Résolution numérique d'un système à diffusion croisée

Soient $u(x,t)$ et $v(x,t)$. On cherche à résoudre numériquement le système d'EDP suivant :

$$\begin{cases} \dot{u} - \Delta(d_1 u + d_{12} uv) &= 0 \\ \dot{v} - \Delta(d_2 v + d_{21} uv) &= 0 \end{cases}$$

On synthétise les coefficients qui apparaissent dans le système dans la matrice suivante :

$$D = \begin{pmatrix} d_1 & d_{12} \\ d_{21} & d_2 \end{pmatrix}$$

On pose également :

$$A = \begin{pmatrix} a_1 & a_{12} \\ a_{21} & a_2 \end{pmatrix} := \left(\frac{\delta t}{\delta x^2}\right) D$$

Implémentation du schéma d'Euler implicite

On discrétise u et v en les u_k^n et v_k^n pour $k \in \{0, \dots, K-1\}$ (espace) et pour $n \in \{0, \dots, N-1\}$ (temps).
Étant donné les $(U_k^-)_k := (u_k^{n-1})_k$ à un instant n donné, on souhaite déterminer les $(U_k)_k := (u_k^n)$ à l'instant n suivant (on en déduira un procédé similaire pour les v_k^n par symétrie).
Ces données sont liées par les équations suivantes, obtenues en discrétisant le système d'EDP :

$$\begin{aligned} \forall k \in \{1, \dots, K-2\}, \qquad \frac{U_k - U_k^-}{\delta t} = & \quad \frac{d_1}{2\delta x^2}\left(U_{k+1} - 2U_k + U_{k-1}\right) \\ & + \quad \frac{d_{12}}{2\delta x^2}\left([UV]_{k+1} - 2[UV]_k + [UV]_{k-1}\right) \end{aligned}$$

Qui se ramènent à :

$$-\left(a_1 + a_2 V_{k-1}\right) U_{k-1} + \left(1 + 2a_1 + 2a_{12} V_k\right) U_k - \left(a_1 + a_2 V_{k+1}\right) U_{k+1} = U_k^-$$

Auxquelles on rajoute deux équations pour les conditions de Neumann (dérivée nulle aux bords) :

$$U_0 = U_2 \quad \text{et} \quad U_{K-3} = U_{K-1}$$

Pour simplifier les calculs ultérieurs et regrouper toutes les équations précédentes, on définit la matrice :

$$\begin{pmatrix} 1 & -1 & 0 & & & 0 \\ -(c_1 + c_2 W_0) & (1 + 2c_1 + 2c_2 W_1) & -(c_1 + c_2 W_2) & & & \\ & \ddots & \ddots & & \ddots & \\ & & -(c_1 + c_2 W_{K-3}) & (1 + 2c_1 + 2c_2 W_{K-2}) & & -(c_1 + c_2 W_{K-1}) \\ 0 & & 0 & 1 & & -1 \end{pmatrix}$$

que l'on note $M(c_1, c_2, W)$. On s'est donc ramené à résoudre :

$$M(a_1, a_{12}, V) \cdot U = U^-$$

Et par symétrie on obtient une équation analogue en échangeant U et V puis 1 et 2 :

$$M(a_2, a_{21}, U) \cdot V = V^-$$

On introduit $X = U \vee V$ (concaténation) et on se ramène à résoudre :

$$g_{X^-}(X) = 0 \qquad \text{où } g_{X^-}(X) := \begin{pmatrix} M(a_1, a_{12}, V)U - U^- \\ M(a_2, a_{21}, U)V - V^- \end{pmatrix}$$

Il s'agit en fait de trouver le zéro d'une fonction non linéaire en les coefficients de X.

Méthode de Newton-Raphson pour trouver ce zéro

Pour ce faire, on utilise la méthode de Newton-Raphson multivariée en linéarisant l'équation avec la formule de Taylor (en posant $X = X^- + \delta X$) :

$$g_{X^-}(X_{\ell+1}) = g_{X^-}(X_\ell) + J_{g_{X^-}}(X_\ell) \cdot \delta X + \mathcal{O}(||\delta X||^2)$$

Il ne reste à initialiser $X_0 = X^-$ puis à résoudre le système linéaire suivant en δX connaissant X_ℓ :

$$J_{g_{X^-}}(X_\ell) \cdot \delta X = -g_{X^-}(X_\ell)$$

Puis on met à chaque fois à jour $X_{\ell+1} \leftarrow X_\ell + \delta X$ avant de recommencer.
On stoppe la méthode de Newton dès lors que $\|\delta X\| < \varepsilon_r$ où ε_r est un seuil de précision à définir, et l'on retient alors la dernière valeur de X calculée comme celle à l'instant $n+1$. On stoppe la simulation lorsque $\|\delta X\|$ dépasse éventuellement un seuil ε_s, ce qui semble indiquer une instabilité numérique.
Pour implémenter un tel algorithme, il reste toutefois à calculer $J_g(X)$.

$$\begin{aligned} J_g(X) &= \left(\frac{\partial g_i}{\partial x_j}\right)_{1\leq i,j\leq 2K} \\ &= \begin{pmatrix} J_U(M(a_1,a_{12},V)\cdot U) & J_V(M(a_1,a_{12},V)\cdot U) \\ J_U(M(a_2,a_{21},U)\cdot V) & J_V M(a_2,a_{21},U)\cdot V \end{pmatrix} \\ &= \begin{pmatrix} M(a_1,a_2,V) & J_V(M(a_1,a_2,V))\cdot U \\ J_U(M(b_1,b_2,U))\cdot V & M(b_1,b_2,U) \end{pmatrix} \\ &=: \begin{pmatrix} P & Q \\ R & S \end{pmatrix} \end{aligned}$$

Par symétrie, il reste à savoir calculer Q. On constate que cette matrice peut être construite en calculant $M(0,a_{12},U)$ et en annulant les coefficients de la première ligne et de la dernière ligne.
On constate que P, Q, R, S sont des matrices tridiagonales.

Une dernière optimisation

Pour des raisons d'efficacité, on « intercale » alors en pratique U et V dans un nouveau vecteur $X = (U_0, V_0, U_1, V_1, \ldots)$, ce qui nous amène à adapter J_g sous la forme d'une grande matrice bande à 7 diagonales :

$$J_g \leftarrow \begin{pmatrix} a_{11} & b_{11} & a_{12} & b_{12} & & & \\ c_{11} & d_{11} & c_{12} & d_{12} & \ddots & & \\ a_{21} & b_{21} & a_{22} & b_{22} & \ddots & \ddots & \\ c_{21} & d_{21} & c_{22} & d_{22} & \ddots & \ddots & \\ & \ddots & a_{32} & b_{32} & \ddots & \ddots & \\ 0 & & \ddots & \ddots & \ddots & \ddots & \ddots \end{pmatrix}$$

Ceci nous permet de ne stocker que les diagonales de cette grande matrice en faisant appel au module *scipy.sparse* et d'avoir accès à des méthodes plus rapides pour résoudre le système linéaire.
Le choix de conditions aux bords périodiques auraient nécessité l'ajout de termes aux extrémités supérieure droite et inférieure gauche de cette matrice, rendant impossible une telle optimisation.