FICHE-OUTIL MATHÉMATIQUE · M2

Tracer et modéliser un nuage de points

Exploiter une série de mesures $(x\,;\,y)$, c'est d'abord **tracer le nuage de points**, puis le **modéliser**.

Dans de nombreux cas, une modélisation affine est appropriée : une **droite** $y = a\,x + b$ (de pente a et d'ordonnée à l'origine b).

Cette fiche donne la marche à suivre **à la main**, **au tableur**, **à la calculatrice** et **en Python**.

---

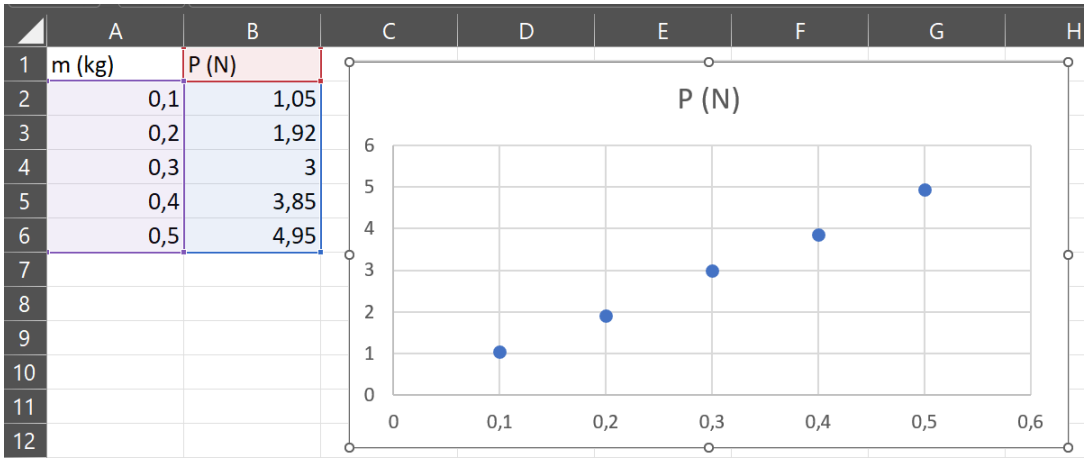
1. Tracer un nuage de points

À la main

- Choisir l'**échelle** de chaque axe d'après la **plus grande valeur** à représenter, de façon à occuper toute la place dévolue ; graduer **régulièrement**.
- **Inclure l'origine** $(0\,;\,0)$ si on veut pouvoir juger un passage par l'origine.
- **Légender** chaque axe (grandeur **et** unité), puis placer les points.

Au tableur (Excel, LibreOffice Calc)

1. Saisir les mesures dans **deux colonnes** : x, puis y.
2. Sélectionner les deux colonnes.
3. **Insertion → Graphique → Nuage de points (XY)**.

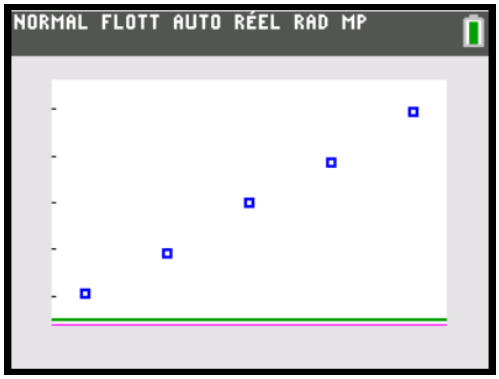


Nuage de points sur Excel.

À la calculatrice (TI)

- *(optionnel)* `2nde` + (mémoire) → **EffTtesListes** : efface les listes existantes.

- `stats` → **EDIT** → saisir les valeurs de x dans **L1**, celles de y dans **L2**.
- `2nde f(x)` (**graph stats**) → **Graphe 1** → **Aff** (activer l'affichage).
- `zoom` → **9 : ZoomStat** : la fenêtre s'ajuste automatiquement au nuage.



Nuage de points (ZoomStat) sur la TI-83.

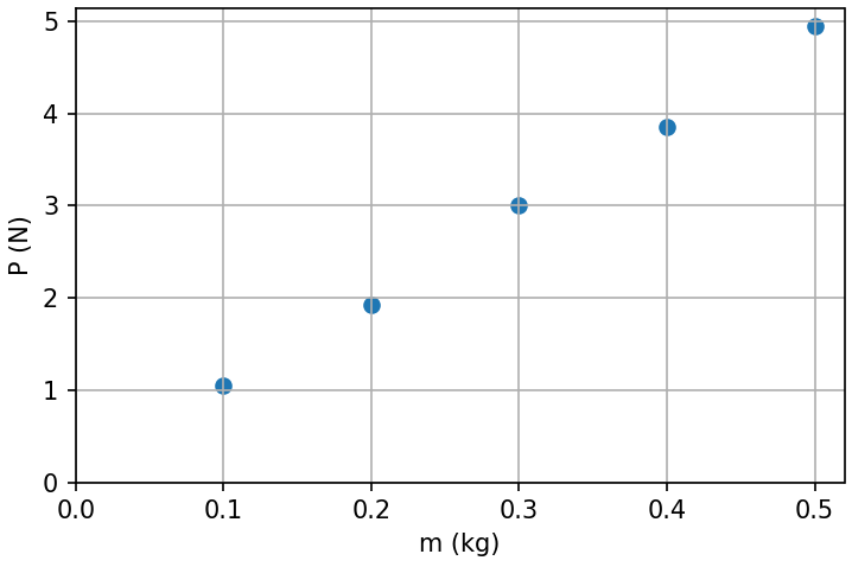
En Python

Saisir les deux séries dans des listes, puis afficher le nuage avec `scatter` (ici les mesures poids-masse de l'exemple) :

```
import matplotlib.pyplot as plt    # bibliothèque de tracé de graphes

m = [0.1, 0.2, 0.3, 0.4, 0.5]      # données en abscisses : masse (kg)
P = [1.05, 1.92, 3.0, 3.85, 4.95]  # données en ordonnées : poids (N)

plt.scatter(m, P)                  # nuage de points (x, y)
plt.xlabel("m (kg)")               # légende abscisses
plt.ylabel("P (N)")                # légende ordonnées
plt.xlim(left=0)                   # axes qui démarrent à 0 (cf. pièges)
plt.ylim(bottom=0)
plt.grid(True)                     # afficher la grille
plt.show()                         # afficher le graphe
```



Sortie graphique du script : le nuage de points.

---

2. Modéliser par une droite

À la main

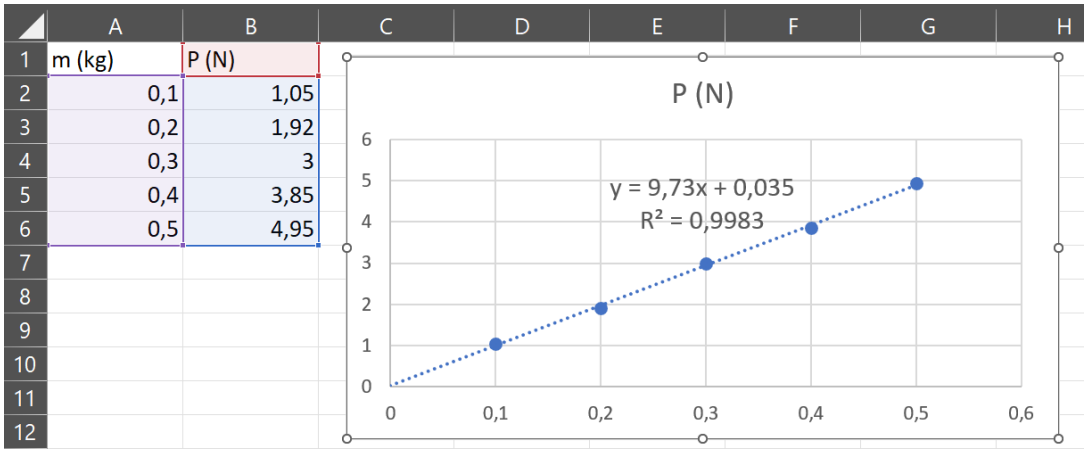
- Si les points semblent alignés aux erreurs de mesure près, tracer la **meilleure droite** (au plus près des points, autant de points de part et d'autre).
- Lire l'**ordonnée à l'origine** b (intersection avec l'axe des ordonnées).
- Mesurer le **coefficient directeur** a avec **deux points** A **et** B **pris sur la droite**, bien écartés :

$$a = \frac{y_B - y_A}{x_B - x_A}.$$

Attention à choisir des points de la droite et non des points de mesure : la meilleure droite prend en compte l'ensemble des mesures et est donc plus précise que chacun des points pris séparément.

Au tableur

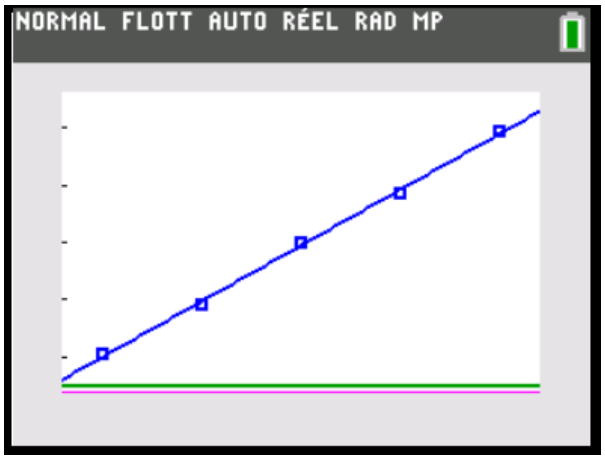
1. Clic droit sur un point de mesure → **Ajouter une courbe de tendance** → **Linéaire**.
2. Cocher **Afficher l'équation** et **Afficher le coefficient** R^2.
3. Vérifier $R^2 > 0{,}98$: les points sont alors **bien alignés**.
4. Lire a et b sur l'équation affichée $y = a\,x + b$.



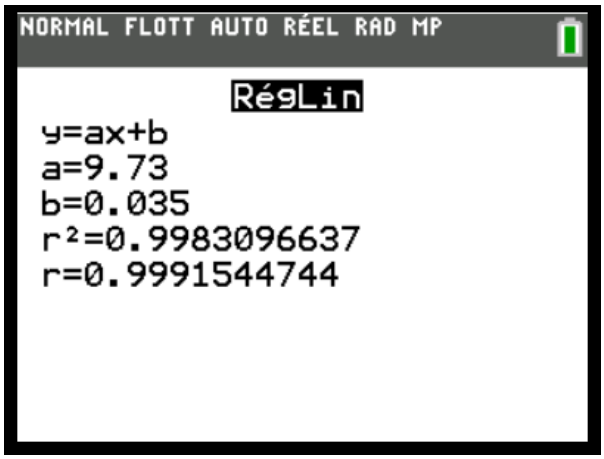
Données, nuage de points et modélisation linéaire au tableur (équation et R^2).

À la calculatrice (TI)

- *(optionnel pour tracer la droite, inutile si le modèle suffit)* `stats` → **CALC** → **RégLin (ax+b)** → **Enr régEQ** → **alpha f4** choisir **Y1** (la droite sera tracée par-dessus le nuage).
- `stats` → **CALC** → **RégLin (ax+b)** → **Calculer**.
- Lire a, b et r^2 ; vérifier $r^2 > 0{,}98$.



Droite de régression sur le nuage (TI-83).



Résultats de RégLin (ax+b) (TI-83).

En Python

La fonction `linregress` de la bibliothèque `scipy.stats` renvoie la pente `slope` (a), l'ordonnée à l'origine `intercept` (b) et la corrélation `rvalue` (r) — comme la calculatrice. Elle accepte directement des listes. On superpose ensuite la droite au nuage :

```
import matplotlib.pyplot as plt                # bibliothèque de tracé de graphes
from scipy.stats import linregress             # fonction "régression linéaire"

m = [0.1, 0.2, 0.3, 0.4, 0.5]                  # masse (kg)
P = [1.05, 1.92, 3.0, 3.85, 4.95]              # poids (N)
```

```
reg = linregress(m, P)                          # réalisation de la régression
a = reg.slope                                   # coefficient directeur
b = reg.intercept                               # ordonnee a l'origine
R = reg.rvalue                                  # coefficient R
print("a =", round(a, 2), " b =", round(b, 3), " r^2 =", round(R ** 2, 4))
```

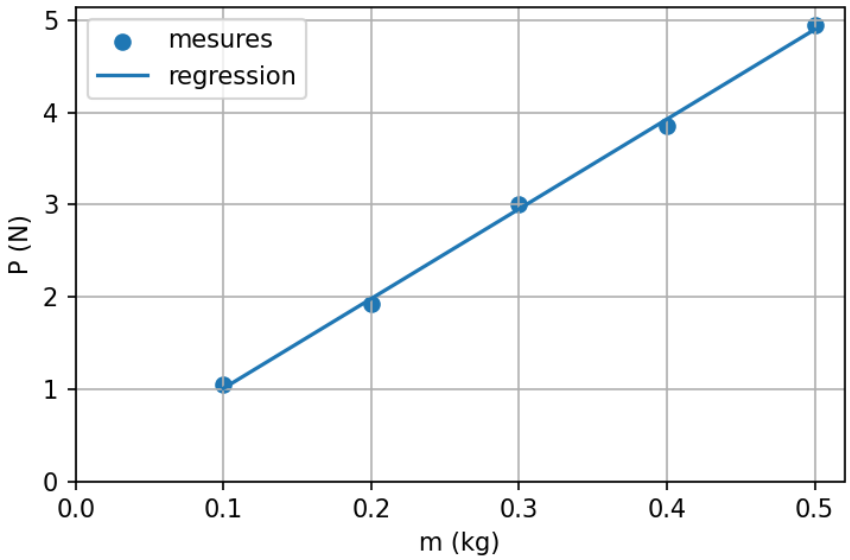
Sortie console :

```
a = 9.73  b = 0.035  r^2 = 0.9983
```

```
# OPTIONNEL : TRACÉ DU GRAPHE

# modèle affine
m_modele = [0, m[-1]]       # m[-1]: dernière valeur de la liste m
P_modele = [b, a*m[-1]+b]   # valeurs correspondantes de P

plt.scatter(m, P, label="mesures")
plt.plot(m_modele, P_modele, label="regression")  # modèle affine
plt.xlabel("m (kg)")
plt.ylabel("P (N)")
plt.xlim(left=0)                                # axes qui demarrent a 0
plt.ylim(bottom=0)
plt.legend()
plt.grid(True)
plt.show()
```



Sortie graphique du script : la droite de régression superposée au nuage.

Pièges fréquents

- **Échelle régulière, axes à 0.** Pour juger un passage par l'origine, les axes doivent inclure 0 et être gradués régulièrement.
- **Pente.** Mesurer a avec deux points pris **sur la droite tracée**, bien écartés.
- **R^2 (ou r^2) ne juge que l'alignement.** Proche de 1, il confirme que les points sont alignés — **pas** que la droite passe par l'origine. Pour une proportionnalité, vérifier *en plus* que b est négligeable (fiche **M3**).