

Pistes et réflexions IA en Physique–Chimie :

Utiliser une IA textuelle pour générer un code Python traçant et modélisant une courbe à partir de données expérimentales

1) GÉNÉRATION DU CODE PYTHON À PARTIR DES DONNÉES EXPÉRIMENTALES :

Utilisation de l'IA TalkAI (pas besoin d'inscription pour essayer gratuitement)

Possible évidemment avec d'autres IA textuelles

<https://talkai.info/fr/chat/>

Proposition de prompt à utiliser (la combinaison **SHIFT + Enter** permet d'aller à la ligne ; un simple espace sépare les données) :

Génère un code python qui permet de tracer sur un graphique en couleur une courbe à partir des données ci-dessous. Tu feras des axes orthonormés avec noms, unités et valeurs sur les axes, et des traits horizontaux et verticaux. Il faudra un titre. Il faudra également afficher l'équation mathématique modélisant cette courbe.

Données :

Position X (en m) 0,535 0,676 0,817 0,959 1,11 1,24 1,37 1,52 1,64

Position Y (en m) 1,32 1,53 1,74 1,94 2,11 2,26 2,40 2,51 2,61

The screenshot shows the TalkAI chat interface. On the left is a sidebar with a 'TalkAI' logo, a 'Nouveau chat' button, and a list of previous chats. The main area is titled 'ChatGPT' and contains a prompt in French asking for a Python script to plot experimental data. The data points are: Position X (m) [0.535, 0.676, 0.817, 0.959, 1.11, 1.24, 1.37, 1.52, 1.64] and Position Y (m) [1.32, 1.53, 1.74, 1.94, 2.11, 2.26, 2.40, 2.51, 2.61]. The chat response shows the generated Python code using NumPy and Matplotlib. The code includes comments in French and uses a polynomial interpolation of degree 2. A 'Copier' button is visible next to the code block.

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

# Données
X = np.array([0.535, 0.676, 0.817, 0.959, 1.11, 1.24, 1.37, 1.52, 1.64])
Y = np.array([1.32, 1.53, 1.74, 1.94, 2.11, 2.26, 2.40, 2.51, 2.61])

# Créer un graphique
plt.figure(figsize=(10, 6))

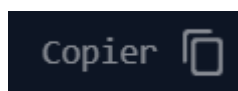
# Tracer la courbe avec couleur
plt.plot(X, Y, color='blue', marker='o', label='Données expérimentales')

# Ajouter les axes avec noms, unités et valeurs
plt.xlabel('Position X (m)')
plt.ylabel('Position Y (m)')

# Tracer des traits horizontaux et verticaux pour chaque point
for x, y in zip(X, Y):
    plt.axhline(y=y, color='gray', linestyle='--', linewidth=0.5, alpha=0.7)
    plt.axvline(x=x, color='gray', linestyle='--', linewidth=0.5, alpha=0.7)

# Titre du graphique
plt.title('Courbe de Position Y en fonction de Position X')
```

Le résultat est le suivant et doit être copié en passant la souris sur le programme et en cliquant sur COPIER en haut à droite :



Programme généré (qui sera différent à chaque lancement du même prompt !)

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

# Données
X = np.array([0.535, 0.676, 0.817, 0.959, 1.11, 1.24, 1.37, 1.52, 1.64])
Y = np.array([1.32, 1.53, 1.74, 1.94, 2.11, 2.26, 2.40, 2.51, 2.61])

# Créer un graphique
plt.figure(figsize=(10, 6))

# Tracer la courbe avec couleur
plt.plot(X, Y, color='blue', marker='o', label='Données expérimentales')

# Ajouter les axes avec noms, unités et valeurs
plt.xlabel('Position X (m)')
plt.ylabel('Position Y (m)')

# Tracer des traits horizontaux et verticaux pour chaque point
for x, y in zip(X, Y):
    plt.axhline(y=y, color='gray', linestyle='--', linewidth=0.5,
alpha=0.7)
    plt.axvline(x=x, color='gray', linestyle='--', linewidth=0.5,
alpha=0.7)

# Titre du graphique
plt.title('Courbe de Position Y en fonction de Position X')

# Définir les limites des axes avec un peu de marge
plt.xlim(min(X) - 0.1, max(X) + 0.1)
plt.ylim(min(Y) - 0.1, max(Y) + 0.1)

# Ajouter une grille
plt.grid(True, which='both', linestyle='--', linewidth=0.5, alpha=0.7)

# Effectuer une interpolation polynomiale de degré 2 (quadratique)
coeffs = np.polyfit(X, Y, 2)
poly_eq = np.poly1d(coeffs)
```

```
# Générer des valeurs pour tracer la courbe lisse
X_fit = np.linspace(min(X), max(X), 200)
Y_fit = poly_eq(X_fit)

# Tracer la courbe modélisée
plt.plot(X_fit, Y_fit, color='red', linewidth=2, label='Modèle
quadratique')

# Afficher l'équation mathématique
equation_str = f'Y = {coeffs[0]:.3f}X² + {coeffs[1]:.3f}X +
{coeffs[2]:.3f}'
plt.text(0.05, 0.95, f'Equation:\n{equation_str}',
transform=plt.gca().transAxes,
        fontsize=12, verticalalignment='top', bbox=dict(boxstyle='round',
facecolor='white', alpha=0.8))

# Ajouter la légende
plt.legend()

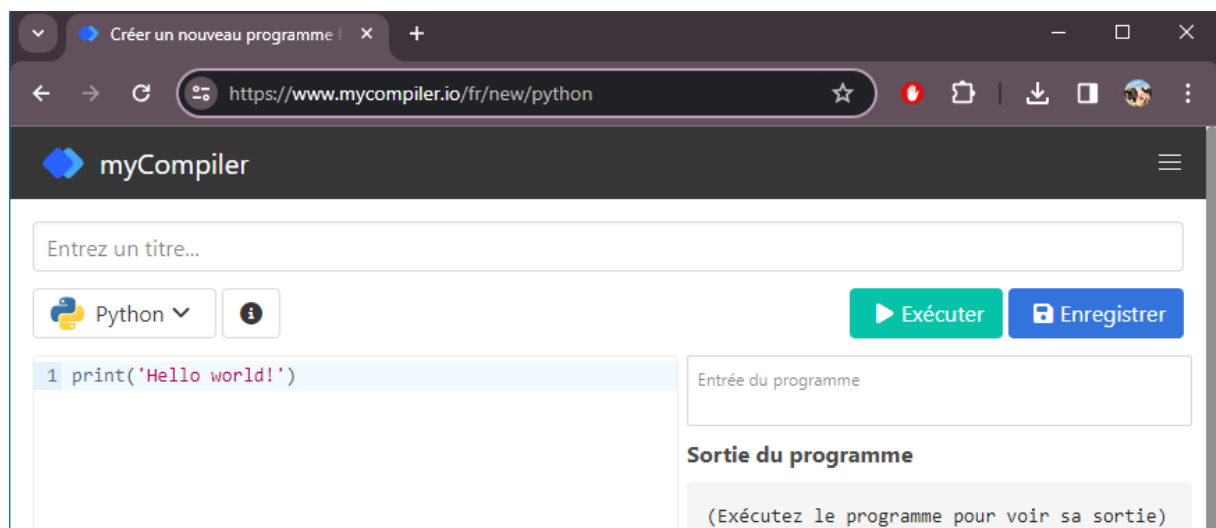
# Afficher le graphique
plt.show()
```

2) COPIER ET LANCER LE CODE PYTHON :

Utilisation d'un IDE Python (Environnement de développement intégré) en ligne : myCompiler.io

Possible également avec d'autres IDE, en ligne ou en local

<https://www.mycompiler.io/fr/new/python>



Remplacer le code déjà écrit par le code généré précédemment (par un coller CTRL + V), puis cliquer sur exécuter pour obtenir le graphique.

myCompiler

Python

Exécuter

Enregistrer

Entrée du programme

Sortie du programme

```
18 # Tracer des traits horizontaux et verticaux pour chaque point
19 for x, y in zip(X, Y):
20     plt.axhline(y=y, color='gray', linestyle='--', linewidth=0.5, alpha=0
21     plt.axvline(x=x, color='gray', linestyle='--', linewidth=0.5, alpha=0
22
23 # Titre du graphique
24 plt.title('Courbe de Position Y en fonction de Position X')
25
26 # Définir les limites des axes avec un peu de marge
27 plt.xlim(min(X) - 0.1, max(X) + 0.1)
28 plt.ylim(min(Y) - 0.1, max(Y) + 0.1)
29
30 # Ajouter une grille
31 plt.grid(True, which='both', linestyle='--', linewidth=0.5, alpha=0.7)
32
33 # Effectuer une interpolation polynomiale de degré 2 (quadratique)
34 coeffs = np.polyfit(X, Y, 2)
35 poly_eq = np.poly1d(coeffs)
36
37 # Générer des valeurs pour tracer la courbe lisse
38 X_fit = np.linspace(min(X), max(X), 200)
39 Y_fit = poly_eq(X_fit)
40
41 # Tracer la courbe modélisée
42 plt.plot(X_fit, Y_fit, color='red', linewidth=2, label='Modèle quadratique')
43
44 # Afficher l'équation mathématique
45 equation_str = f'Y = {coeffs[0]:.3f}X^2 + {coeffs[1]:.3f}X + {coeffs[2]:.3f}
46 plt.text(0.05, 0.95, f'Equation:\n{equation_str}', transform=plt.gca().tr
47         fontsize=12, verticalalignment='top', bbox=dict(boxstyle='round'
48
49 # Ajouter la légende
50 plt.legend()
51
```

Entrée du programme

Sortie du programme

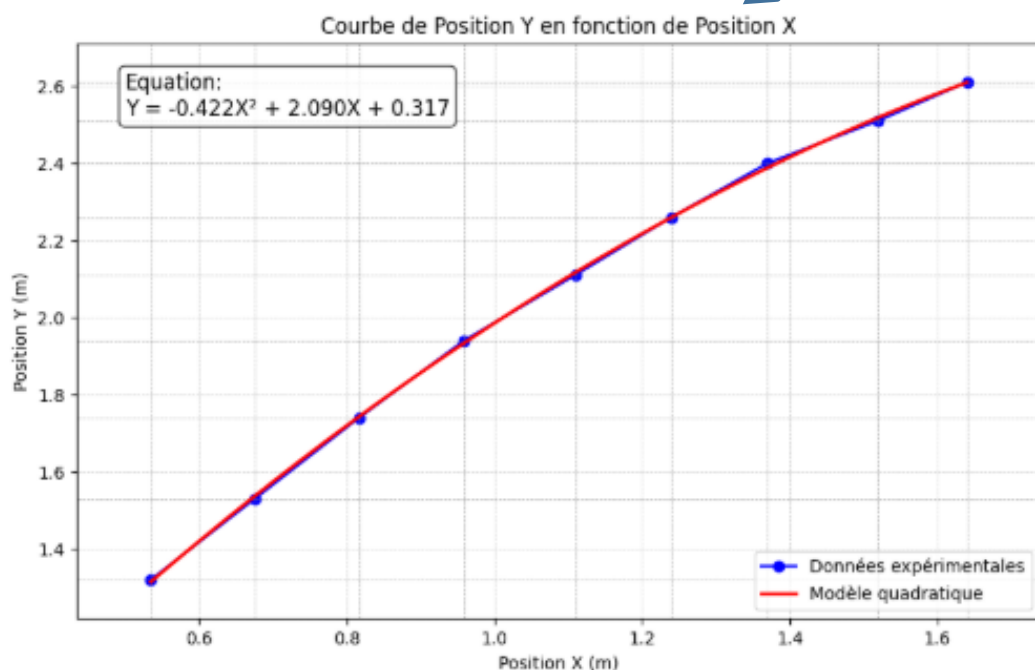
[Execution complete with exit code 0]

Refine your GenAI content with proven evaluation methods

Read on

Maintain search performance with GenAI. [Read On](#)

[www.algolia.com](#)



Il est intéressant de remarquer que si on change les données dans le programme :

- le programme adapte les axes correctement (il faut cliquer à nouveau sur EXECUTER)
- le programme adapte l'équation mathématique (ce n'est pas toujours le cas, certains programmes générés par l'IA font le calcul et propose une équation unique), mais reste cantonné à la fonction polynomial du 2d degré générée au début du programme (partie # *Afficher l'équation mathématique*).

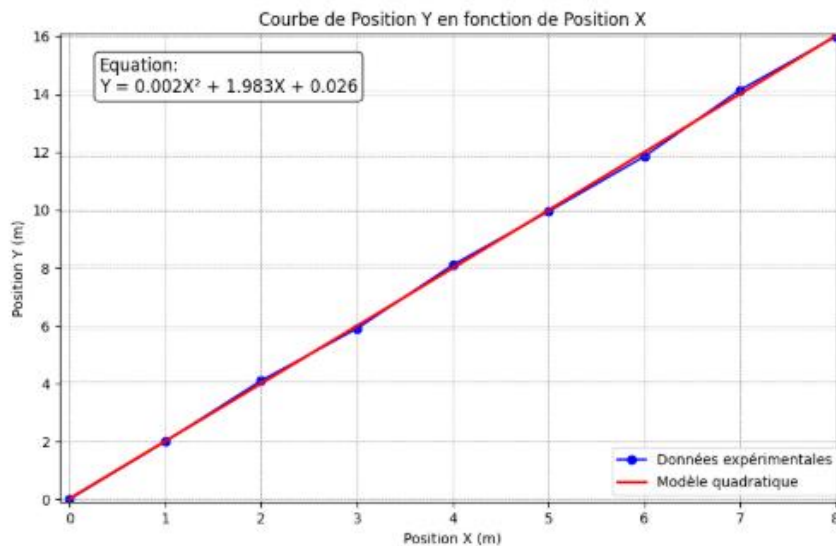
Changement des données expérimentales, test 1, pas de problème :

Lignes 4, 5 et 6

Données

```
X = np.array([0, 1, 2, 3, 4, 5, 6, 7, 8])
```

```
Y = np.array([0, 2, 4.1, 5.9, 8.1, 9.95, 11.85, 14.15, 16])
```

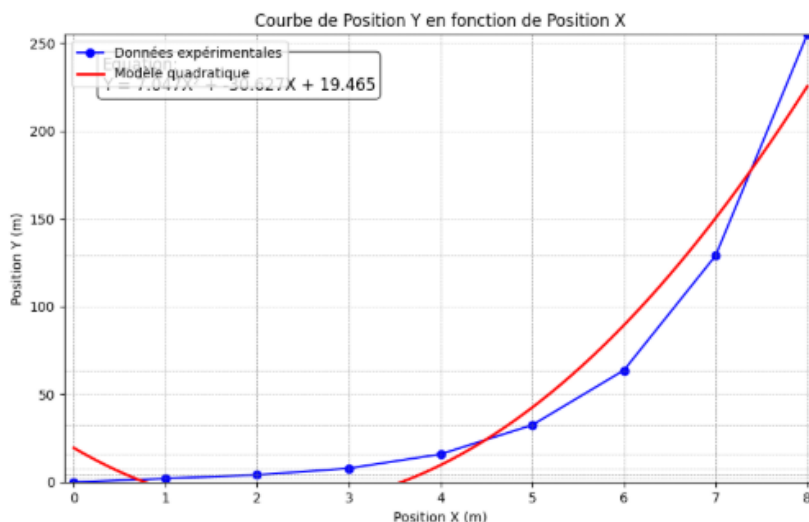


Changement des données expérimentales, test 2 : le programme montre ses limites :

Données

```
X = np.array([0, 1, 2, 3, 4, 5, 6, 7, 8])
```

```
Y = np.array([0, 2.05, 4.1, 7.85, 15.9, 32.5, 63.7, 129, 255])
```



3) RETOURS ET POINTS D'ATTENTION LIÉS À L'IA

Selon les essais, avec le même prompt et la même IA, les programmes générés par l'IA sont différents, notamment la partie modélisation.

Il convient donc de rester prudent et de tester le programme avant de le proposer à ses élèves.

Certains programmes générés ne permettront pas d'avoir un graphique et/ou une équation qui s'adaptent à la modification des données.

Certains autres programmes générés ne fonctionnent tout simplement pas (erreur de syntaxe), mais c'est de plus en plus rare.