U.E : Analyse mathématique pour les fluides incompressibles.

Master Mathematiques et Applications :

M2 CEPS.

T.P : Vorticité.

Enseignant :
M. MEHRENBERGER

Etudiant :
FALL Madou

Aix*Marseille université
Socialement engagée

introduction

L'advection est le processus de transport d'une quantité (par exemple, la température, la salinité) par un fluide en mouvement. Dans ce contexte, la routine d'advection fournie est une implémentation d'une méthode d'advection 2D en utilisant des coordonnées polaires. Cette méthode permet de déplacer des quantités (représentées par la variable "rho") dans un domaine bidimensionnel en utilisant des champs de vitesse (représentés par les variables "Ax" et "Ay"). Le code inclut également des fonctions pour calculer le champ de potentiel et ses composantes de champ électrique à partir du champ scalaire en utilisant une méthode pseudo-spectrale. Le code implémente deux schémas numériques pour résoudre l'équation d'advection :
- `Schéma d'Euler` : C'est un schéma simple et efficace du point de vue du calcul, mais il peut être imprécis pour des vitesses d'advection élevées ou des champs d'écoulement complexes.
- `Schéma Runge-Kutta d'ordre 2` : C'est un schéma plus précis que le schéma d'Euler, mais il nécessite plus de calculs par pas de temps.

Le code `Python` fourni est une routine qui effectue des calculs en utilisant la bibliothèque NumPy, cette routine commence par initialiser les paramètres nécessaires, tels que le nombre de points dans les directions radiale et angulaire, ainsi que les pas de discrétisation. Ensuite, elle parcourt tous les points du domaine et effectue les calculs d'advection en utilisant les schémas numériques choisis.

La routine se compose de plusieurs fonctions, chacune ayant un objectif spécifique :

1. `compute_i0_r(foot, rmax, N)` :
Calcule l'indice entier `i0` correspondant à la position radiale `foot` dans une grille de taille `N` et de rayon `rmax`. Il effectue une division entière pour obtenir l'indice initial et calcule la partie fractionnaire restante (`alpha`).

2. `compute_alpha_r(foot, rmax, N)` :
Similaire à `compute_i0_r`, mais renvoie directement la partie fractionnaire (`alpha`).

3. `compute_lag(x, d, lag)` : Calcule les coefficients du polynôme de Lagrange (`lag`) de degré `d` pour une valeur donnée `x`. Les polynômes de Lagrange sont utilisés pour l'interpolation dans un intervalle spécifique.

4. `compute_i0_bis(foot, xmin, xmax, N)` :
Similaire à `compute_i0_r`, mais conçu pour une grille linéaire générale avec des valeurs minimales `xmin` et maximales `xmax`.

5. `compute_alpha_bis(foot, xmin, xmax, N)` :
Similaire à `compute_alpha_r`, mais pour la grille linéaire générale définie dans `compute_i0_bis`.

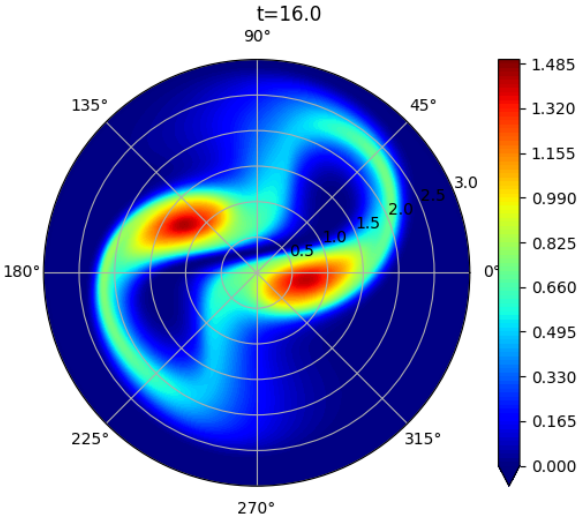
6. `adv_polar_loc_compute(x, y, fin, rmax, Nr, degr, degth, Nth)` :
La fonction principale pour le calcul de l'advection à un emplacement spécifique (`x, y`) dans la grille polaire. Elle utilise les fonctions précédemment définies pour trouver les points de grille voisins dans les directions radiale et angulaire en fonction du champ d'advection (`fin`). Elle utilise des approximations polynomiales (fonctions de Lagrange) pour effectuer ces interpolations.

7. `inv_map_polar(x, y)` :
Convertit les coordonnées cartésiennes (`x, y`) en coordonnées polaires (rayon, angle).

8. `advect2d(rho, Ax, Ay, dt, rmax, scheme, degr, degth, res)` :
La fonction principale qui effectue le calcul d'advection sur l'ensemble du tableau `2D rho`. Elle parcourt chaque point de grille (`i, j`) dans `rho`, `Ax` et `Ay` représentent les vitesses d'advection dans les directions `x` et `y` respectivement, `dt` est le pas de temps pour l'advection, `scheme` spécifie le schéma d'advection (1 pour Eulérien, 2 pour Runge-Kutta d'ordre 2), `degr` et `degth` sont à nouveau les paramètres contrôlant la taille du voisinage pour les calculs d'advection. La fonction calcule la valeur advectée à chaque point de grille en utilisant `adv_polar_loc_compute` et stocke le résultat dans le tableau de sortie `res`.

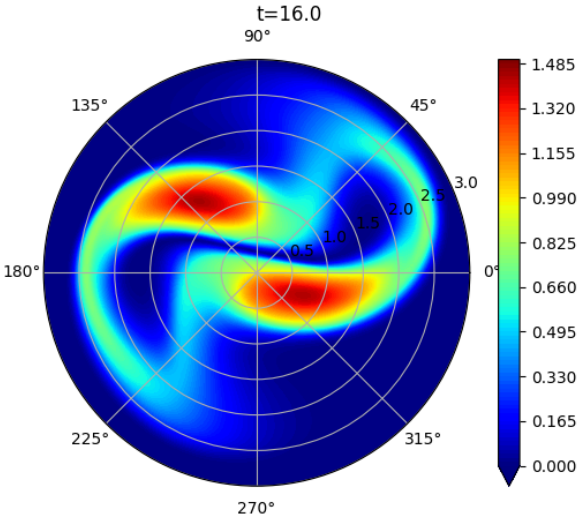
En résumé, le code implémente une méthode aux différences finies pour résoudre l'équation d'advection dans une grille polaire `2D`. Il calcule comment une quantité est transportée par le champ d'advection (`Ax, Ay`) à l'intérieur du domaine circulaire.

Nous testons la routine avec le schémas numérique Euler, on remarque que le code s'exécute rapidement mais reste moins précis, comme le montre la figure ci-dessus. Le fait que le code s'exécute rapidement avec le schéma d'Euler est attendu, car ce schéma est simple à implémenter et ne nécessite qu'une seule évaluation de la fonction source par pas de temps.



Cependant en utilisant la routine avec le schémas numérique Runge-Kutta d'ordre 2, on constate que le code s'execute très lentement mais elle est plus précise que le schémas prècèdent, comme le montre la figure ci-dessus.
Le schéma Runge-Kutta d'ordre 2 est une méthode numérique populaire utilisée pour résoudre des équations, il offre un bon compromis entre précision et complexité par rapport à d'autre méthodes

plus simples. Il s'agit d'une méthode itérative qui avance la solution de l'équation différentielle par petits pas de temps. À chaque pas, il utilise deux évaluations de la fonction source pour obtenir une estimation plus précise de la dérivée. Néanmoins, l'utilisation du schéma numérique Runge-Kutta d'ordre 2 s'avère plus précise que les schémas précédents, mais au prix d'une exécution beaucoup plus lente. Le temps d'exécution est multiplié par 5 environ avec ce schéma. Le choix du schéma numérique dépendra donc d'un compromis entre précision et performance, en fonction des besoins spécifiques de l'application.



Conclusion :

Le choix du schéma numérique dépend de nos besoins. Si la rapidité d'exécution est la priorité absolue, le schéma d'Euler peut être un choix acceptable, mais il est important de s'assurer que la précision est suffisante pour votre application. Si la précision est plus importante, il est préférable d'utiliser un schéma plus précis comme le schéma Runge-Kutta d'ordre 2. Ce code fournit un cadre pour simuler des problèmes d'advection-diffusion en coordonnées polaires. En choisissant le schéma numérique approprié (Euler ou Runge-Kutta) et en ajustant des paramètres comme la résolution de la grille et le pas de temps, les utilisateurs peuvent atteindre l'équilibre souhaité entre précision et efficacité de calcul. Le code montre également comment calculer le champ de potentiel et les composantes du champ électrique à partir du champ scalaire, ce qui peut être utile pour des analyses plus approfondies en électromagnétisme ou dans d'autres applications.