Université de Montpellier **Année Universitaire 2025-2026**

TP noté HAX604X.

Déposer votre ou vos fichiers (2 au maximum) sur moodle dans l'espace de dépôt TP noté 2026. Pas d'envoi par mail! Respectez l'heure limite. *Formats acceptés* ***.ipynb*** *ou* ***.py***. La notation tiendra compte de la validité du code, de sa lisibilité, des commentaires et des légendes des figures. Les deux exercices sont indépendants. Barème indicatif.

1 Schémas implicite et explicite. (14 pts)

Soit l'équation différentielle avec condition initiale : $y' = f(t, y), \quad y(t_0) = y_0$.
Le schéma du point milieu *explicite* est donnée (sur un pas de temps h) par :

$$y_1 = y_0 + h\left(f(t_0 + \frac{h}{2}, y_0 + \frac{h}{2}f(t_0, y_0)\right)$$

Le schéma du point milieu *implicite* est donnée par :

$$y_1 = y_0 + h\left(f(t_0 + \frac{h}{2}, y_0 + \frac{h}{2}f(t_0 + h, y_1)\right).$$

Comme c'est un schéma implicite, à chaque pas de temps y_1 est solution d'un problème de point fixe qu'on résout de manière approchée en effectuant 3 itérations de point fixe, en initialisant par $z = y_0 + hf(t_0, y_0)$. Cela signifie que l'on effectue 3 fois le calcul $z = y_0 + h\left(f(t_0 + \frac{h}{2}, y_0 + \frac{h}{2}f(t_0 + h, z)\right)$ avant de poser $y_1 = z$ et de passer au pas de temps suivant.

1. Codez ces schéma dans deux fonctions `milex(f, t0, tfinal, y0, h)` et `milimp(f, t0, tfinal, y0, h)` qui doivent retourner le vecteur des instants t_n entre `t0` et `tfinal` et le vecteur des valeurs y_n.
2. On prend la fonction $f(t, y) = y^2 - y^3$ et les paramètres : intervalle de temps $[\texttt{t0}, \texttt{tfinal}] = [0, 2000]$, donnée initiale $\texttt{y0} = 0.001 = \texttt{1e}-3$. Tracer sur la même figure pour le pas de temps $h = 1$ les solutions numériques calculées par les schémas explicite et implicite
3. tracer sur le même graphe la solution calculée par Python (`from scipy.integrate import solve_ivp`) en utilisant le solveur `solve_ivp(f, [t0, tfinal], [y0], 'Radau')` (si besoin, consulter l'aide `https://docs.scipy.org/doc/scipy/`.)
4. Tracer les solutions numériques données par vos fonctions `milex` et `milimp` avec un pas de temps $h = 2$. Expliquer pourquoi le *schéma implicite* donne ici des résultats aberrants. (Ecrire en commentaire votre explication).

2 Codage de la méthode des caractéristiques. (8 pts)

On considère l'équation aux dérivées partielles non linéaire $u_t + u\, u_x = 0$ avec la donnée initiale $u(x, t = 0) = f(x)$. On rappelle que les caractéristiques dans le plan (x, t) sont les *droites* d'équation $x = x_0 + tf(x_0)$ et que la solution de l'EDP est donnée par $u(x, t) = f(x_0) \quad \text{si} \quad x = x_0 + tf(x_0)$.

1. On prend comme donnée initiale la fonction

$$f(x) = \begin{cases} 0 & \text{si} & x \leq 0 \\ x & \text{si} & 0 \leq x \leq 1 \\ 1 & \text{si} & 1 \leq x \end{cases}$$

   codée en Python `f = lambda x: (0)*(x<0) + x*(0<x)*(x<1) + (1)*(x>=1)`.
   Tracez les droites caractéristiques dans le plan (x, t) pour $x_0 \in \texttt{np.linspace}(-3, 3, 50)$ et $t \in \texttt{np.linspace}(0, 2, 100)$.
2. Dans une autre figure, tracer les graphes des solutions $x \in [-3, 3] \mapsto u(x, t)$ pour $t = 0,\ t = 1,\ t = 2$.
3. On prend maintenant comme donnée initiale la fonction

$$g(x) = \begin{cases} 1 & \text{si} & x \leq 0 \\ 1 - x & \text{si} & 0 \leq x \leq 1 \\ 0 & \text{si} & 1 \leq x \end{cases}$$

   Tracer à nouveau les droites caractéristiques dans le plan (x, t). Dans une autre figure, tracer les graphes des solutions $x \in [-3, 3] \mapsto u(x, t)$ pour $t = 0,\ t = 1,\ t = 2$. Que se passe-t-il pour $t > 1$? Expliquer (en commentaire.)