Guide d'utilisation de « toolsRef.py »

1 Table des matières

2 Définitions

2.1 Maillage

Un maillage a pour but de diviser en sections une surface ou un volume s'apparentant à un filet. Ce programme crée le maillage de toute pièce d'une surface en connaissant certaines caractéristiques renseigner lors de l'utilisation du programme.

Le maillage réel à obtenir est appelé « maillage physique » et il est obtenu à partir d'un « maillage de référence » ou aussi appelé « computational mesh », « CPU Mesh ».

2.2 Maillage de référence

2.2.1 Numérotation

Afin d'obtenir le maillage final, il est nécessaire de créer un maillage de référence correspondant à un carré divisé en « $n-1$ » sections verticales et « $m-1$ » sections horizontales. Chaque section se croisant, forment diverses géométries :

- Nœuds (Nodes)
- Eléments (Elements)
- Lignes (Edges)

Les éléments correspondent à des surfaces à 2 dimensions composés de 4 nœuds et 4 éléments. Les lignes relient deux nœuds ensemble, revenant ainsi à un corps à une dimension. Les nœuds quant à eux se trouvent aux croisements de lignes et à chaque coin d'un élément.

Le nombre d'élément, nœud et ligne sont obtenable à partir de n et m :

- Nœuds : $N = n * m$
- Eléments : $N\_el = (n-1)(m-1)$
- Lignes horizontales : $N\_ed\_hor = (n-1) * m$
- Lignes verticales : $N\_ed\_ver = n * (m-1)$
- Lignes : $N\_ed = N\_ed\_hor + N\_ed\_ver$

Une numérotation est appliquée à chaque géométries en suivant les règles ci-dessous.

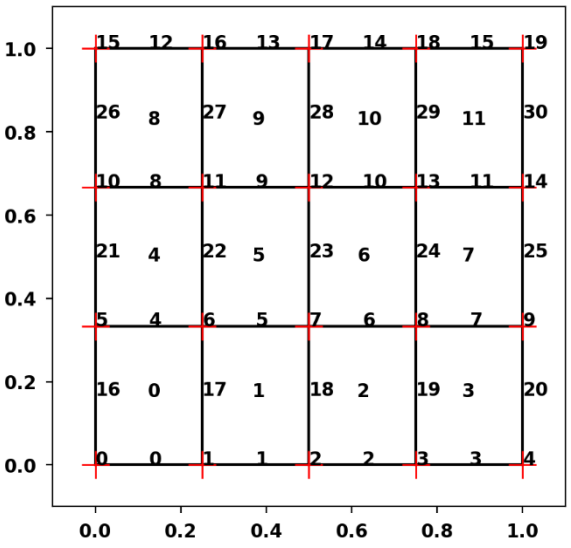


Figure 1 : Exemple de numérotation

- Nœuds :

Les nœuds sont numérotés de gauche à droite, de bas en haut. La numérotation est appliquée sur chaque horizontal en partant de 0 jusqu'à $N-1$.

- Eléments :

La numérotation des éléments suit le même principe que les nœuds : de gauche à droite, de bas en haut mais ici partant de 0 jusqu'à N_el.

- Lignes :

Il est possible de faire la distinction entre des lignes horizontales et verticales et la numérotation s'applique en premier aux lignes horizontales puis verticales. Chaque ligne horizontale est numérotée

de 0 à $N\_ed\_hor - 1$ et les lignes verticales de N_ed_hor à N_ed. Quant à l'ordre de cette numérotation, il respecte toujours la même règle, de gauche à droite et de bas en haut.

A chaque nœud est appliqués un autre type de numérotation selon les axes horizontales et verticales : (i, j), aussi appelé position. i correspond à la position horizontale du nœud alors que j correspond à la position verticale avec $i = 0,1,2 \dots n - 1$ et $j = 0,1,2 \dots m - 1$.

A partir des coordonnées (i, j), il est possible d'obtenir le numéro du nœud :

$$k = i + j * n \tag{1}$$

Et à contrario :

$$j = \left\lfloor \frac{k}{n} \right\rfloor \tag{2}$$

$$i = k - j * n \tag{3}$$

2.2.2 Exemple

Pour $n = 5$ et $m = 4$, un maillage de référence avec sa numérotation serait l'exemple au-dessus. Voici d'autres exemples :

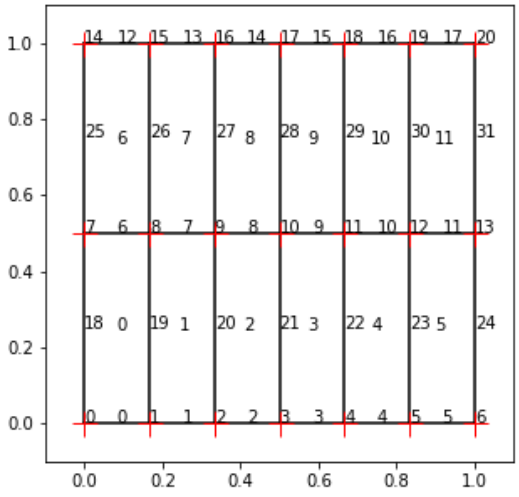


Figure 2 : $n = 7, m = 3$

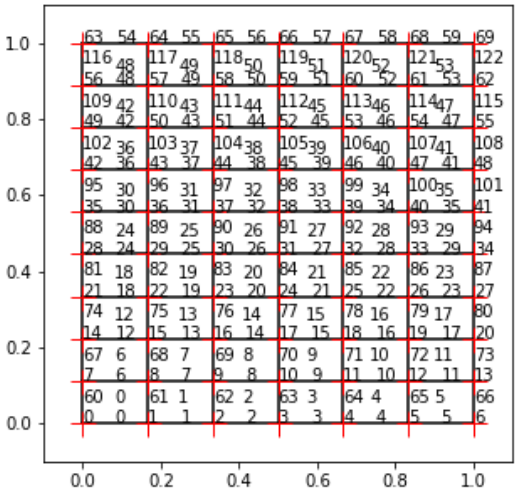


Figure 3 : $n = 7, m = 10$

Les coordonnées (i, j) de certains nœuds de $n = 7, m = 3$ sont par exemple :

- $k = 0 => (0,0)$
- $k = 12 => (5,1)$
- $k = 20 => (6,2)$
- $k = 15 => (1,2)$

Par la suite, l'exemple de $n = 5$ et $m = 4$ est couramment utilisé sur les figures où le maillage est présent.

2.2.3 Caractéristiques supplémentaires

Chaque maillage possède les mêmes dimensions suivant chaque axe, axes que nous plaçons :

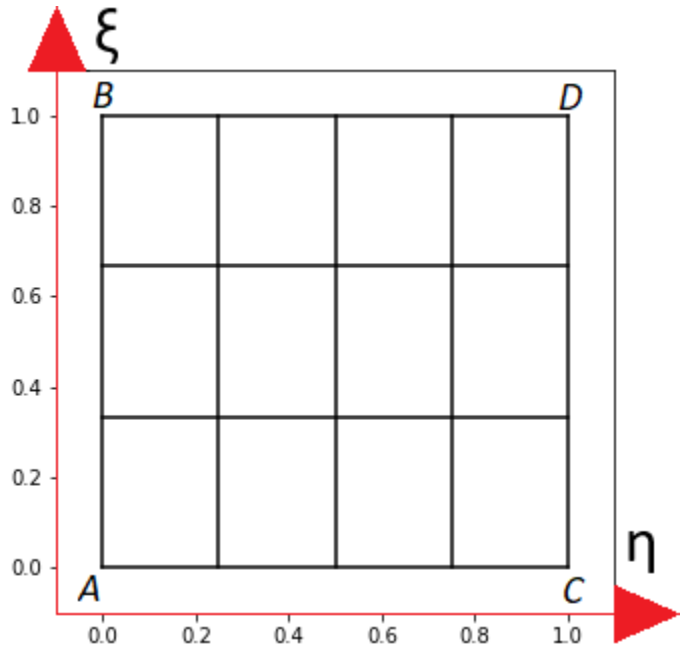


Figure 4 : Maillage de référence avec axes x et y

Les dimensions extérieures du maillage de référence sont toujours de 1 par 1 et chaque élément possèdent les mêmes dimensions :

- Largeur selon $\vec{\xi}$: $l = {}^{1}/_{n}$
- Longueur selon $\vec{\eta}$: $L = {}^{1}/_{m}$

A chaque coin est assigné une lettre allant de A jusqu'à D et cet ordre est toujours respecté, y compris sur le maillage physique.

Les bords du carré du maillage sont aussi nommés :

- r_top
- r_bottom
- r_left
- r_right

Respectivement le bord haut, bas, gauche et droite, ou aussi : BD, AC, AB, CD.

2.2.4 Coordonnées réelles

Les nœuds possèdent des coordonnées (i, j) mais aussi des coordonnées « réelles » (ξ, η) avec $\xi \in [0,1], \eta \in [0,1]$. Les coordonnées réelles d'un point quelconque k sont :

$$\xi = {}^{i}/_{n-1} \tag{4}$$

$$\eta = {}^{j}/_{m-1} \tag{5}$$

3 Utilisation de « toolsRef.py »

3.1 Maillage de référence « class MeshCPU »

3.1.1 Initialisation

Les outils en lien avec le maillage de référence se trouve tous dans «class MeshCPU ».

$$A = MeshCPU(n, m)$$

Avec A une variable, permet de créer les matrices Numpy (array) de chaque nœud, élément et lignes. La fonction « __init__ » appelle des sous-fonction afin de générer les matrices mais il est aussi possible de ne pas générer dans initialisation ces matrices simplement en utilisant :

$$A = MeshCPU(n, m, build = False)$$

Cette option « $build$ » est par défaut sur « $True$ » et ne nécessite pas d'être utilisé afin de construire les matrices.

Lors de l'utilisation de cette commande, le programme génère, pour $build = True$, les matrices de chaque nœud, élément et lignes et ces matrices restent accessible par tout autre fonction de la classe.

- X_ref : matrice $(N, 2)$

Liste des coordonnées pour chaque nœud de 0 à N, avec une ligne correspondant à un nœud et une colonne correspondant à une coordonnée réelle (ξ, η), la première colonne étant ξ et la deuxième η.

Ci-dessous deux exemples des matrices obtenus en fonction de n et m :

```
[[0.         0.        ]
 [0.25       0.        ]
 [0.5        0.        ]
 [0.75       0.        ]
 [1.         0.        ]
 [0.         0.33333333]
 [0.25       0.33333333]
 [0.5        0.33333333]
 [0.75       0.33333333]
 [1.         0.33333333]
 [0.         0.66666667]
 [0.25       0.66666667]
 [0.5        0.66666667]
 [0.75       0.66666667]
 [1.         0.66666667]
 [0.         1.        ]
 [0.25       1.        ]
 [0.5        1.        ]
 [0.75       1.        ]
 [1.         1.        ]]
```

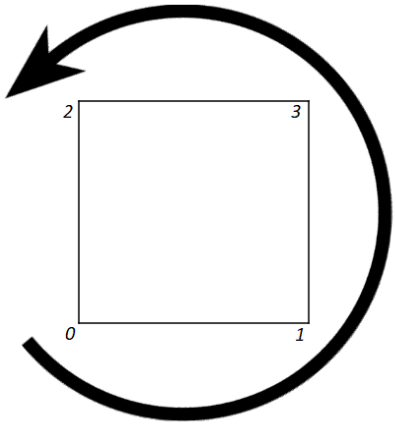
Figure 5 : $n = 5, m = 4$

```
[[0.   0.  ]
 [0.2  0.  ]
 [0.4  0.  ]
 [0.6  0.  ]
 [0.8  0.  ]
 [1.   0.  ]
 [0.   0.25]
 [0.2  0.25]
 [0.4  0.25]
 [0.6  0.25]
 [0.8  0.25]
 [1.   0.25]
 [0.   0.5 ]
 [0.2  0.5 ]
 [0.4  0.5 ]
 [0.6  0.5 ]
 [0.8  0.5 ]
 [1.   0.5 ]
 [0.   0.75]
 [0.2  0.75]
 [0.4  0.75]
 [0.6  0.75]
 [0.8  0.75]
 [1.   0.75]
 [0.   1.  ]
 [0.2  1.  ]
 [0.4  1.  ]
 [0.6  1.  ]
 [0.8  1.  ]
 [1.   1.  ]]
```

Figure 6 : n=6, m=5

- $elem$: matrice $(N\_el, 4)$

Liste des nœuds de chaque élément de 0 à N_el, avec une ligne correspondant à un élément et les colonnes aux 4 nœuds que possède l'élément. Le premier nœud correspond au nœud en bas à gauche de l'élément :



Dans cet exemple ci-contre, nous avons l'unique élément 0 composé des nœuds $0, 1, 2$ et 3. La variable $elem$ serait de la forme :

```
[[0 1 3 2]]
```

Figure 7 : Elément 0 *pour* n=2,m=2

Dans un exemple plus grand, comme $n = 5, m = 4$:

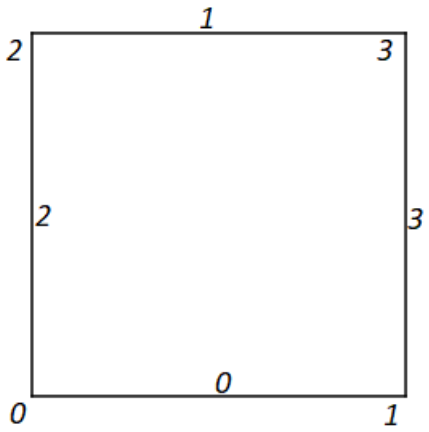
```
[[ 0  1  6  5]
 [ 1  2  7  6]
 [ 2  3  8  7]
 [ 3  4  9  8]
 [ 5  6 11 10]
 [ 6  7 12 11]
 [ 7  8 13 12]
 [ 8  9 14 13]
 [10 11 16 15]
 [11 12 17 16]
 [12 13 18 17]
 [13 14 19 18]]
```

Figure 8 : elem *pour* n=5, m=4

La raison de ce choix d'affiché les numéros des nœuds est qu'il est possible d'accéder simplement avec k aux coordonnées de chaque nœud en utilisant X_ref. Ainsi cette même méthodologie est utilisée pour les lignes ($edges$).

- $edges$

Liste des nœuds composant une ligne de 0 à N_ed, avec les lignes correspondant aux edges et les colonnes aux nœuds. La première colonne correspond au premier nœud et la deuxième colonne au deuxième nœud avec le premier nœud dans le cas des lignes horizontales le nœud à gauche puis le deuxième le nœud à droite ou dans le cas des lignes verticales le nœud en bas est le premier puis le deuxième est en haut. Nous obtenons, en reprenant l'exemple précédent $n = 2, m = 2$:



Ci-contre l'unique élément 0 avec ses nœuds $0,1,2,3$ et ses lignes $0,1,2,3$, sachant que que $0,1$ sont des lignes horizontales et $2,3$ verticales en respectant la numérotation détaillée précédemment.

Ainsi $edges$ serait égale à :

```
[[0 1]
 [2 3]
 [0 2]
 [1 3]]
```

En prenant l'exemple générique $n = 5, m = 4$, nous obtenons $edges$:

```
[[ 0  1]
 [ 1  2]
 [ 2  3]
 [ 3  4]
 [ 5  6]
 [ 6  7]
 [ 7  8]
 [ 8  9]
 [10 11]
 [11 12]
 [12 13]
 [13 14]
 [15 16]
 [16 17]
 [17 18]
 [18 19]
 [ 0  5]
 [ 1  6]
 [ 2  7]
 [ 3  8]
 [ 4  9]
 [ 5 10]
 [ 6 11]
 [ 7 12]
 [ 8 13]
 [ 9 14]
 [10 15]
 [11 16]
 [12 17]
 [13 18]
 [14 19]]
```

Ces matrices correspondent à des matrices Numpy et ainsi il est possible de les manipuler comme n'importe quelle matrice Numpy avec les outils Numpy.

Ainsi afin d'obtenir une ligne spécifique k d'une matrice mat, il suffit de noter :

$$ligne_k = mat[k]$$

Et pour accéder à une colonne c de cette ligne :

$$var = mat[k, c]$$

3.1.2 Liste de chaque corp $listAround(forEach =, get =)$

Cette fonction permet d'obtenir les corps entourant un autre sous forme de matrice, comme les éléments entourant tous les nœuds ou encore les lignes entourant les éléments.

Ainsi en entrée se trouvent deux variables $forEach$ et get avec chacune différentes possibilités.

$$liste = A.listAround(forEach =, get =)$$

Pour $forEach$ et get il existe 3 choix :

- $'Nodes'$
- $'Elements'$
- $'Edges'$

Selon la combinaison choisie, le résultat est différent mais dans chaque cas il s'agit d'énumérer chaque corp get autour de chaque corp $forEach$. get est en option pour $forEach = Nodes$ et renverra la matrice X_ref correspondant aux coordonnées de chaque nœud.

Les dimensions des matrices sont :

		forEach		
	Inputs	$'Nodes'$	$'Elements'$	$'Edges'$
Get	$'Nodes'$	(N,4)	$elem$	$edges$
	$'Elements'$	(N,4)	(N_el,4)	(N_ed,6)
	$'Edges'$	(N,4)	(N_el,8)	$None$
	$None$	X_ref	$None$	$None$

Tableau 1 : Dimensions des matrices selon les entrées $forEach$ et get

Chaque colonne de chaque matrice correspond à un nœud, élément ou ligne en particulier et est de type $integer$. Lorsqu'il n'y a pas de corps à une position où il devrait y en avoir un, $None$ est inscrit dans l'emplacement correspondant de la matrice. Quand cette valeur $None$ est utilisé, puisqu'elle ne peut être que de type $float$, la matrice renvoyée sera en $float$ et non $integer$. L'énumération des corps se fait dans le sens anti horaire et commence toujours à gauche, ou dans les cas où il n'y a pas de corps à gauche, alors en bas à gauche.

Ci-dessous, l'énumération pour chaque combinaison avec l'exemple de $n = 5, m = 4$:

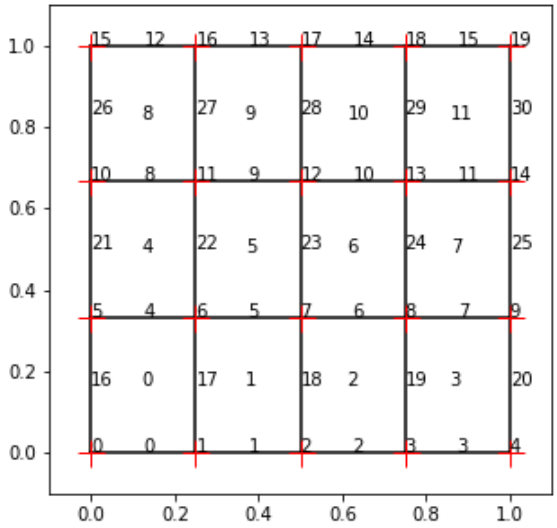


Figure 9 : $n = 5, m = 4$

- $forEach =' Nodes', get = 'Nodes'$

Type : $float$

$$[gauche, bas, droite, haut]$$

```
[[nan nan  1.  5.]
 [ 0. nan  2.  6.]
 [ 1. nan  3.  7.]
 [ 2. nan  4.  8.]
 [ 3. nan nan  9.]
 [nan  0.  6. 10.]
 [ 5.  1.  7. 11.]
 [ 6.  2.  8. 12.]
 [ 7.  3.  9. 13.]
 [ 8.  4. nan 14.]
 [nan  5. 11. 15.]
 [10.  6. 12. 16.]
 [11.  7. 13. 17.]
 [12.  8. 14. 18.]
 [13.  9. nan 19.]
 [nan 10. 16. nan]
 [15. 11. 17. nan]
 [16. 12. 18. nan]
 [17. 13. 19. nan]
 [18. 14. nan nan]]
```

- $forEach =' Nodes', get = 'Edges'$

Type : $float$

$$[gauche, bas, droite, haut]$$

```
[[nan nan  0. 16.]
 [ 0. nan  1. 17.]
 [ 1. nan  2. 18.]
 [ 2. nan  3. 19.]
 [ 3. nan nan 20.]
 [nan 16.  4. 21.]
 [ 4. 17.  5. 22.]
 [ 5. 18.  6. 23.]
 [ 6. 19.  7. 24.]
 [ 7. 20. nan 25.]
 [nan 21.  8. 26.]
 [ 8. 22.  9. 27.]
 [ 9. 23. 10. 28.]
 [10. 24. 11. 29.]
 [11. 25. nan 30.]
 [nan 26. 12. nan]
 [12. 27. 13. nan]
 [13. 28. 14. nan]
 [14. 29. nan 15.]
 [nan 15. nan nan]]
```

- $forEach =' Nodes', get = 'Elements'$

Type : $float$

$$[bas\ gauche, bas\ droite, haut\ droite, haut\ gauche]$$

```
[[nan nan  0. nan]
 [nan nan  1.  0.]
 [nan nan  2.  1.]
 [nan nan  3.  2.]
 [nan nan nan  3.]
 [nan  0.  4. nan]
 [ 0.  1.  5.  4.]
 [ 1.  2.  6.  5.]
 [ 2.  3.  7.  6.]
 [ 3. nan nan  7.]
 [nan  4.  8. nan]
 [ 4.  5.  9.  8.]
 [ 5.  6. 10.  9.]
 [ 6.  7. 11. 10.]
 [ 7. nan nan 11.]
 [nan  8. nan nan]
 [ 8.  9. nan nan]
 [ 9. 10. nan nan]
 [10. 11. nan nan]
 [11. nan nan nan]]
```

- $forEach =' Edges', get = 'Elements'$

Les lignes horizontales et verticales sont légèrement différente, mais le point de départ respecte la règle expliqué précédemment : à gauche sinon en bas à gauche.

Type : $float$

Horizontal :

$$[bas\ gauche, bas, bas\ droite, haut\ droite, haut, haut\ gauche]$$

Vertical :

$$[gauche, bas\ gauche, bas\ droite, droite, haut\ droite, haut\ gauche]$$

```
[[nan nan nan  1.  0. nan]
 [nan nan nan  2.  1.  0.]
 [nan nan nan  3.  2.  1.]
 [nan nan nan nan  3.  2.]
 [nan  0.  1.  5.  4. nan]
 [ 0.  1.  2.  6.  5.  4.]
 [ 1.  2.  3.  7.  6.  5.]
 [ 2.  3. nan nan  7.  6.]
 [nan  4.  5.  9.  8. nan]
 [ 4.  5.  6. 10.  9.  8.]
 [ 5.  6.  7. 11. 10.  9.]
 [ 6.  7. nan nan 11. 10.]
 [nan  8.  9. nan nan nan]
 [ 8.  9. 10. nan nan nan]
 [ 9. 10. 11. nan nan nan]
 [10. 11. nan nan nan nan]
 [nan nan  0.  4. nan nan]
 [nan nan  1.  5.  4.  0.]
 [nan nan  2.  6.  5.  1.]
 [nan nan  3.  7.  6.  2.]
 [nan nan nan nan  7.  3.]
 [nan  0.  4.  8. nan nan]
 [ 0.  1.  5.  9.  8.  4.]
 [ 1.  2.  6. 10.  9.  5.]
 [ 2.  3.  7. 11. 10.  6.]
 [ 3. nan nan nan 11.  7.]
 [nan  4.  8. nan nan nan]
 [ 4.  5.  9. nan nan  8.]
 [ 5.  6. 10. nan nan  9.]
 [ 6.  7. 11. nan nan 10.]
 [ 7. nan nan nan nan 11.]]
```

- $forEach =' Elements', get = 'Edges'$

Type : $integer$

$$[gauche, bas, droite, haut]$$

```
[[16  0 17  4]
 [17  1 18  5]
 [18  2 19  6]
 [19  3 20  7]
 [21  4 22  8]
 [22  5 23  9]
 [23  6 24 10]
 [24  7 25 11]
 [26  8 27 12]
 [27  9 28 13]
 [28 10 29 14]
 [29 11 30 15]]
```

- $forEach =' Elements', get = 'Elements'$

Type : $float$

$$[gauche, bas\ gauche, bas, bas\ droite, droite, haut\ droite, haut, haut\ gauche]$$

```
[[nan nan nan  1.  5.  4. nan]
 [ 0. nan nan nan  2.  6.  5.  4.]
 [ 1. nan nan nan  3.  7.  6.  5.]
 [ 2. nan nan nan nan nan  7.  6.]
 [nan nan  0.  1.  5.  9.  8. nan]
 [ 4.  0.  1.  2.  6. 10.  9.  8.]
 [ 5.  1.  2.  3.  7. 11. 10.  9.]
 [ 6.  2.  3. nan nan nan 11. 10.]
 [nan nan  4.  5.  9. nan nan nan]
 [ 8.  4.  5.  6. 10. nan nan nan]
 [ 9.  5.  6.  7. 11. nan nan nan]
 [10.  6.  7. nan nan nan nan nan]]
```

3.1.3 Récupération de données de l'instance

A l'aide de certaines fonctions, il est possible de récupérer les données calculées dans la classe.

- Récupérer n et m utilisée dans la classe : $A.export\_n()$ et $A.export\_m()$
- Récupérer le nombre de nœuds, éléments ou lignes :

$$A.export\_N() ; A.export\_N\_el() ; A.export\_N\_ed()$$

- Récupérer les listes des nœuds X_ref, des éléments $elem$ et des lignes $edges$:

$$A.export\_X\_ref(build =)$$

$$A.export\_elem(build =)$$

$$A.export\_edges(build =)$$

$build =$ est pas défaut sur $False$. En choisissant $build = True$ alors la matrice à exporter sera reconstruite au préalable. Cela permet d'éviter de réaliser les calculs de nouvelles fois pour obtenir les valeurs déterminées au sein de la classe.

3.1.4 Coordonnées de position (i,j) et réelles (ξ,η)

Les coordonnées réelles d'un nœud k sont obtenu à l'aide de la matrice X_ref générer par la fonction $A.generateNodes()$. La matrice des coordonnées (i,j) de chaque nœud peut être obtenu avec la fonction suivante :

$$A.nodePosition()$$

Cette fonction renvoie une matrice des coordonnées de positions (i,j) de la forme et l'énumération suit la même forme que les coordonnées réelles, de gauche à droite en commençant par le bas. Pour $n=5, m=4$ la matrice est :

```
[[0 0]
 [1 0]
 [2 0]
 [3 0]
 [4 0]
 [0 1]
 [1 1]
 [2 1]
 [3 1]
 [4 1]
 [0 2]
 [1 2]
 [2 2]
 [3 2]
 [4 2]
 [0 3]
 [1 3]
 [2 3]
 [3 3]
 [4 3]]
```

3.1.5 Changement des valeurs de n et m au sein de la classe

Il est aussi possible de modifier n et m dans une classe $A = MeshCPU(n=, m=)$ à l'aide de la fonction $update()$.

Son utilisation est la suivante :

$$A.update(input1, input2 \dots)$$

Avec comme entrées $(input)$:

- $n = valeur\ n$ ($None$ par défaut)
- $m = valeur\ m$ ($None$ par défaut)
- $build = False$ ($True$ par défaut)

Cette fonction permet de remplacer les valeurs n et m au sein de la classe.

3.1.6 Représentation graphique $plot$

La fonction $A.plot()$ permet de créer une représentation graphique du maillage. Elle s'utilise de la manière suivante :

$$A.plot(input1, input2 \dots)$$

Avec comme entrées ($input$) :

- $size\_x = integer$

Correspond à la taille horizontale réelle du graphique (5 par défaut)

- $size\_y = integer$

Correspond à la taille verticale réelle du graphique (5 par défaut)

La taille du graphique n'a pratiquement aucune importance sur l'aspect visuel du graphique.

- $nodes = boolean$

Placer des marqueurs pour les nœuds ? ($True$ par défaut)

- $nodesNames = boolean$

Placer les noms de chaque nœud ? ($True$ par défaut)

- $elementsNames = boolean$

Placer les noms de chaque élément ? ($True$ par défaut)

- $edgesHorizontal = boolean$

Tracer les lignes horizontales ? ($True$ par défaut)

- $edgesVertical = boolean$

Tracer les lignes verticales ? ($True$ par défaut)

- $edgesNames = boolean$

Placer les noms de chaque ligne ? ($True$ par défaut)

3.1.7 Applications

Nous souhaitons obtenir les coordonnées (i, j) et réelles (ξ, η) du deuxième nœud de la ligne droite de l'élément numéro 103 d'un maillage de 10 par 19. Les étapes à suivre sont décrites ci-dessous.

i. Nous prenons donc $n = 10$ et $m = 19$, ce qui fait 162 éléments :

$$A = MeshCPU(n = 10, m = 19)$$

ii. Inscrivons dans une variable la liste des lignes :

$$liste = A.listAround(forEach =' Elements', get =' Edges')[103]$$

iii. Extrayons le numéro de la ligne droite :

$$N\_ligne = liste[2]$$

iv. Ce numéro permet à partir des fonctions d'export d'obtenir les nœuds de cette ligne :

$$noeuds\_lignes = A.export\_edges()$$

v. Il est possible ensuite d'obtenir les nœuds k de la ligne droite de l'élément 103 :

$$ligne2 = noeuds\_ligne[N\_ligne]$$

vi. Et enfin le deuxième nœud de la ligne :

$$k = ligne2[1]$$

vii. Finalement les coordonnées de ce nœud que nous cherchons sont :

$$resultat1 = A.nodePosition()[k]$$

Les coordonnées position (i, j) sont :

```
[ 5 12]
```

Et pour les coordonnées réelles :

$$resultat2 = A.export\_X\_ref()[k]$$

```
[0.55555556 0.66666667]
```

Nous cherchons ensuite à tracer 3 graphiques différents :

- Un graphique avec les noms de chaque corp, avec chaque lignes et nœuds et de taille 5x5 et $n = 5, m = 4$.
- Un graphique avec les noms des nœuds uniquement et les lignes verticales de tracer, sans les nœuds, et pour $n = 9, m = 6$
- Un graphique avec lignes et nœuds mais sans aucuns noms, avec $n = 20, m = 12$.

<u>Premier cas :</u>

$$A = MeshCPU(n = 5, m = 4)$$

$$A.plot()$$

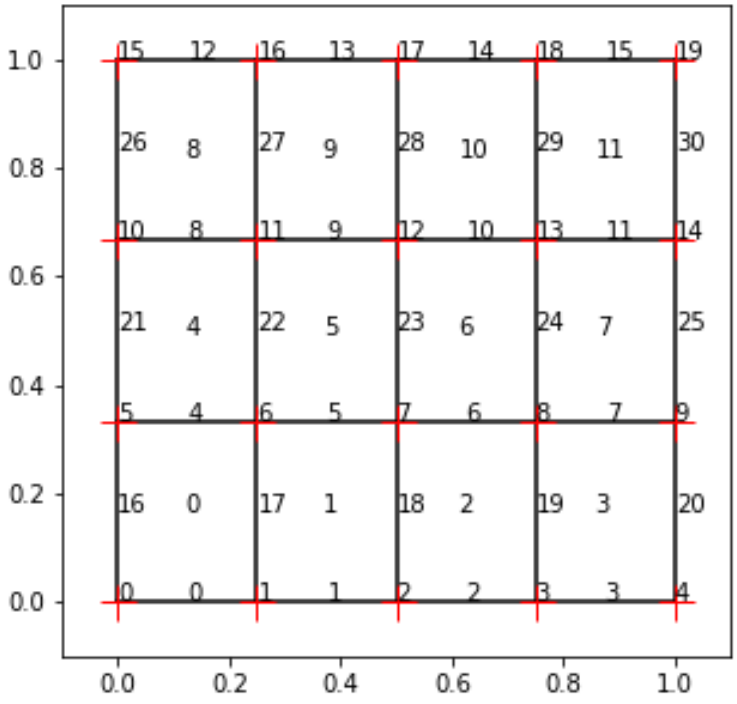


Figure 10 : Premier cas $n = 5, m = 4$

Deuxième cas :

$$A = MeshCPU(n = 9, m = 6)$$

$$A.plot(edgesHorizontal = False, edgesNames = False, elementsNames = False, nodes = False)$$

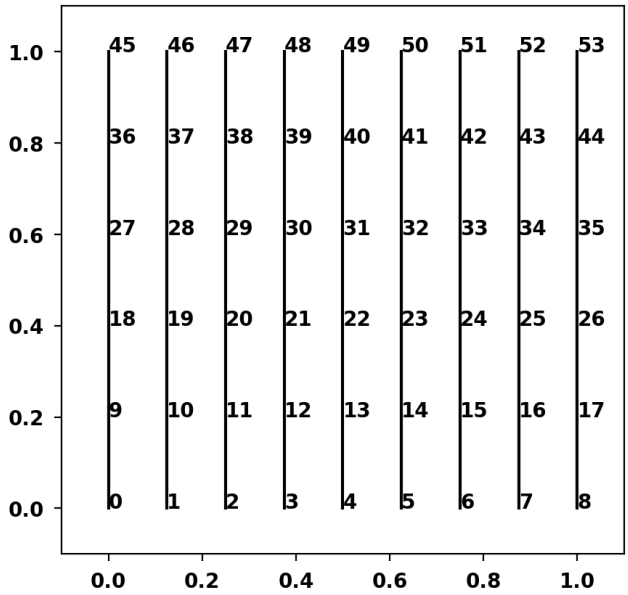


Figure 11 : Deuxième cas $n = 9, m = 6$

Troisième cas :

$$A = MeshCPU(n = 9, m = 6)$$

$$A.plot(nodesNames = False, edgesNames = False, elementsNames = False, nodes = False)$$

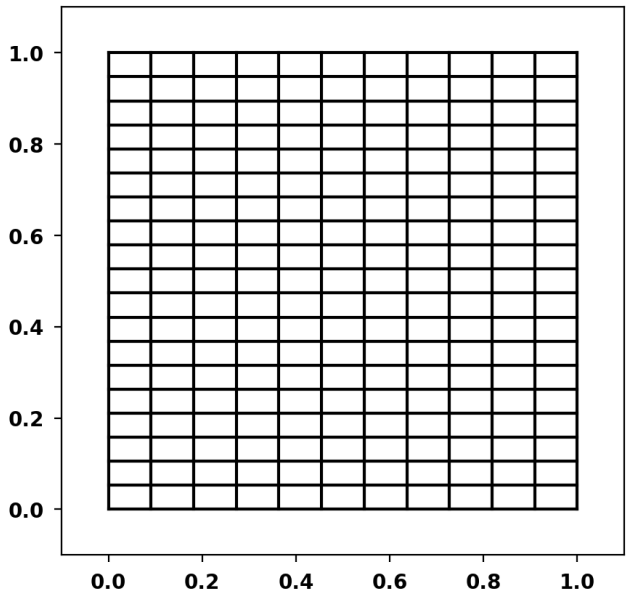


Figure 12 : Troisième cas $n = 20, m = 12$

3.2 Exportation Importation des résultats « ExportImport.py »

Le fichier « ExportImport.py » est un sous-programme de « toolsRef.py » utilisé dans la classe de référence $MeshCPU()$ permettant d'exporter et d'importer les résultats sous différent format :

- $.msh$
- $.vtk$
- $.su2$

La sous-classe $ImportExport()$ du fichier « ExportImport.py » permet d'utiliser deux fonctions principales :

- $exportFile(format =, filename =, build =)$

$filename$ est le nom du fichier de type $string$, par default : $'meshRef'$. Il ne faut pas ajouter l'extension, elle est ajoutée pendant exportation selon le choix de $format$.

$format$ est l'extension du fichier de type $string$, avec trois choix possibles :

- $'vtk'$
- $'su2'$
- $'msh'$

Selon le choix d'extension, le fichier désiré au nom $filename.extension$ sera exporter dans le dossier de projet avec les données écrites sur le fichier au format de l'extension. Ces extensions de fichier peuvent s'ouvrir avec un logiciel de traitement de texte comme « Bloc-notes » ou « Notepad++ ».

$build$ permet de générer ou régénérer les nœuds si besoin, par défaut : $False$.

- $importFile(format =, filename =)$

Les entrées sont les mêmes que pour $exportFile()$ et respectent les mêmes règles :

$filename$ est le nom du fichier sans l'extension de type $string$.

$format$ est l'extension du fichier de type $string$, avec les mêmes trois choix possibles.

Cette fonction vient remplacer la variable $self.X\_ref$ au sein de l'instance par les valeurs du fichier importé.