TD 6 : Algèbre linéaire numérique, ajustement

Exercice 6.1 : Méthode de Gauss et décomposition LU

(a) *Sur papier :* Déroulez à la main l'algorithme de Gauss pour le système d'équations

$$\begin{aligned} 5x_1 + 3x_2 - x_3 &= 0 \\ -2x_1 + 2x_2 - 4x_3 &= 1 \\ -x_2 + x_3 &= 3 \end{aligned}$$

(b) *Sur papier :* Déroulez à la main l'algorithme du cours pour la décomposition LU sur la matrice de coefficients du système linéaire

$$\begin{aligned} 2\,x_2 - 2\,x_3 + 2\,x_4 &= 4 \\ -x_1 + 3\,x_2 + 4\,x_3 + 2\,x_4 &= 2 \\ x_1 - x_2 - 3\,x_4 &= 2 \\ x_1 + x_2 - 2\,x_3 &= 0 \end{aligned}$$

Ensuite, trouvez les x_i avec l'aide des matrices L, U et P.

(c) Réalisez deux fonctions `matriceP(n, i, j)` et `matriceT(n, i, j, r)` qui retournent les matrices $n \times n$ $P_{(i,j)}$ et $T_{(i,j;r)}$ du cours.

(d) En modifiant le code de la fonction `triangulariser(M)` du cours, réalisez une fonction `decomp_LU(M)` qui effectue la décomposition LU de la matrice carrée `M` et qui retourne trois tableaux, à savoir les matrices L, U et P.

(e) *Sur papier :* Trouvez des simples méthodes pour calculer le déterminant d'une matrice triangulaire et d'une matrice de permutation.

(f) Réalisez une fonction `det(M)` qui retourne le déterminant d'une matrice carrée `M`.

Exercice 6.2 : L'algorithme QR

Réalisez une fonction `diagonaliser(A, epsilon=1.0E-5, maxit=30)` qui calcule les valeurs propres et les vecteurs propres de son premier argument la matrice `A`. Servez-vous des fonctions déjà réalisées ainsi que de la fonction `decomp_QR()` du cours. La fonction testera d'abord si `A` est une matrice symétrique et sinon, termine avec un message d'erreur. Puis elle déroulera l'algorithme QR jusqu'à ce que A_n soit triangulaire supérieure, à une erreur de $\mathcal{O}$(`epsilon`) près (trouvez une condition appropriée). Si elle dépasse `maxit` itérations, elle terminera avec un message d'erreur. Enfin elle retournera un tableau qui contient les vecteurs propres et un deuxième qui contient les valeurs propres.

Exercice 6.3 : Algèbre linéaire avec `NumPy` et `SciPy`

Avec l'aide de la documentation de `scipy.linalg`, trouvez une fonction pour resoudre le système d'équations linéaires $A\vec{x} = \vec{b}$ dans cette bibliothèque. Vérifiez vos solutions de l'exercice 6.1 avec un programme qui se sert de cette fonction.

Exercice 6.4 : 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 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 une 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 6.5 : L'algorithme de Levenberg-Marquardt

Implémentez l'algorithme de Levenberg-Marquardt présenté en cours. Vous pouvez prendre le code de la méthode de Gauss-Newton du cours 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 méthode s'arrête).

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

Exercice 6.6 : La résonance Z

Aux années 1990, l'expérience OPAL au CERN a mesuré la section efficace pour la production des pairs de muons dans des collisions électron-positron, qui s'effectue 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 approximativement décrite 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.