Université de Montpellier **Année Universitaire 2023-2024**
Faculté des Sciences
Mathématiques

L3 : module HAX604X Analyse numérique des équations différentielles.

TP 6

Schémas aux différences finies pour l'équation de diffusion.

Ecrire une fonction `diffusion(mu, T, xmin, xmax, dx, dt)` qui calcule la solution $u(x,t)$ de l'EDP $u_t - \mu u_{xx} = 0$ avec la donnée initiale $u(x,0) = f(x)$ sur le segment $x_{min} \leq x \leq x_{max}$ pour $0 \leq t \leq T$.

La fonction `diffusion` retourne une matrice `u = u(j,n)` où $u(j,n)$ est censée approcher $u(x_j, t_n)$. La grille de points (x_j, t_n) est obtenue en discrétisant le rectangle $(x_{min}, x_{max}) \times (0,T)$ avec un pas d'espace δx et un pas de temps δt. Coder les différents schémas suivant (où l'on a noté $r = \mu\delta t/\delta x^2$.)

$$\text{explicite :} \quad u_j^{n+1} = r\,u_{j-1}^n + (1-2r)\,u_j^n + r\,u_{j+1}^n$$

$$\text{implicite :} \quad -r\,u_{j-1}^{n+1} + (1+2r)\,u_j^{n+1} - r\,u_{j+1}^{n+1} = u_j^n$$

$$\text{Crank} - \text{Nicolson :} \quad -\frac{r}{2}\,u_{j-1}^{n+1} + (1+r)\,u_j^{n+1} - \frac{r}{2}\,u_{j+1}^{n+1} = \frac{r}{2}\,u_{j-1}^n + (1-r)\,u_j^n + \frac{r}{2}\,u_{j+1}^n$$

Remarquez qu'il est nécessaire de résoudre des systèmes linéaires tridiagonaux à chaque itération pour les schémas implicites et de Crank-Nicolson. Utilisez le format `sparse` (matrice creuses) En Python`from scipy.sparse import linalg,diags` et les commandes `spdiags` en Matlab.

La fonction tracera sur une même figure les courbes $x \mapsto u(x,0)$ et $x \mapsto u(x,T)$. La condition initiale $f(x)$ pourra être de trois types différents : indicatrice $1_{(x<0)}$, un créneau $1_{(-1/2,1/2)}$ ou une cloche de Gauss $\exp(-25(x - (x_{min} + x_{max})/2)^2)$. Vous pouvez inclure la définition de la fonction $f(x)$ dans une fonction placée à la fin de la fonction principale `diffusion`.

```
 Matlab
function y=f(x)
    %indicatrice 1_(x<0)
     y = 1.*(x<0);
    %fonction creneau 1_(-1/2,1/2)
    %y =  1.*(-0.5<=x).*(x<1/2);
    % cloche de Gauss.
    % y = exp(-25*( x- (xmin + xmax)/2).^2);
    %
end
```

```
 Python
def f(x):
        #decommenter la fonction choisie.
        #fonction 1_(x<0)
         y = 1.*(x<0);
        #fonction creneau 1_(-1/2,1/2)
        # y =  1.*(-0.5<=x)*(x<1/2);
        # cloche de Gauss.
        # y = np.exp(-25*( x- (xmin + xmax)/2)**2);
         return y
```

Pour simplifier, vous prendrez des *conditions limites périodiques* au cours des itérations :

```
 Matlab
 remplacer u(0,n) par u(end,n), u(0,n+1) par u(end,n+1)
remplacer u(end +1, n) par u(1, n) , u(end +1, n+1) par u(1, n+1)
```

```
 Python si M = np.size(x)
u[-1,n] = u[M,n], u[-1,n+1] = u[M,n+1] (c'est en fait automatique en Python)
u[M +1, n] = u[0, n] , u[M +1, n+1] = u[0, n+1]
```

Comparer les résultats des différents schémas pour les paramètres $\mu = 0.5$, $T = 0.1$, $x_{min} = -1$, $x_{max} = 1$ $\delta x = 0.1$ avec les pas de temps suivants $\delta t = 0.005$, $\delta t = 0.01$ et $\delta t = 0.1$. Que se passe-t-il ?
Changer enfin μ en $\mu = -0.2$. Que constatez-vous ?