Rapport Projet Maths

1 Introduction

Le but de ce projet math est de coder plusieurs fonctions (en C++) afin de résoudre l'équation de la Magnéto-Hydro-Dynamique modélisant l'évolution dun fluide conducteur dans un champ magnétique.

2 Contexte mathématiques

1. Le système d'équations de la Magnéto-Hydro-Dynamique (MHD) est :

$$\partial_t \begin{pmatrix} \rho \\ \rho u \\ Q \\ B \end{pmatrix} + \nabla . \begin{pmatrix} \rho u \\ \rho u \otimes u + (p + \frac{B.B}{2})I - B \otimes B \\ (Q + p + \frac{B.B}{2})u - (B.u)B \\ u \otimes B - B \otimes u \end{pmatrix} = 0, \tag{1}$$

$$Q = \rho e + \rho \frac{u.u}{2} + \frac{B.B}{2} \tag{2}$$

$$p = P(\rho, e) = (\gamma - 1)\rho e, \gamma > 1. \tag{3}$$

2. En utilisant la méthode proposée en 2002, ce système devient :

$$\partial_t \begin{pmatrix} \rho \\ \rho u \\ Q \\ B \\ \psi \end{pmatrix} + \nabla . \begin{pmatrix} \rho u \\ \rho u \otimes u + (p + \frac{B.B}{2})I - B \otimes B \\ (Q + p + \frac{B.B}{2})u - (B.u)B \\ u \otimes B - B \otimes u + \psi I \\ c_h^2 B \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

3. Le système de la MHD sous la forme d'une équation conservative donne :

$$\frac{\partial}{\partial t} W + \sum_{i=1}^{d} \frac{\partial}{\partial x_i} F^i(W) = 0.$$

   En considérant : $\partial \star = \frac{\partial}{\partial x_\star}$ et $\partial_i F^i(W) = \sum_{i=1}^{d} \frac{\partial}{\partial x_i} F^i(W)$.
   On obtient l'équation : $\partial_t W + \partial_i F^i(W) = 0$.

   Le vecteur de variables conservatives $\mathbf{W}$ est : $\mathbf{W} = (\rho, \rho\mathbf{u}, Q, \mathbf{B})^T$
   Le vecteur de variables primitives $\mathbf{Y}$ est : $\mathbf{Y} = (\rho, \mathbf{u}, \rho, \mathbf{B})^T$

4. Les fonctions *conservatives(real* Y, real* W)* et *primitives(real* Y, real* W)* permettent de calculer les variables conservatives à partir des variables primitives ou inversement. On applique la formule pour obtenir $\mathbf{p}$ ou $\mathbf{Q}$.

5. La fonction *flux(real* W, real* vn, real* flux)* utilise la formule pour calculer le flux suivant **W** et **n**, à partir du vecteur des variables conservatives et d'un vecteur n de l'espace. On utilise d'abord la fonction *primitives* pour obtenir le vecteur des variables primitives et pouvoir ensuite appliquer la formule.

3 Autre

1. La fonction *Ref2Phy(real* x, real *y, real *z, real *t)* permet d'obtenir les coordonnées *(z,t)* appartenant à [XMIN,XMAX]*[YMIN,YMAX], à partir de coordonnées *(x,y)* appartenant à $[0,1]^2$.
   Pour faire cela, il ne faut pas oublier de centrer la valeur en zéro :
   $z = (x - 0.5) * (X_{MAX} - X_{MIN}) + \frac{(X_{MIN}+X_{MAX})}{2}$.
   $t = (y - 0.5) * (Y_{MAX} - Y_{MIN}) + \frac{(Y_{MIN}+Y_{MAX})}{2}$.

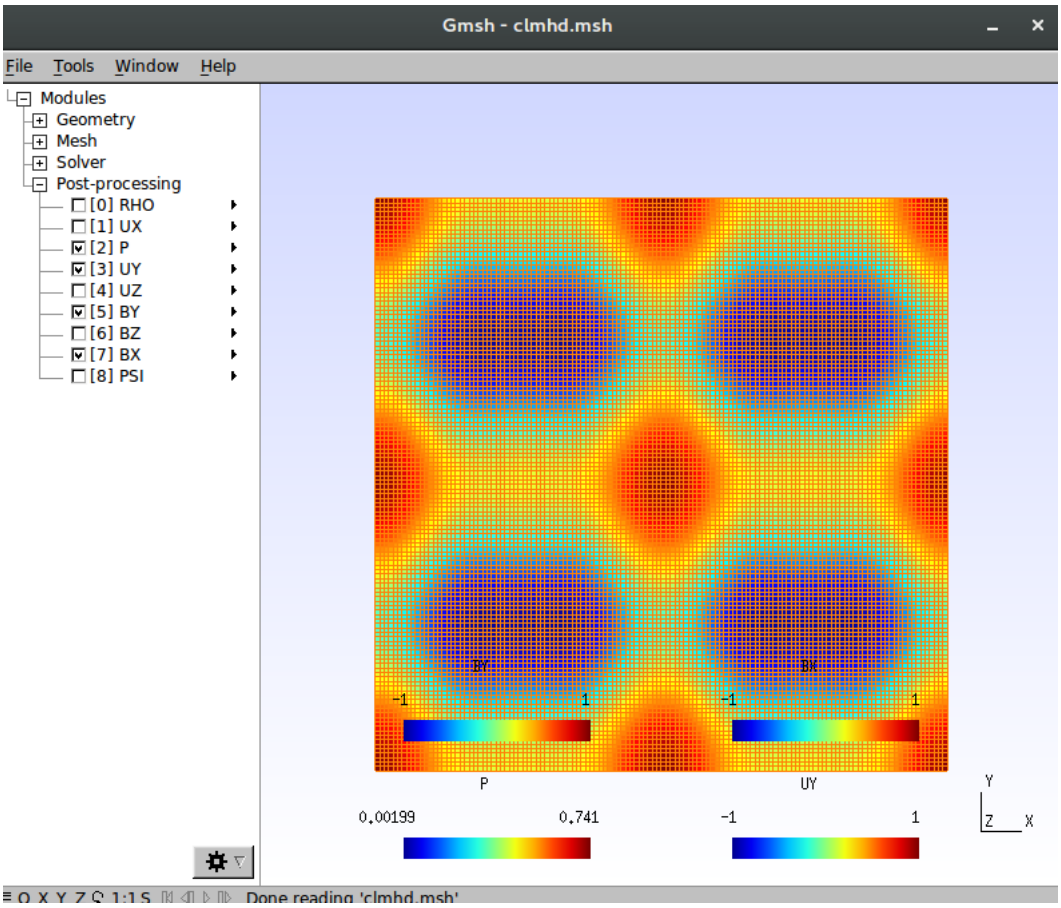
2. (a) Structure du tableau :
   Avant de commencer à remplir le tableau,il a fallu comprendre de quel tableau il s'agissait, et ce qu'il contenait : **w** est un (real *w), c'est-à- dire un tableau de réels à une dimension. Il s'agit en fait d'un tableau à plusieurs dimensions qui a été linéarisé.
   Il y a donc des cellules de coordonnées *(x,y)*, et pour chacune de ces cellules, on a 9 caractéristiques $(\rho, u1, p, u2, u3, B1, B2, B3, \psi)$.
   Le tableau a été linéarisé de la manière suivante : dans w[k* _NXTRANSBLOCK * _NYTRANSBLOCK + j * _NXTRANSBLOCK + i], se trouve la composante k de la cellule (i,j).
   Avec _NXTRANSBLOCK-1 la coordonnée maximale en x, _NYTRANSBLOCK-1 la coordonnée maximale en y, et _M le nombre de composantes d'une cellule *(x,y)*. Ces valeurs sont fixées à (128,128,9) dans le programme.

   Le choix de la méthode de linarisation de w qui consiste à dire que *"dans w[k*_NXTRANSBLOCK*_NYTRANSBLOCK + j * _NXTRANSBLOCK + i], se trouve la composante k de la cellule (i,j)"* a des défauts par rapport à une implémentation du type *"dans w[i*_NXTRANSBLOCK*_M + j*_M + k], on trouve la composante k de [i,j]"*.
   Premièrement, c'est assez contre-intuitif, de plus ce n'est pas performant :
   À chaque étape, on a des formules qui nous donnent les composantes d'une cellule *[x,y]* et on les stocke dans **w**. Dans la deuxième méthode, on doit remplir les cases du tableau qui sont successives, alors que dans la première méthode, on doit accéder à des cases très éloignées du tableau. Or, lorsque l'on accède à une case d'un tableau, plusieurs cases successives sont chargées en mémoire par souci d'optimisation. La deuxème méthode permettrait donc de profiter de cette optimisation.

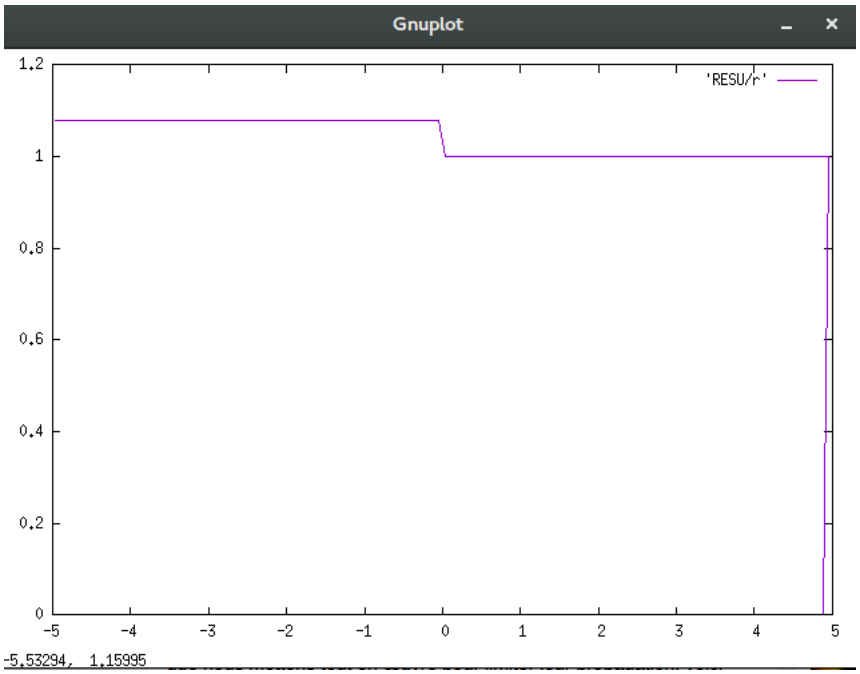
   (b) Algorithme.
   Le principe du programme est le suivant :
   Pour chaque cellule *(i,j)*, on calcule un couple *(i2,j2)* $= (\frac{i}{\_NXTRANSBLOCK}, \frac{j}{\_NYTRANSBLOCK})$.
   On applique ensuite *Ref2PhysMap* sur *i2* et*j2* (qui sont dans $[0,1]^2$),

pour obtenir *(x,y)* dans [xmin, xmax]*[ymin, ymax].
On appelle ensuite *Wexact* sur *(x,y)* et on stocke le résultat dans 'wtmp'.
On affecte finalement les valeurs de 'wtmp' dans 'w' grâce à la formule de linéarisation du tableau vue dans la partie 1.

(c) Résultat de gmsh après l'initialisation du tableau 2D :



3. La fonction *TimesStepCPU1D(real Wn1[_NXTRANSBLOCK*_NYTRANSBLOCK*_M], real* dtt)* récupère les variables conservatives à chaque itération n, afin de calculer la valeur à l'itération n + 1 en appliquant la formule.
Résultat de GnuPlot après exécution du programme en 1D :



4. La fonction *TimesStepCPU2D(real Wn1[_NXTRANSBLOCK*_NYTRANSBLOCK*_M], real* dtt)* ressemble à la fonction précédente, sauf que l'on s'intéresse également aux cases au-dessus et en-dessous.