Présentation simplifiée de *Zohour* (version 2D, v. 0.9.1)

(mise-à-jour 15 juin 2019)

1. Généralités sur les raffinements de maillages.

Zohour (https://perso.univ-rennes1.fr/edouard.canot/zohour/) est un mailleur adaptatif 2D. Donnons tout de suite quelques définitions :

- *mailleur* : programme qui met en œuvre un algorithme de maillage, permettant de découper un domaine géométrique en un certain nombre de portions plus petites, appelées *mailles*. Ce découpage doit être une partition exacte : la réunion des mailles correspond exactement au domaine de départ.
- *adaptatif* : le maillage s'adapte au calcul d'un problème donné. La plupart du temps, l'adaptation est itérative. Un exemple classique concerne par exemple un maillage dont les mailles vont être d'autant plus petites que leur emplacement est proche d'un certain point, ou d'une certaine zone, du domaine initial.
- *2D* : le domaine de départ et toutes les mailles appartiennent à un espace de dimension 2.

Les maillages sont aujourd'hui utilisés dans quasiment toutes les résolutions de problèmes provenant de la discrétisation d'une ou plusieurs équations aux dérivées partielles (Différences Finies, Éléments Finis, Éléments frontières, Volumes Finis, etc. Seules les méthodes de type *meshless* – par ex. les méthodes de point vortex – n'ont pas besoin de maillage).

Parmi les formes de mailles 2D les plus utilisées, on trouve le rectangle (ou plus généralement le quadrilatère) et le triangle.

La représentation d'un maillage 2D peut être :

- *structurée* – les mailles sont repérées par deux indices *(i,j)* et sont par conséquent à 4 côtés ;
- *non structurée* – les mailles sont d'abord définies, via une *table de définition*, par les sommets qui les définissent, puis sont reliées aux autres mailles voisines par une *table de connectivité*. Les mailles ont un nombre arbitraire de côtés, mais souvent elles ont 3 côtés.

La précision de la méthode dépend de la taille des mailles, mais aussi de leur forme. Ainsi, il est connu qu'en Volumes Finis, si on calcule les flux sur les côtés de la maille par une formule simple, alors un maillage rectangulaire donne une meilleure précision qu'un maillage triangulaire quelconque.

Un maillage adaptatif répond à la question suivante : « Comment modifier un maillage afin de raffiner les mailles – c'est-à-dire augmenter la finesse de représentation – en des endroits arbitraires ? ».

Remailler tout le domaine est une possibilité (cf. figure 1), mais outre le coût d'un remaillage global, il faut aussi interpoler toutes les quantités numériques concernées par le solveur ou l'algorithme à tous les nouveaux points du maillage (on perd aussi en précision).

Une méthode ancienne (et naïve) consistait à partir d'un maillage de carrés (ou de rectangles), puis à les subdiviser en quatre, comme le montre la figure 2. Mais le maillage résultant perd une propriété fondamentale requise pour appliquer la plupart des méthodes de discrétisation : il n'est plus conforme – son seul avantage est de présenter des mailles qui sont toutes carrées.

On s'intéresse ici à la méthode de Volumes Finis de type « Node-based » : des nœuds contiennent la valeur d'une fonction *u(x,y)*. À partir de ces nœuds, on construit un maillage de type Voronoï (partition du plan 2D par l'ensemble des médiatrices de deux nœuds voisins). Si les nœuds sont répartis sur des lignes horizontales et verticales équidistantes, le diagramme de Voronoï est un maillage de carrés dont les côtés sont également horizontaux et verticaux. Un maillage de Voronoï possède une propriété très

intéressante : parce que chaque côté d'une maille appartient à la médiatrice de deux nœuds voisins, alors le calcul du *gradient de u(x,y) normal à ce côté* peut être calculé de manière directe (c'est-à-dire simple, donc efficace) et précise.

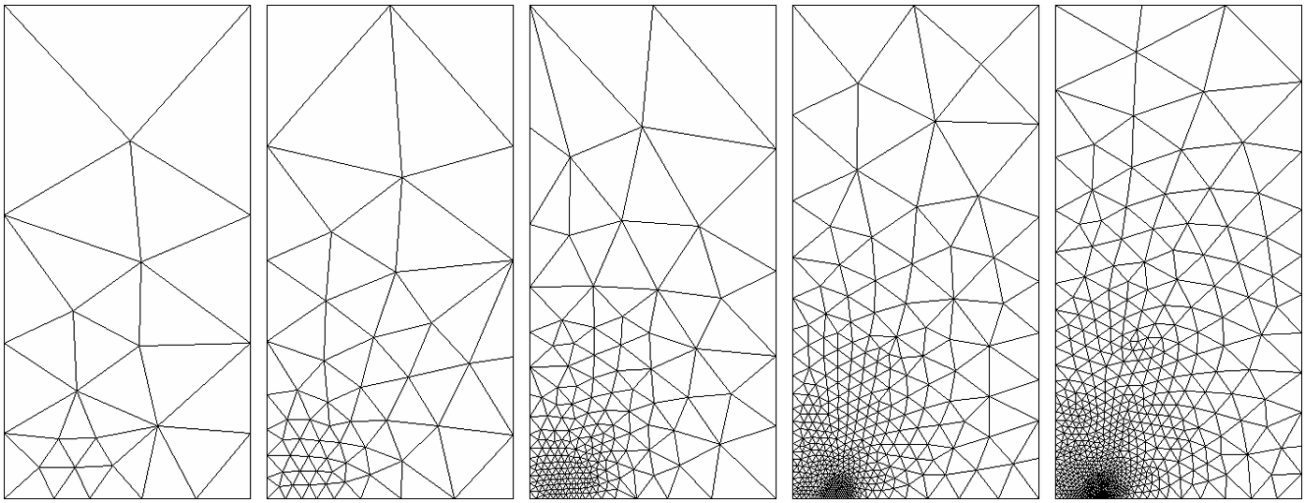

Fig. 1 : Maillages de plus en plus fins au voisinage d'une zone d'intérêt particulière. Ici, le raffinement est global, puisque l'ensemble du maillage est retouché.
(exemple tiré de Emerald Insight, internet)

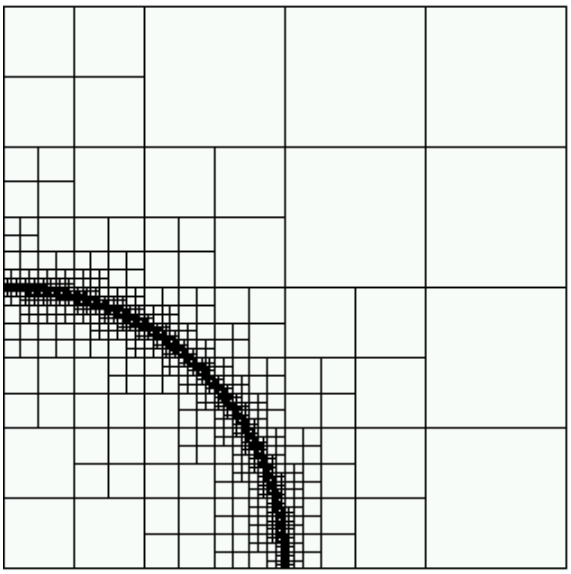

Fig. 2 : Méthode ancienne de raffinement d'un maillage de carrés
(http://www.dealii.org/developer/doxygen/deal.II/step_12.html).
Certains carrés sont découpés en quatre parties égales, de manière récursive.
On remarque que le maillage qui en résulte n'est plus conforme.

Zohour est inspiré de HOMARD (https://www.code-aster.org/V2/outils/homard/fr/intro.html), mais avec un principe différent. HOMARD (dans sa version 2D) raffine un maillage triangulaire, suivant le principe schématisé sur la figure 3. Le site mentionné ci-dessus montre de nombreux exemples.

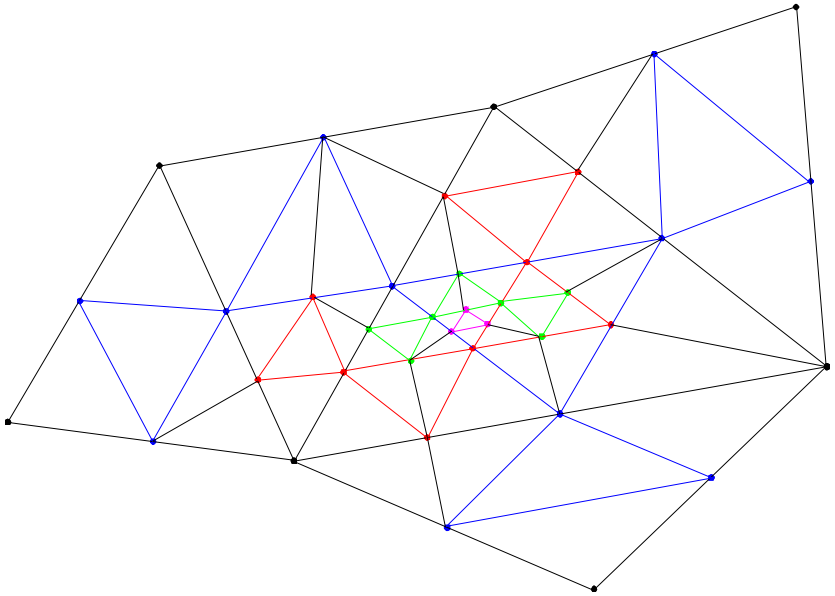

Fig. 3 : Principe de raffinement de HOMARD (EDF), dans sa version 2D triangulaire. Le maillage de base est ici dessiné en noir ; les triangles sont subdivisés en quatre (niveau bleu), en respectant la même qualité de forme. Puis, récursivement, on peut continuer d'autres niveaux de subdivisions : 2e niveau (rouge), 3e niveau (vert), 4e niveau (magenta), etc. On remarque que, pour garder le maillage conforme, il est nécessaire de couper certains triangles en deux.

Zohour partage aussi plusieurs caractéristiques avec le raffinement de Balmelli *et al.* (2002)[1], présenté sur la figure 4.

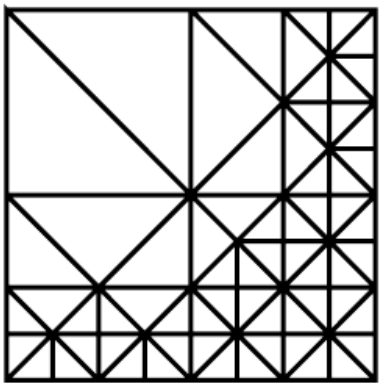

Fig. 4 : Raffinement triangulaire conforme (Balmelli *et al.*, 2002). Tous les triangles sont orientées suivant l'horizontale et la verticale, ce qui donne des avantages certains à l'implémentation informatique. Comme pour notre mailleur *Zohour*, le domaine de départ ne peut être que rectangulaire.

La spécificité de *Zohour* est de retrouver si possible des cellules carrées, et avec une bonne *progressivité* dans la taille des cellules (la plupart des algorithmes adaptatifs ont un facteur 4 en terme de surface[2], celui de Balmelli *et al.* (2002) mentionné plus haut 2 ; *Zohour* quant à lui a un facteur égal à 1.5 (voir la figure 9 en fin d'annexe) : il est donc très progressif.

1 Balmelli L., Liebling T., Vetterli M., 2002, Computational analysis of mesh simplification using global error, *Comp. Geom.: Theory and Application*, Elsevier Sciences.

2 Nous mesurons ici la progressivité en nombre de mailles par unité de surface.

2. Principe du mailleur *Zohour*.

On choisit un domaine initial *carré*, puis on répartit des nœuds suivant des lignes horizontales et verticales équidistantes, y compris la frontière. En fait, le domaine initial peut ne pas être carré, à condition toutefois que les nœuds définissent des cellules carrées.

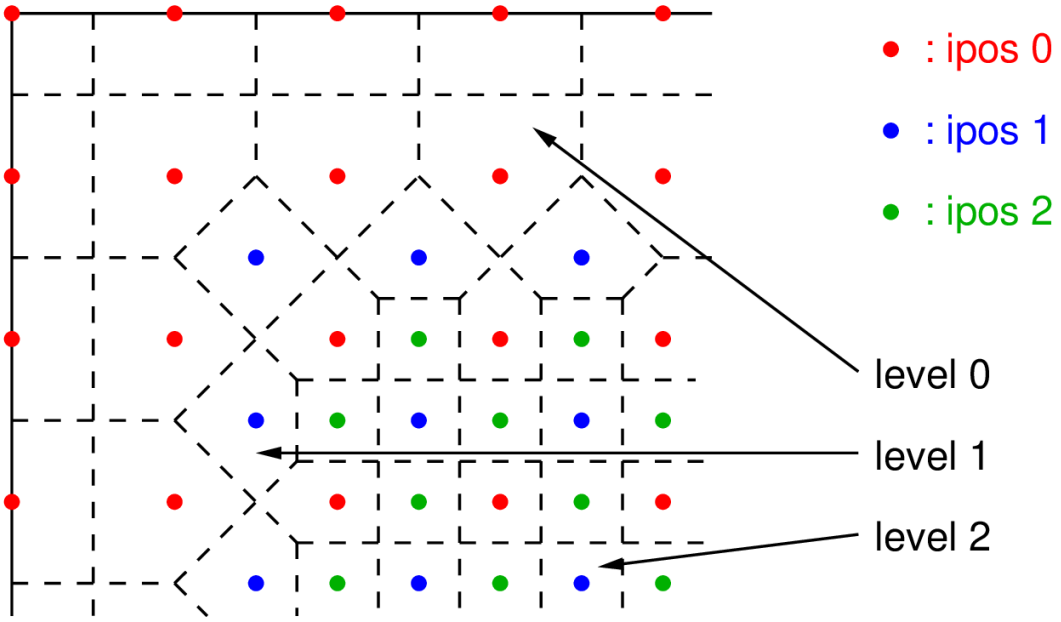

Fig. 5 : Les éléments de base de l'algorithme de *Zohour* sont les nœuds rouges (de *ipos* 0) qui sont inamovibles et répartis initialement sur des lignes équidistantes horizontales et verticales. Les nœuds de *ipos* 1 (en bleu) sont ajoutés à une intersection droite de médiatrices (en pointillées). Les nœuds de *ipos* 2 (en vert) sont ajoutés à une intersection inclinée de médiatrice, et ainsi de suite. Il faut distinguer la valeur de *ipos* de celle de *level* ; ces notions seront définies dans la section 3.

Ces nœuds de base sont appelés nœuds de *ipos* 0 (marqués en rouge sur la figure 5 ci-dessus). Ils ne peuvent en aucun cas être supprimés. Tous les autres nœuds (de *ipos* supérieur ou égal à 1) qui seront ajoutés par la suite peuvent en revanche être supprimés. Certains nœuds sont posés exactement sur la frontière (voir par exemple la figure 7).

Les nœuds ajoutés le sont par « couches successives » : d'abord les nœuds de *ipos* 1, puis ceux de *ipos* 2, etc.

Le principe de *Zohour* est d'ajouter un nouveau nœud uniquement à l'intersection (lignes en pointillés) de quatre angles droits. Au fur et à mesure qu'on raffine les couches (c'est-à-dire après ajouts de nouveaux nœuds de couches supérieurs), on voit apparaître des mailles carrées alternativement *droites* (bords horizontaux et verticaux) et *inclinées* (bords à 45° et 135°).

Les mailles, créées par le diagramme de Voronoï, à la limite d'un nouveau *ipos* peuvent prendre des formes variées. On dénombre ainsi 10 formes différentes pour les mailles internes au domaine; elles sont répertoriées dans la figure 6. Les autres mailles (qui s'appuient sur des nœuds de frontière ou des nœuds de coin), peuvent avoir des formes encore différentes : on parlera de demi-cellule pour les premières et de quart-de-cellule pour les secondes.

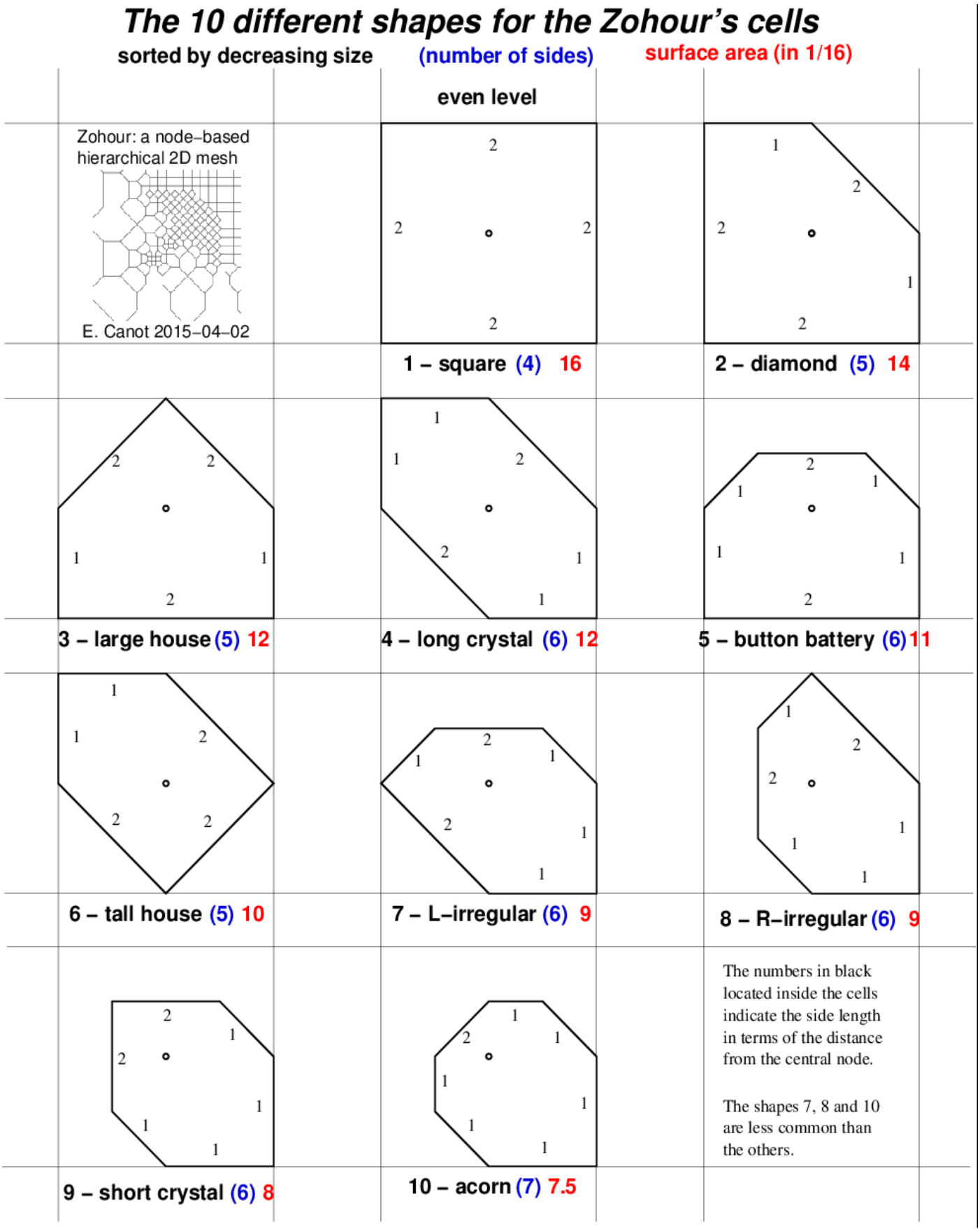

Fig. 6 : Différentes formes de mailles obtenues après des subdivisions (dues aux ajouts de nœuds supplémentaires). Le nom de chacune des formes a été choisie pour rappeler la forme du polygone. Toutes ces formes peuvent être tournées à 90°, 180° ou 270°. La même série existe inclinée à 45°.

On trouvera dans les figures 7 et 8 des exemples de maillages raffinés dans le coin inférieur gauche d'un domaine de base carré.

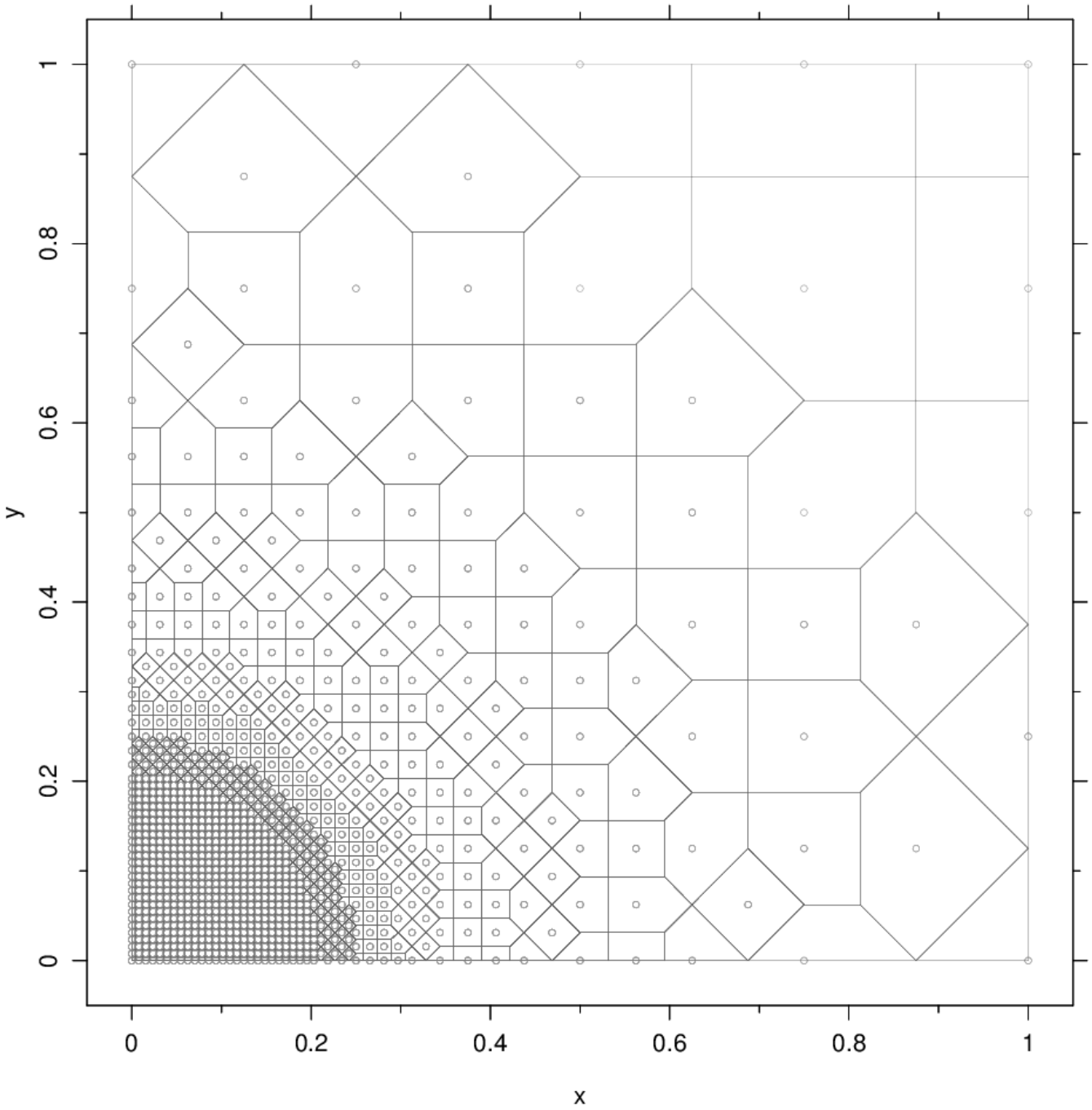

Fig. 7 : Exemple de maillage raffiné par *Zohour* au coin inférieur gauche, qui correspond à un point singulier d'une fonction harmonique. La taille des mailles est d'autant plus petite qu'on se rapproche du point singulier. Les petits ronds représentent la position du nœud interne à chaque cellule.

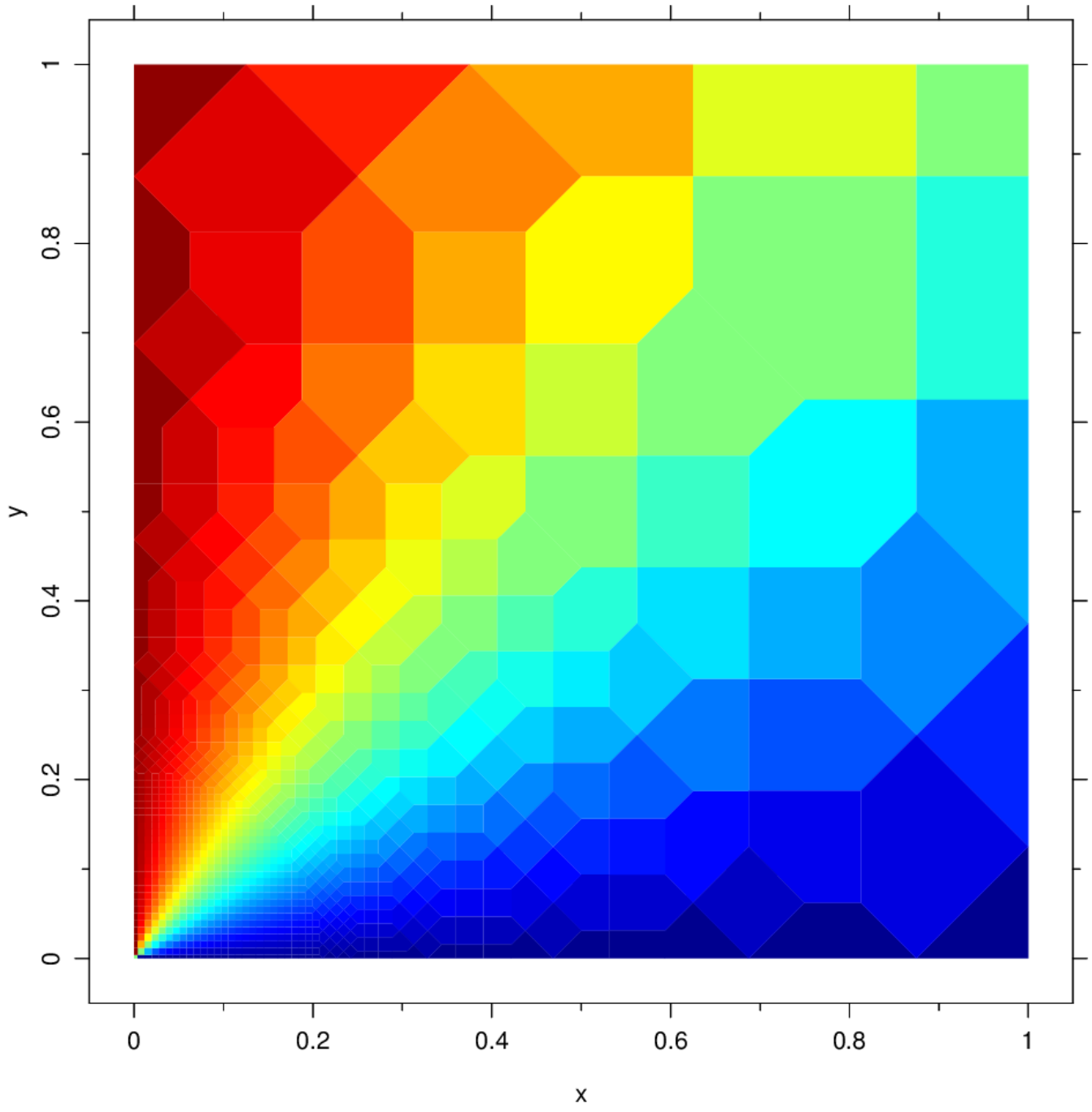

Fig. 8 : Même maillage que celui de la figure 7, utilisé ici pour représenter en couleurs la solution d'un problème de Laplace singulier au coin inférieur gauche. Conditions limites de type Dirichlet pour les côtés x=0 et y=0, mais avec une discontinuités à l'origine.

3. Implémentation de *Zohour* en Fortran 2003.

L'algorithme de *Zohour* 2D est mis en œuvre par l'intermédiaire du langage Fortran 2003. Seuls les nœuds sont réellement stockés ; pour cette raison, en pratique on confond la maille qui entoure chaque nœud avec ce dernier. On emploiera donc indifféremment « maille » et « cellule ».

```
  type, public :: cell

     integer :: id                       ! unique identifier

     integer(kind=1) :: signature(2)     ! cell shape
     double precision :: area            ! cell area

     double precision, public :: x, y    ! coords of internal node

     integer :: ipos  = 0 ! initial level at creation -- useful to check
                          ! if, after many subdivisions or merges, the node
                          ! is aligned on the level grid

     integer :: level = 0 ! current level of subdivision -- also size of the
                          ! surrounding cell, based on the more distant node,
                          ! in terms of a divided factor

     integer :: reserve = 4

     integer :: on_boundary = 0 ! 0 (interior), 1 (boundary), 2 (corner)

     ! this pointer describes the chronological order
     type(cell), pointer         :: prev => null() ! previous cell
     type(cell), pointer, public :: next => null() ! next cell

     ! these pointers describe the spatial links
     class(*), pointer :: N   => null() ! elem at North
     ! edge length in terms of the distance to the cell boundary
     integer           :: Nl  = 2

     class(*), pointer :: NE  => null() ! elem at North-East
     integer           :: NEl = 2

     class(*), pointer :: E   => null() ! elem at East
     integer           :: El  = 2

     class(*), pointer :: SE  => null() ! elem at South-East
     integer           :: SEl = 2

     class(*), pointer :: S   => null() ! elem at South
     integer           :: Sl  = 2

     class(*), pointer :: SW  => null() ! elem at South-West
     integer           :: SWl = 2

     class(*), pointer :: W   => null() ! elem at West
     integer           :: Wl  = 2

     class(*), pointer :: NW  => null() ! elem at North-West
     integer           :: NWl = 2

     ! user data
     double precision, allocatable :: data(:)

  end type
```

Listing 1 : Définition de la structure de base (type dérivé) : `type(cell)`

Le type dérivé `cell`, décrit dans le listing 1 de la page précédente, contient :

- l'identifiant `id`, qui est unique pour chaque nœud. Cet entier est strictement positif.
- la signature de la cellule (voir section suivante).
- les coordonnées `x`, `y` du nœud interne ; ce dernier n'est pas forcément au centre de masse de la cellule (voir la figure 6).
- le numéro de subdivision `ipos`, à la création du nœud. C'est une constante pour le nœud : sa valeur ne change pas au cours du raffinement/déraffinement. Un nœud ayant *ipos*=0 correspond à un nœud du maillage de base, et ne sera jamais enlevé. Cet entier est strictement positif.
- le niveau de subdivision `level`, quant à lui, est lié à la taille globale la cellule ; cet entier va changer de valeur au gré des différentes subdivisions. Cet entier est strictement positif. La parité de `level` permet de reconnaître un carré droit ou incliné à 45°.
- l'entier `reserve`, nombre de coins à 90° qui restent à « écorner » pour changer de forme. Cet entier est toujours compris entre 1 et 4. À chaque fois qu'on ajoute un nouveau nœud à l'un des coins de cette cellule, on décrémente la valeur de `reserve`. Au bout de 4 nœuds ajoutés, la cellule redevient carrée et doit changer de niveau de subdivision (`level`+1) tandis que le `reserve` est remis à la valeur 4. Bien entendu, La définition de `reserve` est différente pour les demi-cellules du bord et pour les quart de cellule des coins du domaine.
- le type de cellule (cellule d'intérieur, cellule de bord, cellule de coin).
- les pointeurs sur `type(cell)`, `prev` et `next`, servent à construire des listes doublement chaînées constituées de cellules ayant certaines caractéristiques.
- les huit pointeurs `N`, `NE`, `E`, `SE`, `S`, `SW`, `W` et `NW` correspondent à des voisins éventuels (pointeurs non nuls) aux directions correspondantes des huit points cardinaux. Ce sont des pointeurs sur `type(cell)`.
- les huit entiers `Nl`, `NEl`, `El`, `SEl`, `Sl`, `SWl`, `Wl` et `NWl` correspondent aux longueurs des côtés, exprimés en terme de la distance entre le nœud interne et le côté voisin. Ces entiers ne peuvent prendre que la valeur 1 ou 2.
- enfin un tableau allouable, `data(:)`, destiné à être manipulé par l'utilisateur pour y stocker des informations concernant certaines variables de son problème.

Un pointeur sur `class(*)` permet de pointer sur n'importe quel type, au contraire des pointeurs ordinaires qui en Fortran sont typés. On parle alors de « unlimited polymorphism » et cette caractéristique n'est compatible qu'avec la norme Fortran 2003.
Dans le cas des pointeurs vers les huit points cardinaux (`N`, `NE`, `E`, `SE`, `S`, `SW`, `W` et `NW`), la cible peut être une autre cellule voisine (`type(cell)`) si elle existe, ou une frontière de `type(boundary)`, dans le cas contraire. Ce type dérivé `boundary` contient les coefficients de l'équation de la droite-frontière et spécifie également le type de condition limite (Dirichlet, Neumann, …).

On remarquera que l'algorithme ne manipule que des entiers, ce qui évite les erreurs d'arrondis qui arrivent inévitablement quand on opère sur des flottants. Seules les coordonnées `x` et `y` sont stockées en réels flottants ; on pourrait toutefois ne pas stocker ces coordonnées et les calculer « à la demande » à partir de certaines informations entières (les nœuds d'un niveau de position *ipos* sont sur une grille calculable à l'avance).

4. Signature d'une forme pour la reconnaissance rapide des cellules.

Lors de la mise en œuvre de la méthode des Volumes Finis à partir d'un maillage de type *Zohour*, il faut connaître la *surface* de chaque cellule. Comme on connaît tous les voisins d'une cellule donnée, on pourrait construire le *polygone* qui définit exactement cette cellule, à partir des *médiatrices* entre les nœuds voisins (c'est le principe d'une partition de Voronoï), puis calculer la surface de ce polygone. Cette manière de faire a été employée dans les premières versions de *Zohour* mais conduit à un coût de calcul important.

La version actuelle s'appuie sur la *signature* des cellules, pour reconnaître rapidement la forme de celles-ci (il n'y a que 10 formes possibles au total, cf. la section précédente).

La signature d'une cellule est stockée dans le `type(cell)` (comme précisé dans le listing 1). On la calcule « à la demande ». Elle est constituée :

- de huit informations précisant si les voisins `N`, `NE`, `E`, `SE`, `S`, `SW`, `W` et `NW` existent (1) ou pas (0), soit 8 bits.
- de huit informations précisant la longueur des côtés `Nl`, `NEl`, `El`, `SEl`, `Sl`, `SWl`, `Wl` et `Nwl`. En soustrayant 1 à cette longueur (qui par définition vaut 1 ou 2), on obtient 8 bits.

En comparant la valeur calculée à des signatures de référence (après permutation circulaire éventuelle, voir ci-dessous), on peut en déduire la surface normalisée (valeur en rouge pour chaque cellule de la figure 6). Le `level` de la cellule permet alors de calculer sa surface réelle. La modification d'un bit particulier en Fortran s'effectue avec la fonction `ibset`.

La comparaison entre la signature actuelle et les signatures de référence doit tenir compte du fait que la cellule n'est pas forcément dans la même orientation que celles dessinées en figure 6. On effectue successivement plusieurs shifts sur les bits de la signature pour « tourner » virtuellement la cellule dans les 8 orientations possibles. C'est la fonction Fortran `ishftc` qui permet d'appliquer une permutation circulaire aux bits d'un entier. L'implémentation est programmée dans la routine `quick_cell_area` (10 valeurs de référence).

Le même principe concerne les cellules de bord (routine `quick_cell_area_boundary`, 16 valeurs de référence) et celles de coin (routine `quick_cell_area_corner`, 3 valeurs de référence).

5. Le calcul du hessien pour un maillage créé par *Zohour*

Cette section ne relève pas, à proprement parler, de la description du mailleur *Zohour*, mais de la mise en œuvre d'une méthode de résolution d'un problème s'appuyant sur un maillage de type *Zohour*.

Étant donné un problème, souvent décrit par un système d'équations aux dérivées partielles, on va chercher à raffiner le maillage en fonction de critères variés, dont le plus fréquent est la variation de la solution. Le hessien *H* (composé des dérivées secondes d'une fonction *f* à deux variables *x* et *y*) est à la base d'un tel critère de raffinement :

$$H=\sqrt{\left(\frac{\partial^2 f}{\partial x^2}\right)^2+2\left(\frac{\partial^2 f}{\partial x\,\partial y}\right)^2+\left(\frac{\partial^2 f}{\partial y^2}\right)^2}$$

C'est un invariant de la matrice hessienne, car il s'exprime en fonction de sa trace et de son déterminant, tous deux également invariants.

Pour une maille carrée, et en se limitant à l'opérateur laplacien, on montre que l'erreur de discrétisation pour un schéma de type Volume Fini est proportionnel à $H\,h^2$, où *h* est la longueur du côté de la cellule ; on peut donc déduire facilement un critère pour choisir la taille de la maille *h* en fonction d'une tolérance donnée à l'avance.

En général, l'estimation de *H* pour une fonction f s'appuyant sur un maillage quelconque est coûteuse. La valeur du Hessien dépend d'une part des valeurs de *f*, d'autre part de la position relative des nœuds voisins autour d'un nœud donné. On approche la fonction *f* (connue de manière discrète seulement) par la fonction quadratique suivante, à six degré de liberté :

$$f(x,y)=c_1+c_2x+c_3y+c_4x^2+c_5y^2+c_6xy$$

Il faut donc trouver au minimum 6 nœuds (en comprenant le nœud central pour lequel on veut calculer le hessien) afin d'établir un système d'équations linéaires à 6 inconnues. Parfois ces 6 nœuds sont mal

disposés (leur alignement respectif est tel que le système est singulier) ; dans ce cas on ajoute des nœuds supplémentaires et on calcule la solution par moindres carrés. Une fois les coefficients c_1 à c_6 calculés, on a alors, de manière triviale :

$$\frac{\partial^2 f}{\partial x^2}=2c_4\,;\qquad \frac{\partial^2 f}{\partial y^2}=2c_5\,;\qquad \frac{\partial^2 f}{\partial x\partial y}=c_6$$

La plupart des mailles générées par *Zohour* sont carrées (voir par exemple figure 6). Comme ces mailles sont facilement détectables grâce à notre structure de données (type dérivé `cell`), on peut faire un pré-calcul et stocké à l'avance l'inverse de la matrice 6x6 mentionnée plus haut.

En choisissant les six nœuds suivants (cas d'un carré centré à l'origine et entouré de quatre autres carrés) :

n1 (0, 0) ; n2 (0, 1) ; n3 (1, 0) ; n4 (0, –1) ; n5 (–1, 0) ; n6 (1, 1)

Dans ce cas, la matrice hessienne est la suivante :

$$A=\begin{pmatrix}1&0&0&0&0&0\\1&0&1&0&1&0\\1&1&0&1&0&0\\1&0&-1&0&1&0\\1&-1&0&1&0&0\\1&1&1&1&1&1\end{pmatrix}$$

C'est une matrice bien conditionnée (*cond* $\simeq$ 8) et nous l'avons inversée exactement (à l'aide de Maple). De plus, il suffit de conserver seulement sa partie inférieure :

$$\begin{pmatrix}\cdot&\cdot&\cdot&\cdot&\cdot&\cdot\\\cdot&\cdot&\cdot&\cdot&\cdot&\cdot\\\cdot&\cdot&\cdot&\cdot&\cdot&\cdot\\-1&0&1/2&0&1/2&0\\-1&1/2&0&1/2&0&0\\1&-1&-1&0&0&1\end{pmatrix}$$

On remarquera que le calcul sera exempt d'erreur d'arrondi, les coefficients étant 1 ou 1/2, tous deux exactement représentable en base 2. Il faut simplement introduire le facteur d'échelle de la maille (lié au niveau de subdivisions) et le hessien se calcul directement par :

$$2c_4=(-2f_1+f_3+f_5)*\mathit{facteur}$$
$$2c_5=(-2f_1+f_2+f_4)*\mathit{facteur}$$
$$c_6=(f_1-f_2-f_3+f_6)*\mathit{facteur}$$

Pour certains maillages, comme celui montré sur la figure 7, le calcul global du hessien pour l'ensemble est extrêmement rapide et on gagne un facteur allant jusqu'à 1500 par rapport au cas ordinaire où on ne fait pas ces pré-calculs. Pour donner un ordre de grandeur du temps CPU, ce dernier vaut 0.14 secondes (sur un PC de performance moyenne de 2015) pour un maillage comportant environ 1 726 000 nœuds. Ce temps est négligeable devant le temps typique pour résoudre un problème de type Laplacien sur le même maillage (en utilisant des matrices creuses et un solveur creux direct) : 10 secondes environ.

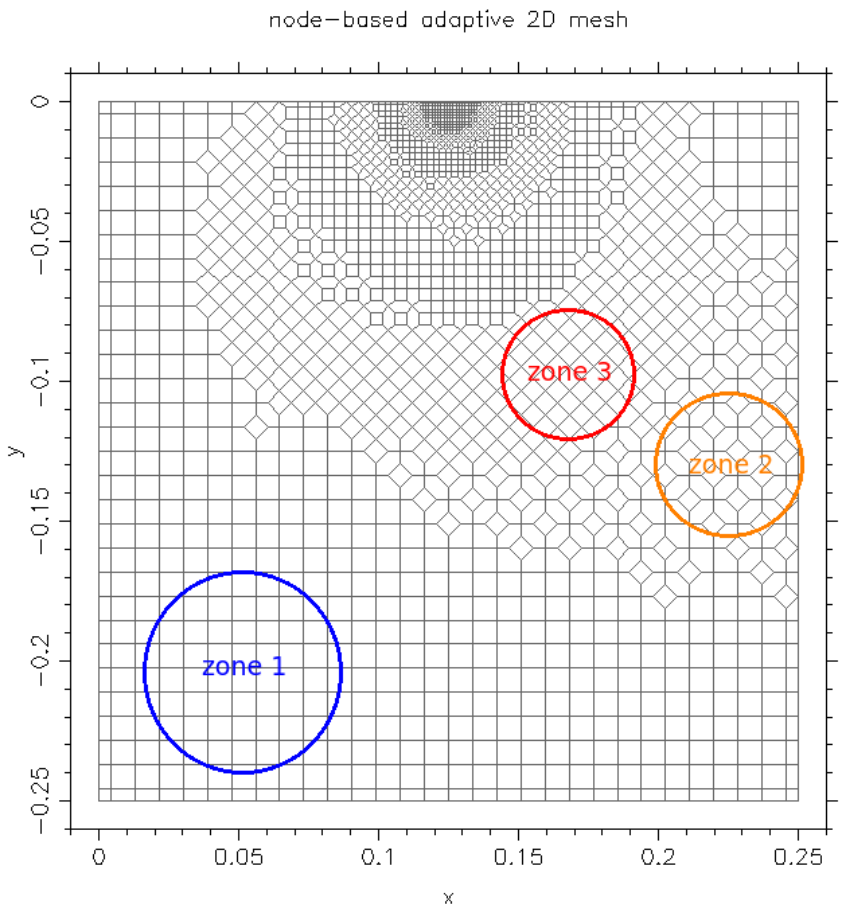

Fig. 9 : Exemple de maillage généré par *Zohour* et montrant la très bonne progressivité de la taille des mailles. Si dans la zone 1 on compte 4 mailles sur une certaine surface, alors on en compte 6 sur la même surface dans la zone 2, ce qui donne un facteur de progressivité de 3/2.