Contrat Été 2023

CARNET DE BORD, UNIVERSITÉ MCGILL

RÉALISÉ DANS LE CADRE
D'UN PROJET POUR

ISMER–UQAR

21/04/2023

Rédaction
Charles-Édouard Lizotte
charles-edouard.lizotte@uqar.ca
ISMER-UQAR
Police d'écriture : **CMU Serif Roman**

Table des matières

1 MUDPACK

1.1 Mise en contexte

N.B. Nous sommes jeudi le 13 avril et je commence enfin à *jouer* avec la suite MUDPACK (Adams 1989). Malheureusement, je n'ai pas pu débuter cette tâche pour l'instant. Jusqu'à maintenant, les problèmes de modes barotropes et baroclines ont été plus graves que je croyais, mais tout semble enfin fonctionner. Les valeurs propres calculées analytiquement sont semblables à celles calculées numériquement.

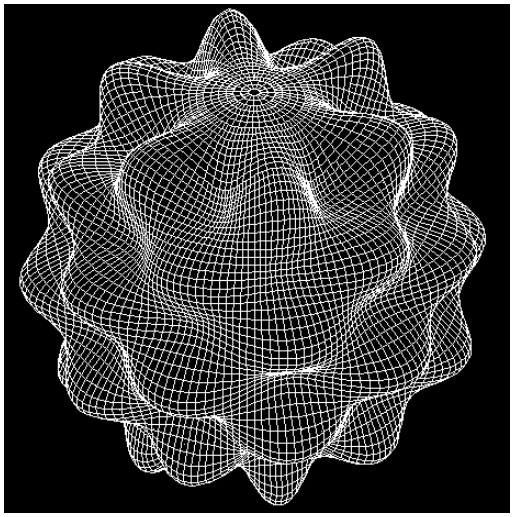

Figure 1 – *Logo MUDPACK*

MUDPACK (Adams 1989) est une suite FORTRAN en développement depuis les années 70. Son usage principal est de résoudre des équations différentielles partielles elliptiques, comme dans le cas qui nous intéresse. Elle fait partie des NCAR Classic Libraries for Geophysics et sont encore régulièrement utilisées par la communauté géophysique. L'accronyme NCAR fait référence à *National Center For Atmospheric Research*, on doute donc que la communauté géophysique tient les suite FORTRAN à bout de bras.

Toutes ces suites FORTRAN sont disponibles sur le GitHub de NCAR et on remarque qu'en date de la l'écriture de ce carnet, ces sous-routines ont été mises à jour il y a moins de deux ans. Je trouve ça exceptionnel pour un langage de programmation que beaucoup de gens trouvent « vieux jeux » et même désuet. Bref, contre toutes attentes, les « excursions » numériques de David et Louis-Philippe commencent à me plaire.

1.2 Téléchargement de la suite MUDPACK

Avant de débuter, on se doute qu'il faut – au même titre que la suite LAPACK – installer le module Fortran et lui assigner un *PATH*, car il n'est probablement pas déjà installé sur les versions usuelles de Ubuntu. À première vue, la suite MUDPACK est introuvable sur la toile et il existe très peu de documentation, mis à part quelques pages GitHub personnelles. Comme pour l'installation de LAPACK, je me demandais s'il y avait un moyen d'installer une archive avec une commande Debian du type

```
>>> sudo apt-get install libmudpack etc etc
```

Pourtant, non : Il n'existe tout simplement pas d'archive MUDPACK pour langage FORTRAN disponible sur les répertoires en ligne de l'installateur Aptitude. Devant ce problème, j'ai demandé l'aide de notre nouvel ami : ChatGPT, pour me trouver un moyen de l'installer. D'abord, ChatGPT a trouvé quelques versions du module mais ces dernières étaient purement obsolètes, dont celle d'un de ses développeurs vedette, Jack Dongarra. Un personnage fort intéressant d'ailleurs qui mériterait sa propre section. Comme le fichier *tar* sur lequel ChatGPT pointait était désuet, je lui ai surtout demandé de m'orienter sur le chemin à prendre pour télécharger ce genre de librairie FORTRAN. Je ne suis pas très accoutumé à télécharger et à travailler avec des fichiers *tar*, mais les étapes étaient simple à suivre.

Je suis donc revenu à l'archive GitHub précédente (celle de NCAR) qui contenait une version à jour de tous les librairies d'algèbre linéaire, dont MUDPACK. Une fois sur place, il est très simple de télécharger le fichier *tar* dans nos *Documents* à l'aide de la commande *wget*, soit

```
>>> cd ~\Documents
>>> wget https://github.com/NCAR/NCAR-Classic-Libraries-for-Geophysics/blob/main/MudPack/mudpack
5.0.1.tar.gz?raw=true -O mudpack5.0.1.tar.gz
```

où le lien est celui obtenu en cliquant « **View Raw** » sur le fichier dans l'archive GitHub. Une fois téléchargé, on peut ouvrir le *tar* à l'aide des commandes

```
>>> gunzip mudpack5.0.1.tar.gz
>>> tar xvf mudpack5.0.1.tar
>>> cd mudpack5.0.1
```

comme il est conseillé de faire dans le README en ligne.

1.3 Installation et compilation de la suite MUDPACK

Il sera nécessaire de constuire et compiler la suite à l'aide d'un *Makefile*. Nous allons donc modifer le fichier *bash make.inc* qui construira un *Makefile* à l'aide des bons compilateurs.

1. En premier lieu, on remplace *-fmod=* par *-J* partout. La commande *fmod* est désuette avec tous les compilateurs modernes, c'est une commande qui dit dans quel dossier mettre le modules issus de la compilation.
2. Deuxièmement, on remplace toutes les mentions à « pgf90 » et « g95 » par « gfortran », tout simplement parce que c'est notre compilateur.

Au final, le fichier devrait avoir une section principale qui ressemble à

```
ifeq ($(UNAMES),Linux)

  PGI := $(shell gfortran 2>&1)

  ifeq ($(PGI),gfortran-Warning-No files to process)

    F90 := gfortran -module ../lib -I../lib
    CPP := gfortran -E

  else

    F90 := gfortran -DG95 -g -J../lib -I../lib
    CPP := gfortran -E -DG95

  endif

  MAKE := gmake
  AR := /usr/bin/ar

endif
```

Bon ! C'est enfin le temps de compiler ! Dans le dossier principal, on exécute la commande *gmake*,

```
>>> gmake
```

Cette action devrait exécuter plusieurs centaines de lignes dans l'invite de commande, mais n'ayons pas peur. L'important à retenir : un nouveau fichier, soit une librairie statique, devrait avoir été crée dans le dossier *lib* : *libmudpack.a*. Ce fichier est une archive, on peut donc s'en servir et le lier à notre code. De manière générale, les librairies se retrouvent dans le répertoire des librairies, soit le même que LAPACK. C'est donc à cet endroit que nous allons créer un répertoire pour la librairie MUDPACK,

```
>>> cd /usr/lib/x86_64-linux-gnu/
>>> sudo mkdir mudpack
>>> cd mudpack
```

On copie la librairie *libmudpack.a* juste ici :

```
>>> sudo cp ~/Documents/mudpack5.0.1/lib/libmudpack.a .
```

Notre librairie maintenant installée, il faut lier notre compilateur à cette nouvelle librairie. Dans notre exécutable *bash*, on devrait avoir quelque chose qui ressemble à

```
#!/bin/bash
mudpack_path=/usr/lib/x86_64-linux-gnu/mudpack
lapack_path=/usr/lib/x86_64-linux-gnu/lapack
gfortran -o poisson-exec mudpack-test.f90 -L$mudpack_path -lmudpack
```

S'il n'y pas pas d'erreur lors de la compilation du code, il est possible de vérifier si la librairie a bien été liée à l'aide de la commande *ldd* sur notre exécutable. Cette commande nous fait essentiellement mention de toutes les librairies utilisées par l'exécutable.

```
>>> ldd poisson-exec
linux-vdso.so.1 (0x00007ffd39130000)
libgfortran.so.5 => /lib/x86_64-linux-gnu/libgfortran.so.5 (0x00007fa3edddb000)
libm.so.6 => /lib/x86_64-linux-gnu/libm.so.6 (0x00007fa3edcf4000)
libc.so.6 => /lib/x86_64-linux-gnu/libc.so.6 (0x00007fa3edacc000)
libquadmath.so.0 => /lib/x86_64-linux-gnu/libquadmath.so.0 (0x00007fa3eda84000)
libgcc_s.so.1 => /lib/x86_64-linux-gnu/libgcc_s.so.1 (0x00007fa3eda64000)
/lib64/ld-linux-x86-64.so.2 (0x00007fa3ee0fc000)
```

2 Conditions frontières et schéma numérique

2.1 Mise en contexte

Pour débuter, nous voulons rajouter des murs aux frontières de notre expérience numérique. Par contre, ceci nous empêche d'emprunter le solveur elliptique précédemment utilisé dans le modèle à deux couches. Ce dernier fonctionnait avec des transformées de Fourrier, il aurait donc fallu créer des réflexions en x et y pour rendre les frontières continues sur un plus grand domaine. Et les réflexions auraient été inverses dans certains cas. Par exemple, la réflexion en x du courant en v aurait été une réflexion négative, tandis que la réflexion en x du courant en u aurait été une réflexion positive pour s'assurer de la continuité dans les 4 quadrants. De plus, on aurait surement souffert du phénomène de Gibbs aux discontinuités. C'est pourquoi nous avons oublié cette idée.

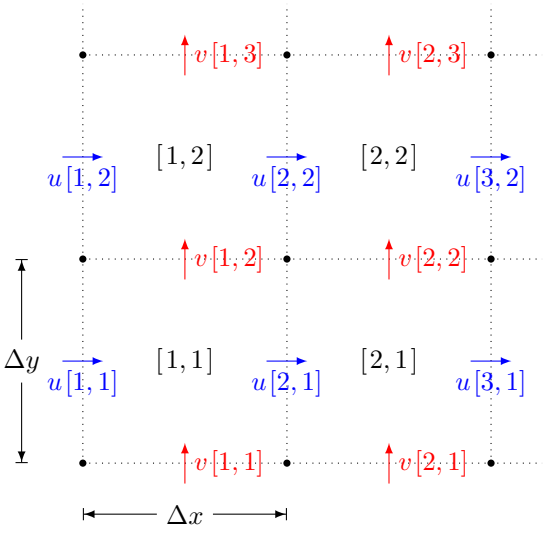

FIGURE 2 – *Représentation de la grille numérique utilisée pour le modèle en eau peu profonde (type Arakawa-C)*

Comme mentionné précédemment, nous utiliserons le solveur elliptique de la suite MUDPACK (ADAMS 1989). Ce solveur utilise plutôt des fonctions de Green ou une technique de relaxation pour résoudre les équations différentielles partielles du second ordre. Par contre, il faudra ajuster les conditions limites de sorte à ce que le courant, sa dérivée première et sa dérivée seconde soient définit aux frontières.

N.B. Mentionnons que le nombre de conditions frontières nécessaires augmente directement avec l'ordre de l'équation différentielle que nous tentons de résoudre.

2.2 Conditions frontière sur les courants (No normal flow)

Pour les courants qui traversent les murs, on applique la condition *no normal flow*. La condition *no normal flow* est une condition de type Dirichlet qui est caractérisée par un courant normal nul aux frontières, bref comme si le

fluide *adhérait* aux murs. C'est donc une condition d'imperméabilité. Mathématiquement, la condition se traduit par

$$\mathbf{u} \cdot \hat{\mathbf{n}} = 0, \tag{2.1}$$

où $\hat{\mathbf{n}}$ est le vecteur normal à la frontière. Numériquement, on peut énoncer que sur une grille cartésienne la condition *no normal flow* symbolise

$$\text{(Frontières verticales)} \qquad u[1,:] = 0 \qquad \text{et} \qquad u[:,nx] = 0, \tag{2.2a}$$
$$\text{(Frontières horizontales)} \qquad v[:,1] = 0 \qquad \text{et} \qquad v[:,ny] = 0, \tag{2.2b}$$

N.B. Si nous utilisons des points fantômes, alors on peut étendre les extrémités des frontières et affirmer que ces derniers sont reliés par les relations

$$\text{(Courant } u\text{)} \qquad u[:,0] = u[:,1] \qquad \text{et} \qquad u[:,ny+1] = u[:,ny], \tag{2.3a}$$
$$\text{(Courant } v\text{)} \qquad v[0,:] = v[1,:] \qquad \text{et} \qquad v[nx+1,:] = v[nx,:]. \tag{2.3b}$$

2.3 Conditions frontières sur la dérivée première (Free slip condition)

Avant tout, mentionnons qu'on fait régulièreement référence à un concept appelé la *no slip condition*. Cette condition se caractérise par l'absence de courant tangeant au mur. Comme le courant normal aux frontières est généralement nul dans les cas à l'étude (*no normal flow*), la *no slip condition* réfère généralement au fait qu'aucun fluide ne bouge au mur.

Par contre, on s'intéresse aujourd'hui à la *free slip condition*. La *free slip condition* tient à l'hypothèse que la couche limite est si petite qu'on peut essentiellement l'ignorer, ce qui est souvent le cas pour l'étude des fluides à grande échelle. Concrétement, il n'y a pas de contrainte de cisaillement au mur, de sorte que

$$\left(\boldsymbol{\tau}_x = \mu \frac{\partial u}{\partial y}\right)\Bigg|_{\{xi,xf\}} = 0\,, \qquad \text{et} \qquad \left(\boldsymbol{\tau}_x = \mu \frac{\partial u}{\partial y}\right)\Bigg|_{\{yi,yf\}} = 0\,. \tag{2.4}$$

où μ est la viscosité (Tan 2018). Ainsi, l'expression 2.4 force la condition frontière sur la dérivée première à satisfaire

$$\boxed{\frac{\partial v}{\partial x}\Bigg|_{\{xa,xf\}} = 0 \;\forall\; y, \qquad \text{et} \qquad \frac{\partial u}{\partial y}\Bigg|_{\{yi,yf\}} = 0 \;\forall\; x.} \tag{2.5}$$

2.4 Calcul de la dérivée seconde en présence de murs

Pour calculer la dérivée seconde, on a besoin de trois points, de sorte que la courbure de notre champ est donnée par

$$\frac{\partial^2 u}{\partial x^2}[i,:] = \frac{u[i-1,:] + u[i+1,:] - 2u[i,:]}{\Delta x^2}. \tag{2.6}$$

Et dans notre schéma numérique, la seconde dérivée se positionne au même point que la quantité choisie, le courant en u dans notre cas.

Par contre, aux frontières, une partie de cette information est cachée derrière le mur. Il faut donc trouver un moyen détourné d'obtenir la dérivée seconde. David a proposé une méthode assez intéressante impliquant les série de Taylor. On peut réaliser deux séries autour des points aux frontières normales. Par exemple, on peut prendre $u[1,:]$ et étendre notre série autour de $u[2,:]$ et $u[3,:]$ pour obtenir

$$u[2,:] = u[1,:] + (\Delta x)\; u'[1,:] + (\Delta x)\frac{u''[1,:]}{2} + \mathcal{O}(3), \tag{2.7a}$$
$$u[3,:] = u[1,:] + (2\Delta x)u'[1,:] + (2\Delta x)^2\frac{u''[1,:]}{2} + \mathcal{O}(3). \tag{2.7b}$$

Maintenant, on soustrait les deux équations de sorte à éliminer $u[1,:]$ et retrouver la dérivée suivante, soit

$$\underbrace{u[3,:] - u[2,:]}_{u'[2,:]} = \Delta x \; u'[1,:] + \frac{3\Delta x^2}{2} u''[1,:] \tag{2.8}$$

et on aboutit à l'expression

$$\boxed{u''[1,:] = \frac{2}{3\Delta x}\big(u'[2,:] - u'[1,:]\big).} \tag{2.9}$$

Le même principe est applicable à chaque frontière et dans chaque direction.

2.5 Étude des conditions frontières et lien avec MUDPACK

Pour chaque frontière, il est possible d'appliquer une condition différente sur la fonction ϕ. L'usager a pour sa part trois choix :

- ⇨ Le domaine est périodiques sur la frontière (*iparm(2) = 0*) ;
- ⇨ La fonction à déterminer (ϕ) est définit aux frontières (Dirichlet) (*iparm(2) = 1*) ;
- ⇨ On spécifie la valeur de la dérivée normale à la frontière (Neumann) et/ou on définit des conditions frontières mixtes (*iparm(2) = 2*).

Dans le cas à l'étude, une condition frontière de type Neumann est préférable. Il est possible de définir des conditions frontières mixtes (Neumann et Dirichlet), mais ce ne sera pas nécessaire pour notre grille.

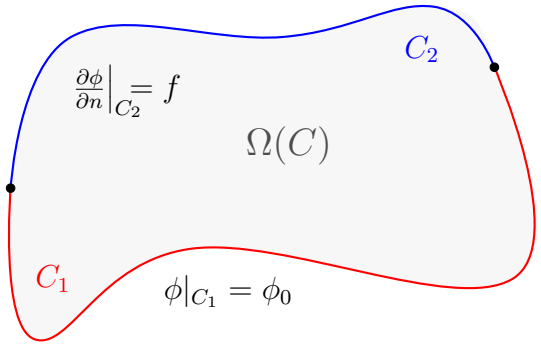

FIGURE 3 – *Condition frontière mixte autour d'une région Ω borné par la courbe $C = C_1 \cup C_2$.*

Prenons en exemple la frontière ouest. Toujours selon la documentation de MUDPACK, les conditions frontière mixtes ont la forme,

$$\partial\phi/\partial x + \alpha(y) * \phi(xa, y) = f(y) \quad \forall \quad y \in [y_i, y_f], \tag{2.10}$$

où $\phi(x, y)$ est la solution à déterminer. Les fonctions $\alpha(y)$ et $f(y)$ restent donc à déterminer pour notre problème aux conditions frontières.

On sait que l'évolution du fluide est décrit par le système d'équations

$$u(t+1) = u(t) + \Delta t G_x(x, y, t) + \partial\phi/\partial x \,, \tag{2.11a}$$
$$v(t+1) = v(t) + \Delta t G_y(x, y, t) + \partial\phi/\partial y \,, \tag{2.11b}$$

où les termes $G_{x,y}$ sont des termes valisent qui englobent tout le côté droit de nos équations du mouvement dans chaque direction (Coriolis, advection horizontale, gradient de la fonction de Bernouilli, etc).

Au mur ouest, on impose la condition d'imperméabilité (*no normal flow*) sur l'équation 2.11a de sorte que $u(x_0, y, t) = 0$ et

$$\left.\frac{\partial\phi}{\partial x}\right|_{x_0} = -\Delta t G_x(x_0, y, t), \quad \Longrightarrow \quad \boxed{f(y) = -\Delta t G_x(x, y, t) \quad \& \quad \alpha(y) = 0} \tag{2.12}$$

Un lecteur avisé serait tenté de vérifier si $G_x = 0$ pour se simplifier la vie. Vérifions l'état des fonctions $G_{x,y}$ à la frontière ouest,

$$G_x(x_0, y) = \underbrace{u \cdot \cancelto{0}{\left(\frac{\partial u}{\partial x}\right)} + v \cdot \cancelto{0}{\left(\frac{\partial u}{\partial y}\right)}}_{\text{Advection}} \underbrace{+ fv}_{\text{Coriolis}} \underbrace{+ \frac{\tau_x}{\rho_o}}_{\text{Vent}} \underbrace{+ D_x}_{\text{Dissip.}}. \tag{2.13}$$

On peut aisément retirer les termes d'avection, étant donné que $u(x_0, y) = \partial u/\partial y\,|_{x_0} = 0 \; \forall \; y$. Malheureusement, les termes $G_{x,y} \neq 0$. Par contre, on note qu'il y aura l'établissement d'un équilibre entre les termes sources et le gradient de pression $\boldsymbol{\nabla}\phi$.

Références

ADAMS, John C (1989). « MUDPACK: Multigrid portable FORTRAN software for the efficient solution of linear elliptic partial differential equations ». In : *Applied Mathematics and Computation* 34.2, p. 113-146.

TAN, Hua (2018). « Applying the free-slip boundary condition with an adaptive Cartesian cut-cell method for complex geometries ». In : *Numerical Heat Transfer, Part B: Fundamentals* 74.4, p. 661-684. URL : https://par.nsf.gov/servlets/purl/10108399.