Cours de modélisation numérique océan-atmosphère
2ème édition – Brest – Novembre 2015

TP #1: advection

1 Introduction

Ce TP a pour but d'explorer un des éléments centraux des modèles: l'équation de transport (aussi appelée équation d'advection) sur le cas 2D

$$\partial_t \phi + \mathbf{u}.\nabla\phi = 0 \tag{1}$$

où $\phi(x, y, t)$ est une concentration de traceur et $\mathbf{u}(x, y) = (u, v)$ un champ de vitesse 2D à divergence nulle $\nabla \cdot \mathbf{u} = 0$. Le code utilise une grille de type C. (1) est codé en formulation flux $\mathbf{u}.\nabla\phi$ est remplacé par $\nabla \cdot (\mathbf{u}\phi)$. Le code fait le choix de traiter séparemment chacune des dérivées partielles: ∂_t avec un certain schéma en temps, ∂_x et ∂_y avec chacun une discrétisation spatiale. Il s'agit de la technique du *dimensional splitting*. Notez qu'il existe des schémas traitant en même temps toutes les dérivées partielles, comme par exemple le schéma de Lax-Wendroff (utilisé dans le modèle océanique MARS par exemple).

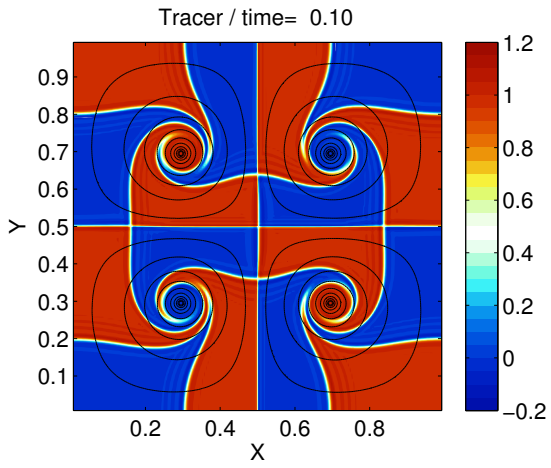

Figure 1: Exemple d'advection: advection d'un dallage par une fonction de courant quadripolaire

Discrétisation On note $\phi_{i,j}$ la valeur moyenne de $\phi(x, y)$ sur la maille (i, j). Nous adoptons donc une vision en **volumes finis**, par opposition en

une vision en différences finies ou $\phi_{i,j}$ serait la valeur de $\phi(x,y)$ au point (x_i, y_j). On note $u_{i,j}$ la composante u au point U à droite et $v_{i,j}$ composante v au point V en haut.

Schéma en temps On note s^n l'état du modèle à l'instant t_n, en l'occurrence $s = (u, v, \phi)$, et Δt le pas de temps. Le schéma en temps consiste à déterminer s^{n+1} connaissant s^n (l'état à l'instant présent) et éventuellement s^{n-1} pour certains schémas (Leap-Frog, Adams-Bashforth etc.). On note $\mathsf{L}[s]$ le second membre

$$\partial_t s = \mathsf{L}[s]\,. \tag{2}$$

Lorsqu'on sélectionne l'advection comme équation du code alors

$$\mathsf{L}[s] = (0, 0, -\nabla(\mathbf{u}\phi))\,.$$

Le code propose les schémas suivants

Euler avant le plus simple de tous. Il est toujours instable sauf lorsqu'il est combiné avec l'upwind 1er ordre mais il est alors très dissipatif. Il est cependant largement utilisé pour d'autres termes (la diffusion par exemple)

$$s^{n+1} = s^n + \Delta t \mathsf{L}[s^n]$$

Heun faisant partie de la famille des schémas de Runge-Kutta. Heun est un schéma à deux pas de temps, encore appelé predicteur-correcteur

$$\begin{aligned} s^{(1)} &= s^n + \Delta t \mathsf{L}[s^n] \\ s^{n+1} &= s^n + \frac{1}{2}\Delta t(\mathsf{L}[s^n] + \mathsf{L}[s^{(1)}]) \end{aligned}$$

RK3 est un schéma de Runge-Kutta à trois niveaux. Il en existe plusieurs celui (SSP-RK3) ne possède que des coefficients positifs et possède de meilleures propriétés de dissipation [*Gottlieb et al.*, 2001]

$$\begin{aligned} s^{(1)} &= s^n + \Delta t \mathsf{L}[s^n] \\ s^{(2)} &= s^n + \frac{1}{4}\Delta t(\mathsf{L}[s^n] + \mathsf{L}[s^{(1)}]) \\ s^{n+1} &= s^n + \frac{1}{6}\Delta t(\mathsf{L}[s^n] + 4\mathsf{L}[s^{(2)}] + \mathsf{L}[s^{(1)}]) \end{aligned} \tag{3}$$

Leap-Frog est un schéma encore largement utilisé (NEMO par ex). Il a le mérite d'être du deuxième ordre en ne calculant qu'une fois $\mathsf{L}[s^n]$

$$s^{n+1} = s^{n-1} + 2\Delta t \mathsf{L}[s^n]$$

Il est presque tout le temps couplé à un filtre d'Asselin venant modifier s^n une fois que s^{n+1} est calculé

$$s^n \leftarrow s^n + \frac{\nu}{2}(s^{n-1} - 2s^n + s^{n+1})$$

où ν est un paramètre réglant l'intensité du filtre. Une valeur usuelle est $\nu = 0.05$. Le filtre d'Asselin permet d'éliminer le bruit numérique (découplage des pas de temps pairs et impairs) mais se paye au prix d'un excès de dissipation. Une amélioration de ce filtre a été apportée par *Williams* [2009] mais elle réduit le CFL.

LF-AM3 est le schéma du modèle ROMS pour la partie barocline de l'écoulement [*Shchepetkin and McWilliams*, 2009]. Il est de type predicteur-correcteur. Il consiste en un pas de temps Leap-Frog corrigé par la deuxième étape d'un schéma Adams-Moulton.

$$s^{(1)} = s^{n-1} + 2\Delta t \mathsf{L}[s^n]$$
$$s^{n+1} = s^n + \Delta t \left[\left(\frac{1}{2} - \gamma \right) \mathsf{L}[s^{(1)}] + \left(\frac{1}{2} + 2\gamma \right) \mathsf{L}[s^n] - \gamma \mathsf{L}[s^{n-1}] \right]$$

avec $\gamma = 1/12$.

Les modèles du domaine OA continuent d'utiliser des schémas à deux étapes du type prédicteur-correcteur. Les schémas RK encore peu utilisés permettent de monter très facilement en ordre et donc en précision [*Baldauf*, 2008].

Voici un récapitulatif des schémas

nom	ordre
Euler avant	1
Heun-RK2	2
SSP-RK3	3
Leap-Frog	2
LF-AM3	3

Table 1: Schémas en temps

Schémas en espace La divergence de flux est codée selon

$$\nabla(\mathbf{u}\phi) \to \frac{1}{\Delta x}\delta_i F^x + \frac{1}{\Delta y}\delta_j F^y$$

où $F^x_{i,j}$ est le flux $u\phi$ selon x au point U et $F^y_{i,j}$ le flux $v\phi$ selon y au point V. Pour calculer le flux, disons $F^x_{i,j}$ on multiplie $u_{i,j}$ avec une interpolation

de ϕ au point U. Soit I_x la matrice d'interpolation. L'interpolation peut être centrée ou décentrée. Lorsque l'interpolation est décentrée, il y a deux matrices possibles: l'interpolation à gauche (I_x^+) et l'interpolation à droite (I_x^-). Aucune des deux n'est supérieure. En fait il faut utiliser l'interpolation upwind (upstream pour de l'eau): I_x^+ si $u > 0$ et I_x^- si $u < 0$. Le flux s'écrit donc

$$F_{i,j}^x = u_{i,j}.\mathsf{I}_x\phi|_{i,j}$$

pour un flux centré et

$$F_{i,j}^x = u_{i,j}^+.\mathsf{I}_x^+\phi|_{i,j} + u_{i,j}^-.\mathsf{I}_x^-\phi|_{i,j}$$

pour un flux upwindé avec $u^+ = \max(u, 0)$ et $u^- = \min(u, 0)$. Le tableau récapitule les différents stencils possibles.

nom	$\phi_{i-2,j}$	$\phi_{i-1,j}$	$\phi_{i,j}$	$\phi_{i+1,j}$	$\phi_{i+2,j}$	$\phi_{i+3,j}$
up1			1			
ce2			1/2	1/2		
up3		-1/6	5/6	1/3		
ce4		-1/12	7/12	7/12	-1/12	
up5	1/30	-13/60	47/60	9/20	-1/20	

Table 2: Stencils for interpolation of $\phi_{i,j}$ at point $U_{i,j}$ using finite volume values at T points, where 'up' stands for upwind and 'ce' for centered. $u > 0$ is assumed. Point $U_{i,j}$ corresponds to point $\phi_{i+1/2,j}$.

En pratique le code vous permet de tester les combinaisons indiquées dans le tableau 3.

	up1	ce2	up3	ce4	up5
Euler	dissipatif	instable	instable	instable	instable
Heun-RK2	absurde				
SSP-RK3					
Leap-Frog					
LF-AM3					

Table 3: Combinaisons possibles et leur propriété majeure (dissipatif, dispersif, instable). Certaines combinaisons sont instables, d'autres sont absurdes: ordres en espace et temps trop différents. A vous de compléter le reste du tableau.

2 Consignes pratiques

Vous lancer le code avec le script `tp1_advection.m` . Voici ce que vous pouvez changer

param.config change la géométrie et l'écoulement

param.tracerpattern change $\phi(x, y)$ à $t = 0$

param.timestepping change le schéma en temps

param.upwind change le schéma en espace

param.fixed_dt flag activant un pas de temps constant (1) ou un pas de temps variable (0)

param.cfl choisit le pas de temps via le contrôle de λ

param.dt pas de temps

param.nx change la résolution

Pour arrêter le code en cours d'exécution appuyer sur la touche `q` dans la fenêtre de visualisation. Si vous utiliser `Ctrl-C` vous aurez un problème d'affichage lors de votre prochaine expérience (faire `close all` dans ce cas).

3 Propriétés numériques

3.1 Stabilité

Les schémas en temps étant tous **explicites** ils ont une zone de stabilité. Soit λ le paramètre CFL

$$\lambda = \frac{u_m \Delta t}{\Delta x} \tag{4}$$

avec u_m la vitesse max dans le domaine. Alors soit le schéma est toujours instable, soit il est stable pour $\lambda < C$, avec $C = \mathcal{O}(1)$ une constante dépendant de la combinaison schémas temps/espace. Lorsque (4) n'est pas satisfait l'intégration explose (diverge): les quantités se mettent à croitre indéfiniment. Il s'agit d'une instabilité numérique, elle se manifeste généralement au démarrage par des oscillations à l'échelle de la maille dont l'amplitude croit avec le temps.

Votre mission : Explorer différentes combinaisons du tableau 3 et déterminer empiriquement C pour chaque combinaison. En pratique, cela

consiste à trouver le plus grand Δt permettant une intégration longue sans explosion.

3.2 Conservation

On montre que les schémas en flux conservent le traceur par construction. Concrètement

$$\partial_t \sum_{i,j} \phi_{i,j} \, \Delta x \Delta y = 0$$

Votre mission : il suffit de regarder la quantité `tracer` dans la ligne de commande, elle indique la quantité intégrale. Voyez-vous pourquoi la formulation flux garantit la conservation ?

3.3 Couplage des directions

Votre mission : travailler avec l'écoulement uniforme incliné. Vous pouvez changer l'angle (vers la fin du script). L'avantage d'avoir un schéma multi-pas de temps (e.g. predicteur-correcteur) par rapport à un schéma un pas de temps (LF) est que les flux diagonaux sont bien pris en compte dans le premier cas, pas dans le deuxième. Comprenez-vous pourquoi?

3.4 Dispersion vs. Dissipation

On explore les notions de reversibilité, création d'extrema (positivité), dissipation de variance.

Votre mission : Transporter un patch par la fonction de courant de votre choix. Observez les différences majeures entre les schémas upwind et les schémas centrés. Observer que la seule combinaison capable de garantir la positivité est Euler-avant/upwind ordre 1. Si vous voulez garantir cette propriété, par exemple empêcher d'avoir une salinité négative, empêcher d'avoir une humidité négative alors il vous faudra requérir à un schéma nonlinéaire. Tous les schémas données ici sont linéaires.

4 Propriétés physiques

On étudie les propriétés de quelques fonctions de courants classiques

4.1 Rotation solide vs. vortex

Cette partie vous illustre les différences entre une rotation solide: toutes les parcelles tournent en bloc (avec une vitesse angulaire Ω et une rotation de type vortex. Dans le premier cas `param.config=4` la vorticité de l'écoulement est uniforme $\zeta = 2\Omega$, dans le deuxième cas `param.config=8` la vorticité est à support compact, localisée dans le vortex et nulle à l'extérieur.

Votre mission : Observer comment un patch isolé évolue. Dans le cas de la rotation solide, activer un pas de temps fixe, régler le pas de temps pour que le code soit stable et arrangez vous pour réaliser une rotation complète et que le temps final soit exactement 1 (`time=param.tend=1`). Comparer alors l'état final avec l'initial (script `plot_final_vs_initial.m`).

4.2 Cisaillement

Un écoulement cisaillé correspond à $u(x, y) = y$.

Votre mission : Observer avec la distribution `param.tracerpattern='tiles'` la différence avec un écoulement uniforme.

4.3 Déformation

La fonction de courant d'une pure déformation est $\psi(x, y) = xy$ correspondant à $u = -x$, $v = y$. En pratique à cause de la finitude du domaine cela n'est pas simple à réaliser. L'ingrédient important est la structure des lignes de courants au voisinage de 0. On approchera cette configuration avec une distribution de vorticité quadrupolaire (4 vortex de signes alternés disposés au sommet d'un carré) `param.config=7` (Fig. 1).

Votre mission : Observer comment la présence de **séparatrices**, i.e. de lignes de courant se comportant comme des barrières. Pour cela ajuster les conditions initiales pour faire ressortir l'effet. Combien de régions distinctes y a-t-il? Observer comment l'étirement (*stirring)* conduit au mélange (*mixing*).

4.4 Ecoulement chaotique

Regarder comment un champ de vitesses un tout petit peu compliqué `param.config=6` conduit rapidement à une distribution de traceur extrêmement compliquée avec notamment des directions où le traceur est comprimé, d'autres où le traceur est étiré.

Votre mission : Observer comment un traceur initialement localisé finit réparti partout. Jouer sur la position du patch initial. Observer le rôle clef

des séparatrices dans les propriétés finales. Observer comment le *stirring* augmente globalement le *mixing*.

4.5 Advection d'une image

Le code peut advecter n'importe quelle concentration initiale. Jusqu'à présent nous avons pris des formes très simples mais rien ne nous empêche de transporter les teintes de gris d'une image.

Votre mission : Utiliser le `param.tracerpattern='bunny'` et observer comment l'image se transforme.

5 Conclusion

L'advection est une des briques élémentaires d'un code fluide. La résolution numérique de ce problème est un sujet de recherche actuel très dynamique. Avec ce TP vous avez pu vous familiariser avec les notions de bases qu'on retrouve dans tous les codes océan/atmosphère de notre communauté.

References

Baldauf, M., Stability analysis for linear discretisations of the advection equation with runge–kutta time integration, *Journal of Computational Physics*, *227*(13), 6638–6659, 2008.

Gottlieb, S., C.-W. Shu, and E. Tadmor, Strong stability-preserving high-order time discretization methods, *SIAM review*, *43*(1), 89–112, 2001.

Shchepetkin, A. F., and J. C. McWilliams, Computational kernel algorithms for fine-scale, multi-process, long-time oceanic simulations, *Handbook of Numerical Analysis: Computational Methods for the Atmosphere and Oceans*, *119*(182), 01,202–0, 2009.

Williams, P. D., A proposed modification to the robert-asselin time filter*, *Monthly Weather Review*, *137*(8), 2538–2546, 2009.