École normale supérieure
L3 geosciences
Travaux dirigés de Fortran – Second semestre

Lionel GUEZ

18 mai 2022

Table des matières

Rappel : si vous avez besoin de consulter la norme Fortran 2003, vous pouvez trouvez sa version quasi-définitive en ligne : `https://wg5-fortran.org/N1601-N1650/N1601.pdf`. Notamment, la description des procédures intrinsèques est au § 13.7, page 316.

1 Écriture et appel de fonction, programme à deux fichiers

Écrivez un programme qui demande à l'utilisateur d'entrer une suite de trois caractères et qui indique si les caractères sont dans l'ordre ASCII croissant, décroissant, ou dans le désordre. Le programme redemandera à l'utilisateur d'entrer trois nouveaux caractères jusqu'à ce que l'utilisateur entre trois étoiles : ***. Utilisez une fonction logique qui retourne vrai si une suite de trois caractères est dans l'ordre croissant. Fonction intrinsèque utile : lle.

2 Écriture et appel de subroutine, argument tableau

Écrivez un programme qui affiche la forme représentée sur la figure (1), à l'aide de simples caractères, par exemple des étoiles : *. Utilisez une procédure

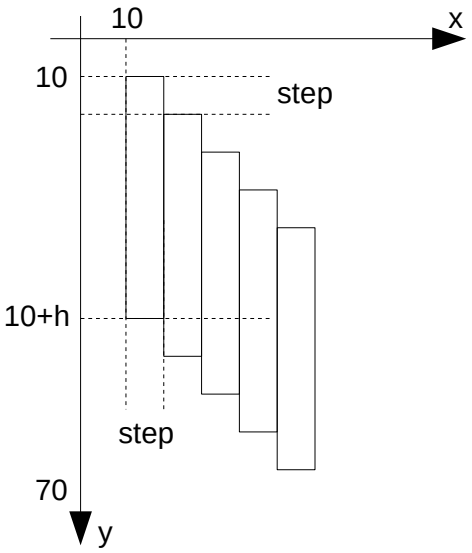

FIGURE 1 – Dessin de rectangles.

qui ajoute un rectangle unique. Respectez la paramétrisation et la spécification de cette procédure choisies dans le cours d'algorithmique : les arguments sont la position du coin inférieur gauche, la largeur et la hauteur. Testez dans la procédure que le rectangle ne déborde pas de la figure.

3 Utilisation d'une procédure de bibliothèque

La procédure quadrat de la bibliothèque Jumble trouve les racines réelles d'une équation du second degré à coefficients réels. Écrivez un programme qui demande à l'utilisateur des valeurs des coefficients, appelle `quadrat` et écrit les résultats. S'il existe une racine double, le programme l'indiquera et n'affichera qu'une valeur.

Le module donnant accès à la bibliothèque Jumble a pour nom `jumble`. La bibliothèque `Jumble` est installée sur les machines du 4^{e} étage dans le répertoire :

```
/home/guez/.local/lib
```

et l'interface compilée du module de la bibliothèque est dans le répertoire :

```
/home/guez/.local/include
```

4 Instructions open, write, close ; pseudo-boucle

Dans cet exercice, vous allez écrire quelques statistiques sur les nombres premiers.

Partez du programme du premier semestre qui trouve la liste des nombres premiers inférieurs à un entier n, à l'aide du crible d'Ératosthène.

1. Faites en sorte que les sorties du programme soient enregistrées dans un fichier au lieu d'être seulement affichées à l'écran, sans modifier le programme en Fortran.
2. Modifiez le programme pour qu'il écrive :
   — à l'écran, le nombre de nombres premiers inférieurs à n ;
   — à l'écran, la liste de ces nombres premiers, seulement s'il y en a moins de 30 ;
   — à l'écran, l'intervalle moyen entre deux nombres premiers inférieurs à n ;
   — un fichier texte, que vous appellerez `prime_intervals.txt` contenant la liste des différences entre deux nombres premiers successifs, une valeur par ligne, sans ligne de titre ;
   — à l'écran, un message annonçant la création d'un fichier contenant les intervalles.

   Est-il possible de simplement rediriger la sortie standard au lieu de créer un fichier dans le programme ? Utilisez la procédure new_unit de la bibliothèque Jumble. Fonction intrinsèque utile : pack. Compilez et testez votre programme avec de petites valeurs de n.
3. Visualisez l'histogramme des différences entre nombres premiers successifs avec le logiciel de votre choix (Matplotlib, Gnuplot ...) pour une grande valeur de n, par exemple 10^7. Dans Matplotlib par exemple, si vous avez lu les valeurs des intervalles dans un tableau `x` alors la commande :

   ```
   plt.hist(x, bins = np.arange(x.max() + 1) + 0.5, log = True)
   ```

   affiche l'histogramme avec un axe des ordonnées logarithmique. Vous devriez observer une décroissance linéaire de l'histogramme.

5 Lecture d'un fichier, nombre de lignes connu

Le fichier recensement.txt contient pour chaque département (colonne 1) la population en 1999 (colonne 2) et celle en 1990 (colonne 3) (source INSEE). Vous pouvez voir que les 95 premiers départements sont numérotés de 1 à 95 et les 4 derniers de 971 à 974.

1. Écrivez un programme qui enregistre ces données dans deux tableaux : un vecteur pour les numéros des départements, et un tableau à deux dimensions pour les valeurs de population. On pourra supposer dans le programme que le nombre de départements dans `recensement.txt` est de 99. Lisez le fichier simplement en redirigeant l'entrée standard. Le programme affichera le numéro des départments dont l'évolution de la population entre 1990 et 1999 est, en valeur absolue, supérieure à 5 %.

   Procédure intrinsèque Fortran pouvant vous être utile : `abs`.

   Vous devriez obtenir la liste suivante de départements dont la population a évolué de plus de 5 % : 1 4 5 15 17 23 26 27 30 31 33 34 35 38 40 44 45 60 66 67 68 73 74 77 83 84 85 86 95 971 972 973 974.

2. Modifiez le programme pour qu'il demande en outre à l'utilisateur de saisir un numéro de départment et affiche l'évolution de sa population et son rang en terme de nombre d'habitants en 1999 et 1990. On pourra supposer dans le programme que les départements apparaissent dans `recensement.txt` dans l'ordre croissant de leurs numéros. Est-il toujours possible de simplement rediriger l'entrée standard pour lire le fichier ? Nota bene : le chemin du fichier lu ne doit pas être écrit dans le programme (c'est un conseil de style de programmation, notamment pour la portabilité du programme), vous aurez donc à copier le fichier `recensement.txt` dans votre répertoire ou à créer un lien symbolique.

   Procédure intrinsèque Fortran pouvant vous être utile : `count`.

6 Lecture d'un fichier, nombre de lignes inconnu

(Adapté de Holton, 2004, *An Introduction to Dynamic Meteorology*.)

Téléchargez le fichier tropical_temp.csv. Vous pouvez voir le contenu de ce fichier avec par exemple un tableur comme celui de LibreOffice (ou avec un éditeur de texte quelconque, ou avec la commande cat). Ce fichier contient des valeurs de température T à différents niveaux de pression p, qui proviennent de sondages atmosphériques dans la région tropicale.

Le but de cet exercice est d'écrire un programme qui calcule à partir de ces données les profils correspondants de la hauteur de géopotentiel Z, de la température potentielle θ et du paramètre de stabilité statique :

$$S_p := -\frac{T}{\theta}\partial_p\theta$$

Vous calculerez Z en utilisant l'équation hypsométrique. Le programme demandera à l'utilisateur d'entrer la valeur de Z au niveau de pression le plus élevé.

Rappels. En notant Φ le géopotentiel et R la constante massique des gaz parfaits pour l'air sec :

$$R = 287\,\mathrm{J\ K^{-1}\ kg^{-1}}$$

l'équation hypsométrique s'écrit :

$$\partial_p \Phi = -\frac{RT}{p}$$

On a :

$$Z = \Phi / g_0$$

avec :

$$g_0 = 9{,}806\,65\,\mathrm{m\ s^{-2}}$$

La température potentielle est :

$$\theta = T(p_{\mathrm{ref}}/p)^{\kappa}$$

avec :

$$p_{\mathrm{ref}} = 1\,000\,\mathrm{hPa}$$
$$\kappa = 2/7$$

Indications.

— Pour Z, intégrez l'équation hypsométrique entre deux niveaux de pression et calculez l'intégrale en supposant que T varie à peu près linéairement avec le logarithme de la pression entre les deux niveaux de pression.

— Pour S_p, vous approximerez la dérivée de θ au niveau de pression d'indice i par la différence finie :

$$(\partial_p \theta)_i \approx \frac{\theta_{i+1} - \theta_{i-1}}{p_{i+1} - p_{i-1}}$$

Si le tableau des pressions est indicé de 1 à n, le tableau contenant les valeurs de S_p aura donc une plage d'indices de 2 à $n - 1$.

Appelez la procédure count_lines de la bibliothèque Jumble pour compter le nombre n de couples (p, T) dans le fichier `tropical_temp.csv`. Est-il possible de simplement rediriger l'entrée standard pour lire le fichier ?

Calculez la température potentielle, en utilisant une affectation de tableau à tableau (n'utilisez pas de boucle).

Créez un nouveau fichier avec un suffixe `.csv` contenant les colonnes p, Z, θ et S_p. Pensez à mettre une première ligne de titres dans le fichier. Séparez les valeurs par des espaces. Aux premier et dernier niveau de pression, comme S_p n'est pas défini, écrivez la chaîne de caractère « NaN » à la place dans la colonne correspondante.

Est-il possible de simplement rediriger la sortie standard pour créer le fichier ?

Visualisez les résultats avec le logiciel graphique de votre choix (Gnuplot, Matplotlib, Grace…). Vous devriez obtenir les figures 2 et 3.

7 Namelist, combinaison Fortran – Bash – visualisation

Écrivez un programme qui permette d'enregistrer dans un fichier les termes de la suite (x_n, y_n) définie par la récurrence suivante :

$$x_{n+1} = b y_n + f(x_n)$$
$$y_{n+1} = -x_n + f(x_{n+1})$$

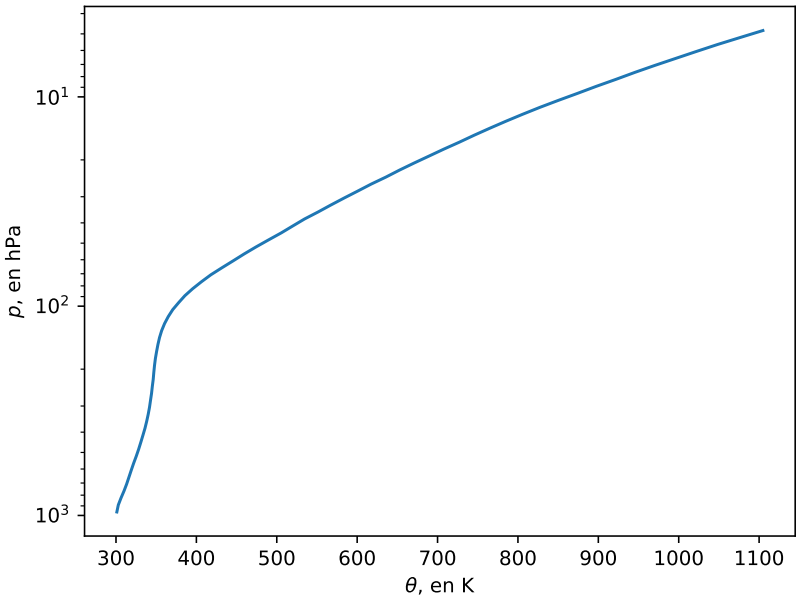

FIGURE 2 – Exercice 6, température potentielle.

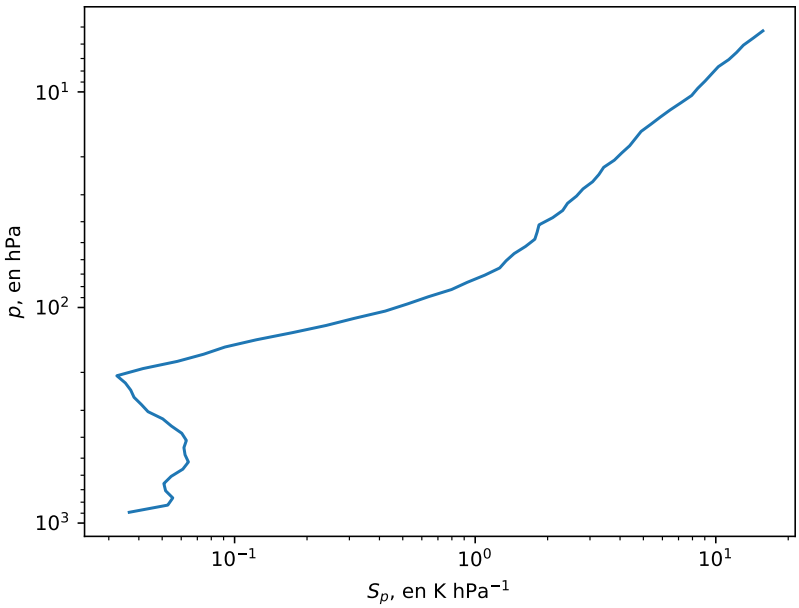

FIGURE 3 – Exercice 6, paramètre de stabilité statique.

avec :

$$f(x) = ax + \frac{2(1-a)x^2}{1+x^2}$$

et les constantes $a = -0{,}6$ et $b = 0{,}99$. L'utilisateur doit pouvoir choisir à l'exécution la condition initiale (x_0, y_0) et l'indice final n. Programmez l'entrée de ces trois variables dans une namelist. Choisissez $x_0 = 0{,}09$, $y_0 = 2{,}76$ et, par exemple, $n = 10$, comme valeurs par défaut.

Lancez l'exécution avec les valeurs par défaut de x_0, y_0 et n. Visualisez la position des points (x_i, y_i) pour $i = 0, \ldots, n$ avec le logiciel de votre choix (Matplotlib, Gnuplot ...). Avec Matplotlib par exemple, utilisez les options de plot :

```
marker = ",", linestyle = " "
```

pour tracer seulement un pixel par couple (x_i, y_i) et pour que les pixels ne soient pas reliés entre eux.

Écrivez un script en Bash, qui prend en argument n sur la ligne de commande, et qui exécute le programme en Fortran et trace le graphique pour cette valeur de n. Le script doit donc fournir à l'exécutable Fortran la namelist contenant la valeur de n choisie.

Avec le script, testez des valeurs croissantes de n (essayez par exemple des puissances de 10 successives). De manière analogue, vous pouvez aussi tester l'influence des valeurs de x_0 et y_0.

Rappel du cours sur Unix. Vous aurez besoin d'enregistrer dans le script ce qui doit normalement être lu au clavier, pour le programme en Fortran et éventuellement pour le logiciel graphique. Pour cela, vous pouvez utiliser la technique suivante :

```
./mon_programme <<EOF
ce que j'écrirais normalement au clavier
EOF
```

8 Double précision, argument optionnel, programme à trois fichiers sources

Écrivez un programme qui :

— demande à l'utilisateur sa date de naissance exacte (jour et heure, à la seconde près) ;

— retourne à l'écran la durée écoulée depuis sa naissance en jours, heures, minutes et secondes.

Une solution économique consiste à utiliser la date julienne. La date julienne est le nombre de jours depuis le 1er janvier 4713 avant J.-C., à 12 h, temps universel, dans un calendrier proleptique grégorien. (Elle est utilisée en astronomie.) Cf. par exemple Is there a formula for calculating the Julian Day?.

Écrivez une procédure qui reçoit une date quelconque sous la forme année, mois, jour (nombres entiers) et heure avec fraction d'heure (un nombre réel), et qui renvoie la date julienne (nombre réel). Vous écrirez aussi une procédure qui transforme un triplet d'entiers, heures, minutes, secondes en un nombre réel heure avec fraction d'heure.

Procédures intrinsèques Fortran utiles : `date_and_time`, floor, dble. Nota bene : `date_and_time` a des arguments optionnels.

9 Argument procédure, variable de module,

(Adapté de Holton, 2004, *An Introduction to Dynamic Meteorology.*)

Dans le plan f, nous considérons un écoulement en équilibre hydrostatique vertical, avec un champ de géopotentiel idéalisé donné par :

$$\Phi(x, y, p, t) = \Phi_0(p) + \Phi' \sin[k(x - ct)] \cos ly$$

où Φ', k, c, l sont des constantes strictement positives et Φ' est petit par rapport à Φ_0. (« An idealized representation of a midlatitude synoptic disturbance in an atmosphere with no zonal mean flow. ») Le géopotentiel est donc la somme d'une fonction de la pression $\Phi_0(p)$ et d'une petite perturbation ondulatoire sinusoïdale. c est la vitesse de phase de l'onde (dans la direction zonale), k et l sont les nombres d'ondes dans les directions x et y.

Nous supposons que l'écoulent est en équilibre géostrophique :

$$\begin{cases} u = -\frac{1}{f}\partial_y \Phi \\ v = \frac{1}{f}\partial_x \Phi \\ w = 0 \end{cases} \tag{1}$$

En posant :

$$U' = \Phi' k / f$$

le système 1 donne :

$$\begin{cases} u = \frac{l}{k} U' \sin[k(x - ct)] \sin ly \\ v = U' \cos[k(x - ct)] \cos ly \end{cases}$$

Nous supposons $f > 0$ donc aussi $U' > 0$.

La divergence isobare du vent géostrophique est nulle et le vent géostrophique ne dépend pas ici du niveau de pression donc on peut définir une fonction de courant $\psi(x, y, t)$ telle que :

$$u = \partial_y \psi$$
$$v = -\partial_x \psi$$

Par comparaison avec le système (1), on peut choisir :

$$\psi(x, y, t) := -\Phi(x, y, p_0, t)/f$$

où p_0 est un niveau de pression arbitraire. La fonction de courant est donc proportionnelle au géopotentiel.

On suppose que :

$$2\pi/k = 3\,000\,\text{km}$$
$$2\pi/l = 4\,000\,\text{km}$$

Avec matplotlib, tracez $(\Phi - \Phi_0)/f$ sur un domaine horizontal de taille $(2\pi/k, 2\pi/l)$, à une date quelconque. Choisissez juste le noir comme couleur de contour, matplotlib distinguera alors les valeurs négatives de $(\Phi - \Phi_0)/f$ par des lignes pointillées. Ceci vous permet de connaître le sens du vent sur chaque contour.

La trajectoire d'une parcelle d'air $(X(t), Y(t))$ est donnée par la résolution du système de deux équations différentielles d'ordre 1 :

$$\frac{\mathrm{d}X}{\mathrm{d}t}(t) = u(X(t), Y(t), t)$$
$$\frac{\mathrm{d}Y}{\mathrm{d}t}(t) = v(X(t), Y(t), t)$$

Écrivez un programme en Fortran qui calcule la trajectoire d'une parcelle d'air à partir d'une position quelconque à $t = 0$, pendant une durée quelconque. Le programme écrira dans un fichier la position de la parcelle à un certain nombre n de dates, à intervalle de temps constant, au long de la trajectoire.

Le programme pourra fixer les valeurs de k et l mais il demandera à l'utilisateur de choisir dans une ou plusieurs namelists les valeurs de :

— la durée de l'intégration (par défaut 10 jours) ;
— n (nombre de dates auxquelles la position doit être connue, en incluant la date initiale et la date finale) (par défaut 41, ce qui fait un intervalle de 6 h entre deux dates pour une durée d'intégration de 10 jours) ;
— U' (par défaut $10\,\mathrm{m\ s^{-1}}$) ;
— c (défaut $5\,\mathrm{m\ s^{-1}}$) ;
— la position de la parcelle à $t = 0$ (défaut $(0, 0)$) ;

Le programme écrira dans un fichier les namelists utilisées.

Pour intégrer le système d'équations différentielles, utilisez la procédure odeint de la bibliothèque Numer_Rec_95, avec rkqs comme argument effectif correspondant à l'argument muet stepper. Les procédures odeint et rkqs sont accessibles via le module `numer_rec_95` :

```
use numer_rec_95, only: odeint, rkqs
```

Lancez une exécution avec les valeurs d'entrée par défaut.

Écrivez un script en Python qui lit et trace la trajectoire complète (produite par le programme en Fortran). Superposez sur un même graphique la trajectoire et les lignes de courant à la date initiale de la trajectoire. Lisez la namelist utilisée par le programme en Fortran avec le module f90nml. Cf. figure 4.

Créez une animation où l'on voit à chaque instant les lignes de courant et une portion de trajectoire jusqu'à cet instant. Vous pouvez par exemple choisir les dix dernières dates jusqu'à l'instant courant. Pour créer l'animation, vous devez écrire une fonction qui fait le tracé à un instant donné, puis donner cette fonction en argument de la classe FuncAnimation de Matplotlib. Prenez exemple sur le listing (1). Il est aussi possible de créer une animation Gif, en modifiant l'instruction save :

```
my_anim.save("trajectory.gif", writer = "imagemagick")
```

Utilisez alors le programme gifview (qui fait partie de gifsicle) pour visualiser l'animation.

Pour les plus avancés. Superposez sur un même graphique trois trajectoires (utilisez trois couleurs) partant de $y = -750$ km, $y = 0$, $y = 750$ km, et les lignes de courant à la date finale des trajectoires. Créez une animation sur laquelle on voit à chaque instant les lignes de courant et une portion des trois trajectoires jusqu'à cet instant. Refaites le graphique et l'animation en augmentant la vitesse de phase c à 10 et 15 $\mathrm{m\ s^{-1}}$.

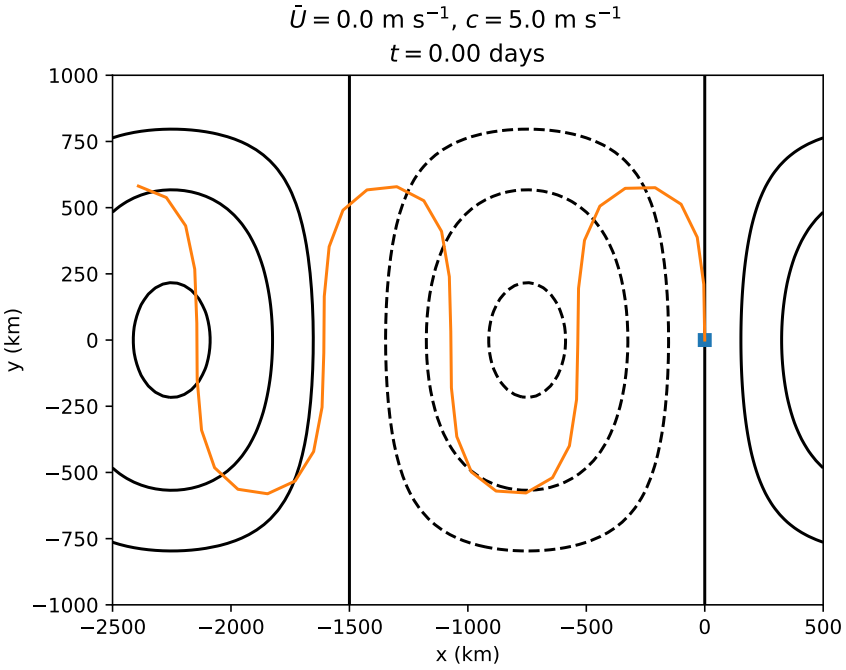

FIGURE 4 – Exercice 10. En orange, trajectoire pendant 10 jours d'une parcelle d'air partant de $(x, y) = (0, 0)$ à $t = 0$ (position marquée d'un carré bleu). En noir, lignes de niveau de $\Phi - \Phi_0(p_0)$ à $t = 0$, en trait continu pour un niveau positif, en pointillés pour un niveau négatif. $U' = 10\,\mathrm{m\ s^{-1}}$.

Listing 1 – Squelette pour créer une animation. On suppose ici que la variable t contient les dates auxquelles les positions de la parcelle sont disponibles.

```
from matplotlib import animation
...
my_fig = plt.figure()

def func(i, ...):
    plt.cla()
    ... # plotting commands for snapshot at date i

my_anim = animation.FuncAnimation(my_fig, func, np.size(t),
                                  fargs = (...))
my_anim.save("trajectory.mp4")
```