Volumes finis pour les lois de conservation scalaires

L'objectif de ce TP est d'écrire un code (dans le langage de votre choix) pour la résolution approchée de l'équation scalaire

$$\begin{cases} \partial_t u + \partial_x f(u) = 0, \quad t > 0, x \in (0,1), \\ u(0,\cdot) = u^0, \quad x \in (0,1), \end{cases} \tag{1}$$

$u = u(x,t) \in \mathbb{R}$, avec un schéma de volumes finis à 3 points écrit sous la forme

$$\begin{cases} \dfrac{u_j^{n+1} - u_j^n}{\Delta t} + \dfrac{f_{j+1/2}^n - f_{j-1/2}^n}{\Delta x} = 0, \quad n \in \mathbb{N}, j = 1, \dots, J, \\ u_j^0 = u^0(x_j), \quad j = 1, \dots, J \end{cases} \tag{2}$$

où J est le nombre de cellules utilisées pour mailler l'intervalle $(0,1)$, $\Delta x = 1/J$, et $f_{j+1/2}^n = F(u_j^n, u_{j+1}^n)$ est un flux numérique à préciser. Le calcul de $f_{1/2}^n$ et de $f_{J+1/2}^n$ se fera en utilisant des conditions aux limites bien choisies (voir ci-après).

Pour cela, je vous suggère de choisir la structure générale de code suivante.

1) Déclaration des fonctions : u^0 (qu'on pourra noter `u0`), f et sa dérivée f'.

2) Déclaration des variables et initialisation : J, Δx (noté `dx`), `X` $= (x_j)_j$ (vecteur des positions des centres de mailles, $x_j = (j-1/2)\Delta x$ pour $j = 1, \dots, J$), `U = u0(X)` (dans un langage qui permet cette écriture vectorielle), temps final `T`, temps courant `t = 0`, indice en temps courant `n = 0`.

3) Boucle en temps du type `while t<T`, dans laquelle figurent les opérations

- calcul de la vitesse de propagation maximale des ondes sur tout le domaine de calcul, calcul du pas de temps Δt (`dt`) en respectant la condition CFL et en pensant, après le calcul du pas de temps souhaité, à écrire `dt = min(dt, T - t)` (pourquoi ?), incrémentation de t et de n ;
- boucle en espace pour le calcul des flux à gauche `fG(j)` $= f_{j-1/2}$, puis calcul des flux à droite `fD(j)` $= f_{j+1/2}$;
- boucle en espace pour le calcul du nouveau vecteur `U` (dans les langages scilab, matlab ou python, ceci peut se faire de la manière vectorielle `U = U - dt/dx*(fD - fG)`) et affichage éventuel de la solution (pour voir le « film » de la solution) ;
- affichage de la solution à l'instant final, et éventuelle sauvegarde.

Dans la suite, on pourra essayer différents schémas numériques (méthode de Godunov, méthode de Murman-Roe, méthode de Rusanov) et les comparer entre eux, faire varier le nombre de Courant et vérifier expérimentalement l'optimalité des conditions de stabilité.

Remarque (en forme de question). Pourquoi est-il proscrit, pour la mise à jour de `U`, d'écrire une unique boucle sur `j` du type (disons, en langage Matlab et avec un flux de type Rusanov)

```
for j = 1:J
fG = 0.5*(f(u(j-1) + f(u(j)) + 0.5*c*(u(j-1) - u(j));
fD = 0.5*(f(u(j) + f(u(j+1)) + 0.5*c*(u(j) - u(j+1));
u(j) = u(j) - dt/dx*(fD - fG);
```

?
Cette question ne porte pas sur la définition de `c` ni sur la question des conditions aux limites. Elle peut sembler étrange, mais la boucle ci-dessus est l'erreur la plus fréquente.

4) a) On pourra choisir dans un premier temps $f(u) = \pm u^2/2$ et

$$u_0(x) = \begin{cases} u_L & \text{si} \quad x < 0.5 \\ u_R & \text{si} \quad x > 0.5 \end{cases}$$

avec $(u_L, u_R) = (-1, 2)$ puis $(u_L, u_R) = (2, -1)$ et des conditions aux limites numériques de type Neumann, et ensuite

$$u_0(x) = \sin(2\pi x)$$

et des conditions aux limites périodiques.
Le temps final sera choisi de manière *adhoc* de telle sorte que les ondes aient le temps de se former sans totalement sortir du domaine.
b) On pourra choisir dans un second temps une fonction flux ni convexe ni concave, par exemple $f(u) = u^3$, et

$$u_0(x) = \begin{cases} u_L & \text{si} \quad x < 0.5 \\ u_R & \text{si} \quad x > 0.5 \end{cases}$$

avec des états gauche et droit situés dans deux zones de convexité différente de f, par exemple $(u_L, u_R) = (2, -2)$.