ECINADS-D3.1

RENUMEROTATION DE ILU

LEMMA/INRIA

Table des matières

1 Renumérotation

Dans ce travail, nous envisageons des calculs de mécanique des fluides sur des maillages très anisotropes dans les couches limites. L'usage de ces maillages entraine des problèmes de convergence des systèmes à résoudre. Pour améliorer la convergence, une renumérotation de type "Linelet" a été implémentée dans le code Ananas. Cette renumérotation des nœuds du maillage, couplée avec un préconditionneur ILU et un solveur numérique comme la méthode itérative GMRES, nous permettra d'accélérer la résolution numérique utilisant les éléments finis.

1.1 GMRES et préconditionneur ILU

La plupart des méthodes itératives utilisées pour la résolution de grands systèmes linéaires utilisent le procédé de la projection matricielle sur un sous-espace. L'idée est d'extraire une solution approximative $\tilde{u}$ du problème $A.u = f$ dans un sous-espace de R^N.

Supposons que A est une matrice réelle $(N * N)$ et que K et L soient deux sous-espaces de R^N de dimension m. Une technique de projection sur le sous-espace orthogonal à K et L est un procédé qui trouve une solution $\tilde{u}$ approximative pour $Au = f$, en imposant les conditions que $\tilde{u}$ appartient à K et que le nouveau vecteur résiduel est orthogonal à L, c'est-à-dire :

$$\text{trouver } \tilde{u} \in K \text{ tel que } f - A\tilde{u} \perp L.$$

GMRES est un cas particulier de méthode de projection, fondé sur le fait que $K = K_m$ et $L = AK_M$, où K_m est le sous-espace m-ème de Krylov. Le sous-espace de Krylov d'ordre m : $K_m(A, r_0)$, généré par une matrice A $(N * N)$ et un vecteur r_0 de dimension N, est par définition le sous-espace vectoriel engendré par :

$$K_m(A, r_0) = vect(r_0, Ar_0, A^2r_0, ..., A^{m-1}r_0),$$

En général, la convergence des méthodes itératives dépend des propriétés spectrales de la matrice. En effet, la vitesse de convergence d'une méthode itérative dépend significativement du conditionnement de la matrice. Pour améliorer le taux de convergence, nous pouvons appliquer un préconditionnement.
Le préconditionnement est un moyen simple de transformer le système linéaire de départ en un autre sytème équivalent, mais mieux conditionné, et donc susceptible d'être plus facile à résoudre avec le solveur itératif choisi.
Si nous avons une matrice M qui se rapproche des coefficients de la matrice A d'une certaine façon, le système transformé :

$$M^{-1}Au = M^{-1}f,$$

a la même solution que le système d'origine $A.u = f$, mais les propriétés spectrales de la matrice $M^{-1}A$ peuvent être plus favorables.
Nous appelons la matrice M préconditionneur. En pratique, le calcul du produit matrice-vecteur $M^{-1}.x$ devrait être le moins coûteux possible lorsqu'on l'applique à un vecteur arbitraire x.

Une des façons les plus simples de définir un préconditionneur consiste à effectuer une factorisation "Incomplete Lower Upper" (ILU) de la matrice originale A.
Ceci entraîne une décomposition de la forme $A = LU - R$ où L et U sont des matrices respectivement triangulaires inférieure et supérieure de A, et R est le résidu ou l'erreur de la factorisation.

Cette factorisation incomplète connue sous le nom "ILU (0)" est assez facile et peu coûteuse à effectuer.
Considérons une matrice creuse A dont les éléments sont $a_{i,j}$, $i, j = 1, .., n$. Une factorisation ILU consiste à calculer une matrice creuse triangulaire inférieure L et une matrice creuse triangulaire inférieur U de sorte que la matrice résiduelle $R = LU - A$ réponde à certaines contraintes, comme ayant des coefficients nuls.

Un algorithme général pour la construction de la factorisations ILU peut être obtenu en effectuant une élimination de Gauss et en délaissement certains éléments non diagonaux.
L' algorithme de calcul ILU pour une matrice A de dimension $(N*N)$ est donné par :

$for\ r := 1\ step1\ until\ n-1\ do$
$d := 1/a_{rr}$
$for\ i := (r+1)\ step1\ until\ n\ do$
$if\ (i,r) \in S\ then$
$e := d_{ai,r}$
$a_{i,r} := e$
$for\ j := (r+1)\ step1\ until\ n\ do$
$if\ ((i,j) \in S)\ and\ ((r,j) \in S)\ then$
$a_{i,j} := a_{i,j} - ea_{r,j}$
$end\ if$
$end(j - loop)$
$endif$
$end(i - loop)$
$end(r - loop)$

Ici, S représente l'ensemble des éléments de la matrice A.

Il existe différents types de préconditionneurs ILU. La méthode de factorisation ILU sans "fill-in", que l'on peut traduire par sans "remplissage", désignée par ILU(0),

consiste à annuler dans les matrices L et U les coefficients qui se trouve être nuls dans la matrice A.

Le préconditionneur ILU(0) préserve donc la structure de la matrice originale dans son résultat. L'utilisation de la factorisation ILU(0) comme préconditionneur est très fréquente lors des résolutions de systèmes linéaires pour des calculs de simulations de flux, de part son efficacité et ses besoins en mémoire modérées.

1.2 Influence de la renumérotation

En discrétisant le domaine du problème (EDP) que l'on considère, nous allons créer un maillage de telle manière que nous aurons une "bonne" matrice A , c'est-à-dire que la matrice A sera facile à traiter afin d'obtenir la solution du problème. Les paramètres qui ont effet sur le conditionnement de la matrice A sont le maillage et la numérotation des noeuds.

Le conditionnement d'une matrice positive se définit comme :

$$K(A) = \frac{\sigma_{max}(A)}{\sigma_{min}(A)},$$

où $\sigma_{max}(A)$ et $\sigma_{min}(A)$ représentent donc respectivement les valeurs propres maximale et minimale de la matrice A.

En analyse numérique, le conditionnement mesure la dépendance de la solution d'un problème numérique par rapport aux données du problèmes, ceci afin de contrôler la validité d'une solution calculée par rapport à ces données. En effet, les données d'un problème numérique dépendent en général de mesures expérimentales et sont donc entachées d'erreur.
De façon plus générale, on peut dire que le nombre de conditionnement associé à un problème est une mesure de la difficulté de résolution numérique du problème. Un problème possédant un nombre de conditionnement bas est dit bien conditionné et un problème possédant un nombre de conditionnement élevé est dit mal conditionné.

Une autre difficulté liée au calcul numérique du problème est associè à la structure interne de la matrice, plus précisément à la largeur de bande de la matrice.
On définit la largeur de bande basse d'une matrice A comme le plus petit entier p tel que les entrées a_{ij} sont nulles pour $i > j + p$. De même, on définit la largeur de bande haute comme le plus petit entier p tel que les entrées a_{ij} sont nulles pour $i < j + p$.
Par exemple une matrice tridiagonale à une largeur de bande basse de 1 et une largeur de bande haute de 1.
Les matrices avec des petites largeurs de bande haute et basse sont nommées des

matrices bande et des algorithmes plus efficaces que ceux sur les matrices creuses existent. Une renumérotation des entrées suffit à changer ces caractéristiques.
Par exemple l'algorithme de Cuthill–McKee permet de réduire la largeur de bande d'une matrice creuse et symétrique, et de nombreuses autres méthodes existent pour réduire cette largeur de bande. Le problème de la réduction de la largeur de bande d'une matrice consiste à trouver une permutation des lignes et des colonnes qui maintient les éléments non nuls dans une bande qui soit aussi proche que possible de la diagonale principale de la matrice. Avec une plus courte largeur de bande, la matrice sera plus facilement traités par le solveur numérique employé (GC, GMRES,..).

Afin d'éviter une grande largeur de bande de la matrice, nous allons choisir de manière appropriée une renumérotation des noeuds du maillage, qui influeront directement sur la structure interne de la matrice.
Il existe deux méthodes de numérotation des noeuds d'un maillage : l'ordre lexicographique et l'ordre lexicographique orthogonal.
Pour une renumérotation des noeuds avec un ordre lexicographique, nous utilisons la formulation suivante :

$$k = i + (j-1)i_{max} \text{ avec } u_k = u_{i,j}, \text{ où } u_k \equiv a_{i,j}.$$

En appliquant cette formule pour la renumérotation des noeuds, nous allons obtenir une numérotation des noeuds comme la figure (13) et une matrice tridiagonale de même forme que la figure (14). L'application d'une méthode ILU pour une telle matrice nous conduit à une solution exacte de notre système linéaire.

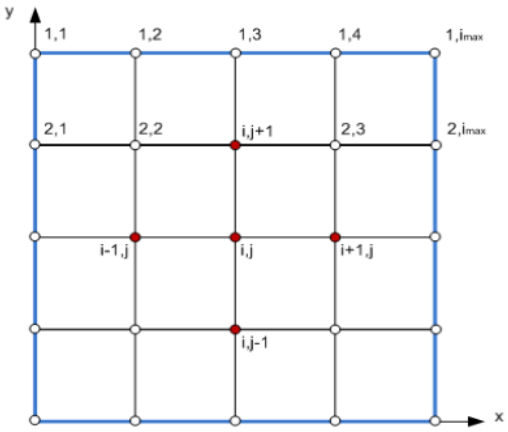

Fig. 1 – Ordre lexicographique de numérotation des noeuds .

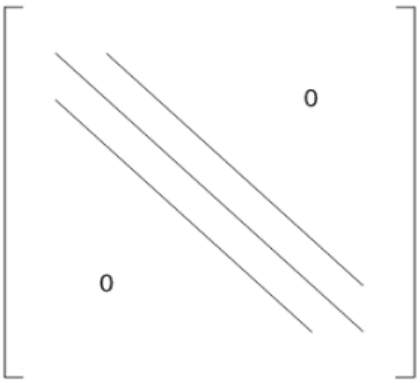

FIG. 2 – Matrice générée par une numérotation lexicographique.

Pour une numérotation des nœuds avec un ordre lexicographique orthogonal, nous utiliserons la formule suivante :

$$k = j + (i-1)j_{max} \text{ avec } u_k = u_{i,j}, \text{ où } u_k \equiv a_{i,j}.$$

En appliquant cette formule pour la renumérotation des noeuds, nous allons obtenir une numérotation des noeuds comme la figure (15) et une matrice de même structure que la figure (16), qui a une largeur de bande égale à $i_{max} - 1$. L'application d'une méthode ILU pour une telle matrice ne nous conduira pas à une solution exacte de notre système linéaire, mais plutôt à une solution approchée d'assez mauvaise qualité.

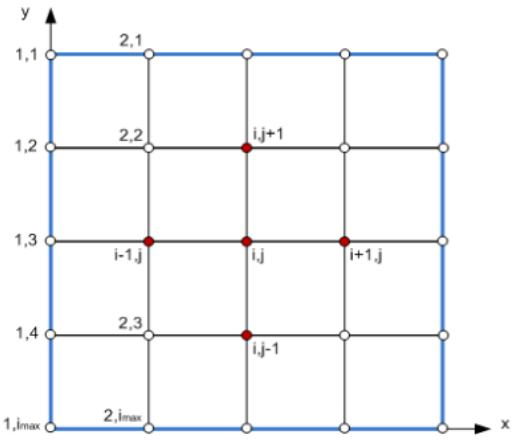

FIG. 3 – Ordre lexicographique orthogonal de numérotation des noeuds .

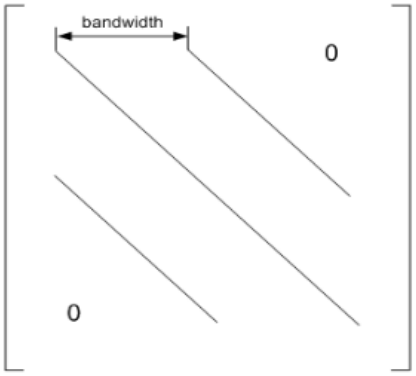

FIG. 4 – Matrice générée par une numérotation lexicographique.

1.3 Algorithme de renumérotation par *linelet*

Afin d'obtenir un bon préconditionneur, la construction d'un ensemble approprié de *linelet* est cruciale. L'idée est de commencer avec un ensemble de points source dans le maillage. Pour chaque point source i, l'arrête (i, j), reliant le point source i avec un point voisin j, qui aura la longueur la plus petite est choisie comme direction de la *linelet* .
Le nouvel ensemble de point j ainsi construit deviendra alors la nouvelle série de points sources. La procédure est répétée jusqu'à ce qu'on ne puisse plus créer de it linelet .

Une description plus détaillée de la méthodologie est présentée ci-dessous. Nous allons tout d'abord décrire la méthode générale de Soto O., Löhner R. et Camelli F. [14], puis nous décrirons par la suite l'algorithme de renumérotation par *linelet* que nous avons créé pendant ce travail et implémenté dans le code, qui diffère légèrement.

– Première étape : Choisir l'ensemble dse points sources des *linelet* , c'est-à-dire les points de départ.

  Cette tâche se fait en marquant les points entourés par des singularités du maillage, c'est-à-dire les points pour lesquels le rapport entre l'arrête la plus courte et la plus longue qui les relient à leurs voisins, est inférieure à une valeur donnée.
  Ensuite, les points sources appartenant à la même *linelet* sont éliminés comme suit : pour chaque point source i, l'arrête de longueur minimale qui le relie au point j, ainsi que l'arrête de longueur minimale qui le relie le point j à un point k, sont trouvées.
  Ensuite, deux situations peuvent se produire : si k et i sont différents, i est éliminé parce que l'arrête (j, k) est plus petite que l'arrête (i, j), et donc la *linelet* commence probablement par k et passe par j et i.

La deuxième option est quand i et k sont au même point, alors le point source peut être i ou j. Dans ce cas, le point où l'arrête avec le plus proche voisin suivant est la plus grande doit être éliminé.

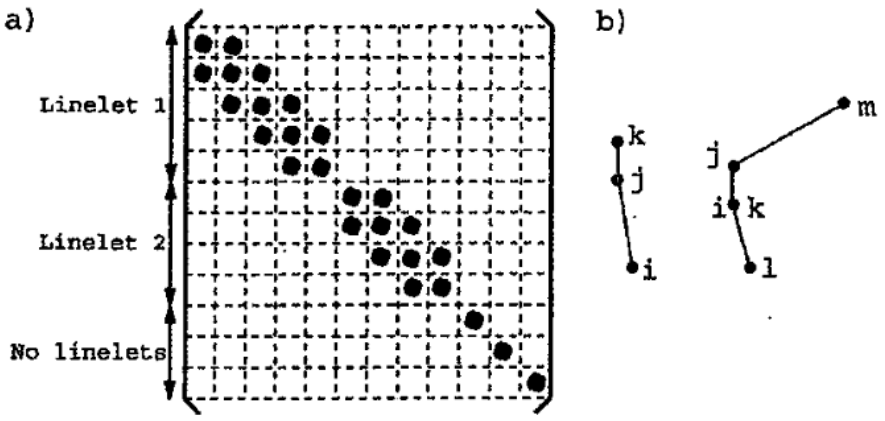

FIG. 5 – (a) structure de la matrice préconditionneur et (b) orientation du processus d'élimination de points sources.

– Deuxième étape : Créer les *linelet* .

Pour chaque point i marqué comme point d'origine, son plus proche voisin j qui n'appartient pas à une autre *linelet* et qui remplit la condition :

$$\frac{l_{ij}}{l_i^m} \leq \epsilon$$

est choisi. Ici, l_{ij} désigne la longueur de l'arrête (i, j), et l_i^m la longueur maximale des arrêtes entourant i. j sera le point suivant de la *linelet* correspondante et le nouveau point source de la *linelet* .
Ensuite, toutes les *linelet* sont mises à jour, c'est-à-dire cette étape est répétée, en prenant comme nouveaux points d'origine les points j. L'algorithme s'arrête lorsqu'aucune *linelet* supplémentaire ne peut être créée.

– Troisième étape : Création dans l'autre sens.

On répéte la deuxième étape, mais à partir du deuxième voisin le proche des points d'origine des *linelet* . Ceci est à prendre en compte dans les régions où les *linelet* peuvent se développer dans les deux directions.

– Quatrième étapes : Renumérotation des nœuds.

Les noeuds doivent être renumérotés suivant la direction des *linelet* . Ceci est crucial pour obtenir la structure désirée tri-diagonale de la matrice.
Tout d'abord, les nœuds d'une *linelet* sont renumérotés d'un bout à l'autre de la *linelet* . Ensuite, une seconde *linelet* est renumérotée , et ainsi de suite jusqu'à ce que toutes les *linelet* aient été recouvertes. Enfin, les points restant qui ne sont pas sur des *linelet* sont numérotés aléatoirement (dans la pratique, ils seront très peu nombreux).

Passons maintenant à notre algorithme de renumérotation. Nous allons décrire comment se déroule chaque étapes :

1.Repérage des nœuds frontières :

Pour toutes les faces f_i de tous les tétraèdres du maillage,
si f_i est une face de logique +-2 (c'est-à-dire une face adhérente),
alors les trois nœuds de la face f_i sont marqués comme appartenant à la couche limite 1. On numérote ensuite tous les points marqués comme appartenant à la couche limite 1.

2.Calcul des distances entre les noeuds :

On fixe une longueur maximale des linelets (longueur en nombre de noeuds consécutifs qui formeront les *linelet*). L'algorithme se compose de deux grandes boucles :
Pour une longueur de *linelet* de 1 à longueur max, faire :
Pour tous les points de la couche limite, faire :
On calcule la distance entre tous le points de la couche 1 et tous les points voisins, c'est-à-dire les points reliés par une arrête.
On obtient alors les distances minimales qui relient les points de la couche limite 1 aux points les plus proches.

3.Trouver les voisins :

Maintenant que l'on a les distances minimales, on trouve le voisin de le plus proche pour chaque nœud de la couche limite 1.
Ensuite, avec tous les voisins de la première couche, nous avons formé une deuxième couche de point, qui devient la nouvelle couche limite. On recommence encore ce procédé pour trouver les voisins de chaque point de la deuxième couche, ainsi de suite...
Au final, nous avons un nombre de couches égal au maximum à la longueur max des linelets que nous avions fixée.

4.Renumérotation des nœuds des *linelet* :

Au début cette étape., nous avons tous les voisins les plus proches de chaque point avec comme point origine les nœuds de la couche limite 1.
Ensuite pour la construction des linelets, on en prend une au hasard et on renumérote les noeuds, de façon consécutive, en partant à chaque fois du nœuds de la première couche.
La renumérotation des noeuds de chaque *linelet* s'arrête lorsqu'on retombe sur un successeur qui a été déjà renuméroté dans une autre *linelet* . Puis on passe à la *linelet* suivante, jusqu'à les avoir toutes parcourues.

5.Renumérotation des noeuds en dehors des *linelet* :

Pour les nœuds du maillage qui n'appartiennent pas à une *linelet* , nous les renumérotons ensuite de façon aléatoire.

6.Mettre à jour les tableaux du maillage :

Enfin, nous mettons à jour les tableaux qui caractèrisent le maillage, c'est-à-dire les tableaux qui définissent le numéro global de chaque nœud dans le maillage, le numéro local des nœuds dans les faces des triangles ainsi que les numéros des faces des triangles.

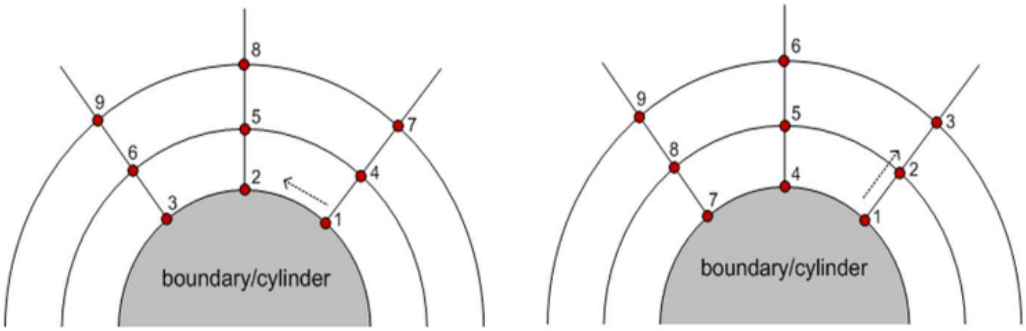

FIG. 6 – Exemple de renumérotations lexicographique (à gauche) et par *linelet* (à droite), sur un maillage cylindrique.

1.4 Comparaisons de différentes numérotations

Nous prouvons le bon fonctionnement de cet algorithme de renumérotations *linelet* dans un exemple de maillage 3D avec couche limite et très anisotrope :

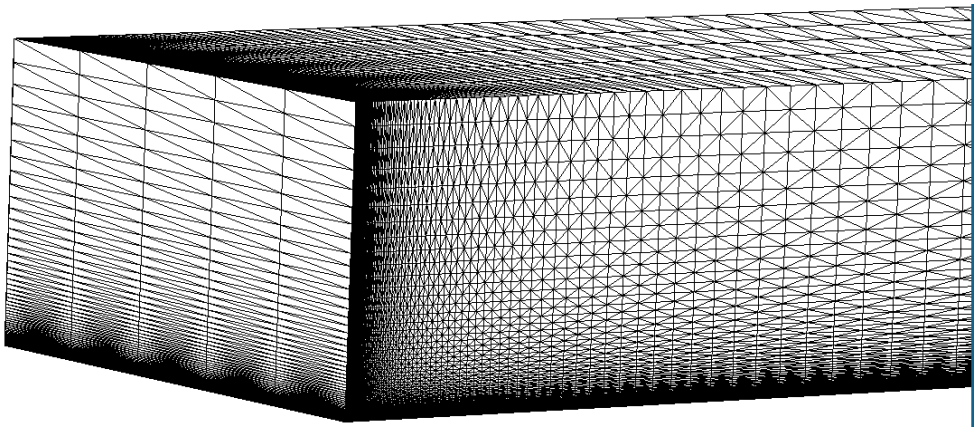

FIG. 7 – Maillage utilisé pour les test de renumérotation.

On compare l'impact des différentes renumérotations par le calcul du résidu :

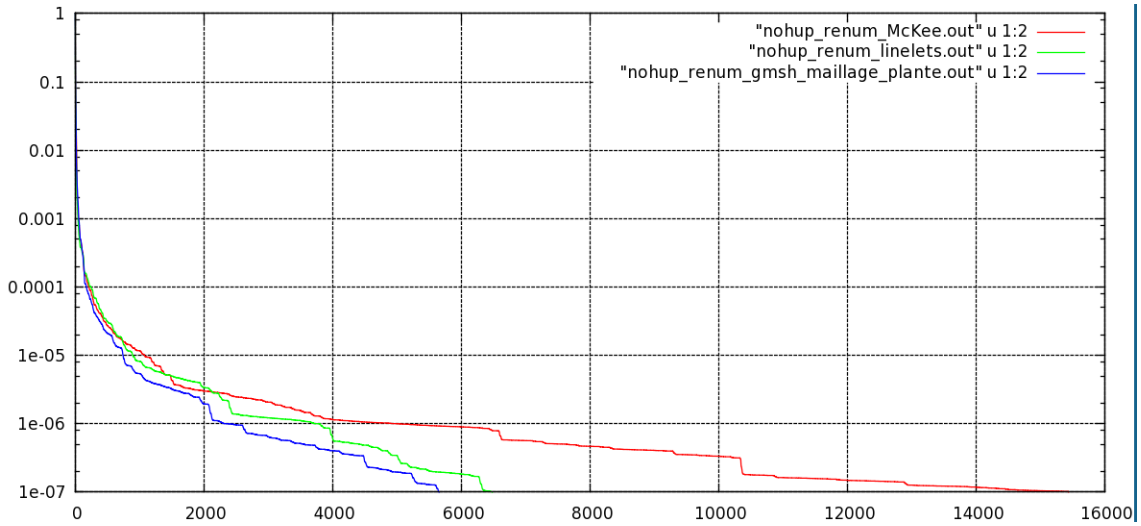

FIG. 8 – Courbes de la convergence du résidu en fonction du nombre d'itération.

Ces tests numériques ont été lancé avec une tolérance du résidu égale à 10^{-7}. La courbe bleu représente la solution optimale, qui est la numérotation du logiciel GMSH [15] où ce maillage structuré a été créé. Nous remarquons aisément que la renumérotation Cuthill MacKee implémentée dans le code ANANAS n'est pas bonne, la convergence s'effectue après un grand nombre d'itérations, tandis que notre renumérotation *linelet* est très bonne, par le fait que l'on est très proche de la numérotation optimale, qui est la renumérotation lexicographique.

2 Bibliographie

[1] Soto O., Lohner R. and Camelli F. - " A linelet preconditioner for incompressible flow solvers ", *International Journal of Numerical methodes for Heat and Fluid Flow* **13** (2003), no. 1 vol. 13, p. 133-147.