



# Document de prise en main

## Banc de calibration de la centrale inertielle d'un smartphone

PFE24-T-325

**Objectifs du document** : Dans ce document, nous détaillerons la prise en main de notre projet, avec la mise en place de l'interface web.  
Cela comprend l'installation de l'environnement, la structure du code et la navigation dans l'interface.

## Historique des révisions

Name	Date	Changes
FERRE Kenta	10/02/2025	Rédaction du dossier
DEPELLEY Nicolas	15/02/2025	Mise en page du document

## I- Mise en place de l'environnement

Le projet a été développé sur VSCode et l'IDE Arduino. VSCode permet de cumuler en un seul environnement le code Python, HTML, CSS et JavaScript. La liaison entre ces codes est facilitée par la création d'un environnement virtuel « venv » géré par la bibliothèque « flask ».

L'IDE Arduino est liée à VSCode via le port COM série sur lequel sera branchée la carte Arduino. Une bibliothèque est nécessaire pour créer cette communication entre le code Python et Arduino.

### 1. Installation des librairies

Avant de faire quoi que ce soit, il est obligatoire de taper la commande suivante dans le terminal du dossier du projet « **python -m venv venv** ». Cela permettra de mettre en place la structure de l'environnement virtuel « venv » pour le bon fonctionnement du projet.

Contrairement au code Arduino, le code Python nécessite de nombreuses dépendances afin de faire fonctionner les diverses fonctions qui constituent le code. Afin de faciliter l'installation des différentes bibliothèques, un fichier texte nommé « requirements.txt » est présent dans le dossier de travail VSCode. Il permet de télécharger l'ensemble des bibliothèques d'un seul coup. Pour cela, il suffit de saisir la commande suivante dans le terminal : « **pip install -r requirements.txt** ».

Pour le code HTML/JavaScript, les bibliothèques sont présentes sous la forme de liens qui renvoient vers des bibliothèques en ligne, sans nécessiter aucun téléchargement. Il est toutefois possible de télécharger les bibliothèques afin de pouvoir exécuter correctement le script même sans connexion.

### 2. Application complémentaire

Physics ToolBox est une application permettant d'obtenir les données brutes de différents capteurs (notamment d'un gyroscope) d'un téléphone. Elle permet d'enregistrer les données mesurées dans un fichier Excel après utilisation. C'est ainsi que nous récupérerons les valeurs de notre gyroscope ou de notre accéléromètre.

### 3. Consignes pour l'exécution du code

Il est obligatoire d'avoir le moniteur série de l'IDE Arduino fermé avant d'exécuter le code Python. Si cela n'est pas fait, le code refusera de fonctionner, car le moniteur série sera trop occupée pour recevoir d'autres informations et l'information envoyé par Python ne sera pas reçue.

Pour exécuter le code, il suffit de lancer le serveur Flask via la commande « **flask run** » dans le terminal.

## II- Structure du Code Source

### 1. Structure du dossier projet

Les différents fichiers sont organisés de façon spécifique et automatique grâce à l'initialisation de l'environnement virtuel et du serveur Flask. Le dossier principal du projet devrait comporter les dossiers suivants :










	__pycache__	04/02/2025 12:03	Dossier de fichiers	
	static	03/02/2025 11:57	Dossier de fichiers	
	templates	30/01/2025 17:30	Dossier de fichiers	
	venv	30/01/2025 18:31	Dossier de fichiers	
	app	03/02/2025 16:51	Fichier source Python	14 Ko
	bode	03/02/2025 15:03	Fichier source Python	3 Ko
	correctif	04/02/2025 11:46	Fichier source Python	8 Ko
	fonction	30/01/2025 19:25	Fichier source Python	9 Ko
	key_data	03/02/2025 16:43	Fichier source Python	4 Ko

Figure 1 : Dossiers composant le projet

Dans le même dossier, on trouvera l'ensemble des fichiers Python constituant le projet. Le but des fonctions Python est de mettre en place la communication entre les autres aspects et langages du projet mais également de gérer les différents fichiers de données ainsi que leur traitement.

Le fichier principal du projet doit s'appeler « app.py ». Ce fichier sera le premier lu lors de l'exécution du code. Dans ce fichier, nous trouverons diverses fonctions, chacune reliée par une route qui permet de faire la liaison entre le langage Python et un autre langage informatique. C'est également dans ce fichier que doit se trouver avant toute fonction la création de l'objet Flask afin de mettre en place le serveur.

```
1  ## Déclaration des bibliothèques
2  from flask import Flask,render_template,request,jsonify
3  import time, os
4  import serial
5  import pandas as pd
6
7  ## Import des fonctions
8  from fonction import rotation, epuration, stat, calcul_stat, track_path, conversion, load_csv_auto
9  from correctif import calibrate_gyro
10 from key_data import calibrate_gyroscope
11 from bode import save_bode_data_to_json, load_data
12
13 ## Initialisation de l'objet Flask
14 app = Flask(__name__)
15
16 ## Variables
17 last_file_processed = None
18
19 ## Initialisation des différentes pages du site internet
20 @app.route("/") ## Accès à la Home Page
21 def home():
22     return render_template("Site.html")
23
24 @app.route("/angle", methods=['POST','GET']) ## Accès discret à la fonction angle
25 def angle():
```

Figure 2 : Extrait du code de la fonction principale

On peut voir devant chaque fonction l'application d'une route pour le transfert des données. La route reprend l'objet Flask et exécutera toujours la fonction liée à la route par défaut, qui est ici « / ». La fonction « render\_template() » permet d'afficher la page HTML demandée.

Dans le dossier « venv » se trouvent l'ensemble des bibliothèques et d'autres fichiers permettant le bon fonctionnement de l'environnement virtuel.

Le dossier « templates » contient quant à lui l'ensemble des fichiers HTML. Enfin, le dossier « static », contient les fichiers CSS et JavaScript ainsi que d'autres documents tels que nos fichiers de données, des images, etc.

## 2. Structure des scripts

Le code suit, peu importe le langage la même logique rédactionnelle. Les bibliothèques sont déclarées dans un premier temps, pour ensuite appeler les fonctions utiles des autres fichiers. Les objets sont ensuite mis en place avant de déclarer les variables globales puis les fonctions.

Le code en langage JavaScript est scindé en deux parties. La partie de détection des actions se trouve dans le fichier du code HTML correspondant et le fichier Javascript contient dans un ordre similaire les fonctions de traitement des requêtes ainsi que celles d'envoi et de réception des données vers le code de langage Python. Des lignes de commentaires séparent les différentes fonctions secondaires (qui ne sont pas liées directement à l'envoi et à la réception des données) afin de rendre la lecture du fichier plus aisée.

### III- Navigation dans l'interface

Cette partie vient en complément de la page que vous pourrez trouver directement dans l'interface web. Elle vient détailler chaque onglet de l'interface.

#### 1. Page « Tests »

Le code permet à ce jour de faire tourner un moteur pas-à-pas à différentes vitesses, mais également à une rotation choisie via la page « test ». Les boutons envoient chacun une information différente qui va influencer la commande que l'IDE d'Arduino va transmettre à la carte.

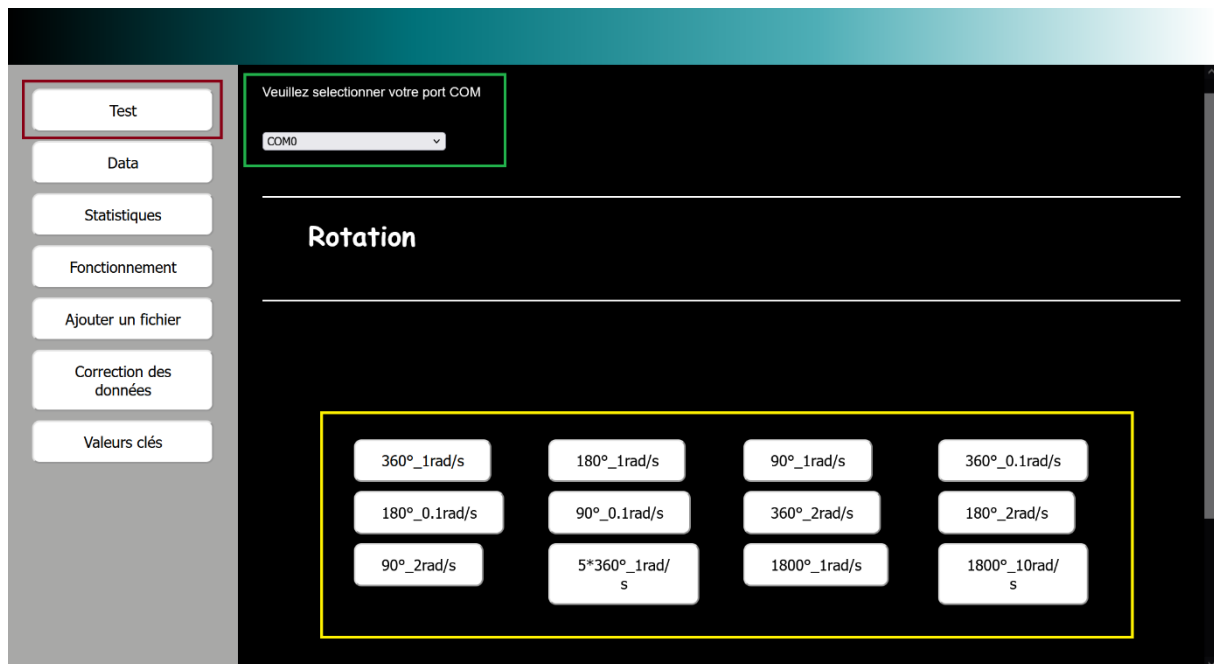


Figure 3 : Page de tests

#### 2. Page « Data »

Dans la page « Data », on retrouve différents menus déroulants qui dans l'ordre prennent : le nom du téléphone, l'exercice effectué, la vitesse, la distance parcourue, le fichier voulu. Tout cela permet de parcourir la base de données en suivant la nomenclature des fichiers afin d'en afficher les données brutes ou sous forme d'un graphe tout en apportant des informations complémentaires aux données.

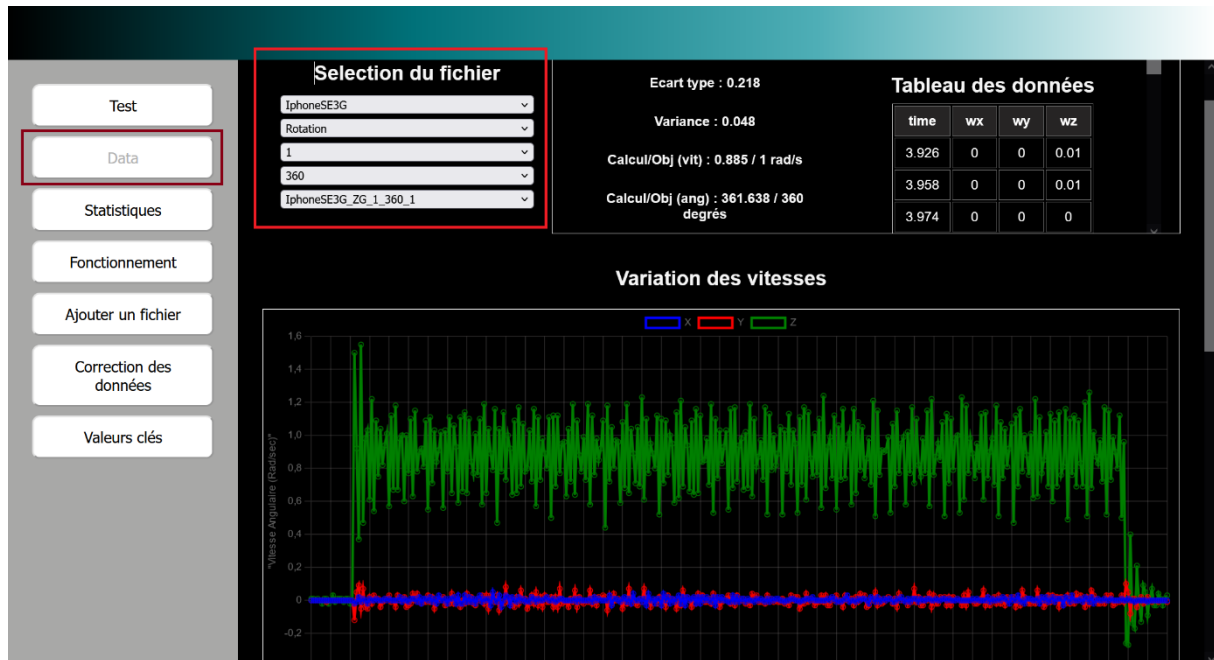


Figure 4 : Page d'affichage des données pour un test

### 3. Page « Statistiques »

La page « Statistiques » est complémentaire à la page précédente en affichant cette fois-ci seulement les informations complémentaires, mais pour l'ensemble des tests effectués sur le téléphone choisi. Si un téléphone contient différents fichiers pour un test identique, alors la moyenne des informations sera affichée.

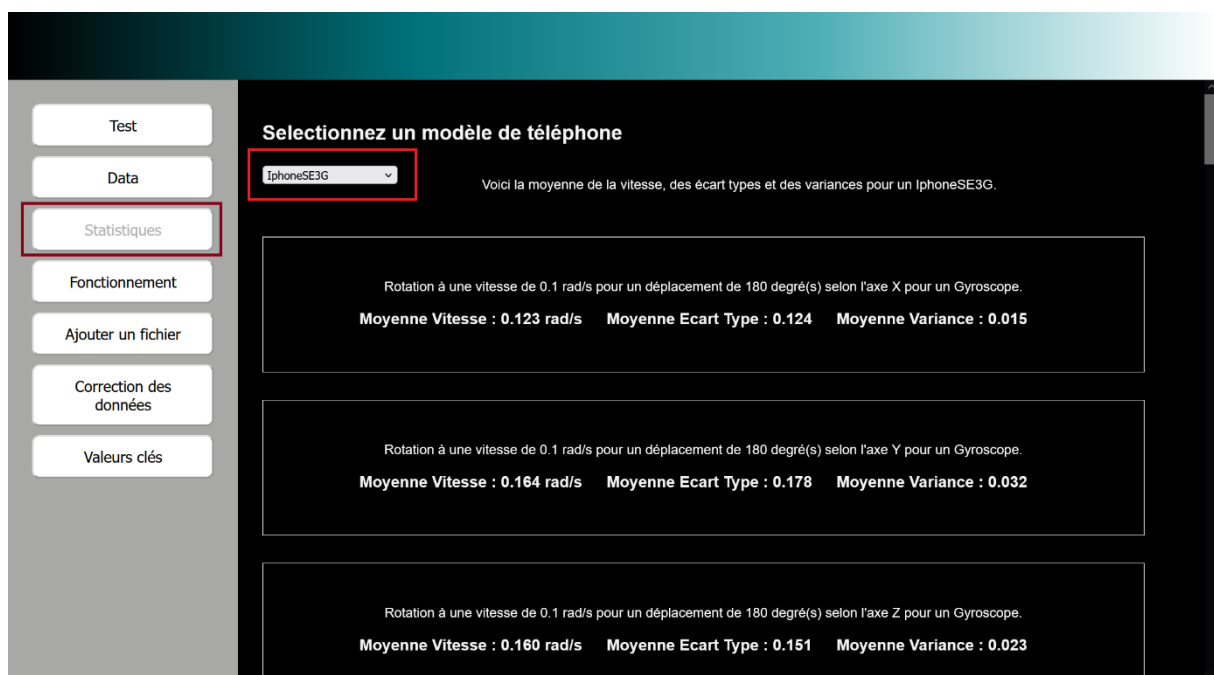


Figure 5 : Page d'affichage des données pour un téléphone

#### 4. Page « Fonctionnement »

La page « Fonctionnement » affiche l'ensemble des consignes d'utilisation et des prérequis pour la bonne exécution du code.

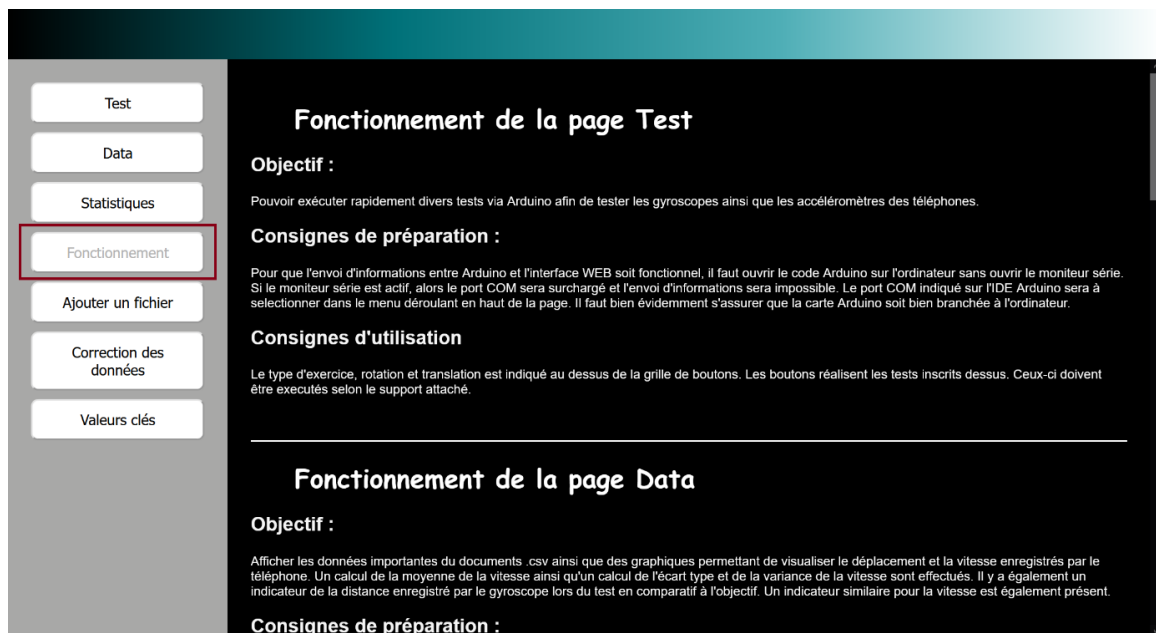


Figure 6 : Page indiquant le fonctionnement du site

#### 5. Page « Ajout d'un fichier »

L'avant-dernière page qui sera présentée (« Correction des données » n'étant pas fonctionnelle), « Ajouter un fichier » permet comme son nom l'indique, de mettre de façon simple un nouveau fichier dans la base de données. Si le test est nouveau alors l'emplacement pour ce nouveau test sera automatiquement créé. Plusieurs fichiers peuvent être mis en même temps mais il faut cependant faire attention notamment au type d'exercice afin de ne pas déranger la classification de la base de données.

**Attention !!** La nomenclature du fichier du fichier doit être la suivante :

***Téléphone\_AxeAppareil\_Vitesse\_Distance\_Numéro***

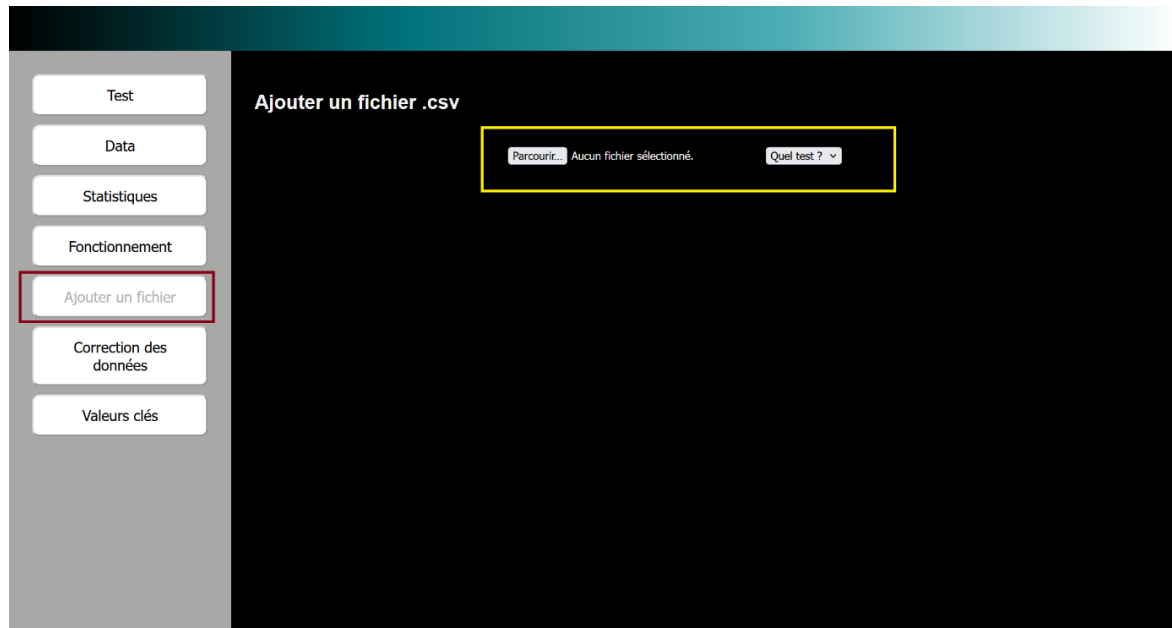


Figure 7 : Page permettant l'ajout de données dans la base de données

## 6. Page « Valeurs clés »

La dernière page de notre site permet d'afficher des données de correction qui peuvent être apportés aux données mais également une démonstration de leur application sur les données avec un comparatif des résultats. Un diagramme de Bode peut également être affiché afin de donner plus d'informations sur le signal mesuré par le gyroscope, contrairement aux contraintes qu'il a réellement reçues.

X	Y	Z
0.000024844720496894407	-0.0003105590062111801	0.0002484472049689441

X	Y	Z
1	1	1.242994458185917

X	Y	Z
1.4732556964439438	1.4732556964439438	0
5.48557939389778	5.48557939389778	0
0	0	1

Figure 8 : Affichage des facteurs de correction



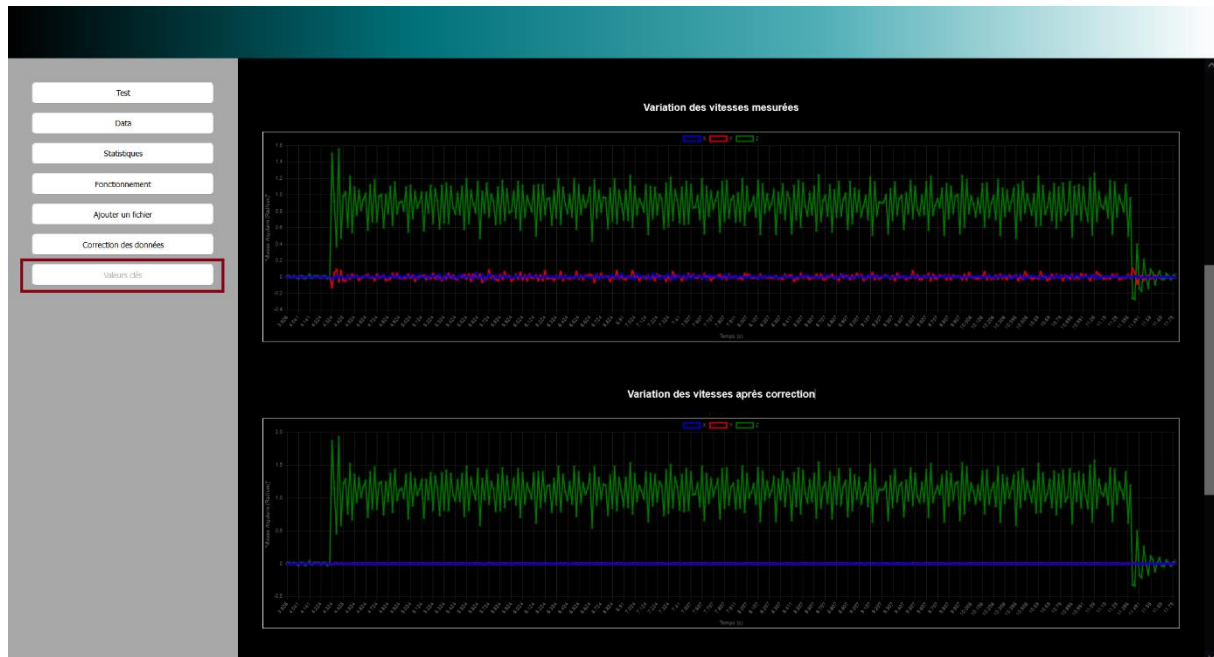


Figure 9 : Courbes des données pré et post traitement

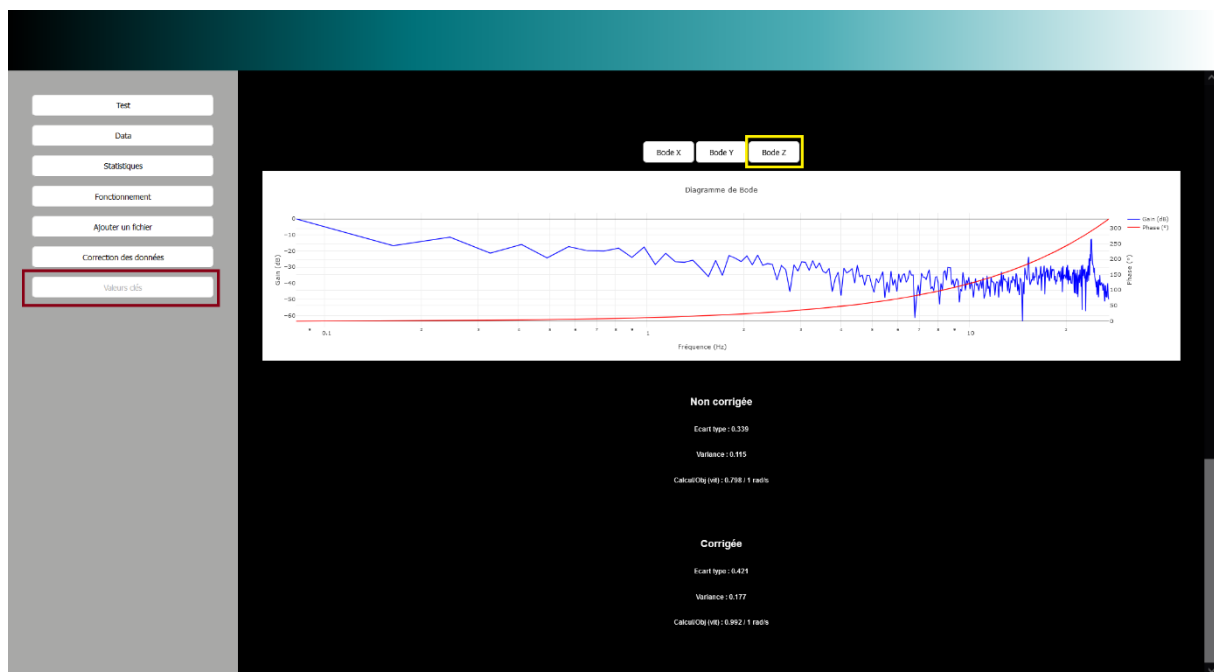


Figure 10 : Diagramme de Bode selon l'axe Z et infos complémentaires

## 7. Page à développer « Correction des données »

La page « Correction des données » serait à compléter en gérant l'affichage des données post-traitement. Actuellement, le code permet de filtrer les données de deux manières distinctes afin de pouvoir effectuer une comparaison.

Une autre fonctionnalité serait de pouvoir recevoir en continu des données et donc permettre de mesurer le biais en temps réel afin d'améliorer la correction des données.

De la même manière, pouvoir obtenir une rotation sur les trois axes en simultané serait intéressant et permettrait une correction des données encore accrue en obtenant des matrices de correction et des facteurs d'échelle sur tous les axes à la fois.

## IV- Reprise du projet

### 1. Bonnes Pratiques de Développement

Pour assurer un développement efficace dans la lignée de ce qui a été fait, il est recommandé de suivre les mêmes pratiques indiquées précédemment dans ce document. Néanmoins, libre à vous de reprendre le projet et de le développer à votre manière en changeant des aspects comme la nomenclature des fichiers tests par exemple.

### 2. Résolution de bugs

Si une fonction ne fonctionne pas mais que le message d'erreur n'apparaît pas dans le terminal de VSCode, il peut être possible que l'erreur soit liée à l'HTML ou JavaScript et que celle-ci soit présente dans la console de la page internet (F12). Si aucun bug n'apparaît sur aucune des deux consoles et que le problème est lié à un problème d'affichage il est possible que celui-ci provienne du code HTML ou CSS qui, même s'il rencontre certaines incohérences mineures peut s'exécuter sans relever d'erreurs.

Peu importe le type de bug, il est toujours intéressant de laisser des « `print()` » le long de l'exécution du code afin de localiser le bug ou encore de mettre en place les fonctions « `try()` » et « `catch()` » qui permettent de renvoyer une erreur plus spécifique au problème afin de mieux le comprendre tout en permettant à l'application de ne pas totalement s'arrêter.

### 3. Collaboration

Pour collaborer efficacement, il faut bien découper le projet en diverses fonctions et se mettre d'accord entre les membres participant à cette partie du projet d'une nomenclature logique pour le nom des variables afin de ne pas avoir de confusion. Mettre au point un fichier contenant le nom des variables, leurs types et l'information peut permettre de rendre la fusion des fonctions plus simple et efficace. Il est évidemment recommandé d'utiliser Git par exemple pour permettre de plus facilement gérer le projet au niveau informatique en le partageant tout en conservant les versions précédentes du projet en cas de nécessité.

## V- Conclusion

Nous voudrions vous souhaiter bonne chance dans votre projet. Nous espérons que notre travail vous permettra d'avancer comme vous le souhaitez et vous mènera vers la réussite.