TD 3 : Interpolation et approximation polynômiale

Interpolation

Exercice 1. (Algorithme des différences divisées) Construire une fonction qui prend en entrée une fonction, un entier n et un intervalle $[a, b]$ et qui renvoit les différences divisées de f associée à la subdivision uniforme à $n+1$ points sur $[a, b]$. *Attention :* on fera attention à bien gérer la mémoire. Comment cette fonction se transpose-t-elle dans le cas non uniforme ?

Exercice 2. Construire une fonction prenant en entrée un entier n, une fonction f un intervalle $[a, b]$, et qui renvoit le polynôme d'interpolation de Lagrange de f associé à la subdivision uniforme à $n+1$ points de $[a, b]$. On pourra utiliser un algorithme de Horner qui construit la suite des coefficients du polynôme à partir de son écriture via les différences divisées.

Tester l'interpolation de Lagrange sur la fonction exponentielle. La tester ensuite sur

$$f : \begin{array}{rcl} [-1,1] & \longrightarrow & \mathbb{R} \\ x & \longmapsto & \dfrac{1}{1+25x^2}. \end{array}$$

Que remarque-t-on ?

Exercice 3. (Optimalité du résultat de convergence)

1. On se replace dans le cas d'une subdivision uniforme $x_i = a + i\frac{b-a}{n} = a + ih$. On note $\pi_{n+1}(x) = \prod_{i=0}^{n}(x - x_i)$ et $\varphi(s) = |s(s-1)\cdots(s-n)|$.
   (a) On rappelle la formule de Stirling : $n! \sim \sqrt{2\pi n}(n/e)^n$ lorsque $n \to +\infty$. Montrer qu'à partir d'un certain rang,
   $$\varphi\left(\frac{1}{2}\right) \geqslant \frac{1}{2\sqrt{2}}\left(\frac{n}{e}\right)^n.$$
   (b) En déduire que $\|\pi_{n+1}\|_\infty \geqslant \frac{1}{n\sqrt{8}}\left(\frac{b-a}{e}\right)^{n+1}$ à partir d'un certain rang.
2. On cherche à réduire l'erreur dans l'approximation uniforme de l'interpolation en minimisant $\|\pi_{n+1}\|_\infty$. On définit ainsi les points de *Tchebychev* : on considère la suite de polynômes T_n définie par $T_0 = 1$, $T_1 = X$ et pour tout $n \geqslant 1$,
   $$T_{n+1} = 2XT_n - T_{n-1}.$$
   (a) Établir le degré ainsi que le coefficient dominant de T_n, et montrer que pour tout $n \in \mathbb{N}$, $T_n(x) = \cos(n \arccos(x))$.
   (b) Déterminer les racines de T_n (que l'on notera $(\lambda_k^n)_{0 \leqslant k \leqslant n}$).
   (c) Montrer que $X^n - 2^{-n}T_n$ est la meilleure approximation uniforme de X^n d'ordre n. En d'autres termes,
   $$2^{-n}T_n = \arg\min\{\|\pi\|_\infty, \pi \in \mathbb{R}^n[X] \text{ unitaire}\}.$$
   (*Rappel :* $p \in \mathbb{R}_n[X]$ est la meilleure approximation uniforme d'ordre n de f si, et seulement si, $f - p$ équioscille sur au moins $n+2$ points.)
   (d) On définit alors la subdivision à $n+1$ points :
   $$x_i = \frac{a+b}{2} + \frac{b-a}{2}\lambda_i^{n+1}.$$
   Pour ce choix de x_i, expliciter le polynôme π_{n+1} correspondant en fonction de T_{n+1}, et montrer qu'alors
   $$\|\pi_{n+1}\|_\infty = 2\left(\frac{b-a}{4}\right)^{n+1}.$$

3. Programmer une méthode d'interpolation complète aux points de Tchebychev (avec l'algorithme de différences divisées), et la comparer avec celle pour des points équirépartis et pour la fonction f de l'exercice précédent.

Approximation

Exercice 4. (Les polynômes de Bernstein - convergence en norme L^∞) Construire une fonction qui prend en entrée une fonction f définie sur $[0,1]$ et un entier n, et renvoit son $n$ième polynôme de Bernstein $B_n(f) = \sum_{k=0}^{n} C_n^k f(k/n) X^k (1-X)^{n-k}$. Illustrer graphiquement la convergence de $B_n(f)$ vers f.

Exercice 5.

1. On va construire un jeu de données modélisant une fonction "cosinus bruitée"
   (a) Construire une suite de 50 points $(x_i)_{1\leqslant i\leqslant 50}$ de $[0,1]$ générés aléatoirement et rangés dans l'ordre croissant. (On pourra utiliser la fonction `rand` de `numpy.random` et une fonction de tri built-in.)
   (b) Construire une suite de points $(y_i)_{1\leqslant i\leqslant 50}$ définis par $y_i = \cos(x_i) + \varepsilon_i$, avec ε_i un bruit aléatoire donné par une gaussienne centrée de variance 0.01.
   (c) Tracez l'ensemble des points (x_i, y_i) ainsi que la fonction cosinus sur $[0,1]$.
2. Construire une fonction qui prenne en entrée un entier p et le jeu de données (x_i, y_i) et qui renvoie le polynôme de degré p de meilleure approximation au sens des moindres carrés de ce jeu de données, i.e.
$$P_{bruit} = \arg\min_{P\in\mathcal{P}_p} \sum_{i=1}^{50} |y_i - P(x_i)|^2.$$
   Tracer ce polynôme pour différents (petits) degrés. On pourra utiliser des solveurs linéaires built-in de `numpy`.
3. Refaire la question précédente avec le jeu de donnée modifié avec $y_1 = -1$. Cela modifie-t-il drastiquement le polynôme de meilleure approximation ?
4. Que retrouve-t-on comme polynôme dans le cas où le degré p et le nombre de points n vérifient $p = n-1$?
5. Comparer le polynôme d'approximation trouvé sur la solution bruitée avec celui de même degré donné par l'approximation continue au sens des moindres carrés de la fonction non bruitée, i.e.
$$P_{exact} = \arg\min_{P\in\mathcal{P}_p} \int_0^1 |\cos(t) - P(t)|^2 \mathrm{d}t.$$
6. Refaire l'exercice avec la fonction :
$$\begin{array}{rccl} f: & [0,1] & \longrightarrow & \mathbb{R} \\ & x & \longmapsto & \begin{cases} \cos(x) & \text{si } x \leqslant 0.5 \\ \cos(x)+2 & \text{sinon} \end{cases} \end{array}$$

Exercice 6. (Le cas de la régression linéaire) On se donne une série statistique $(x_i, y_i)_{1\leqslant i\leqslant n}$ et deux réels a et b qui minimisent le résidu : $R = \sum_{i=1}^{n} |y_i - ax_i - b|^2$.

1. Écrire $u = \begin{pmatrix} a \\ b \end{pmatrix}$ comme solution du système linéaire
$$B^T B u = B^T y$$
   où l'on explicitera B, $B^T B$ et $B^T y$.
2. En déduire des formules explicites pour a et b en fonction des x_i et des y_i.
3. Interpréter a en terme de variance et de covariance de la série statistique.