Ox$ ) et des vitesses turbulentées si bien que :

$$\mathbf{U}(x, y, z) = (\bar{U} + \tilde{u}(x, y, z), \tilde{v}(x, y, z), \tilde{w}(x, y, z)) \quad (1)$$

Dans la partie 1.1.2, on utilisera plutôt un référentiel terrestre.

Les études sur la modélisation du vent commencent dans les années 1950 avec les travaux de Davenport. Cela s'est avéré nécessaire, typiquement après les années 1940 lors de la destruction spectaculaire du pont de Tacoma à cause de phénomènes aéroélastiques.

Dans cette partie, nous nous intéressons à la détermination de l'ensemble des composantes. Après avoir vu comment variait la composante moyenne en fonction de la qualité du terrain, nous préciserons deux voies possibles. Les méthodes diagnostiques, reposant sur les lois de la mécanique des fluides, qui permettent de connaître la direction et la force du vent stationnaire, et les méthodes spectrales, introduisant déjà une part d'aléatoire aideront à appréhender la turbulence.

### 1.1.1 Modèles explicites

La notion de vitesse moyenne reste somme toute relative dans la mesure où elle résulterait de l'intégration de la vitesse sur une certaine durée. Plusieurs échelles de temps caractéristiques peuvent être choisies : le jour qui correspond aux variations météorologiques, la minute qui prend en compte la turbulence atmosphérique. Nous nous plaçons dans ce dernier cas en moyennant sur une dizaine de minutes, ce qui correspond à une durée caractéristique du cas d'utilisation du parachute. La vitesse moyenne varie alors souvent en fonction de l'altitude  $z$  en fonction du frottement en sol qui dépend également de la nature du terrain. Dans la suite, on ne s'attardera pas au profil du vent moyen au-dessus de l'océan, où la prise en compte des courants marins influe grandement sur la vitesse du vent<sup>1</sup>.

#### Modèle logarithmique

**Dépendance de la rugosité de Prandtl** Le modèle qui est le plus souvent pris pour la vitesse moyenne est issu de nombreux articles, dont [20] et [17], et est construit à partir d'une analyse dimensionnelle sur la contrainte de cisaillement du vent :

$$\bar{U}(z) = \frac{U_*}{\kappa} \ln \left( \frac{z - z_h}{z_0} \right) \quad (2)$$

où  $z_0$  est une hauteur qui dépend du type de terrain et de sa *rugosité*,  $\kappa = 0.42$  est la constante de Von Karman,  $U_*$  a la dimension d'une "vitesse" de friction. Cette grandeur

1. Ceci est par exemple étudié dans [17] qui développe aussi le vent en présence de cyclones.

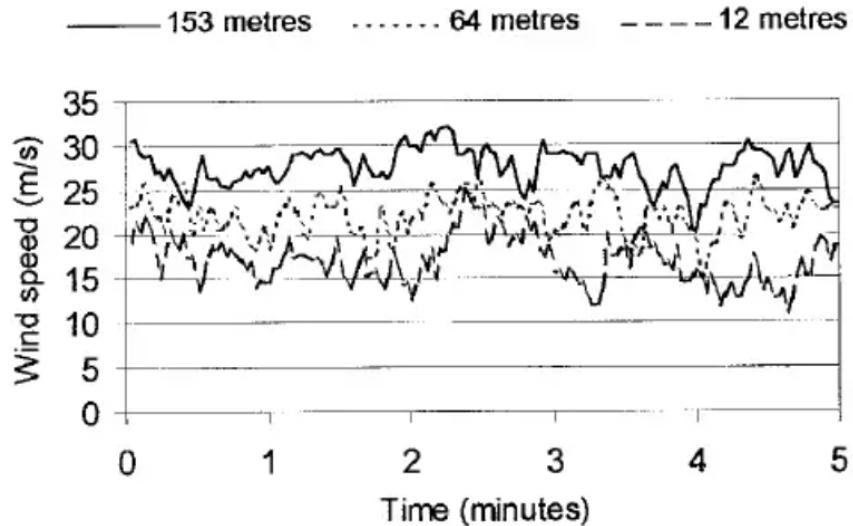


FIGURE 1 – L’intensité du vent est d’autant plus forte que l’on s’élève. Celui-ci est cependant loin d’être constant et les variations peuvent être très significatives, de l’ordre de 30%.

dépend du lieu et est la plus difficile à estimer<sup>2</sup>. Pour une même hauteur du sol, la vitesse de l’écoulement sera d’autant plus faible que la rugosité du lieu observé est importante, autrement dit, que les obstacles seront plus élevés et nombreux. Si la valeur moyenne  $\bar{U}_{10}$  du vent à une hauteur de 10 m a été relevée, alors

$$U_* = \kappa \frac{\bar{U}_{10}}{\ln\left(\frac{10}{z_0}\right)} \quad (3)$$

Il est possible de perfectionner ce modèle. Dans les zones très rugueuses (les forêts ou les villes par exemple), l’altitude  $z$  est souvent corrigée par  $z - z_h$ , où  $z_h \simeq \frac{3}{4}H$ ,  $H$  étant la hauteur caractéristique de l’obstacle et le modèle précédent ne s’applique donc que pour  $z > z_h$ . Par ailleurs, on peut inclure l’effet de la rotation de la Terre en prenant en compte la correction de Coriolis [22], qui utilise la vitesse de rotation  $\Omega$  et la latitude géographique  $\phi$ <sup>3</sup>. Tout cela revient à écrire :

$$\bar{U}(z) = \frac{U_*}{\kappa} \left( \ln\left(\frac{z - z_h}{z_0}\right) + 34.52\Omega \sin(\phi) \frac{z}{U_*} \right) \quad (4)$$

La rugosité ne prend néanmoins pas en compte la présence d’obstacles précis. L’écoulement de l’air, en vertu de la conservation de la masse, est pourtant modifié par la topographie et sa vitesse augmente alors à proximité du relief.

**Complément pour certains obstacles** Les travaux de [24] s’attardent particulièrement sur le cas de certains reliefs : les arêtes, les collines et les falaises visualisables sur Fig. 2. En se référant aux images, ils proposent de multiplier la vitesse du modèle précédent par un *coeffcient de site*  $k_s(x, z)$  qui dépend de la projection sur le relief considéré selon :

$$k_s(x, z) = 1 + S_m \frac{1}{\left(1 + 3\left(\frac{x}{nL}\right)^p\right)^2 \left(1 + a\frac{z}{L}\right)} \quad (5)$$

2. Typiquement, au niveau des étendues d’eau, [22] propose la relation  $U_* = \sqrt{\frac{z_0 g}{A}}$  avec  $A = 0.0167$ .

3. A  $\phi = 45^\circ$ , cela donne un ordre de grandeur  $2\Omega \sin(\phi) \simeq 10^{-4} \text{s}^{-1}$ .

Type de surface	$z_0$ (m)
Rase campagne ou aéroports	0.05
Campagne avec haies, habitat dispersé	0.2
Zone urbanisée, industrielle, forestière	0.75
Zone urbaine, bâtiments de hauteur supérieure à 15 m	2.00
Vaste étendue d'eau	0.005

TABLE 1 – Paramètre de rugosité en fonction du type de terrain.

où tous les coefficients  $S_m$ ,  $n$ ,  $p$  dépendent du relief selon Tab. 2 et Tab. 3.

Pente de l'obstacle	$G$	$L$	$a$	$n$	$p$
$\phi < 0.4$	$\phi$	$L$	2.0	2.0	2.0
$\phi > 0.4$	0.4	$2.5H$	2.0	2.0	2.0

TABLE 2 – Paramètres topographiques du modèle de Davenport pour une arête ( $S_m = 2.3G$ ) ou une colline ( $S_m = 2.3G \frac{B/L_0}{B/L_0+0.4}$ )

Pente de l'obstacle	$G$	$L$	$a$	$n(x < 0)$	$n(x > 0)$	$p(x < 0)$	$p(x > 0)$
$\phi < 0.4$	$\phi$	$L$	2.0	1.0	5.0	2.0	1.0
$\phi > 0.4$	1	$H$	0.6	0.5	10	2.0	1.0

TABLE 3 – Paramètres topographiques du modèle de Davenport pour une falaise ( $S_m = 1.3G$ )

### Modèle puissance

La loi en puissance pour le vent n'a aucun fondement théorique mais elle vient remplacer la loi logarithmique pour pallier deux défauts. La loi logarithmique est en effet plus délicate à intégrer et il n'est pas licite de l'utiliser dès lors que  $z < z_h$ . Les spécialistes en ingénierie du vent proposent alors :

$$\bar{U}(z) = \bar{U}_{10} \left( \frac{z}{10} \right)^\alpha \quad (6)$$

Si on désire réaliser une correspondance entre la loi logarithmique et la loi de puissance, à une hauteur  $z_{cor}$ , il est possible d'utiliser la relation :

$$\alpha \log \left( \frac{z_{cor}}{z_0} \right) = 1 \quad (7)$$

La Figure 3 met en parallèle les deux modèles retenus jusque ici.

#### 1.1.2 Modèles Diagnostiques

##### Introduction

Les modèles diagnostiques ont été très étudiés à partir de la fin des années 1980 en parallèle des modèles dits "prognostic" qui résolvent un jeu très complet d'équations de l'atmosphère (Navier-Stokes, humidité de l'air, rayonnement du sol) au prix d'un coût de calcul très élevé. Ils sont notamment plébiscités pour leur capacité à produire des champs de

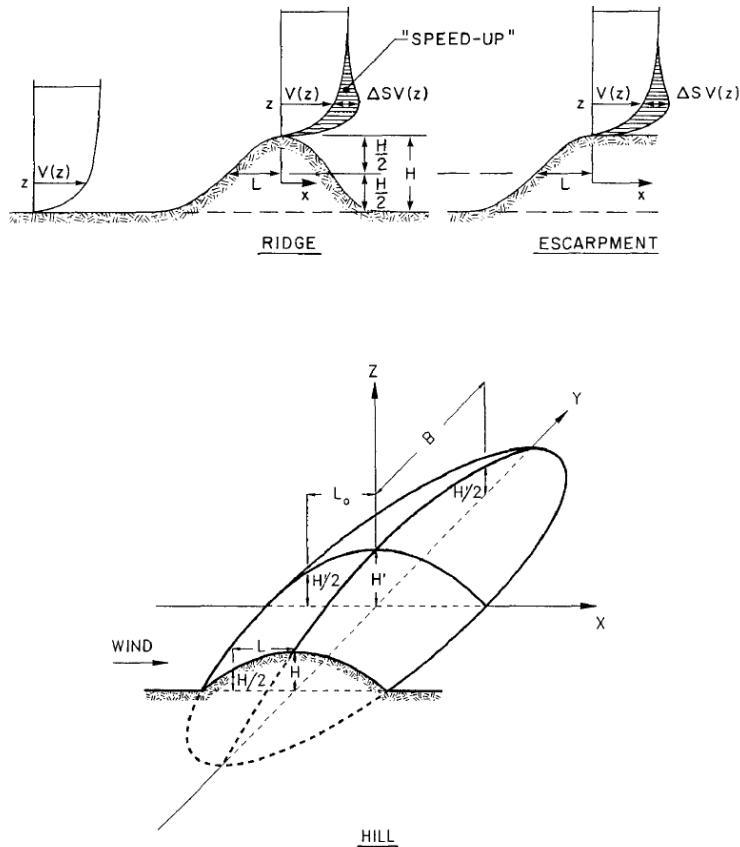


FIGURE 2 – Les modèles de Lemelin, Surry et Davenport pour modéliser une colline (*hill*), une falaise (*escarpment*) ou une arrête (*ridge*). La pente moyenne  $\phi = \frac{H}{L}$  dépend de la hauteur caractéristique  $H$  de l'obstacle et de sa longueur moyenne  $L$ .

vent cohérents avec les observations, prenant en compte le terrain et nécessitant un temps de calcul faible [18]. Ils ont ainsi souvent été utilisés pour l'anticipation de catastrophes (propagation de feux de forêts ou de nuages nocifs). Le principe général de ces méthodes est de "ré-équilibrer" un champ de vitesses du vent interpolé à un instant  $t$  en rajoutant comme contraintes une ou plusieurs équations de Navier-Stokes en incompressible [12]. Deux types de méthodes diagnostiques sont détaillés dans la suite de cette partie : l'une basé sur le respect de l'équation de continuité et l'autre rajoutant en plus l'équation de la quantité de mouvement.

#### Les méthodes dites "*Mass Conservative*"

Les méthodes *Mass Conservative* (**MC**) proposent une correction d'un champ de vent interpolé en résolvant en tout point du maillage l'équation de conservation de la masse :

$$\nabla \cdot \mathbf{U} = 0 \quad (8)$$

Pour ce faire, plusieurs méthodes existent et sont détaillées dans l'article [18]. Parmi elles, la méthode *Variational Calculus* semble être préférée par une large part de la com-

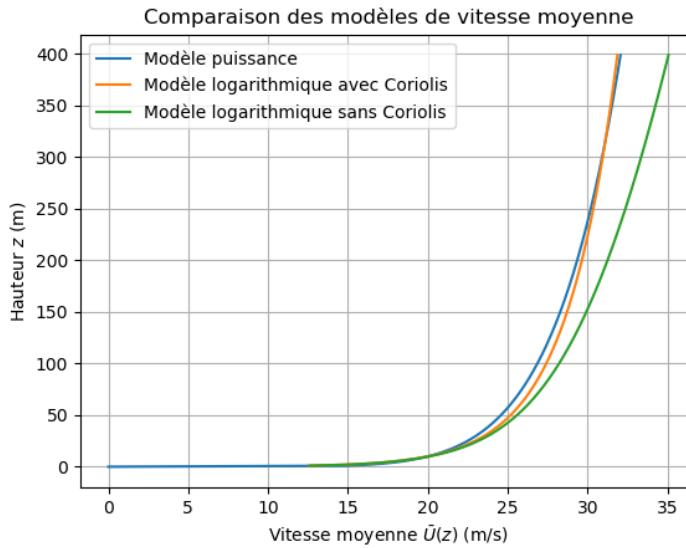


FIGURE 3 – Ce graphique permet de voir la concordance des deux lois pour le vent à basse altitude : le modèle de Prandtl avec le terme de Coriolis est très proche du modèle en puissance. Ici, la correspondance est faite à 10 m,  $z_0 = 0.02$  m,  $\bar{U}_{10} = 20$  m/s (et donc  $U_* = 1.38$  m/s),  $\alpha = 0.128$  et  $\phi = 45^\circ$ .

munauté scientifique pour sa capacité à traiter les trois composantes du vent en simultané et sans trop d'ambiguïtés. C'est celle que l'on détaillera dans cette partie.

L'idée est de minimiser la fonctionnelle

$$E(u, v, w) = \int_{\Omega} [\alpha_1^2(u - \hat{u})^2 + \alpha_1^2(v - \hat{v})^2 + \alpha_2^2(w - \hat{w})^2] \, d\Omega \quad (9)$$

sous la contrainte de (8) qui se réécrit :

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z} = 0 \quad (10)$$

En appliquant la théorie des multiplicateurs de Lagrange, on peut combiner (9) et (10) pour former la fonctionnelle :

$$F(u, v, w, \lambda) = \int_{\Omega} \left[ \alpha_1^2(u - \hat{u})^2 + \alpha_1^2(v - \hat{v})^2 + \alpha_2^2(w - \hat{w})^2 + \lambda \left( \frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z} \right) \right] \, d\Omega \quad (11)$$

qui est celle que l'on cherche maintenant à minimiser.

Détaillons les différents paramètres :

- $\Omega$  est le domaine d'étude.
- $(u, v, w)$  sont les trois composantes du champ des vitesses.
- $(\hat{u}, \hat{v}, \hat{w})$  sont les valeurs initiales des composantes du champ des vitesses.
- $\lambda$  est le paramètre de Lagrange

- $\alpha_1, \alpha_2$  sont les modules de précision de Gauss (*Gauss precision moduli*). Ils fixent dans quelle mesure la composante de vitesse associée est autorisée à changer. Par exemple  $\alpha_1 = \alpha_2 = 1$  veut dire que les composantes horizontales et verticales peuvent évoluer dans la même amplitude. Ils sont donc un reflet de la stabilité atmosphérique.

Le minimum de (11) est trouvé grâce aux solutions des équations d'*Euler-Lagrange* associées

$$u = \hat{u} + \frac{1}{2\alpha_1^2} \frac{\partial \lambda}{\partial x}, \quad v = \hat{v} + \frac{1}{2\alpha_1^2} \frac{\partial \lambda}{\partial y}, \quad w = \hat{w} + \frac{1}{2\alpha_2^2} \frac{\partial \lambda}{\partial z}, \quad \frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z} = 0 \quad (12)$$

et qui sont soumises à la condition aux limites :

$$\lambda \delta \mathbf{U} \cdot \mathbf{n} = 0 \text{ sur } \Gamma, \text{ la frontière de } \Omega \quad (13)$$

En combinant (12) et (13) et en posant  $\alpha = \alpha_1/\alpha_2$ , il vient :

$$\frac{\partial^2 \lambda}{\partial x^2} + \frac{\partial^2 \lambda}{\partial y^2} + \alpha \frac{\partial^2 \lambda}{\partial z^2} = -2 \left( \frac{\partial \hat{u}}{\partial x} + \frac{\partial \hat{v}}{\partial y} + \frac{\partial \hat{w}}{\partial z} \right) \quad (14)$$

On peut donc ainsi calculer  $\lambda(x, y, z)$  à partir de (14) puis remonter à  $(u, v, w)$  en réinjectant la solution dans (12). Pour la condition aux limites (13), il est conseillé de prendre  $\lambda = 0$  pour une frontière dont on ne connaît pas le flux qui la traverse et  $\partial \lambda / \partial n = 0$  dans le cas opposé.

[18] référence de nombreux codes de calcul utilisant cette méthode. On peut notamment citer les codes MATHEW [30] et MINERVE [14] qui ont largement été repris et fait leurs preuves pour des applications comme l'analyse du transport de nuages toxiques et l'analyse de potentiel énergétique éolien.

Le principal défaut du *Variational Calculus* est l'ambiguïté qui existe autour des valeurs des paramètres  $\alpha_1$  et  $\alpha_2$  : sont-elles fixes ou variables sur le domaine ? Comment peut-on les relier à la stabilité de l'atmosphère ? Certaines analyses ont été réalisées pour relier  $\alpha$  au nombre de Froude *Fr* ou de Stroudhal *St* (cf. Fig. 4) mais la valeur finale de ce dernier reste tout de même à l'appréciation de l'opérateur.

D'un point de vue pratique, les méthodes **MC** sont régulièrement utilisées pour interpoler des champs de vent calculés par les modèles météo pour des méso-échelles. Elles permettent ainsi de passer d'une résolution de l'ordre de 3 km dans les meilleurs cas à une résolution de l'ordre de la centaine de mètres. Elles sont capables de bien prendre en compte les effets de terrain comme les vallées ou les collines et donnent des résultats satisfaisants par rapport aux observations. En revanche, elles sont incapables de modéliser des effets très locaux comme les phénomènes de décollement et de recirculation le long des pentes sous le vent. En effet, ces méthodes se contentent de corriger des champs interpolés, elles ne peuvent en aucun cas prédire des phénomènes non présent dans les données d'entrées. En outre, il a été montré que réaliser une interpolation à partir d'effets locaux conduit à une augmentation de l'erreur car les méthodes ne sont pas capables de conserver le caractère local du phénomène mais le diffuse vers les zones voisines [10]. Cela implique également que la qualité des résultats d'un modèle diagnostique dépend de la qualité des données d'entrées et de la méthode d'interpolation.

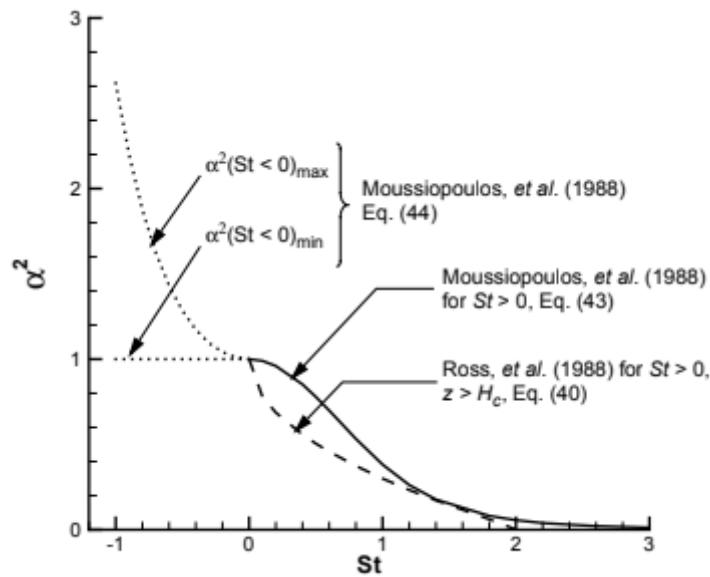


FIGURE 4 – Propositions de lois pour  $\alpha$  en fonction de  $St$  (d'après [18])

### En rajoutant l'équation de Quantité de Mouvement

A la suite des méthodes **MC**, l'idée de rajouter en plus l'équation de Quantité de Mouvement (**QdM**) fut rapidement envisagée. Un ajout permettrait en effet de modéliser des effets locaux purement dus à la topographie comme la recirculation sur les pentes sous le vent et donc de s'affranchir du principal inconvénient des méthodes **MC**. Cependant, à l'époque des premiers modèles diagnostiques dans les années 1990, la résolution de l'équation de **QdM** passait par sa linéarisation sous l'hypothèse des petites perturbations. Une telle hypothèse était donc incompatible avec des topographies complexes et de telles méthodes étaient délaissées au profit des méthodes **MC** [18].

L'amélioration constante des méthodes de *CFD* et l'augmentation de la puissance de calcul des ordinateurs au cours des 20 dernières années a complètement changé la situation. Il est en effet aujourd'hui possible de résoudre les équations de Navier-Stokes avec des méthodes numériques relativement simples et rapides. Ainsi, les méthodes de type RANS sont particulièrement adaptées à notre problématique de calculer rapidement un champ de vitesse du vent à partir de modèles météo. Ces modèles utilisent généralement des schémas de fermetures  $k - \varepsilon$ . La comparaison de différentes variantes du modèle  $k - \varepsilon$  dans [23] a montré que le modèle de turbulence RNG  $k - \varepsilon$  est le plus adapté pour des topographies complexes. Bien évidemment des méthodes de type LES donneraient des résultats encore meilleurs mais leur coût et le temps de calcul qu'elles requièrent les rendent inutilisables pour notre problème.

### Le maillage du domaine

Une problématique importante lors de l'utilisation d'une méthode diagnostique est la représentation du terrain dans le maillage du domaine.

**L'approche classique** est d'utiliser un maillage Cartésien ( $x, y, z$ ) classique et de forcer la vitesse du vent à zéro dans toutes les cases où l'on retrouve un morceau de terrain. On obtient alors une frontière en marches d'escalier dont la similitude avec la réalité dépend de la taille de maille utilisée.

**Une seconde méthode** beaucoup plus précise passe par l'utilisation d'un maillage *TFC*

(Terrain Following Coordinates) dans lequel la coordonnées  $z$  est remplacées par :

$$\sigma = \frac{z - h(x, y)}{H(x, y) - h(x, y)} \quad (15)$$

Où  $h$  est la hauteur du terrain en  $(x, y)$  et  $H$  est la hauteur du domaine. On obtient alors une 3ème coordonnée variant de 0 à 1. La condition aux limites en  $\sigma = 0$  est tout simplement  $\frac{\partial \sigma}{\partial t} = 0$ . En revanche, cela complexifie l'écriture des différentes vitesses et équations et cela nécessite de mettre en place des matrices de passage entre le repère *TFC* et le repère Cartésien.

### WindNinja

Dans ce domaine des modèles diagnostiques, un logiciel open source appelé **WindNinja** est disponible [5]. Il a été utilisé pour réaliser plus de 7 millions de simulations dans plus de 30 pays en 2018 [33]. Il a été développé par le RMRS Missoula Fire Science Laboratory pour la prédition de la propagation des feux de forêts avec l'objectif de pouvoir l'utiliser sur des ordinateurs peu puissants avec un temps de calcul faible. Depuis la dernière mise à jour en mai 2019, il propose deux solveurs pour le calcul : un solveur Mass-Conservative et un solveur CFD. Dans les deux cas les paramètres d'entrée minimum sont : un fichier numérique de terrain, le type de végétation et une donnée initiale sur le vent à 10 m du sol. Cette donnée initiale peut-être un vent moyen pour l'ensemble du domaine, les données de stations météo ou des prévisions météorologiques pour la zone. En sortie, **WindNinja** retourne le vent sur l'ensemble du domaine et sur 20 couches réparties en moyenne entre 1.92 m au dessus du sol et 931 m au dessus du sol [34].

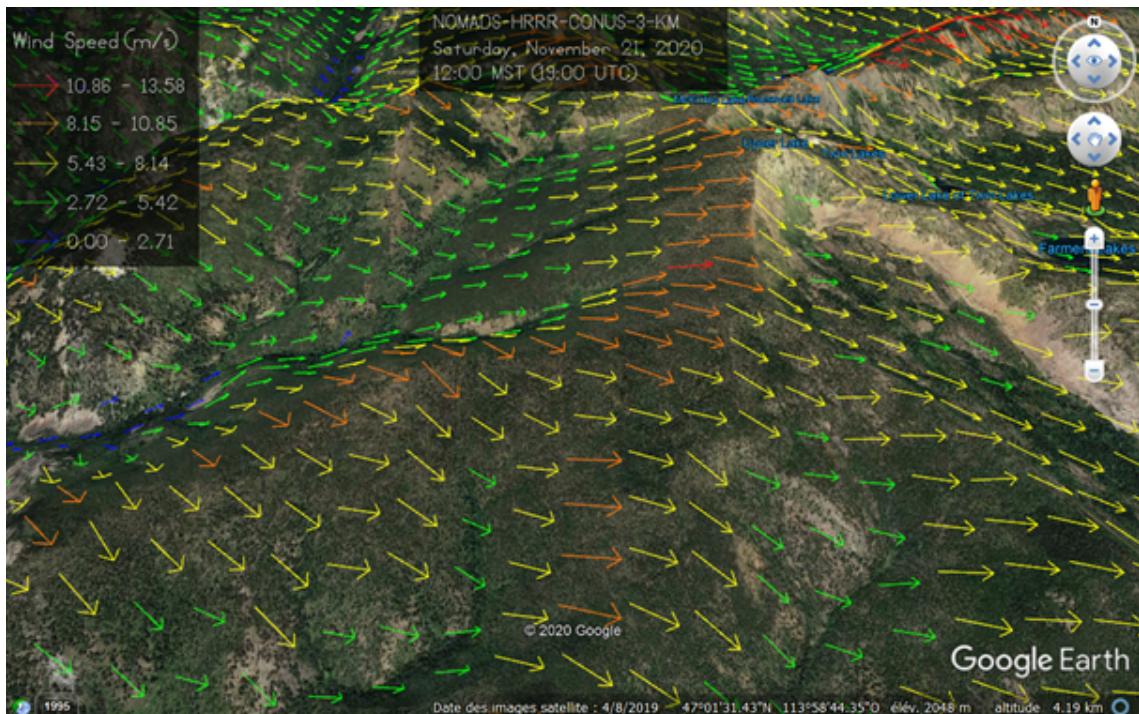


FIGURE 5 – Exemple de calcul réalisé par **WindNinja** à partir de prévisions météo NOMADS-HRRR

Les différentes spécificités de **WindNinja** son résumées dans le tableau ci-dessous :

	options		nombre de mailles		
<b>solveur</b>	diurnal slope flow	non-neutral stability	25k	50k	100k
<b>Mass-Conservative CFD</b>	oui	oui	10s	25s	35s
	oui	non	1h	2h	3h

TABLE 4 – Résumé des options et temps de calcul en fonction du solveur

Le solveur *Mass-Conservative* utilise la technique *Variational Calculus* décrite à la partie 1.1.2. Les modules de Gauss sont fixés tous les deux à 1 pour modéliser une atmosphère neutre. La résolution est faite par une méthode Elements Finis en utilisant une formulation faible et la méthode de Galerkin pour la résolution de (14). Le système d'équations obtenues sur le domaine est résolu par la méthode du gradient conjugué [12].

Le solveur CFD utilise le code *OpenFoam* avec une méthode SIMPLE pour approximer les équations RANS. Une méthode Volumes Finis avec un schéma *linear upwind* est utilisée pour la résolution des équations. Le modèle de turbulence est le modèle  $k - \varepsilon$  standard [33]. Il est à noter que ce solveur a été implémenté tout récemment et pourra être la cible de nouvelles améliorations dans les mois qui viennent.

Pour les deux solveurs, un maillage *TFC* avec des cellules hexahédrales est utilisé. La taille des cellule augmente avec l'altitude de façon à avoir un raffinement près du sol [12].

Deux options de micro-météo sont disponibles pour affiner les résultats du solveur :

- **diurnal slope flow** permet de prendre en compte les ascendances sur les pentes pendant la journée. Elle nécessite la définition de l'heure et de la date, de la couverture nuageuse et de la température en plus des autres paramètres. Pour le solveur CFD, les valeurs de corrections sont calculées à partir d'un calcul avec le modèle **MC** de la configuration
- **non-neutral stability** modification de la stabilité de l'atmosphère en fonction du gradient de température.

### 1.1.3 Méthodes spectrales

Les méthodes spectrales présentées ci-dessous essaient d'approcher la turbulence en décomposant la perturbation selon plusieurs modes élémentaires. Les deux modèles développé sont essentiellement *hybrides*, à la fois déterministes et statistiques, dans la mesure où ils font intervenir une part d'aléatoire dans la reconstitution du vent turbulent.

#### Spectres de turbulences

La connaissance du spectre de turbulence permet de comprendre comment est répartie l'énergie de la composante fluctuante ( $\tilde{u}, \tilde{v}, \tilde{w}$ ) de la vitesse en terme de fréquences. L'utilisation de ce spectre de puissance (PSD, *Power Spectral Density*) suppose cependant que la direction du vent est déjà connue. Les modèles de Kaimal [13] qui prolongent les études pionnières de Dryden et Von Karman et qui prennent en compte le terrain, font référence. Pour la composante turbulente longitudinale  $\tilde{u}$  selon [17], [6] et [20] :

$$\frac{S_u(f)}{\sigma_u^2} = \frac{4\ell_u^x}{\bar{U}(z)} \frac{1}{\left(1 + 70.7 \left(\frac{f\ell_u^x}{\bar{U}(z)}\right)^2\right)^{5/6}} \quad (16)$$

où  $\sigma_u$  est l'écart-type de la composante  $u$  et  $\ell_u^x$  est homogène à une longueur. Naturellement, ce sont deux paramètres qui sont délicats à connaître. Une méthode inverse qui consiste à mesurer le spectre du vent et ajuster ensuite ces paramètres en prévision de leur utilisation peut être envisagée. Ceci doit être fait pour les trois composantes  $\tilde{u}$ ,  $\tilde{v}$ ,  $\tilde{w}$ . Les sources [25], [7] et [3] utilisent toujours la même forme, mais les valeurs numériques retenues sont parfois différentes<sup>4</sup>.

Pour le vent latéral et vertical ( $k \in \{v, w\}$ ), [20] utilise un modèle similaire :

$$\frac{S_k(f)}{\sigma_k^2} = \frac{4\ell_k^x}{\bar{U}(z)} \frac{1 + 188.4 \left(2 \frac{f\ell_k^x}{\bar{U}(z)}\right)^2}{\left(1 + 70.7 \left(2 \frac{f\ell_k^x}{\bar{U}(z)}\right)^2\right)^{11/6}} \quad (17)$$

[7] propose précisément de prendre pour  $\ell_u$  et  $\sigma_u$  les valeurs :

$$\begin{aligned} \ell_u = \ell_v &= \frac{z}{(0.177 + 0.00823z)^{1.2}} & \sigma_u = \sigma_v &= \frac{U_*}{(0.177 + 0.00823z)^{0.4}} \\ \ell_w &= z & \sigma_w &= U_* \end{aligned} \quad (18)$$

La connaissance de ce spectre en puissance, en utilisant une transformée de Fourier inverse, devrait alors permettre de retrouver l'évolution temporelle du vent comme l'expliquent l'article [3] ou le lien [29]. La figure 6 compare ainsi deux profils de vent, dans le domaine fréquentiel et temporel. La conversion se fait ainsi, en choisissant un nombre  $N$  de modes, une fréquence de seuil  $f_s$  et un temps  $T = \frac{N}{f_s}$  :

- Pour chaque entier  $l$  entre 1 et  $N$ , on définit  $\sigma_{X_l}^2 = \frac{T}{2\pi} S_u(f_l)$  où  $f_l < f_s$ .
- Une variable aléatoire  $X_l$  suivant la loi  $\mathcal{N}(0, \sigma_{X_l}^2)$  est générée.
- La prise de la partie réelle de la transformée de Fourier rapide inversée du signal permet de retrouver la valeur temporelle du vent selon :

$$u_t = \bar{U} + \frac{2\pi f_s}{N} \sum_{l=1}^N X_l \cos\left(\frac{2\pi l}{N} t\right) = \bar{U} + 2\pi f_s \Re(\mathcal{F}^{-1}(X)(t)) \quad (19)$$

Une des difficultés est ici de connaître la valeur de  $N$  pour tronquer le signal. Numériquement, cette méthode s'avère en chaque point très avantageuse en terme de temps, une fois que la composante moyenne  $\bar{U}$  est calculée.

### Introduction à la cohérence spatiale

Ce qui suit permet de simuler un vent turbulent spatialement corrélé et est expliqué par les sources [21], [20] et [6] en détail. Même si le travail ne s'oriente *a priori* pas vers ces méthodes, l'esquisse de leur présentation nous a semblé pertinente.

**Fonctions de cohérence** Les rafales ont une taille qui est limitée : celles-ci peuvent ne pas être présentes sur toute la trajectoire du parachute. Aussi, techniquement, le parachute peut être soumis à des efforts qui sont alors différents en fonction de son enveloppement par la rafale : les maxima de vitesse ne sont pas présents sur toute la voilure et les charges sont donc diminuées si la taille des rafales est petite devant la taille de notre objet volant. Les fonctions de cohérence et de corrélation sont alors utilisées pour prendre en compte

4. Le coefficient 70.7 change, tout comme les valeurs de  $\ell_u$  et  $\sigma_u$ , parfois simplement pris égaux à  $z$  et  $U_*$  comme pour  $S_w$ .

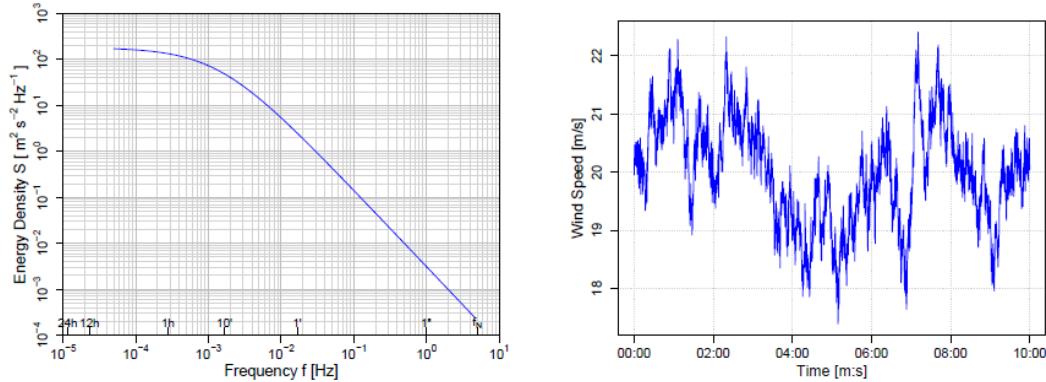


FIGURE 6 – A gauche, le spectre en fréquence dérivée de Von Karman, et à droite la reconstitution d'un vent suivant ces variations [3].

cet aspect localisé. On utilise la *fonction de cohérence exponentielle* entre deux points de l'espace  $P_1(x_1, y_1, z_1)$  et  $P_2(x_2, y_2, z_2)$  :

$$\gamma_k^{1,2}(f) = \exp \left( -2f \frac{\sqrt{C_k^x \Delta_x^2 + C_k^y \Delta_y^2 + C_k^z \Delta_z^2}}{\bar{U}(z_1) + \bar{U}(z_2)} \right) \quad (20)$$

Avec  $k \in \{u, v, w\}$ , où  $C_k^i$  est un coefficient de décroissance exponentielle qui dépend *a priori* du parachute et  $\Delta_i$  désigne la distance selon la coordonnée  $i$  entre les deux points. Si nécessaire, il est ensuite possible de définir la fonction de cohérence croisée.

**Simulation par décomposition bi-orthogonale** L'objectif de la décomposition est d'écrire :

$$\tilde{u}(x, y, z, t) = \sum_{k=1}^{\infty} \alpha_k \psi_k(t) \phi_k(x, y, z) \quad (21)$$

Où les *topos*  $\phi_k$  et les *chronos*  $\psi_k$  forment tous deux une famille orthonormée. [21] mentionne qu'il a été prouvé que pour tout entier  $k$ ,  $\alpha_k$  est simultanément une valeur propre de la matrice de cohérence spatiale (pour le topo  $\phi_k$ ) et une valeur propre de la matrice de corrélation temporelle (pour le chrono  $\psi_k$ ). La matrice de cohérence spatiale se calcule, par exemple, entre les noeuds  $m$  et  $n$  par

$$\mathcal{CS}_k[m, n] = \sum_l \sqrt{S_{k,m}(f_l) S_{k,n}(f_l)} \gamma_k^{1,2}(f_l) \quad (22)$$

Plusieurs inconvénients peuvent apparaître : puisque l'on réalise une superposition, il faut savoir quand tronquer les sommes considérées (celle pour le calcul de la matrice puis pour la décomposition). La superposition, même finie, de tous ces modes nécessite aussi de connaître avec une précision d'autant plus forte tous les coefficients qui interviennent dans les fonctions de cohérence.

#### 1.1.4 Synthèse

Les méthodes présentées ici ont permis de voir comment le vent pouvait être modélisé. Retenons que le modèle diagnostique permet, à partir d'un champ de vent interpolé d'obtenir un champ physiquement cohérent, qui donne alors les composantes de la vitesse en tout

point et dans les trois directions de l'espace. La cohérence du champ est mise en place en implémentant une ou plusieurs équations de Navier-Stokes. Le niveau le plus simple intègre seulement l'équation de continuité dans l'interpolation. Cette méthode dite *Mass Conservative* permet d'obtenir une bonne estimation du champ de vent avec un temps de calcul très court. Cependant des phénomènes locaux comme le décollement ou la recirculation ne peuvent pas être modélisés sans l'intégration en plus de l'équation de *Quantité de Mouvement*. Cette ajout implique l'utilisation de méthode CFD de type RANS, augmentant la complexité du calcul et son temps d'exécution. Notons qu'il existe déjà des solutions open source qui intègrent ces méthodes comme le logiciel WindNinja, créé initialement pour la prédiction de la propagation des feux de forêt.

Le champ de vent connu pourrait être obtenu via des relevés ou des prédictions météorologiques au format GRIB pour le vent de surface, puis extrapolé en altitude grâce à la connaissance du terrain et aux modèles explicites présentés. Cette méthode ne permet cependant pas de faire varier temporellement le vent, c'est à ce moment que pourraient servir les spectres de turbulence pour proposer une évolution possible du vent. La turbulence relevant de phénomènes intrinsèquement aléatoire, l'utilisation de modèles statistiques, présentés ci-dessous, pourrait s'avérer pertinente.

## 1.2 Approche par l'étude statistique

Contrairement à la méthode précédente, l'étude statistique est certainement la méthode la moins coûteuse en terme de temps de calcul mais nécessite un nombre important de données afin d'être précis dans la détermination des paramètres : par exemple, [19] utilise 2024 h de données de vitesse de vent obtenues par une station météorologique, ou encore [27] utilise un jeu de données composées de période de 3 h quotidiennes étalées sur 10 ans, et enfin [1] et [16] utilisent des bases de près de 9000 entrées par différentes stations météorologique.

Cependant, un des problèmes fondamentaux est que cette méthode ne permet de connaître *a priori* que la vitesse du vent et non pas sa direction, même si certains articles, comme [27], proposent une d'établir une rose des vents statistiques.

### 1.2.1 Distributions statistiques

La méthode de la distribution statistique, consistant à trouver la loi statistique la plus pertinente pour modéliser notre jeu de données est certainement une des plus faciles à mettre en place, des langages tels que **R** permettant même une exploitation de ces données très simplifiée.

La littérature nous a montré que de nombreuses lois classiques avaient étaient utilisées : Weibull, Rayleigh, Lomax, Extremum généralisé, Gamma, Weibull inverse, Log-logistique... ([31],[1])

Toutefois, la loi de Weibull à deux paramètres et la loi de Rayleigh semblent être les plus utilisées pour la modélisation de la vitesse du vent car la plus flexible à manipuler.

Cependant, ces lois, bien que facile à calculer, peuvent mener à des erreurs de calculs lorsque nous traitons des distributions de vent à queue lourde (*heavy-tail distribution*, [31]). À ce titre, de nouvelles lois furent créées par association pour essayer de représenter au mieux ces données, comme la Marshall-Olkin extended Lindley distribution ([1]) :

$$f_{c,\sigma}(x) = \frac{c\sigma^2(1+x)\exp(-\sigma x)}{(1+\sigma)\left[1-(1-c)\left(1+\frac{\sigma x}{1+\sigma}\right)\exp(-\sigma x)\right]^2} \quad (23)$$

Modèle	Paramètres	Fonction de densité
Weibull	$\alpha, \beta$	$f_{\alpha,\beta}(x) = \frac{\alpha}{\beta} \left(\frac{x}{\beta}\right)^{\alpha-1} \exp\left(-\left(\frac{x}{\beta}\right)^\alpha\right)$
Rayleigh	$\sigma$	$f_\sigma(x) = \frac{x}{\sigma^2} \exp\left(-\frac{x^2}{2\sigma^2}\right)$
Gamma	$\sigma, c$	$f_{\sigma,c}(x) = \frac{1}{\Gamma(c)\sigma^c} x^{c-1} \exp\left(-\frac{x}{\sigma}\right)$

TABLE 5 – Modèle de distribution

Ou encore la Marshall-Olkin Power Lomax - MOPLx ([31]) :

$$f_{\alpha,\beta,\gamma,\lambda}(x) = \frac{\alpha\beta\gamma\lambda^\alpha x^{\beta-1}(\lambda + x^\beta)^{-\alpha-1}}{[1 - (1 - \gamma)\lambda^\alpha(\lambda + x^\beta)^{-\alpha}]^2} \quad (24)$$

Grâce à cette dernière, nous pouvons retrouver la loi Marshall-Olkin Lomax ( $\beta = 1$ ) ou Power-Lomax ( $\gamma = 1$ ) qui ont été étudiées par de nombreux articles.

L'article [31] a effectué une comparaison des différents modèles les plus communément utilisé, basés sur les critères suivant :

$$\begin{aligned} R^2 &= 1 - \frac{\sum_{i=1}^n \left( \hat{F}(X_{(i)}) - \frac{i}{n+1} \right)^2}{\sum_{i=1}^n \left( \hat{F}(X_{(i)}) - \bar{\hat{F}}(X_{(i)}) \right)^2} \\ RMSE &= \left[ \frac{1}{n} \sum_{i=1}^n \left( \hat{F}_i - \frac{i}{n+1} \right)^2 \right]^{\frac{1}{2}} \\ AIC &= 2K - 2 \log L \end{aligned} \quad (25)$$

Oo :

- $\hat{F}_i$  est la fonction de répartition estimée pour la  $i$ -ème observation ordonnée,
- $\bar{\hat{F}} = \frac{1}{n} \sum_{i=1}^n \hat{F}(X_{(i)})$ ,
- $n$  est la taille de l'échantillon,
- $K$  est le nombre de paramètres,
- $L$  est le maximum de vraisemblance.

Il ressort de cette analyse (détailée dans l'article [31]) que la loi MOPLx est la loi qui modélise le mieux les vitesses du vent (Fig. 7) et l'article [15] détaille quant à lui les différentes propriétés mathématiques de cette loi.

### 1.2.2 Méthodes stochastiques

La méthode stochastique est sans doute celle dont le concept est le plus simple, mais qui nécessite la possibilité de pouvoir stocker plusieurs matrices, dépendant de la taille de la finesse souhaitée du modèle ([16] utilise des matrices 13x13 et 26x26, tandis que [28] utilise des matrices 7x7), et d'effectuer divers calcul sur ces matrices.

Une base de données de vitesse de vent est nécessaire pour calculer les matrices de transition d'états, d'autant plus grandes que les plages de vitesse du vent sont fines.

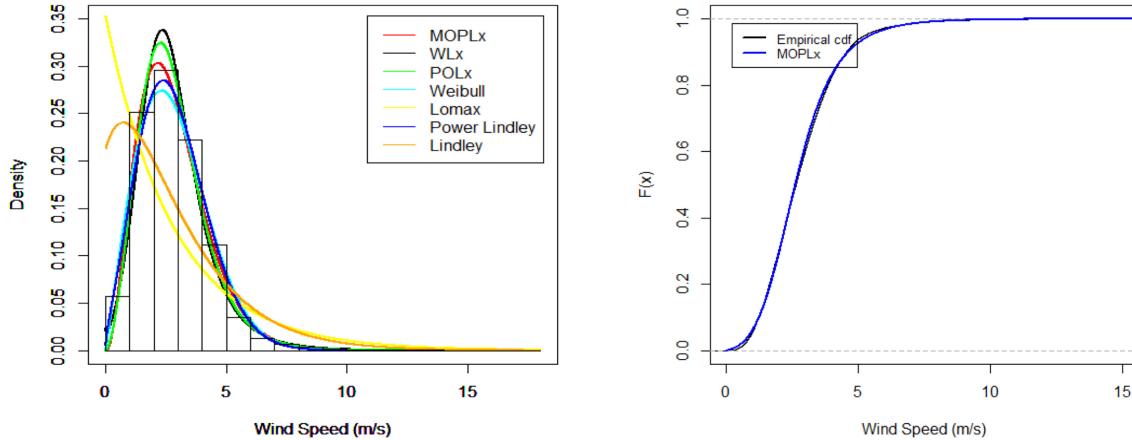


FIGURE 7 – Histogramme et densité des modèles à la station Haripur, Pakistan

Dans le cas de chaîne de Markov simple pour un pas de temps fixé, comme détaillé dans [16], on obtient donc :

$$A = \begin{bmatrix} p_{1,1} & p_{1,2} & \dots & p_{1,n} \\ p_{2,1} & p_{2,2} & \dots & p_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ \vdots & \vdots & \ddots & \vdots \\ p_{n,1} & p_{n,2} & \dots & p_{n,n} \end{bmatrix} \quad (26)$$

avec les probabilités  $p_{i,j}$  de (26) calculées à partir de :

$$p_{i,j} = \frac{m_{i,j}}{\sum_{j=1}^n m_{i,j}} \quad (27)$$

où  $m_{i,j}$  représente le nombre de transition de l'état  $i$  vers l'état  $j$ .

Cependant, dans ce cas-ci, l'état dans lequel on se trouve ne dépend que de l'état précédent.

L'article [9] présente quant à lui une modélisation plus complexe avec des chaînes semi-markoviennes du premier et second ordre (l'état futur est influencé respectivement par l'état présent et par l'état présent et précédent).

Les résultats sont présentés dans les graphiques suivants :

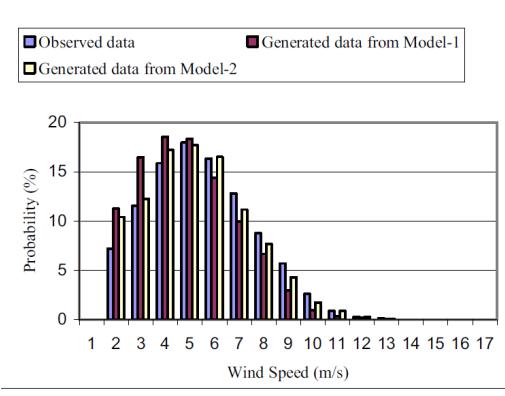


FIGURE 8 – Résultats [16]

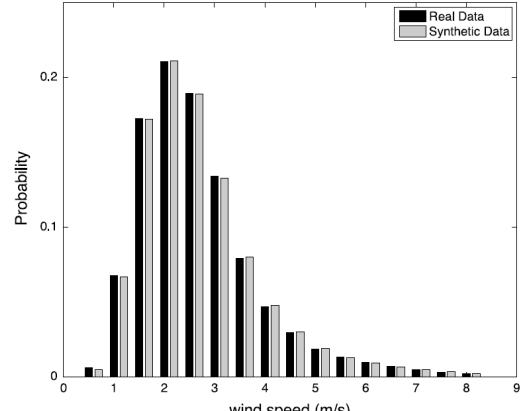


FIGURE 9 – Résultats [9]

Dans Fig. 8, Model-1 correspond à 13 états possibles pour la vitesse du vent, et Model-2 à 26.

On peut donc en conclure logiquement qu'au plus la finesse de la plage souhaitée est fine au plus la modélisation est précise, même si les chaînes semi-markoviennes sont plus précises.

### 1.2.3 Séries temporelles

Cette méthode est consacré à la prédiction de la vitesse du vent, ce qui peut faire défaut à une méthode comme celle détaillée dans la partie 1.2.1 où on ne prédit aucune vitesse.

Dans les algorithmes détaillés dans [19] et [4], les auteurs utilisent des séries temporelles auto-régressives d'ordre 2 :

$$x(t) = \phi_{1,k}x(t-1) + \phi_{2,k}x(t-2) + \epsilon(t) \quad (28)$$

Où  $\phi_1, \phi_2$  sont les coefficients d'auto-régressions inconnus, et  $\epsilon \sim \mathcal{N}(0, \sigma_\epsilon)$ , un bruit blanc gaussien, avec  $\sigma_\epsilon$  estimé dans [4]-2.c par la méthode de *Yule-Walker recursion*.

Néanmoins, dans l'article [4] aucun modèle n'avait été développé à l'époque de sa publication pour des variables aléatoires avec une distribution de Weibull (la plus commune selon [31]) en utilisant une similarité de la fonction de Weibull avec un paramètre de forme de 3, 6 et une Gaussienne.

Dans [19], ce problème est contourné en utilisant un filtre de Kalman pour estimer les futures valeurs.

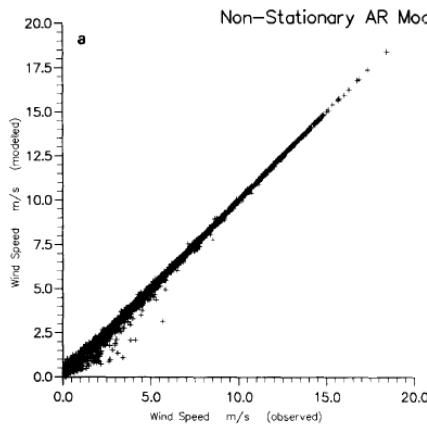


FIGURE 10 – Vitesse de vent estimée VS observée pour une prédition de 0h

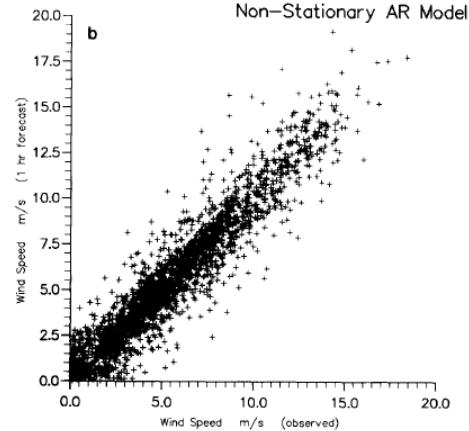


FIGURE 11 – Vitesse de vent estimée VS observée pour une prédition de 1h

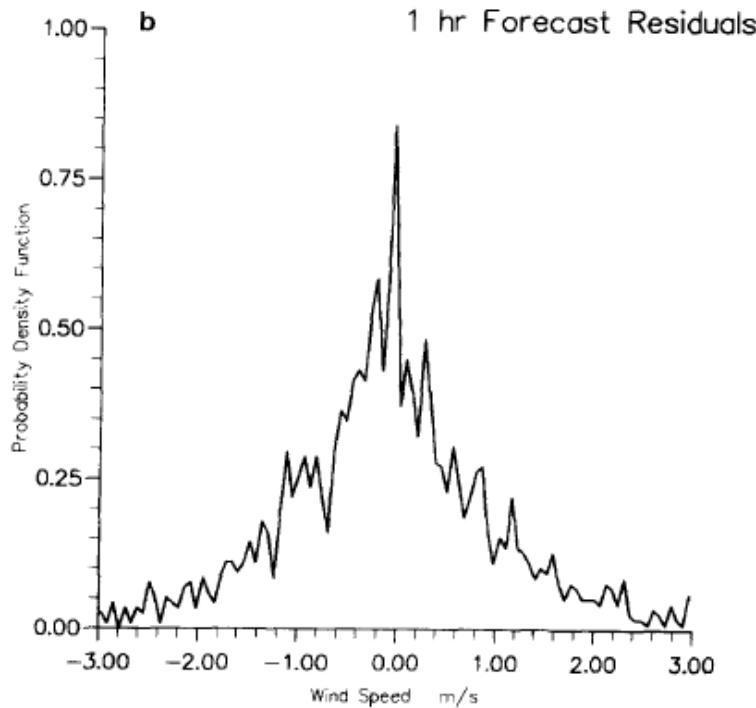


FIGURE 12 – Erreur résiduelle pour une prédiction de 1h

On observe donc une assez bonne modélisation instantanée du vent (Fig. 10 et Fig. 11), mais une dispersion de plus en plus importante due à une erreur résiduelle croissante (Fig. 12).

#### 1.2.4 Réseaux de neurones

Cette méthode a montré des résultats très prometteurs dans les différents articles que nous avons étudiés ([26], [2] et [32]), mais des niveaux de compétences algorithmiques que nous ne maîtrisons pas pour le moment, ils ont donc été temporairement mis de coté afin de se concentrer sur des méthodes statistiques plus traditionnelles.

Cependant, il nous a semblé important de notifier de la présence de ces méthodes et de les référencer.

#### 1.2.5 Caractéristiques des données

Lors des différentes analyses statistiques effectuées, de nombreux phénomènes de saisonnalité ont été observées, notamment sur une journée (Fig. 13), ou que certaines directions étaient privilégiées au cours du temps (Fig. 14).

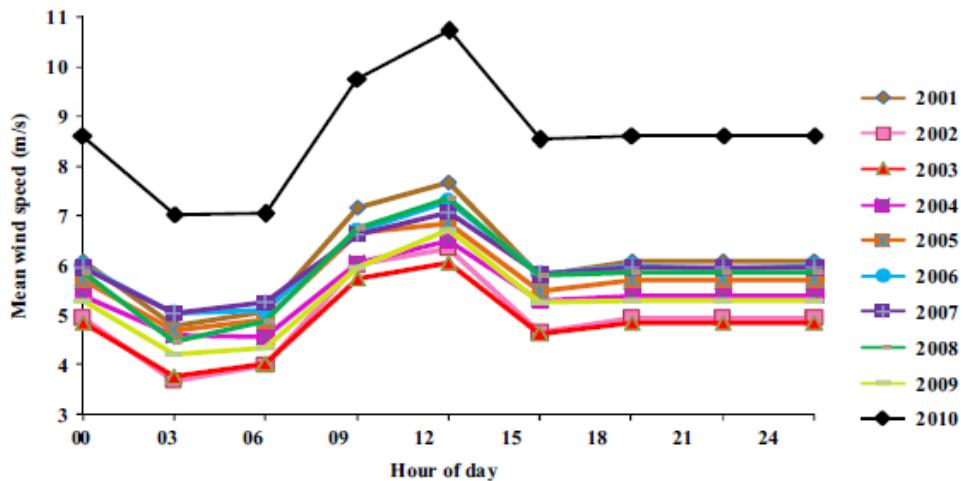


FIGURE 13 – Variation journalière de la vitesse de la vitesse du vent dans la région de Firouzkooch (Iran) - [27]

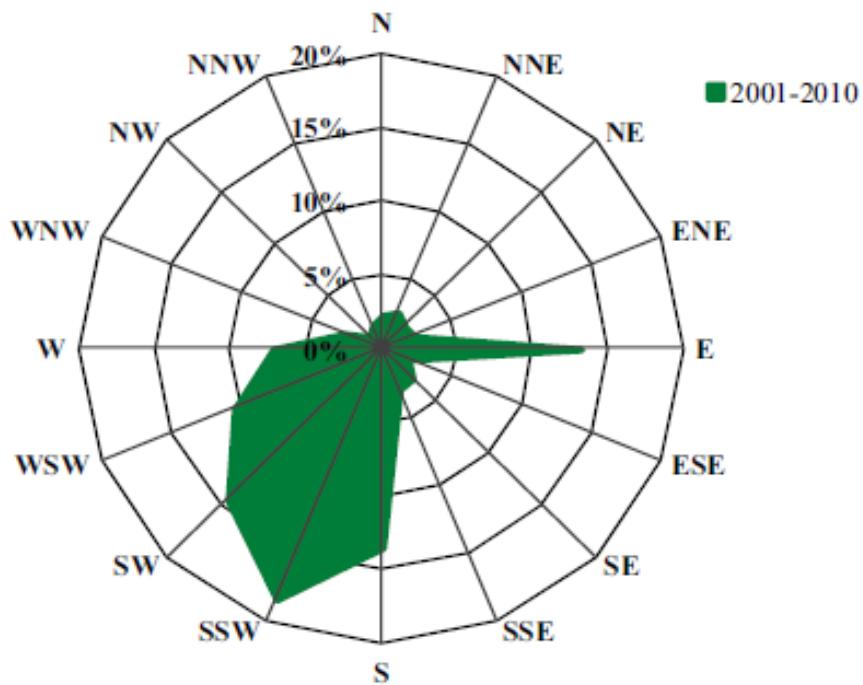


FIGURE 14 – Rose des vents dans la région de Firouzkooch (Iran) - [27]

### 1.2.6 Synthèse

Ces modèles présentent donc de nombreux avantages : rapidité de calculs pour des méthodes statistiques simples (1.2.1, 1.2.2) ou alors grande adaptabilité de prévision (1.2.3, 1.2.4) s'ils sont alimentés par de nombreuses données. Cependant, il s'agit d'une modélisation qui manque de dynamisme et de prise en compte de morphologie du terrain, en se basant uniquement sur des données de vent mesurées à des endroits fixes. Elle n'est pas assez adaptée à la modélisation des vitesses dans un hyper-cube, mais plus à la génération

de champ de vent *type*, étant donné une topographie générique de lieu et de comportement de vent, une fois combiné avec des méthodes pour étendre la mesure du vent à toutes les dimensions (1.1.1).

## 2 Démarche retenue

Afin de répondre à la problématique, il a été décidé de se baser sur des résultats calculés par **WindNinja** et de rajouter par-dessus des "sur-couches" de codes permettant de correspondre parfaitement au cahier des charges établi. Le calcul est alors décomposé en deux étapes :

1. Calcul d'un premier cube de vent (appelé `wind_cube`) par **WindNinja** à partir de données de vent et de terrains spécifiées par l'utilisateur ou téléchargées directement par le logiciel.
2. Extrapolation du cube jusqu'à une élévation a.g.l définie par l'utilisateur.

Ce calcul sera effectué par une nouvelle classe Python appelée `wind_class` qui permettra en outre de réaliser diverses opérations d'interpolation et de modélisation de la turbulence au sein du `wind_cube`. La théorie utilisée par les "sur-couches" d'extrapolation et de turbulence est explicitée dans ce qui suit.

### 2.1 Extrapolation en altitude

Une fois que le calcul d'un premier champ de vent a été réalisé par **WindNinja**, il est nécessaire de réaliser une extrapolation en altitude. En effet, **WindNinja** calcule le vent jusqu'à une élévation a.g.l d'environ 900 m alors que le cahier des charges impose un champ de vent jusqu'à 4000 m a.g.l.

Pour réaliser cette extrapolation, nous nous basons sur les lois empiriques présentées au sein de la bibliographie (1.1.1). Les figures suivantes comparent différents types d'extrapolation sur les trois composantes du vent ainsi que sur sa norme. Elles ont été réalisés à partir de résultats **WindNinja** avec en entrée un vent à 10 m a.g.l venant du Nord pour une vitesse de 36 km/h.

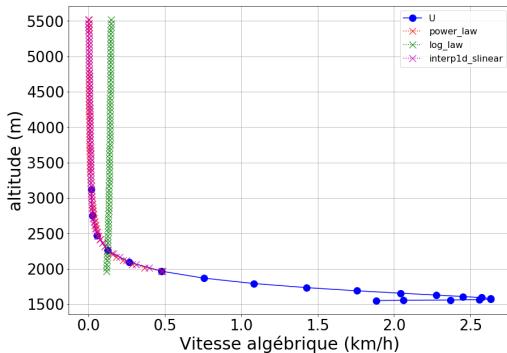


FIGURE 15 – Extrapolations pour la composante  $u$

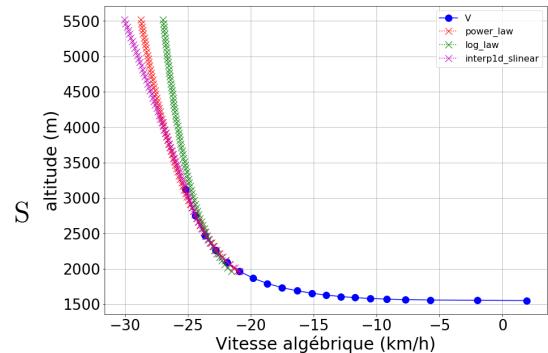


FIGURE 16 – Extrapolations pour la composante  $v$

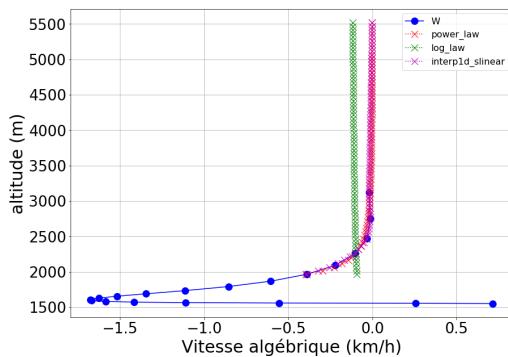


FIGURE 17 – Extrapolations pour la composante  $w$

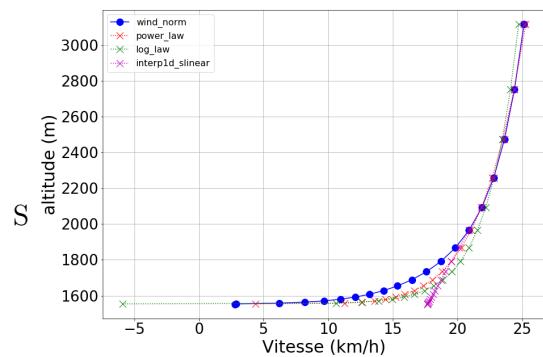


FIGURE 18 – Extrapolations pour la norme

On remarque sur ces figures que l'extrapolation à partir de la loi logarithmique ne marche que pour la norme du vent ou la composante  $v$  qui correspond à la composante principale du vent de notre cas test. Cela est cohérent au vu de la formule de la loi logarithmique  $\bar{U}(z) = \frac{U_*}{\kappa} \ln \left( \frac{z-z_h}{z_0} \right)$  qui ne permet pas d'avoir une vitesse qui tend vers 0 lorsque  $z$  tend vers les hauteurs "infinies". En revanche, la loi puissance semble capable de réaliser une interpolation correcte pour tous les cas. Elle sera donc privilégiée à une simple extrapolation linéaire car elle est connue dans la littérature pour permettre une meilleure représentation du comportement asymptotique du vent.

Reste alors à fixer le nombre de points à partir desquels sera réalisé la correspondance de la loi puissance pour l'extrapolation. Il s'agit ici de prendre en compte suffisamment de points pour obtenir le comportement asymptotique de la vitesse du vent tout en s'affranchissant des effets du terrain comme un éventuel point de rebroussement (cf. Figures 15 et 17) en enlevant les points les plus proches de la surface. En théorie, la loi puissance ne dépend que de deux paramètres et une extrapolation pourrait donc être réalisée à partir de seulement deux points. C'est ce que nous avons décidé de faire ici. Ce *fitting* est réalisé à partir de la fonction `curve_fit` de la librairie `Scipy`.

## 2.2 Composante temporelle : turbulence

Nous précisons dans la suite comment l'aspect temporel, basé sur les statistiques de la turbulence, a été incorporé à notre modèle pour pouvoir être utilisé dans l'entraînement d'un correcteur.

### 2.2.1 Analyse préliminaire de données expérimentales

**Cadre des mesures** Nous avons contacté Madame Sara ALAOUI-SOSSE, doctorante au sein de l'ISAE-SUPAERO et de l'ONERA, qui disposait de mesures expérimentales du vent, réalisées à Lannemezan. Le laboratoire d'aérologie qui s'y trouve dispose d'un mât d'une hauteur de 60 m sur lequel le vecteur vitesse du vent était observé, à trois hauteurs différentes (30 m, 45 m et 60 m). Réalisées le 24 mars 2016 avec une fréquence de mesure de 10 Hz pendant 24 heures, ces mesures nous ont été précieuses, à la fois pour confirmer nos modèles, pour corriger quelques une des constantes, et pour réaliser des essais.

A chaque instant  $t$ , on dispose<sup>5</sup> de la composante Est-Ouest, Nord-Sud et verticale du

5. Ces données ne sont pas fournies dans les livrables : il faut d'abord remplir un accord sur le site <http://p2oa.aero.obs-mip.fr/spip.php?article202> pour pouvoir les utiliser.

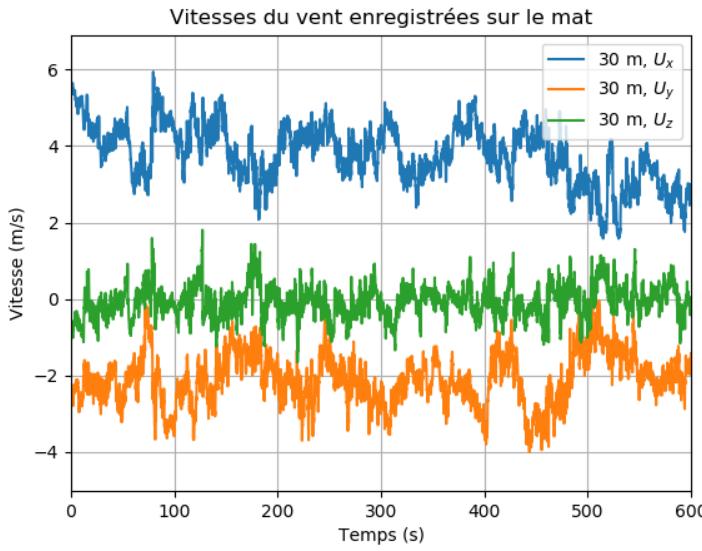


FIGURE 19 – Evolution des trois composantes du vent pendant une durée de 10 minutes.

vent. Cela nous permet alors de déterminer l'intensité du vent, sa direction en partant du principe que la moyenne est prise sur une durée de 10 minutes.

**Estimation des densités spectrales de puissance** Les données ont d'abord été utiles pour calculer leurs densités spectrales de puissance (DSP). Nous avions des modèles théoriques mais nous voulions voir dans quelle mesure ceux-ci étaient adaptés, ou alors comment les corriger. L'estimation de la DSP est néanmoins relativement difficile dans la pratique en raison du caractère aléatoire des mesures.

La DSP  $S_x$  d'un signal  $x$  s'écrit comme la transformée de Fourier de son autocorrélation (théorème de Wiener-Khintchine) :

$$S_x(f) = \mathcal{F}(\mathcal{R}_{xx})(f) = \int_{-\infty}^{+\infty} R_{xx}(t) e^{-2\pi i f t} dt$$

$$S_x(f) = \int_{-\infty}^{+\infty} \left( \int_{-\infty}^{+\infty} x(s+t)x(s) ds \right) e^{-2\pi i f t} dt$$

Néanmoins, le calcul des auto-corrélations et sa transformée de Fourier n'ont pas donné de résultats convaincants. Nous avons donc du passer par une approximation de la DSP qui exploite la méthode des périodogrammes de Lomb-Scargle. Pour ce faire, on utilise la propriété d'approximation :

$$S_x(f) = \lim_{T \rightarrow +\infty} \frac{\mathbb{E}(|\mathcal{F}(x|_{[0,T]})|^2)}{T}$$

Dans notre cas, il ne serait pas adapté de chercher à prendre une valeur  $T$  longue : la moyenne n'a de sens que sur des périodes inférieures à 15 minutes. Concrètement, nous choisissons donc une durée  $T = 10$  minutes puis observons le signal n° $k$  sur le segment  $[t_k, t_k + T]$  (avec recouvrement possible des valeurs  $t_k$ ). Pour chacun de ces échantillons, on en déduit  $S_{x,k} = \frac{|\mathcal{F}(x|_{[t_k, t_k + T]})|^2}{T}$  qui correspondrait à la DSP de la seule donnée n° $k$ . On réalise alors ceci pour  $N = 50$  échantillons et on en prend la moyenne, que l'on peut lisser. Cela fournit enfin la DSP de notre acquisition.

Le "signal" à considérer pour nous correspond à la composante turbulente dans le repère principal / latéral. Ainsi, on passe du champ de vent  $(u, v, w)$  à  $(\tilde{u}, \tilde{v}, \tilde{w})$  en calculant la direction moyenne  $\theta$  et la valeur moyenne dans la direction principale  $\bar{U}$  sur la durée  $T$ , puis en réalisant la rotation idoine.

Ci-dessous (Figure 20) se trouve par exemple le spectre correspondant à la coordonnée principale  $\tilde{u}$ . On y voit le spectre interpolé et lissé, et les autres spectres théoriques.

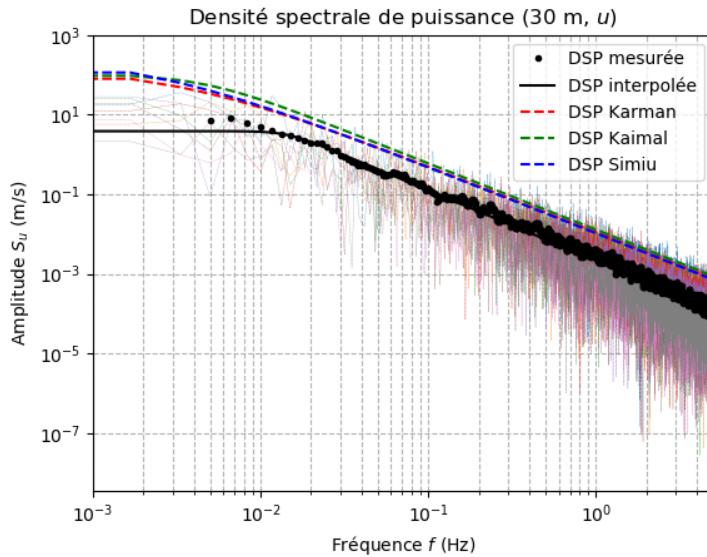


FIGURE 20 – Comparaison des densités spectrales : mesure, interpolation, théories.

Ci-dessous se trouvent également les deux jeux de données pour les composantes verticale (Figure 22) et latérale (Figure 21).

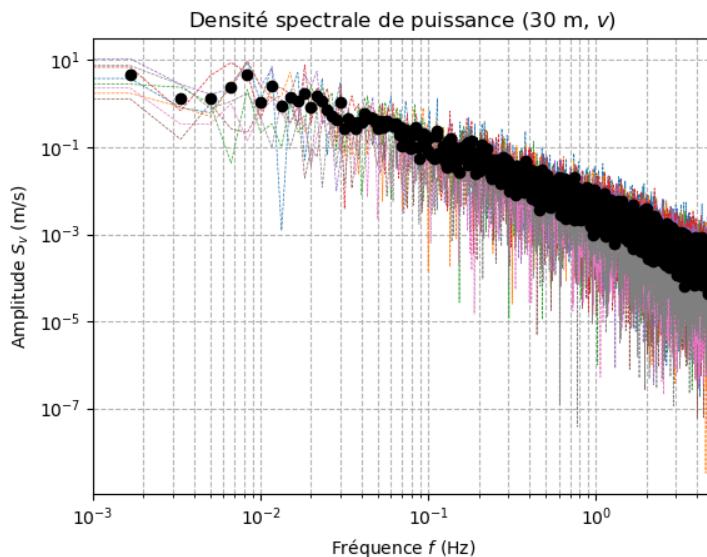


FIGURE 21 – Densité spectrale de puissance du vent latéral.

Nous constatons que les spectres théoriques concordent plutôt bien avec le spectre interpolé. Plus que l'*offset* qui est ici une composante qui peut être réglée, le plus important

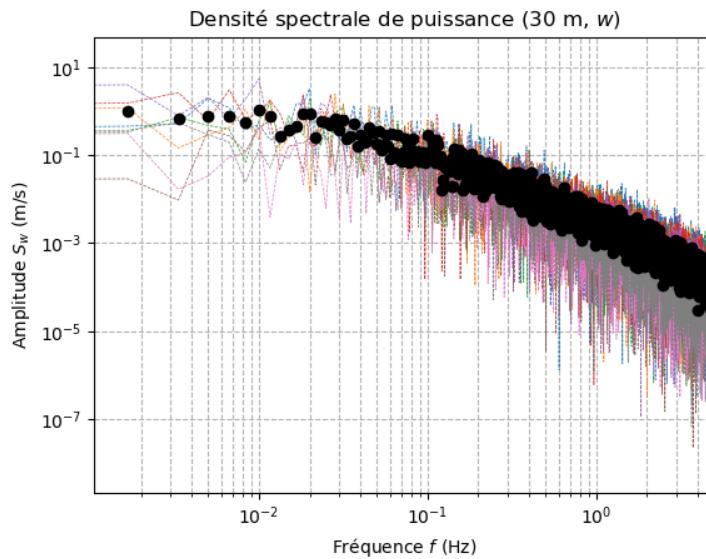


FIGURE 22 – Densité spectrale de puissance du vent vertical.

consiste à regarder les pentes pour les "hautes" fréquences ainsi que la fréquence de coupure. Les données mesurées concordent avec les modèles estimés.

Néanmoins, les trois hauteurs (Figure 23) étant assez proches, nous ne sommes pas parvenus à mettre en évidence un apport net de la hauteur de la mesure sur les densités spectrales. Nous nous fierons donc aux modèles théoriques sur cet aspect.

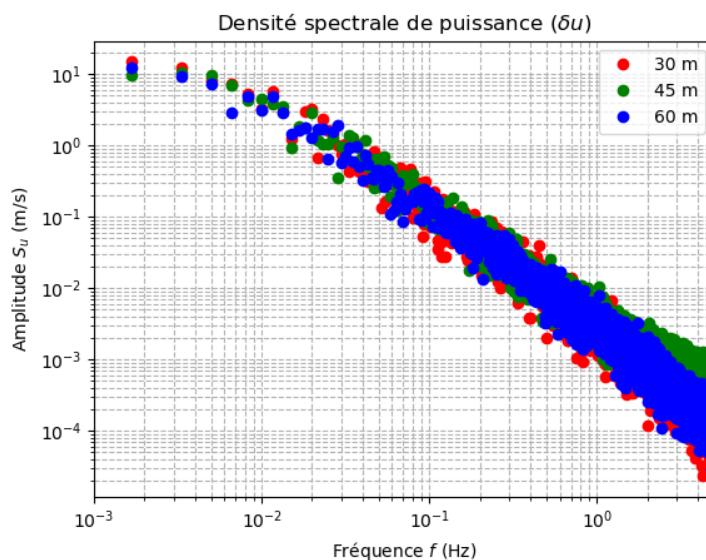


FIGURE 23 – Densité spectrale de puissance en fonction de l'altitude mesurée.

### 2.2.2 Incorporation de la turbulence

On précise ici comment est calculé le vent turbulent en *un point* du cube de vent.

**Utilisation de spectres** Nous utilisons des spectres hybrides, déterminés par Kaimal et corrigés par Coyle et Wickenheiser. Les formules utilisées sont les suivantes :

$$\frac{S_u(f)}{\sigma_u^2} = \frac{4\ell_u^x}{\bar{U}(z)} \frac{1}{\left(1 + 70.7 \left(\frac{f\ell_u^x}{\bar{U}(z)}\right)^2\right)^{5/6}}$$

$$\frac{S_k(f)}{\sigma_k^2} = \frac{4\ell_k^x}{\bar{U}(z)} \frac{1 + 188.4 \left(2 \frac{f\ell_k^x}{\bar{U}(z)}\right)^2}{\left(1 + 70.7 \left(2 \frac{f\ell_k^x}{\bar{U}(z)}\right)^2\right)^{11/6}}$$

$$\ell_u = \ell_v = \frac{z}{(0.177 + 0.00823z)^{1.2}} \quad \ell_w = z$$

Les graphiques des Figures 24 et 25 permettent de voir l'influence de la vitesse et de l'altitude sur l'amplitude de la turbulence. Afin de prendre en compte le fait que la turbulence diminue sensiblement avec la hauteur, nous introduisons un coefficient supplémentaire en multipliant  $S_u$  par  $\min\left\{\frac{\eta_z - 1}{1000 - 0}z + 1, \eta_z\right\}$  (avec  $\eta_z = 0.1$ ).

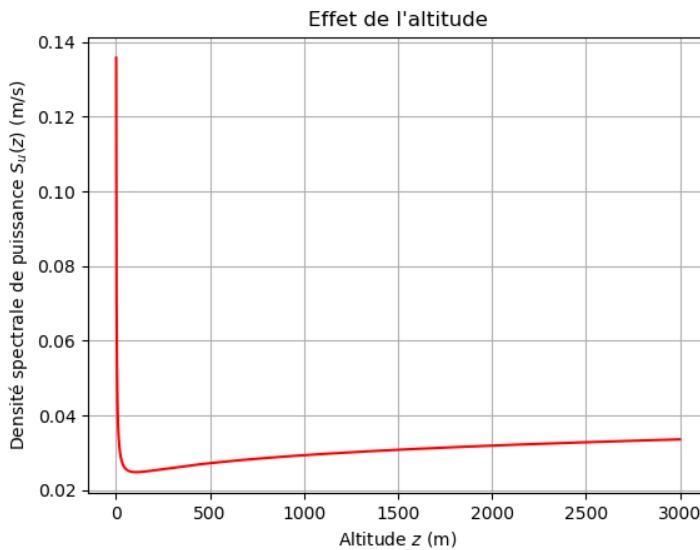


FIGURE 24 – La composante turbulente est d'autant plus forte qu'on est proche du sol puis elle augmente quelque peu avec l'altitude.

Nous utilisons une fréquence de référence  $f_s = 10$  Hz. Le vent turbulent va s'écrire comme une somme de signaux sinusoïdaux aux amplitudes aléatoires. Étant donné un nombre  $N = T f_s$  de modes, on va calculer chacune des composantes aléatoires de la façon suivante :

- Pour chaque entier  $k$  entre 1 et  $N$ , on définit la fréquence d'étude  $f_k = \frac{k}{2N} f_s$  et l'écart-type  $\sigma_{X_k}^2 = \frac{T}{2\pi} S_u(f_k)$  (pour la perturbation principale du vent).
- Une valeur aléatoire  $X_k$  suivant la loi  $\mathcal{N}(0, \sigma_{X_k}^2)$  est alors générée.
- La prise de la partie réelle de la transformée de Fourier rapide inversée du signal permet de retrouver la valeur temporelle du vent selon la formule :

$$x_t = \bar{U} + \frac{2\pi f_s}{N} \sum_{k=1}^N X_k \cos\left(i \frac{2\pi k}{N} t\right) = \bar{U} + 2\pi f_s \Re(\mathcal{F}^{-1}(X)(t))$$

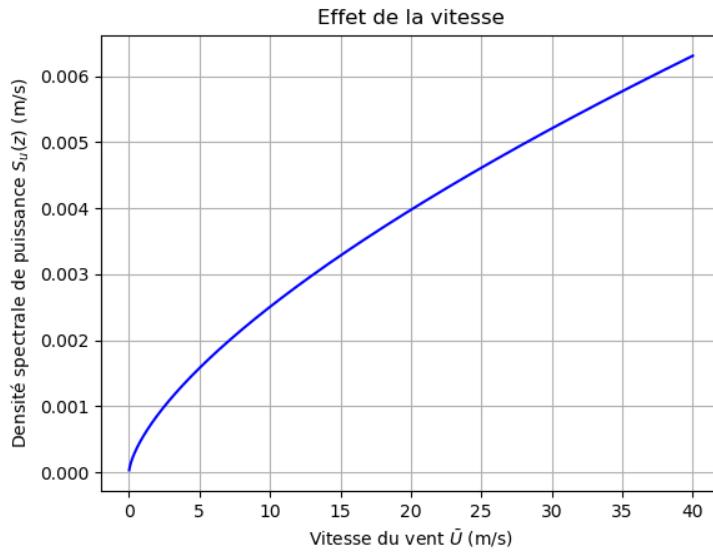


FIGURE 25 – La composante turbulente est d'autant plus forte que la vitesse stationnaire est élevée.

Ci-dessous, on peut voir sur les Figures 26 et 27 un profil de vent généré par nos simulations.

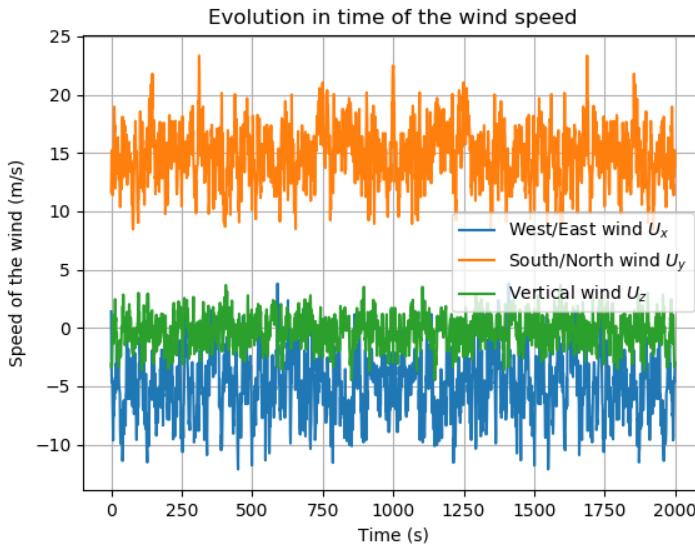


FIGURE 26 – Vent turbulent généré par les méthodes spectrales dans le repère géographique.

**Validation réciproque** Afin de s'assurer de la pertinence du code, et pour le corriger si nécessaire, nous avons cherché à estimer la DSP de notre propre vent turbulent. Comme nous connaissons celle qui a servi de support pour le générer, nous devrions la retrouver. En moyennant sur une série de 13 mesures (voir la Figure 28), on observe bien une concordance. Cette similitude s'observe à nouveau principalement sur le profil et non pas sur l'amplitude,

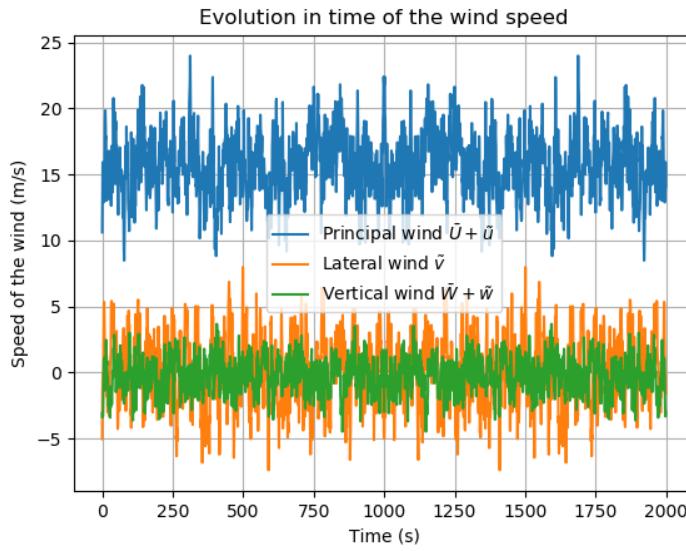


FIGURE 27 – Vent turbulent généré par les méthodes spectrales dans le repère du vent.

qui va dépendre entre autres de l'altitude  $z$ , et qui est un des paramètres sur lequel nous pouvons jouer.

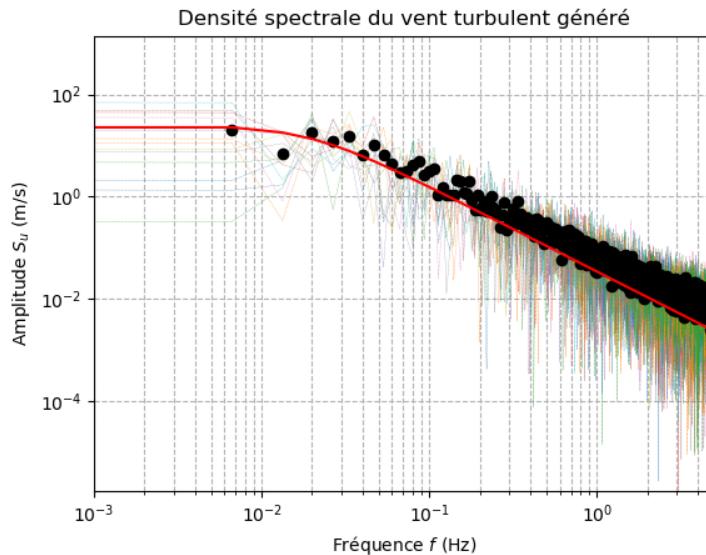


FIGURE 28 – Densité spectrale de puissance du signal généré.

### 2.2.3 Gestion de la cohérence spatiale

Pour pouvoir proposer un champ de vent prenant en compte la turbulence en 3D, nous nous appuyons sur une méthode assez proche de l'utilisation de spectres, mais qui introduit une part d'aléatoire de manière différente. Cette méthode se base sur la synthèse de Cole et Wickenheiser [8], ainsi que sur les travaux de Forstall [11]. Leur idée consiste à introduire une *fonction d'étalement* notée  $D_s(\theta) = D_0 \cos^{2s}(\theta)$  satisfaisant entre autres

$\int_{-\pi}^{\pi} D_s(f, \theta) d\theta = 1$  et qui va venir moduler l'amplitude du spectre.

Si nous notons  $\varphi$  un angle permettant de caractériser le sens de propagation du vent<sup>6</sup>, et que nous notons  $(x_u, y_u, z)$  les coordonnées d'un point  $(x, y, z)$  après la rotation d'angle  $\varphi$ , on peut écrire la composante turbulente dans la direction  $k \in \{u, v, w\}$  en ce point selon :

$$u_k(x, y, z, t) = \int_{f_0}^{f_{max}} \int_{-\pi/2}^{\pi/2} \sqrt{2D_s(\theta)S_k(f, \bar{U}, z)} \dots \\ \cos\left(\frac{2\pi f}{\bar{U}_{10}}(x_u \cos(\theta) + y_u \sin(\theta)) - 2\pi ft + \psi(\theta, f)\right) d\theta df$$

La fonction  $\psi$  est une fonction aléatoire prenant ses valeurs uniformément dans  $[0, 2\pi]$ . Par rapport aux méthodes spectrales décrites précédemment, l'aléatoire ne joue donc que sur la *phase* et non plus sur l'*amplitude*. Une telle écriture permet de comprendre que le vent turbulent se propage, ce qui assure une bonne cohérence du modèle. Nous discrétisons les fréquences de  $f_0 = 0$  jusqu'à  $f_{max} = \frac{f_s}{2}$  et faisons le choix de prendre  $s = 1$ , ce qui impose aussi  $D_0 = \frac{2}{\pi}$  et qui conduit à poser :

$$u_k(x, y, z, t) = \sum_{p=1}^{N_f} \sum_{i=1}^{N_\theta} \sqrt{\frac{4}{\pi} \cos^2(\theta_i) S_k(f_p, \bar{U}, z)} \dots \\ \cos\left(\frac{2\pi f_p}{\bar{U}_{10}}(x_u \cos(\theta_i) + y_u \sin(\theta_i)) - 2\pi f_p t + \psi_{ip}\right) \Delta\theta \Delta f$$

On peut alors reformer la vitesse du vent selon<sup>7</sup> :

$$u(x, y, z, t) = u(x, y, z) + \tilde{u}(x, y, z, t) \cos(\overline{\varphi(x, y, z)}) - \tilde{v}(x, y, z, t) \sin(\overline{\varphi(x, y, z)}) \\ u(x, y, z, t) = v(x, y, z) + \tilde{u}(x, y, z, t) \sin(\overline{\varphi(x, y, z)}) + \tilde{v}(x, y, z, t) \cos(\overline{\varphi(x, y, z)}) \\ w(x, y, z, t) = w(x, y, z) + \tilde{w}(x, y, z, t)$$

Ci-dessous sur la Figure 29 se trouvent une itération du vent turbulent que nous obtenons qui permet de comparer, à proximité du sol sur des altitudes basses, le vent stationnaire et le vent turbulent. Le zoom effectué permet de voir que globalement, la direction est conservée en moyenne, mais que de nombreuses perturbations s'appliquent.

---

6. On ne prend pas forcément la direction du vent telle que nous l'avons définie pour simplifier les changements de base.

7. Ici, on a donc par exemple  $u_k(x, y, z, t) = \tilde{u}(x, y, z, t)$ .

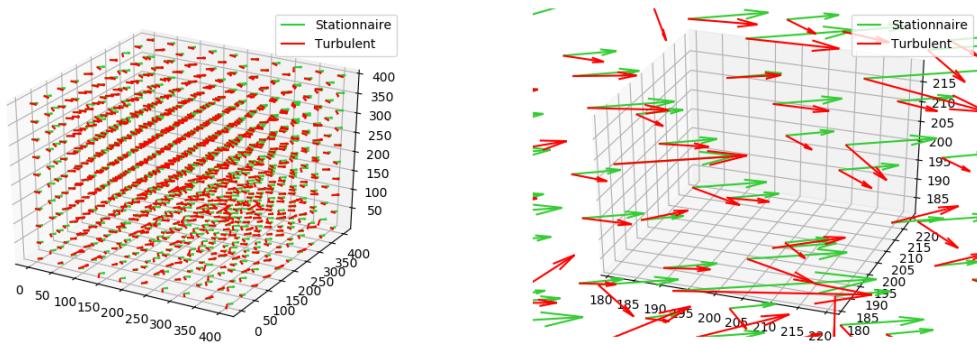


FIGURE 29 – Cube de turbulence : à partir d'un vent stationnaire de base, on peut obtenir une possibilité d'évolution turbulente du vent en tout point, qui soit cohérente spatialement et temporellement.

### 3 Notice du code

Le programme s'utilise comme un librairie Python, dont nous allons maintenant décrire les fonctions écrites, sous réserve d'installation des pré-requis.

#### 3.1 Pré-requis

Un certain nombre de pré-requis à installer sont nécessaires pour faire fonctionner correctement le code :

- **WindNinja** : Constructible sous Unix et Windows à partir du GitHub (<https://github.com/firelab/windninja>) ou directement installable sur le site du Firelab Américain (<https://www.firelab.org/document/windninja-software>, uniquement sous Windows). Le code fonctionne avec les versions 3.6.0, 3.7.0 et 3.7.1. Il devrait fonctionner avec les versions suivantes si aucun changement majeur n'est apporté.
- **Python 3.7** : **Attention** le code ne fonctionne pas avec une autre version de Python ! Il est fortement conseillé de créer un environnement python sous python 3.7. Pour se faire, vous pouvez suivre la marche à suivre donnée en partie 3.8.
- **Les librairies suivantes** : numpy, scipy, matplotlib, meshio, pandas, cfgrib, pyproj, pygrib, json, os, webbrowser, typing\_extensions, datetime, time, glob et xarray. La marche à suivre pour importer cfgrib et pygrib est détaillée dans la partie 3.8. Par ailleurs, ces librairies ne sont utiles QUE pour l'utilisation des librairies grib\_functions et grib\_functions\_man qui sont facultatives pour le reste du calcul.

Par ailleurs, il est important de noter que pour les fichiers de terrain, notre code ne peut lire que des fichiers GeoTiff (\*.tif) dans une limite de surface de 50 km × 50 km. Les fichiers météo, quant à eux, doivent être au format NetCDF (\*.nc). Des fonctions du bloc grib\_functions.py permettent de récupérer des fichiers de ce type à partir de fichiers GRIB.

Une fois que **WindNinja** est bien installé, il est recommandé de faire un essai à partir de la *GUI WindNinja* qui vient avec quelque cas test ainsi que de lire la documentation du logiciel, notamment pour en comprendre les différentes options de simulation.

### 3.2 Préparation du configFile.json

Le configFile est le fichier qui contient tous les paramètres de la simulation à réaliser. Il doit être configuré par l'opérateur avant chaque simulation et doit respecter une convention bien précise. On retrouve ainsi plusieurs blocs. Quelques points communs entre ces blocs sont à noter :

- Le format de chaque ligne est le suivant : ("paramètre" : valeur,)
- Les paramètres et les string doivent être mis entre guillemets
- Attention à ne pas oublier la virgule en fin de ligne sauf pour la dernière ligne de chaque bloc,
- Parfois certaines lignes ne sont à écrire que pour un type de simulation et pas dans les autres,
- Les possibilités pour certains paramètres ont été volontairement réduites par rapport à **WindNinja** de façon à garantir le bon fonctionnement du code.

Des exemples types de configFile se trouvent en *Annexe A* et peuvent servir de base pour les simulations.

#### 3.2.1 Le bloc "def"

Ce bloc définit les paramètres généraux de la simulation. Les différents paramètres à spécifier sont les suivants :

paramètres	type	définition	possibilités	exemple
"version"	string	version de WindNinja		"3.7.1"
"name"	string	nom à donner à la simulation		"Missoula"
"date"	string	date de la simulation au format year-month-day		"2021-03-21"
"mmtFile"	bool	si le fichier de terrain est amené par l'utilisateur	True, False	True
"windFile"	bool	si le fichier météo (.nc) est amené par l'utilisateur	True, False	False

#### 3.2.2 Le bloc "extrapolation"

Ce bloc définit les paramètres pour l'extrapolation en altitude. Les différents paramètres à spécifier sont les suivants :

paramètres	type	définition	possibilités	exemple
"nb_layer_extrapolation"	int	Nombre de points en altitude à rajouter		7
"elevation_max"	float	Elevation maximale de l'extrapolation (m)		4000.0

#### 3.2.3 Le bloc "windNinjaSimulations"

Ce bloc définit les paramètres d'entrée de la simulation **WindNinja**. Selon le type de simulation que l'on souhaite réaliser, il peut prendre plusieurs formes mais dans tous les cas on retrouve les mêmes étapes.

#### Données de terrain

Le première étape définit les données de terrain. Tous les paramètres ne sont pas les mêmes si le fichier est apporté par l'utilisateur ou doit être téléchargé par **WindNinja**. Attention, il ne faut surtout pas qu'il y ait de mélange de paramètres entre les deux cas dans le fichier !

Cas	paramètres	type	définition	possibilités	exemple
Si fichier à télécharger	"fetch_elevation"	string	Nom du fichier à télécharger	".*,tif"	"terrain.tif"
	"elevation_source"	string	Type de données de terrain	"gmted"	"gmted"
Si fichier apporté par utilisateur	"elevation_file"	string	Chemin complet du fichier de terrain		"C :/data/terrain.tif"
Commun	"x_center"	float	Longitude du point central du domaine		-114.89
	"y_center"	float	Latitude du point central du domaine		48.6
	"x_buffer"	float	Demi-longueur du domaine selon l'axe Est/Ouest		5.0
	"y_buffer"	float	Demi-longueur du domaine selon l'axe Nord/Sud		5.0
	"buffer_units"	string	Unité des demi-longueurs	"kilometers"	"kilometers"
	"mesh_choice"	string	Type de maillage du terrain	"coarse", "medium", "fine"	"coarse"
	"vegetation"	string	Végétation dominante	"grass", "brush", "trees"	"grass"
	"time_zone"	string	Time-zone du domaine	<i>Annexe B</i>	"America/Denver"

## Données de vent

On définit ensuite les paramètres d'entrée du vent. Encore une fois, plusieurs options sont possibles selon si l'on spécifie simplement un vent uniforme sur tout le domaine (*domainAverageInitialization*) ou un fichier météo (*wxModelInitialization*). Pour ce dernier cas, on peut soit spécifier un fichier d'entrée soit le télécharger via **WindNinja**.

Cas	paramètres	type	définition	possibilités	exemple
Commun	"initialization_method"	string	Méthode de définition du vent en entrée	"domainAverageInitialization", "wxModelInitialization"	
Domain Average	"input_speed"	float	Vitesse du vent en entrée		36.0
	"input_speed_units"	string	Unités pour la vitesse d'entrée	"mps"	"mps"
	"input_direction"	float	Direction d'où vient le vent 0° correspond au Nord, 90° à l'Est	[0°, 360°]	50.0
	"input_wind_height"	float	Hauteur du vent en entrée		10.0
	"units_input_wind_height"	string	Unités de la hauteur du vent en entrée	"m"	"m"
Fichier à télécharger	"wx_model_type"	string	Type de modèle utilisé	"UCAR-GFS-GLOBAL-0.5-DEG"	
	"forecast_duration"	int	Durée du modèle à télécharger	3	3
Fichier utilisateur	"forecast_filename"	string	Chemin complet du fichier météo	".*.nc"	"C :/data/fichier.nc"
Commun	"non_neutral_stability"	bool	Activation modèle instabilité atmosphérique	true, false	false

Finalement, il ne reste plus qu'à définir les Outputs :

paramètres	type	définition	possibilités	exemple
"output_wind_height"	float	Hauteur du vent en sortie (pdf)		10.0
"units_output_wind_height"	string	Unités pour la hauteur du vent	"m"	"m"
"output_speed_units"	string	Unités pour la vitesse du vent en sortie	"mps"	"mps"
"write_pdf_output"	bool	Ecriture d'une carte de vent pdf	true, false	false
"write_vtk_output"	bool	Ecriture des fichiers VTK	true	true
"num_threads"	int	Nombre de threads	1	1

A noter que la sortie d'un fichier PDF permettant d'avoir une vue 2D du résultat à une hauteur spécifiée est laissée à l'appréciation de l'utilisateur mais qu'il est obligatoire que la simulation retourne des fichiers VTK.

### 3.3 Description de la classe

Pour faciliter l'utilisation, il a été décidé que le programme serait utilisable sous la forme d'une librairie Python, obtenu via une classe nommée : **wind**.

### 3.3.1 Les attributs

Python ne proposant pas la possibilité de choisir le niveau d'accès des attributs, ceux-ci sont accessibles par l'utilisateur depuis n'importe quel fichier l'utilisant. Il est donc à la discréption de l'utilisateur de ne pas modifier directement ces attributs et de ne passer que par les méthodes mises à disposition.

- `wind_cube` : un dictionnaire composé des listes suivantes :
  - `Position` : Tous les triplets de coordonnées  $(x, y, z)$  du cube de vent,  $(0,0,0)$  étant le point le plus au sud-ouest de la plus basse altitude du cube, selon les règles suivante :
    - $x$  : la composante suivant la latitude,
    - $y$  : la composante suivant la longitude,
    - $z$  : la composante suivant la verticale locale.
  - `Wind_speed` : Les vitesse  $(u, v, w)$  du vent situé au couple  $(x, y, z)$  dans le cube suivant la règle suivante :
    - $u$  : la composante de l'Ouest vers l'Est,
    - $v$  : la composante du Sud vers le Nord,
    - $w$  : la composante verticale, des plus basses vers les plus hautes altitudes.
  - `Surface_altitude` : l'altitude de la surface (a.s.l.) situé au couple  $(x, y, z)$  dans le cube.
- `list_point` : un dictionnaire, pour faciliter certains calculs, composé des listes suivantes :
  - $x$  : Toutes les valeurs de  $x$ ,
  - $y$  : Toutes les valeurs de  $y$ ,
  - $z$  : Toutes les valeurs de  $z$ .
- `nb_point` : le nombre de points dont est composé le cube (soit la taille du tableau `Position`),
- `nb_layer_extrapolation` : le nombre de couches dont sera composé le cube entre l'altitude maximale fournie par `WindNinja` et l'altitude maximale souhaitée,
- `elevation_max` : l'élévation (a.g.l.) maximale souhaitée,
- `date` : la date des données de simulations (celle de la sous-section 3.2.1),
- `location` : les coordonnées les plus au Sud-Ouest du cube de simulation,
- `folder_name` : le chemin du dossier dans lequel sont rangés les différents éléments de la simulation.

### 3.3.2 Les méthodes

Ci-après se trouve un liste exhaustive des méthodes utilisées par l'utilisateur avec les paramètres, quelques-unes sont détaillées plus précisément dans la suite de cette section. Il est à noter que chaque fonction est commentée selon la norme `numpy` de création des *docsstring* : un détail est donc accessible via n'importe quel environnement de développement Python prenant en charge ce type.

- `create_wind_cube(input_path, simu_name, output_path = "")` : crée le cube de la simulation à partir des données situées dans `input_path` et stocke toutes les informations dans le dossier `output_path` (qui vaut par défaut `input_path`), `simu_name` apparaissant dans le nom du fichier où les données seront stockées,

- `import_wind_cube(file_name)` : importe des données de simulations déjà effectuées et stockées dans le fichier `.json` dont le chemin est : `file_name`,
- `export_wind_cube` : exporte les données de simulations au format JSON dans le dossier de la simulation avec le nom suivant : `exported_data_[nom simulation]_[date].json`,
- `get_data` : retourne toutes les données de la simulation dans un dictionnaire selon la même structure que le fichier `.json`,
- `get_point(latitude, longitude, elevation = 0, altitude = 0, plot = False)` : retourne les paramètres de vents suivants aux coordonnées passées en argument : les trois composantes de la vitesse  $u, v, w$ , la norme de la vitesse, la norme de la vitesse projetée dans le plan horizontal et la direction de provenance du vent, en degrés (entre  $0^\circ$  et  $360^\circ$ ,  $0^\circ$  correspondant à un vent venant du Nord et  $90^\circ$  à un vent venant de l'Est). Une rose des vents est également affichée si `plot = True`. Le choix est laissé à l'utilisateur entre l'élévation (a.g.l.) et l'altitude (a.s.l.),
- `turbulence(latitude, longitude, elevation, time, plot)` : retourne les mêmes composantes du vent que `get_point(latitude, longitude, elevation)`, mais en rajoutant une perturbation turbulente aléatoire, et affiche la rose des vents associée si `plot = True`,
- `profil_turbulence(latitude, longitude, elevation, timestep, nb_points, plot)` : retourne l'évolution de la vitesse du vent aux coordonnées passées en argument durant 15 minutes au pas de temps souhaité. Un graphique est également affiché si `plot = True` avec une discrétisation spatiale de `nb_points`,
- `plot_wind_surface(axis, coord, alt, nb_points, plot)` : calcule le profil du vent selon l'axe et les coordonnées souhaitées et les affiche si `plot = True` avec une discrétisation spatiale de `nb_points`,
- `plot_wind_cube(xlim, ylim, zlim, nb_points, plot)` : calcule le vent à l'intérieur du cube délimité en paramètres et l'affiche si `plot = True` avec une discrétisation spatiale de `nb_points`,
- `plot_wind_cube_turbulent(xlim, ylim, zlim, T, dt, nb_points, plot)` : calcule le vent en y ajoutant une composante turbulente à l'intérieur du cube délimité en paramètres et affiche les différents cubes si `plot = True` avec une discrétisation spatiale de `nb_points`.
- `get_surface_altitude(latitude, longitude)` : retourne l'altitude a.s.l. des coordonnées en argument.
- `cube_coordinates` : retourne les coordonnées du plus point le plus au Sud-Ouest et les du point le plus au Nord-Est sous la forme d'un dictionnaire.

## 3.4 Fonctions utilisant le temps

### 3.4.1 Fonctionnement

Deux fonctions permettant d'inclure les variations temporelles sont présentes dans le code. Ces deux fonctions ne vont pas renvoyer la valeur estimée du vent à l'instant  $t$  et à une certaine position, mais une valeur *possible* qui se base sur des méthodes de spectre de puissance. Ces fonctions ne sont donc pas là pour prédire le champ de vent réel, mais pour éventuellement entraîner un correcteur à une "certaine réalité".

**Turbulence instantanée** La fonction `turbulence` demande en entrée la position du point d'étude et une durée  $\Delta t$  correspondant à la durée entre le moment où on souhaite la mesure et le moment de référence des données d'entrée. Elle récupère alors le vecteur vitesse stationnaire correspondant à ce point *via* les fonctions précédentes. Direction et norme plane sont alors calculées.

Un nombre  $N = \Delta t f_s$  qui correspondra au nombre de modes,  $f_s$  étant la fréquence d'échantillonage, est alors calculé. La fonction renvoie alors les perturbations  $\tilde{u}$ ,  $\tilde{v}$  et  $\tilde{w}$  à ajouter au vent stationnaire, la somme des deux est alors retournée à l'utilisateur.

Du fait du caractère aléatoire, deux appels successifs à la fonction `turbulence` ne renverront évidemment pas le même résultat.

**Profil de turbulence sur 15 minutes** La fonction `profil_turbulence` demande en entrée la position du point d'étude et un pas de temps  $dt$ . Comme `profil_turbulence`, elle récupère la vitesse stationnaire en ce point et renvoie avec un algorithme similaire un profil d'évolution du vent pendant une durée de 15 minutes avec un échantillonage de  $dt$ . Ce retour se fait sous la forme de graphiques (donnant le vent dans les directions principale, latérale, Est-Ouest, Nord-Sud et verticale).

Le calcul du signal temporel se fait via un appel à la transformation inverse de Fourier rapide, ce qui permet de garantir un temps de calcul tout à fait raisonnable (de l'ordre de 10 s pour  $dt = 0.01$  s).

### 3.4.2 Modifications par l'exploitant

Pour notre problème, nous avons fait le choix de considérer un certain jeu de valeurs, que ce soit pour établir les fonctions de spectre de densité de puissance, ou pour fixer les amplitudes de turbulence ou de rafale. Ces valeurs fixes diffèrent assez souvent selon la littérature, et elles peuvent être modifiées dès le début du code.

Les valeurs par défaut qui sont entrées sont :

- $A_z$  : `A_Z` = 0.177,
- $B_z$  : `B_Z` = 0.00823,
- $C_z$  : `C_Z` = 1.2
- $\alpha_p$  : `PENTE_PRIN` = 5/6,
- $D_p$  : `D_PRIN` = 70.7,
- $\alpha_w$  : `PENTE_LATW` = 11/6,
- $D_w$  : `D_LATW` = 188.4,
- $T$  : `TIME_MAX` = 900,
- $\eta_g$  : `CORREC_GLOBAL` = 1,
- $\eta_z$  : `CORREC_ALTI` = 0.1
- $r_u$  : `RATIO_PRIN` = 0.11,
- $r_v$  : `RATIO_LAT` = 0.11,
- $r_w$  : `RATIO_VERT` = 0.06.

Ces valeurs interviennent notamment dans les formules des spectre  $S_u$  et  $S_k$  pour  $k \in \{v, w\}$  :

$$\frac{S_u(f)}{\sigma_u^2} = \frac{4\ell_u^x}{\bar{U}(z)} \frac{1}{\left(1 + D_p \left(\frac{f\ell_u^x}{\bar{U}(z)}\right)^2\right)^{\alpha_p}} \quad \frac{S_k(f)}{\sigma_k^2} = \frac{4\ell_k^x}{\bar{U}(z)} \frac{1 + D_w \left(2 \frac{f\ell_k^x}{\bar{U}(z)}\right)^2}{\left(1 + D_p \left(2 \frac{f\ell_k^x}{\bar{U}(z)}\right)^2\right)^{\alpha_w}}$$

$$\ell_u = \ell_v = \frac{z}{(A_z + B_z z)^{C_z}} \quad \ell_w = z \quad \sigma_k = r_k \eta_g \min \left\{ \frac{\eta_z - 1}{1000 - 0} z + 1, \eta_z \right\} U_{10}$$

Leur modification n'aura aucun impact sur les fonctions donnant le vent stationnaire. Dans notre cadre, nous nous sommes servis des données de mesure à Lannemezan pour figer certains des coefficients.

### 3.4.3 Un exemple pas à pas

Mettons que l'on souhaite observer le vent à une longitude  $43.6^\circ$ , une latitude  $2.43^\circ$  et une élévation de 15 m contenues dans le cube de vent, une minute après les données initiales. L'utilisateur entrera `turbulence(43.6, 2.43, 15, 60)`. Il lui sera retourné la rose des vents (voir Figure 30) ainsi que le champ de vitesse  $-0.867, 14.968, 1.315, 15.051, 14.993, 176.682$ , ce qui permettra d'en déduire que le vent vient essentiellement du Sud ( $U_x = -0.86$  m/s,  $U_y = 14.97$  m/s,  $U_z = 1.31$  m/s pour une norme totale de  $\bar{U} = 15.05$  m/s).

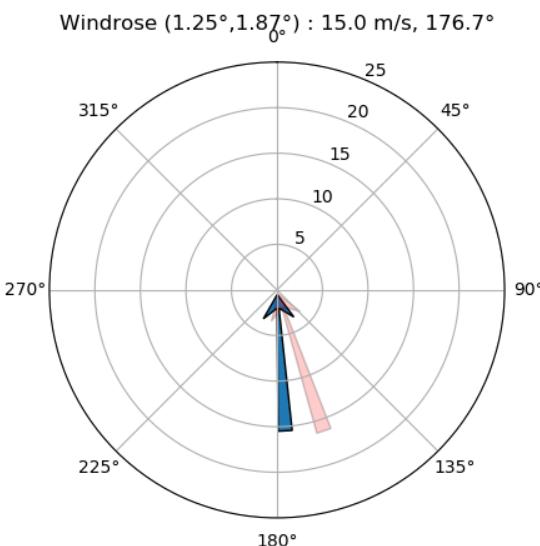


FIGURE 30 – Rose des vents : la vitesse turbulente est en bleu, la vitesse stationnaire en rouge clair.

Pour le profil turbulent, pour les mêmes coordonnées, l'utilisateur récupérera les listes exploitables d'évolution du vent dans les deux repères, ainsi que la représentation graphique de la Figure 26.

## 3.5 Fonctions pour l'extrapolation

Le bloc réalisant l'extrapolation en altitude du `wind_cube` est composé d'un total de six fonctions. La fonction principale du bloc est la fonction `main` qui va réaliser en chaque point du maillage une extrapolation selon l'axe vertical à partir des données calculées par **WindNinja**. Pour cela, on réalise un *fitting* en chaque point de la fonction puissance discutée dans la partie 2.1 puis on calcule les  $N$  points supplémentaires jusqu'à une élévation a.g.l max, ces deux derniers paramètres étant définis par l'utilisateur au moment du lancement du calcul. Pour cela, quatre autres fonctions sont utilisées en interne.

La fonction `get_vert_profile` permet de récupérer l'ensemble des points de données en une même coordonnée ( $x, y$ ) du maillage. Elle permet donc de récupérer les résultats de **WindNinja** en chaque point du maillage.

La fonction `alt_to_elev` permet de passer de la convention altitude (a.s.l) à la convention élévation (a.g.l). En effet, il est important que la correspondance de la loi puissance soit effectuée sur des points respectant cette convention. Cela permet par ailleurs d'avoir un maillage homogène le long de l'axe  $z$ , ce qui facilitera certaines représentations graphiques ainsi que l'interpolation au sein du `wind_cube`.

La fonction `get_Zlist_pos` permet de récupérer les coordonnées  $z$  de tous les points situés en une position ( $x, y$ ) dans le plan horizontal.

La fonction `get_extrap_law` permet de récupérer les paramètres optimaux pour la loi puissance correspondant à un point ( $x, y$ ). Pour cela elle utilise la fonction `curve_fit` de la librairie `scipy.optimize`.

Enfin la dernière fonction, `power_law` est tout simplement l'écriture de la loi puissance.

### 3.6 Fonctions pour l'interpolation et l'affichage des résultats

Notre classe propose diverses fonctions pour réaliser des interpolations au sein du `wind_cube`, créer et afficher des résultats. La fonction `get_point` permet de récupérer la donnée de vent en n'importe quel point du cube à partir de ses coordonnées en longitude et latitude. La fonction `plot_wind_cube` permet pour la première d'interpoler un `wind_cube` au sein du `wind_cube` principal et éventuellement de l'afficher graphiquement tandis que `plot_wind_surface` permet de faire la même chose mais pour un profil de vent en un point ou une surface à altitude constante. La fonction `plot_wind_cube_turbulent` permet également d'afficher l'influence de la turbulence au cours du temps sur plusieurs cubes de vents. Pour cela elles utilisent toutes les deux les mêmes fonctions intermédiaires :

- `get_surf` permet de construire les meshgrid X,Y et Z de la surface. Elle n'est utilisée que si un affichage graphique est demandé par l'utilisateur.
- `get_tick_argminmax` permet de récupérer les arguments d'un segment d'axe.
- `get_interv` récupère, pour chaque axe du `wind_cube`, les segments de taille minimale permettant d'englober l'ensemble du domaine demandé par l'utilisateur.
- `get_interp_data` à partir de la taille des segments minimaux elle décide de ne sélectionner ou non qu'une certaine partie du `wind_cube` pour réaliser l'interpolation. Elle réalise ensuite cette interpolation et retourne les résultats sous forme de meshgrid.

#### 3.6.1 Remarque importante pour les méthodes d'affichage

Il est important de noter que le cube de vent obtenu en sortie de l'extrapolation a pour convention de coordonnée  $z$  l'élévation a.g.l. Cela a pour effet de "mettre à plat" la surface comme si on regardait une carte et pour conséquence de ne pas avoir un cube homogène lorsque l'on passe en convention altitude. Le point le plus haut au-dessus du point de la surface le plus bas aura une altitude inférieure à celle du point le plus haut au-dessus du point de la surface le plus haut. Cette inhomogénéité est particulièrement marquée pour des terrains très escarpés. A la suite de cela s'est posée la question de la convention à utiliser

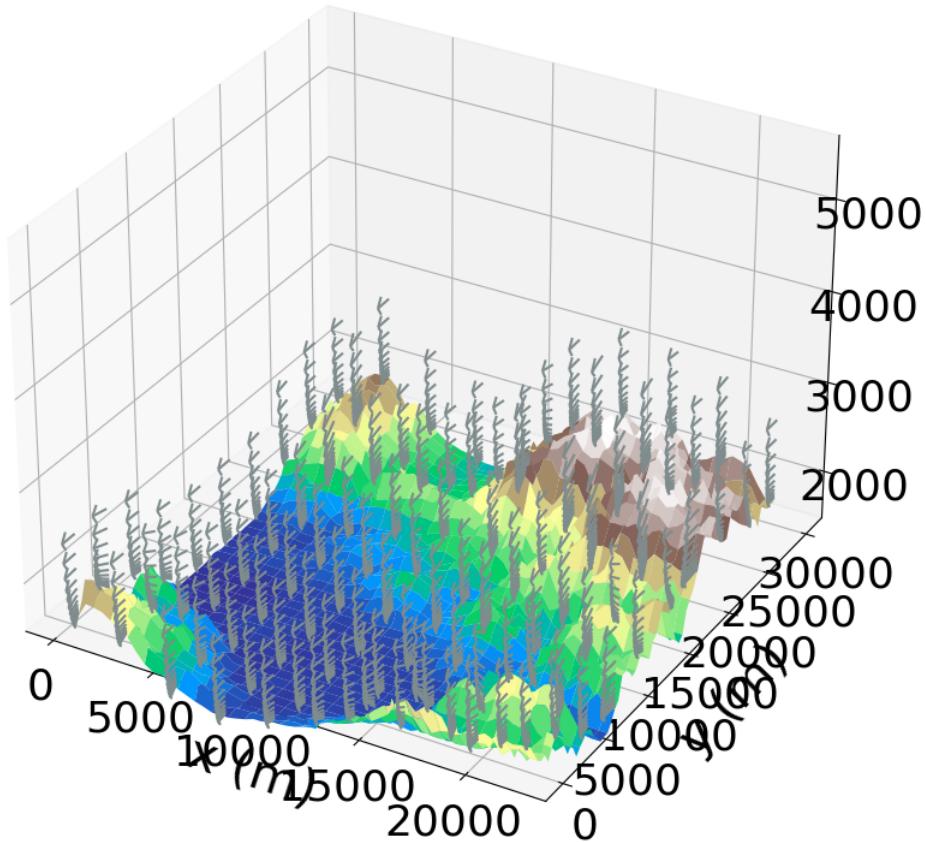


FIGURE 31 – Un exemple de `wind_cube`.

en sortie des fonctions `plot_wind_cube` et `plot_wind_surface` et des conséquences que cela pouvait avoir sur les données en sortie. On se retrouve alors avec trois cas différents :

- **Wind\_profile** : Dans ce cas on souhaite obtenir le profil de vent en un point. Il est conventionnel pour ce cas d'avoir les données retournées et les graphiques en convention élévation. C'est donc la convention qui a été choisie.
- **Wind\_surface** : Cette fois-ci le but est d'obtenir une surface de vent à  $z$  constant. Du point de vue de notre application qui est le largage de colis *via* les parachutes, nous avons pensé qu'il était plus intéressant que le  $z$  en question soit une altitude constante (affichage d'une tranche d'atmosphère). Une conversion est donc réalisée avant l'interpolation. Cela a deux conséquences :
  1. Une augmentation du temps de calcul de l'interpolation du fait que les points en entrée ne représentent plus un ensemble homogène.
  2. Selon l'altitude demandée, certains points ne seront pas calculés. C'est le cas si l'altitude demandée est inférieure à l'altitude de la surface ou si elle est au-dessus de l'altitude maximale calculée lors de l'extrapolation. Dans le premier cas cela se voit sur l'affichage graphique, dans le deuxième un avertissement est généré pour prévenir l'utilisateur.
- **Wind\_cube** : C'est le cas le plus complexe. En effet, la vision que nous avons sur ce `wind_cube` est celle d'un petit `wind_cube` à l'intérieur du grand. Par soucis de

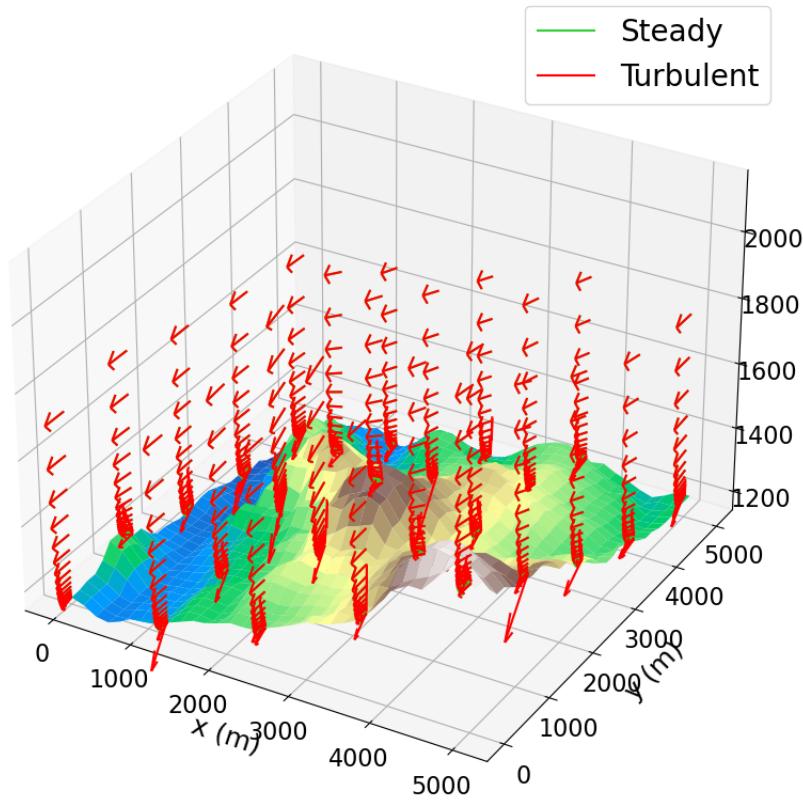


FIGURE 32 – Un exemple de `wind_cube` turbulent : la turbulence est peu perceptible dès que l'on monte en élévation.

cohérence nous avons donc décidé de garder la convention élévation pour les données de sortie. En revanche, lors de l'affichage graphique il nous est apparu qu'il était plus intéressant d'avoir la surface en 3D ce qui impliquait une conversion en convention altitude. **Attention** seul l'affichage est réalisé en convention altitude, les données de sortie sont en convention élévation !

### 3.7 Cas d'utilisation

Dans cette section, un exemple de calcul réalisé avec la classe python créée est présenté. Il s'agit d'un calcul à partir d'un fichier de terrain avec téléchargement des données météo par **WindNinja**.

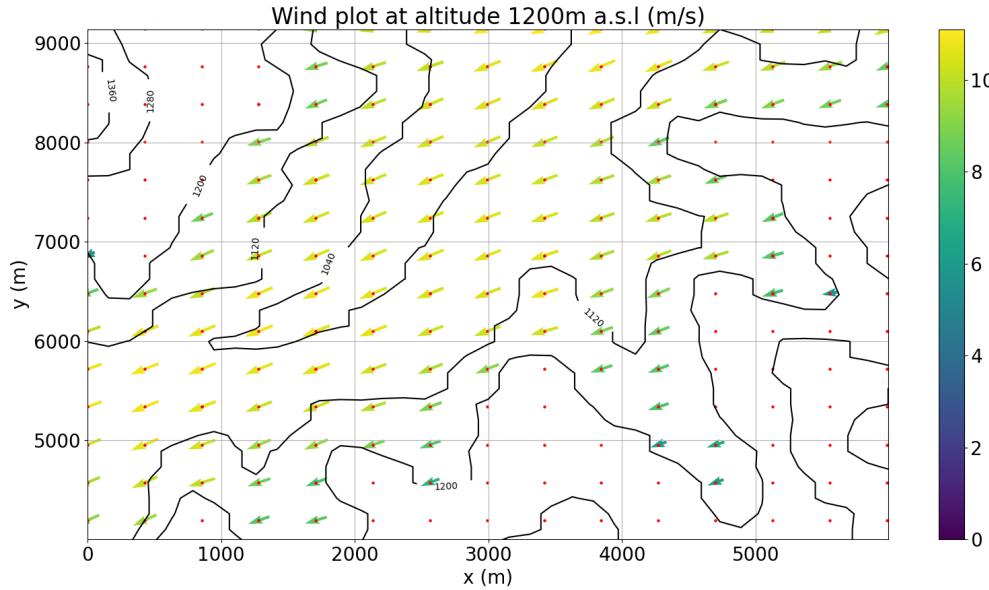


FIGURE 33 – Un exemple de wind\_surface.

### 3.7.1 Préparation du calcul

```

wind_ninja_functions.py  configFile.json  wind_test.py
Schéma : Aucun schéma sélectionné>

1  {
2    "def": {
3      "version": "3.7.1",
4      "name": "Missoula",
5      "date": "2021-03-21",
6      "mmtfile": true,
7      "gribfile": false
8    },
9    "extrapolation": {
10      "nb_layer_extrapolation": 7,
11      "elevation_max": 4000.0
12    },
13    "windNinjaSimulations": {
14      "elevation_file": "missoula.tif", ②
15      "x_center": -114.89,
16      "y_center": 48.6,
17      "x_buffer": 5,
18      "y_buffer": 5,
19      "buffer_units": "kilometers",
20      "vegetation": "grass",
21      "mesh_choice": "coarse",
22      "time_zone": "America/Denver",
23
24      "initialization_method": "wxModelInitialization",
25      "wx_model_type": "UCAR-GFS-GLOBAL-0.5-DEG", ③
26      "forecast_duration": 3,
27
28      "non_neutral_stability": false,
29
30      "output_wind_height": 10.0,
31      "units_output_wind_height": "m",
32      "output_speed_units": "mps",
33      "write_pdf_output": true,
34      "write_vtk_output": true,
35
36      "num_threads": 1
37    }
38  }
39

```

The screenshot shows the PyCharm IDE interface. On the left, the code editor displays `wind_ninja_functions.py` with several parameters highlighted by red circles: ② `elevation_file` pointing to `missoula.tif`, and ③ `wx_model_type` pointing to `UCAR-GFS-GLOBAL-0.5-DEG`. On the right, the "Explorateur de solutions" (Solution Explorer) shows the project structure under `spatio-temporal_wind_modeling`, including files like `configFile.json`, `missoula.tif`, `desktop.ini`, and various Python scripts. The `wind_test.py` file is currently selected.

FIGURE 34 – Préparation du calcul

Un nouveau dossier 'Test' est créé pour cette simulation. A l'intérieur on retrouve les données spécifiées par l'utilisateur, à savoir le fichier de terrain 'missoula.tif' et le config-

File.json (1). A l'intérieur de ce dernier on retrouve les spécifications du cas d'étude avec le nom du fichier de terrain (2) et la méthode de téléchargement du fichier météo (3).

### 3.7.2 Lancement du calcul

The screenshot shows a code editor with three tabs: 'wind\_ninja\_functions.py', 'configFile.json', and 'wind\_test.py'. The 'wind\_test.py' tab is active, displaying the following Python code:

```
#% Import
import wind_class as wind ④

# Data gathering
folder = "C:/Users/abeil/Source/Repos/spatio-temporal_wind_modeling/scr/Test"

wind_test = wind.wind()
wind_test.create_wind_cube(folder, "test", folder + "/output/") ⑤
```

FIGURE 35 – Lancement du calcul

Pour lancer le calcul il est nécessaire d'importer la classe `wind_class` et de créer un nouvel objet `wind` puis d'utiliser la méthode `create_wind_cube` avec comme paramètres le dossier d'entrée, le nom de la simulation et le dossier de sortie. La fenêtre suivante s'affiche alors et permet de suivre l'avancement du calcul.

```
# Beginning simulation
#####
C:/WindNinja/WindNinja-3.7.1/bin/WindNinja_cli C:/Users/abeil/Source/Repos/spatio-temporal_wind_modeling/scr/Test/output/test.cfg -
-output_path C:/Users/abeil/Source/Repos/spatio-temporal_wind_modeling/scr/Test/output/
Run 0: The path C:/Users/abeil/Source/Repos/spatio-temporal_wind_modeling/scr/Test/output/ does not exist, writing outputs to default location.
Run 0: Reading elevation file...
Run 0: Simulation time is 2021-Mar-22 11:00:00 MST
Run 0: Run number 0 started with 1 threads.
Run 0: Generating mesh...
Run 0: Initializing flow...
Run 0: Building equations...
Run 0: Solving...
Run 0 (solver): 25% complete
Run 0 (solver): 56% complete
Run 0 (solver): 75% complete
Run 0 (solver): 90% complete
Run 0 (solver): 98% complete
Run 0 (solver): 100% complete
Run 0: Writing output files...
Run 0: Meshing time was 0.002819 seconds.
Run 0: Initialization time was 0.027489 seconds.
Run 0: Equation building time was 0.127000 seconds.
Run 0: Solver time was 0.363308 seconds.
Run 0: Output writing time was 1.720770 seconds.
Run 0: Total simulation time was 2.289587 seconds.
Run 0: Run number 0 done!
#####
# Beginning extrapolation
#####
Run :  0 %
D:\Anaconda\envs\PIE\lib\site-packages\scipy\optimize\minpack.py:829: OptimizeWarning: Covariance of the parameters could not be estimated
    category=OptimizeWarning)
Run :  10 %
Run :  20 %
Run :  30 %
Run :  40 %
Run :  50 %
Run :  60 %
Run :  70 %
Run :  80 %
Run :  90 %
Run :  100 %
#####
# End of extrapolation #
#####
# Beginning export
#####
# End of simulation, state = True
#####
Press any key to continue . . . -
```

FIGURE 36 – Calcul en cours

### 3.7.3 Exploitation des résultats

Un fois le calcul terminé, le dossier de sortie a été rempli avec les différents fichiers de sortie :

- Le modèle météo téléchargé par **WindNinja** (6)
- Un export du **wind\_cube** (7)
- Les fichiers de configuration de la simulation complète (.json) et de **WindNinja** (.cfg) (8)

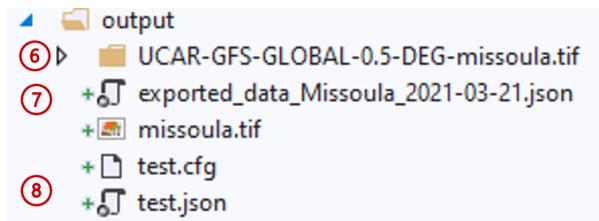


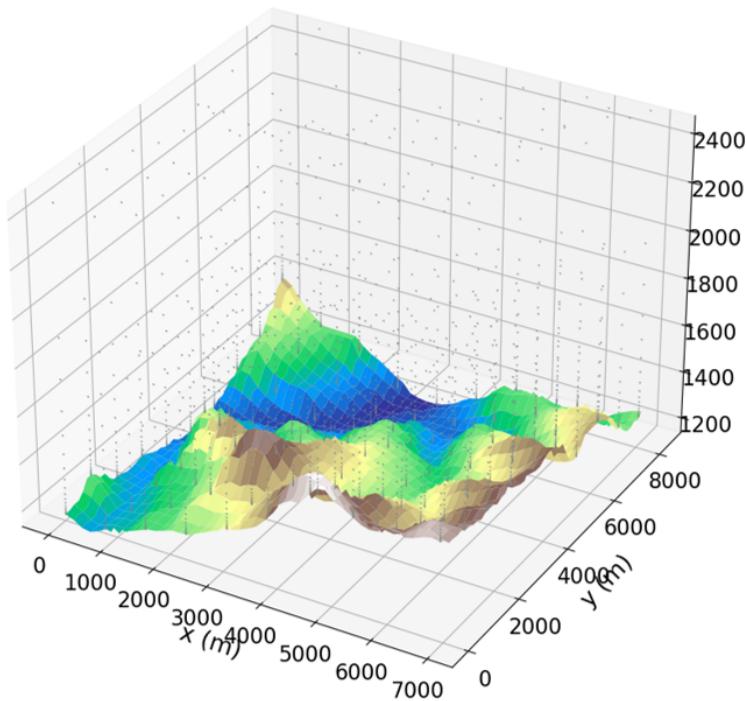
FIGURE 37 – Fichiers de sortie

On peut alors si on le souhaite afficher graphiquement un bout du **wind\_cube**. Pour cela on exécute les deux commandes de la figure 38. La première permet de récupérer les distances en x et y en km du point (48.63, -114.84) depuis le point en bas à gauche du domaine. La seconde permet de lancer l'interpolation et le plot.

```
x,y = wind.flat_distance_point(wind_test._location,[48.63,-114.84])
wind_test.plot_wind_cube([0,x*1000],[0,y*1000],[0,1000],plot=True)
```

FIGURE 38 – Commandes d'affichage

Et voilà le résultat !

FIGURE 39 – Le `wind_cube` interpolé

### 3.8 Fonctions pour la récupération et la conversion des fichiers GRIB

Cette partie correspond à un module externe dont l'utilisateur peut se servir s'il veut convertir un fichier grib en un fichier nc, récupérer des données grib du jour d'internet et les convertir en nc, manipuler des données d'un fichier grib donné en entrée ou manipuler des données de fichiers grib du jour récupérés par le code sur internet.

Notre programme s'appuie sur des données de vents que nous collectons depuis le site de Météo France. Les fichiers récupérés sont sous le format GRIB. Nous devons donc premièrement manipuler des fichiers GRIB sous Python, filtrer les données pour ne garder que les données qui nous intéressent (les grib contiennent plusieurs champs : températures, vitesses de vent, etc. ce qui justifie qu'il faut donc les filtrer) et les convertir en des formats qui nous permettent de traiter ces données. Quatre fichiers Python sont fournis, contenant les mêmes fonctions à quelques petites différences près. Le premier, `grib_function.py`, va directement télécharger les fichiers GRIB nécessaires (il ne peut télécharger que des données du jour même) et retourner une datafram contenant les données de vents à l'heure et l'altitude choisie. Le second fichier, `grib_functions_man.py` permet à l'utilisateur de choisir le fichier GRIB en entrée. Le troisième fichier, `grib_nc_man.py`, va convertir un fichier GRIB donné en entrée en fichier netcdf. Enfin, le dernier fichier, `grib_nc_auto.py` va directement télécharger les fichiers GRIB nécessaires (il ne peut télécharger que des données du jour même) et les convertir en fichier netcdf. Les fonctions utilisées sont décrites ci-dessous :

- `open_file_grib` permet de vérifier si le fichier GRIB dont le nom et le chemin sont donnés en paramètre (paramètre sous la forme `chemin_nom`) existe ou non. Si le fichier n'existe pas, le code sort le message "still downloading".

- `get_grib_from_web` permet de télécharger le fichier GRIB demandé selon les paramètres rentrés et vérifie que le téléchargement a été réalisé correctement. Les trois paramètres demandés sont le type de paquet (SP1 pour les données à la surface, HP1 sinon), l'heure pour laquelle on souhaite avoir les données (sous type de string, écrire '05' pour 5 heures du matin, '13' pour 13 heures), et le chemin du dossier où les fichiers téléchargés sont stockés.
- `get_grib_from_path` va lire le fichier GRIB demandé selon les paramètres rentrés.
- `spped` va renvoyer la racine de la somme des carrés des deux paramètres.
- `getDateMessage` retourne le datetime auquel les données ont été connectées. Pour accéder au premier message d'un fichier GRIB, il suffit d'utiliser `GRIB`.
- `chosen_param` permet de lire le message demandé selon les paramètres rentrés.
- `main` va retourner une `pandas dataframe` contenant les informations de vent (composante *u* et *v*) pour l'altitude, l'heure et les bornes géographiques choisies. La fonction génère par la même occasion deux fichiers au format `netcdf` correspondant aux conversions des deux fichiers GRIB (SD1 et HP1) en `netcdf`. Le paramètre doit être au format (par exemple, '`C:/Users/33658/Desktop/PIE_data_testOutput.nc`').

**Remarque** Ces codes utilisent deux libraires dont l'importation peut être laborieuse : `pygrib` et `cfgrib`. Pour les importer, il faut impérativement créer un environnement Anaconda ayant une version Python = 3.7. Pour ce faire, il faut entrer la commande :

- `conda create -n py37 python==3.7` dans une invite de commande anaconda (ici l'environnement sera appelé `py37`, le nom est totalement arbitraire). Ensuite, pour activer cet environnement, il faut taper :
  - `conda activate py37` Ensuite, des commandes classiques que sont :
  - `conda install -c conda-forge cfgrib`
  - `conda install -c conda-forge pygrib` qui permettent de télécharger les librairies.
- Il faut de préférence lancer le code directement dans l'invite de commande (sans passer par `Spyder` ou autre environnement équivalent) en se mettant dans le dossier contenant le fichier python à lancer (par des commandes `cd`) puis taper la commande :
- `python <nomdufichier.py>`

### 3.9 Fonctions pour la conversion des fichiers VTK

Le programme **WindNinja** nous crée en output un fichier `.vtk`, que nous convertissons grâce à la librairie `meshio` en `numpy array`. Le fichier `vtk_function` ne contient qu'une seule fonction, qui réalise la conversion.

- `main` convertit le fichier `vtk` donné en paramètre en `numpy array`.

## 4 Rapport d'essais

### 4.1 Première phase d'essais

#### 4.1.1 Présentation de la démarche

Les données sont les mêmes que les données utilisées en 2.2.1.

Pour s'assurer que les données calculées par **WindNinja** étaient correctes, nous avons effectué les tests suivants :

1. Découpage des données en 20 heures de références,
2. A l'aide des données mesurées, calcul de la vitesse du vent moyennée pendant 10 minutes autour de la station,
3. Exécution de la simulation à l'aide de ces vitesses et d'un modèle numérique du terrain,
4. Comparaisons avec les données mesurées par la station aux trois altitudes dont nous disposons.

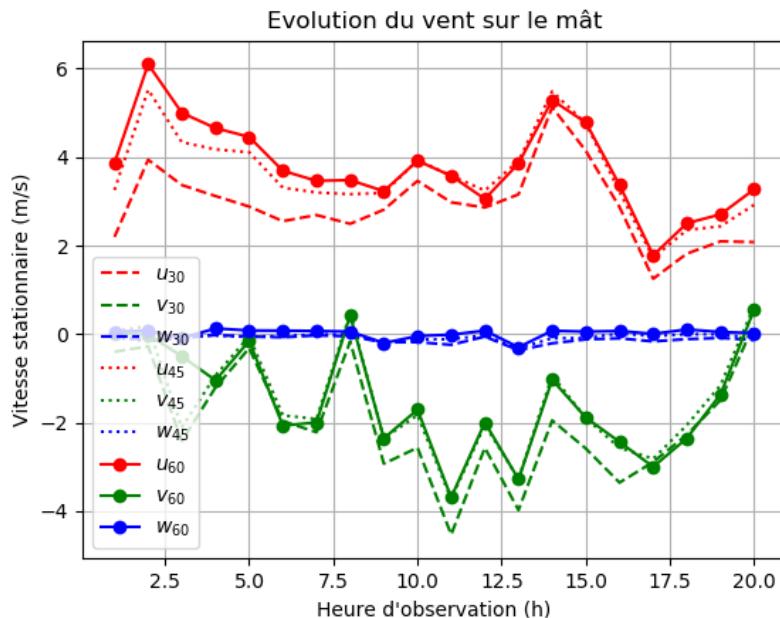


FIGURE 40 – Données d'entrée des essais

#### 4.1.2 Présentation des résultats

Nous avions six variables à comparer, avec 60 valeurs pour chacune d'entre elles :

- $u$  : composante  $u$  de la vitesse Ouest-Est,
- $v$  : composante  $v$  de la vitesse Sud-Nord,
- $w$  : composante  $w$  de la vitesse verticale,
- norme :  $\sqrt{u^2 + v^2 + w^2}$ ,
- norme plane :  $\sqrt{u^2 + v^2}$ .

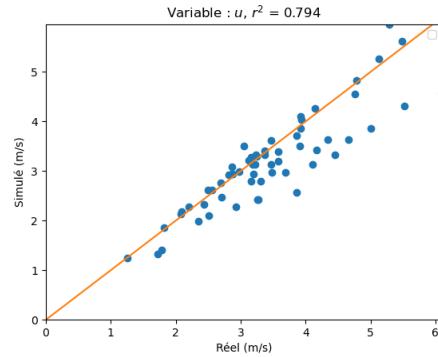


FIGURE 41 – Données simulées VS réelles pour la composante  $u$ .

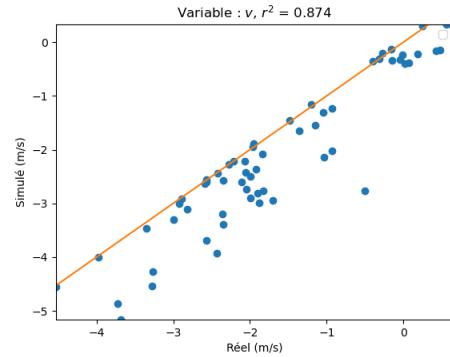


FIGURE 42 – Données simulées VS réelles pour la composante  $v$ .

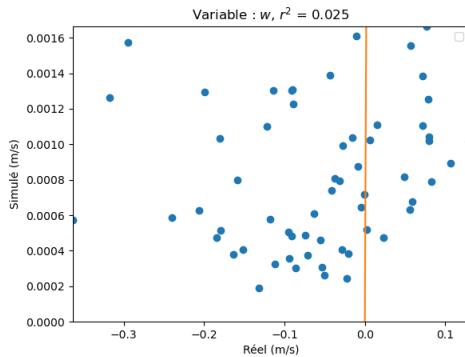


FIGURE 43 – Données simulées VS réelles pour la composante  $w$ .

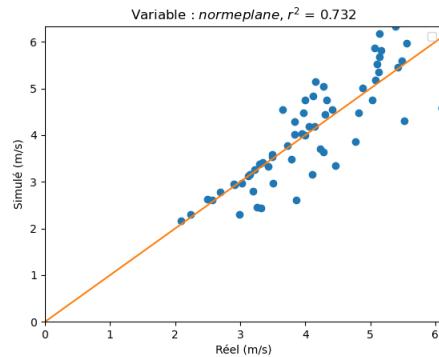


FIGURE 44 – Données simulées VS réelles pour la norme plane de la vitesse.

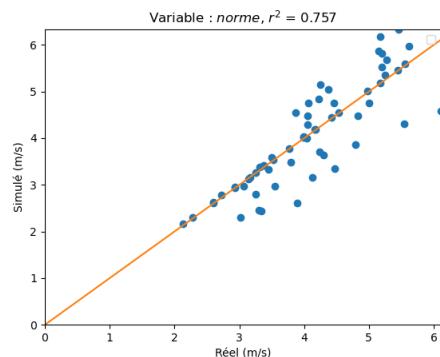


FIGURE 45 – Données simulées VS réelles pour la norme de la vitesse.

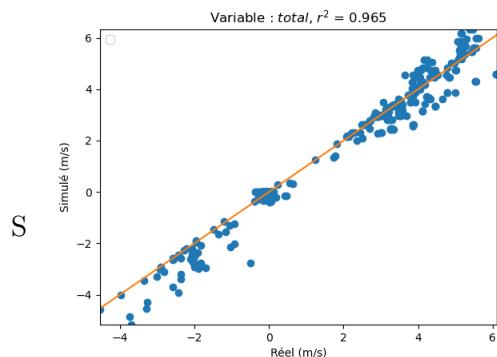


FIGURE 46 – Données simulées VS réelles pour toutes les variables.

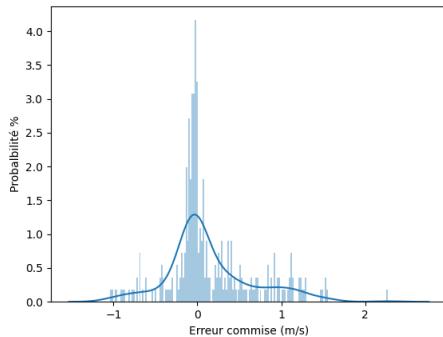


FIGURE 47 – Répartition de l'erreur commise par la simulation.

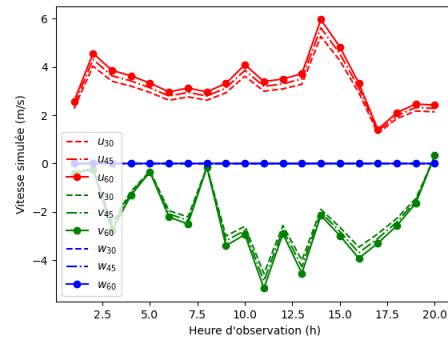


FIGURE 48 – Vitesse simulée en fonction de l'heure d'observation.

Variable	Nombre d'éléments	$r^2$	Corrélation
$u$	60	0.794	0.891
$v$	60	0.874	0.935
$w$	60	0.025	0.160
Norme	60	0.757	0.870
Norme plane	60	0.732	0.856
Ensemble des variables	360	0.965	0.982

TABLE 6 – Récapitulatif des résultats des tests

Les résultats obtenus sont très prometteurs : les variables semblent hautement corrélées (cf. Table 6).

Il faut cependant remarquer plusieurs éléments : ne disposant que de 60 éléments par variable, cette régression linéaire est très sensible à la variabilité des données, mais si l'on regarde l'intégralité des variables et de leur mesure (soit 360 éléments), ce résultat est beaucoup plus clair. Enfin, la variable  $w$  est extrêmement faible en amplitude (Fig. 40) et sa valeur est très sensible au micro-paramètres de l'environnement (climat, ensoleillement, taux d'humidité, heure de la journée etc.) qui sont extrêmement difficiles à modéliser dans des simulations raisonnables en temps et complexité, ce qui explique la corrélation si faible. Cependant, lorsque nous remettons la série de mesures en  $w$  dans le contexte de l'ensemble des variables ou simplement de la norme des vitesses, cette erreur est immédiatement absorbée.

Pour finir, nous pouvons observer sur la Figure 47 que les erreurs commises sont assez bien centrée autour de 0 m/s.

#### 4.1.3 Ajout de la turbulence

Les essais de la phase 1 ont été terminés en ajoutant à la composante stationnaire une composante turbulente. Celle-ci utilise les spectres développés précédemment, l'objectif était simplement de s'assurer que les amplitudes concordaient. Ces résultats apparaissent sur les Figures 49, 50 et 51.

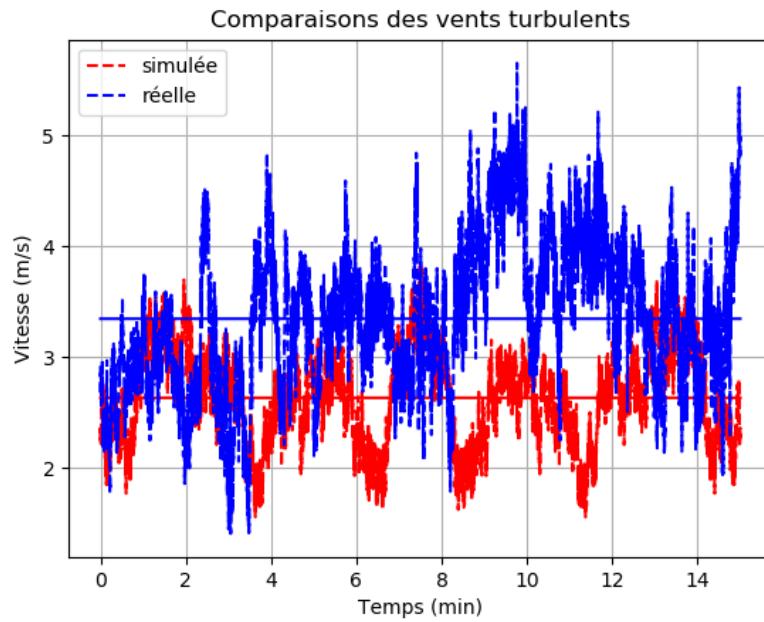


FIGURE 49 – Vitesse simulée et réelle sur un échantillon (composante  $u$ ).

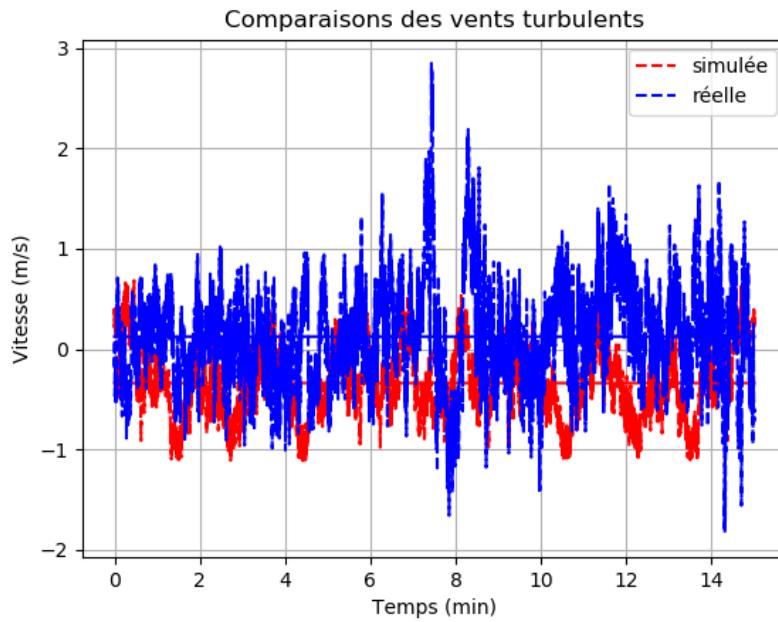


FIGURE 50 – Vitesse simulée et réelle sur un échantillon (composante  $v$ ).

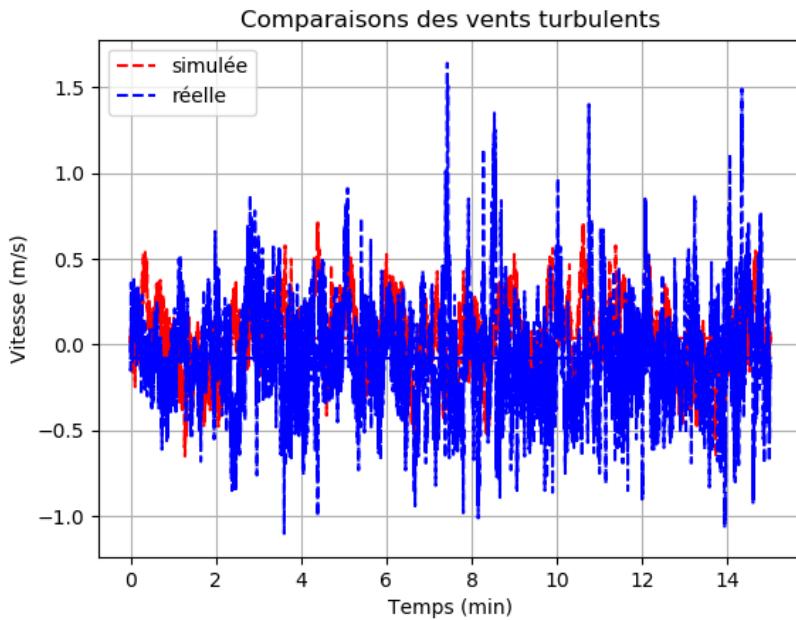


FIGURE 51 – Vitesse simulée et réelle sur un échantillon (composante  $w$ ).

Pour le même pas de temps  $dt = 0.1$  s, on observe des profils similaires. Les différences d'amplitudes restent raisonnables, de l'ordre de 10% à 20% : ceci vient des spécificités du lieu qui ne peuvent pas être totalement décrites par les modèles turbulents que nous avons adoptés. Notons qu'on distingue aussi un écart entre les moyennes temporelles : celui-ci vient simplement de l'erreur du vent stationnaire. Cette erreur n'est quasiment pas visible pour les essais sur la composante  $w$ , celle-ci ayant souvent une moyenne nulle.

## 4.2 Seconde phase d'essais

### 4.2.1 Présentation de la démarche

Pour cette phase d'essais, nous avons utilisé les données provenant d'aéroports, car les données étaient enregistrées et disponibles via une API (<https://developers.synopticdata.com/mesonet/>). Nous avons alors sélectionné dix aéroports français, pour tenir compte de différents facteurs : emplacements géographiques variés, données disponibles, temps de simulation raisonnable pour la somme de tous les aéroports (plus de 45 min de simulations pour les 10 aéroports sur un ordinateur performant) ...

- |                      |                          |
|----------------------|--------------------------|
| 1. Toulouse-Blagnac  | 6. Cherbourg - Maupertus |
| 2. Ajaccio           | 7. Quimper - Cornouaille |
| 3. Annecy            | 8. Strasbourg            |
| 4. Salon-de-Provence | 9. Vannes - Meucon       |
| 5. Paris Le Bourget  | 10. Cayenne Rochambeau   |

Pour chacun des aéroports, lorsque disponibles, la vitesse et la direction du vent furent récupérées, et pour obtenir suffisamment de données, la simulation fut répétée 3 fois :

1. 17/03/2021 à 15h00

2. 19/03/2021 à 18h00
3. 20/03/2021 à 09h00

#### 4.2.2 Présentation des résultats

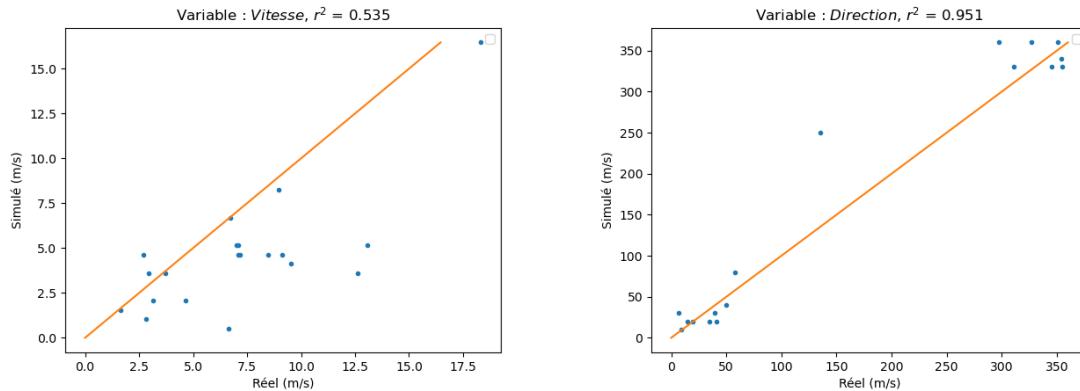


FIGURE 52 – Données simulées VS réelles pour la vitesse

FIGURE 53 – Données simulées VS réelles pour la direction

Variable	Nombre d'éléments	$r^2$	Corrélation
Vitesse	20	0.535	0.732
Direction	17	0.951	0.975

TABLE 7 – Récapitulatif des résultats des tests

Les directions de vent obtenus sont très hautement corrélées (Fig. 53), cependant les données concernant les mesures de vitesse du vent le sont beaucoup moins (0.732, cf. Table 7 et Fig. 52). Cela peut s'expliquer par plusieurs raisons : la vitesse peut être affectée par de nombreux facteurs : des rafales brusques et violentes qui augmentent la vitesse moyenne recensée par les stations, ou dans la façon même dont les données sont enregistrée par la base de données. De plus, se basant sur des relevés météorologiques, ces données ne sont pas exactes et peuvent avoir des erreurs qui ne sont pas dues à la simulation **WindNinja**.

## Conclusion

Ce rapport a permis de montrer l'ensemble du travail réalisé sur la conception d'un code permettant de simuler précisément le vent à partir de modèles météo et de fichiers numériques de terrains. L'étude bibliographique a permis de mettre en exergue l'intérêt des méthodes diagnostiques pour réaliser une interpolation du vent prenant en compte le terrain ainsi que la pertinence de l'approche statistique pour simuler un niveau de turbulence cohérent avec la réalité. Dans ce contexte, il est apparu intéressant d'utiliser le logiciel **WindNinja** développé par le Missoula FireLab pour réaliser la partie la plus délicate du problème, à savoir le calcul du vent à partir des données météo de terrains. Nous avons alors construit une classe Python permettant de créer et manipuler des cubes de vent ou `wind_cube` autour de ce logiciel. Lors de la création d'un nouveau `wind_cube`, notre code lance un calcul **WindNinja** à partir des paramètres d'entrées spécifiés dans un fichier JSON par l'utilisateur. Dans un second temps, ces résultats sont extrapolés en altitude à partir de lois empiriques pour permettre de répondre au cahier des charges de la DGA. Enfin, de nombreuses fonctions de simulations de la turbulence, d'interpolation et d'affichage graphiques sont également implémentées dans la classe pour permettre à l'utilisateur un usage optimal des `wind_cube`. Afin de valider le code ainsi créé, la fin de ce rapport se concentre sur une série de tests réalisés à partir de données expérimentales ou disponibles auprès de certains aéroports français. Il en ressort que notre code permet d'obtenir des données tout à fait cohérentes en vue de l'entraînement d'un correcteur : les deux phases d'essais permettent en particulier de montrer que la direction du vent est une donnée très bien restituée mais que la norme est plus ambitieuse à approcher. Une concordance parfaite entre les données expérimentales et les données simulées aurait été illusoire puisque les simulations reposent sur des données qui ne sont pas parfaitement fiables et parce que certains phénomènes ne peuvent être pris en compte.

La classe Python `wind_class` permettra ainsi de disposer de données précises et cohérentes avec le terrain pour l'entraînement de la loi de commande d'un parachute autoguidé ou le calcul de son point de largage. Cependant, il a été observé que les temps de calcul peuvent être relativement conséquents lors de l'utilisation des méthodes d'interpolation sur des maillages fins. Les méthodes utilisées étant celles implémentées de façon générique dans la librairie `scipy` de Python, la conception de méthodes adaptées à notre problème pourrait permettre d'améliorer ces temps de calcul. Enfin, pour l'instant, seul le solveur *Mass Conservative* de **WindNinja** peut-être utilisé car notre code ne peut pas récupérer les données retournées par le solveur *CFD*. Au vu du gain que peut représenter ce dernier pour le calcul sur des configurations topographiques complexes, il nous semblerait intéressant d'approfondir ce sujet, ce qui passerait par la lecture et l'exploitation de fichiers de type `SHAPEFILE`.

## Références

- [1] Fatma GüL AKGÜL et Birdal ŞENOĞLU. « Comparison of wind speed distributions : a case study for Aegean coast of Turkey ». In : *Energy Sources, Part A : Recovery, Utilization, and Environmental Effects* 0.0 (2019), p. 1-18. DOI : 10.1080/15567036.2019.1663309. eprint : <https://doi.org/10.1080/15567036.2019.1663309>. URL : <https://doi.org/10.1080/15567036.2019.1663309>.
- [2] H. B. AZAD, S. MEKHALEF et V. G. GANAPATHY. « Long-Term Wind Speed Forecasting and General Pattern Recognition Using Neural Networks ». In : *IEEE Transactions on Sustainable Energy* 5.2 (2014), p. 546-553. DOI : 10.1109/TSTE.2014.2300150.
- [3] Emmanuel BRANLARD. « Generation of Time Series from a Spectrum : Generation of Wind Time Series from the Kaimal Spectrum, Generation of Wave Time Series from Jonswap Spectrum ». In : (2010).
- [4] Barbara G. BROWN, Richard W. KATZ et Allan H. MURPHY. « Time Series Models to Simulate and Forecast Wind Speed and Wind Power ». In : *Journal of Climate and Applied Meteorology* 23.8 (août 1984), p. 1184-1195. ISSN : 0733-3021. DOI : 10.1175/1520-0450(1984)023<1184:TSMTSA>2.0.CO;2. eprint : [https://journals.ametsoc.org/jamc/article-pdf/23/8/1184/4977360/1520-0450\(1984\)023\\_1184\\_tsmtsa\\_2\\_0\\_co\\_2.pdf](https://journals.ametsoc.org/jamc/article-pdf/23/8/1184/4977360/1520-0450(1984)023_1184_tsmtsa_2_0_co_2.pdf). URL : [https://doi.org/10.1175/1520-0450\(1984\)023<1184:TSMTSA>2.0.CO;2](https://doi.org/10.1175/1520-0450(1984)023<1184:TSMTSA>2.0.CO;2).
- [5] B BRUTLE et al. *WindNinja*. URL : <https://weather.firelab.org/windninja/index.html>.
- [6] Arnaud Guillin CHRISTIAN CREMONA. « Développement d'algorithmes de simulation de champs de vitesse de vent ». In : *Atmospheric Chemistry and Physics* 16.8 (1997), p. 61-118.
- [7] Kenan COLE et Adam WICKENHEISER. *Spatio-Temporal Wind Modeling for UAV Simulations*. 2019. arXiv : 1905.09954 [math.OC].
- [8] Kenan COLE et Adam WICKENHEISER. *Spatio-Temporal Wind Modeling for UAV Simulations*. 2019. arXiv : 1905.09954 [math.OC].
- [9] Guglielmo D'AMICO, Filippo PETRONI et Flavio PRATTICO. « First and second order semi-Markov chains for wind speed modeling ». In : *Physica A : Statistical Mechanics and its Applications* 392.5 (2013), p. 1194 -1201. ISSN : 0378-4371. DOI : <https://doi.org/10.1016/j.physa.2012.11.022>. URL : <http://www.sciencedirect.com/science/article/pii/S0378437112009879>.
- [10] S FINARDI et al. « Evaluation of different wind field modeling techniques for wind energy applications over complex topography ». In : *Journal of Wind Engineering and Industrial Aerodynamics* 74 (1998), p. 283-294.
- [11] George FORRISTALL et Kevin EWANS. « Worldwide Measurements of Directional Wave Spreading ». In : *Journal of Atmospheric and Oceanic Technology - J ATMOS OCEAN TECHNOL* 15 (avr. 1998). DOI : 10.1175/1520-0426(1998)015<0440:WMODWS>2.0.CO;2.
- [12] Jason M FORTHOFER, Bret W BUTLER et Natalie S WAGENBRENNER. « A comparison of three approaches for simulating fine-scale surface winds in support of wildland fire management. Part I. Model formulation and comparison against measurements ». In : *International Journal of Wildland Fire* 23.7 (2014), p. 969-981.

- [13] K. FORTUNIAK et Włodzimierz PAWLAK. « Selected Spectral Characteristics of Turbulence over an Urbanized Area in the Centre of Łódź, Poland ». In : *Boundary-Layer Meteorology* 154 (août 2014), p. 137-156. DOI : 10.1007/s10546-014-9966-7.
- [14] P GEAI. *Methode D'interpolation et de Reconstruction Tridimensionnelle d'un Champ de Vent : le Code d'Analyse Objective MINERVE Report EDF/DER*. Rapp. tech. HE/34-87.03, 1987.
- [15] Muhammad HAQ et al. « Marshall-Olkin Power Lomax Distribution : Properties and Estimation Based on Complete and Censored Samples ». In : 9 (jan. 2020), p. 48-62. DOI : 10.5539/ijsp.v9n1p48.
- [16] F. O. HOCAOGLU, O. N. GEREK et M. KURBAN. « The Effect of Markov Chain State Size for Synthetic Wind Speed Generation ». In : *Proceedings of the 10th International Conference on Probabilistic Methods Applied to Power Systems*. 2008, p. 1-4.
- [17] John HOLMES. *Wind Loading of Structures*. CRC Press, 2001, p. 58-80.
- [18] G. HOMICZ. « Three-Dimensional Wind Field Modeling : A Review ». In : 2002.
- [19] Z. HUANG et Z.S. CHALABI. « Use of time-series analysis to model and forecast wind speed ». In : *Journal of Wind Engineering and Industrial Aerodynamics* 56.2 (1995), p. 311 -322. ISSN : 0167-6105. DOI : [https://doi.org/10.1016/0167-6105\(94\)00093-S](https://doi.org/10.1016/0167-6105(94)00093-S). URL : <http://www.sciencedirect.com/science/article/pii/016761059400093S>.
- [20] Pascal HÉMON. *Vibrations des structures couplées avec le vent*. Editions de l'Ecole Polytechnique, 2006, p. 36-45.
- [21] Pascal HÉMON et Françoise SANTI. « Simulation of a spatially correlated turbulent velocity field using biorthogonal decomposition ». In : *Journal of Wind Engineering and Industrial Aerodynamics* 95 (jan. 2007), p. 21-29. DOI : 10.1016/j.jweia.2006.04.003.
- [22] ESDU INTERNATIONAL. *Characteristics of wind speed in the lower layers of the atmosphere near the ground : strong winds (neutral atmosphere)*. 1982.
- [23] Hyun Goo KIM, V.C PATEL et Choung Mook LEE. « Numerical simulation of wind flow over hilly terrain ». In : *Journal of Wind Engineering and Industrial Aerodynamics* 87.1 (2000), p. 45 -60. ISSN : 0167-6105. DOI : [https://doi.org/10.1016/S0167-6105\(00\)00014-3](https://doi.org/10.1016/S0167-6105(00)00014-3). URL : <http://www.sciencedirect.com/science/article/pii/S0167610500000143>.
- [24] D.R. LEMELIN, D. SURRY et A.G. DAVENPORT. « Simple approximations for wind speed-up over hills ». In : *Journal of Wind Engineering and Industrial Aerodynamics* 28.1 (1988), p. 117 -127. ISSN : 0167-6105. DOI : [https://doi.org/10.1016/0167-6105\(88\)90108-0](https://doi.org/10.1016/0167-6105(88)90108-0). URL : <http://www.sciencedirect.com/science/article/pii/0167610588901080>.
- [25] Jakob MANN. « Wind field simulation ». In : *Probabilistic Engineering Mechanics* 13.4 (1998), p. 269 -282. ISSN : 0266-8920. DOI : [https://doi.org/10.1016/S0266-8920\(97\)00036-2](https://doi.org/10.1016/S0266-8920(97)00036-2). URL : <http://www.sciencedirect.com/science/article/pii/S0266892097000362>.
- [26] M. MAO et al. « Performance comparison of models for fast short-term wind speed prediction ». In : *2013 4th IEEE International Symposium on Power Electronics for Distributed Generation Systems (PEDG)*. 2013, p. 1-5. DOI : 10.1109/PEDG.2013.6785654.

- [27] S.H. PISHGAR-KOMLEH, A. KEYHANI et P. SEFEEDPARI. « Wind speed and power density analysis based on Weibull and Rayleigh distributions (a case study : Firouzkooh county of Iran) ». In : *Renewable and Sustainable Energy Reviews* 42.C (2015), p. 313-322. DOI : 10.1016/j.rser.2014.10.02. URL : <https://ideas.repec.org/a/eee/reneus/v42y2015icp313-322.html>.
- [28] R. I. PUTRI, A. PRIYADI et M. H. PURNOMO. « Stochastic Petri Nets for very short-term wind speed modeling ». In : *2015 IEEE International Conference on Computational Intelligence and Virtual Environments for Measurement Systems and Applications (CIVEMSA)*. 2015, p. 1-4. DOI : 10.1109/CIVEMSA.2015.7158619.
- [29] RESEARCHGATE. *From PSD to time series*. <https://www.researchgate.net/post/How-do-I-generate-time-series-data-from-given-PSD-of-random-vibration-input>.
- [30] Christine A SHERMAN. « A mass-consistent model for wind fields over complex terrain ». In : *Journal of applied meteorology* 17.3 (1978), p. 312-319.
- [31] Muhammad Ahsan UL HAQ et al. « Marshall–Olkin Power Lomax distribution for modeling of wind speed data ». In : *Energy Reports* 6 (2020), p. 1118 -1123. ISSN : 2352-4847. DOI : <https://doi.org/10.1016/j.egyr.2020.04.033>. URL : <http://www.sciencedirect.com/science/article/pii/S2352484720302365>.
- [32] Ramón VELÓ, Paz LÓPEZ et Francisco MASEDA. « Wind speed estimation using multilayer perceptron ». In : *Energy Conversion and Management* 81 (2014), p. 1 -9. ISSN : 0196-8904. DOI : <https://doi.org/10.1016/j.enconman.2014.02.017>. URL : <http://www.sciencedirect.com/science/article/pii/S0196890414001277>.
- [33] Natalie WAGENBRENNER, Jason FORTHOFER et Bret BUTLER. « Development and Evaluation of a RANS-based CFD solve in WindNinja ». In : (2019).
- [34] Natalie S WAGENBRENNER et al. « Downscaling surface wind predictions from numerical weather prediction models in complex terrain with WindNinja ». In : *Atmospheric Chemistry and Physics* 16.8 (2016), p. 5229-5241.

## Annexe A : Les autres options des Simulations WindNinja

Listing 1 – configFile.json exemple avec apport de fichier terrain("\*.tif") mais sans apport de fichier vent de la part de l'utilisateur avec l'initialisation moyenne sur le domaine (`domainAverageInitialization`).

```

1  {
2      "def": {
3          "version": "3.7.1",
4          "name": "LFBO",
5          "date": "2021-03-15",
6          "mntFile": true,
7          "windFile": false
8      },
9      "extrapolation": {
10         "nb_layer_extrapolation": 5,
11         "elevation_max": 1000.0
12     },
13     "windNinjaSimulations": {
14         "elevation_file": "path/fileName.tif",
15         "x_center": 43.6291,
16         "y_center": 1.3638,
17         "x_buffer": 1.0,
18         "y_buffer": 1.0,
19         "buffer_units": "kilometers",
20         "vegetation": "grass",
21         "mesh_choice": "medium",
22         "time_zone": "Europe/Paris",
23         "non_neutral_stability": false,
24         "initialization_method": "domainAverageInitialization",
25         "input_speed": 36.0,
26         "input_speed_units": "mps",
27         "input_direction": 50.0,
28         "input_wind_height": 10.0,
29         "units_input_wind_height": "m",
30         "output_wind_height": 30.0,
31         "units_output_wind_height": "m",
32         "output_speed_units": "mps",
33         "write_pdf_output": true,
34         "write_vtk_output": true,
35         "num_threads": 1
36     }
37 }
```

Listing 2 – configFile.json exemple avec apport de fichier terrain("\*.tif") et fichier vent("\*.nc") de la part de l'utilisateur avec l'initialisation donnée météo (wxModelInitialization).

```

1  {
2      "def": {
3          "version": "3.7.1",
4          "name": "LFBO",
5          "date": "2021-03-15",
6          "mntFile": true,
7          "windFile": true
8      },
9      "extrapolation": {
10         "nb_layer_extrapolation": 5,
11         "elevation_max": 1000.0
12     },
13     "windNinjaSimulations": {
14         "elevation_file": "path/fileName.tif",
15         "x_center": 43.6291,
16         "y_center": 1.3638,
17         "x_buffer": 1.0,
18         "y_buffer": 1.0,
19         "buffer_units": "kilometers",
20         "mesh_choice": "medium",
21         "vegetation": "grass",
22         "time_zone": "Europe/Paris",
23         "non_neutral_stability": false,
24         "initialization_method": "wxModelInitialization",
25         "forecast_filename": "path/fileName.nc",
26         "output_wind_height": 30.0,
27         "units_output_wind_height": "m",
28         "output_speed_units": "mps",
29         "write_pdf_output": true,
30         "write_vtk_output": true,
31         "num_threads": 1
32     }
33 }
```

Listing 3 – configFile.json exemple sans apport de fichier de la part de l'utilisateur avec l'initialisation météo (`wxModelInitialization`).

```

1  {
2      "def": {
3          "version": "3.7.1",
4          "name": "LFBO",
5          "date": "2021-03-15",
6          "mntFile": false,
7          "windFile": false
8      },
9      "extrapolation": {
10         "nb_layer_extrapolation": 5,
11         "elevation_max": 1000.0
12     },
13     "windNinjaSimulations": {
14         "fetch_elevation": "path/fileName.tif",
15         "elevation_source": "gmtd",
16         "x_center": 43.6291,
17         "y_center": 1.3638,
18         "x_buffer": 1.0,
19         "y_buffer": 1.0,
20         "buffer_units": "kilometers",
21         "mesh_choice": "medium",
22         "vegetation": "grass",
23         "time_zone": "Europe/Paris",
24         "non_neutral_stability": false,
25         "initialization_method": "wxModelInitialization",
26         "wx_model_type": "UCAR-GFS-GLOBAL-0.5-DEG",
27         "forecast_duration": 3,
28         "output_wind_height": 30.0,
29         "units_output_wind_height": "m",
30         "output_speed_units": "mps",
31         "write_pdf_output": true,
32         "write_vtk_output": true,
33         "num_threads": 1
34     }
35 }
```

**Annexe B : date\_time\_zonespec.csv**

TABLE 8 – Fuseaux horaires possibles.

ID	Décalage GMT
Africa/Abidjan	+00 :00 :00
Africa/Accra	+00 :00 :00
Africa/Addis_Ababa	+03 :00 :00
Africa/Algiers	+01 :00 :00
Africa/Asmara	+03 :00 :00
Africa/Bamako	+00 :00 :00
Africa/Bangui	+01 :00 :00
Africa/Banjul	+00 :00 :00
Africa/Bissau	+00 :00 :00
Africa/Blantyre	+02 :00 :00
Africa/Brazzaville	+01 :00 :00
Africa/Bujumbura	+02 :00 :00
Africa/Cairo	+02 :00 :00
Africa/Casablanca	+00 :00 :00
Africa/Ceuta	+01 :00 :00
Africa/Conakry	+00 :00 :00
Africa/Dakar	+00 :00 :00
Africa/Dar_es_Salaam	+03 :00 :00
Africa/Djibouti	+03 :00 :00
Africa/Douala	+01 :00 :00
Africa/El_Aaiun	+00 :00 :00
Africa/Freetown	+00 :00 :00
Africa/Gaborone	+02 :00 :00
Africa/Harare	+02 :00 :00
Africa/Johannesburg	+02 :00 :00
Africa/Kampala	+03 :00 :00
Africa/Khartoum	+03 :00 :00
Africa/Kigali	+02 :00 :00
Africa/Kinshasa	+01 :00 :00
Africa/Lagos	+01 :00 :00
Africa/Libreville	+01 :00 :00
Africa/Lome	+00 :00 :00
Africa/Luanda	+01 :00 :00
Africa/Lubumbashi	+02 :00 :00
Africa/Lusaka	+02 :00 :00
Africa/Malabo	+01 :00 :00
Africa/Maputo	+02 :00 :00
Africa/Maseru	+02 :00 :00
Africa/Mbabane	+02 :00 :00
Africa/Mogadishu	+03 :00 :00
Africa/Monrovia	+00 :00 :00
Suite à la page suivante.	

**TABLE 8 – Suite de la page précédente.**

ID	Décalage GMT
Africa/Nairobi	+03 :00 :00
Africa/Ndjamena	+01 :00 :00
Africa/Niamey	+01 :00 :00
Africa/Nouakchott	+00 :00 :00
Africa/Ouagadougou	+00 :00 :00
Africa/Porto-Novo	+01 :00 :00
Africa/Sao_Tome	+00 :00 :00
Africa/Timbuktu	+00 :00 :00
Africa/Tripoli	+02 :00 :00
Africa/Tunis	+01 :00 :00
Africa/Windhoek	+01 :00 :00
America/Adak	-10 :00 :00
America/Anchorage	-09 :00 :00
America/Anguilla	-04 :00 :00
America/Antigua	-04 :00 :00
America/Araguaina	-03 :00 :00
America/Aruba	-04 :00 :00
America/Asuncion	-04 :00 :00
America/Barbados	-04 :00 :00
America/Belem	-03 :00 :00
America/Belize	-06 :00 :00
America/Boa_Vista	-04 :00 :00
America/Bogota	-05 :00 :00
America/Boise	-07 :00 :00
America/Buenos_Aires	-03 :00 :00
America/Cambridge_Bay	-07 :00 :00
America/Cancun	-06 :00 :00
America/Caracas	-04 :00 :00
America/Catamarca	-03 :00 :00
America/Cayenne	-03 :00 :00
America/Cayman	-05 :00 :00
America/Chicago	-06 :00 :00
America/Chihuahua	-07 :00 :00
America/Cordoba	-03 :00 :00
America/Costa_Rica	-06 :00 :00
America/Cuiaba	-04 :00 :00
America/Curacao	-04 :00 :00
America/Danmarkshavn	+00 :00 :00
America/Dawson	-08 :00 :00
America/Dawson_Creek	-07 :00 :00
America/Denver	-07 :00 :00
America/Detroit	-05 :00 :00
America/Dominica	-04 :00 :00
America/Edmonton	-07 :00 :00
America/Eirunepe	-05 :00 :00
Suite à la page suivante.	

**TABLE 8 – Suite de la page précédente.**

ID	Décalage GMT
America/El_Salvador	-06 :00 :00
America/Fortaleza	-03 :00 :00
America/Glace_Bay	-04 :00 :00
America/Godthab	-03 :00 :00
America/Goose_Bay	-04 :00 :00
America/Grand_Turk	-05 :00 :00
America/Grenada	-04 :00 :00
America/Guadeloupe	-04 :00 :00
America/Guatemala	-06 :00 :00
America/Guayaquil	-05 :00 :00
America/Guyana	-04 :00 :00
America/Halifax	-04 :00 :00
America/Havana	-05 :00 :00
America/Hermosillo	-07 :00 :00
America/Indiana/Indianapolis	-05 :00 :00
America/Indiana/Knox	-05 :00 :00
America/Indiana/Marengo	-05 :00 :00
America/Indiana/Vevay	-05 :00 :00
America/Indianapolis	-05 :00 :00
America/Inuvik	-07 :00 :00
America/Iqaluit	-05 :00 :00
America/Jamaica	-05 :00 :00
America/Jujuy	-03 :00 :00
America/Juneau	-09 :00 :00
America/Kentucky/Louisville	-05 :00 :00
America/Kentucky/Monticello	-05 :00 :00
America/La_Paz	-04 :00 :00
America/Lima	-05 :00 :00
America/Los_Angeles	-08 :00 :00
America/Louisville	-05 :00 :00
America/Maceio	-03 :00 :00
America/Managua	-06 :00 :00
America/Manaus	-04 :00 :00
America/Martinique	-04 :00 :00
America/Mazatlan	-07 :00 :00
America/Mendoza	-03 :00 :00
America/Menominee	-06 :00 :00
America/Merida	-06 :00 :00
America/Mexico_City	-06 :00 :00
America/Miquelon	-03 :00 :00
America/Monterrey	-06 :00 :00
America/Montevideo	-03 :00 :00
America/Montreal	-05 :00 :00
America/Montserrat	-04 :00 :00
America/Nassau	-05 :00 :00
Suite à la page suivante.	

**TABLE 8 – Suite de la page précédente.**

ID	Décalage GMT
America/New_York	-05 :00 :00
America/Nipigon	-05 :00 :00
America/Nome	-09 :00 :00
America/Noronha	-02 :00 :00
America/North_Dakota/Center	-06 :00 :00
America/Panama	-05 :00 :00
America/Pangnirtung	-05 :00 :00
America/Paramaribo	-03 :00 :00
America/Phoenix	-07 :00 :00
America/Port-au-Prince	-05 :00 :00
America/Port_of_Spain	-04 :00 :00
America/Porto_Velho	-04 :00 :00
America/Puerto_Rico	-04 :00 :00
America/Rainy_River	-06 :00 :00
America/Rankin_Inlet	-06 :00 :00
America/Recife	-03 :00 :00
America/Regina	-06 :00 :00
America/Rio_Branco	-05 :00 :00
America/Rosario	-03 :00 :00
America/Santiago	-04 :00 :00
America/Santo_Domingo	-04 :00 :00
America/Sao_Paulo	-03 :00 :00
America/Scoresbysund	-01 :00 :00
America/Shiprock	-07 :00 :00
America/St_Johns	-03 :30 :00
America/St_Kitts	-04 :00 :00
America/St_Lucia	-04 :00 :00
America/St_Thomas	-04 :00 :00
America/St_Vincent	-04 :00 :00
America/Swift_Current	-06 :00 :00
America/Tegucigalpa	-06 :00 :00
America/Thule	-04 :00 :00
America/Thunder_Bay	-05 :00 :00
America/Tijuana	-08 :00 :00
America/Tortola	-04 :00 :00
America/Vancouver	-08 :00 :00
America/Whitehorse	-08 :00 :00
America/Winnipeg	-06 :00 :00
America/Yakutat	-09 :00 :00
America/Yellowknife	-07 :00 :00
Antarctica/Casey	+08 :00 :00
Antarctica/Davis	+07 :00 :00
Antarctica/DumontDUrville	+10 :00 :00
Antarctica/Mawson	+06 :00 :00
Antarctica/Mcmurdo	+12 :00 :00
Suite à la page suivante.	

**TABLE 8 – Suite de la page précédente.**

ID	Décalage GMT
Antarctica/Palmer	-04 :00 :00
Antarctica/South_Pole	+12 :00 :00
Antarctica/Syowa	+03 :00 :00
Antarctica/Vostok	+06 :00 :00
Arctic/Longyearbyen	+01 :00 :00
Asia/Aden	+03 :00 :00
Asia/Almaty	+06 :00 :00
Asia/Amman	+02 :00 :00
Asia/Anadyr	+12 :00 :00
Asia/Aqtau	+04 :00 :00
Asia/Aqtobe	+05 :00 :00
Asia/Ashgabat	+05 :00 :00
Asia/Baghdad	+03 :00 :00
Asia/Bahrain	+03 :00 :00
Asia/Baku	+04 :00 :00
Asia/Bangkok	+07 :00 :00
Asia/Beirut	+02 :00 :00
Asia/Bishkek	+05 :00 :00
Asia/Brunei	+08 :00 :00
Asia/Calcutta	+05 :30 :00
Asia/Choibalsan	+09 :00 :00
Asia/Chongqing	+08 :00 :00
Asia/Colombo	+06 :00 :00
Asia/Damascus	+02 :00 :00
Asia/Dhaka	+06 :00 :00
Asia/Dili	+09 :00 :00
Asia/Dubai	+04 :00 :00
Asia/Dushanbe	+05 :00 :00
Asia/Gaza	+02 :00 :00
Asia/Harbin	+08 :00 :00
Asia/Hong_Kong	+08 :00 :00
Asia/Hovd	+07 :00 :00
Asia/Irkutsk	+08 :00 :00
Asia/Istanbul	+02 :00 :00
Asia/Jakarta	+07 :00 :00
Asia/Jayapura	+09 :00 :00
Asia/Jerusalem	+02 :00 :00
Asia/Kabul	+04 :30 :00
Asia/Kamchatka	+12 :00 :00
Asia/Karachi	+05 :00 :00
Asia/Kashgar	+08 :00 :00
Asia/Katmandu	+05 :45 :00
Asia/Krasnoyarsk	+07 :00 :00
Asia/Kuala_Lumpur	+08 :00 :00
Asia/Kuching	+08 :00 :00
Suite à la page suivante.	

**TABLE 8 – Suite de la page précédente.**

ID	Décalage GMT
Asia/Kuwait	+03 :00 :00
Asia/Macao	+08 :00 :00
Asia/Macau	+08 :00 :00
Asia/Magadan	+11 :00 :00
Asia/Makassar	+08 :00 :00
Asia/Manila	+08 :00 :00
Asia/Muscat	+04 :00 :00
Asia/Nicosia	+02 :00 :00
Asia/Novosibirsk	+06 :00 :00
Asia/Omsk	+06 :00 :00
Asia/Oral	+05 :00 :00
Asia/Phnom_Penh	+07 :00 :00
Asia/Pontianak	+07 :00 :00
Asia/Pyongyang	+09 :00 :00
Asia/Qyzylorda	+06 :00 :00
Asia/Qatar	+03 :00 :00
Asia/Rangoon	+06 :30 :00
Asia/Riyadh	+03 :00 :00
Asia/Saigon	+07 :00 :00
Asia/Sakhalin	+10 :00 :00
Asia/Samarkand	+05 :00 :00
Asia/Seoul	+09 :00 :00
Asia/Shanghai	+08 :00 :00
Asia/Singapore	+08 :00 :00
Asia/Taipei	+08 :00 :00
Asia/Tashkent	+05 :00 :00
Asia/Tbilisi	+04 :00 :00
Asia/Tehran	+03 :30 :00
Asia/Thimphu	+06 :00 :00
Asia/Tokyo	+09 :00 :00
Asia/Ujung_Pandang	+08 :00 :00
Asia/Ulaanbaatar	+08 :00 :00
Asia/Urumqi	+08 :00 :00
Asia/Vientiane	+07 :00 :00
Asia/Vladivostok	+10 :00 :00
Asia/Yakutsk	+09 :00 :00
Asia/Yekaterinburg	+05 :00 :00
Asia/Yerevan	+04 :00 :00
Atlantic/Azores	-01 :00 :00
Atlantic/Bermuda	-04 :00 :00
Atlantic/Canary	+00 :00 :00
Atlantic/Cape_Verde	-01 :00 :00
Atlantic/Faeroe	+00 :00 :00
Atlantic/Jan_Mayen	+01 :00 :00
Atlantic/Madeira	+00 :00 :00
Suite à la page suivante.	

**TABLE 8 – Suite de la page précédente.**

ID	Décalage GMT
Atlantic/Reykjavik	+00 :00 :00
Atlantic/South_Georgia	-02 :00 :00
Atlantic/St_Helena	+00 :00 :00
Atlantic/Stanley	-04 :00 :00
Australia/Adelaide	+09 :30 :00
Australia/Brisbane	+10 :00 :00
Australia/Broken_Hill	+09 :30 :00
Australia/Darwin	+09 :30 :00
Australia/Hobart	+10 :00 :00
Australia/Lindeman	+10 :00 :00
Australia/Lord_Howe	+10 :30 :00
Australia/Melbourne	+10 :00 :00
Australia/Perth	+08 :00 :00
Australia/Sydney	+10 :00 :00
Europe/Amsterdam	+01 :00 :00
Europe/Andorra	+01 :00 :00
Europe/Athens	+02 :00 :00
Europe/Belfast	+00 :00 :00
Europe/Belgrade	+01 :00 :00
Europe/Berlin	+01 :00 :00
Europe/Bratislava	+01 :00 :00
Europe/Brussels	+01 :00 :00
Europe/Bucharest	+02 :00 :00
Europe/Budapest	+01 :00 :00
Europe/Chisinau	+02 :00 :00
Europe/Copenhagen	+01 :00 :00
Europe/Dublin	+00 :00 :00
Europe/Gibraltar	+01 :00 :00
Europe/Helsinki	+02 :00 :00
Europe/Istanbul	+02 :00 :00
Europe/Kaliningrad	+02 :00 :00
Europe/Kiev	+02 :00 :00
Europe/Lisbon	+00 :00 :00
Europe/Ljubljana	+01 :00 :00
Europe/London	+00 :00 :00
Europe/Luxembourg	+01 :00 :00
Europe/Madrid	+01 :00 :00
Europe/Malta	+01 :00 :00
Europe/Minsk	+02 :00 :00
Europe/Monaco	+01 :00 :00
Europe/Moscow	+03 :00 :00
Europe/Nicosia	+02 :00 :00
Europe/Oslo	+01 :00 :00
Europe/Paris	+01 :00 :00
Europe/Prague	+01 :00 :00
Suite à la page suivante.	

**TABLE 8 – Suite de la page précédente.**

ID	Décalage GMT
Europe/Riga	+02 :00 :00
Europe/Rome	+01 :00 :00
Europe/Samara	+04 :00 :00
Europe/San_Marino	+01 :00 :00
Europe/Sarajevo	+01 :00 :00
Europe/Simferopol	+02 :00 :00
Europe/Skopje	+01 :00 :00
Europe/Sofia	+02 :00 :00
Europe/Stockholm	+01 :00 :00
Europe/Tallinn	+02 :00 :00
Europe/Tirane	+01 :00 :00
Europe/Uzhgorod	+02 :00 :00
Europe/Vaduz	+01 :00 :00
Europe/Vatican	+01 :00 :00
Europe/Vienna	+01 :00 :00
Europe/Vilnius	+02 :00 :00
Europe/Warsaw	+01 :00 :00
Europe/Zagreb	+01 :00 :00
Europe/Zaporozhye	+02 :00 :00
Europe/Zurich	+01 :00 :00
Indian/Antananarivo	+03 :00 :00
Indian/Chagos	+06 :00 :00
Indian/Christmas	+07 :00 :00
Indian/Cocos	+06 :30 :00
Indian/Comoro	+03 :00 :00
Indian/Kerguelen	+05 :00 :00
Indian/Mahe	+04 :00 :00
Indian/Maldives	+05 :00 :00
Indian/Mauritius	+04 :00 :00
Indian/Mayotte	+03 :00 :00
Indian/Reunion	+04 :00 :00
Pacific/Apia	-11 :00 :00
Pacific/Auckland	+12 :00 :00
Pacific/Chatham	+12 :45 :00
Pacific/Easter	-06 :00 :00
Pacific/Efate	+11 :00 :00
Pacific/Enderbury	+13 :00 :00
Pacific/Fakaofo	-10 :00 :00
Pacific/Fiji	+12 :00 :00
Pacific/Funafuti	+12 :00 :00
Pacific/Galapagos	-06 :00 :00
Pacific/Gambier	-09 :00 :00
Pacific/Guadalcanal	+11 :00 :00
Pacific/Guam	+10 :00 :00
Pacific/Honolulu	-10 :00 :00
Suite à la page suivante.	

**TABLE 8 – Suite de la page précédente.**

ID	Décalage GMT
Pacific/Johnston	-10 :00 :00
Pacific/Kiritimati	+14 :00 :00
Pacific/Kosrae	+11 :00 :00
Pacific/Kwajalein	+12 :00 :00
Pacific/Majuro	+12 :00 :00
Pacific/Marquesas	-09 :30 :00
Pacific/Midway	-11 :00 :00
Pacific/Nauru	+12 :00 :00
Pacific/Niue	-11 :00 :00
Pacific/Norfolk	+11 :30 :00
Pacific/Noumea	+11 :00 :00
Pacific/Pago_Pago	-11 :00 :00
Pacific/Palau	+09 :00 :00
Pacific/Pitcairn	-08 :00 :00
Pacific/Ponape	+11 :00 :00
Pacific/Port_Moresby	+10 :00 :00
Pacific/Rarotonga	-10 :00 :00
Pacific/Saipan	+10 :00 :00
Pacific/Tahiti	-10 :00 :00
Pacific/Tarawa	+12 :00 :00
Pacific/Tongatapu	+13 :00 :00
Pacific/Truk	+10 :00 :00
Pacific/Wake	+12 :00 :00
Pacific/Wallis	+12 :00 :00
Pacific/Yap	+10 :00 :00
GMT	+00 :00 :00
UTC	+00 :00 :00