BONNET/TERRAS Fanny
GM4
INSA de Rouen

juin-octobre 1999

ETUDE DE LA STABILITE NUMERIQUE DU CODE ORCA

pour le compte du

Pôle de modélisation
de l'Institut Pierre Simon Laplace

Table des matières

Remerciements

Je tiens à remercier tout particulièrement les personnes qui ont encadré ce stage, Marie-Alice Foujols, Fabienne Jézéquel et Maurice Imbard, pour leur aide, leur patience et leur sympathie.

Je tiens également à remercier l'équipe "Calcul à Haute Performance et Validation" du LIP6, les personnes de l'Institut Pierrre Simon Laplace (IPSL) et du Laboratoire d'Océanographie DYnamique et de Climatologie (LODYC) pour leur aide et leur accueil chaleureux.

Je remercie aussi Clémentine Said, KimThai, Anne Fladenmüller et Arnaud Knippel pour leur accueil chaleureux.

Introduction

Les océans, qui couvrent 70% du globe et contiennent 97% de l'eau présente sur Terre, jouent un rôle primordial dans le système climatique. Le Laboratoire d'Océanographie DYnamique et de Climatologie (LODYC) a créé d'importants codes, à la pointe de la recherche, permettant de simuler les océans. La propagation des erreurs d'arrondi due à l'arithmétique à virgule flottante des ordinateurs peut fausser les résultats et engendrer des instabilités numériques. Pour pouvoir valider les codes de simulation des océans, il est important de connaître leur qualité numérique.

Une première étude [8] en 1998 a montré que l'on pouvait implanter la bibliothèque CADNA, qui permet de valider numériquement tout programme de calcul scientifique, dans une version simplifiée du code de simulation océanique OPA 8.0.

Suite à cette étude, ce stage a pour but de valider le code ORCA, version océan mondial du code OPA 8.1, grâce à cette bibliothèque. Après avoir pris connaissance des environnements informatiques, il a fallu implanter la bibliothèque CADNA dans le code ORCA. Dans un deuxième temps, la qualité numérique des variables a été étudiée. Enfin, nous avons analysé les pertes brutales de précision ainsi que les éventuelles instabilités numériques.

Chapitre 1

Présentation de l'environnement de travail

1.1 Présentation de l'IPSL

L'I.P.S.L. (L'Institut Pierre Simon Laplace) a pour objectif de mettre en commun des moyens spatiaux, informatiques, développements expérimentaux, bibliothèque, accueil, observatoires, unités d'enseignement et gestion, au service des *sciences de l'environnement global*, de façon à rationaliser leur utilisation et à minimiser leur coût.

Il regroupe pour cela six laboratoires de recherche localisés en Région Parisienne :

- Le Centre d'études des Environnements Terrestre et Planétaires (CETP)
- Le Laboratoire de Météorologie Dynamique (LMD)
- Le Laboratoire de Physique et Chimie Marines (LPCM)
- **Le Laboratoire d'Océanographie DYnamique et de Climatologie (LODYC)**
- Le Laboratoire des Sciences du Climat et de l'Environnement (LSCE)
- Le Service d'Aéronomie (SA)

LODYC Le LODYC est une Unité Mixte de Recherche (UMR 7617) dépendant du CNRS, département des Sciences de l'Univers, de l'Université Pierre et Marie Curie (Paris VI), et de l'Institut de Recherche pour le Développement (IRD). Il regroupe quelques 80 chercheurs, enseignants-chercheurs, ingénieurs, techniciens, administratifs et doctorants localisés à Jussieu.

La vocation première du LODYC est l'étude des processus dynamiques gouvernant la circulation océanique, la compréhension des mécanismes gouvernant l'évolution du système climatique terrestre où l'océan joue un rôle important et l'étude des cycles biogéochimiques océaniques, en particulier celui du carbone, qui mettent en jeu, entre autres, la biosphère marine.
Les principaux projets du laboratoire relèvent :

- d'une activité expérimentale fondée sur l'implication forte du LODYC dans les campagnes à la mer ;

- d'une activité de développement instrumental ;
- de l'interprétation conjointe des observations spatiales et in situ ;
- d'une activité de modélisation théorique, conceptuelle et statistique avancée, liée à l'interprétation des données ;
- **d'une activité de modélisation numérique, le laboratoire ayant développé un modèle de circulation générale océanique : OPA.**

Le laboratoire participe activement à l'enseignement supérieur et l'accueil de doctorants dans les domaines de l'océanographie, de la climatologie et de l'étude des cycles biogéochimiques océaniques.

1.2 ORCA

L'océanographie dynamique est une science récente. La structure détaillée des courants, les déplacements de masse d'eau, et la répartition des propriétés physiques et chimiques dans la mer sont loin d'être bien compris. Les processus physiques qui conduisent les courants et déterminent les propriétés physiques de l'eau de mer sont nombreux, complexes et se produisent au travers d'un vaste spectre spatio temporel. Ils résultent de la circulation induite par l'action du vent sur la surface de la mer et de la circulation liée à l'hétérogénité de la température et de la salinité.

Comprendre et simuler l'océan possède des similitudes avec le problème de prévision du temps. C'est sans doute pour cela que, historiquement, beaucoup de techniques utilisées dans le domaine de la météorologie ont été appliquées au problème océanographique.

OPA Le code OPA (Océan PArallélisé) est un **modèle de simulation de la circulation océanique à grande échelle**[1]. Il résout les équations de la mécanique des fluides i.e. les équations de Navier-Stockes, simplifiées en équations primitives.
Le modèle OPA est défini avec les conditions suivantes :

- Les hypothèses de Boussinesq, d'incompressibilité et d'hydrostatisme[2]
- La sphéricité de la Terre
- La hauteur de l'océan par rapport au rayon de la Terre est négligé
- L'utilisation d'une équation suppplémentaire (l'équation de l'énergie cinétique turbulente TKE) pour calculer le coefficient de diffusion sur la verticale
- L'utilisation d'une équation d'état (expression de la densité) qui est fonction de la température, de la salinité et de la pression et qui est approximée par des polynômes selon la formulation officielle de l'UNESCO (United Nations Educational, Scientific and Cultural Organization)

La méthode numérique utilisée est celle des différences finies sur une grille "C" dans la classification d'Arakawa. Un algorithme de gradient conjugué préconditionné est utilisé pour résoudre une équation elliptique.

1. A grande échelle signifie que la longueur des mailles de résolution est supérieure à une dizaine de kilomètres

2. Un fluide est hydrostatique si on néglige ses mouvements verticaux

Le modèle OPA peut réaliser des simulations d'océans académiques (bassins carrés) ou des configurations plus complexes (océans tropicaux, océan global (version ORCA)). Pour cela il peut avoir une bathymétrie[3] complexe, prendre en compte des îles et avoir des conditions initiales ou des contraintes réalistes.

Ce code permet de calculer l'évolution dans le temps de paramètres tels que **la vitesse du courant, la température, la pression et la salinité**.

ORCA La configuration ORCA est la configuration océan mondial du modèle OPA 8.1. Tous les océans sont représentés[4]. Dans cette configuration globale, le pôle Nord est un point singulier dans la grille latitude/longitude et pose des problèmes de stabilité numérique (mailles tendant vers zéro aux pôles). Pour cette raison, la grille du modèle ORCA est modifiée dans l'hémisphère Nord. Une première solution envisagea de déplacer le pôle Nord vers un pôle artificiel sur l'Himalaya, ainsi le point singulier ne se situait pas dans un océan. Cependant certaines mailles étaient trop déformées c'est pourquoi deux pôles artificiels ont finalement été adoptés, l'un en Russie et l'autre dans le Canada qui forment les deux foyers d'un ensemble d'ellipses (Voir le maillage d'ORCA en ANNEXE 2)

ORCA est définie avec les conditions initiales suivantes :

- une bathymétrie réaliste, déduite des données de Smith et Sandwell
- Prise en compte de 14 îles[5]
- Les observations de Levitus [6]pour la température et la salinité
- Les conditions de surface - flux de chaleur, flux d'eau, pression exercée par le vent - données sous forme de moyennes journalières établies sur 10 ans par le Centre Européen de Prévision Météorologique à Moyen Terme (CEPMMT).

Le code ORCA a été écrit en Fortran 77 (sauf pour le répertoire ioipsl que l'on n'utilisera pas), compatible Fortran 90. Ce programme comprend **20000 à 30000 lignes de codes** réparties en **9 fichiers ayant l'extension .f** - fichiers source Fortran -, **103 fichiers .F** - fichiers avant l'appel au précompilateur cpp - et **77 fichiers .h** - fichiers qui sont inclus dans les fichiers source par le précompilateur cpp. Le précompilateur cpp (standard sous UNIX) est utilisé non seulement pour inclure les fichiers d'extension .h dans des fichiers d'extension.F, mais aussi pour compiler ou non certaines parties des codes sources. En effet, l'utilisateur peut choisir différentes options qui vont déterminer le contenu de certains fichiers d'extension.f.

Calculateur CRAY J90 Toutes les simulations seront effectuées sur le calculateur CRAYJ90 du CCR (Centre de Calcul Recherche et Réseau de Jussieu). Voici quelques-unes de ses caractéristiques :

- Période d'horloge : 10 nanosecondes

3. fonds sous-marins

4. l'océan Atlantique, Pacifique, Indien, Arctique et Antartique

5. (1)L'Amérique, associée au Grœland, (2)L'Antartic, (3)La Nouvelle-Zélande, (4)L'Australie, (5)Madagascar, (6)La Nouvelle-Guinée, (7)Les Célèbes, (8)Bornéo, (9)La Tasmanie, (10)Les Philippines (représentée comme une seule île), (11)Cuba, (12)L'Islande, (13)Spitsberg, (14)Le Japon

- Architecture : Vectorielle 64 bits
- Processeurs : 8
- Capacité mémoire : 1 gigaoctet
- Capacité disque : 22 gigaoctets
- Unités de calcul par processeur : 2
- Mégaflops par processeur (pointe) : 200

Les variables utilisées dans ORCA sont codées en simple précision. Sur CRAYJ90, les réels en simple précision sont codés sur 64 bits (dont 48 bits pour la mantisse) et ont donc 13 chiffres significatifs. Il est à noter que l'arithmétique du CRAYJ90 ne respecte pas la norme IEE 754 de 1985 [9] selon laquelle les réels en double précision sont codés sur 64 bits (dont 52 bits de mantisse) et ont donc 15 chiffres significatifs [6].

1.3 Présentation du LIP6

Le Laboratoire d'Informatique de Paris 6 (LIP6) est une unité mixte de recherche en informatique de l'Université Pierre et Marie Curie et du CNRS (UMR 7606). Avec un effectif d'environ 320 personnes (doctorants compris), il est structuré en neuf thèmes scientifiques :

1. **Algorithmique numérique et parallélisme (ANP)**
2. Apprentissage et acquisition de connaissances (APA)
3. Architecture des systèmes intégrés et micro-électronique (ASIM)
4. Calcul formel (CALFOR)
5. Objets et Agents pour Systèmes d'Information et de Simulation (OASIS)
6. Réseaux et performances (RP)
7. Sémantique, preuve et implantation (SPI)
8. Systèmes répartis et coopératifs(SRC)
9. Systèmes d'aide à la décision et à la formation (SYSDEF)

Le thème Algorithmique Numérique et Parallélisme comprend deux équipes :

- **Calcul à Haute Performance et Validation (CHPV)**
- Optimisation et Modélisation

L'équipe Calcul à Haute Performance et Validation (CHPV) travaille sur deux projets :

1. Projet PAN (Parallélisation d'Algorithmes Numériques)
2. Projet VAN (Validation des Algorithmes Numériques).
   Ce dernier projet consiste à contrôler et valider les logiciels scientifiques, c'est-à-dire à faire en cours d'exécution du programme :
   - L'analyse de la propagation des erreurs d'arrondi

6. Les réels en simple précision sont codés sur 32 bits dont 23 bits de mantisse, ce qui correspond à 7 chiffres significatifs

- La détection des instabilités numériques
- Le contrôle des tests et des branchements
- L'estimation de la précision de tout résultat de calcul
- L'estimation de l'influence des incertitudes des données sur les résultats fournis

Ce problème est abordé grâce à l'approche stochastique. Celle-ci permet de tenir compte de la compensation des erreurs d'arrondi et de donner une bonne estimation des résultats.

Ce projet vise comme application tout type de calcul scientifique en virgule flottante sur machine séquentielle ou parallèle.

1.4 La méthode CESTAC et la bibliothèque CADNA

Le logiciel CADNA (Control of Accuracy and Debugging for Numerical Applications) conçu par *Jean VIGNES* et *Jean-Marie CHESNEAUX* [3, 8], tous deux membres de l'équipe Calcul à Haute Performance et Validation (CHPV), permet de valider les résultats de tout programme de calcul scientifique exécuté sur ordinateur. CADNA se présente comme une librairie utilisable après la compilation lors de l'édition de liens, qui permet de mettre en œuvre automatiquement et de manière synchrone la méthode CESTAC (Contrôle et Estimation STochastique des Arrondis de Calculs) conçue par *Michel LA PORTE* et *Jean VIGNES*.

La méthode CESTAC Fondée sur une approche probabiliste, la méthode CESTAC permet d'**estimer le nombre de chiffres significatifs exacts de tout résultat fourni par un programme informatique**[11, 12, 13].

La méthode CESTAC consiste à exécuter plusieurs fois le même programme de calcul en propageant différemment les erreurs d'arrondi. On obtient ainsi pour un même calcul, des résultats différents. La partie commune à tous les résultats représente la partie fiable, l'autre étant la partie non significative.

L'implémentation synchrone de la méthode CESTAC consiste à effectuer N fois (N=2 ou 3) chaque opération arithmétique, en utilisant l'arithmétique aléatoire[7], avant d'exécuter la suivante. Tout se passe comme si N programmes de calculs s'exécutaient en simultanéité sur N ordinateurs synchronisés." Soit R le résultat d'une opération arithmétique quelconque. On dispose alors de N résultats intermédiaires R_i. On choisit la moyenne

$$\overline{R} = \frac{1}{N} \sum_{i=1}^{N} R_i$$

comme représentant informatique de R et on calcule le nombre de chiffres significatifs exacts de $\overline{R}$ grâce à la formule

$$C_{\overline{R}} = log_{10} \left(\frac{\sqrt{N}.|R|}{s.\tau_\beta} \right)$$

7. Tout résultat R d'une opération arithmétique qui n'est pas un flottant est encadré par deux flottants successifs R^- et R^+. L'arithmétique aléatoire consiste à retenir aléatoirement R^- et R^+ avec la même probabilité.

avec s l'écart type de R,

$$s^2 = \frac{1}{N-1} \sum_{i=1}^{N} \sqrt{\left(R_i - \overline{R}\right)}$$

et $\forall \beta \in [0,1], \exists \tau_\beta \in \mathbf{R}$ tel que

$$P\left(|\overline{R} - r| \leq \frac{s.\tau_\beta}{\sqrt{N}}\right) = \beta$$

avec r le résultat mathématique de R.
Dans la version de la bibliothèque CADNA qui est utilisée N = 2, $\beta = 0,95.$ et $\tau_\beta = 12,706.$

Le zéro informatique Le concept de zéro informatique [14] est essentiel dans la méthode CESTAC.
Définition : Un résultat informatique R est un zéro informatique si R=0 en étant significatif ou bien si R est quelconque mais non significatif.

Concrètement, avec la méthode CESTAC, un résultat R, représenté par N résultats R_i, sera un zéro informatique si l'une des deux conditions suivantes est remplie:

1. $\forall i, R_i = 0$
2. $C_{\overline{R}} \leq 0$

En pratique, l'ordinateur ne peut distinguer le zéro informatique du zéro mathématique. Avec ce nouveau concept, une arithmétique stochastique a été dévelopée avec de nouvelles définitions pour les relations d'ordre et d'égalité, qui prennent en compte la qualité numérique des opérandes.

La bibliothèque CADNA La bibliothèque CADNA implémente automatiquement la méthode CESTAC dans tout code écrit en Fortran, C ou ADA. Avec CADNA, on peut utiliser de nouveaux types numériques : les types stochastiques. Cette bibliothèque contient les définitions de toutes les opérations arithmétiques et les relations d'ordre pour les variables stochastiques. Elle permet d'estimer le nombre de chiffres significatifs exacts de n'importe quelle variable stochastique. En outre, CADNA détecte les éventuelles instabilités numériques causées par un zéro informatique. Par exemple, CADNA détecte les divisions instables car elles sont dues à un dénominateur non significatif. Un zéro informatique est noté par CADNA **@.0.**

Chapitre 2

Instrumentation d'ORCA avec la bibliothèque CADNA

La première partie de ce stage consiste à implémenter CADNA dans le code ORCA de façon à valider ou non par la suite ses résultats.

CADNA a déjà été implémenté par le passé dans le modèle OPA, version 8, configuration bassin accadémique [7]. Cette étude s'est donc réalisée dans un contexte simplifié, avec des données initiales de température et de salinité identiques sur tout le bassin et avec des flux de chaleurs et de pressions dues au vent, constants et réguliers. Cette première étude a montré que l'on pouvait instrumenter CADNA dans OPA.

Dans notre cas, on prend désormais en considération un océan mondial plus complexe, avec une température et une salinité représentative de la variabilité observée dans la nature et avec des contraintes de chaleur et de vent réalistes. De plus, les courants ("la dynamique du modèle") sont initialisés à partir d'un fichier de reprise.

La mise en œuvre de la bibliothèque CADNA comporte 6 étapes principales :

2.1 L'appel à la bibliothèque CADNA

Les types stochastiques que la bibliothèque CADNA utilise sont définis dans un module (au sens du Fortran 90) qui doit donc être "visible" des programmes, modules et sous-programmes. Pour cela, il convient de faire précéder les déclarations de la pseudo-instruction USE CADNA.

Dans ORCA tous les fichiers.F contiennent l'instruction

```
#include"parameter.h"
```

et donc contiendront le fichier parameter.h, c'est pourquoi l'instruction USE CADNA a été seulement rajoutée dans ce fichier et non dans tous les .F. D'autre part, elle a été rajoutée dans les 9 fichiers .f en dessous de chaque instruction SUBROUTINE ou FUNCTION et avant la pseudo instruction IMPLICIT NONE.

2.2 L'initialisation

Cette initialisation a pour but :

- d'ouvrir le fichier cadna_stability_f90.lst (correspondant à l'unité logique 58) qui contient les messages éventuels d'instabilités numériques
- d'initialiser l'utilisation de l'arithmétique aléatoire.

Cette initialisation est à ajouter à la suite des déclarations du programme principal (*modele.f*). Elle est réalisée par l'appel d'un sous-programme :
call CADNA_INIT(K,ind_div,ind_mul,ind_test,ind_intr,ind_math,ind_perte,SEUIL)

Le premier argument, K, de type entier, permet de limiter la taille du fichier cadna_stability_f90.lst.

- Si K = -1, le fichier contiendra tous les messages des instabilités détectées.
- Si K = 0, le fichier ne contiendra aucun message d'instabilité.
- Si K = N, le fichier contiendra les N premiers messages.

Les autres arguments sont de types logiques et valent vrai si l'on veut detecter les instabilités correspondantes :

ind_div : pour des divisions par un "zéro informatique"

ind_mul : pour des multiplications où les deux opérandes sont des "zéro informatiques"

ind_test: pour la détection dans les relations d'ordre où la différence entre les deux opérandes est un "zéro informatique"

ind_intr : pour le cas où les fonctions intrinsèques *int, aint, abs, mod, sign, dim* génèrent des instabilités

ind_math : pour le cas où les fonctions mathématiques *sqrt, exp, log, log10, sin, cos, tan, asin, acos, atan, atan2, sinh, cosh, tanh* génèrent des instabilités.

ind_perte : pour détecter la perte brutale de précision d'une opération arithmétique élémentaire

Le dernier argument, S, de type entier, représente le nombre de chiffres significatifs exacts à partir duquel on considère qu'il y a perte brutale de précision. Sa valeur par défaut est 4.

Dans un premier temps K a été initialisé à 0 pour ne pas allonger le temps d'exécution.

Dans le fichier *modele.f*, en plus de l'instruction

```
use cadna
```

on a donc ajouté l'instruction suivante:

```
 call cadna_init(0,ind_div=.FALSE., ind_test=.FALSE.,
$ ind_mul=.FALSE.,ind_intr=.FALSE., ind_math=.FALSE.,
$ ind_perte=.FALSE.)
```

2.3 Les modifications de type

La bibliothèques CADNA définit quatre nouveaux types numériques, les types stochastiques :

TYPE (SINGLE_ST) déclare un stochastique simple précision

TYPE (DOUBLE_ST) déclare un stochastique double précision

TYPE (COMPLEX_ST) déclare un stochastique complexe simple précision

TYPE (DOUBLE_COMPLEX_ST) déclare un stochastique complexe double précision

Tous les opérateurs arithmétiques et logiques, les fonctions intrinsèques et mathématiques du fortran 77 ont été redéfinis pour être compatibles avec ces nouveaux types.

Tous les réels présents dans ORCA étaient des réels "simple précision" (REAL ou REAL*8) et ont donc été remplacés par des stochastiques de type TYPE (SINGLE_ST). Pour cela nous avons commencé par remplacer toutes les variables réelles locales, étant donné qu'elles commencent toutes par z[1], elles étaient facilement repérables. Nous avons trouvé de telles variables dans 66 fichiers .F. Dans Fioopa.F, certaines variables sont restées réelles car elles ne seront pas utilisées[2], nous avons donc dû laisser également les tableaux *tabdta* et *tabglo* comme réels. Les types réels ont été remplacés dans 41 fichiers .h et 5 fichiers .f. Cependant nous avons pas modifié les variables réelles[3] correspondant à des "masques" car elles valent toujours 0. ou 1.[4]. La subroutine *lbc* utilise parfois comme argument des "masques". Nous avons donc du redéfinir une subroutine *lbc2* avec le premier argument de type *REAL* au lieu de *TYPE(SINGLE_ST)* (*lbc*). Ces deux subroutines se trouvent dans le fichiers *lbc.F*.

Toutes les variables réelles définies par l'instruction PARAMETER ont gardé leur type réel car ces variables ne varient pas au cours du programme et donc leur nombre de chiffres significatifs exacts ne peut évoluer.

La modification des types entraîne une modification pour les variables initialisées au moyen de l'instruction DATA. En effet une variable stochastique est représentée par deux réels donc il faut doubler l'initialisation. La variable *zrnfm* est initialisée avec un DATA mais, dans un premier temps, nous avons choisi de la laisser en réel.

2.4 Modification des fonctions on-line

CADNA n'accepte pas les fonctions on-line car l'opération d'égalité qui correspond à la définition d'une fonction avec des arguments n'a pas été surdéfinie. Par exemple dans le fichier *domhgr.F* la définition on-line de la fonction fslam :

```
#include "parameter.h"
#include "common.h"
      REAL fslam
      REAL pi, pj
      RAL  zstlam, zdx
```

1. Par convention, le code ORCA est écrit suivant la règle d'ECMWF appelée DOCTOR [Gibson,1986], dans laquelle toute variable réelle locale doit commencer par un z.

2. Elles sont utilisées lorsque le calcul est effectué avec d'autres machines que le CRAYJ90

3. tmask, fmask, bmask, umask, vmask, tpol, upol, vpl, fpol, freeze, freezn, tmsea

4. Un océan est défini comme un grand parallélépipède maillé. Pour les mailles où il y a de la terre, le masque correspondant vaut 0., celles où il y a de l'eau, il vaut 1.

```
      dx    = 1.
      zstlam = 0.
      fslam(pi,pj) = zstlam + zdx * ( pi-1. + float(nimpp-1) )
```

où *nimpp* est un entier défini dans common.h, a été remplacée par

```
      FUNCTION fslam( zaux1,zaux2 )
      #include "parameter.h"
      #include "common.h"
      REAL zaux1, zaux2
      TYPE(SINGLE_ST) pi, pj, fslam
      TYPE(SINGLE_ST) zdx, zstlam
      pi=zaux1
      pj=zaux2
      zdx    = 1.
      zstlam = 0.
      fslam = zstlam + zdx * ( pi-1. + float(nimpp-1) )
      RETURN
      END
```

Ainsi toutes les fonctions on-line présentes dans *domhgr.F -fslam, fsdila, fsdjla, fsdiph, fsdjph-* ont été redéfinies au début de ce fichier.

D'autres part toutes les fonctions on-line définies dans le fichier *stafun.h* ont été redéfinies dans un **nouveau fichier : *Functionexp.F***. Il s'agit des fonctions *fsx, fsy, fsz, fsfzpt, fsass, fsdept, fsdepu, fsdepv, fsdepf, fsdepw, fsdepuw, fsdepvw, fse3t, fse3u, fse3v, fse3f, fse3w, fse3uw, fse3vw, fsahmu, fsahmv, fsahmt, fsahmf, fsahmuw, fsahmvw, fsahmt, fsahmf,f sahtt, fsahtu, fsahtv,f sahttw, fsaeiu, fsaeiv* et *fsaeiw*.

D'autre part un autre fichier a été créé : ***Functionexp2.F***[5] qui contient les fonctions

- *ffsig et fsdsig* qui étaient définies on-line dans *domzgr.s.h*
- *fsdep et fse3* qui étaient définies on-line dans *domzgr.z.h*
- *fsalbt* qui était définie on-line dan dans *trabbl.F*

Functionexp2.F contient aussi la fonction *cvmgp2* dont on parlera par la suite.

2.5 Modification de l'appel de certaines fonctions standard

Dans le fichier *domzgr.z.h* les appels aux fonctions génériques[15] ALOG et ALOG10 ont été remplacés par des appels aux fonctions LOG et LOG10.

De plus les fonctions MIN et MAX, dans un programme utilisant CADNA ne doivent comporter que deux arguments. Par conséquent, dans le fichier *dommsk.F* des appels aux fonctions min et max ont été modifiés. Par exemple l'instruction

```
fmask(ji,jj,jk) = shlat
   $      * min( 1., max( zwf(ji+1,jj), zwf(ji,jj+1),
```

5. Un deuxième fichier a été créé pour gagner du temps car rajouter une fonction au fichier *Functionexp.F* entraînait sa recompilation ce qui était assez long

```
     $                     zwf(ji-1,jj), zwf(ji,jj-1) ) )
```

a été remplacée par

```
 fmask(ji,jj,jk) = shlat
     $      * min( 1., max( zwf(ji+1,jj), max( zwf(ji,jj+1),
     $                max( zwf(ji-1,jj), zwf(ji,jj-1) ) ) ) )
```

Dans le fichier *alllib.F* il y a l'instruction INTRINSIC MAX or la fonction est redéfinie dans CADNA donc elle n'est plus intrinsèque. Cette instruction a donc été mise en commentaire.

D'autre part dans le fichier *Fetilib.f* deux variables appelées *nint* et *dim* sont utilisées or CADNA utilise deux fonctions de même nom. Donc *nint* a été renommée en *nint0* et *dim* en *dim0*.

2.6 Modification des fonctions du compilateur Fortran sur CRAY

Le code OPA utilise des fonctions, qui possèdent des arguments réels, propres au compilateur Fortran sur CRAY : *cvmgp, cvmgm, cvmgz, sdot*. Elles n'ont pas été redéfinies dans CADNA avec des arguments stochastiques. Nous les avons redéfinies sous un autre nom pour que les fonctions correspondantes du CRAY ne soient pas appelées en premier. Ainsi *cvmgp* a été redéfini dans *Functionexp2.F* par *cvmgp2*, *cvmgm* par *cvmgm2*, *cvmgz* par *cvmgz2* et *sdot* par *sdot2*.
Les appels aux fonctions *cvmgp2, cvmgm2, cvmgz2* utilisaient des passages par valeur, ce qu'il a fallu modifié car les valeurs passées étaient alors des réels et non des stochastiques.

2.7 Modification des opérateurs vectoriels

Comme la bibliothèque CADNA est en fait un module écrit en Fortran 90, son implémentation dans des codes écrits dans ce langage est possible. Cependant la surcharge des opérations sur les tableaux n'a pas encore été définie dans la version actuelle de CADNA[6]. Les opérations sur les tableaux doivent donc être effectuées à l'aide de boucles.

Par exemple, dans le fichier *allib.f* il y avait une affectation directe de tableau (bien que le programme soit écrit en Fortran 77, certaines instructions isolées comme la suivante appartiennent au Fortran 90) :

```
real*8 b(n,n)
b = 0.0
```

Cette instruction a été remplacée par :

```
type(single_st)b(n,n)
DO jj=1,n
   DO ji=1,n
```

6. L'implémentation automatique de CADNA dans un programme écrit en Fortran 90 est actuellement en cours d'étude.

```
          b(ji,jj) = 0.0
        END DO
      END DO
```

2.8 Modification des entrées

Dans un premier temps on ne tiendra pas compte de l'influence que peut avoir l'incertitude des données.

La fonction de lecture READ ne permet pas de lire directement des variables de type stochastique. Il faut rajouter une variable locale réelle, que l'on va lire, et on affecte ensuite la variable stochastique à cette valeur. Dans ORCA, nous avons généralement appelé cette variable réelle *zaux*. Par exemple dans le fichier *domzgr.s.h* on lisait le réel *zdata* :

```
     REAL zdata(jpidta,jpjdta)
     DO jj=jpjdta,1,-1
          READ(numhba,9209) ij,(zdata(ji,jj),ji=il1,il2)
        END DO
        il1=il1+ifreq
      END DO
 9206 FORMAT(3x,15(3x,i3,3x))
 9209 FORMAT(i3,15f8.1)
```

où *jpidta* et *jpjdta* sont deux entiers et *numhba* l'unité logique des fichiers de bathymétrie. Dans la version "cadnatisée" on écrit :

```
      TYPE(SINGLE_ST) zdata(jpidta,jpjdta)
      REAL zaux(jpidta,jpjdta)
      DO jj=jpjdta,1,-1
          DO ji=il1,il2
            READ(numhba,9209) ij,zaux(ji,jj)
            zdata(ji,jj)=zaux(ji,jj)
          END DO
        END DO
        il1=il1+ifreq
      END DO
 9206 FORMAT(3x,15(3x,i3,3x))
 9209 FORMAT(i3,15f8.1)
```

Le fichier *parlec.F* va lire des variables dans le fichier *namelist* du répertoire SCRIPTS (en dehors du répertoire ORCA où se trouvent tous les autres fichiers). Dans le fichier parlec.F pour toutes les variables stochastiques auxquelles sont affectées une valeur du fichier *namelist*, on a ajouté une variable réelle locale, qui elle est lue. On a donc changé le nom de la variable correspondante dans le fichier namelist. Par exemple on avait dans *parlec.F*

```
    REAL atfp, rdt, rdtmin, rdtmax, rdth,
    NAMELIST/namdom/ atfp, rdt, rdtmin, rdtmax, rdth
    READ  ( numnam, namdom )
```

et dans le fichier *namelist*

```
&NAMDOM
   atfp   =   0.1
   rdt    = 5760.,
   rdtmin = 5760.,
   rdtmax = 5760.,
   rdth   =  800.
 &END
```

Après "cadnatisation", on a dans *parlec.F*

```
TYPE(SINGLE_ST) atfp, rdt, rdtmin, rdtmax, rdth
REAL zatfp, zrdt, zrdtmin, zrdtmax, zrdth
NAMELIST/namdom/  zatfp, zrdt, zrdtmin, zrdtmax, zrdth
READ  ( numnam, namdom )
atfp=zatfp
rdt=zrdt
rdtmin=zrdtmin
rdtmax=zrdtmax
rdth=zrdth
```

et dans le fichier *namelist*

```
&NAMDOM
   zatfp   =   0.1,
   zrdt    = 5760.,
   zrdtmin = 5760.,
   zrdtmax = 5760.,
   zrdth   =  800.,
 &END
```

2.9 Modification des sorties

De façon à fournir les résultats avec leur précision associée, il convient de modifier les ordres d'écriture en utilisant la fonction **STR()** de CADNA. Cette fonction traite un argument de type stochastique et retourne une chaîne de caractère contenant la représentation décimale avec exposant de l'argument. Seuls les **chiffres décimaux significatifs exacts** sont contenus dans la chaîne. La précision est alors immédiatement lisible. Lorsque l'argument est un "zéro informatique", la chaîne retournée est : **@.0**.

Comme la fonction STR() n'accepte qu'un argument, si plusieurs variables stochastiques sont écrites dans une même instruction WRITE il faut autant d'appels à la fonction STR() que de sorties. Voici un exemple tiré du fichier *diadyn.F* :

```
WRITE (numtrd,9401) str(zumo( 1)/tvolu), str(zvmo( 1)/tvolv)
```

Le format nécessaire pour l'affichage d'un stochastique sur CRAYJ90 est *a21*, ce qui correspond à une chaîne de 21 caractères. Cependant pour que les affichages en colonnes soit plus lisibles, nous avons choisi *a23* comme format.

Chapitre 3

Qualité numérique des variables

Le nombre de tests et le nombre d'itérations prises en compte ont été fortement limités par le temps de calcul nécessaire sur le calculateur CRAY J90. L'importance du coût de calcul est principalement due à CADNA qui casse la vectorisation du code ORCA. En outre à partir de fin septembre, les temps de calcul ont fortement augmenté car les utilisateurs du CRAY J90 aussi !

Par exemple, 31 itérations mi-septembre ont demandé 1 jour et demi de calcul soit, avec le temps d'attente, 2 jours et 5 heures. Fin septembre, 30 itérations ont demandé 2 jours et 18 heures soit, avec le temps d'attente, 5 jours et 6 heures. Alors que, sans CADNA, 30 itérations ont demandé seulement 301 secondes !

Avec CADNA, un job de 30 itérations utilise 86 Mmots de 8 octets de mémoire. Sans CADNA, il en utilise 50.

Les variables du modèle, la température tn, la salinité sn, les composantes du vecteur horizontal du courant (un, vn), sont liées entre elles par des équations [5].

Une variation de la température ou de la salinité en un point modifie la densité de l'océan et influe sur le courant. Inversement, une modification du courant affecte la température et la salinité par l'intermédiaire, en autres, des termes advectifs.

De plus l'océan est contraint en surface par les conditions atmosphériques (flux solaire, précipitations, etc)

Pour appréhender la stabilité numérique du modèle, on a donc mesuré :

- l'évolution de la précision des variables au cours de l'intégration (donc des calculs) en supposant que la précision des données initiales est maximale au départ
- l'impact de la précision d'une ou plusieurs variable(s), un et vn et/ou tn, sur l'ensemble des variables, en supposant que celle-ci soit inférieure à la précision maximale de départ
- l'impact de la précision d'une donnée extérieure, qsr, sur l'ensemble des variables

Pour chaque variable, le nombre moyen de chiffres significatifs exacts est donné pour les trente premières itérations (à partir de la 2ème, sauf pour q à partir de la 3ème), celui-ci a été calculé sans incertitude sur les données.

La variation du nombre de chiffres significatifs exacts est donnée au vu des 30 itérations.

3.1 qsr : radiation solaire

(en $watt.m^{-2}$, variable 2D)
qsr représente le flux solaire qui intervient dans les conditions limites de surface pour l'évolution de la température. La variable *qsr* est lue dans un fichier de forcages atmosphériques et n'est pas modifiée au cours des itérations donc elle garde 14 chiffres significatifs exacts.

3.2 q : condition de flux de chaleur à la surface

(en $watt.m^{-2}$, variable 2D)
q représente le flux total qui intervient dans les conditions limites de surface pour l'évolution de la température. Il comprend le flux solaire *qsr*. La moyenne du nombre de chiffres significatifs exacts de q varie entre **11,02 et 11,8** (voir Fig 6.1 et 6.2, ANNEXE 3). Il est à noter que q est la seule variable dont la précision augmente au cours de la première itération. D'autre part, la précision moyenne de q paraît se stabiliser au cours des 14 dernières itérations autour de 11 chiffres significatifs exacts.

Le nombre de chiffres significatifs exacts varie entre 6 et 14 (voir Fig 6.3, ANNEXE 3).

La précision de q dépend de celle de *qsr* et de *tn* mais est indépendante de celle de *un* et *vn* (voir Fig 6.4, ANNEXE 3). Plus précisement, si l'on impose à *qsr* une précision de 10,76 chiffres significatifs exacts (au lieu de 14), q possède à la $1^{ère}$ itération 10,2 chiffres significatifs exacts et sa précision augmente légèrement lors des itérations suivantes.
Si l'on impose à *tn* une perte de 3,8 chiffres significatifs exacts, q perd le même nombre de chiffres significatifs exacts à la première itération (par rapport à la courbe de q, sans incertitude sur les données) et sa précision augmente légèrement au cours des itérations suivantes.

3.3 tn : température

(en degrés Celsius, variable 3D)
La température dépend, d'une part des équations, d'autre part des flux de chaleur à la surface donc de *qsr* et de q. La moyenne du nombre de chiffres significatifs exacts de *tn* varie entre **11,64 et 13,02** (voir Fig 6.5 et 6.6, ANNEXE 4). La précision moyenne de *tn* diminue régulièrement et semble se stabiliser au cours des 5 dernières itérations autour de 11,65 chiffres significatifs exacts.

Le nombre de chiffres significatifs exacts varie entre 5 et 14 (voir Fig 6.7, ANNEXE 4).

Bien que les conditions de surfaces de la température dépendent de *qsr*, une incertitude sur cette donnée extérieure n'a pas d'influence notable sur *tn*.
Une incertitude sur *un* et *vn* influence légèrement *tn* (voir Fig 6.8, ANNEXE 4).
Avec une incertitude sur la variable *tn* elle-même, en la mettant à 10,22 chiffres significatifs exacts, la précision diminue très légèrement au cours des 10 premières itérations. Il faudrait visualiser ce résultat sur plus d'itérations pour voir si la précision augmente.

3.4 sn : salinité

(en psu : gramme par litre, variable 3D)
La moyenne du nombre de chiffres significatifs exacts de *sn* varie entre **13,00 et 13,90** (voir Fig 6.9 et 6.10, ANNEXE 5). La précision moyenne de *sn* est irrégulière de la $9^{ème}$ à la $11^{ème}$ itération, plus légèrement pour les 15 dernières itérations. Pour 15 dernières itérations, la précision semble se stabiliser autour de 13,075 chiffres significatifs exacts. ***sn* possède donc une très bonne précision.**

Le nombre de chiffres significatifs exacts varie entre 9 et 14 (voir Fig 6.11, ANNEXE 5).

La précision de *sn* n'est pas du tout influencée par celle de *qsr*. Cela paraît à priori normal puisqu'elle est indirectement concernée par *qsr* à travers *tn*, qui elle est peu influencée par cette variable comme nous l'avons vu en 3.3.
Par contre *un*, *vn* et *tn*, 3 variables principales de l'océan influence la quatrième variable principale, *sn* (voir Fig 6.12, ANNEXE 5). C'est la seule variable étudiée où l'incertitude sur *un* et *vn* se cumule avec l'incertitude sur *tn*.
On remarque que plus on avance dans les itérations, moins les incertitudes sur les données initiales semblent avoir de l'influence.

3.5 rdn : densité

(en $kg.m^{-3}$, variable 3D)
rdn représente la densité. Elle dépend de l'équation d'état qui intervient dans l'équation du mouvement et qui, pour l'océan, est une fonction non linéaire de la température et de la salinité. Donc la densité *rdn* dépend de *tn* et de *sn*. La moyenne du nombre de chiffres significatifs exacts de *rdn* varie entre **11,93 et 11,99** (voir Fig 6.13 et 6.14, ANNEXE 6). **C'est la variable dont la précision moyenne est la plus stable parmi celles étudiées**. En regardant de très près, on s'aperçoit que la précision de la densité diminue d'environ 0,01 à 0,02 chiffres significatifs exacts toutes les dix itérations, ce qui est très faible, relativement aux autres variables.

Le nombre de chiffres significatifs exacts varie entre 6 et 14 (voir Fig 6.15, ANNEXE 6).

Une incertitude sur *qsr* et sur *un* et *vn* n'influence pas de façon significative la précision de *rdn*. En revanche elle est influencée par une incertitude sur la température via l'équation d'état : quand on enlève 3,8 chiffres significatifs exacts à la précision de *tn*, la précision de *rdn* perd en moyenne 0,6 chiffres significatifs exacts (voir Fig 6.16, ANNEXE 6). **Une incertitude sur *un*, *vn* et *tn* équivaut ici à une incertitude sur *tn*.**

3.6 bn2n : fréquence de Brunt-Vaisala

(en s^{-2}, variable 3D)
Cette variable indique la flotabilité d'une particule. Elle est calculée par une fonction non linéaire dépendant de la température et de la salinité assez semblable à celle de la densité. La moyenne du nombre de chiffres significatifs exacts de *bn2n* varie entre **10,83 et 11,58** (voir Fig 6.17 et 6.18, ANNEXE 7). Après la $1^{ère}$ itération, la précision moyenne de *bn2n* diminue régulièrement et semble se stabiliser au cours des 5 dernières itérations autour de 10.85.

Le nombre de chiffres significatifs exacts varie entre 1 et 14 (voir Fig 6.19, ANNEXE 7).

La précision de *bn2n* n'est pas influencée par celle de *qsr*, mais par celle de *un* et *vn* et de façon plus importante par celle de *tn* (voir Fig 6.20, ANNEXE 6). Avec une incertitude sur ces trois dernières variables, on ne tient compte quasiment que de celle de *tn*. Ici aussi, on s'aperçoit que **plus on avance dans les itérations, moins les incertitudes sur les données ont de l'influence.**

3.7 en : énergie cinétique turbulente

(en Joules, variable 3D)
en représente l'énergie cinétique turbulente. Elle intervient dans une équation du modèle qui sert à caractériser le type d'écoulement (s'il est laminaire (stratifié) ou s'il est brassé (turbulent)). Elle dépend entre autres de *bn2n*. La moyenne du nombre de chiffres significatifs exacts de *en* varie entre **13,42 et 13,55** (voir Fig 6.21 et 6.22, ANNEXE 8). La précision moyenne de *en* est stable autour de 13,5 chiffres significatifs exacts. En regardant de plus près, on s'aperçoit que la précision diminue jusqu'à la $18^{ème}$ itération et semble réaugmenter pendant les 12 itérations suivantes. Il serait intéressant de regarder ce résutat sur plus d'itérations.

Le nombre de chiffres significatifs exacts varie entre 5 et 14 (voir Fig 6.23, ANNEXE 8). Si, au cours de l'exécution, *en* n'appartient plus à un intervalle de valeurs, on lui affecte la borne correspondante de l'intervalle, *en* possède alors 14 chiffres significatifs exacts. **Les résultats sur *en* sont donc fortement biaisés**. Ce qui explique le pic à 14 chiffres significatifs exacts (voir Fig 6.23, ANNEXE 8) et la forte stabilité de la précision de la variable.

La précision de *en* n'est pas influencée par celle de *qsr*, légèrement par celle de *un* et *vn* et de façon un peu plus importante par *tn*. **On voit clairement que**

l'imprécision donnée à *tn* inhibe celle des autres et que son influence diminue au fur et à mesure que l'on avance dans les itérations.

3.8 un : vitesse horizontale du courant, composante zonale

(en $m.s^{-1}$, variable 3D)

un représente la composante horizontale de la vitesse du courant suivant l'axe des abscisses (l'axe des longitudes). Cette variable est régie par l'équation du mouvement et dépend donc de *tn* et *sn*, entre autres, via *rdn*. La moyenne du nombre de chiffres significatifs exacts de *un* varie entre **9,33 et 10,2** (voir Fig 6.25 et 6.26, ANNEXE 9). La précision de *un* diminue jusqu'à la $16^{ème}$ itération et semble augmenter par la suite.

Nous verrons par la suite que, d'une façon générale, les variables dynamiques du modèle : *un, vn, wn*, ont une précision inférieure aux variables dites traceurs : *tn, sn, rdn, ...* En effet les calculs de l'équation du mouvement sont plus nombreux et plus complexes et font intervenir un algorithme numérique[1] pour calculer une partie des termes sous forme d'équation elliptique.

Le nombre de chiffres significatifs exacts de *un* varie entre 3 et 14 (voir Fig 6.27 ANNEXE 9).

Nous avons étudié de plus près la répartition géographique de la précision de *un* à la $11^{ème}$ itération. Tout d'abord la répartition sur l'ensemble du globe, à la surface (voir ANNEXE 16), montre qu'à l'équateur, là où les courants sont les plus importants, la précision de *un* est globalement meilleure. Le long de la latitude 50 Sud, la précision de *un* est homogène avec 10 chiffres significatifs exacts.

Nous avons établi deux cartes comparables, respectivement à 95m (voir ANNEXE 17) et à 512m (voir ANNEXE 18). Elles montrent que plus on descend dans les profondeurs, plus *un* est imprécis.

Une coupe du premier schéma suivant l'équateur montre que même à des profondeurs plus importantes, la précision reste bonne à cet endroit. Une coupe Nord-Sud suivant la longitude 160 Ouest, au travers de l'océan Pacifique, confirme la remarque précédente : plus on descend dans les profondeurs de l'océan, moins *un* est précis. On peut alors supposer que le courant doit être plus faible en profondeur, ce qui expliquerait le nombre de chiffres significatifs exacts plus faible (voir le phénomène de cancellation au chapitre suivant).

On a donc utilisé un programme de visualisation [17] qui permet, à partir de données en 2D de visualiser les valeurs d'une variable et sa précision en même temps (voir ANNEXE 19). Globalement, on remarque que **les valeurs de *un*, en valeur absolue, sont proportionnelles à leur précision**. Ce qui confirme ce que nous avions supposé précédemment.

1. L'algorithme du gradient conjugué préconditionné

La précision de *un* est influencée par celle de *tn* mais est indépendante de celle de *qsr* lors des premières itérations (voir Fig 6.28, ANNEXE 9). Cela peut être expliqué par le fait que la densité *rdn*, qui dépend entre autres de la température *tn*, intervient dans l'équation du mouvement qui régit l'évolution du courant (de composantes *un, vn* et *wn*).

3.9 vn : vitesse horizontale du courant, composante méridienne

(en $m.s^{-1}$, variable 3D)
vn représente la composante horizontale de la vitesse du courant suivant l'axe des ordonnées (l'axe des latitudes). La moyenne du nombre de chiffres significatifs exacts de *vn* varie entre **9,40 et 9,84** (voir Fig 6.29 et 6.30, ANNEXE 10). **La précision moyenne de *vn* est plus régulière que celle de *un***. Elle semble se stabiliser sur les dernières itérations à 9,4 chiffres significatifs exacts.

Le nombre de chiffres significatifs exacts varie entre 2 et 14 (voir Fig 6.31, ANNEXE 10).

La précision de *vn* est indépendante de celle de *qsr* mais dépend de celle de *tn* (voir Fig 6.32, ANNEXE 10), pour les mêmes raisons que *un*.

3.10 wn : vitesse verticale du courant

(en $m.s^{-1}$, variable 3D)
wn est diagnostiqué à partir de *un* et*vn*. La moyenne du nombre de chiffres significatifs exacts de *wn* varie entre **7,95 et 8,60** (voir Fig 6.33 et 6.34, ANNEXE 11). **La précision moyenne de *wn* suit la même courbe que celle de *un* mais avec environ 1,4 chiffres significatifs exacts en moins**.

Le nombre de chiffres significatifs exacts varie entre 1 et 14 (voir Fig 6.35, ANNEXE 11).

On peut formuler les mêmes remarques concernant la précision de *wn* (voir Fig 6.36, ANNEXE 11) avec incertitude sur les données que pour *un*.

3.11 rotn : vorticité relative

(en s^{-1}, variable 3D)
rotn est un opérateur mathématique (le rotationnel) qui intervient dans les équations de la dynamique. Il dépend des dérivées spatiales de *un* et de *vn*. La moyenne du nombre de chiffres significatifs exacts de *rotn* varie entre **9,49 et 10,22** (voir Fig 6.37 et 6.38, ANNEXE 12). A partir de la $2^{ème}$ itération, la précision de *rotn* diminue régulièrement et semble se stabiliser autour de 9,5 chiffres significatifs exacts.

Le nombre de chiffres significatifs exacts varie entre 2 et 14 (voir Fig 6.39, ANNEXE 12).

Les remarques quand à l'influence de l'incertitude sur les données sur *rotn*(v oir Fig 6.40, ANNEXE 12) sont du même ordre que pour *vn*.

3.12 hdivn : divergence horizontale

(en s^{-1} , variable 3D)
hdivn est un opérateur mathématique (la divergence horizontale) qui intervient dans les équations de la dynamique. Il dépend des dérivés spatiales de *un* et de *vn*. La moyenne du nombre de chiffres significatifs exacts de *hdivn* varie entre **8,06 et 8,70** (voir Fig 6.41 et 6.42, ANNEXE 13). La courbe de sa précision à la même allure que celle de la précision de *un*.
Le nombre de chiffres significatifs exacts varie entre 1 et 14 (voir Fig 6.43, ANNEXE 13).

La précision de *hdivn* n'est pas influencée par celle de *qsr*, mais l'est légèrement par celle de *un* et *vn*, et de façon plus importante par celle de *tn* (voir Fig 6.44, ANNEXE 13).

3.13 spgu : gradient horizontal de pression de surface

(variable 2D)
spgu représente la composante horizontale du gradient de pression de surface. Elle est déduite de l'équation de mouvement. La moyenne du nombre de chiffres significatifs exacts de *spgu* varie entre **7,62 et 8,03** (voir Fig 6.45 et 6.46, ANNEXE 14). La précision de cette variable paraît assez "bruitée", surtout entre la $20^{ème}$ et la $25^{ème}$ itération.
Le nombre de chiffres significatifs exacts varie entre 2 et 14 (voir Fig 6.47, ANNEXE 14).

La précision de *spgu* n'est pas influencée par celle de *qsr*, mais l'est légèrement par celle de *un* et *vn*, et de façon plus importante par celle de *tn* (voir Fig 6.48, ANNEXE 14). Les résulats sur cette variable montrent bien que plus on avance dans les itérations, moins la précision de *spgu* est influencée par des incertitudes sur les données.

3.14 spgv : gradient vertical de pression de surface

(variable 2D)
spgv représente la composante verticale du gradient de pression de surface. Elle est

déduite de l'équation de mouvement. La moyenne du nombre de chiffres significatifs exacts de *spgv* varie entre **7,82 et 8,23** (voir Fig 6.49 et 6.50, ANNEXE 15).

Le nombre de chiffres significatifs exacts varie entre 2 et 14 (voir Fig 6.47, ANNEXE 15).

Concernant l'influence des incertitudes sur les données (voir Fig 6.48, ANNEXE 15), on peut tirer les mêmes remarques que pour la variable précédente.

3.15 Récapitulatif

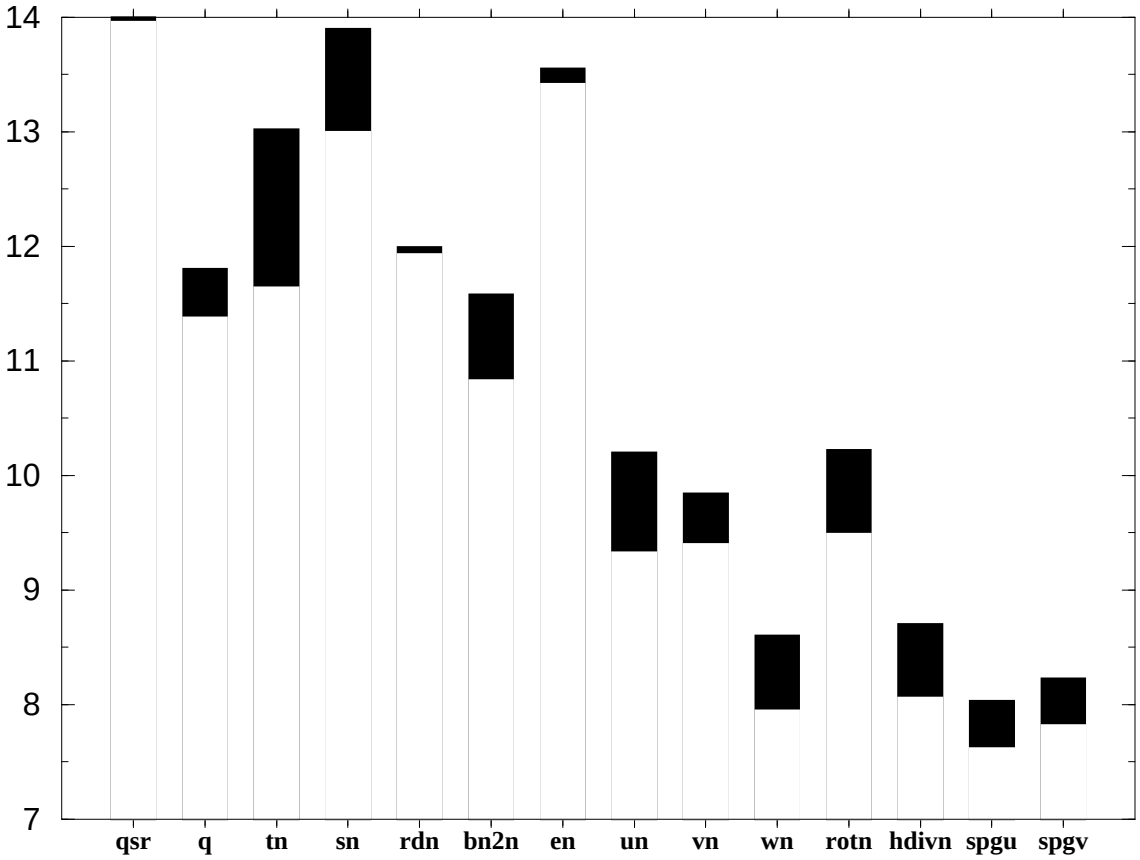

FIG. 3.1 – *Intervalle de précision des variables étudiées*

Dans l'ensemble, les variables ont une bonne précision qui diminue légèrement au cours des itérations, plus ou moins régulièrement. Les variables dites traceurs : *tn*, *sn*, *rdn* et *bn2n* ont une meilleure précision que les variables dynamiques du modèle : *un*, *vn*, *wn*, *rotn*, *hdivn*, *spgu* et *spgv*. Cela s'explique par l'importance et la complexité des calculs de l'équation du mouvement, qui font intervenir les variables dynamiques du modèle, et qui utilisent un algorithme numérique[2] pour calculer une partie des termes sous forme d'équation elliptique.

2. L'algorithme du gradient conjugué préconditionné

car ces derniers nécessitent des calculs plus nombreux et plus complexes et font intervenir un algorithme numérique

La plus grande perte de précision se produit lors de la première itération, excepté pour les variables *rotn* et q où la plus grande perte se produit lors de la deuxième, respectivement de la troisième, itération.

La répartition du nombre de chiffres significatifs exacts, à chaque itération, pour toutes les variables (sauf pour en), peut être approchée par une gaussienne.

Nous avons également remarqué pour quasiment toutes les variables que les pertes de précision dues à des incertitudes sur les données ne se cumulent pas.

D'autre part, d'une façon générale, ces pertes de précision s'estompent au cours des itérations. On peut donc supposer que sur un grand nombre d'itérations (1000 ou 1500) les incertitudes sur les données n'ont plus aucune influence.

Si les hypothèses précédentes sont vraies, et si elles le sont pour chaque variable et chaque donnée, cela implique que **la précision de chaque variable converge vers une valeur moyenne donnée** (dépendant de la nature de ses calculs). Dans ce cas, si l'on considère un grand nombre d'itérations, **initialiser le code à partir d'un fichier *restart*** (contenant des données déjà calculées) **n'a pas d'influence sur la précision des résultats**.

D'autres tests, notamment sur un grand nombre d'itérations, permettraient de confirmer ou non ces hypothèses et, le cas échéant de trouver la précision moyenne de chaque variable.

Chapitre 4

Pertes brutales de précision

Une fonctionnalité récente de CADNA permet de détecter toutes les opérations élémentaires où se produit une perte brutale de précision.
Plus précisément, si Z = X *op* Y, où *op* est un opérateur binaire, on a un message de perte brutale de précision, si

$$\min(\ \text{cestac(X)},\text{cestac(Y)}\ ) \geq \text{cestac(Z)} + \text{S}$$

La fonction *cestac()* renvoie le nombre de chiffres significatifs exacts de son argument. S est un entier que l'on choisit. On a affecté dans un premier temps S à 4 puis S à 6.

Nous n'avons considéré les pertes brutales de précision qu'au premier pas de temps de l'exécution car c'est à cette itération que la plupart des variables perdent le plus de précision.

variable	nb moyen de chiff. sign. exacts perdus	
	lors de la 1^{re} itération	entre la 2^{me} et la 30^{me} itération
q	-0.14	2.27
tn	0.26	2.08
sn	0.1	0.9
rdn	0.01	0.06
bn2n	1.79	1.01
en	0.03	0.54
un	3.68	0.95
vn	4.12	0.48
wn	4.72	0.71
rotn	1.32	3.2
hdivn	5.26	0.68
spgu	5.86	0.52
spgv	5.81	0.37

q est en partie calculé lors de la phase d'initialisation. Il regagne en moyenne 0,14 chiffre significatif exact lors de la 1^{re} itération, c'est pourquoi on trouve '-0.14' dans le tableau.

Toutes les pertes brutales de précision décrites ci-après [1] illustrent le phénomène classique de **cancellation**[16].

4.1 Pertes brutales de précision dans la subroutine *dtatem*

Dans le fichier *dtatem.F*, il y a quatre pertes brutales de précision, dues à la ligne suivante :

```
tdta(ji,jj,jk)=((1.-zxy)*temdta(ji,jj,jk,1)+zxy*temdta(ji,jj,jk,2))
```

Lors du premier pas de temps, les deux opérandes de l'addition ont 14 chiffres significatifs exacts. Le résultat, lui, n'en possède que 10, pour les coordonnées suivantes :

```
ji = 52;      jj = 122;     jk = 3;
ji = 149;     jj = 145;     jk = 5;
ji = 161;     jj = 32;      jk = 6;
ji = 67;      jj = 28;      jk = 8;
```

Une perte de quatre chiffres significatifs exacts ne paraît pas poser de problèmes ici.

Remarque :

Si l'on change l'ordre du calcul de la façon suivante :

```
tdta(ji,jj,jk) = zxy*(temdta(ji,jj,jk,2)
                        -temdta(ji,jj,jk,1))+temdta(ji,jj,jk,1)
```

On obtient deux pertes brutales de précision supplémentaires, pour :

```
ji = 110;      jj = 110;     jk = 2;
ji = 120;      jj = 31;      jk = 8;
```

4.2 Pertes brutales de précision dans la subroutine *eos*

Plusieurs lignes du fichier *eos.F* génèrent des pertes brutales de précision. Tout d'abord à la ligne :

```
zr1= ( ( ( ( 6.536332e-9*zt-1.120083e-6 )*zt + 1.001685e-4)*zt
        -9.095290e-3 )*zt + 6.793952e-2 )*zt + 999.842594
```

La deuxième addition génère 63 pertes brutales de 4 ou 5 chiffres significatifs exacts. Les deux opérandes de cette addition possèdent 14 chiffres significatifs exacts et sont de l'ordre de 10^{-2}. Le résultat, dans ces 63 cas, en possède 9 ou 10 et est de l'ordre de 10^{-4} ou 10^{-5}. Comme on additionne à ce résultat *999.842594* [2], la variable *zr1* possède 14 chiffres significatifs exacts dans tous les cas. Donc **ces pertes**

1. Elles correspondent bien à des points des océans : les masques correspondant aux variables valent 1

2. On affecte au pire le $13^{ème}$ chiffre après la virgule de *999.842594* soit son $16^{ème}$ chiffre significatif exact, or on ne travaille que sur 14 chiffres donc cela n'a ancun impact.

brutales de précision n'ont aucune conséquence.

La ligne :

```
zb = zbw + ze * zs
```

génère 36 pertes de 4 à 5 chiffres significatifs exacts car le produit $ze * zs$ donne un nombre proche de l'opposé de zbw.

La ligne :

```
zb1=   (-0.1909078*zt+7.390729 ) *zt-55.87545
```

génère une centaine de pertes de 4 à 5 chiffres significatifs exacts dues à la deuxième soustraction.

Enfin à la ligne

```
rdn(ji,jj,jk) = ( zrhop / ( 1.0 - zh/( zk0 - zh*( za-zh*zb ) ) )
    - rau0 ) / rau0 * tmask(ji,jj,jk)
```

la quatrième soustraction génère 83 pertes brutales de 4 ou 5 chiffres significatifs exacts.

4.3 Pertes brutales de précision dans la subroutine *bn2*

Le fichier *bn2.F* génère des pertes brutales de précision, dues à la ligne :

```
zs = 0.5*( sn(ji,jj,jk) + sn(ji,jj,jk-1) ) - 35.0
```

La deuxième soustraction génère **3133 pertes brutales de 4 à 5 chiffres significatifs exacts**.

4.4 Pertes brutales de précision dans la subroutine *inihdf*

Le fichier *inihdf.dyn.coef2d.h* génère des pertes brutales de précision à la ligne suivante :

```
ahm2(ji,jj) = zcoff * ahm0 + (1.-zcoff) * ahm2(ji,jj)
```

En effet, à chaque fois que *zcoff* vaut 1 avec 14 chiffres significatifs exacts, la soustraction génère **11818 pertes brutales de 14 chiffres significatifs exacts**. Le résultat de cette soustraction est alors non significatif.

Même si *zcoff* possède 14 chiffres significatifs exacts, les deux nombres qui forment son stochastique diffèrent de quelques bits. Donc quand on soustrait *zcoff* à 1, le stochastique correspondant est composé de deux nombres différents, donc non significatifs.
Par exemple pour ji=27 et jj=21, le stochastique *zcoff* est formé des deux nombres 1.000000000000014 et 0.9999999999999929 et possède 14 chiffres significatifs exacts, mais le stochastique *1-zcoff* est formé des deux nombres : $-1.4210854715202E-14$ et $7.105427357601052E-15$ et ne possède aucun chiffre significatif exact.

Cela signifie que lorqu'on travaille avec des réels, *zcoff* ne vaut pas tout-à-fait 1 et que donc sa différence avec 1 ne vaut pas 0 mais un nombre de l'ordre de 10^{-14} ou 10^{-15} qui n'est pas significatif.

$ahm2(ji, jj)$ vaut 40000 quels que soient ji et jj, $(1. - zcoff) * ahm2(ji, jj)$ est donc de l'ordre de 10^{-5}. Comme $zcoff * ahm0$ est de l'ordre de 40000, la deuxième opérande de l'addition ne vient pas le perturber. Le résultat $ahm2(ji, jj)$ possède 14 chiffres significatifs exacts. Donc **les pertes brutales de précision n'ont ici aucune conséquence**.

Il ne sera analysé par la suite que les pertes brutales de 6 chiffres significatifs exacts et plus.

4.5 Pertes brutales de précision dans la subroutine *flx*

Plusieurs lignes du fichiers *flx.forced.ecmwf.cli.h* génèrent des pertes brutales de précision.

```
1)   zicopa=tmask(ji,jj,1)*cvmgp2(zaux,zaux+1.,tn(ji,jj,1)-ztgel)

2)        qrp(ji,jj) = (  (1.-ziclim)  *zqrp +
     ziclim * ( (1-zicopa)*zqri+zicopa*zqrj )  ) * tmask(ji,jj,1)

3)          erp(ji,jj) = (  (1.-ziclim)* (1.-zicopa)* zerp   +
     ziclim * ( (1.-zicopa)*zeri+zicopa*zerp )  ) *tmask(ji,jj,1)

4)           emp(ji,jj) = (  (1.-ziclim)*(1.-zicopa)*ze  +
      (1.-ziclim)*zicopa*zei1 + ziclim*zicopa*zei2
       + runoff(ji,jj) + erp(ji,jj) + zwres  ) * tmask(ji,jj,1)

5)   tn(ji,jj,1)=max(tn(ji,jj,1),ztgel)

6)   zqrp = ztrp*(tb(ji,jj,1)-ztdta)

7)   zqrj = ztrp*min(zaux,tb(ji,jj,1)-ztgel)

8)   zerp = zsrp *(sb(ji,jj,1)-sdta(ji,jj,1))/(sb(ji,jj,1)+zepsi)
```

Ces lignes se situent dans une même boucle sur ji et jj.

La première ligne génère 15 pertes brutales de 6 à 7 chiffres significatifs exacts car *tn(ji,jj,jk)* et *ztgel* ont des valeurs proches. Cependant, dans les 15 cas, le résultat *zicopa* possède 14 chiffres significatifs exacts. **Les pertes de précision n'ont ici aucune conséquence**.

La deuxième ligne génère **environ 1300 pertes brutales de 14 chiffres significatifs exacts**. Dans tous ces cas, le stochastique *zicopa* vaut 1 avec 14

chiffres significatifs exacts, mais les deux nombres qui le forment diffèrent de quelques bits. *1.-zicopa* est formé des deux nombres $-1.4210854715202E-14$ et $7.105427357601052E-15$ et n'est donc pas significatif.
Quand *ziclim* vaut 1 et *zqrj* 0 au bit près avec 14 chiffres significatifs exacts, **le résultat** vaut *(1-zicopa)*zqri* et **n'est pas significatif**. Quand *ziclim* vaut 1 et que *zqrj* est différent de 0 (de l'ordre de 10^{-2}), le résultat vaut *zqrj* et **possède 3 à 8 chiffres significatifs exacts**. Enfin quand *ziclim* vaut zéro (dans 125 cas seulement) le résultat vaut *zqrp* et possède alors 14 chiffres significatifs exacts.

La troisième ligne génère **environ 1300 pertes brutales de 14 chiffres significatifs exacts**. *(1.-zicopa)* n'est pas significatif pour les mêmes raisons que précédemment.
zeri est de l'ordre de 10^{-5} donc *(1.-zicopa)* zeri* est de l'ordre 10^{-19} à 10^{-20}. Comme *zerp* est de l'ordre de 10^{-3} à 10^{-6}, il n'est pas affecté par l'addition de *(1.-zicopa)* zeri*. Comme, **dans la plupart des cas**, *ziclim* vaut 1 avec 14 chiffres significatifs exacts, le stochastique *erp(ji,jj)* vaut *zerp* et possède 10 à 14 chiffres significatifs exacts pour les cas observés. **Les pertes brutales de précison n'ont** alors **pas de conséquences**.
Dans les 125 cas où *ziclim* vaut 0, le résultat n'est pas significatif car il vaut *(1.-zicopa)* zerp*.

La quatrième ligne engendre 45 pertes brutales de précision. Pour les mêmes raisons que pour la deuxième ligne, *1.-zicopa* n'est pas significatif. Cependant, dans les 45 cas, *ziclim* vaut 1. au bit près donc *1.-ziclim* vaut 0 avec 14 chiffres significatifs exacts. Le produit *(1.-ziclim)*(1.-zicopa)* vaut 0 avec 14 chiffres significatifs exacts. **Ces pertes brutales de précision n'ont aucune conséquence.**

A la cinquième ligne (voir aussi à 3.2.3 Instabilités dues à la fonction *fsfzpt*), la soustraction *tn(ji,jj,1)-ztgel* opérée par la fonction *max* entraîne **15 pertes de 5 à 7 chiffres significatifs exacts** car *tn(ji,jj,1)* et *ztgel* ont des valeurs proches. **Le résultat *tn(ji,jj,1)* possède 2 à 4 chiffres significatifs exacts.**

La sixième ligne entraîne **582 pertes brutales de précision**. *ztdta* est proche de *tb(i,jj,1)* et possède 8 à 11 chiffres significatifs exacts. *tb(ji,jj,1)-ztdta* et par conséquent *zqrp* n'en possèdent que 3 à 6.

La septième ligne n'entraîne que 51 pertes brutales de précision pour les mêmes types de raison que pour la ligne précedente.

Enfin la huitième ligne génère 8 pertes brutales de 6 à 8 chiffres significatifs exacts car *sb(ji,jj,1)* et *sdta(ji,jj,1)* ont des valeurs proches. Le résultat possède le même nombre de chiffres significatifs exacts.

4.6 Pertes brutales de précision dans la subroutine *hdfslp*

Plusieurs lignes du fichier *hdfslp.F* génèrent des pertes brutales de précision.

```
1)     zav = 1. / e2v(ji,jj) * ( rdn(ji,jj+1,jk) - rdn(ji,jj,jk) )

2)     zau = 1. / e1u(ji,jj) * ( rdn(ji+1,jj,jk) - rdn(ji,jj,jk) )

3) zai = zcoef1* ( (rdn(ji+1,jj,jk)-rdn(ji,jj,jk))* umask(ji,jj,jk)
    + ( rdn(ji+1,jj,jk-1) - rdn(ji,jj,jk-1) )    * umask(ji,jj,jk-1)
    + ( rdn(ji,jj,jk-1) - rdn(ji1,jj,jk-1) )  * umask(ji-1,jj,jk-1)
    + ( rdn(ji,jj,jk)-rdn(ji-1,jj,jk) )    * umask(ji-1,jj,jk) )
          * tmask(ji,jj,jk)

4) zaj = zcoef2* ( (rdn(ji,jj+1,jk)-rdn(ji,jj,jk)) *vmask(ji,jj,jk)
   + ( rdn(ji,jj+1,jk-1) - rdn(ji, jj ,jk-1) ) *vmask(ji, jj ,jk-1)
   + ( rdn(ji,jj,jk-1) - rdn(ji,jj-1,jk-1) ) * vmask(ji,jj-1,jk-1)
   + ( rdn(ji,jj,jk) - rdn(ji,jj-1,jk) )   *vmask(ji,jj-1,jk) )
          * tmask(ji,jj,jk)

5)  wslpi(ji,jj,jk) = ( zwz(ji-1,jj-1) + zwz(ji+1,jj-1)
                      + zwz(ji-1,jj+1) + zwz(ji+1,jj+1)
                      + 2.*( zwz(ji  ,jj-1) + zwz(ji-1,jj  )
                      + zwz(ji+1,jj  ) + zwz(ji  ,jj+1) )
                      + 4.*  zwz(ji,jj  )       ) * zcofw

6)  wslpj(ji,jj,jk) = ( zww(ji-1,jj-1) + zww(ji+1,jj-1)
                      + zww(ji-1,jj+1) + zww(ji+1,jj+1)
                      + 2.*( zww(ji  ,jj-1) + zww(ji-1,jj  )
                      + zww(ji+1,jj  ) + zww(ji  ,jj+1) )
                      + 4.*  zww(ji,jj  )       ) * zcofw
```

La soustraction de la première ligne génère **760 pertes de 6 à 8 chiffres significatifs exacts qui se répercutent sur le résultat *zav***. On passe de 12 ou 13 chiffres significatifs exacts pour *rdn(ji,jj+1,jk)* et *rdn(ji,jj,jk)* à 5 ou 6 chiffres significatifs exacts pour *zav*.
Par exemple pour ji=45, jj=7, *rdn(ji,jj+1,jk)* vaut $0.749985168559E-002$ avec 12 chiffres significatifs exacts, *rdn(ji,jj,jk)* vaut $0.749986710919E-002$ avec 12 chiffres significatifs exacts, *rdn(ji,jj+1,jk)* - *rdn(ji,jj,jk)* vaut $-0.154236E-007$ avec 6 chifres significatifs exacts. Comme *e2v(ji,jj)* vaut $0.5676280405038E+005$ avec 14 chiffres significatifs exacts, *zav* vaut -0.271720E-012 avec 6 chiffres significatifs exacts.

Sur le même modèle, la soustraction de la deuxième ligne génère **690 pertes de**

6 à 8 chiffres significatifs exacts qui se répercutent sur *zau*.

La troisième ligne génère **2232 pertes brutales de 6 chiffres significatifs exacts ou plus. Le résultat *zai* possède alors moins de 8 chiffres significatifs exacts dans environ 800 cas**.

La quatrième ligne génère **2266 pertes brutales de 6 chiffres significatifs exacts ou plus. Dans 840 de ces cas environ, *zaj* possède moins de 8 chiffres significatifs exacts.**

La cinquième ligne génère **26528 pertes brutales de 6 chiffres significatifs exacts ou plus. Le résultat possède moins de 8 chiffres significatifs exacts dans 150 cas environ**. (Il en possède 14 dans environ 21500 cas).

De la même façon, la sixième ligne génère **31985 pertes brutales de 6 chiffres significatifs exacts ou plus. Le résultat possède moins de 8 chiffres significatifs exacts dans environ 100 cas**. (Il en possède 14 dans 29100 cas).

4.7 Pertes brutales de précision dans la subroutine *trazad*

Le fichier *trazad.F* génère **1293 pertes brutales de 6 à 10 chiffres significatifs exacts** dues à la ligne :

```
sa(ji,jj,jk) =  sa(ji,jj,jk) + zsa
```

Le résultat *sa(ji,jj,jk)* possède 4 à 8 chiffres significatifs exacts dans 1293 cas.

4.8 Pertes brutales de précision dans la subroutine *dynhpg*

Le fichier *dynhpg.F* génère des pertes brutales de précision, dues aux lignes :

```
1)   zhpi(ji,jk) = zhpi(ji,jk-1) + zcoef1 *
                (  ( rdn(ji+1,jj,jk) + rdn(ji+1,jj,jk-1) )
                 - ( rdn(ji,jj,jk)   + rdn(ji,jj,jk-1)   )  )
                           / e1u(ji,jj)
2)  zhpj(ji,jk) = zhpj(ji,jk-1) + zcoef1 *
                (  ( rdn(ji,jj+1,jk) + rdn(ji,jj+1,jk-1) )
                 - ( rdn(ji,jj,  jk) + rdn(ji,jj  ,jk-1) )  )
                              / e2v(ji,jj)
```

La soustraction de la première ligne engendre 428 pertes brutales de 6 à 7 chiffres significatifs exacts. Le résultat *zhpi(ji,jk)* possède alors 6 à 12 chiffres significatifs

exacts.
De la même manière, la soustraction de la deuxième ligne engendre 386 pertes brutales de 6 à 7 chiffres significatifs exacts. Pour ces 386 cas, *zhpj(ji,jk)* possède 7 à 12 chiffres significatifs exacts.

4.9 Pertes brutales de précision dans la subroutine *dynspg*

Le fichier *dynspg.freesurf.cstvol.h* génère des pertes brutales de précision aux lignes suivantes :

```
1)            ua(ji,jj,jk) = ua(ji,jj,jk) + spgu(ji,jj)

2)            va(ji,jj,jk) = va(ji,jj,jk) + spgv(ji,jj)
```

La première ligne en **génère 128**. **Le résultat *ua(ji,jj,jk)* possède alors 2 à 8 chiffres significatifs exacts**.

La deuxième ligne en **génère 276**. **Le résultat *va(ji,jj,jk)* possède alors 3 à 8 chiffres significatifs exacts**.

4.10 Pertes brutales de précision dans la subroutine *bn2*

Le fichier *bn2.F* génère des pertes brutales de précision dues à la ligne :

```
bn2n(ji,jj,jk) = zgde3w * zbeta * tmask(ji,jj,jk)
               * ( zalbet * ( tn(ji,jj,jk-1) - tn(ji,jj,jk) )
                          - ( sn(ji,jj,jk-1) - sn(ji,jj,jk) ) )
```

La première et la troisième soustraction génèrent **11354 pertes brutales de précision**. **Le résultat possèdent 4 à 8 chiffres significatifs exacts dans 9453 cas**. Il en possède 9 à 13 dans les autres cas.

4.11 Pertes brutales de précision dans la subroutine *div*

La première et la deuxième soustraction de la ligne suivante du fichier *div.F* génère 33 pertes brutales de précision :

```
hdivn(ji,jj,jk) = (  zwu(ji,jj) - zwu(ji-1,jj)
                  + zwv(ji,jj) - zwv(ji  ,jj-1)  ) / zbt
```

Le résultat possède alors 3 à 9 chiffres significatifs exacts.

Chapitre 5

Etude des instabilités numériques

Lors de l'exécution du code, la bibliothèque CADNA permet de vérifier son bon déroulement, du point de vue numérique. A chaque détection d'une instabilité, un message peut être écrit dans le fichier *cadna_stability_f90.lst* qui a été renommé en *instabilite*. Cependant pour localiser les instabilités, nous avons eu recours à un debugeur, ce qui a encore augmenté le coût de calcul.

5.1 Instabilités dans la subroutine *domhgr*

Dans le fichier *domhgr.F*, il y a quatre tests instables, qui se trouvent dans une boucle sur *ji* (qui correspond à la longitude) et *jj* (qui correspond à la latitude) :

```
      IF ( tmask(ji,jj,1).NE.0. ) THEN

1)          IF ( e1t(ji,jj).LT.ze1min ) THEN
                ze1min = e1t(ji,jj) * tmask(ji,jj,1)

2)          IF ( e2t(ji,jj).LT.ze2min ) THEN
                ze2min = e2t(ji,jj) * tmask(ji,jj,1)

3)          IF ( e1t(ji,jj).GT.ze1max ) THEN
                ze1max = e1t(ji,jj) * tmask(ji,jj,1)

4)          IF ( e2t(ji,jj).GT.ze2max ) THEN
                  ze2max = e2t(ji,jj) * tmask(ji,jj,1)
```

Ces quatre tests sont instables car la différence entre les opérandes gauches et droites des différents tests n'est pas significative. Nous allons voir que **cela n'a pas d'importance ici**.
Prenons l'exemple du premier test. *tmask(ji,jj,1)* vaut 0 ou 1, ici il vaut 1. Pour $ji = 45$ et $jj = 2$, on affiche grâce à la fonction *str()* de CADNA les chiffres significatifs exacts des variables *e1t(ji,jj)* et *ze1min*. Pour ces deux variables on obtient : $0.4709605108493E+005$

Ces variables ont 14 chiffres significatifs décimaux exacts[1]. Cependant les nombres correspondant à chacun de ces stochastiques diffèrent par leur dernier bit. Donc la différence entre *e1t(ji,jj)* et *ze1min* n'est pas significative en binaire. CADNA considère le test comme instable. L'instruction conditionnée par ce test est exécutée et *ze1min* vaut alors *e1t(ji,jj)* au bit près.

5.2 Instabilités dans la subroutine *day*

Dans le fichier *day.F*, il y a trois instabilités dues aux lignes suivantes :

```
nday0 = int( gday0 )
nday = nit000 + int( rday / rdttra(1) )
nday1 = int( gday1 )
```

Concernant la première ligne, la variable stochastique *gday0* est formée des deux nombres suivants : 3650 et **3649.99...9**. Elle a été calculée à la ligne précédente :

```
gday0   = ( float( nit000-1 ) * rdttra(1) ) / rday
```

La fonction fortran *int* tronque la variable passée en argument. Les deux nombres correspondant à la variable stochastique *gday0* n'ont pas la même partie entière :

```
int(3650.) = 3650
int(3649.99...9) = 3649
```

Un message d'instabilité est donc généré. A la variable *nday0*, est affecté l'entier 3649.

Pour éviter le message d'instabilité, une solution consiste à remplacer la fonction *int* par *nint* qui, elle, arrondit la valeur de l'argument à l'entier le plus proche. Ainsi ***nday0* vaut 3649 sans cette modification et 3650 avec**.

La variable *gday1* possède la même valeur que *gday0* donc nous avons également remplacé *int* par *nint*.

Les deux nombres correspondant au stochastique *rday/rdttra(1)* sont 15 et 14.99...9. Ces deux nombres n'ont pas la même partie entière : 15 et 14. La fonction *int* génère donc un message d'instabilité. Nous l'avons, ici aussi, remplacée par *nint*.

Il n'y a plus d'instabilité avec la fonction *nint*.

5.3 Instabilités dues à la fonction *fsfzpt*

Certaines instabilités sont dues à la fonction *cvmgp2*. Cette fonction, qui se trouve dans le fichier *Functionexp2.F*, est appelée dans le fichier *flx.forced.ecmwf.cli.h* à la ligne suivante :

```
zicopa=tmask(ji,jj,1)*cvmgp2(zaux,zaux+1., tn(ji,jj,1)-ztgel)
       avec zaux initialisé à 0.
```

Nous rappelons ici le contenu de la fonction *cvmgp2* :

```
FUNCTION cvmgp2(a,b,c)
#include"parameter.h"
```

1. Par convention la fonction *str()* de CADNA affiche au maximum 13 chiffres signicatifs exacts.

```
#include"common.h"
TYPE(SINGLE_ST) a,b,c,cvmgp2
IF(c.ge.0.0) THEN
   cvmgp2=a
ELSE
   cvmgp2=b
ENDIF
RETURN
END
```

Or nous nous sommes aperçus que *tn(ji,jj,1)-ztgel* est ici non significatif car *tn(ji,jj,1)* est nul et *ztgel* n'est pas significatif donc le test ci-dessus est instable. En effet *ztgel* est défini quelques lignes auparavant, toujours dans le fichier *flx.forced.ecmwf.cli.h* :

```
ztgel = fsfzpt(sn(ji,jj,1),zaux1)      avec zaux1 = 0.
```

Quand *sn(ji,jj,1)* vaut 0.0 avec 14 chiffres significatifs exacts, ce qui est fréquent lors du premier pas de temps, ztgel devient non significatif.

La fonction *fsfzpt* est la suivante :

```
FUNCTION fsfzpt( pfs, pfp)
#include "parameter.h"
#include "common.h"
TYPE(SINGLE_ST) pfs, pfp, fsfzpt
fsfzpt = ( -0.0575 + 1.710523e-3 * sqrt(pfs)
$                  - 2.154996e-4 *      pfs  ) * pfs
$                  - 7.53e-4 * pfp
RETURN
END
```

Quand les deux arguments de *fsfzpt* sont nuls, le résultat devrait être nul. Mais il vaut en fait (pour jk = 1) :

```
pour ji=1, jj=1
        ztgel = 0.E+0 et -1.882500000000017E-4,
                       (aucun chiffre significatif exact)
pour ji=2, jj=1
        ztgel = 0.E+0 et -2.055657645087178E-9,
                       (aucun chiffre significatif exact)
pour ji=3, jj=1
        ztgel = 0.E+0 et -2.649804830298333E-9,
                        (aucun chiffre significatif exact)
...
pour ji=10, jj=1
        ztgel = 0.E+0 et  3.375610567218238E-9,
                         (aucun chiffre significatif exact)
```

et ainsi de suite pour toutes les coordonnées où *sn(ji,jj,1)* vaut 0.0.

Comme ztgel est non significatif, cela entraîne des instabilités dans la fonction *max* [2] et dans la fonction *min* dans le fichier *flx.forced.ecmwf.cli.h* aux lignes suivantes :

```
tn(ji,jj,1)=max(tn(ji,jj,1),ztgel)
ztdta = max(sst(ji,jj),ztgel)
zqrj = ztrp*min(zaux,tb(ji,jj,1)-ztgel)
```

Quelle que soit l'itération pour laquelle *ztgel* est non significatif, *tb(ji,jj,jk)* vaut 0 avec 14 chiffres significatifs exacts. Donc *tb(ji,jj,1)-ztgel* est non significatif et *max(tn(ji,jj,1),ztgel), max(sst(ji,jj),ztgel) et min(zaux,tb(ji,jj,1)-ztgel)* valent 0 avec 14 chiffres significatifs exacts.

Cependant, pour chaque coordonnées où *ztgel* est non significatif, cela correspond à un point de la terre (et non des océans). Donc **ces instabilités ne sont pas à prendre en compte**.

5.4 Instabilités dans la subroutine *trazdf*

Dans le fichier *trazdf.isopycnal.h*, il se produit des multiplications instables dues à la ligne suivante :

```
        avt(ji,jj,jk) = avt(ji,jj,jk)
$         + fsahtw(ji,jj,jk) * ( wslpi(ji,jj,jk)*wslpi(ji,jj,jk)
$                              +wslpj(ji,jj,jk)*wslpj(ji,jj,jk) )
```

wslpi(ji,jj,jk) est 15 fois non significatif et *wslpj(ji,jj,jk)* 16 fois, mais pour des indices différents. Chacun de ces indices correspond bien à un point de l'océan (*tmask(ji,jj,jk)* vaut 1).

Ces instabilités ont très peu de conséquences sur le résultat *avt(ji,jj,jk)*.

Prenons l'exemple de *wslpi(ji,jj,jk)* non significatif. Pour les coordonnées correspondantes, *avt(ji,jj,jk)* est de l'ordre de 10 avec 14 chiffres significatifs exacts, exceptés *avt(72,77,2)* qui est de l'ordre de 10^{-3} avec 12 chiffres significatifs exacts et *avt(60,81,2)* qui est de l'ordre de 10^{-2} avec 13 chiffres significatifs exacts. Sachant que pour le résultat *avt(72,77,2)*, la perte d'un chiffre significatif exact est due à l'imprécision de la donnéé *avt(72,77,2)*, **le fait que *wslpi(ji,jj,jk)* soit non significatif n'entraîne la perte que d'un chiffre significatif exact sur *avt(ji,jj,jk)*, et cela pour deux coordonnées seulement**.

5.5 Récapitulatif

Aucune instabilité grave n'a été détectée dans le code. Du fait de l'importance des temps de calcul, seules les cinq premières itérations ont été testées. Il est à

2. N.B. : avec CADNA, $max(0.0, @.0) = 0.0$ mais sans CADNA, $max(0.0, 3.375610567218238E-9) = 3.375610567218238E-9$. Donc **l'utilisation de CADNA peut augmenter la précision des résultats**, donc par la même le nombre de chiffres significatifs exacts des résultats.

noter que si les lignes de codes engendrant des instabilités à la première itération en ont engendré aux itérations suivantes, elles n'ont pu être détectées.

Le code est donc relativement stable numériquement.

Chapitre 6

Grille de déroulement du stage

Semaine du 31 mai au 4 juin : Mise en place de l'environnement informatique nécessaire au stage à l'IPSL et au LIP6. Apprentissage de l'éditeur de texte Emacs. Lecture de documentations sur OPA8.1 et CADNA.

Semaine du 7 au 11 juin : Instrumentation d'ORCA avec la bibliothèque CADNA.

Semaine du 14 au 18 juin : Fin de la "Cadnatisation" d'ORCA. Apprentissage du logiciel de traitement de texte professionnel Latex. Rapport sur la première et la deuxième partie du stage.

Semaine du 21 au 23 juin et du 1^{er} au 2 juillet : Traitement des erreurs de compilation. Diverses corrections sur la première partie du rapport.

Semaine du 5 au 9 juillet : Réunion avec Fabienne Jezequel, Marie-Alice Foujols et Maurice Imbard pour faire une mise au point sur l'avancement du stage et voir les directions à prendre. Apprentissage du debugger *totalview*. Début de l'étude des instabilités et du nombre de chiffres significatifs exacts des variables.

Semaine du 12 au 16 juillet : Obtention du nombre de chiffres signicatifs des variables et de leur répartition jusqu'à la quatrième, puis jusqu'à la septième itération. Apprentissage du debugger *cdbx*. Suite de l'étude des instabilités.

Semaine du 19 au 23 juillet : Suite de l'étude des instabilités.

Semaine du 26 au 30 juillet : idem. Essais d'obtention de la précision des variables pour 10 itérations.

Semaine du 30 août au 3 septembre : idem. Début de l'étude des pertes brutales de précision qui a continué jusqu'à la fin du stage. Etude des instabilités.

Semaine du 6 au 10 septembre : Obtention de la précision des variables pour 10 itérations. Apprentissage de *xmgr* pour la mise en forme des résultats. Etude des instabilités.

Semaine du 13 au 17 septembre : Obtention de la précision des variables pour 30 itérations, avec une incertitude sur qsr. Etude des instabilités.

Semaine du 20 au 24 septembre : Obtention de la précision des variables sur 30 itérations, sans incertitude sur les données.

Semaine du 27 au 1^{er} octobre : Obtention de la précision des variables sur 5 itérations, avec une incertitude sur un et vn puis avec une incertitude sur tn.

Semaine du 4 au 8 octobre : Obtention de la précision des variables sur 10 itérations, avec incertitude sur un et vn ainsi que sur tn. Présentation orale des résultats (au LIP6).

Semaine du 11 au 15 octobre : Etude des instabilités sur 5 itérations. Présentation orale des résultats (au LODYC).

Semaine du 18 au 22 octobre : Recherche des pertes brutales de précision. Rédaction du rapport.

Semaine du 25 au 28 octobre : Obtention des derniers résultats. Rédaction finale du rapport.

Conclusion

Travailler sur un code, à la pointe de la recherche, qui simule tous les océans, est particulièrement intéressant et motivant. D'autant plus que mon travail a permis de contrôler ce code d'un point de vue numérique.

Ce stage m'a également familiarisé avec la bibliothèque CADNA, qui, j'en suis convaincue, sera de plus en plus utilisée, que ce soit en milieu universitaire ou industriel.

Ce stage a donc permis de montrer la bonne qualité numérique du code ORCA.

Toutes les variables étudiées ont une bonne précision qui se stabilise au cours des itérations. En effet, sous réserve que les résultats soient confirmés par des tests sur un nombre important d'itérations[1], la précision de chaque variable converge vers une valeur particulière, qui dépend de la nature de ses calculs dans le code. Ce qui signifie que la longueur des simulations n'est pas proportionnelle à l'imprécision des variables, contrairement à ce que l'on aurait pu penser.

D'autre part les lignes de code engendrant les principales pertes brutales de précision ont été mises en évidence.

Enfin aucune instabilité majeure n'a été détectée durant les cinq premières itérations[2]. Il est à noter que le nombre de tests s'est trouvé limité, la durée des calculs étant particulièrement importante avec la bibliothèque CADNA.

Si, dans le futur, CADNA est vectorisé, il sera alors possible de lancer des simulations d'ORCA, avec la bibliothèque CADNA, sur un nombre bien plus important d'itérations. Cela permettrait de confirmer et d'appuyer les ébauches de résultats mis en évidence tout au long de ce stage. On pourrait également envisager d'utiliser la bibliothèque CADNA pour évaluer la qualité de la convergence dans l'algorithme du gradient conjugué.

1. De l'ordre de 1000 ou 1500 itérations par exemple
2. Les instabilités trouvées à la première itération, qui n'avaient pas de conséquence sur les calculs n'ont pas été testées sur les itérations suivantes

Bibliographie

[1] Site de l'IPSL : http://www.ipsl.jussieu.fr

[2] Site de LODYC : http://www.lodyc.jussieu.fr

[3] Site du LIP6 : http://www.lip6.fr

[4] Caroline TESSIER, La configuration ORCA : Calculs des moyennes zonales et des flux Nord-Sud, Rapport de stage d'initiation à la recherche, Novembre 1998 - Juin 1999

[5] Gurvan Madec, Pascale Delecluse, Maurice Imbard et Claire Lévy, Notes du Pôle de Modélisation de l'IPSL n°11 de décembre 1998 sur OPA 8.1, Laboratoire d'Océanologie DYnamique et de Climatologie

[6] Levitus S., 1982, Climatological Atlas of the World Ocean, NOAA professional paper 13, december 1982

[7] Fabienne JÉZÉQUEL et Jean-Marie CHESNEAUX, Etude de la stabilité numérique du code OPA-version 8.0
http://www.ipsl.jussieu.fr/modelisation/liste-notes-tech.html

[8] Jean-Marie CHESNEAUX, Stéphane GUILAIN et Jean VIGNES, La bibliothèque CADNA, Présentation et utilisation
http://www-anp.lip6.fr/chpv/francais/cadna/index.html

[9] Jean-Marie CHESNEAUX, L'arithmétique stochastique et le logiciel CADNA, Habilitation à diriger des recherches, Université de Pierre et Marie Curie, Paris (1995)

[10] IEEE Standard for Binary Floating-Point Arithmetic, ANSI/IEEE Standard 754-1985. Institue of Electrical and Engineers, Août 1985.

[11] Michel LA PORTE et Jean VIGNES, Algorithmes numériques-Analyse et mise en œuvre, Vol. 1, Arithmétique des Ordinateurs-Systèmes linéaires, Editions Technip, Paris, 1974

[12] Jean VIGNES, Estimation de la précision des résultats de logiciels numériques, *La Vie des Sciences* 7 (2) (1990) 93-145

[13] Jean VIGNES, A stochastic arithmetic for reliable scientific computation, *Math. Comput. Simulation* 35 (1993) 223-261

[14] Jean VIGNES, Zéro mathématique et zéro informatique, C.R. *Acad. Sci. Paris Sér. I Math.* 303 (1986) 997-1000; also : *La Vie des Sciences* 4 (1) (1987) 1-13

[15] M. DUBESSET, J.VIGNES, Fortran, Le langage normalisé, Editions Techni, Paris, 1991

[16] Qualité des calculs sur ordinateurs, Rédaction collégiale, Editions Masson, Paris, 1997

[17] Jean-Luc LAMOTTE, A new approach for the study of surface interpolation with uncertain data using the CADNA library, proceeding of the International Workshop on Reliable Computations and Interval Algebra, pages 9-10, Sozopol, Bulgaria, Sept 1999.

ANNEXE 1

Liste des procédures appelées

La liste ci-dessous présente les principales procédures nécessaires à l'exécution de la version du code ORCA qui nous a été fournie. Chaque procédure 'nom' correspond à une subroutine 'nom.F'. Les procédures sont classées par ordre d'appel.

opa	programme principal

Première partie : initialisation

inipar	initialisation des paramètres
parcst	initialisation des constantes
parlec	lecture du fichier namelist
parctl	options de contrôle
ctlopn	ouverture d'un fichier et vérifications
inidom	initialisation du domaine
dommba	bathymétrie
dommsk	masques
domhgr	maillage horizontal
domzgr.z	maillage vertical (coordonnée z)
domzgr.s	maillage vertical (coordonnée s)
domstp	pas de temps
domwri	écriture d'un fichier de masques
inisol	initialisation de l'équation elliptique
solmat	calcul de la matrice des îles
inidta	initialisation des données
dtatem	lecture de la température initiale ou de rappel
read3d	routine de lecture 3D
dtasal	lecture de la salinité initiale ou de rappel
read3d	routine de lecture 3D
dtacof	coefficients de rappel
read3	routine de lecture 3D
write3	routine d'écriture 3D
inidtr	initialisation des traceurs dynamiques
dtrlec	lecture du fichier restart
read3	routine de lecture 3D
read2	routine de lecture 2D
dtrtem	initialisation de la température
dtrsal	initialisation de la salinité
eos	équation d'état
bn2	fréquence de Brunt-Vaisala
wzv	vitesse verticale

ANNEXE 1

inihdf	schéma de diffusion horizontale
inizdf	schéma de diffusion verticale
zdftke	équation de l'énergie turbulante (fermeture de 1,5)
step	routine de pas de temps

Deuxième partie : boucle sur le pas temps

Forcages et données

day	calendrier pour le couplage
ocfzpt	température de congélation de l'eau de mer
dtatem	lecture de la température initiale ou de rappel
read3d	routine de lecture des données 3D
dtasal	lecture de la salinité initiale ou de rappel
read3d	routine de lecture des données 3D
tau.forced	contrainte du vent en mode forcé
read2d	routine de lecture 2D
repere	changement de maillage pour vecteur
dtasst	données de la température à la surface de l'océan
flx.forced	contrainte flux de chaleur et évaporation/précipitation

Physique des océans

zdftke	équation de l'énergie turbulente (fermeture de 1.5)
zdfevd	schéma d'ajustement vertical
zdfbfr	frottement des fonds
hdfslp	calcul de la pente isopycnale
lbc	conditions de périodicité

Tendances des traceurs

trahad	advection horizontale des traceurs
tradmp	rappel des traceurs
trahdf.isopycnal	schéma de diffusion horizontale des traceurs
trazad	schéma d'advection verticale des traceurs
traqsr	schéma de flux solaire pénétrant
trazdf.implicit	schéma de diffusion verticale des traceurs

ANNEXE 1

Tendances des dynamiques

dynkeg	gradient d'énergie cinétique
dynvor.enstrophy	terme de vorticité
dynvor.energy	terme de vorticité
dynvor.combined	terme de vorticité
dynhdf.laplacian	diffusion horizontale des dynamiques
dynhpg	calcul de gradient de pression hydrostatique
dynzad	schéma d'advection verticale des dynamiques
dynzdf.implicit	schéma de diffusion verticale des dynamiques
dynspg	gradient de pression de surface
solpcg	algorithme du gradient conjugué

changement de pas de temps

tranxt	calcul des tendances des traceurs
dynnxt	calcul de l'équation dynamique

variables de diagnostic

eos	diagnostic de l'équation détat
bn2	diagnostic de la fréquence de Brunt-Vaisala
div	diagnostic de la divergence horizontale
cur	diagnostic de la vorticité ou du tourbillon relatifs
wzv	vitesse verticale
stpctl	contrôle de l'exécution en fin de pas de temps
diadyn	diagnostic de la tendance des dynamiques
diamxl	diagnostic de l'épaisseur des couches non-homogènes
diahth	diagnostic de l'épaisseur des thermoclines

fin de la boucle sur le pas de temps

ANNEXE 2

Maillage utilisé par ORCA

ANNEXE 3

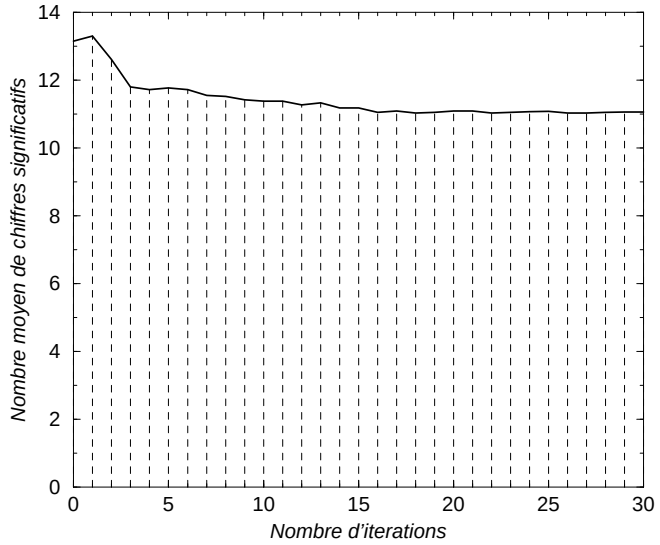

FIG. 6.1 – *Evolution de la précision de* **q** *au cours des 30 premières itérations*

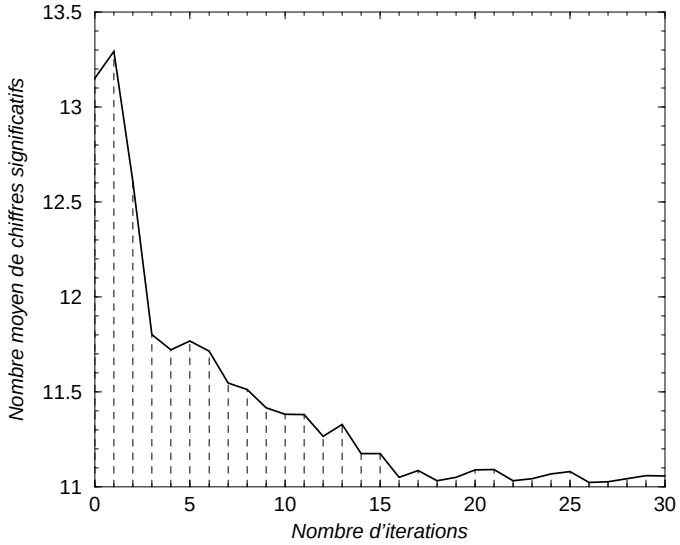

FIG. 6.2 – *Zoom sur le schéma précédent. Amplitude de 2.4 chiffres*

ANNEXE 3

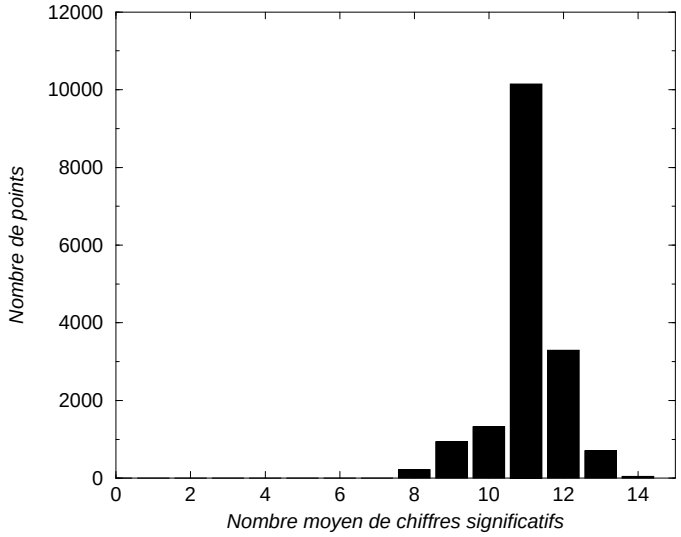

FIG. 6.3 – *Répartition du nombre de chiffres significatifs de* $\mathbf{q}$ *à la* 30^e *itération*

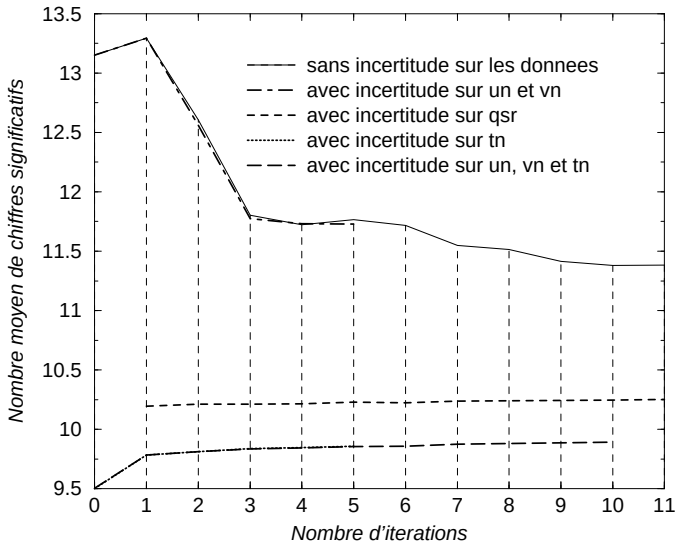

FIG. 6.4 – *Evolution de la précision de* $\mathbf{q}$ *au cours des 11 premières itérations*

ANNEXE 4

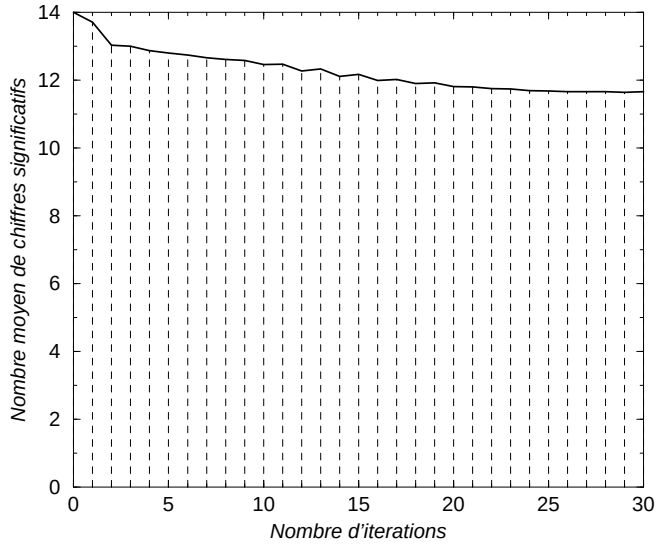

FIG. 6.5 – *Evolution de la précision de* **tn** *au cours des 30 premières itérations*

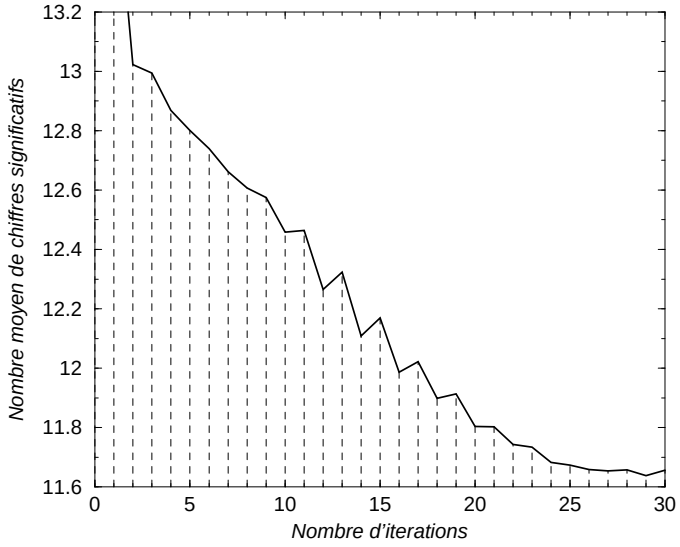

FIG. 6.6 – *Zoom sur le schéma précédent. Amplitude de 1.6 chiffre*

ANNEXE 4

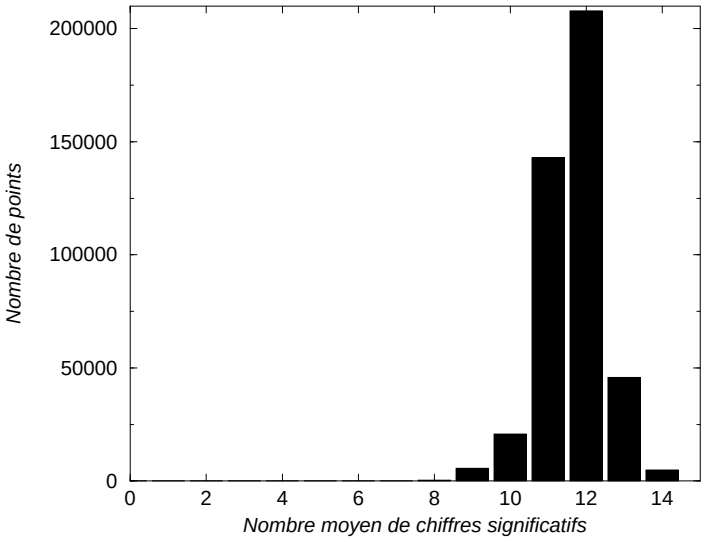

FIG. 6.7 – *Répartition du nombre de chiffres significatifs de* **tn** *à la* 30^e *itération*

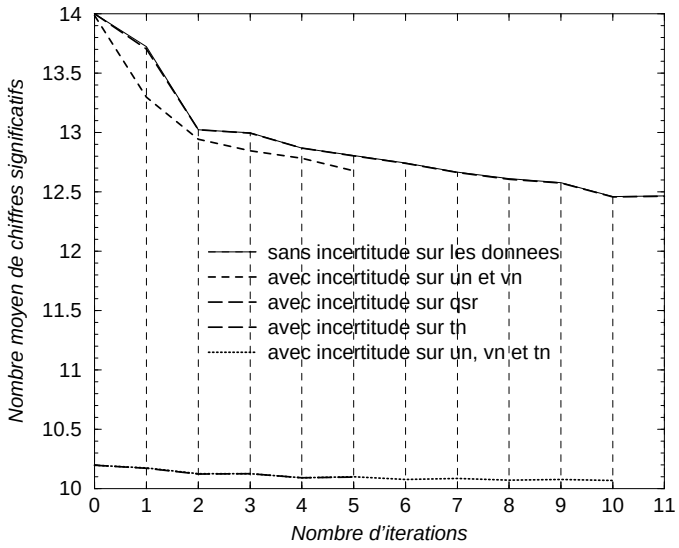

FIG. 6.8 – *Evolution de la précision de* **tn** *au cours des 11 premières itérations*

ANNEXE 5

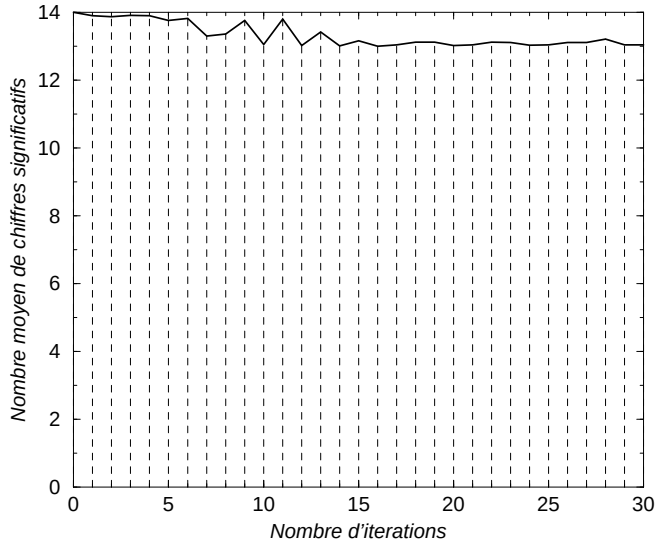

FIG. 6.9 – *Evolution de la précision de* **sn** *au cours des 30 premières itérations*

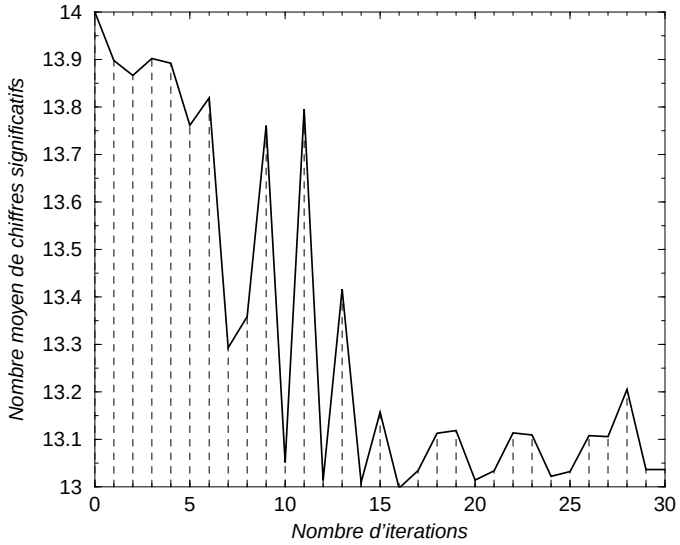

FIG. 6.10 – *Zoom sur le schéma précédent. Amplitude de 1 chiffre*

ANNEXE 5

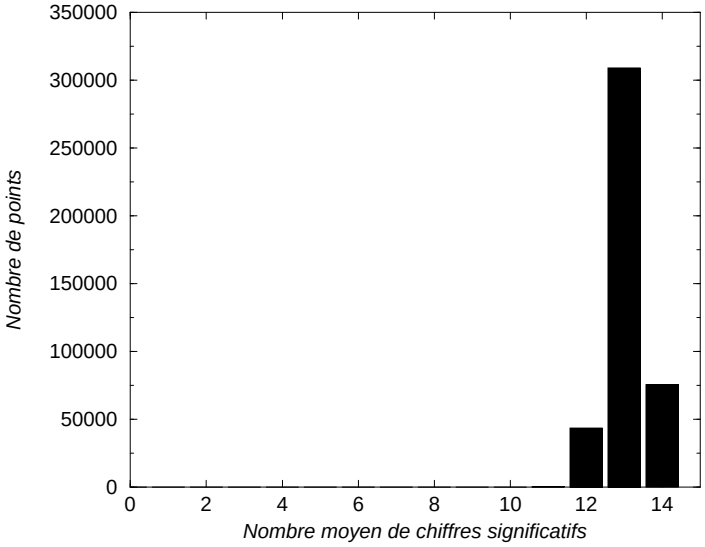

FIG. 6.11 – *Répartition du nombre de chiffres significatifs de* **sn** *à la* 30^e *itération*

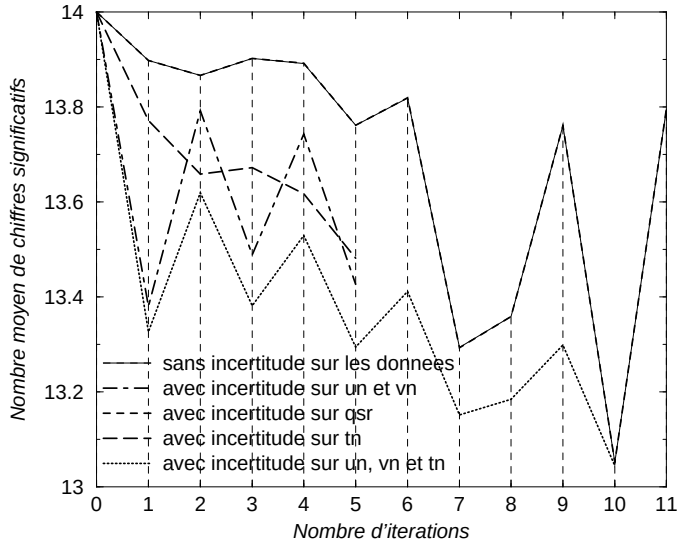

FIG. 6.12 – *Evolution de la précision de* **sn** *au cours des 11 premières itérations*

ANNEXE 6

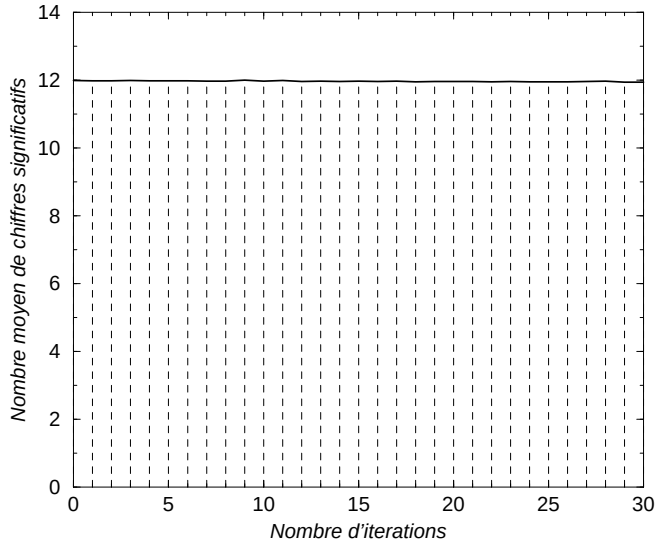

FIG. 6.13 – *Evolution de la précision de* **rdn** *au cours des 30 premières itérations*

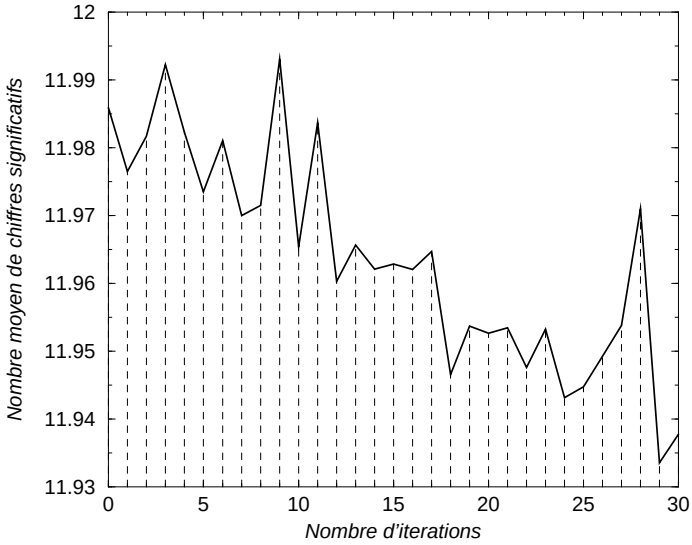

FIG. 6.14 – *Zoom sur le schéma précédent. Amplitude de 0.07 chiffre*

ANNEXE 6

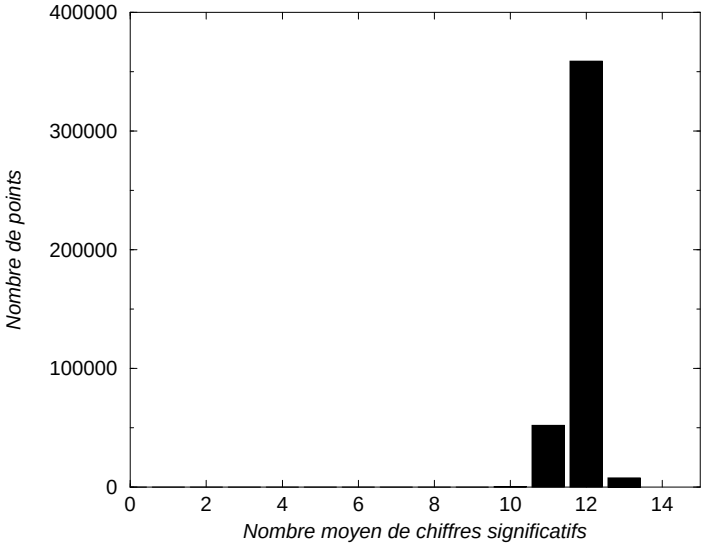

FIG. 6.15 – *Répartition du nombre de chiffres significatifs de* **rdn** *à la* 30^e *itération*

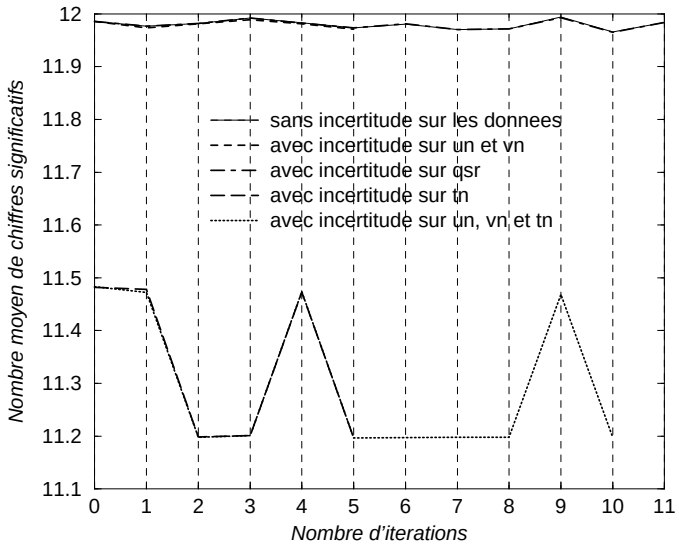

FIG. 6.16 – *Evolution de la précision de* **rdn** *au cours des 11 premières itérations*

ANNEXE 7

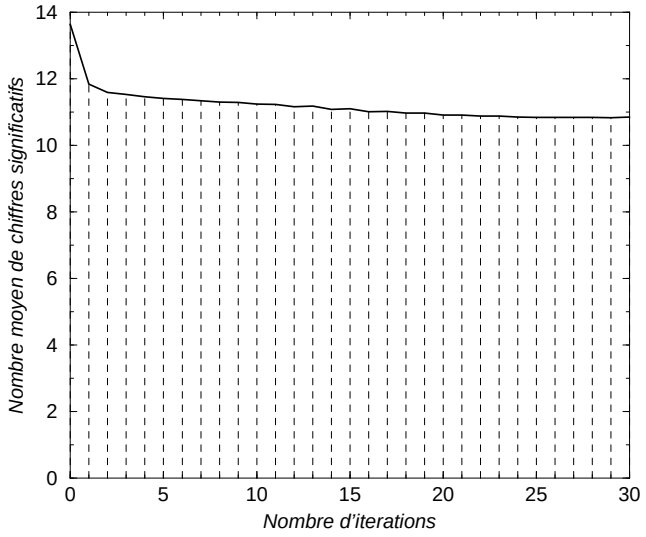

FIG. 6.17 – *Evolution de la précision de* **bn2n** *au cours des 30 premières itérations*

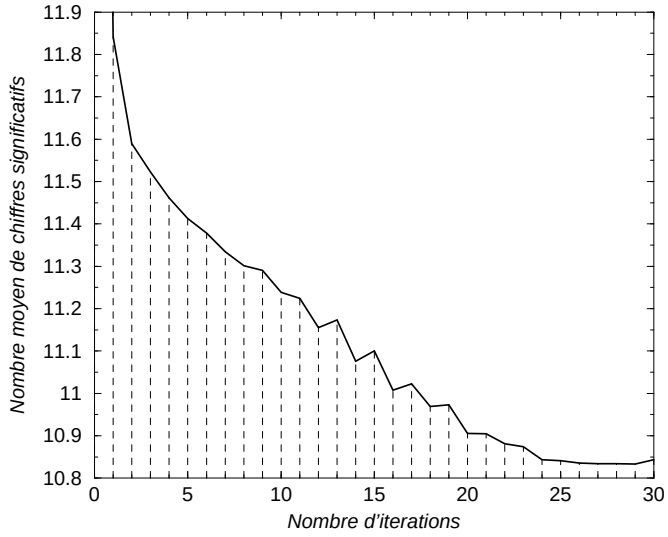

FIG. 6.18 – *Zoom sur le schéma précédent. Amplitude de 1.1 chiffre*

ANNEXE 7

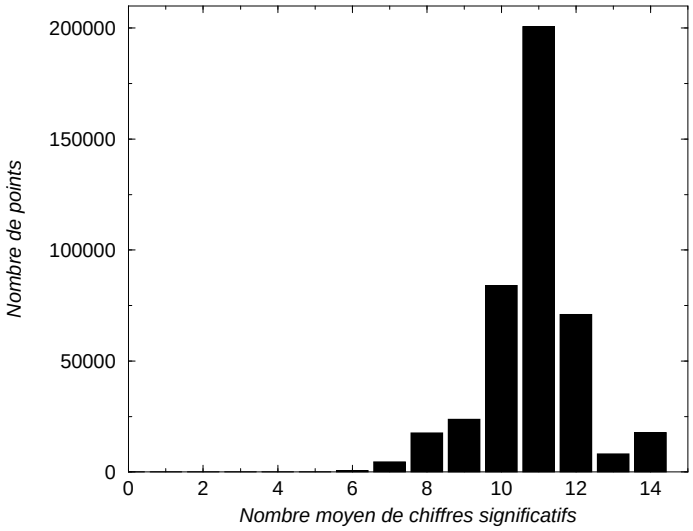

FIG. 6.19 – *Répartition du nombre de chiffres significatifs de* **bn2n** *à la* 30^e *itération*

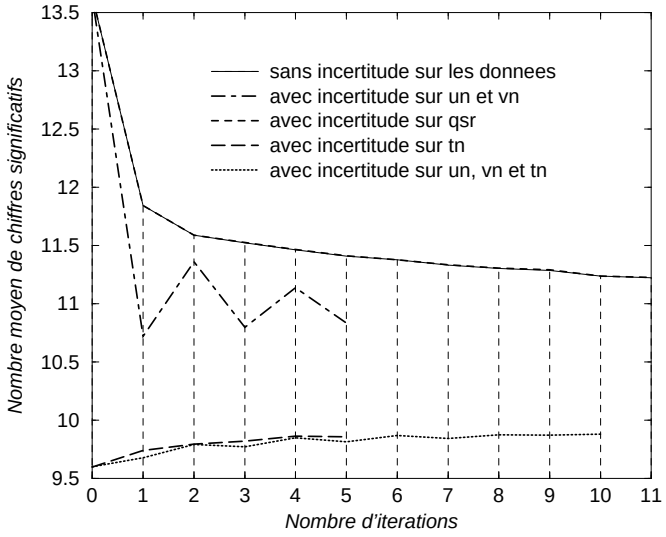

FIG. 6.20 – *Evolution de la précision de* **bn2n** *au cours des 11 premières itérations*

ANNEXE 8

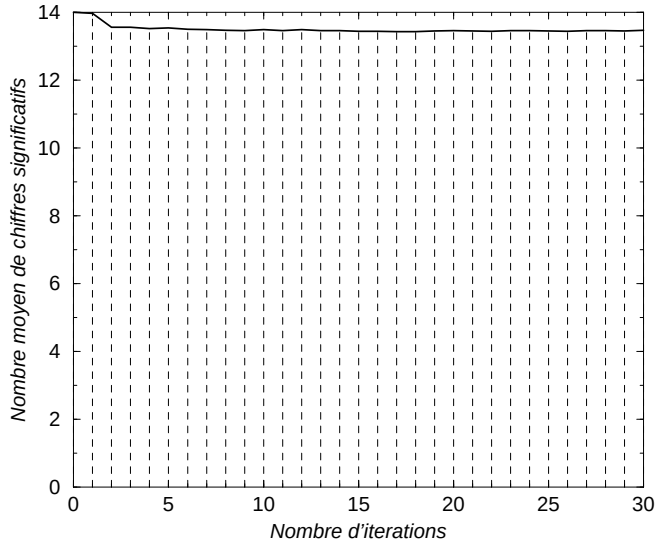

FIG. 6.21 – *Evolution de la précision de* **en** *au cours des 30 premières itérations*

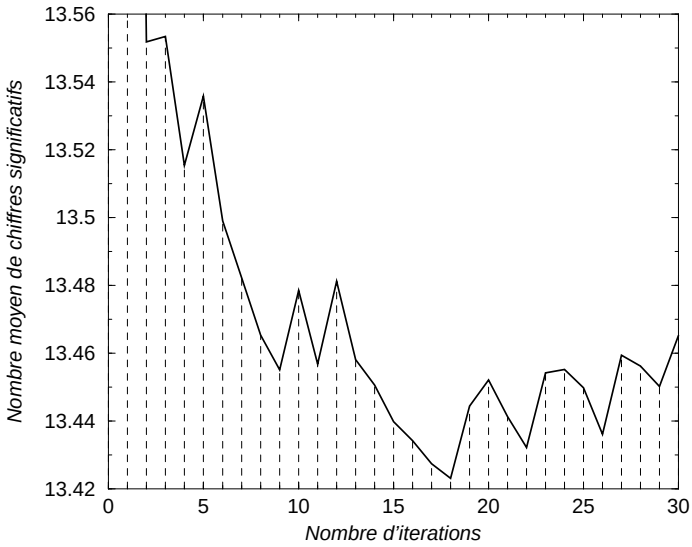

FIG. 6.22 – *Zoom sur le schéma précédent. Amplitude de 0.14 chiffre*

ANNEXE 8

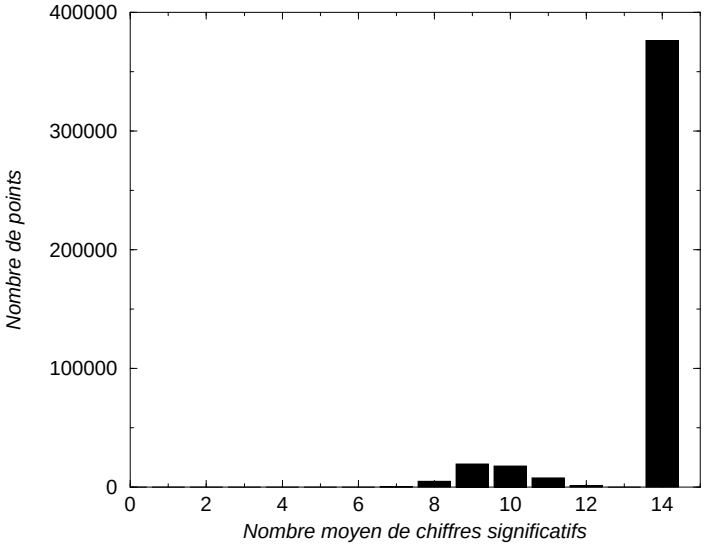

FIG. 6.23 – *Répartition du nombre de chiffres significatifs de* **en** *à la* 30^e *itération*

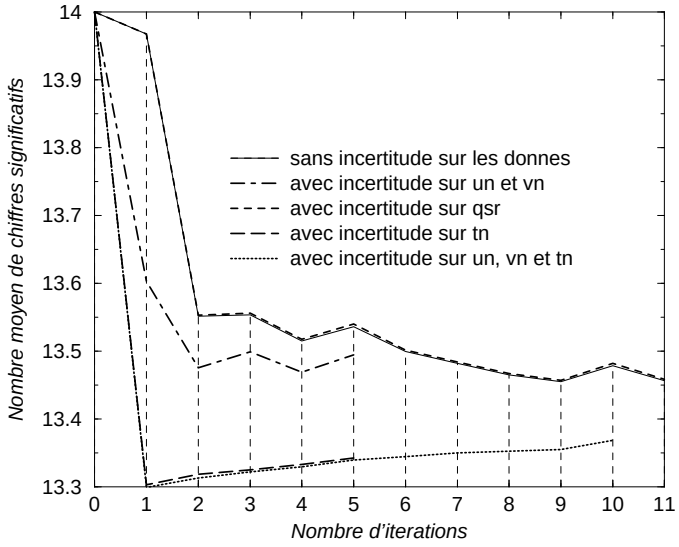

FIG. 6.24 – *Evolution de la précision de* **en** *au cours des 11 premières itérations*

ANNEXE 9

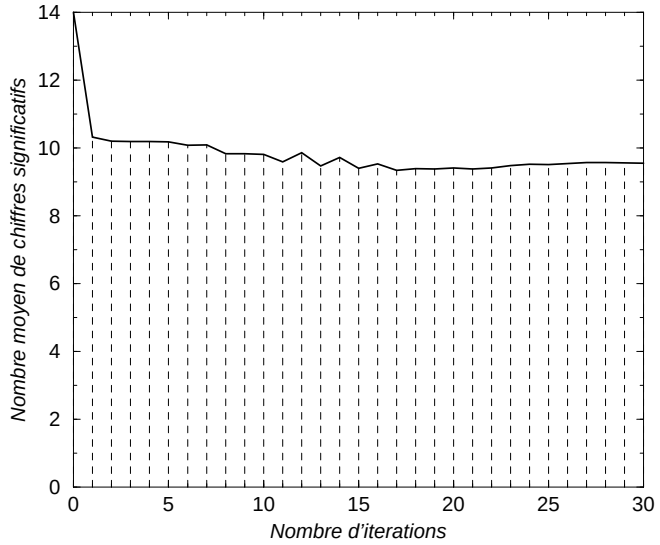

FIG. 6.25 – *Evolution de la précision de* **un** *au cours des 30 premières itérations*

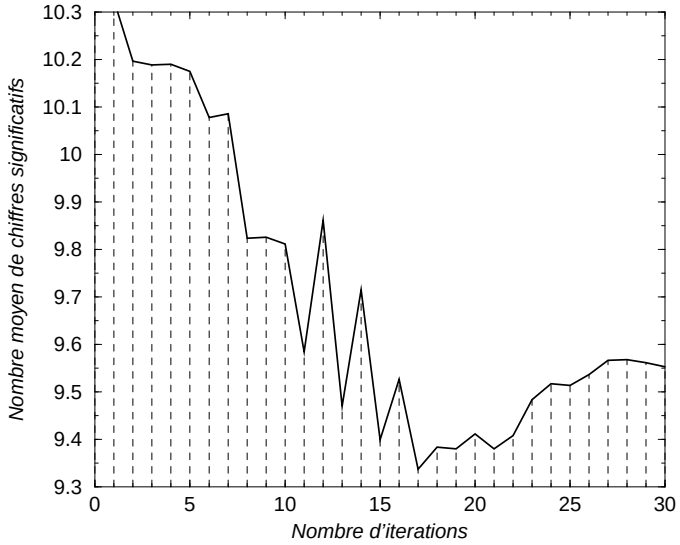

FIG. 6.26 – *Zoom sur le schéma précédent. Amplitude de 1 chiffre*

ANNEXE 9

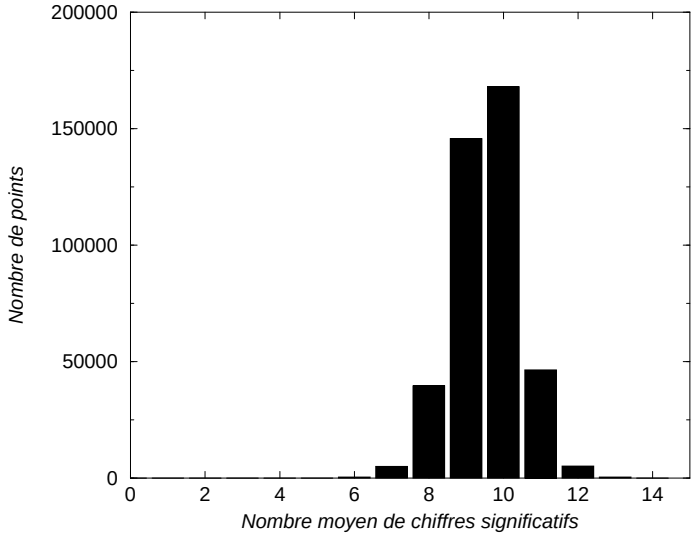

FIG. 6.27 – *Répartition du nombre de chiffres significatifs de* **un** *à la* 30^e *itération*

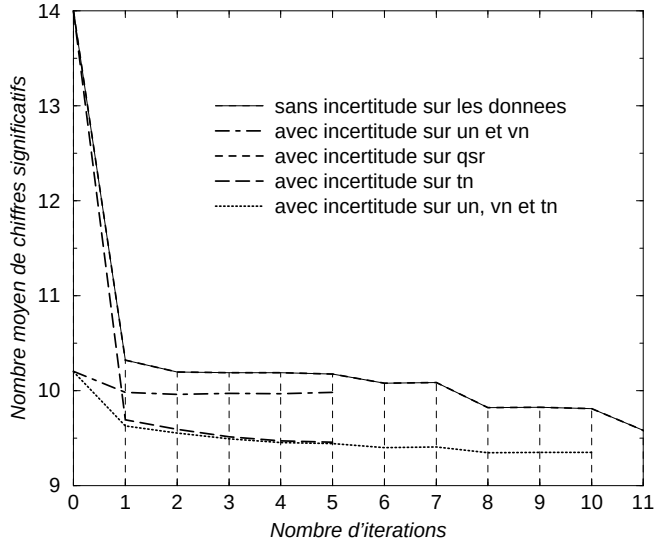

FIG. 6.28 – *Evolution de la précision de* **un** *au cours des 11 premières itérations*

ANNEXE 10

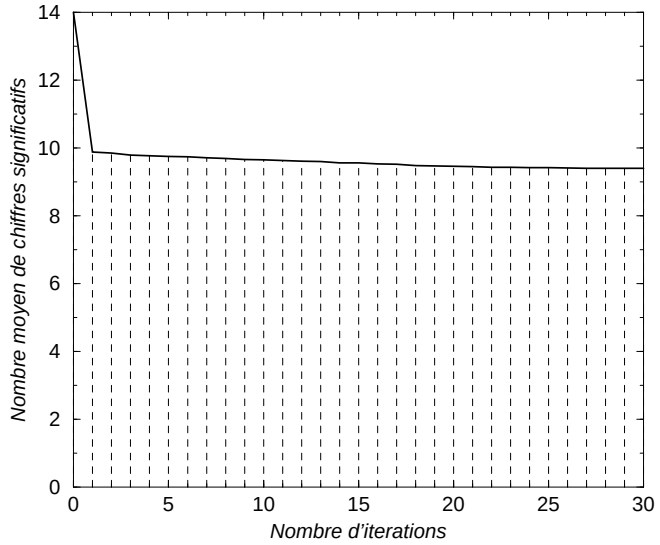

FIG. 6.29 – *Evolution de la précision de* **vn** *au cours des 30 premières itérations*

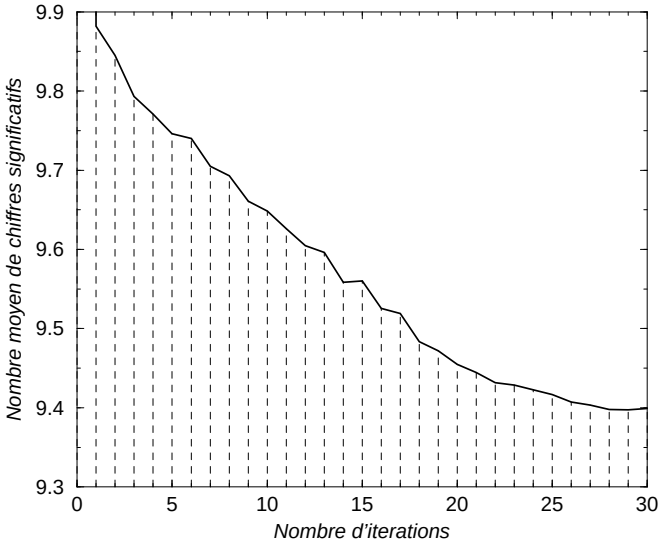

FIG. 6.30 – *Zoom sur le schéma précédent. Amplitude de 0.7 chiffre*

ANNEXE 10

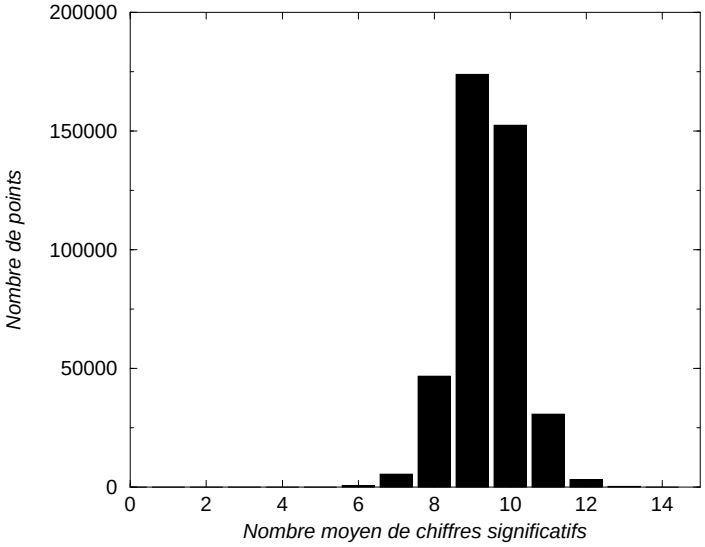

FIG. 6.31 – *Répartition du nombre de chiffres significatifs de* **vn** *à la* 30^e *itération*

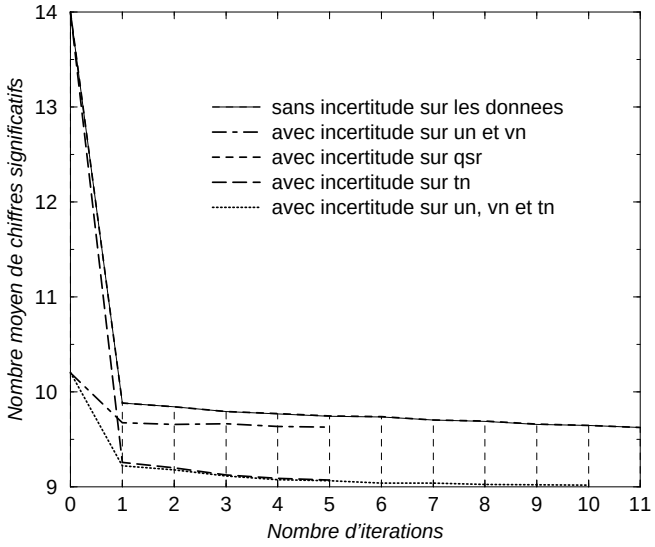

FIG. 6.32 – *Evolution de la précision de* **vn** *au cours des 11 premières itérations*

ANNEXE 11

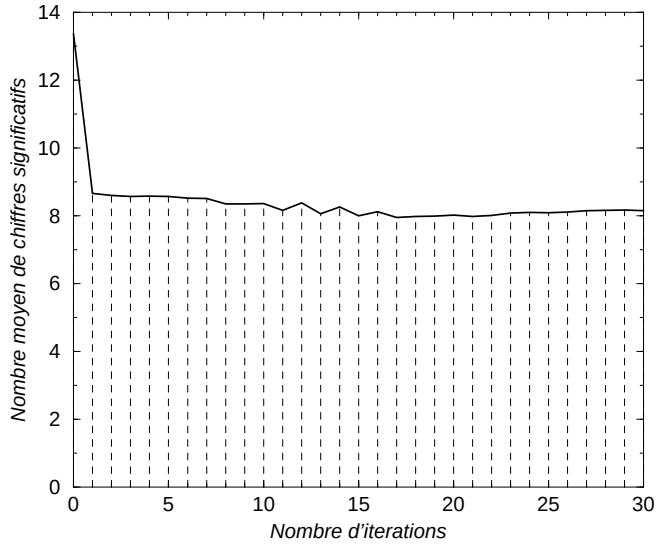

FIG. 6.33 – *Evolution de la précision de* **wn** *au cours des 30 premières itérations*

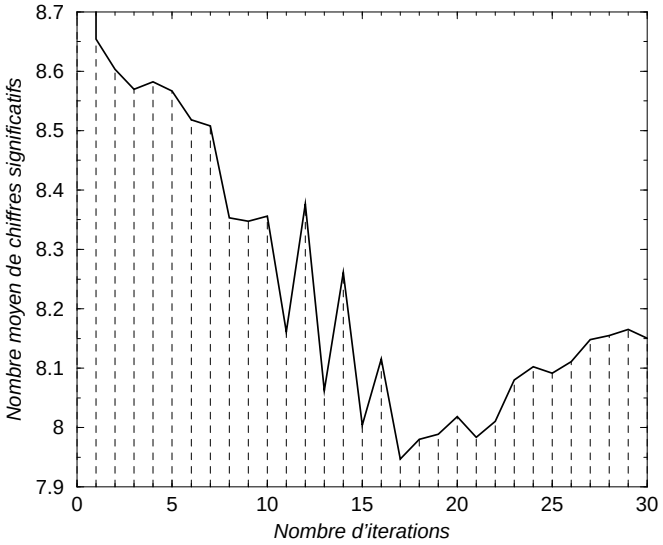

FIG. 6.34 – *Zoom sur le schéma précédent. Amplitude de 0.8 chiffre*

ANNEXE 11

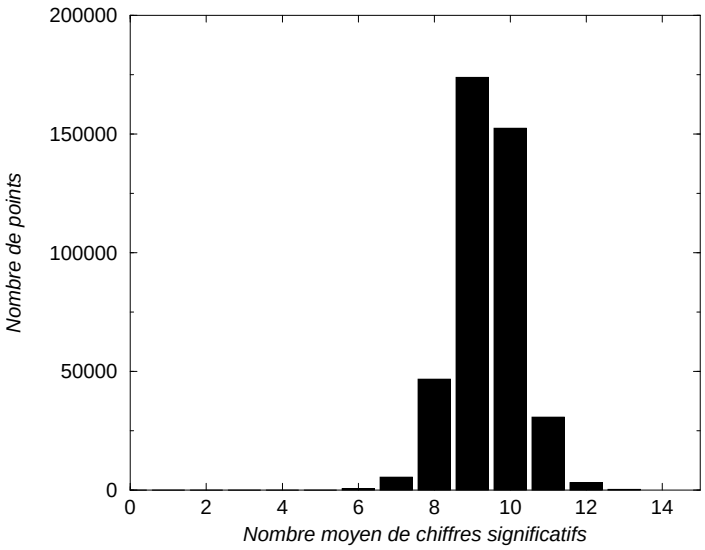

FIG. 6.35 – *Répartition du nombre de chiffres significatifs de* **wn** *à la* 30^e *itération*

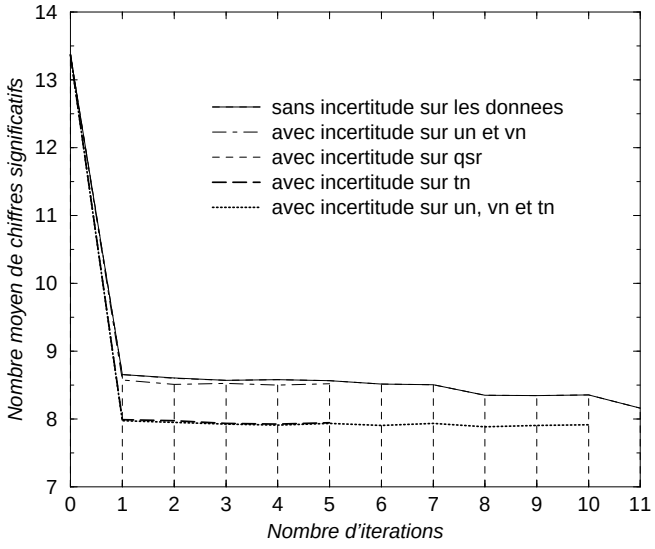

FIG. 6.36 – *Evolution de la précision de* **wn** *au cours des 11 premières itérations*

ANNEXE 12

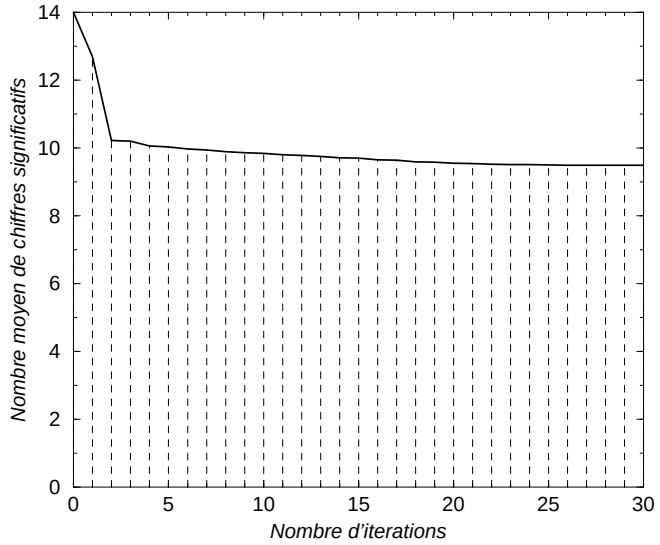

FIG. 6.37 – *Evolution de la précision de* **rotn** *au cours des 30 premières itérations*

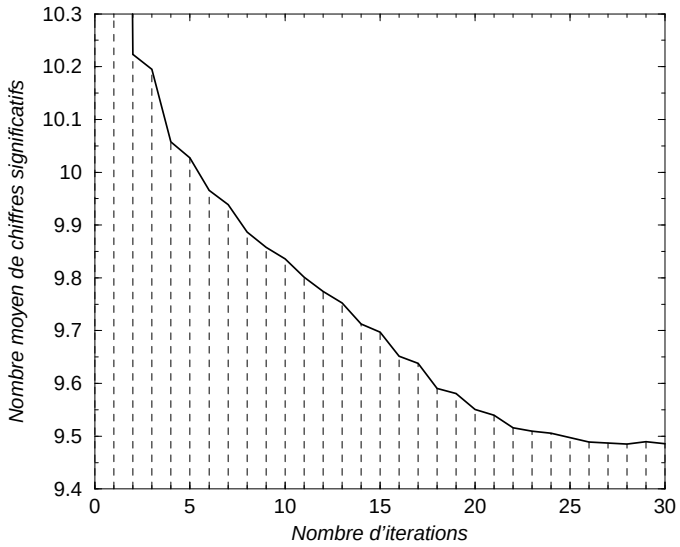

FIG. 6.38 – *Zoom sur le schéma précédent. Amplitude de 0.9 chiffre*

ANNEXE 12

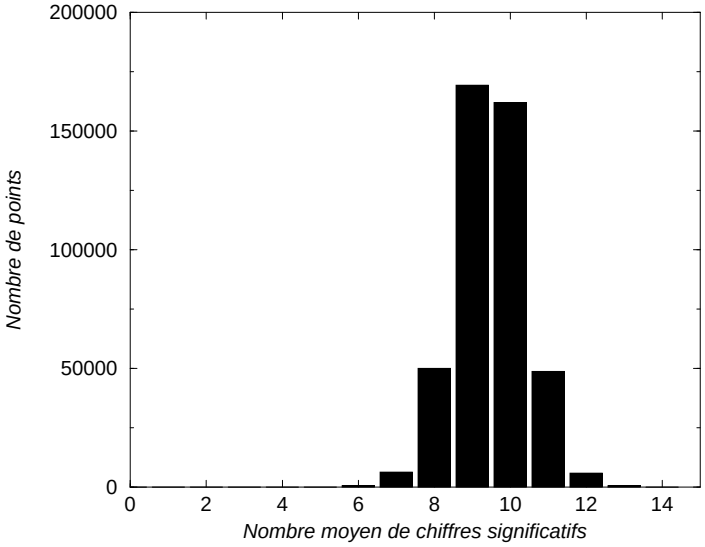

FIG. 6.39 – *Répartition du nombre de chiffres significatifs de* **rotn** *à la* 30^e *itération*

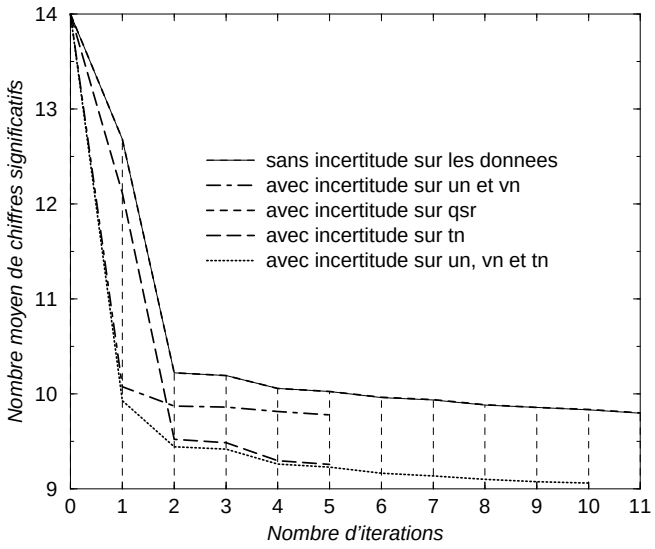

FIG. 6.40 – *Evolution de la précision de* **rotn** *au cours des 11 premières itérations*

ANNEXE 13

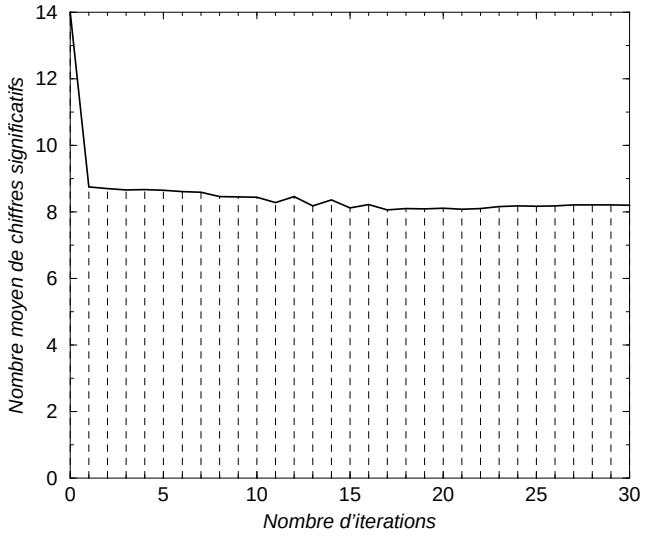

FIG. 6.41 – *Evolution de la précision de* **hdivn** *au cours des 30 premières itérations*

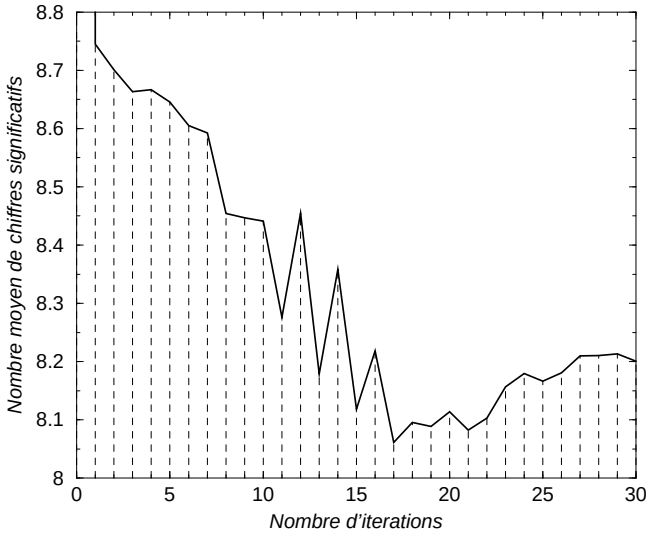

FIG. 6.42 – *Zoom sur le schéma précédent. Amplitude de 0.8 chiffre*

ANNEXE 13

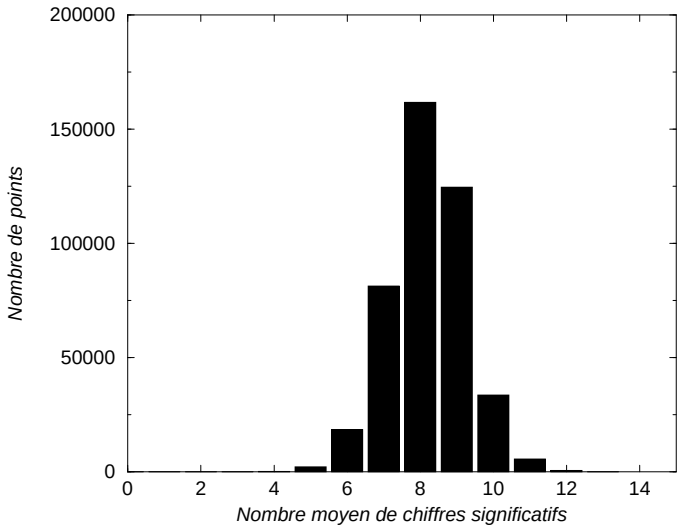

FIG. 6.43 – *Répartition du nombre de chiffres significatifs de* **hdivn** *à la* 30^e *itération*

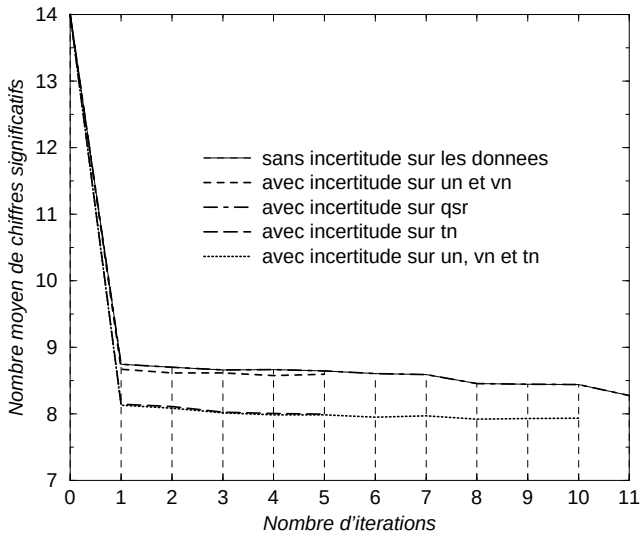

FIG. 6.44 – *Evolution de la précision de* **hdivn** *au cours des 11 premières itérations*

ANNEXE 14

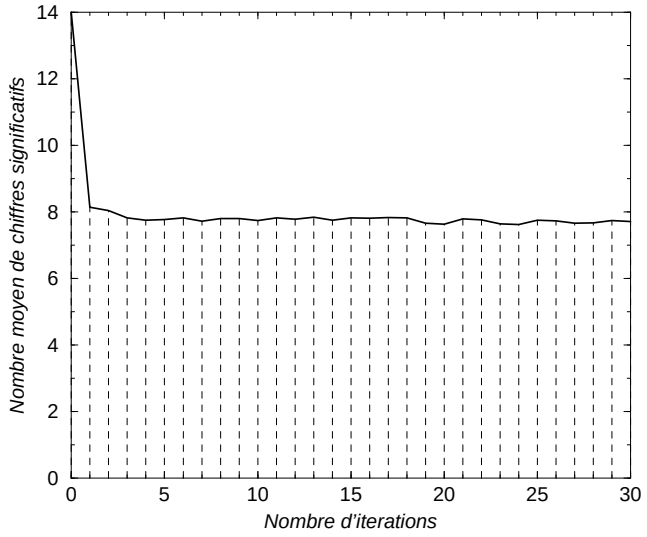

FIG. 6.45 – *Evolution de la précision de* **spgu** *au cours des 30 premières itérations*

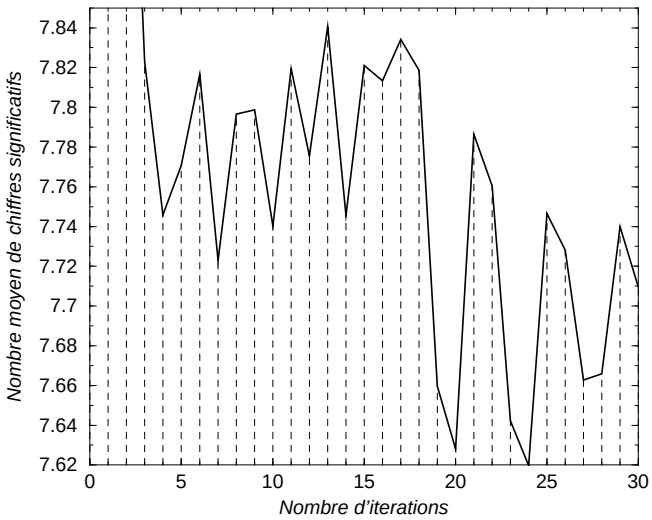

FIG. 6.46 – *Zoom sur le schéma précédent. Amplitude de 0.22 chiffre*

ANNEXE 14

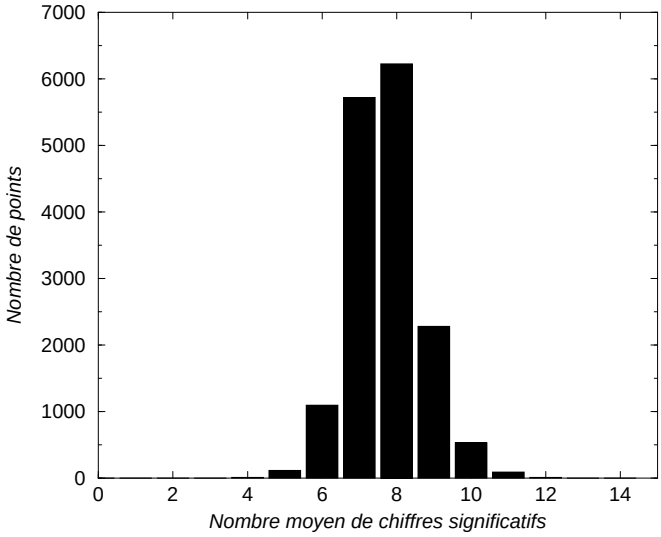

FIG. 6.47 – *Répartition du nombre de chiffres significatifs de* **spgu** *à la* 30^e *itération*

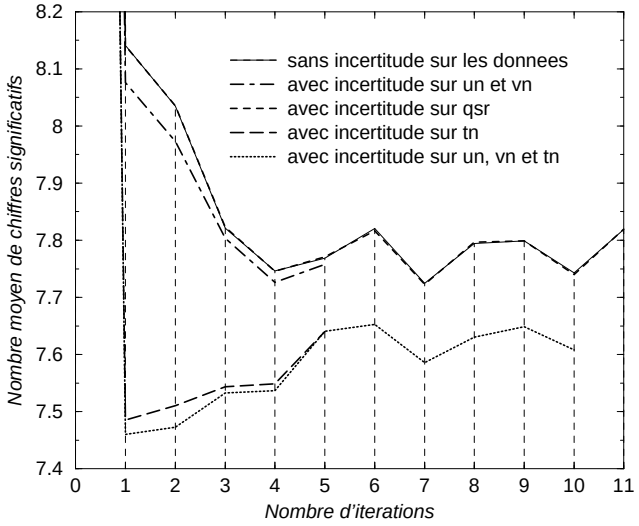

FIG. 6.48 – *Evolution de la précision de* **spgu** *au cours des 11 premières itérations*

ANNEXE 15

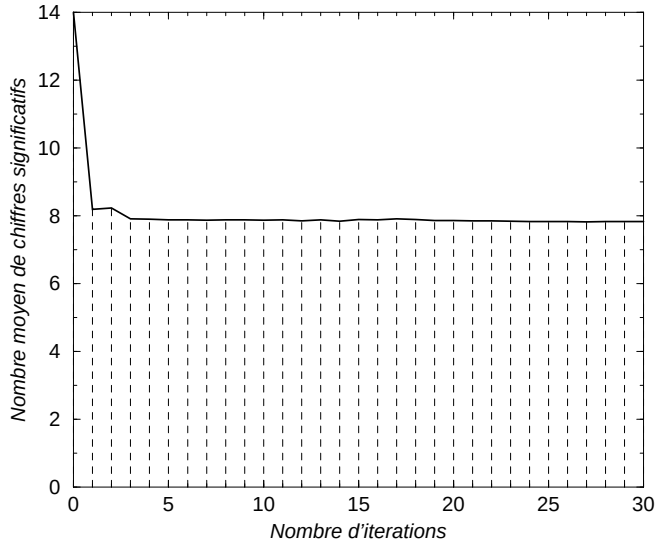

FIG. 6.49 – *Evolution de la précision de* **spgv** *au cours des 30 premières itérations*

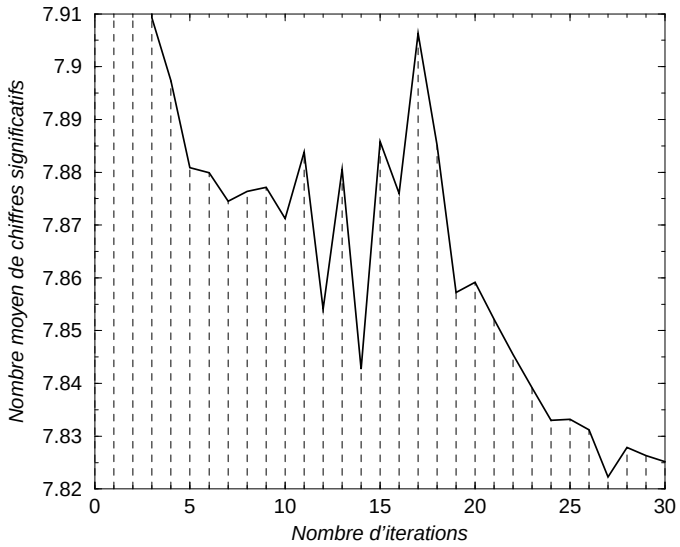

FIG. 6.50 – *Zoom sur le schéma précédent. Amplitude de 0.09 chiffre*

ANNEXE 15

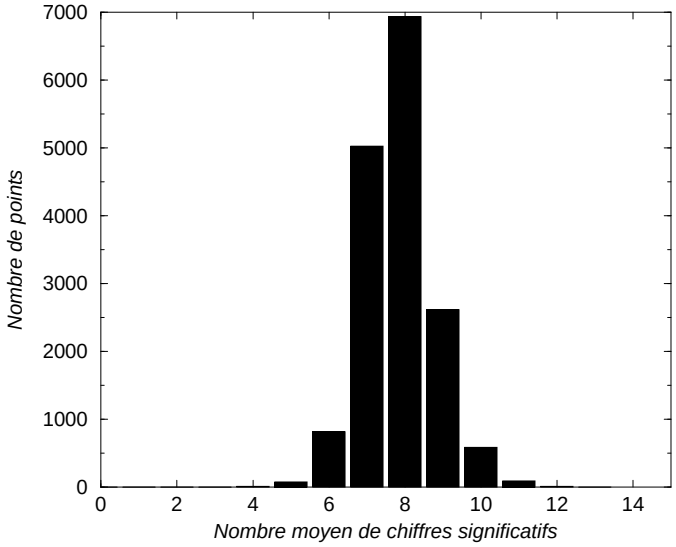

FIG. 6.51 – *Répartition du nombre de chiffres significatifs de* **spgv** *à la* 30^e *itération*

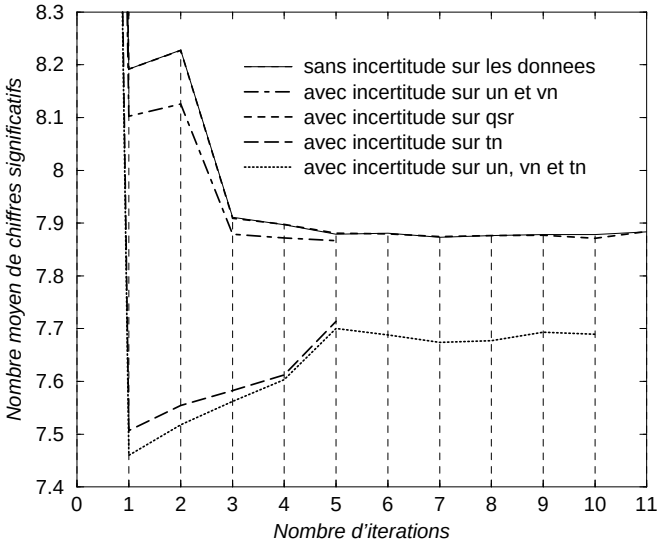

FIG. 6.52 – *Evolution de la précision de* **spgv** *au cours des 11 premières itérations*

ANNEXE 16

ANNEXE 17

ANNEXE 18

ANNEXE 19

Relation entre la précision et les valeurs de *un*

Détail de la coupe Nord-Sud à travers le Pacifique. L'axe des abscisses représente la profondeur, l'axe des ordonnées les longitudes à la surface et le troisième axe représente les valeurs de *un*.

Les couleurs claires représente une bonne précision les couleurs foncés une mauvaise précision (relativement à la précision moyenne de *un*).