TP « Transfert de rayonnement » - Ecole d'Aussois 2009

Frédéric Paletou, U. Toulouse, CNRS, OMP/LATT

Général

Dans ce TP nous aborderons le cas élémentaire, mais incontournable, de la résolution du problème de diffusion monochromatique hors-ETL dans une couche 1D plan parallèle semi-infinie. Pour ce cas, il s'agit de résoudre les équations suivantes (cf. Mihalas 1978, ou bien le cours en ligne de R. Rutten @ http://www.astro.uu.nl/~rutten/Lecture_notes.html) :

- $J=\Lambda[S]$
- $S=(1-\varepsilon)J + \varepsilon B$

où $\Lambda[S]$ représente la solution formelle de l'équation de transfert i.e., la solution J pour S connue, B la fonction de Planck (par la suite normalisée à 1 et constante dans la couche), et ε la probabilité de destruction collisionnelle, quantité caractérisant l'écart à l'ETL et liée aussi à l'albedo par la relation suivante : $a=(1-\varepsilon)$.

L'intérêt de ce cas réside dans le fait qu'il admet une solution analytique, assez facile à dériver, *en se plaçant dans l'approximation d'Eddington.*

Nous proposons d'utiliser diverses méthodes, de l'inversion directe du système linéaire ***AS=b*** que l'on peut déduire des équations précédentes, à la méthode – à déconseiller – de la Λ-itération, ou méthode de Picard, puis celle dite « ALI » ou méthode de Jacobi pour le transfert hors-ETL, et enfin les méthodes de Gauss-Seidel et *Successive Over-Relaxation* (SOR).

Nous utiliserons diverses ressources écrites en Python :

- `RT.py` pour le/les formal solvers,
- `RTdirect.py` pour la résolution directe par inversion de A,
- `li.py` pour la Λ-itération,
- `ali.py` pour la méthode ALI,
- `gs.py` pour les méthodes de Gauss-Seidel ou SOR.

Les paramètres d'entrée seront :

- `taumax` : épaisseur optique totale de la couche
- `npdec` : nombre de points par décade dans la grille en opacité
- `eps` : le paramètre de destruction collisionnelle
- `niter` : le nombre d'itérations (sauf pour la méthode directe)

Premier exercice : inversion directe

Après avoir dérivé la solution d'Eddington (`sedd`), analyser et utiliser le programme `RTdirect.py`. On appellera par la suite T_e le maximum de l'erreur relative entre la solution (`s`) et la solution de référence (`sedd`).

Pour que la comparaison entre la solution numérique et `sedd` *fasse sens, il faut se placer systématiquement dans des conditions de couche « effectivement épaisse » i.e., dans le cas où* `taumax >> 1/sqrt(eps)` *dans le cas de la diffusion monochromatique (cf. notion de longueur de thermalisation).*

Deuxième exercice : Λ-itération (trop) simple...

Ou pourquoi faut-il proscrire la Λ-itération ! Utiliser le programme `li.py` avec la contrainte précédente, et pour diverses valeurs de ε.

On constatera le caractère « pseudo-convergent » de la Λ-itération, caractérisée, sauf dans certains cas, par une décroissance rapide de l'erreur relative d'une itération à l'autre (`relerr`) mais pour un processus itératif conduisant à une « solution » numérique qui peut être différente de plusieurs ordres de grandeur de la solution de référence.

Troisième exercice : Λ-itération accélérée ou ALI

A l'origine, la méthode ALI était basée sur l'utilisation d'un opérateur approché (de l'opérateur complet Λ) pas nécessairement diagonal. Dans un article *importantissime* pour le transfert de rayonnement numérique, Olson, Auer et Buchler (OAB : 1986, JQSRT 35, 431) ont montré que le schéma optimal est celui utilisant la diagonale exacte de Λ.

Nous utiliserons la version ALI-OAB qui est une méthode de Jacobi (`ali.py`).

- Pour un même cas (`taumax`, `npdec`, `eps`) mais en ajustant les valeurs de `niter` pour que T_e (ALI) atteigne un plateau, comparer les résultats de `li.py` et de `ali.py`.

- Pour un même cas (`taumax`, `eps`), comparer les valeurs « à la convergence » de T_e (ALI) en fonction du raffinement de la grille spatiale e.g., `npdec`= 5, 8, 11 ..., 20. On constatera en même temps l'effet de `npdec` sur le taux de convergence de la méthode (`niter` pour atteindre une valeur constante de T_e ?).

Quatrième exercice : méthodes GS/SOR

Enfin, nous allons utiliser les méthodes de Gauss-Seidel (GS) et *Successive Over-Relaxation* (SOR), et les comparer à la méthode de ALI-Jacobi. Le programme (`gs.py`) est identique pour les 2 méthodes : GS fonctionne avec ω=1 tandis que SOR fonctionne avec un facteur de sur-relaxation ω compris entre 1 et 2.

- Comparer GS et SOR avec ω=1.5
- Essayer d'autres valeurs de ω comprises entre 1 et 2 et comparer à GS.

Bibliographie

Des références importantes sont précisées dans les commentaires du module `RT.py` sous la forme de liens ADS.