Python pour la physique en MP2I

Graphiques, incertitudes et ajustements

1 Pourquoi utiliser Python en physique ?

En MP2I, Python est utilisé pour :
— tracer des courbes expérimentales ;
— exploiter des mesures ;
— représenter des incertitudes ;
— réaliser des ajustements (régressions) ;
— modéliser des phénomènes physiques.
Les bibliothèques essentielles sont :
— `numpy` : calcul numérique ;
— `matplotlib` : tracé de graphiques.

2 Premier programme Python

Voici un exemple minimal pour tracer une courbe (à réaliser).

```
import numpy as np
import matplotlib.pyplot as plt

x = np.linspace(0,10,100)
y = np.sin(x)

plt.plot(x,y)

plt.xlabel("x")
plt.ylabel("sin(x)")
plt.title("Premier graphique")
plt.grid()

plt.show()
```

Méthode

Toujours suivre cet ordre :

1. importer les bibliothèques ;
2. définir les données ;
3. tracer la courbe ;
4. ajouter titres et labels ;
5. afficher avec `plt.show()`.

3 Notions essentielles de NumPy

La bibliothèque `numpy` permet de manipuler des tableaux.
Exemple :

```
import numpy as np

x = np.linspace(0, 5, 6)
print(x)
```

On obtient une liste de valeurs régulièrement espacées.

Attention

Ne jamais oublier :
- `import numpy as np`
- les dimensions des tableaux doivent être compatibles

Sinon Python renvoie une erreur.

4 Commandes essentielles de Matplotlib

La bibliothèque `matplotlib.pyplot` permet de tracer des graphiques.
On l'importe systématiquement ainsi :

```
import matplotlib.pyplot as plt
```

4.1 Tracer une courbe : plot()

La commande principale est :

```
plt.plot(x, y)
```

Elle relie les points par une courbe.

Méthode

— `plot` = courbe continue (modèle)
— `scatter` = points expérimentaux

4.2 Nuage de points : scatter()

Pour des mesures expérimentales :

```
plt.scatter(x, y)
```

4.3 Ajout des labels

```
plt.xlabel("Temps (s)")
plt.ylabel("Position (m)")
```

4.4 Titre et grille

```
plt.title("Mouvement d'un mobile")
plt.grid()
```

4.5 Légende

Lorsque plusieurs courbes sont tracées :

```
plt.plot(x, y1, label="modele")
plt.plot(x, y2, label="experience")

plt.legend()
```

4.6 Nouvelle figure

```
plt.figure()
```

Permet de créer un nouveau graphique.

4.7 Sauvegarder un graphique

```
plt.savefig("graphique.png")
```

4.8 Résumé des commandes

Commande	Rôle
plot()	courbe continue
scatter()	points expérimentaux
xlabel()	axe x
ylabel()	axe y
title()	titre
grid()	grille
legend()	légende
figure()	nouvelle figure
savefig()	sauvegarde
show()	affichage

Attention

Erreur fréquente :

- oublier `plt.show()` → rien ne s'affiche
- inverser `x` et `y`
- oublier les labels → graphique inexploitable en TP

5 Données expérimentales

En physique, on distingue toujours :
— les **mesures expérimentales** ;
— les **modèles théoriques**.

5.1 Tracer des points expérimentaux

Exemple simple :

```
import numpy as np
import matplotlib.pyplot as plt

x = [1,2,3,4,5]
y = [2.1, 4.0, 6.2, 8.1, 9.9]

plt.scatter(x, y)
plt.xlabel("x")
plt.ylabel("y")
plt.grid()
plt.show()
```

5.2 Relier les points (modèle)

Pour représenter une tendance :

```
plt.plot(x, y)
```

Méthode

— scatter = réalité expérimentale
— plot = modèle ou interpolation

5.3 Superposition expérience + modèle

```
x = np.array([1,2,3,4,5])
y_exp = np.array([2.1, 4.0, 6.2, 8.1, 9.9])

y_modele = 2*x

plt.scatter(x, y_exp, label="experience")
plt.plot(x, y_modele, label="modele")

plt.legend()
plt.grid()
plt.show()
```

6 Incertitudes expérimentales

En MP2I, toute mesure doit être accompagnée d'une incertitude.
On note :

$$x \pm \Delta x$$

6.1 Représentation avec barres d'erreur

La commande essentielle est :

```
plt.errorbar(x, y, yerr=dy, xerr=dx, fmt='o', capsize=4)
```

6.2 Exemple complet (A réaliser)

```
import numpy as np
import matplotlib.pyplot as plt

x = np.array([1,2,3,4,5])
y = np.array([2.0, 4.1, 6.0, 8.2, 10.1])

dx = 0.1
dy = 0.2

plt.errorbar(x, y, xerr=dx, yerr=dy,
             fmt='o', capsize=5)

plt.xlabel("x")
plt.ylabel("y")
plt.title("Mesures avec incertitudes")
plt.grid()
plt.show()
```

Attention

Erreurs fréquentes :
- oublier `capsize` → barres peu lisibles
- confondre `xerr` et `yerr`
- utiliser `plot` au lieu de `errorbar`

7 Interprétation physique

Les barres d'erreur permettent de :
- vérifier un modèle ;
- estimer une précision ;
- comparer deux séries de mesures.

8 Ajustement affine (régression linéaire)

En TP de physique, on cherche souvent à vérifier une relation du type :

$$y = ax + b$$

On utilise pour cela une régression linéaire.

8.1 Commande numpy.polyfit()

La fonction essentielle est :

```
np.polyfit(x, y, 1)
```

Le "1" signifie : ajustement affine.

8.2 Exemple simple

```
import numpy as np
import matplotlib.pyplot as plt

x = np.array([1,2,3,4,5])
y = np.array([2.1, 4.0, 6.1, 8.2, 9.9])

coef = np.polyfit(x, y, 1)

a = coef[0]
b = coef[1]

print(a, b)
```

8.3 Tracer la droite d'ajustement

Une fois les coefficients obtenus :

```
y_fit = a*x + b

plt.scatter(x, y)
plt.plot(x, y_fit)
plt.grid()
plt.show()
```

8.4 Interprétation physique

Dans de nombreux TP :
— la pente a a une signification physique ;
— l'ordonnée à l'origine b représente un biais expérimental.

Méthode

Pour exploiter une droite :

1. tracer les points expérimentaux ;
2. effectuer un ajustement linéaire ;
3. afficher la droite ;
4. interpréter la pente physiquement.

8.5 Exemple physique : loi d'Ohm

On étudie :

$$U = RI$$

On trace U en fonction de I.

```
I = np.array([0.1,0.2,0.3,0.4,0.5])
U = np.array([0.9,1.8,2.7,3.6,4.5])

coef = np.polyfit(I, U, 1)

R = coef[0]

U_fit = R*I

plt.scatter(I, U)
plt.plot(I, U_fit)
plt.xlabel("I (A)")
plt.ylabel("U (V)")
plt.grid()
plt.show()
```

Attention

Erreurs fréquentes :

- inverser x et y
- oublier que la pente dépend du choix des axes
- confondre modèle et données expérimentales

9 TP complet : loi d'Ohm

On étudie la relation entre la tension U et l'intensité I :

$$U = RI$$

L'objectif est de déterminer expérimentalement la résistance R.

9.1 Données expérimentales

```
import numpy as np
import matplotlib.pyplot as plt

I = np.array([0.10, 0.20, 0.30, 0.40, 0.50])
U = np.array([0.95, 1.92, 2.88, 3.81, 4.78])
```

9.2 Ajout des incertitudes

On suppose :

— incertitude sur I : $\Delta I = 0.01$ A

— incertitude sur U : $\Delta U = 0.05$ V

```
dI = 0.01
dU = 0.05
```

9.3 Représentation graphique

```
plt.errorbar(I, U, xerr=dI, yerr=dU,
             fmt='o', capsize=4,
             label="mesures")
```

9.4 Ajustement linéaire

```
coef = np.polyfit(I, U, 1)

R = coef[0]
U0 = coef[1]

print("R =", R)
print("U0 =", U0)
```

9.5 Droite d'ajustement

```
U_fit = R*I + U0

plt.plot(I, U_fit, label="ajustement")
plt.xlabel("I (A)")
plt.ylabel("U (V)")
plt.grid()
plt.legend()
plt.show()
```

9.6 Interprétation physique

On obtient une relation affine :

$$U = RI + U_0$$

— R est la résistance du conducteur ;
— U_0 représente une erreur systématique ou un offset.

Méthode

Pour exploiter un TP :

1. entrer les données ;
2. ajouter les incertitudes ;
3. tracer les points ;
4. effectuer la régression linéaire ;
5. tracer la droite ;
6. interpréter les paramètres.

Attention

Ne jamais :
— oublier les unités ;
— confondre pente et ordonnée à l'origine ;
— tracer une courbe sans points expérimentaux ;
— oublier les barres d'erreur.

10 Fiche méthode : réussir un TP Python en CPGE

10.1 Méthode générale

Dans tout TP de physique, on suit toujours la même démarche :

1. Importer les bibliothèques (`numpy`, `matplotlib`)
2. Entrer les données expérimentales
3. Ajouter les incertitudes
4. Tracer les points expérimentaux
5. Réaliser un ajustement si nécessaire
6. Interpréter physiquement les résultats

10.2 Erreurs classiques

Attention

Les erreurs les plus fréquentes en TP :
- oublier `plt.show()` ;
- inverser x et y ;
- oublier les unités ;
- confondre modèle et mesures ;
- ne pas tracer les points expérimentaux ;
- oublier les barres d'erreur.

11 Résumé des commandes essentielles

Commande	Rôle
`np.array()`	créer un tableau
`np.linspace()`	valeurs régulières
`plt.plot()`	courbe théorique
`plt.scatter()`	points expérimentaux
`plt.errorbar()`	incertitudes
`np.polyfit()`	régression linéaire
`plt.xlabel()`	axe x
`plt.ylabel()`	axe y
`plt.legend()`	légende
`plt.grid()`	grille
`plt.show()`	affichage
`plt.savefig()`	sauvegarde

12 Exercices

Exercice 1 : courbe simple

Tracer la fonction :

$$y = x^2$$

pour $x \in [-5; 5]$.
Ajouter un titre, une grille et des labels.

Exercice 2 : données expérimentales

On donne les mesures suivantes :

$$x = [1, 2, 3, 4, 5], \quad y = [1.2, 2.1, 3.0, 4.2, 5.1]$$

Tracer les points expérimentaux avec des barres d'erreur :

$$\Delta x = 0.1, \quad \Delta y = 0.2$$

Exercice 3 : loi physique

On étudie une relation du type :

$$y = ax$$

À partir des données expérimentales :

$$x = [1, 2, 3, 4, 5], \quad y = [2.0, 4.1, 6.0, 8.2, 10.1]$$

— tracer les points ;
— effectuer une régression linéaire ;
— déterminer a ;
— interpréter physiquement le résultat.

Conclusion

Ce document regroupe les bases essentielles pour :
— tracer des courbes en physique ;
— exploiter des données expérimentales ;
— représenter des incertitudes ;
— réaliser des ajustements linéaires.