Documentation du programme PyDyCo

Pour des questions, vous pouvez me contacter : mhd.amn.mamouni@gmail.com

Table des matières

1 Introduction **2**

2 Prérequis en logiciel **3**
2.1 Sous Windows OS . . . 3
2.2 Sous Ubuntu . . . 4

3 Utilisation **5**
3.1 Création du réseau de dipôles thermoélectriques . . . 5
3.2 Définition des conditions aux bords . . . 7
3.3 Précision et condition d'arrêt . . . 8
3.4 Visualisation des résultats . . . 8

4 Architecture du programme **10**
4.1 organigramme de programation . . . 12
4.2 Les fonctions utilisées . . . 12
4.2.1 Les fonctions et instructions PySpice . . . 13
4.2.2 Les fonctions PyDyCo et principe de fonctionnement . . . 14

A Exemple **27**
A.0.1 Potentiel électrique . . . 28
A.0.2 Température . . . 29
A.0.3 Réseau de création d'entropie . . . 29
A.0.4 Courant de particules . . . 30
A.0.5 Courant de de chaleur . . . 31
A.0.6 Courant d'énergie . . . 32
A.0.7 Courant d'entropie . . . 33
A.0.8 comparaison entre un fichier vtk, portant l'extension vc et un fichier vtk simple . . . 33

Chapitre 1

Introduction

Le programme PyDyCo est une version du programme DyCo utilisant Python.
La notice suivante permet d'expliquer les points importants de ce programme. Elle se divisera en 3 parties :

— Une partie qui détaillera les prérequis en logiciel
— Une partie qui expliquera comment utiliser le programme
— Et une dernière qui explique le fonctionnement de PyDyCo avec toutes les fonctions utilisées. Cette partie permettra aux utilisateurs, si ils le souhaitent, d'apporter des modifications au programme.

Chapitre 2

Prérequis en logiciel

⚠ **Avant de commencer l'installation, il faut lire tout le chapitre.** ⚠

Afin de pouvoir lancer le programme il faut disposer de :

- Anaconda (Spyder) ici
- PySpice ici,
- ngspice (qui est sensé être installé avec Pyspice)
- numpy (installé avec Anaconda)
- matplotlib (installé avec Anaconda)
- Paraview ici

Des modifications doivent se faire selon la machine utilisée, Lors de l'installation de Ngspice ;

2.1 Sous Windows OS

Sous windows, il est impératif de suivre les instructions suivantes lors de l'installation, à savoir

- Après l'installation de ngspice-30_dll_64.zip , il faut décompresser le fichier dans un dossier portant le nom de **Spice64_dll** qui sera situé dans : `C:\Program Files\Spice64_dll`.
- Vous trouvez accompagné de ce manuel un dossier ng_start64_binaries.7z Après l'extraction, copiez le fichier **bin** contenu dedans dans le dossier `C:\Program Files\Spice64_dll` sous le nom **dll-vs**.
- Lancer l'installation du paquet avec en entrée dans la console IPython dans Spyder : [–] !pip install PySpice.
- Après l'installation de PySpice, le programme rencontrera des erreurs dues à l'existence de plusieurs variables et aux manques de certaines fi-

chiers. les etapes a effectuer pour les corriger portant le nom de ngcomplex. Il suffit s'aller sur le fichier **api.h** et de changer la variable ngcomplex en variables en ngcompleax.

Le programme PyDyCo est prêt à être utilisé, il suffit de lancer Spyder et de traiter **py_dyco.py** comme tout programme ayant l'extension **.py**

2.2 Sous Ubuntu

Sous Ubuntu, les modifications à faire se situent au niveau des fonctions PySpice, il faut donc :

— Accéder au fichier `simulation.py` se situant sur le chemin suivant `PySpice\Spice\simulation.py`, et transformer l'utilisation de PySpice non pas en **Shared** mais plutôt en **Subprocess**. pour y parvenir il faut dé-commenter la **670ème** ligne et de commenter la **671ème** ligne.

FIGURE 2.1 – Modificatios à apporter sous Ubuntu

Le programme PyDyCo est prêt à être utilisé, il suffit d'exécuter **py_dyco.py** avec **Python3** comme tout programme ayant l'extension **.py**

Chapitre 3

Utilisation

⚠ Si vous êtes sous Windows et Spyder, pensez à exécuter le fichier **foncdyco.py** qui contient les fonction du programme avant d'exécuter le programme. le fichier est contenu dans le dossier du même nom.

3.1 Création du réseau de dipôles thermoélectriques

Le réseau de dipôles utilisés sous PyDyCo est sous la forme suivante :

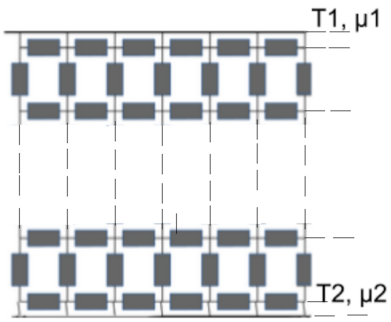


FIGURE 3.1 – La forme du réseau sur PyDyCO

L'utilisateur rentre dans un fichier pré-processing nommé **matériaux.txt**, le réseau composé de différents matériaux. tel que chaque lettre signifie :

- **n** : Semis-conducteurs Types n
- **p** : Semis-conducteurs Types p
- **m** : Métaux

— **v** : Vide
— **s** : Échantillon quelconque

Par exemple, la figure suivante montre un réseau de dipôles thermoélectriques et sont équivalent dans le fichier **matériaux.txt** :

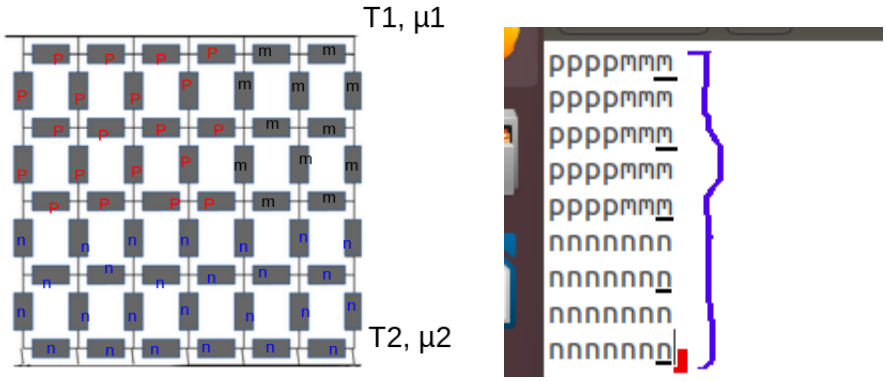


FIGURE 3.2 – Le réseau et son equivalent dans le fichier pré-processing

⚠⚠Pour éviter les erreurs liées à l'exécution du programme, les règles suivantes doivent être respectées :

1. Le nombre de lignes doit être impair.
2. Ne pas revenir à la ligne au dernier caractère écrit.
3. Le dernier dipôle de chaque ligne impair n'est pas pris en considération. (voir figure 3.2)

La modification des caractéristiques physiques des dipôles se fait dans le programme PyDyCo entre la ligne 74 et la ligne 96 :

```
#RESISTANCE THERMIQUE DE {P,S,M,N,K,V}
RTP=1/3.194
RTS=1/3.194
RTM=1/3.194
RTN=1/3.194
RTK=1/3.194
RTV=1/3.194
#RESISTANCE ELECTRIQUE DE {P,S,M,N,K,V}
REP=0.04
RES=0.04
REM=0.04
REN=0.04
REK=0.04
REV=0.04
#COEFICIENT SEEBECK  DE {P,S,M,N,K,V}
SP=200e-6
SS=1000e-3
SM=-5e-6
SN=-200e-6
SK=10e-3
SV=10e-9
```

FIGURE 3.3 – Les caractéristiques des matériaux : résistances thermiques, résistances électriques, et coefficients Seebeck

3.2 Définition des conditions aux bords

Les conditions aux bords imposées sont μ_1, μ_2 et T_1, T_2. elles peuvent être changées en allant à la ligne **28**
Le potentiel électrochimique μ_2 est considéré comme **nul** et reste donc inchangé. Il est donc pris comme référence et reste nul. Nous nous intéressons qu'à la **différence** de potentiel électrochimique $\Delta\mu$, d'oú l'unique variation de μ_1.
Deux méthodes sont possibles pour imposer les conditions aux bords.

```
mu1=1.5
T1=400
mu2= 0
T2=750
```

FIGURE 3.4 – Les conditions aux limites à définir

— Soit en imposant les conditions aux bords instantanément.
— Soit en augmentant l'écart graduellement.

Ce choix peut être fait en commentant ou en décommentant certaines instructions comme indiqué dans le programme.

Dans le cas ou l'on souhaite augmenter l'écart graduellement, il est possible de spécifier le pas de l'incrémentation pour le potentiel et pour la température.

Il est important de noter que nous n'avons pas trouvé de différence entre les deux configurations

```
#pou une augmentation graduelle
m=0.001 #le debut de l'incrémentation du potentiel électrique
#Preciser la valeur du pas :
#la valeur du pas pour le circuit électrique:
pasv=0.1
#la valeur du pas pour le circuit thermique :
past=10

#Si vous ne souhaitez pas augmenter le potentiel graduelement, décommentez cette instruction:
m=mu1 #<= cette instruction


if T1>T2:
    T=T2
    #Si vous  souhaitez augmenter la temeprature graduelement, et T1>T2 commentez ces deux instructions:
    T=T1  #<=cette instruction
    Tmax =T1 #<=cette instruction
elif T1<T2:
    T= T1
    #Si vous souhaitez  augmenter la temeprature graduelement, et T1<T2 commentez ces deux instructions:
    T=T2  #<=cette instruction
    Tmax =T2 #<=cette instruction
elif T1==T2:
    T=T1
    Tmax=T
```

FIGURE 3.5 – Les modidifications à faire pour définir la méthode à imposer pour définir les conditions aux limites

3.3 Précision et condition d'arrêt

La condition de sortie de la boucle 4.1 est définie par l'écart quadratique entre les valeurs des potentiels (électrique et thermique) la à $n-ième$ itération et la $(n-1)-ième$ itération.
La sortie de la boucle est enclenchée lorsque les écarts quadratiques atteignent une certaine valeur définie par l'utilisateur. Nous pouvons régler cette précision en modifiant la valeur des variables **precisionV** et **precisionT**.

3.4 Visualisation des résultats

Après l'exécution du programme, le résultat est visualisé sur Paraview. Les grandeurs physiques extraites sont rangées dans des fichiers du même nom, avec des extensions **.vtk**. Ils portent les noms suivants :

- **IN.vtk** : Courant électrique.
- **IN_vc.vtk** : Courant électrique avec les composantes suivant X et les composantes suivant Y séparées .
- **IQ.vtk** : Courant de chaleur.
- **IQ_vc.vtk** : Courant de chaleur avec les composantes suivant X et les composantes suivant Y séparées.
- **IE.vtk** : Courant d'énergie.

— **IE_vc.vtk** :Courant d'énergie avec les composantes suivant X et les composantes suivant Y séparée .
— **IS.vtk** : Courant de d'entropie.
— **IS_vc.vtk** :Courant de d'entropie avec les composante suivant X et les composantes suivant Y séparées.
— **temp.vtk** : Champs de température.
— **volt.vtk** : Champs de potentiel électrique.
— **S.vtk** : Champs de création d'entropie.

Chapitre 4

Architecture du programme

Dans le programme PyDyCo, le couplage entre le courant d'énergie et le courant de particule est mis en évidence suivant une approche itérative. Le réseau thermoélectrique est scindé en deux sous-réseaux; un sous réseau électrique pur avec des résistances électriques, et un sous réseau thermique pur avec des résistances thermiques. Les deux sous-réseaux sont ensuite couplés grâce à la matrice d'Onsager et au coefficient Seebeck.

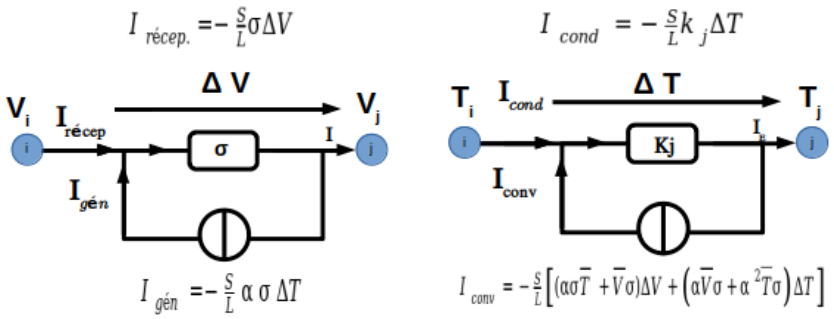


FIGURE 4.1 – Gauche. dipôle électrique avec une contribution thermique. Droite dipôle thermique avec une contribution électrique.

Le couplage se manifeste par un sous réseau de générateurs de courants électriques pour le sous-réseau électrique et un sous réseau de générateur de courant d'énergie pour le réseau thermique.

Nous nous retrouvons donc avec un dipôle électrique passif dit "récepteur" avec une contribution thermique active dite "générateur" sous forme d'un générateur de courant électrique. Pour la partie thermique, nous nous retrouvons

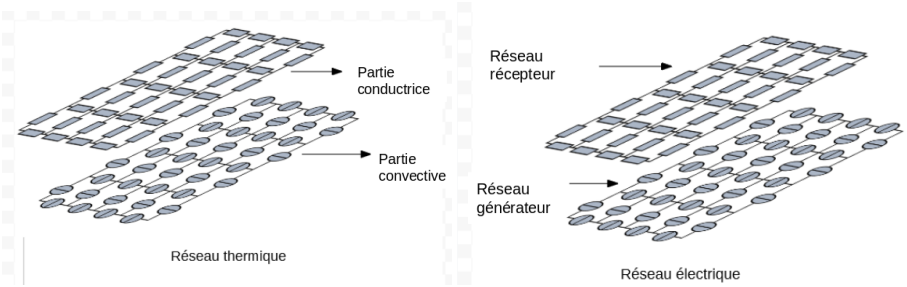
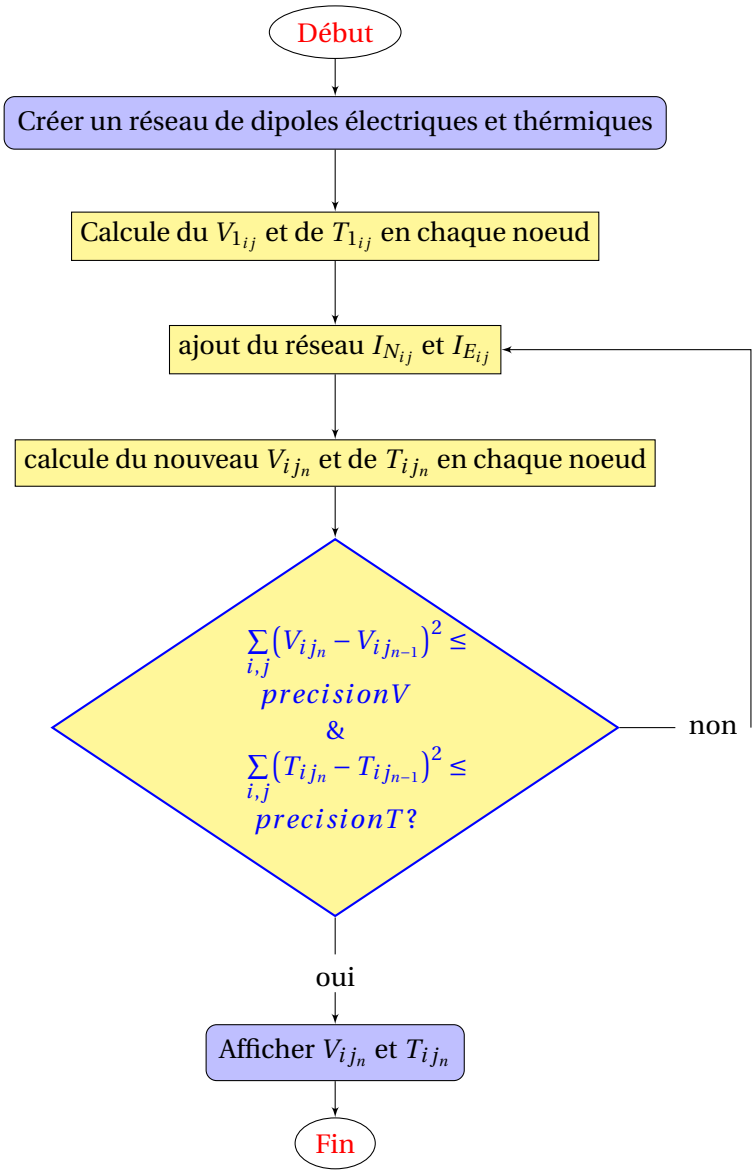


FIGURE 4.2 – Gauche. Réseau thermique avec une contribution électrique . Droite Réseau électrique avec une contribution thermique

avec un dipôle passif dit "conducteur" ainsi qu'une contribution électrique dite "convective" sous forme de générateur de courant d'énergie.

Dans toute cette notice, il nous arrivera de considérer dans notre explication un réseaux thermoélectrique unique, comme il peut nous arriver de considérer deux réseaux distincts : un réseau thermique, et un réseau électrique.

4.1 organigramme de programation



4.2 Les fonctions utilisées

Le programme PyDyCo a été codé en utilisant la librairie PySpice. Grâce à cette librairie nous avons créé nos propres fonctions qui permettent d'avoir un

programme léger et facilement maniable. Dans les sections suivantes, nous allons détailler comment sont construites les fonctions utilisées.

4.2.1 Les fonctions et instructions PySpice

Fonctions et instructions de création de dipôles

Les sous librairies dont on fait appel pour python sont les suivantes :

```
import PySpice.Logging.Logging as Logging
logger = Logging.setup_logging()
from PySpice.Spice.Netlist import Circuit
from PySpice.Unit import *
```

Pour créer un circuit électrique on procède comme suit :

```
[La variable du circuit] = Circuit("nom du circuit")
```

Afin simplifier l'illustration nous prenons [**La variable du circuit**] =**circuit**. Ce choix est totalement arbitraire, nous aurions pu utiliser **patate** comme variable du circuit cela est donc sans erreur.
Sur PyDyCo, nous avons deux variables de circuit : La variable **circuit** qui est associée au circuit électrique, et la variable **circuiT** qui est associée au circuit thermique.

Lorsque nous voulons créer un dipôle, la syntaxe utilisée est comme suit :

```
[nomducircuit].[dipole]('variable','node1','node2','valeur')
```

Par exemple dans notre cas :

```
circuiT.R('A','in','out',50)
```

crée une résistance thermique dans le circuit thermique. cette résistance sera appelée **RA**, et est située entre les noeuds 'in' et 'out' et a pour valeur 50 $m^2.K/W$.

```
circuit.I(1,1,0,50)
circuit.V(2,2,0,5)
```

signifient que nous avons créé respectivement :

- Un générateur de courant portant la variable **I1** et sera entre le nœud '1' et le nœud '0' Le générateur aura pour valeur 50 Ampères.
- Un générateur de tension portant la variable **V2**, qui se situe entre le nœud '2' et'0' et ayant un différence de potentiel électrique de $\Delta V = 5Volts$.

Par exemple, si l'on revient à notre cas où nous avons choisi **patate** comme variable du circuit, la création d'une résistance sera comme suit :
patate.R(2,1,0,5)
Pour PySpice, le noeud '0' représente le potentiel terrestre.

Instructions de calcule

Après avoir créer le circuit électrique (ou thermique) ; le calcule est enclenché en utilisant ces lignes d'instructions pour le circuit électrique :

```
simulator = circuit.simulator(temperature=25, nominal_temperature=25)
```

ou ces lignes d'instruction pour le circuit thermique :

```
simulator = circuiT.simulator(temperature=25, nominal_temperature=25)
```

Ces instructions nous permettent d'identifier le potentiel électrique ou la valeur de la température en chaque nœud du réseau thermoélectrique. les valeur seront après classées dans un dictionnaire **analysis** à l'aide de l'instruction suivante

analysis = **simulator.operating_point**()

Pour afficher la valeur des potentiels ou de la température en chaque nœud, il suffit de faire une boucle sur les dictionnaires pour pouvoir les extraire :

```
for node in analysis.nodes.values():
  print ('Node {}: {:5.2f} V'.format(str(node), float(node)))
```

Pour extraire les courants, on procède par une boucle ressemblant à celle faite pour extraire les potentiels.

```
for branche in analysis.branches.values():
    print('branche {}: {:5.19f} A'.format(str(branche),float(branche)))
```

Il est important de commencer par une instruction qui permet de bien définir le signe du courant dans le réseau :

```
for resistance in tp :
    resistance.minus.add_current_probe(circuit)
```

tel que **tp** est un **tuple** contenant toutes les résistances du réseau :

```
tp= (circuit.R1, circuit.R2,..., circuit.RN)
```

4.2.2 Les fonctions PyDyCo et principe de fonctionnement

Le programme PySpice, est un programme suivant un réseau bien précis, afin de tout faciliter à l'utilisateur, des fonctions ont été créées .

Les informations sur les dipôles thermoélectriques (valeur des résistances thermiques et électriques de chaque dipôle ainsi que les coefficients Seebeck) sont récupérées et classées dans des tableaux. Ces tableaux contiennent certains

éléments indésirables(il s'agit des derniers éléments des lignes impaires). Ces éléments serons ensuite enlevés grâce à la fonction **tran_mat**.

Dans la section qui suit, nous allons définir les fonctions créées sur PyDyCo par ordre d'utilisation et expliquer leurs utilités dans le programme. Vous pouvez retrouver toutes ces fonctions dans le fichier **foncdyco.py**

tran_mat(lin,col,r)

La fonction agit sur les matrices qui contiennent les valeurs des résistances électriques et thermiques ainsi que les valeurs des coefficients Seebeck. Elle permet d'éliminer le dernier élément de chaque ligne impaire.

— Les éléments d'entrée

1. **lin** : représente le nombre de lignes dans mon réseau.
2. **col** : le nombre de colonnes dans mon réseau.
3. **r** : Un tableau 2D qui représente les valeurs des caractéristiques des dipôles (Résistances électriques, thermiques, et coefficients Seebecks).

— Les éléments de sortie

Un tableau d'une dimension sans les éléments inutiles (dernier élément de chaque ligne impaire).

Circuit_creation_R(cir,dip,lin,col)

La fonction permet de créer un réseau de résistances électriques ou thermiques, suivant le schéma du fichier **Materiaux.txt**.

— Les variables d'entrée

1. **cir** : le nom du circuit, à définir entre :
   — **circuit** pour le circuit électrique.
   — **circuiT** pour le circuit thermique.
2. **dip** : représente un tableau de valeurs des résistances thermiques pour **circuiT**, ou un tableau de valeurs des résistances électriques pour **circuit**. Il s'agit des tableaux obtenus grâce à la fonction **tran_mat**
3. **lin** : nombre de lignes dans le réseau.
4. **col** : nombre de colonnes dans le réseau.

— Les variables de sortie :

1. Création d'un réseau de dipôles passifs.

2. Un tuple regroupant toutes les résistances créées. Ce tuple sera ensuite utilisé lors du lancement du calcul des courants électriques ou d'énergie.

Circuit_creation_V(cir,lin,col,va,vb)

La fonction permet de définir le potentiel électrique et la température entre les bords du réseau

— les variables d'entrée

1. **cir** : le nom du circuit, a définir entre
   — **circuit** pour le circuit électrique (Pour imposer le potentiel électrique aux bords).
   — **circuiT** pour le circuit thermique (Pour imposer les températures aux bords).
2. **lin** : Nombre de lignes dans le réseau.
3. **col** : Nombre de colonnes dans le réseau.
4. **va** : Représente la valeur du potentiel en haut du réseau pour **circuit**, et la valeur de la température en haut du réseau pour le **circuiT**.
5. **vb** : Représente la valeur du potentiel en bas du réseau pour **circuit**, et la valeur de la température en bas du réseau pour le **circuiT**.

— Les variables de sortie

La fonction à la sortie crée des générateurs de tension aux bords du réseau, ou fixe des températures aux bords.

Après avoir créé un réseau de dipôles thermiques et électriques, nous manifestons le couplage thermoélectrique avec des générateurs de courants électriques et des générateurs de courant d'énergie.

m_moy(lin,col,vn1,vn2)

La fonction permet de calculer les moyennes du potentiel (électrique ou thermique) entre chaque dipôle dans réseau.

— les variables d'entrée

1. **lin** : nombre de lignes dans le réseau.
2. **col** : nombre de colonnes dans le réseau.
3. **vn1** : Il s'agit d'un tableau qui contient des éléments en format string, correspondant au nom de chaque nœud du réseau (Dans le programme il porte le nom de sn pour le réseau électrique et snt pour le réseau thermique).
4. **vn2** : Il s'agit d'un tableau qui contient des éléments en format float, qui correspondent à la valeur du potentiel (électrique ou thermique) de chaque nœud du réseau (fn pour le réseau électrique et fnt pour le réseau thermique).

   Les deux vecteurs (ou tableaux) **vn1** et **vn2** sont symétriques en terme de position i.e. la position du nom de chaque nœud dans le vecteur **vn1** et la position où se trouve sa valeur en terme de potentiel (électrique ou thermique) dans le vecteur **vn2** sont les mêmes.

— Les variables de sortie
Un tableau en 1D (ou vecteur) qui regroupe les moyennes entre les potentiels aux bords de chaque dipôle.

---

delt_m(lin,col,vn1,vn2)

Cette fonction permet de calculer les différences de potentiel (électrique ou thermique) entre chaque dipôle.

— les variables d'entrée

1. **lin** : nombre de lignes dans le réseau.
2. **col** : nombre de colonnes dans le réseau.
3. **vn1** : il s'agit d'un tableau qui contient des éléments en format string correspondants au nom de chaque nœud du réseau (Dans le programme ils portent le nom de sn pour le réseau électrique et snt pour le réseau thermique).
4. **vn2** : il s'agit d'un vecteur qui contient des éléments en format floats, et qui correspondent à la valeur du potentiel (électrique ou thermique) de chaque nœud du réseau (fn pour le réseau électrique et fnt pour le réseau thermique).

   Les deux vecteurs (ou tableaux) **vn1** et **vn2** sont symétriques en terme de position i.e. la position du nom de chaque nœud dans

le vecteur **vn1** et la position ou se trouve sa valeur en terme de potentielo (électrique ou thermique) dans le vecteur **vn2** sont les mêmes.

— Les variables de sortie
Un tableau en 1D (ou vecteur) qui regroupe les différence de potentiels entre les bords de chaque dipôle.

---

iq (alpha,sigma,tm,vm,delv,delt)

La fonction permet de calculer les valeurs des générateurs de courant d'énergie associées à chaque dipôle électrique.

— Les variables d'entrée

1. **alpha** : représente les valeurs des coefficients Seebek de chaque élément du réseau, sous la forme d'un tableau.
2. **sigma** : représente la valeur des conductances électriques (l'inverse des résistances).
3. **tm** : vecteur représentant la valeur des températures moyennes entre chaque dipôle.
4. **vm** : vecteur représentant la valeur des potentiels électriques moyens entre chaque dipôle.
5. **delv**vecteur représentant les valeurs des différences de potentiels électriques entre chaque dipôle.
6. **delt** : vecteur représentant les valeurs des différences de températures entre chaque dipôle thermique.

— Les variables de sortie
En sortie, nous retrouvons la valeur des courants d'énergie générés par le réseau électrique pour chaque dipôle thermique.

In (alpha,sigma,delt)

La fonction permet de calculer les valeurs des générateurs de courant électrique associées à chaque dipôle thermoélectrique.

— Les variables d'entrée

1. **alpha** : représente les valeurs des coefficients Seebek de chaque élément du réseau, sous la forme d'un tableau.

2. **sigma** : représente les valeurs des conductances électriques (l'inverse des résistances)
3. **delt** : vecteur représentant les valeurs des différences de température entre chaque dipôle thermique.

— Les variables de sortie
En sortie, nous retrouvons les valeurs des courants électriques générés par le réseau thermique pour chaque dipôle électrique.

Circuit_creation_I(cir,dip,lin,col)

Permet de rajouter les contributions thermiques sur le réseau électrique ; et les contributions électriques sur le réseau thermique.

— Les variables d'entrée

1. **cir** : le nom du circuit, à définir entre :
   — **circuit** pour le circuit électrique
   — **circuiT** pour le circuit thermique.
2. **dip** : il s'agit d'un vecteur qui comprend les valeurs des générateurs de courant électrique pour **circuit** et des valeurs de générateurs de courants d'énergie pour **circuiT** . Ces valeurs sont générées à partir de la fonction **In** et de la fonction **Iq**.
3. **lin** : nombre de lignes dans le réseau.
4. **col** : nombre de colonnes dans le réseau.

— Les variables de sortie Création d'un générateur de courant électrique ou thermique (dépend de la variable **cir**) dans les circuits électrique et thermique.

Invdelt_m(lin,col,vn1,vn2)

Calcule la différence de l'inverse des potentiels (électriques ou thermiques) entre chaque dipôle du réseau.

— Les variables d'entrée

1. **lin** : nombre de lignes dans le réseau.
2. **col** : nombre de colonnes dans le réseau.

3. **vn1** : il s'agit d'un vecteur qui contient des string correspondant au nom de chaque nœud du réseau : (**sn** pour le réseau électrique et **snt** pour le réseau thermique).
4. **vn2** : il s'agit d'un vecteur qui contient des float correspondant à la valeur du potentiel (électrique ou thermique) de chaque nœud du réseau (fn pour le réseau électrique et fnt pour le réseau thermique).

   Les deux vecteurs **vn1** et **vn2** sont symétriques en terme de position i.e. la position du nom de chaque noeud correspond dans le vecteur **vn1 :** retrouve sa valeur en la même position dans le vecteur **vn2**

— Les variables de sortie
Un tableau contenant la différence de l'inverse des potentiels entre chaque dipôle.

---

beta(a,b)

Calcule l'écart quadratique moyen entre deux vecteurs représentant une quantité physique de la $n-ième$ itération et la $(n-1)-ième$ itération.

— les variables d'entrée

1. **a** : vecteur de la $n-ième$ itération (potentiel électrique ou thermique).
2. **b** : vecteur de la $(n-1)ième$ itération (potentiel électrique ou thermique).

— Les variables de sortie
$\sum_{i,j}\left(a_{ij}-b_{ij}\right)^2$

---

chalfonc(x)

Netwk(lin, col, si, fi)

Cette fonction permet de transformer un tableau de la forme d'un vecteur 1D en une matrice de la taille du réseau. Elle est utilisée pour le potentiel électrique et la température.

— Les variables d'entrée :

1. **lin** : nombre de lignes dans le réseau.
2. **col** : nombre de colonnes dans le réseau.
3. **si** : c'est l'équivalent de la variable **vn1** pour la fonction **Invdelt_m**, il s'agit donc d'un tableau qui contient des éléments en format string correspondant au nom de chaque nœud du réseau (Dans le programme ils portent le nom de sn pour le réseau électrique et snt pour le réseau thermique).
4. **fi** : c'est l'équivalent de la variable **vn2** pour la fonction **Invdelt_m**, il s'agit donc d'un tableau qui contient des éléments en format float, correspondant à la valeur du potentiel (électrique ou thermique) de chaque nœud du réseau (fn pour le réseau électrique et fnt pour le réseau thermique).

   Les deux vecteurs **si** et **fi** sont symétriques en terme de position i.e. la position du nom de chaque nœud dans le vecteur **si** retrouve sa valeur dans la même position dans le vecteur **fi**.

— Les variables de sortie

Cette fonction renvoie à la sortie une matrice de la taille du réseau contenant les valeurs des potentiels. La position d'une valeur dans la matrice est la même que la position du noeud dans un circuit.

---

vtk_temp(mat)

La fonction crée un fichier **volt.vtk** permettant de visualiser les températures sur **Paraview**.

— Les variables d'entrée

1. **mat** : représente une matrice contenant les valeurs de la température en chaque nœud.

— Les variables de sortie

Un fichier **temp.vtk** qui représente les températures dans le réseau

— Modifications intéressantes

Les modifications intéressantes que nous pouvons faire sur le fichier, consistent en la redéfinition de l'espacement entre chaque maille dans le réseau, suivant l'axe des Y. La forme du matériau affiché est actuellement carrée, nous avons effectué une sorte normalisation dans la ligne **427** du fichier **foncdyco** :
c.write('\n\t'+**str**(**float**(**i**))+'\t'+**str**(**float**(−**j**∗**x**/**y**))+'\t'+**str**(**float**(**k**))) Afin de

retrouver les vraies dimensions du réseau il suffit d'enlever le terme ***x/y**

---

vtk_volt(mat)

La fonction crée un fichier **volt.vtk** permettant de visualiser les potentiels électriques sur **Paraview**

— Les variables d'entrée

1. **mat** : représente une matrice contenant les valeurs du potentiel électrique en chaque nœud.

— Les variables de sortie
Un fichier **volt.vtk** qui représente le potentiel électrique dans le réseau.

— Modifications intéressantes
Les modifications intéressantes que nous pouvons faire sur le fichier, consistent en la redéfinition de l'espacement entre chaque maille dans le réseau suivant l'axe des Y. La forme du matériau affiché est actuellement carrée, nous avons effectué une sorte normalisation dans la ligne **404** du fichier **foncdyco** :
c.**write**('\n\t'+**str**(**float**(**i**))+'\t'+**str**(**float**(−**j**∗**x**/**y**))+'\t'+**str**(**float**(**k**))) Afin de retrouver les vraies dimensions du réseau il suffit d'enlever le terme ***x/y**

---

carr(lin,col,vec)

La fonction agit sur les tableaux en 1D et les transforme en une matrice de la taille du réseau. Elle fait le même travail que la fonction **netwk**.

— Les variables d'entrée

1. **mat** : représente une matrice contenant les valeurs des potentiels (électriques ou thermiques).

— les variables de sortie
Un fichier **volt.vtk** .

---

vtk_IN(I)

La fonction crée un fichier **IN.vtk** permettant de visualiser le courant de particules sous Paraview. .

— Les variables d'entrée

1. **I** : représente une matrice contenant les valeurs des courants de particules. Ces courants sont placés dans la matrice selon leurs positions dans le réseau.

— les variables de sortie
Un fichier **IN.vtk**

---

vtk_IN_vc(I,Mat)

La fonction crée un fichier **IN_vc.vtk** permettant de visualiser le courant de particules sous Paraview. .

— Les variables d'entrée

1. **I** : représente une matrice contenant les valeurs des courants de particules.
2. **Mat** : Souvent c'est la matrice de la distribution des potentiels ; elle permet de soutirer les dimensions de mon réseau.

— les variables de sortie
Un fichier **IN_vc.vtk**. Les composantes suivant X et suivant Y sont superposées, de sorte à ne pouvoir visualiser qu'une composante à la fois.

---

vtk_IE(I)

La fonction crée un fichier **IE.vtk** permettant de visualiser le courant d'énergie sous Paraview. .

— Les variables d'entrée

1. **I** : représente une matrice contenant les valeurs des courants d'énergies. Ces courants sont placés dans la matrice selon leurs positions dans le réseau.

— les variables de sortie
Un fichier **IE.vtk**

---

vtk_IE_vc(I,Mat)

La fonction crée un fichier **IE_vc.vtk** permettant de visualiser le courant d'énergie sous Paraview. .

— Les variables d'entrée

1. **I** : représente une matrice contenant les valeurs des courants d'énergie.
2. **Mat** : Souvent c'est la matrice de la distribution des potentiels ; elle permet de soutirer les dimensions de mon réseau.

— les variables de sortie
Un fichier **IE_vc.vtk**. Les composantes suivant X et suivant Y sont superposées, de sorte à ne pouvoir visualiser qu'une composante à la fois.

---

vtk_IQ(I)

La fonction crée un fichier **IQ.vtk** permettant de visualiser le courant de chaleur sous Paraview.

— Les variables d'entrée

1. **I** : représente une matrice contenant les valeurs des courants de chaleur. Ces courants sont placés dans la matrice selon leurs positions dans le réseau.

— les variables de sortie
Un fichier **IQ.vtk** .

---

vtk_IQ_vc(I,Mat)

La fonction crée un fichier **IQ_vc.vtk** permettant de visualiser le courant de chaleur sous Paraview. .

— Les variables d'entrée

1. **I** : représente une matrice contenant les valeurs des courants de chaleur.

2. **Mat** :Souvent c'est la matrice de la distribution des potentiels ; elle permet de soutirer les dimensions de mon réseau.

— les variables de sortie
Un fichier **IQ_vc.vtk**. Les composantes suivant X et suivant Y sont superposées, de sorte à ne pouvoir visualiser qu'une composante à la fois.

---

vtk_Is(I)

La fonction crée un fichier **Is.vtk** permettant de visualiser le courant d'entropie sous Paraview.

— Les variables d'entrée

1. **I** : représente une matrice contenant les valeurs des courants d'entropie. Ces courants sont placés dans la matrice selon leurs position dans le réseau.

— les variables de sortie
Un fichier **Is.vtk**

---

vtk_Is_vc(I,Mat)

La fonction crée un fichier **Is_vc.vtk** permettant de visualiser le courant d'entropie sous Paraview.

— Les variables d'entrée

1. **I** : représente une matrice contenant les valeurs des courants d'entropie.
2. **Mat** :Souvent c'est la matrice de la distribution des potentiels ; elle permet de soutirer les dimensions de mon réseau.

— les variables de sortie
Un fichier **Is_vc.vtk**. Les composantes suivant X et suivant Y sont superposées, de sorte à ne pouvoir visualiser qu'une composante à la fois.

---

vtk_S(I)

La fonction crée un fichier **S.vtk** permettant de visualiser la création d'entropie dans le réseau sous Paraview.

— Les variables d'entrée

1. **I** : représente une matrice contenant les valeurs de la quantité de création d'entropie. Ces valeurs sont placées dans la matrice selon leurs position dans le réseau.

— les variables de sortie
Un fichier **S.vtk**

Annexe A

Exemple

Dans cette exemple, le schéma modélisé est comme suit :

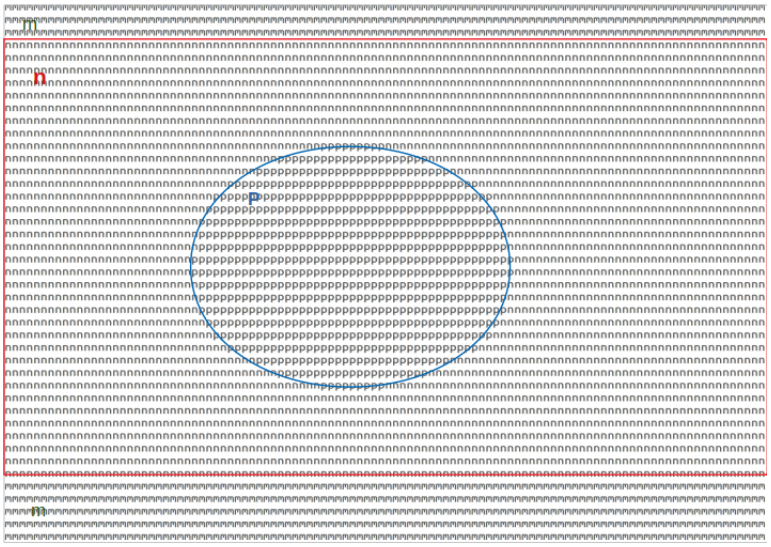


FIGURE A.1 – réseau étudié (La partie métalique est beacoup plus grande qu'elle n y parait. En raison du manque d'espace, nous n'avons pas pu tout inclure.)

Les conditions au bords sont comme suit :

— $\mu_1 = 0V$
— $\mu_2 = 0V$
— $T_1 = 350K$
— $T_2 = 300K$

Les paramètres des matériaux :

— $R_m = 10^{-6}\Omega$
— $R_p = 10^{-3}\Omega$
— $R_n = 10^{-3}\Omega$
— $\kappa_m = 0.5 W.K^{-1}$
— $\kappa_p = 20 W.K^{-1}$
— $\kappa_n = 20 W.K^{-1}$
— $\alpha_m = -5\mu V.K^{-1}$
— $\alpha_n = -200\mu V.K^{-1}$
— $\alpha_p = 200\mu V.K^{-1}$

Pour les précisions les valeurs sont comme suit :

— $precisionV = 10^{-10} V^2$
— $precisionT = 10^{-6} K^2$

Nous lançons le programme et nous lisons les fichier **.vtk** sous Paraview :

A.0.1 Potentiel électrique

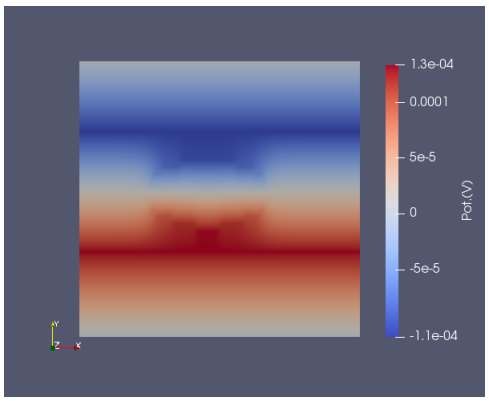


FIGURE A.2 – Résultats obtenus pour le champs de potentiel électrique

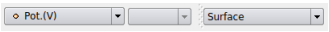


FIGURE A.3 – Configuration à éffectuer sur Paraview

A.0.2 Température

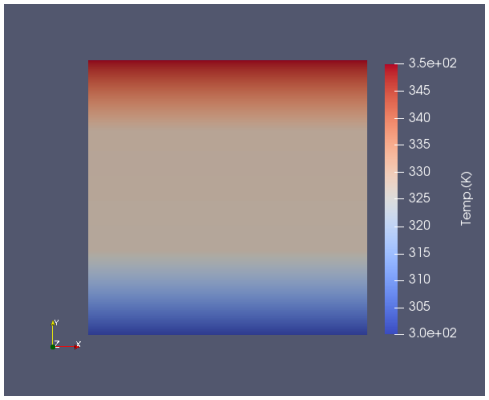


FIGURE A.4 – Résultats obtenus pour le champs de température

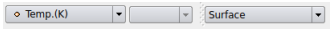


FIGURE A.5 – Configuration à éffectuer sur Paraview

A.0.3 Réseau de création d'entropie

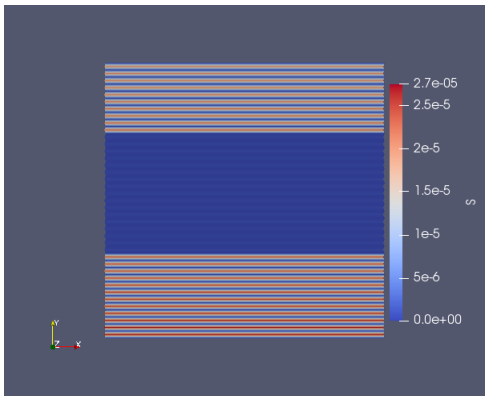


FIGURE A.6 – Résultats obtenu pour le réseau de création d'entropie (unité en SI)

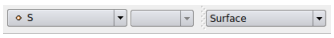
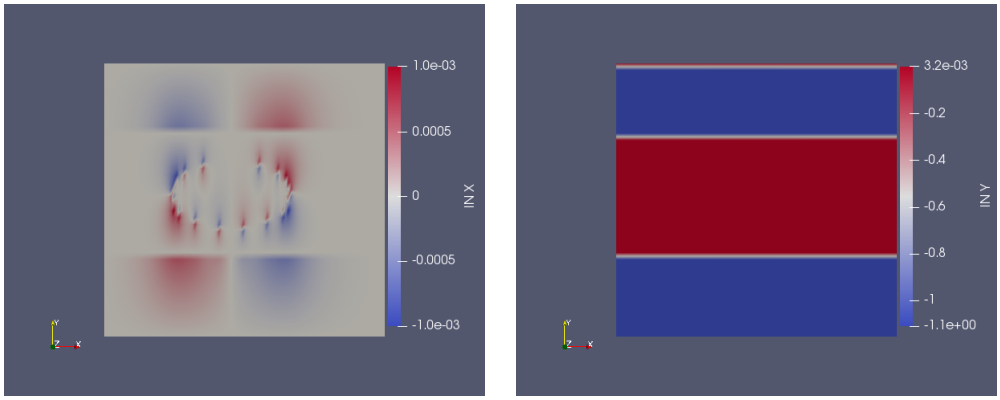


FIGURE A.7 – Configuration à éffectuer sur Paraview

A.0.4 Courant de particules

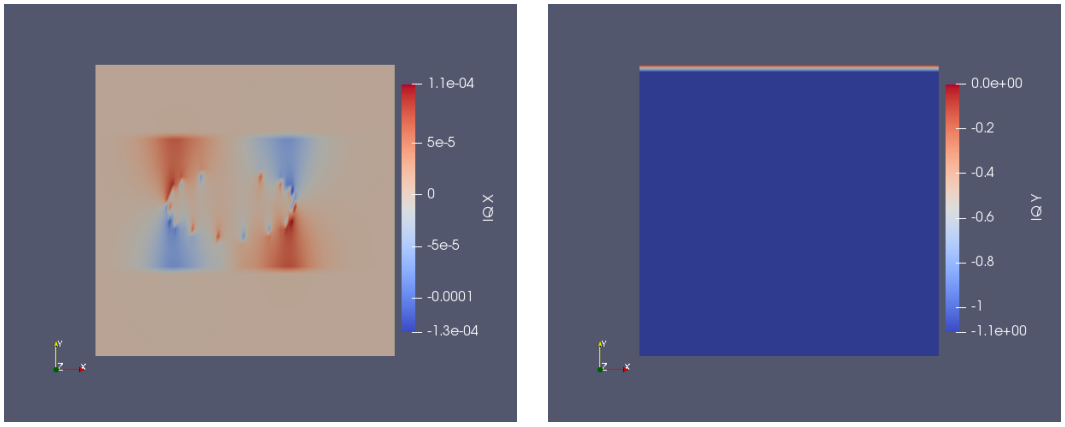


Suivant X Suivant Y

FIGURE A.8 – Courant de particiules Res : Gauche.suivant l'axe X; Droite. suivant l'axe Y (unité : SI)

FIGURE A.9 – Configuration à éffectuer sur Paraview

A.0.5 Courant de de chaleur

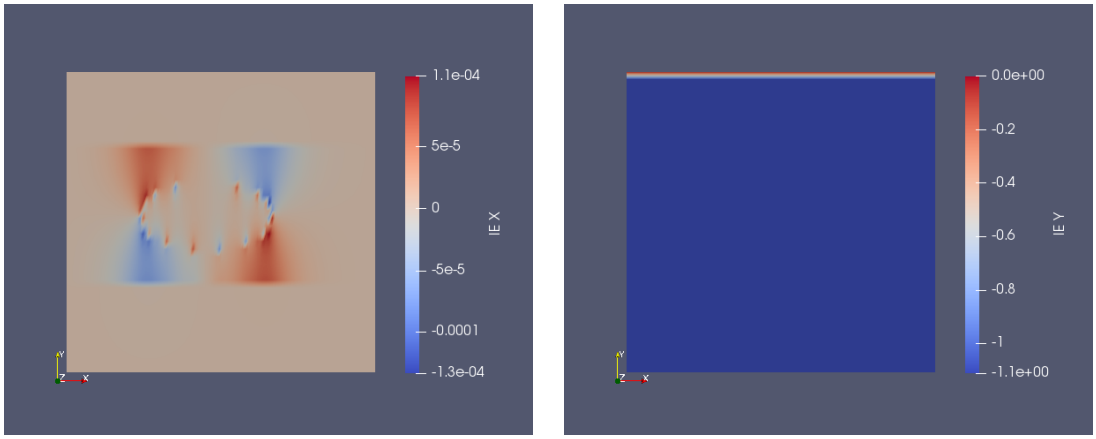


Suivant X Suivant Y

FIGURE A.10 – Courant de chaleur Res : Gauche.suivant l'axe X Droite. suivant l'axe Y (unité : SI)

FIGURE A.11 – Configuration à éffectuer sur Paraview

A.0.6 Courant d'énergie

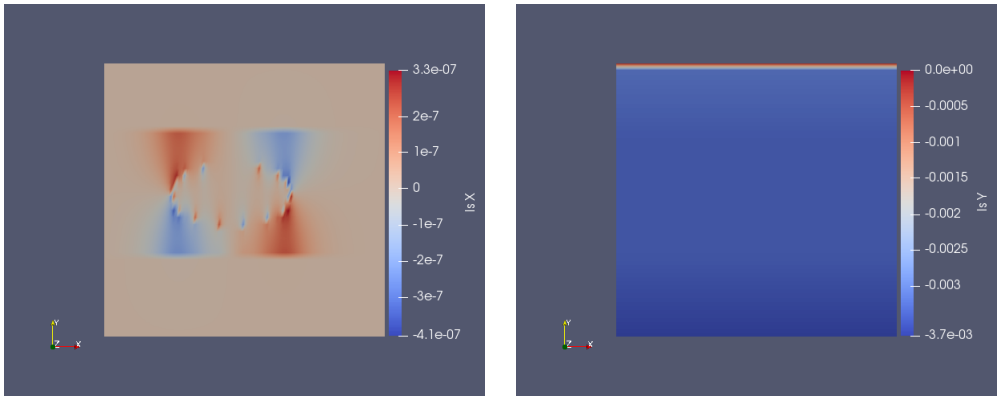


Suivant X Suivant Y

FIGURE A.12 – Courant d'énergie Res : Gauche.suivant l'axe X Droite. suivant l'axe Y (unité : SI)

FIGURE A.13 – Configuration à éffectuer sur Paraview

A.0.7 Courant d'entropie



Suivant X Suivant Y

FIGURE A.14 – Courant d'entropie, Res : Gauche.suivant l'axe X Droite. suivant l'axe Y (unité : SI)

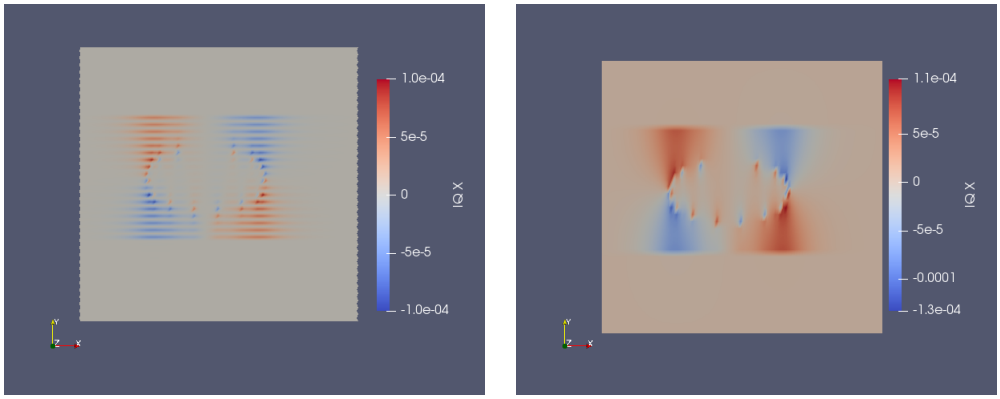
FIGURE A.15 – Configuration à éffectuer sur Paraview

A.0.8 comparaison entre un fichier vtk, portant l'extension vc et un fichier vtk simple

Les vecteurs sont enregistrés de deux manières sous Paraview.
La première, c'est en considérant notre circuit contenant des lignes alternées, avec des dipôles horizontales et des dipôles verticales. Chaque dipôle occupera une cellule...C'est les fichiers de courants simples.
La deuxième méthode consiste à inclure deux dipôles dans la même cellule : un dipôle horizontale et un dipôle verticale occuperons donc la même cellule. On pourra donc regarder qu'une seule composante à la fois (Soit selon X, soit selon Y). Il s'agit des fichiers de courants qui contiennent l'extension **_vc**.

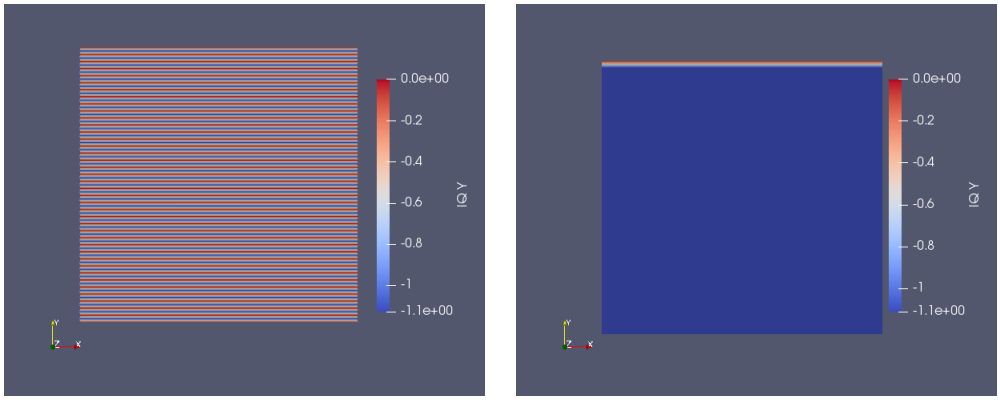
Illustration

Comparaison entre le fichier **IQ.vtk** et le fichier **IQ_vc.vtk** :



Fichier **IQ.vtk** Fichier **IQ_vc.vtk**

FIGURE A.16 – Comparaison entre le fichier **IQ.vtk** et le fichier **IQ_vc.vtk**, suivant l'axe X.



Fichier **IQ.vtk** Fichier **IQ_vc.vtk**

FIGURE A.17 – Comparaison entre le fichier **IQ.vtk** et le fichier **IQ_vc.vtk**, suivant l'axe Y.