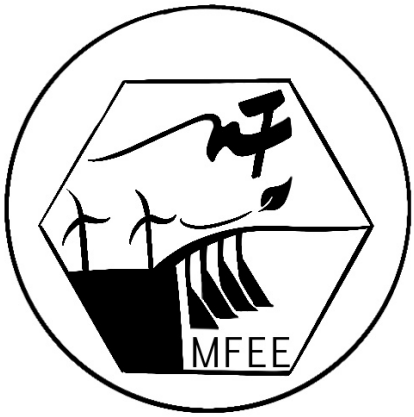
TOULOUSE INP N7

STATISTIQUES

Bureau d'étude : Étude statistique de la vitesse du vent

Laura MATHIEU
Grégoire HUSSER

Département : MFEE
Année : 2022-2023



Mars 2023

Table des matières

Partie 1

Introduction

Il est possible de modéliser la vitesse du vent par une variable aléatoire, notée Y, suivant une loi de Weibull de paramètres θ et p, que l'on note $\mathcal{W}(\theta, p)$, représentée sur l'histogramme 1.1. L'objectif de ce bureau d'étude est, tout d'abord, d'analyser certaines propriétés d'un estimateur du paramètre θ, pour p fixé, obtenu à partir de n données suivant une loi de Weibull $\mathcal{W}(\theta, p)$. Il est ensuite question d'étudier un test statistique permettant de déterminer si on est dans une période de vent calme, correspondant à $\theta < 1\ m.s^{-1}$, ou une période de vent fort, correspondant à $\theta > 1\ m.s^{-1}$. Enfin, le but est d'analyser un fichier de données contenant des vitesses de vent.

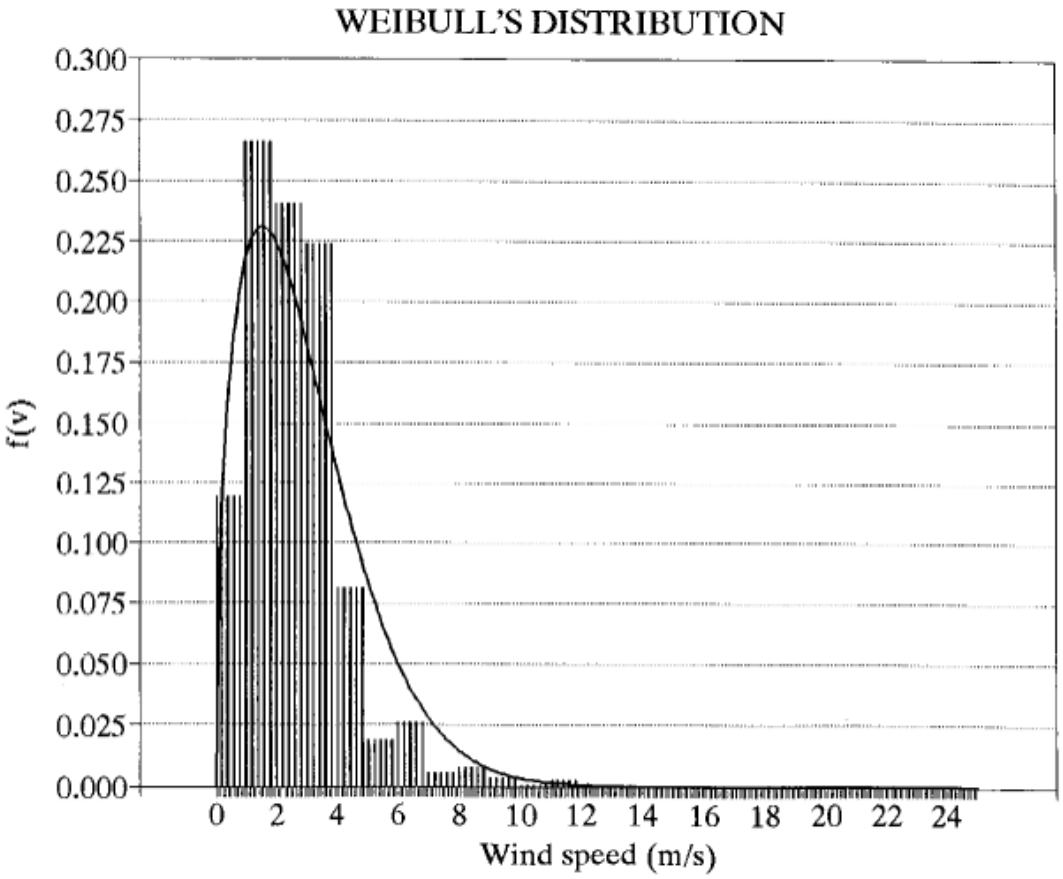


FIGURE 1.1 – Histogramme de la vitesse de vent comparé à la densité de Weibull [1]

Partie 2

Génération d'un signal test

On cherche à générer un signal test reproduisant des mesures de vent réelles. On considère que la vitesse du vent peut être modélisée par une loi de Weibull, l'objectif est alors de produire une matrice dont chaque colonne contient une réalisation du signal.

On obtient cela grâce à la fonction Y = *generer(θ, p, N, K)* (voir annexes 8.4.2). Cette fonction a pour paramètres d'entrée les paramètres θ et p de la loi de Weibull, le nombre K de réalisations et le nombre N de points de chaque réalisation. Pour coder cette fonction, on utilise le fait que si $F(x, \theta, p)$ est la fonction de répartition d'une loi de Weibull $\mathcal{W}(\theta = 3.3, p = 1.5)$ et que X est une variable aléatoire suivant la loi uniforme $\mathcal{U}(]0, 1[)$, alors $Y = F^{-1}(X, \theta, p)$ suit la loi de Weibull $\mathcal{W}(\theta = 3.3, p = 1.5)$. De plus, on utilise la fonction $rand(N, K)$ de Matlab qui génère une matrice de taille $N \times K$ contenant K réalisations indépendantes d'une loi uniforme $\mathcal{U}(]0, 1[)$.

On peut vérifier le fonctionnement de cette fonction en comparant l'histogramme des valeurs générées à partir de la fonction en prenant pour paramètres $\theta = 3.3$, $p = 1.5$, $N = 1000$ et $K = 1$ avec la densité d'une loi de Weibull $\mathcal{W}(\theta = 3.3, p = 1.5)$. On obtient alors le graphique 2.1.

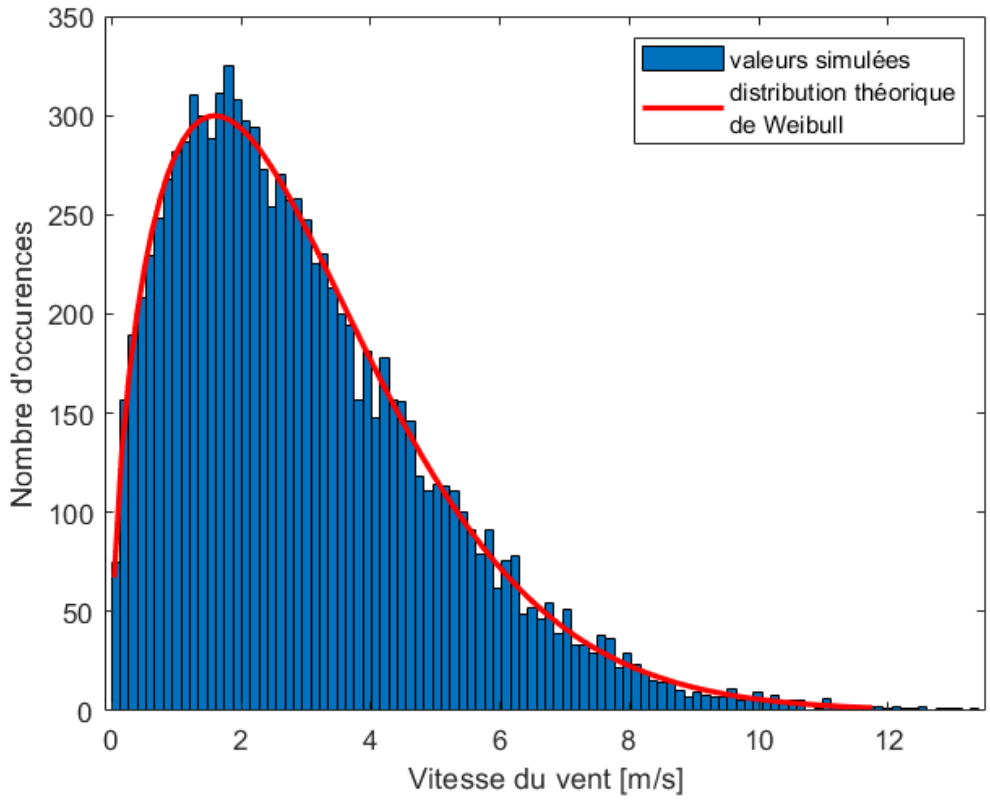


FIGURE 2.1 – Histogramme de la vitesse de vent obtenu à l'aide de la fonction *generer*

On remarque que l'histogramme des valeurs simulées correspondent à la densité de probabilité d'une loi de Weibull. La fonction *generer* répond donc bien à nos attentes.

Un autre moyen de vérifier la cohérence de la fonction est de comparer la moyenne et la variance des valeurs obtenues avec la moyenne et la variance théorique d'une loi de Weibull.

On calcule la moyenne des valeurs simulées en utilisant la fonction *mean* de Matlab et la variance avec la fonction *var*. Dans un second temps, on utilise les formules théoriques d'une loi de Weibull :

- Moyenne : $\mu = \theta\ \Gamma(1 + \frac{1}{p})$
- Variance : $\nu = \theta^2\ \Gamma(1 + \frac{2}{p}) - \mu^2$

Avec les mêmes valeurs que précédemment, on obtient :

- Moyenne : $E_{th}(Y) = 2,98$ et $E_{num}(Y) = 2,97$
- Variance : $Var_{th}(Y) = 4,09$ et $Var_{num}(Y) = 4,01$

On peut également représenter graphiquement les moyennes et variances des K réalisations sur la figure 2.2 suivante :

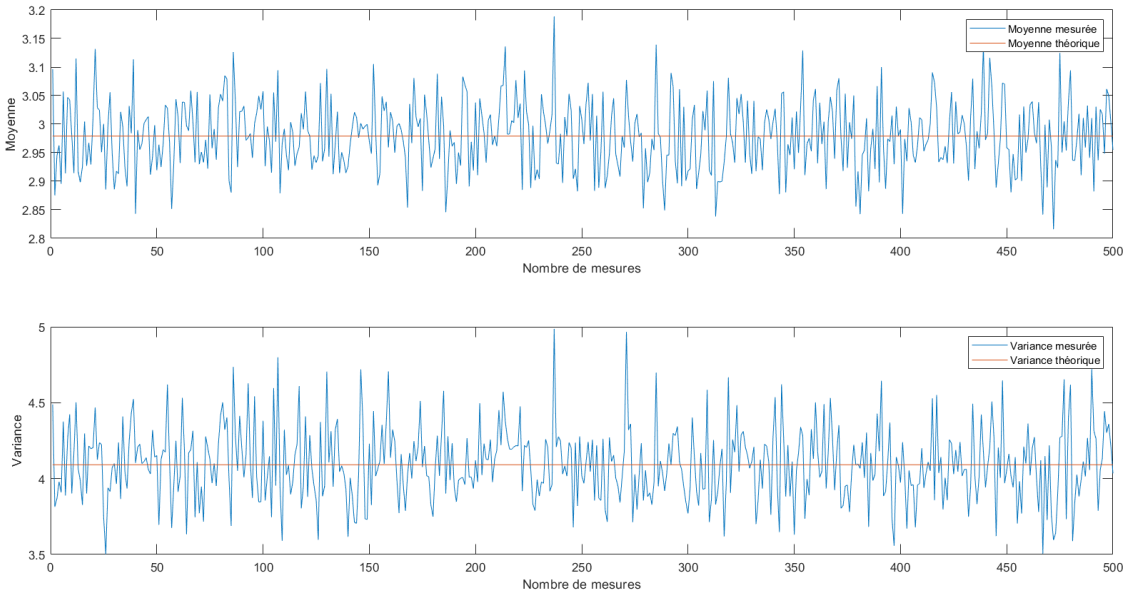


FIGURE 2.2 – Comparaison de la moyenne et de la variance des valeurs théoriques et générées

On trouve des valeurs très proches avec les deux méthodes. C'est donc un argument de plus qui confirme le bon fonctionnement de notre fonction.

Partie 3

Estimation statistique

Dans cette partie, on souhaite étudier l'estimateur du maximum de vraisemblance de $a = \theta^p$, dont on connaît également la moyenne et la variance puisque les calculs sont répertoriés en annexe 8.1.

Pour cela, nous avons créé une fonction sur Matlab nommée *alpha_est = estimateur_mv(Y, p, N, K)*. Cette fonction a pour paramètres d'entrée la matrice Y contenant les valeurs suivant une loi de Weibull, le paramètre de la loi de Weibull p, le nombre N de points de chaque réalisation et enfin K le nombre de réalisations. Elle renvoie ensuite l'estimateur du maximum de vraisemblance $\hat{a}_{MV}(k)$ pour chaque réalisation k.

On peut tester la fonction en fixant des paramètres et en comparant la moyenne et la variance des $\hat{a}_{MV}(k)$ obtenus avec la moyenne et la variance théorique. On choisit ici $\theta = 3,3$, $p = 1,5$, $K = 500$ et $N = 1000$, et on obtient le graphique représenté en figure 3.1.

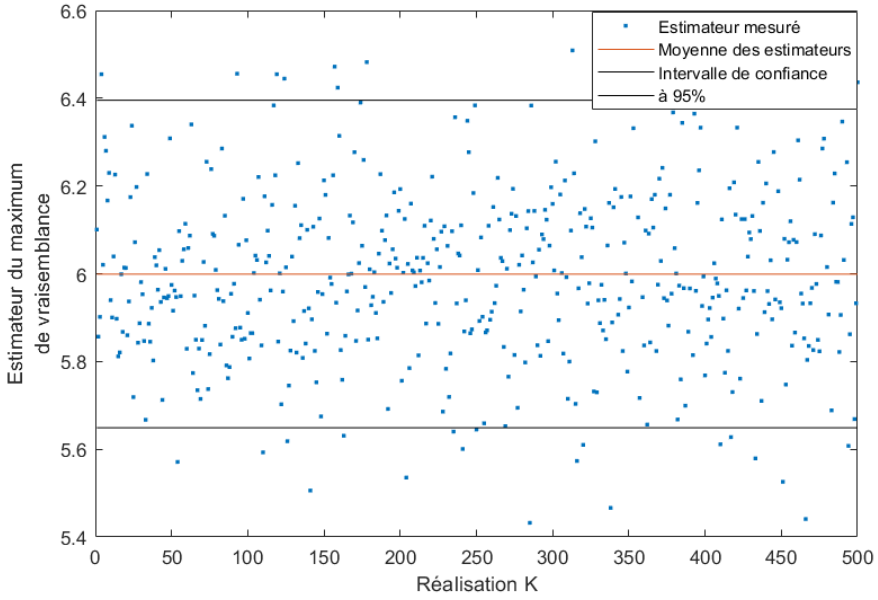


FIGURE 3.1 – Représentation des estimateurs de maximum de vraisemblance pour les différentes réalisations K

On remarque, à partir de ce graphique, que les valeurs de $\hat{a}_{MV}(k)$ sont quasiment toutes comprises dans l'intervalle de confiance à 95 %, déterminé en annexe 8.2, et sont en majorité réparties autour de la moyenne théorique. On peut également calculer la moyenne et la variance en utilisant les fonctions *mean* et *var* de Matlab pour les comparer aux valeurs des moyenne et variance théoriques calculée avec les formules déterminées en annexe 8.1 :

- Moyenne : $E_{th}(\hat{a}_{MV}) = \theta^p = 5,995$ et $E_{num}(\hat{a}_{MV}) = 5,985$
- Variance : $Var_{th}(\hat{a}_{MV}) = \frac{\theta^{2p}}{N} = 0,0359$ et $Var_{num}(\hat{a}_{MV}) = 0,0327$

Au vu de la concordance des résultats, la fonction créée renvoie bien les valeurs de l'estimateur de maximum de vraisemblance.

Partie 4

Détection

L'objectif de cette partie est d'étudier les performances d'un test statistique qui permet de détecter un vent calme, de vitesse $a_0 < 1\ m.s^{-1}$, ou un vent fort, de vitesse $a_1 > 1\ m.s^{-1}$, à partir des données $y_1,\ y_2, ...,\ y_N$. Les hypothèses de ce test sont :

$$H_0 : a = a_0 \qquad H_1 : a = a_1 > a_0$$

L'étude théorique de ce test est décrite en annexe 8.3.

Pour l'étude de ce test, on souhaite écrire deux fonctions sur Matlab qui calculent la puissance du test statistique, théorique pour l'un et celle obtenue à partir des données générées par la fonction *generer* étudiée dans la partie 2.

4.1 Puissance théorique du test

Pour obtenir la puissance théorique du test, on crée la fonction $pi\_theorique(a_0, a_1, L)$ qui prend en paramètres d'entrée les valeurs de a_0 et a_1 les vitesses de vent relatives aux hypothèses respectives H_0 et H_1, et L le nombre de degrés de liberté de la loi du chi2 utilisée dans la fonction. En sortie, on obtient la puissance théorique π du test pour des valeurs de probabilité de fausse alarme α comprises dans $\{0,01\ ;\ 0,02\ ;\ ...\ ;\ 0,99\}$. À partir de ces valeurs de π calculées pour différents α, on peut tracer les courbes COR du test statistique.

On trace les courbes COR en fixant les paramètres $a_0 = 0,9$ et $a_1 = 1,5$ pour différentes valeurs de $N \in \{10\ ;\ 20\ ;\ 50\}$, comme représenté sur le graphique 4.1.

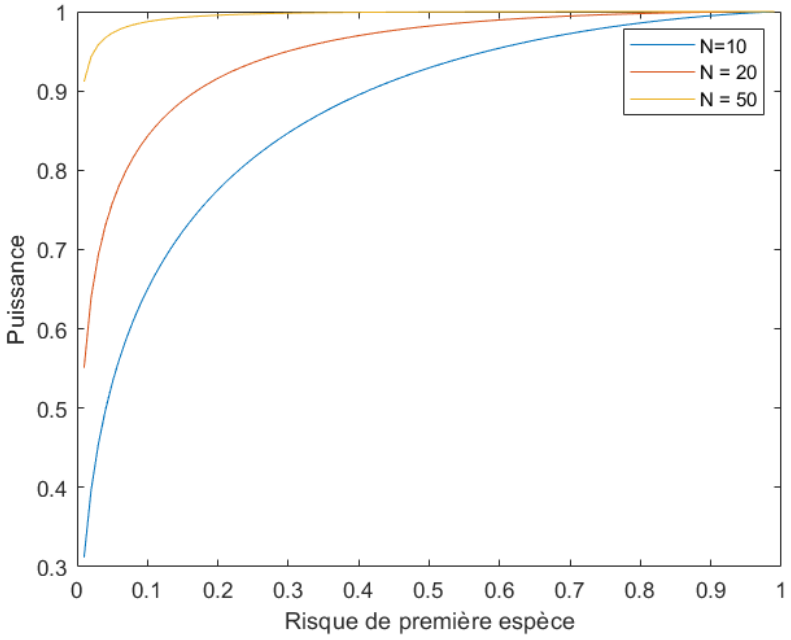


FIGURE 4.1 – Courbes COR pour différentes valeurs de N

Plus N est grand, plus la puissance du test est grande pour α fixé. Ce constat est cohérent puisqu'il y a plus de données donc les résultats sont plus fiables.

On trace les courbes COR en fixant les paramètres $a_0 = 0,9$ et $N = 20$ pour différentes valeurs de $a_1 \in \{1,2\ ;\ 1,5\ ;\ 2\}$, comme représenté sur le graphique 4.2.

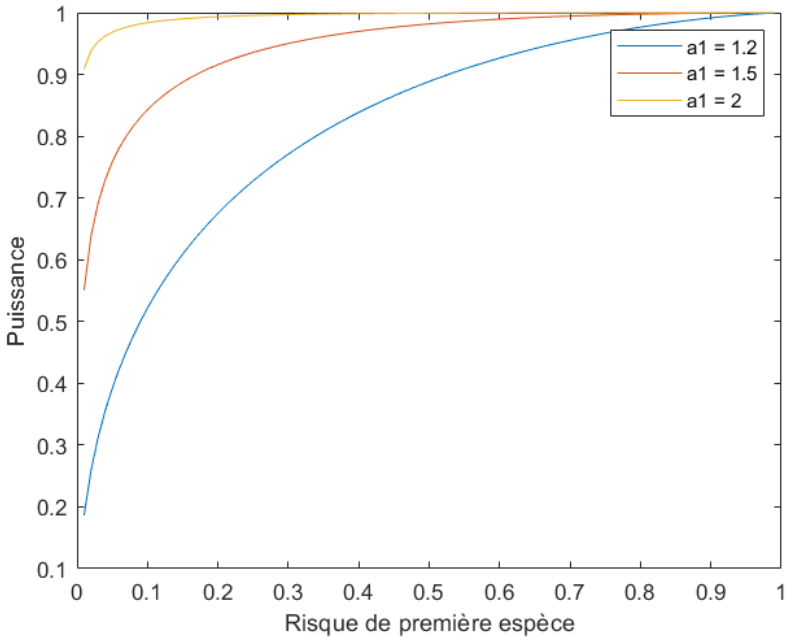


FIGURE 4.2 – Courbes COR pour différentes valeurs de a_1

Plus a_1 est grand, plus la puissance du test est grande pour α fixé. Cette observation est en accord avec la formule théorique de π indiquée en annexe 8.3. En effet, si a_1 augmente, son inverse diminue. Par croissance de la fonction de répartition, $G_{2N}(\frac{a_0}{a_1}G_{2N}^{-1}(1-\alpha))$ diminue également, et donc, $\pi = 1 - G_{2N}(\frac{a_0}{a_1}G_{2N}^{-1}(1-\alpha))$ augmente. Plus généralement, plus l'écart entre a_0 et a_1 augmente, meilleur est la performance du test.

4.2 Puissance estimée du test

Afin d'obtenir une puissance estimée $\hat{\pi}$ du test à partir de la fonction *generer*, on crée une fonction $pi\_estimee(a_0, a_1, L, K, p)$ qui prend pour paramètres d'entrée les valeurs de a_0 et a_1 les vitesses de vent relatives aux hypothèses respectives H_0 et H_1, et L le nombre de degrés de liberté de la loi du chi2 utilisée dans la fonction, comme pour la puissance théorique. De plus, le nombre K de réalisations et p le paramètre de la loi de Weibull sont des paramètres d'entrée de la fonction. La fonction renvoie ensuite la puissance estimée du test pour des valeurs de α comprises dans $\{0,01\ ;\ 0,02\ ;\ ...\ ;\ 0,99\}$.

De la même manière que pour la puissance théorique du test, on peut tracer les courbes COR associées à ce test statistique. L'intérêt est de comparer les courbes COR obtenues des deux façons. On les trace donc sur le même graphique avec les paramètres suivants : $a_0 = 0,9$, $a_1 = 1,5$, $N = 20$ et $K = 1000$ ou $K = 50000$.

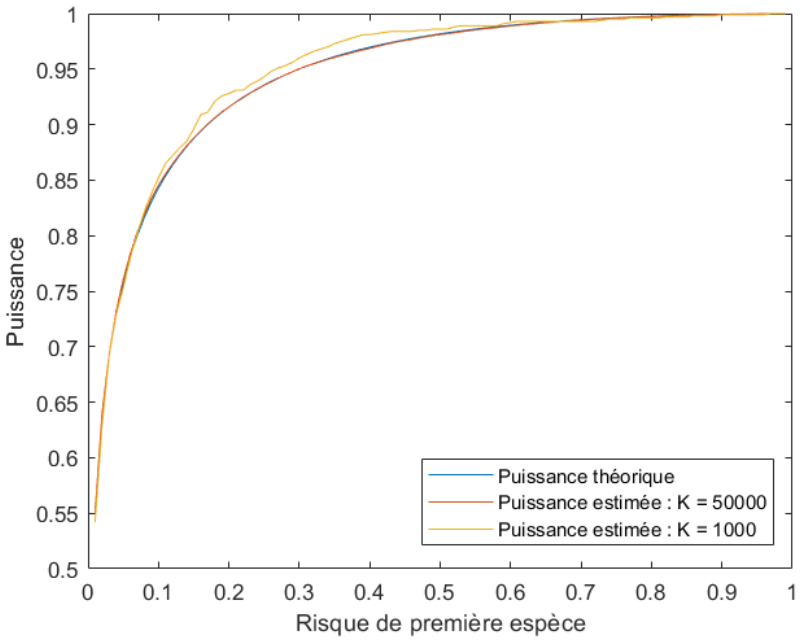


FIGURE 4.3 – Comparaison des courbes COR théorique et estimées

Pour $K = 1000$, la courbe COR estimée n'est pas lisse et s'éloigne de la courbe théorique. Il manque des données pour que la puissance calculée soit fiable. Pour $K = 50000$, la courbe concorde très bien avec la théorie. On remarque bien que plus il y a de données, plus on se rapproche de la théorie.

Partie 5

Analyse d'un fichier de données

5.1 Fichier de données du sujet

L'objectif de cette partie est d'analyser le fichier de données *wind.mat* contenant des mesures de vitesse de vent. On représente ces mesures dans l'histogramme 5.1 suivant :

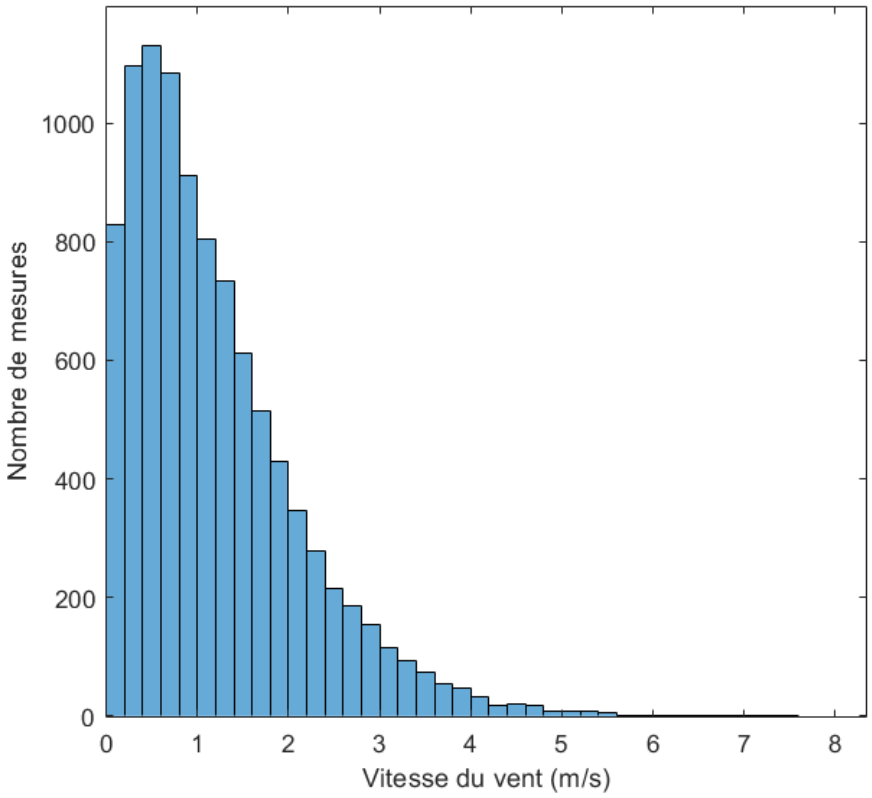


FIGURE 5.1 – Histogramme des vitesses de vent mesurées

On souhaite savoir si modéliser la vitesse du vent par une loi de de Weibull $\mathcal{W}(\hat{\theta}, \hat{p})$ est cohérent à l'aide d'un test de Kolmogorov.

On utilise la fonction Matlab *wblfit* afin de déterminer les paramètres de la distribution de Weibull associés à ce jeu de données. On obtient alors $\hat{\theta} = 1.29$ et $\hat{p} = 1.29$.

On souhaite ensuite représenter sur un même graphique la fonction de répartition théorique d'une loi de Weibull $\mathcal{W}(\hat{\theta}, \hat{p})$ et la fonction de répartition empirique associée aux données de vitesses de vent contenues dans le vecteur *test*. Pour la fonction de répartition théorique, on utilise la formule donnée par les tables : $F(x; \theta, p) = (1 - exp(-(\frac{x}{\theta})^p))\mathbb{I}_{\mathbb{R}^+}(x)$. Pour la fonction de répartition empirique, la valeur en chaque point correspond au nombre d'occurrences de vent inférieures ou égales à cette valeur divisé par le nombre de mesures. On effectue ce calcul pour un échantillon de 100 valeurs, et sur toutes les données du fichier, soit 9 839 valeurs. On obtient le graphique 5.2 suivant.

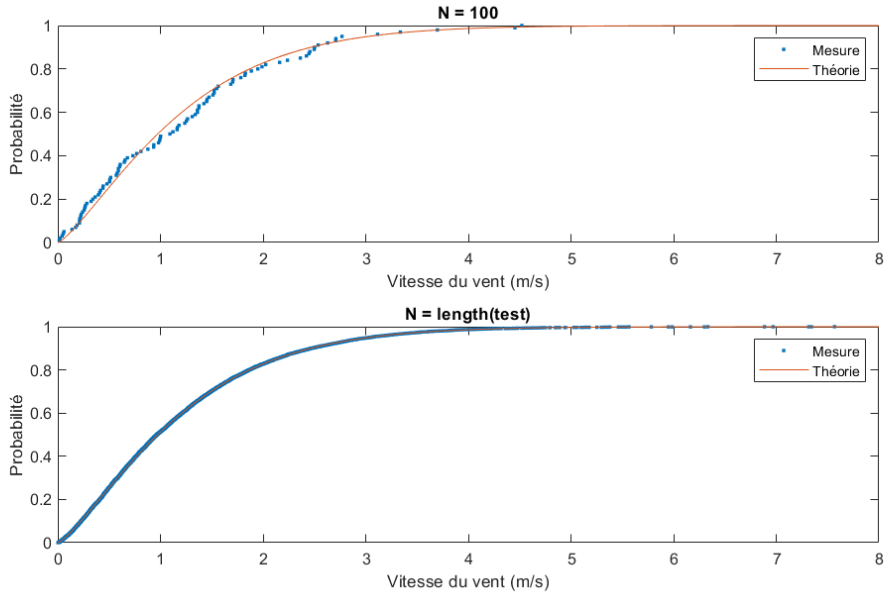


FIGURE 5.2 – Comparaison des fonctions de répartition théorique et empirique

On remarque que pour l'ensemble des données du vecteur test utilisées, la fonction de répartition empirique se superpose à la fonction de répartition théorique d'une loi de Weibull. Cependant, si on ne considère que $K = 100$ données, la précision du calcul de la fonction de répartition empirique est grandement altérée. Il faut donc avoir beaucoup de données pour retrouver un résultat qui colle avec la théorie.

Afin d'avoir un critère plus adapté pour déterminer si notre échantillon de données suit une loi de Weibull $\mathcal{W}(\hat{\theta}, \hat{p})$, on peut utiliser le test de Kolmogorov. Pour cela, on peut utiliser la fonction *kstest* de Matlab ou bien coder ce test en créant une fonction *ecarts* qui calcule les écarts E_i^+ et E_i^- définis par :

$$E_i^+ = \left| \frac{i}{N} - F_W(y_i; \hat{\theta}, \hat{p}) \right|$$
$$E_i^- = \left| \frac{i-1}{N} - F_W(y_i; \hat{\theta}, \hat{p}) \right|$$

avec N le nombre de données du vecteur *test*
et $F_W(y_i; \hat{\theta}, \hat{p})$ la fonction de répartition d'une loi de Weibull $\mathcal{W}(\hat{\theta}, \hat{p})$.

La fonction *ecarts* renvoie donc deux réels notés $D1$ et $D2$ correspondants aux valeurs de ces deux écarts. Elle prend pour paramètres d'entrée les données du test trié *test_tri* et le rang du calcul i. Il faut ensuite choisir le maximum de toutes les valeurs calculées pour obtenir la statistique de test D. On trouve $D = 0.005$.

On calcule également le seuil de test : $\lambda_\alpha = \frac{1}{\sqrt{N}} K^{-1}(1-\alpha)$[2]. On obtient $\lambda_{0.05} = 0.0137$. La valeur de D est bien inférieure à la valeur du seuil de test donc les données sont distribuées selon une loi de Weibull $\mathcal{W}(\hat{\theta}, \hat{p})$.

Avec la fonction *kstest* de Matlab appliquée au vecteur *test_tri*, on retrouve la même valeur de statistique de test $D = 0,005$, le même seuil, et on reçoit le booléen 0 nous indiquant que les données sont bien distribuées selon une loi de Weibull $\mathcal{W}(\hat{\theta}, \hat{p})$.

5.2 Mesures de vent à Toulouse-Blagnac

Par curiosité, nous avons voulu effectuer l'étude précédente sur les mesures de vent de l'année 2022 à Toulouse-Blagnac. Attention, cette étude porte sur les mesures de rafales de vent et non le vent moyen, induisant peut-être la nécessité de modéliser ces valeurs de manière différente. Ces données sont accessibles en open-data sur la plateforme **Infoclimat**. Après une mise en forme des données à l'aide du programme Python *test_reel.py*, les données sont prêtes à être chargées dans le code Matlab précédent pour analyse.

On trace d'abord l'histogramme des valeurs mesurées pour avoir un premier aperçu de nos données :

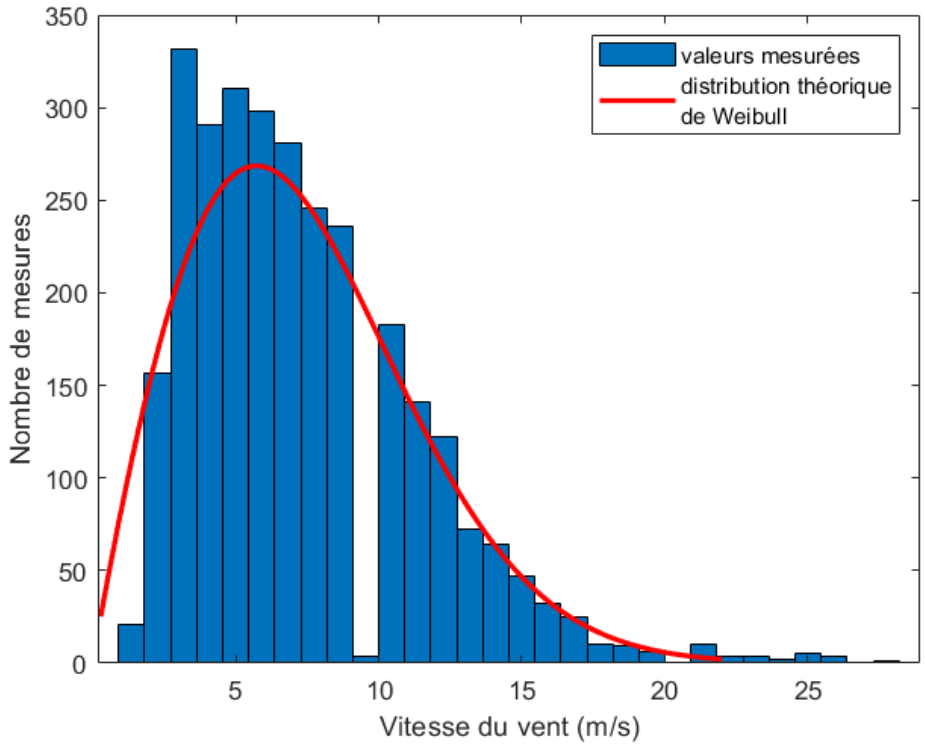


FIGURE 5.3 – Vitesses de vent mesurées à Toulouse-Blagnac

On note sur cette figure une différence notable entre la distribution de Weibull et les valeurs de vent mesurées, bien que la tendance reste correcte. Afin de poursuivre l'étude, la figure 5.4 compare les fonctions de répartition théorique et mesurée :

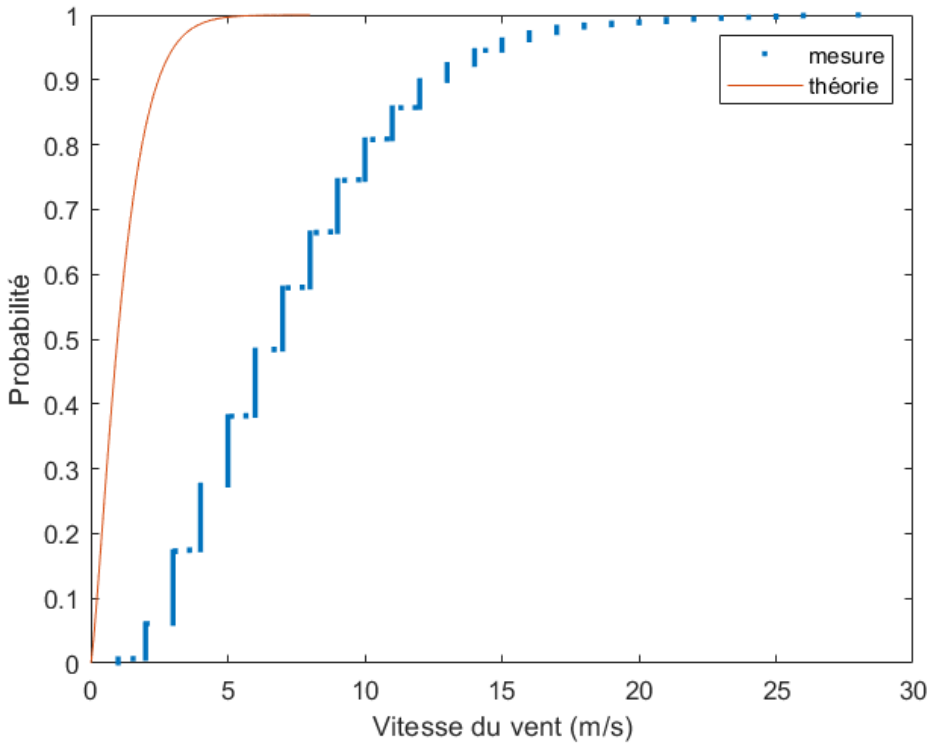


FIGURE 5.4 – Comparaison des fonctions de répartition théorique et empirique

Là encore, la différence est plus importante qu'avec le fichier précédent. La dispersion ne peut pas venir seulement du faible nombre de points, puisque le fichier est composé de 2917 mesures.

Le test de Kolmogorov confirme le fait que les rafales de vent mesurées à Toulouse-Blagnac ne suivent pas une distribution de Weibull. En effet, la fonction *kstest* renvoie le booléen 1, signifiant que ces valeurs ne suivent pas une distribution de Weibull. Cela est confirmé par le fait que $D > \lambda_{0,05}$ puisque $\lambda_{0.05} = 0.0251$ et $D = 0.0707$. Les rafales de vent ne suivent donc pas une distribution de Weibull.

Il existe cependant plusieurs limites à cette conclusion. Premièrement, les données utilisées sont certes nombreuses, mais relativement espacées dans le temps. Elles ne sont donc pas parfaitement représentative de la réalité. Ensuite, dans l'article [1] sur lequel se base cette étude, il n'est pas question des rafales de vent, mais bien du vent moyen. Il est regrettable que les données récupérées ne permettent pas d'exploiter les mesures de vent moyen.

Finalement, ce travail supplémentaire permet de noter qu'il existe une différence importante entre une étude effectuée avec des données choisies pour mener à bien l'étude (cf. partie 5.1) et une étude de cas réel avec des données brutes.

Partie 6

Conclusion

Pour conclure, cette étude nous a permis de développer 3 points.

Premièrement, elle a été l'occasion d'apprendre à mener une étude statistique complète à l'aide de Matlab.

De plus, elle a permis de voir concrètement comment fonctionnent les tests statistiques et ce qu'ils représentent.

Finalement, le contexte nous a permis de voir qu'il n'est pas possible d'effectuer des prévisions météorologiques sur la simple base d'une étude statistique comme précisé dans l'article [1]. Cette étude peut tout de même être utilisée afin de connaître le potentiel d'un espace pour l'implantation d'un parc éolien par exemple. En effet, la distribution des vents moyens est correctement modélisée par une distribution de Weibull à condition de considérer un temps assez long. Il n'a pas l'air d'être de même pour les rafales de vent qui ne suivent pas cette distribution à Toulouse sur l'année 2022.

Au niveau du code, il nous est difficilement possible de porter un regard critique sur nos méthodes de code car il s'agit de notre première expérience avec Matlab. Il est fort probable qu'après plusieurs études menées à l'aide de ce langage, nous serons en mesure d'améliorer la structure de ce code et sa clarté.

Partie 7

Références

[1] M. Martín, L.V. Cremades et J.M. Santabàrbara. « Analysis and modelling of time series of surface wind speed and direction ». In : *Int. J. Climatol.* 19.2 (fév. 1999), p. 197-209. issn : 0899-8418, 1097-0088. url : https://onlinelibrary.wiley.com/doi/10.1002/(SICI)1097-0088(199902)19:2%3C197::AID-JOC360%3E3.0.CO;2-H.

[2] Jean-Yves Tourneret. « Statistique - Slides 1ère année MFEE ». In : ().

Partie 8

Annexes

8.1 Détermination du maximum de vraisemblance

On veut montrer que l'estimateur du maximum de vraisemblance θ, paramètre de la loi de Weibull $\mathcal{W}(\theta, p)$, construit à partir des observations $y_1,\ y_2, ...,\ y_N$ est défini par :

$$\hat{\theta} = \left(\frac{1}{N} \sum_{i=1}^{N} Y_i^p \right)^{1/p}$$

On suppose les Y_i indépendants, alors la vraisemblance vaut :

$$L(y_1, ...,\ y_N|\theta) = \prod_{i=1}^{N} f(y_i; \theta)$$

avec f la densité de probabilité d'une loi de Weibull $\mathcal{W}(\theta, p)$

$$\Rightarrow L(y_1, ...,\ y_N|\theta) = \left(\tfrac{p}{\theta}\right)^N \prod_{i=1}^{N} \left(\tfrac{y_i}{\theta}\right)^{p-1} exp\left[-\left(\tfrac{y_i}{\theta}\right)^p\right] \text{ (pour } y_i > 0)$$

$$\Rightarrow lnL = N\ lnp\ -\ n\ ln\theta\ +\ \sum_{i=1}^{N}(p-1)ln(y_i)\ -\ N(p-1)\ ln\theta\ -\ \sum_{i=1}^{N}\left(\tfrac{y_i}{\theta}\right)^p$$

$$\Rightarrow \tfrac{\partial lnL}{\partial \theta} = -\tfrac{N}{\theta}\ -\ \tfrac{N(p-1)}{\theta}\ +\ p\sum_{i=1}^{N}\left(\tfrac{y_i^p}{\theta^{p-1}}\right)$$

$$\Rightarrow \tfrac{\partial lnL}{\partial \theta} = -\tfrac{Np}{\theta}\ +\ p\sum_{i=1}^{N}\left(\tfrac{y_i^p}{\theta^{p-1}}\right)$$

On cherche le maximum de la fonction lnL :

$$\frac{\partial lnL}{\partial \theta} = 0 \Leftrightarrow \frac{Np}{\theta} = \frac{p}{\theta^{p-1}} \sum_{i=1}^{N} y_i^p$$

$$\Leftrightarrow \theta^p = \frac{1}{N} \sum_{i=1}^{N} y_i^p$$

$$\Leftrightarrow \theta = \left(\frac{1}{N} \sum_{i=1}^{N} y_i^p \right)^{1/p}$$

On obtient alors l'estimateur du maximum de vraisemblance de θ :

$$\hat{\theta}_{MV} = \left(\frac{1}{N} \sum_{i=1}^{N} Y_i^p \right)^{p}$$

Cependant, pour simplifier les calculs, on étudie pour la suite l'estimateur du maximum de vraisemblance de $a = \theta^p$ qui, grâce à l'invariance fonctionnelle, est défini par :

$$\hat{a}_{MV} = \frac{1}{N} \sum_{i=1}^{N} Y_i^p$$

Montrons que cet estimateur est un **estimateur non biaisé** de $a = \theta^p$:

$$E(\hat{a}_{MV}) = \frac{1}{N}\sum_{i=1}^{N} E(Y_i^p)$$

$$\text{Or} \quad E(Y_i^p) = \int_0^{+\infty} x^p \frac{p}{\theta}\left(\frac{x}{\theta}\right)^{p-1} exp\left(-\left(\frac{x}{\theta}\right)^p\right)\ dx$$

$$= \frac{p}{\theta^p}\int_0^{+\infty} x^{2p-1} exp\left(-\left(\frac{x}{\theta}\right)^p\right)\ dx$$

$$\text{On pose} \quad u = \left(\frac{x}{\theta}\right)^p \quad \text{d'où} \quad du = \frac{p}{\theta^p}x^{p-1}$$

$$E(Y_i^p) = \int_0^{+\infty} \theta^p u\ exp\left(-u\right)\ du$$

$$= \theta^p\left(\Big[-u\ exp(-u)\Big]_0^{+\infty} - \int_0^{+\infty} exp(-u)\ du\right) \quad \text{par intégration par parties}$$

$$= \theta^p$$

$$\text{D'où} \quad E(\hat{a}_{MV}) = \frac{N\theta^p}{N} = \theta^p = a$$

On en déduit donc le biais de $\hat{a}_{MV}$:

$$b_n(\hat{a}_{MV}) = E(\hat{a}_{MV}) - a = a - a = 0$$

Cet estimateur est donc **non biaisé**. Montrons maintenant qu'il est **convergent**. Comme les Y_i sont indépendantes, on a :

$$Var(\hat{a}_{MV}) = \frac{1}{N^2}\sum_{i=1}^{N} Var(Y_i^p)$$

$$\text{Or} \quad Var(Y_i^p) = E(Y_i^{2p}) - E(Y_i^p)^2$$

$$\text{Avec} \quad E(Y_i^{2p}) = \int_0^{+\infty} x^{3p-1}\frac{p}{\theta^p} exp\left(-\left(\frac{x}{\theta}\right)^p\right)\ dx$$

$$= \int_0^{+\infty} u^2\theta^{2p} exp(-u)du \quad \text{en posant} \quad u = \left(\frac{x}{\theta}\right)^p$$

$$= \theta^{2p}\int_0^{+\infty} u^2 exp(-u)du$$

Par intégration par parties, on obtient :

$$\begin{aligned} E(Y_i^{2p}) &= \theta^{2p} \left(\Big[-u^2 \ exp(-u) \Big]_0^{+\infty} - \int_0^{+\infty} 2u \ exp(-u) \ du \right) \\ &= 2\theta^{2p} \end{aligned}$$

$$\text{D'où} \quad Var(\hat{a}_{MV}) = \frac{1}{N^2} \ N\theta^{2p} = \frac{\theta^{2p}}{N} = \frac{a^2}{N}$$

Comme $\lim\limits_{N \to +\infty} b_N(\hat{a}_{MV}) = 0$ et $\lim\limits_{N \to +\infty} v_N(\hat{a}_{MV}) = \lim\limits_{N \to +\infty} \frac{a^2}{N} = 0$, cet estimateur est **convergent**.

Montrons maintenant qu'il est **efficace** en montrant qu'il est égal à la borne de Cramér-Rao notée $BRC(\theta^p)$, sachant qu'il est sans biais.

$$BRC(\theta^p) = \frac{1}{-E\left(\frac{\partial^2 ln \ L}{\partial a^2}\right)}$$

$$\text{Et} \quad \frac{\partial^2 ln \ L}{\partial a^2} = \frac{N}{a^2} - \frac{2}{a^3} \sum_{n=1}^{N} y_n^p$$

$$\begin{aligned} \text{D'où} \quad E\left(\frac{\partial^2 ln \ L}{\partial a^2}\right) &= \frac{N}{a^2} - \frac{2}{a^3} \ N \ a \\ &= \frac{N}{a^2} - \frac{2N}{a^2} \\ &= -\frac{N}{a^2} \end{aligned}$$

$$\text{On en déduit donc que} \quad BRC(a) = \frac{a^2}{N} = v_n(\hat{a}_{MV})$$

Finalement, cet estimateur est **efficace**.

8.2 Intervalle de confiance

On cherche à déterminer un intervalle de confiance à 95 % de $\hat{a}_{MV}$.

D'après le théorème central limite, la variable aléatoire $U = \frac{\hat{a}_{MV} - a}{\sqrt{a^2/N}}$ suit une loi normale $\mathcal{N}(0,1)$. Pour trouver les bornes de l'intervalle, on cherche b tel que $P(-b < U < b) = 0,95$, comme représenté sur la figure 8.1.

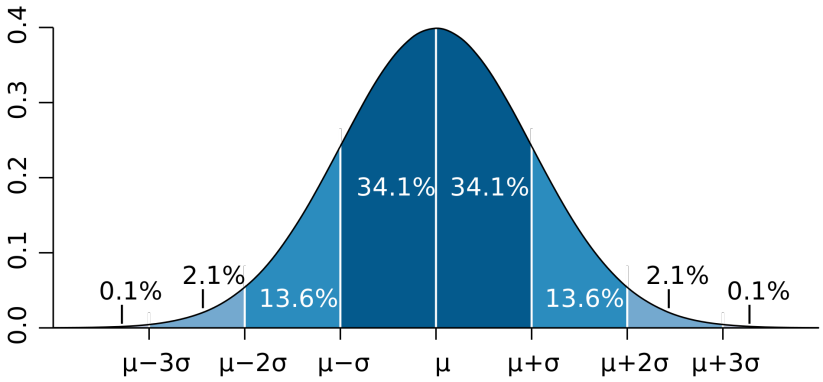


FIGURE 8.1 – Intervalle de confiance d'une loi normale *Source Wikipédia*

Par symétrie de la loi normale, b vérifie $P(U < b) = 0,975$. En utilisant la fonction de répartition inverse de la loi normale $F^{-1}_{\mathcal{N}(0,1)}$, on trouve $b = F^{-1}_{\mathcal{N}(0,1)}(0,975) \simeq 1,96$.

Or, on a : $P(-1,96 < U < 1,96) = P(a(1 - \frac{1,96}{\sqrt{N}}) < \hat{a}_{MV} < a(1 + \frac{1,96}{\sqrt{N}})) = 0,95$.

On en déduit l'intervalle de confiance à 95 % pour $\hat{a}_{MV}$:

$$I = \left[a(1 - \tfrac{1,96}{\sqrt{N}}); a(1 + \tfrac{1,96}{\sqrt{N}})\right]$$

8.3 Étude du test statistique de détection

On souhaite étudier les performances du test statistique qui permet de détecter un vent calme ou un vent fort à partir des données y_1, y_2, ..., y_N. Les hypothèses de ce test sont :

$$H_0 : a = a_0 \qquad H_1 : a = a_1 > a_0$$

On souhaite d'abord montrer que si Y_i suit une loi de Weibull $\mathcal{W}(\theta, p)$ de densité de probabilité f, alors $Z_i = \frac{2}{a}Y_i^p$ suit une loi du chi2 à 2 degrés de liberté. Pour cela, on calcule la densité de probabilité π de Z_i en utilisant le changement de variable bijectif $z = \frac{2}{a}y^p \Leftrightarrow y = (\frac{a}{2}z)^{1/p}$. Tous les y_i sont positifs donc tous les z_i sont également positifs.

Le jacobien de cette transformation est $|\frac{dy}{dz}| = \frac{1}{p}(\frac{a}{2})^{1/p} z^{1-\frac{1}{p}}$.

En utilisant la formule de changement de variable, on obtient :

$$\pi(z) = f((\frac{a}{2}z)^{1/p})|\frac{dy}{dz}|$$
$$\pi(z) = \frac{1}{2}exp(-\frac{z}{2})\mathbb{I}_{\mathbb{R}^+}(z)$$

On reconnaît la densité de probabilité d'une loi du chi2 à 2 degrés de liberté.

Dans un second temps, l'objectif est de trouver la statistique de test associée à ces hypothèses en appliquant le théorème de Neyman-Pearson :

On rejette H_0 si $\dfrac{L(y_1,\ y_2, ...,\ y_N|H_1)}{L(y_1,\ y_2, ...,\ y_N|H_0)} > S_\alpha$

On rejette H_0 si $lnL(y_1,\ y_2, ...,\ y_N|H_1) - lnL(y_1,\ y_2, ...,\ y_N|H_0) > K_\alpha$

La log-vraisemblance est définie par :

$$ln\ L(y_1,\ y_2, ...,\ y_N; a) = Nln(\tfrac{a}{p}) + (p-1)\sum_{i=1}^{N} ln(y_n^p) - \tfrac{1}{a}\sum_{i=1}^{N} y_n^p ln(y_n)$$

D'où la condition devient :

On rejette H_0 si $\quad ln\ L(y_1,\ y_2, ...,\ y_N|H_1) - ln\ L(y_1,\ y_2, ...,\ y_N|H_0) > K_\alpha$

On rejette H_0 si $\quad Nln(a_0) - Nln(a_1) + \sum_{i=1}^{N} y_i^p(\frac{1}{a_0} - \frac{1}{a_1}) > K_\alpha$

On rejette H_0 si $\quad (\frac{1}{a_0} - \frac{1}{a_1})\sum_{i=1}^{N} y_i^p > U_\alpha$

On rejette H_0 si $\quad \sum_{i=1}^{N} y_i^p > \lambda_\alpha$ car $a_1 > a_0$

On en déduit d'après le théorème de Neyman-Pearson que la statistique de test associée à ces hypothèses est :

$$T(Y) = \sum_{i=1}^{N} Y_i^p$$

Par définition, la région critique du test est l'ensemble $\left\{(y_1,\ y_2, ...,\ y_N)|\ \sum_{i=1}^{N} y_i^p > \lambda_\alpha\right\}$.

Pour déterminer le seuil de décision λ_α, on fixe la probabilité de fausse alarme α. En notant G_{2N} la fonction de répartition d'une loi χ^2_{2N}, cette probabilité vaut :

$\alpha = P(\text{Rejet de } H_0|H_0 \text{ vraie})$
$\alpha = P(T > \lambda_\alpha|a = a_0)$
$\alpha = 1 - P(T \leq \lambda_\alpha|a = a_0)$
$\alpha = 1 - P(\sum_{i=1}^{N} \frac{2}{a} Y_i^p \leq \frac{2}{a}\lambda_\alpha|a = a_0)$

D'après ce qui précède, les Y_i suivent une loi du chi2 χ^2_{2N}, d'où :

$\alpha = 1 - G_{2N}(\frac{2}{a_0}\lambda_\alpha)$

En inversant la dernière expression, on trouve un seuil de :

$$\boxed{\lambda_\alpha = \tfrac{a_0}{2} G_{2N}^{-1}(1-\alpha)}$$

On peut calculer de la même manière que α, le risque de seconde espèce β. Son expression est :

$$\beta = G_{2N}(\frac{2}{a_1}G_{2N}^{-1}(1-\alpha)$$

On en déduit la puissance théorique du test en fonction de α :

$$\boxed{\pi = 1 - G_{2N}(\frac{a_0}{a_1}G_{2N}^{-1}(1-\alpha))}$$

8.4 Codes

8.4.1 Programme principal

```
clear all
close all

global p

p=1.5;

%% Partie 1
N1B = 10000;
K1B = 1;
theta = 3.3;
Y1 = generer(N1B,K1B,theta);

figure
histfit(Y1, sqrt(N1B),"weibull");

moyenne1b = mean(Y1);
variance1b = var(Y1);

[moyennewbth , variancewbth] = param(theta);

N1C = 1000;
K1C = 500;

Y2 = generer(N1C,K1C,theta);

figure
histfit(Y2(:,1));

L_moy = [];
L_var = [];

moy_th = theta * gamma(1+(1/p));
var_th = theta * theta * gamma(1+(2/p)) - moy_th*moy_th;

for j = 1:K1C
    L_moy(end+1) = mean(Y2(:,j));
    L_var(end+1) = var(Y2(:,j));
end
```

```
40
41 figure
42 subplot(2,1,1), plot([1:K1C],L_moy, [1:K1C],repelem(↩
      moyennewbth,K1C));
43 subplot(2,1,2), plot([1:K1C],L_var, [1:K1C],repelem(↩
      variancewbth,K1C));
44
45 %% Partie 2
46
47 amv = estimateur_mv(Y2,N1C,K1C);
48
49 moy_amv = mean(amv);
50 var_amv = var(amv);
51
52 moy_amv_th = theta^p;
53 var_amv_th = ((theta^p)^2)/N1C;
54
55 figure
56 plot([1:K1C],amv,'.', [1:K1C], repelem(moy_amv,K1C), [1:K1C],↩
       repelem(moy_amv*(1+(1.96/sqrt(N1C))),K1C), "k", [1:K1C], ↩
      repelem(moy_amv*(1-(1.96/sqrt(N1C))),K1C), "k");
57
58
59 %% Partie 3
60
61 % N variable
62 a0 = 0.9;
63 a1 = 1.5;
64 N=[10 20 50];
65
66 % Dans une boucle
67 % figure
68 % for i = 1:length(N)
69 %     L = 2*N(i);
70 %     pi = pi_theorique(a0,a1,L);
71 %     subplot(length(N),1,i), plot([0.01:0.01:0.99],pi)
72 % end
73
74 % Un par un pour effectuer un seul plot
75 figure
76 pi10 = pi_theorique(a0,a1,2*N(1));
77 pi20 = pi_theorique(a0,a1,2*N(2));
78 pi50 = pi_theorique(a0,a1,2*N(3));
79
80 plot([0.01:0.01:0.99],pi10,[0.01:0.01:0.99],pi20↩
      ,[0.01:0.01:0.99],pi50)
81
82 %a1 variable
83 a0 = 0.9;
84 a1 = [1.2 1.5 2];
85 N=20;
```

```
86 L = 2*N;
87
88 % Dans une boucle
89 % figure
90 % for i = 1:length(a1)
91 %     pi = pi_theorique(a0,a1(i),L);
92 %     subplot(length(a1),1,i), plot([0.01:0.01:0.99],pi)
93 % end
94
95 % Un par un pour effectuer un seul plot
96 figure
97 pi12 = pi_theorique(a0,a1(1),L);
98 pi15 = pi_theorique(a0,a1(2),L);
99 pi2 = pi_theorique(a0,a1(3),L);
100
101 plot([0.01:0.01:0.99],pi12,[0.01:0.01:0.99],pi15↩
    ,[0.01:0.01:0.99],pi2)
102
103 % Simulation
104
105 a0 = 0.9;
106 a1 = 1.5;
107 N = 20;
108 K = 1000;
109
110 pi_est = pi_estimee(a0,a1,2*N,K);
111
112 figure
113 plot([0.01:0.01:0.99],pi_est);
114
115 % Comparaison courbes COR
116
117 a0 = 0.9;
118 a1 = 1.5;
119 N = 20;
120 K = [50000 1000];
121 alpha = [0.01:0.01:0.99];
122
123 pi_th = pi_theorique(a0,a1,2*N);
124 pi_est1 = pi_estimee(a0,a1,2*N,K(1));
125 pi_est2 = pi_estimee(a0,a1,2*N,K(2));
126
127 figure
128 plot(alpha,pi_th,alpha,pi_est1,alpha,pi_est2)
129
130 %% Partie 4
131
132 load('wind.mat')
133
134 figure
135 histogram(test)
```

```
136
137 param_est = wblfit(test);
138
139 vent = [0:0.01:8];
140 f_repart_th = [];
141
142 for i = 1:length(vent)
143     f_repart_th(end+1) = 1 - exp(-(vent(i)/param_est(1))^(↩
        param_est(2)));
144 end
145
146 % Pour 100 données
147
148 test_tri100 = sort(test(1:100));
149
150 f_repart_mes100 = [];
151
152 for i = 1:length(test_tri100)
153     f_repart_mes100(end+1) = i/length(test_tri100);
154 end
155
156 % Pour toutes les données
157
158 test_tri = sort(test);
159
160 f_repart_mes = [];
161
162 for i = 1:length(test_tri)
163     f_repart_mes(end+1) = i/length(test);
164 end
165
166 figure
167 subplot(2,1,1),plot(test_tri100,f_repart_mes100,".", vent, ↩
    f_repart_th)
168 subplot(2,1,2),plot(test_tri,f_repart_mes,".", vent, ↩
    f_repart_th)
169
170 % Test de Kolmogorov
171
172 % Calcul des écarts
173 L_Eplus = [];
174 L_Emoins = [];
175
176 for i = 1:length(test_tri)
177     [D1,D2] = ecarts(test_tri,i);
178     L_Eplus(end+1) = D1;
179     L_Emoins(end+1) = D2;
180 end
181
182 D = max(max(L_Emoins),max(L_Eplus))
183
```

```
184 % Test de Kolmogorov-Smirnov avec la fonction kstest
185
186 [h_t, p_t, ksstat_t, cv_t] = kstest(test_tri, 'CDF', [↩
      test_tri, wblcdf(test_tri, param_est(1), param_est(2))])
187
188 % Calcul du seuil lambda_alpha
189 lambda_alpha = kolminv(1-0.05)/sqrt(length(test_tri))
190
191 %% Analyse des mesures de Toulouse_Blagnac
192
193 load('mesure_Toulouse.mat')
194
195 figure
196 histfit(mesure, 30,"weibull");
197
198 figure
199 histogram(mesure)
200
201 param_est_Toul = wblfit(mesure)
202
203 mesure_tri = sort(mesure).';
204
205 f_repart_Toul = [];
206
207 for i = 1:length(mesure_tri)
208     f_repart_Toul(end+1) = i/length(mesure);
209 end
210
211 figure
212 plot(mesure_tri,f_repart_Toul,".", vent, f_repart_th)
213
214 % Test de Kolmogorov
215
216 % Calcul des écarts
217 L_Eplus_Toul = [];
218 L_Emoins_Toul = [];
219
220 for i = 1:length(mesure_tri)
221     [D1,D2] = ecarts(mesure_tri,i);
222     L_Eplus_Toul(end+1) = D1;
223     L_Emoins_Toul(end+1) = D2;
224 end
225
226 D = max(max(L_Emoins_Toul),max(L_Eplus_Toul))
227
228 % Test de Kolmogorov-Smirnov avec la fonction kstest
229
230 [h_tou, p_tou, ksstat_tou, cv_tou] = kstest(mesure_tri, 'CDF'↩
      , [mesure_tri, wblcdf(mesure_tri, param_est_Toul(1), ↩
      param_est_Toul(2))])
231
```

```
232 % Calcul du seuil lambda_alpha
233 lambda_alpha = kolminv(1-0.05)/sqrt(length(mesure_tri))
```

8.4.2 Fonctions

```
1  function [Y] = generer(N, K, theta)
2      global p
3      X = rand(N,K);      % Matrice de taille NxK avec des ↩
          nombres compris entre  0 et 1
4      Y = zeros(N,K);     % Création d'une matrice remplie de 0↩
          dont les valeurs seront à modifier
5      for i=1:N
6          for j=1:K
7              Y(i,j) = theta * log(1/(1-X(i,j)))^(1/p);  % ↩
                  Utilisation de la fonction de repartition ↩
                  inverse
8          end
9      end
10 end
```

```
1 function [moyth, varth] = param(theta)
2 %Calcul de la moyenne et variance théoriques d'une loi de ↩
     Weibull
3     global p
4     moyth = theta * gamma(1+1/p);
5     varth = theta * theta * gamma(1 + 2/p) - moyth * moyth;
6 end
```

```
1 function [alpha_est] = estimateur_mv(Y, N, K)
2     global p
3     alpha_est=sum(Y.^p,1)/N;
4 end
```

```
1 function [pi] = pi_theorique(a0,a1,L)
2     L_alpha = [0.01:0.01:0.99];
3     pi = repelem(0,length(L_alpha));
4     for i = 1:length(L_alpha)
5         lambda = a0 * chi2inv(1 - L_alpha(i),L)/2;
6         pi(i)=1-chi2cdf(2*lambda/a1,L);
7     end
8 end
```

```
1 function [pi_chap] = pi_estimee(a0,a1,L,K)
2     global p
3
4     % Générer le signal
5     theta=a1^(1/p);
6     Y=generer(L/2, K, theta);
```

```
7
8       L_alpha = [0.01:0.01:0.99];
9
10      % Statistique de Test
11      T=sum(Y.^p);
12
13      % Calcul du seuil
14      lambda = a0 * chi2inv(1 - L_alpha,L)/2;
15
16      for i =1:length(L_alpha)
17          pi_chap(i) = length(find(T>lambda(i)))/length(T);
18      end
19  end
```

```
1  function [Eplus,Emoins] = ecarts(test,i)
2      N = length(test);
3      param_est = wblfit(test);
4      Eplus = abs(i/N-wblcdf(test(i),param_est(1),param_est(2))↩
           );
5      Emoins = abs((i-1)/N-wblcdf(test(i),param_est(1),↩
           param_est(2)));
6  end
```

8.4.3 Code Python

```
1  import numpy as np
2  import csv
3  from scipy.io import savemat
4
5  # Recupérer les données du csv en séparant les colonnes par ↩
       des ;
6
7  with open('export_infoclimat.csv', 'r') as f:
8      reader = csv.reader(f, delimiter=';')
9      data = list(reader)
10
11 mesure = []
12 for k in range(6,len(data)):
13     mesure.append(float(data[k][8]))
14
15 # Convertir les valeurs en m/s
16
17 mesure = np.array(mesure)
18 mesure = mesure/3.6
19
20 # Enregistrer la liste dans un fichier .mat
21 savemat('mesure_Toulouse.mat', {'mesure': mesure})
```