TD 5 : Algèbre linéaire, ajustement

Exercice 5.1 : Méthode de Gauss

(a) *Sur papier :* déroulez à la main l'algorithme de Gauss comme il a été présenté au cours pour le système d'équations

$$\begin{aligned} -4x_2 + x_3 &= 3 \\ x_1 + 3x_2 - x_3 &= 0 \\ -2x_1 - 2x_2 + 2x_3 &= 1 \end{aligned}$$

Vérifiez le résultat avec un programme sur ordinateur.

(b) (⋆) Une des étapes de l'algorithme de Gauss est le choix d'un pivot avant d'éliminer une colonne. Afin de minimiser les erreurs d'arrondi, il convient de choisir comme pivot l'élément de la colonne dont la valeur absolue est la plus grande (plutôt que le premier élément non nul trouvé, comme dans le code du cours). Modifiez le programme `gauss.py` du cours pour implémenter cette variante de recherche de pivot.

Exercice 5.2 : Décompositions matricielles

(a) *Sur papier*, trouvez une méthode efficace pour calculer le déterminant d'une matrice triangulaire.

Puis, réalisez une fonction `absdet(M)` qui retourne la valeur absolue $|\det(M)|$ du déterminant d'une matrice carrée `M`, calculé à partir de sa décomposition LU. Pour le calcul de cette dernière, servez-vous de la fonction `scipy.linalg.lu`.
Indication : Le déterminant d'une matrice de permutation P est ± 1.

(b) (⋆) Réalisez une fonction `det(M)` qui retourne le déterminant de M, signe compris.
Indication : En fait, $\det(P) = \pm 1$ peut être calculé efficacement avec la formule de Laplace (car toutes les lignes de P sont zéro hormis un seul élément qui est 1). Servez-vous de la fonction `numpy.delete` : `numpy.delete(P, i, 0)` retourne `P` avec la ligne `i` supprimée et `numpy.delete(P, j, 1)` retourne `P` avec la colonne `j` supprimée.

(c) Réalisez une fonction `diagonaliser(A, epsilon=1.0E-5, maxit=30)` qui calcule les valeurs propres et les vecteurs propres de la matrice `A` avec l'algorithme QR. Le paramètre ϵ correspond à la précision numérique demandée. Servez-vous de la fonction `scipy.linalg.qr`.

Votre fonction déroulera les itérations de la méthode QR jusqu'à ce que A_n soit triangulaire supérieure, à une erreur numérique de $\mathcal{O}($`epsilon`$)$ près (trouvez une condition appropriée). Si elle dépasse `maxit` itérations, elle terminera avec un message d'erreur. Elle retournera un tableau contenant les vecteurs propres et un deuxième contenant les valeurs propres.

Exercice 5.3 : Régression linéaire

L'activité d'une source radioactive en fonction du temps obéit la loi exponentielle $A(t) = A_0\, e^{-\lambda t}$ avec A_0 l'activité initiale à $t = 0$ et $\lambda = 1/\tau$ la constante de désintégration, c.-à-d. l'inverse de la durée de vie moyenne d'un noyau. On mesure (avec une erreur de mesure sur A estimée de $\pm 1\%$)

temps t [h]	activité A [PBq]
0	2.02
5	1.95
10	1.93
20	1.88
40	1.73
80	1.50

On souhaite déterminer λ de ces données par une régression linéaire. Transformez alors d'abord la loi de désintégration ci-dessus en relation linéaire entre les paramètres β_1 et β_2,

$$f(t;\, \beta_1,\, \beta_2) = \beta_1 + \beta_2\, t$$

avec $\beta_1(A_0)$ et $\beta_2(\lambda)$ des fonctions simples. Trouvez le $\vec{\beta}$ qui minimise χ^2 numériquement et déduisez-en la valeur de λ et de τ. (Pour l'ajustement, vous pouvez traiter l'incertitude sur les données comme constante — rendez-vous compte pourquoi.)

Exercice 5.4 : L'algorithme de Levenberg-Marquardt ($\star$)

Implémentez l'algorithme de Levenberg-Marquardt présenté en cours. Vous pouvez prendre le code de la méthode de Gauss-Newton comme point de départ. Réalisez alors une fonction `lev_mar(t, y, f, gradf, beta0, lam0=1.E-4, epsilon = 1.E-2)` dont les arguments sont

- deux tableaux `t` et `y` contenant les données,
- une fonction `f` représentant la fonction modèle $f(t;\, \vec{\beta})$,
- une liste de fonctions `gradf` contenant les dérivées du modèle selon les paramètres,
- un tableau `beta0` des valeurs de départ des paramètres,
- la valeur de départ `lam0` pour λ,
- le paramètre `epsilon` pour tester la convergence (si χ^2 diminue par moins que `epsilon` entre deux itérations, la fonction s'arrêtera et renverra les résultats).

La fonction retournera χ^2 et le tableau des paramètres ajustés.

Exercice 5.5 : La résonance Z

Dans les années 1990, l'expérience OPAL au CERN a mesuré la section efficace pour la production des paires de muons dans des collisions électron-positron, qui a lieu principalement par un échange d'une particule Z virtuelle. Pour une énergie proche de $m_Z c^2$, la section efficace en fonction de l'énergie E est approximée par la fonction de Breit-Wigner,

$$\sigma(E) = \frac{N^2\, E^2}{(m_Z^2 c^4 - E^2)^2 + m_Z^2 c^4 \Gamma_Z^2}$$

où N est une constante de normalisation, m_Z est la masse du boson Z et $\Gamma_Z = \hbar/\tau_Z$ est sa largeur de désintégration. Vous trouverez σ pour quelques énergies E entre 85 et 95 GeV dans le fichier `zpeak.txt` (source : `https://www.hepdata.net/record/ins538108`).

Réaliser un programme Python qui importe ces données et les utilise pour trouver m_Z et Γ_Z, soit par la méthode de Gauss-Newton, soit par celle de Levenberg-Marquardt.

Indications : Il convient de travailler en unités où $\hbar = 1$ et $c = 1$; ainsi $[E] = [m_Z] = [\Gamma_Z] =$ GeV. A noter qu'il faudra aussi trouver N par ajustement, et choisir des valeurs de départ appropriées pour que la méthode converge.