Etude de la stabilité numérique du code OPA - version 8.0

pour le compte du

Pôle de modélisation
de l'Institut Pierre Simon Laplace

par

Fabienne Jézéquel, Jean-Marie Chesneaux
Laboratoire d'Informatique de Paris 6
Université Paris 6
4, place Jussieu, 75252 Paris Cedex 05
Fabienne.Jezequel@lip6.fr, Jean-Marie.Chesneaux@lip6.fr

Table des matières

Chapitre 1

Présentation de l'étude

Le code OPA 8.0 permet de simuler une circulation océanique dans un cadre très général. Ce code calcule l'évolution dans le temps, sous certaines contraintes, de paramètres tels la température, la vitesse du courant, la pression hydrostatique, la salinité. L'étude consiste à mesurer la qualité numérique des résultats fournis par le code.

1.1 Le code OPA 8.0

Le code OPA (Océan PArallélisé) [3, 4, 5, 10, 11, 12, 15] a été écrit en Fortran 90 pour calculateurs CRAY. Le code OPA comprend environ 130 fichiers ayant l'extension *.F*, *.f* ou *.h*.
L'extension *.f* caractérise les fichiers source Fortran, l'extension *.F* les fichiers avant l'appel au précompilateur cpp et l'extension *.h* les fichiers qui seront inclus dans des 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*. Le code qui nous a été fourni a notamment les options suivantes :

- monotasking (le code est séquentiel)
- utilisation de la formulation de Jackett - McDougall pour l'équation d'état
- pas d'île
- pas de glace
- solveur barotropique : gradient conjugué préconditionné
- utilisation de l'énergie cinétique turbulente et d'une méthode implicite pour la diffusion verticale
- pas de couplage.

1.2 La bibliothèque CADNA

La bibliothèque CADNA [1, 7, 2] permet une analyse de la stabilité numérique. En effet, CADNA peut fournir le nombre de chiffres significatifs exacts de tout résultat obtenu avec l'arithmétique virgule flottante des ordinateurs. CADNA permet également de détecter les instabilités numériques qui se produisent en cours d'exécution.

On appelle nombre de chiffres significatifs exacts le nombre de chiffres décimaux en commun entre le résultat informatique X fourni par l'ordinateur et le résultat mathématique exact x inconnu. Ce nombre est donné par la formule :

$$C_{x,X} = Log_{10}\Big(\frac{|X+x|}{2.|X-x|}\Big).$$

La méthode CESTAC de Jean Vignes [6, 8, 9, 13, 14] permet d'estimer ce nombre sans connaître x. La bibliothèque CADNA repose sur l'implémentation automatique de la méthode CESTAC.

La bibliothèque CADNA s'applique à des programmes écrits en Fortran 77 mais nécessite un compilateur Fortran 90 pour la génération d'un exécutable.

1.3 Les étapes de l'étude

Le code OPA qui nous a été fourni a été écrit en Fortran 90 pour calculateurs CRAY tels que la machine vectorielle CRAY J90 ou la machine parallèle CRAY T3E. Le matériel utilisé pour l'étude est une station SUN HyperSparc sous le système SUN OS 4.1.4 munie d'un compilateur Fortran 90 NAGware version 2.0.

L'étude comporte plusieurs étapes :

- portage du code en Fortran 90 sur station de travail SUN sans utilisation de la bibliothèque CADNA
- obtention d'un exécutable utilisant la bibliothèque CADNA
- analyse générale de la propagation des erreurs d'arrondi avec la détection des zones de dégradation numérique.

Chapitre 2

Portage du code en Fortran 90 sur station SUN

Le code OPA 8.0 ayant été écrit en Fortran 90 pour CRAY, la première phase de l'étude consiste à porter le code sur une station de travail SUN.
Plusieurs problèmes de portabilité sont apparus du fait des différences entre le compilateur Fortran 90 sur CRAY et le compilateur Fortran 90 NAGware utilisé sur station SUN. Ces problèmes sont dus soit aux entrées-sorties, soit à la précision utilisée.

2.1 Problèmes dus aux entrées-sorties

La procédure *parlec* lit les donnés dans le fichier *namelist*. Une option à la compilation permet sur CRAY de garder pour le fichier *namelist* la syntaxe du Fortran 77. Afin de compiler tout le code en Fortran 90 sur SUN, les instructions de lecture du fichier *namelist* ainsi que le fichier lui-même ont dus être modifiés.

Dans domwri.F l'écriture des tableaux tridimensionnels tmask, umask, vmask et fmask dans le fichier meshmask provoquait un débordement du "buffer" de sortie. Plusieurs modifications ont donc été apportées au traitement du fichier meshmask. Notamment celui-ci est déclaré comme "formatted" au lieu de "unformatted" lors de son ouverture. Les instructions d'écriture "write(nummsh)" sont remplacées par "write(nummsh,*)"

Pour faciliter la lecture du fichier restart.output, celui-ci est également déclaré comme "formatted" au lieu de "unformatted" lors de son ouverture.

Dans *diawri.F*, les fichiers du type ex0O..VO960100 et ex0O..SO960100 (leur extension dépend de l'itération en cours lors de la sauvegarde) sont ouverts grâce à la commande "assign -F f77 -N ieee $nom\,de\,fichier$". Cette commande ne pouvant être exécutée sur SUN, l'ouverture de ces fichiers a due être modifiée.

2.2 Problèmes dus à la précision utilisée

Le code Fortran qui nous a été fourni utilise des variables de type REAL, codées sur CRAY sur 64 bits. En Fortran sur SUN, pour que les variables soient codées sur 64 bits, elles doivent être déclarées en double précision. Le code OPA a donc été modifié de manière à être exécuté en double précision sur SUN.

Cependant la longueur de la mantisse peut varier selon la machine utilisée. Sur CRAY C90, les réels codés sur 64 bits ont une mantisse de 48 bits et ont donc 13 chiffres significatifs. Sur les machines respectant la norme IEEE 754 telles que les stations SUN, les calculateurs CRAY T3D

et T3E, les réels codés sur 64 bits ont une mantisse de 53 bits (52 bits variables et un bit fixé à 1) et ont donc 15 chiffres significatifs. Les travaux sur la qualité numérique des résultats du code OPA effectués sur SUN peuvent être étendus à d'autres calculateurs. En effet, le nombre de chiffres significatifs perdu au cours d'un calcul ne dépend pas de la précision utilisée.

Outre le remplacement de "REAL" par "DOUBLE PRECISION" dans la totalité du code, les modifications suivantes ont été effectuées.

Les fonctions amax1 et amin1, acceptant des arguments de type REAL, ont été remplacées par les fonctions génériques max et min.

Tous les paramètres réels ont été changés afin que ceux-ci aient la précision souhaitée. Par exemple, le paramètre 0.2 doit être transformé dans un calcul en double précision en 0.2D0.

La fonction *cvmgp* est appelée dans les fichiers *dommsk.F* et *zdfmix.tke.h*. Cette fonction était déclarée ainsi :

```
      FUNCTION CVMGP(A,B,C)
      REAL A,B,C
      IF(C.GE.0.)THEN
      CVMGP=A
      ELSE
      CVMGP=B
      ENDIF
      RETURN
      END
```

Le type de la fonction est alors implicitement REAL. La fonction *cvmgp* a été modifiée pour qu'elle soit définie double précision :

```
      FUNCTION CVMGP(A,B,C)
      IMPLICIT NONE
      DOUBLE PRECISION A,B,C,CVMGP
      IF(C.GE.0.D0)THEN
      CVMGP=A
      ELSE
      CVMGP=B
      ENDIF
      RETURN
      END
```

La même opération a due être effectuée pour la fonction *sdot*. Celle-ci est utilisée dans les fichiers *bsfpcg.F* et *dynspg.F*. La fonction initialement définie ainsi :

```
      FUNCTION SDOT(I,X,J,Y,K)
      REAL X(1),Y(1)
      SDOT=0.
      DO 1 N=1,I
      SDOT=SDOT+X(1+(N-1)*J)*Y(1+(N-1)*K)
1     CONTINUE
      RETURN
      END
```

a été modifiée, afin d'être définie en double précision :

```
      FUNCTION SDOT(I,X,J,Y,K)
      IMPLICIT NONE
      DOUBLE PRECISION X(1),Y(1),SDOT
      SDOT=0.D0
      DO 1 N=1,I
      SDOT=SDOT+X(1+(N-1)*J)*Y(1+(N-1)*K)
1     CONTINUE
      RETURN
      END
```

L'exécution du code en simple précision sur SUN n'était en fait pas possible du fait de l'ordre de grandeur de certaines variables. Par exemple, dans le fichier zdfmix.tke.h, examinons les instructions suivantes :

```
      z0cray=1.D-53

      DO jk=2,jpkm1
        DO ji=1,jpi
          emxl(ji,jj,jk) = sqrt( 2.D0*en(ji,jj,jk)
     $                /max(bn2n(ji,jj,jk),z0cray) )
        END DO
      END DO
```

En simple précision, les variables de l'ordre de 10^{-53} sont considérées comme nulles. Lorsque bn2n(ji,jj,jk) est nul, il se produit une erreur à l'exécution due à la division par zéro.

Chapitre 3

Code avec CADNA

3.1 Modifications du code pour l'utilisation de CADNA

Le code OPA a été modifié afin de permettre l'utilisation de la bibliothèque CADNA. Certaines modifications sont nécessaires quel que soit le code étudié ; d'autres sont spécifique au code OPA.

3.1.1 Modifications classiques

Le principe de la bibliothèque CADNA repose sur l'existence de nouveaux types numériques définis dans la biliothèque : les types stochastiques.
Ces types sont :

- TYPE (SINGLE_ST) : stochastique simple précision
- TYPE (DOUBLE_ST) : stochastique double précision
- TYPE (COMPLEX_ST) : complexe stochastique simple précision
- TYPE (DOUBLE_COMPLEX_ST) : complexe stochastique double précision

Les déclarations de variables stochastiques s'effectuent comme pour les types numériques classiques. Par exemple, TYPE (DOUBLE_ST) X,Y,Z . Le contrôle de la précision est effectué uniquement sur les types stochastiques. Bien sûr, la bibliothèque contient les définitions de tous les opérateurs et fonctions classiques du Fortran 77 pour les types stochastiques.

La bibliothèque CADNA fonctionnant comme un module, il est nécessaire d'ajouter l'instruction USE CADNA en tête de chaque unité de compilation.

De plus, CADNA a besoin d'une fonction d'initialisation : CADNA_INIT(k). Elle doit être appelée une fois et une fois seulement. Ce doit être la première instruction du programme principal. Dans l'instruction CALL CADNA_INIT(1000), le nombre 1000 signifie que l'on demande la détection des mille premières instabilités uniquement. Pour une détection non limitée, il faut mettre le paramètre -1, pour déconnecter les détections, le paramètre 0.

Une autre modification imposée par CADNA concerne les entrées-sorties.
En lecture, l'initialisation de variables stochastiques par des constantes doit se faire par l'intermédiaire de variables numériques classiques.

```
DOUBLE PRECISION X
READ(*,*)X
```

devient :

```
TYPE (DOUBLE_ST) X
```

```
DOUBLE PRECISION AUX
READ(*,*) AUX
X=AUX
```

Pour les sorties, les impressions se font à l'aide de la fonction STR de CADNA qui transforme une variable de type stochastique en chaîne de caractères. Il est donc aussi nécessaire de modifier les instructions FORMAT en conséquence. Avec la fonction STR, seuls les chiffres significatifs exacts sont affichés, ce qui permet une visualisation immédiate de la précision des résultats.

3.1.2 Modifications spécifiques au code OPA

Suppression du fichier namelist

Afin de simplifier les initialisations de variables stochasiques effectuées dans le fichier *parlec.F*, toutes les lectures dans la namelist ont été supprimées. La variables sont directement initialisées dans la procédure *parlec*.

Suppression de fonctions on-line

La procédure *domhgr* utilise des fonctions "on-line", telle que la fonction *fslam* définie comme suit :

```
DOUBLE PRECISION fslam
fslam(pi,pj) = zstlam + zdx * ( dble(pi)-1.D0 + dble(nimpp-1) )
```

La fonction "on-line" *fsdjla* retourne zéro :

```
DOUBLE PRECISION  fsdjla
fsdjla=0.D0
```

Lorsque l'on déclare la fonction *fsdjla* de type (DOUBLE_ST), on ne peut écrire :

```
TYPE (DOUBLE_ST) fsdjla
fsdjla(pi,pj) = 0.D0
```

puisque la constante 0.D0 n'est pas de type stochastique. L'utilisation d'une variable temporaire nécessite une affectation avant la définition de la fonction et n'est donc pas possible. Par conséquent, la fonction *fsdjla* n'est plus définie comme une fonction "on-line" dans le code avec CADNA. Celle-ci est définie comme suit :

```
FUNCTION fsdjla(pi,pj)
use cadna
implicit none
TYPE (DOUBLE_ST) pi,pj,fsdjla
fsdjla=0.D0
END
```

La même opération a dû être effectuée avec la fonction "on-line" *fsdiph* qui retourne également zéro.

Modification de l'appel aux fonction min et max

Dans le code OPA, les fonctions min et max sont parfois appelées avec plus de deux arguments. Or dans un code avec CADNA, ces fonctions doivent avoir exactement deux arguments. Les appels aux fonctions min et max ont donc été modifiés en conséquence.

Par exemple, l'instruction suivante dans le fichier *dommsk.F* :

```
            fmask(ji,jj,jk) =shlat * min(1.D0,
     $                  max( zwf(ji+1,jj),zwf(ji,jj+1),
     $                  zwf(ji-1,jj), zwf(ji,jj-1) ) )
```

devient :

```
            fmask(ji,jj,jk) =shlat * min(1.D0,
     $                  max(zwf(ji+1,jj), max(zwf(ji,jj+1),
     $                  max(zwf(ji-1,jj), zwf(ji,jj-1) ) ) ) )
```

3.1.3 Exécution du code avec CADNA

Les premières exécutions du code ont été effectuées avec un faible nombre d'itérations en temps. Les résultats présentés au chapitre 4 ont été obtenus après 1500 itérations en temps. Le pas de temps étant une heure, 1500 itérations correspondent à une simulation de la circulation océanique pendant plus de deux mois.

Le temps d'exécution du code est tout-à-fait raisonnable jusqu'à 10 itérations,
sans utilisation du debugger :
2 min sans CADNA
10 min avec CADNA et écriture de toutes les instabilités dans le fichier de traces.

Le temps d'exécution du code jusqu'à 1500 itérations ne permet pas d'effectuer facilement tous les tests souhaités,
sans utilisation du debugger :
3h30 sans CADNA
16h avec CADNA
en utilisant le debugger "dbx" :
19h avec CADNA.

Les tests effectués montrent que l'utilisation de CADNA augmente le temps d'exécution d'OPA d'un facteur 5.

Chapitre 4

Résultats

4.1 Analyse des traces

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 numérique, un message peut être écrit dans le fichier des traces *cadna_stability_f90.lst*.

On distingue cinq types d'instabilités numériques :

- les multiplications instables ; ce sont les multiplications pour lesquelles les deux opérandes sont non significatives.
- les divisions instables ; une division instable est causée par un diviseur non significatif.
- les tests instables ; ce sont les tests pour lesquels la différence entre les deux opérandes est non significative. Le débranchement pris correspond alors à celui de l'égalité.
- les instabilités survenues lors de l'appel à une fonction intrinsèque (int, aint, abs, mod, sign, dim)
- les instabilités survenues lors de l'appel à une fonction mathématique (sqrt, exp, log, log10, sin, cos, tan, asin, acos, atan, atan2, sinh, cosh, tanh)

Dans le code OPA, ont été détectées des instabilités numériques causées par des multiplications, des racines carrées, des divisions, et des tests.

4.1.1 multiplications instables

Le nombre de multiplications instables survenues à chaque itération a été déterminé. Les multiplications instables surprennent par leur nombre et leur irrégularité. Au cours des 185 premières itérations en temps, surviennent plus de 500000 multiplications instables qui génèrent un fichier de traces d'environ 32 Moctets.
A la seconde itération, surviennent 6809 multiplications instables ; puis jusqu'à l'itération 73, au maximum 10 multiplications instables sont détectées. A partir de l'itération 90, plusieurs milliers de multiplications instables sont générées à chaque itération.
La figure 7.1 en annexe présente l'évolution du nombre de multiplications instables jusqu'à l'itération 185.

Ces multiplications instables sont générées par six instructions :

1. Dans le fichier *zdfmix.tke.h* :

```
zsh2 = zdudz*zdudz + zdvdz*zdvdz
```

A la seconde itération, cette instruction génère 6142 multiplications instables sur les 6809 multiplications instables détectées à cette itération.

Cette instruction est exécutée dans quatre boucles imbriquées :

```
ds step.F      DO jstp = nit000,nitend (1 a 1500)
ds zdfmix.tke.h   DO 2000 jk=ktask+1,jpkm1,kcpu (2 a 19)
                     DO jj=2,jpjm1  (2 a 39)
                        DO ji=2,jpim1 (2 a 39)
                           zsh2 = zdudz*zdudz + zdvdz*zdvdz
                        ENDDO
                     ENDDO
            2000 continue
               ENDDO
```

Les instructions suivantes utilisant $zsh2$ sont :

```
esh2(ji,jj,jk) = zsh2
zri  = max(bn2n(ji,jj,jk),0.D0)/(zsh2+1.D-20)
```

L'addition avec 10^{-20} dans le calcul de zri permet d'éviter une éventuelle division par zéro.

Durant les premières itérations en temps, plus précisément pendant 757 itérations, lorsque la variable $zsh2$ est non significative, sa valeur est strictement inférieure à 10^{-20}. Le diviseur $zsh2 + 10^{-20}$ reste alors significatif.

A certaines itérations, la variable $zsh2$ peut cependant être non significative et supérieure à 10^{-20} (au maximum de l'ordre de 10^{-6}). Il serait donc judicieux de remplacer le diviseur $zsh2 + 10^{-20}$ par $zsh2 + 10^{-5}$, si cela n'influe pas sur la signification physique de zri.

2. Dans le fichier *dynzad.F* :

```
zwuw(ji,jk)= zun * ( un(ji,jj,jk-1)-un(ji,jj,jk) )
```

A la seconde itération, cette instruction génère 667 multiplications instables sur les 6809 multiplications instables détectées à cette itération.

Cette instruction est exécutée dans deux boucles imbriquées :

```
DO jk=2,jpkm1
   DO ji=2,jpim1
      zun = 0.5D0*( e1t(ji+1,jj)*e2t(ji+1,jj)*wn(ji+1,jj,jk)+zw0
      zwuw(ji,jk)= zun * ( un(ji,jj,jk-1)-un(ji,jj,jk) )
   END DO
END DO
```

Le tableau $zwuw$ est ensuite utilisé pour le calcul de la variable zua.

3. Dans le fichier *bsfpcg.F* :

```
rnorme=sdot(jpi*jpj,gcr,1,zgwgt,1)
```

Cette instruction correspond à deux lignes identiques du fichier *bsfpcg.F* qui génèrent toutes deux des multiplications instables.

rnorme est le produit scalaire des tableaux *gcr* et *zgwgt*. Cette instruction est équivalente à :

```
rnorme=0.D0
DO N=1,jpi*jpj
   rnorme=rnorme+gcr(N)*zgwgt(N)
ENDDO
```

Lors du premier calcul de *rnorme*, sa valeur est 40 fois non significative, son ordre de grandeur varie entre 10^{-11} et 10^{-8}.

Lors du second calcul de *rnorme*, sa valeur est 344 fois non significative, son ordre de grandeur varie toujours entre 10^{-11} et 10^{-8}.

4. Dans le fichier *bsfpcg.F*:

```
radd=sdot(jpi*jpj,gcdes,1,zgwgt,1)
```

radd est le produit scalaire des tableaux *gcdes* et *zgwgt*. Cette instruction est équivalente à :

```
radd=0.D0
DO N=1,jpi*jpj
   radd=radd+gcdes(N)*zgwgt(N)
ENDDO
```

5. Dans le fichier *dynspg.F*:

```
rnorme=sdot(jpi*jpj,gcb,1,zgwgt,1)
```

rnorme est le produit scalaire des tableaux *gcb* et *zgwgt*. Cette instruction est similaire au calcul du produit scalaire des tableaux *gcr* et *zgwgt* dans le fichier *bsfpcg.F*.

6. Dans le fichier *dynber.F*:

```
        DO jj=2,jpj
          DO ji=2,jpi
            zwh(ji,jj) = zrau0r*hprn(ji,jj,jk)
     $          +0.25*( un(ji-1,jj  ,jk)*un(ji-1,jj  ,jk)
     $          + un(ji  ,jj  ,jk)*un(ji  ,jj  ,jk)
     $          + vn(ji  ,jj-1,jk)*vn(ji  ,jj-1,jk)
     $          + vn(ji  ,jj  ,jk)*vn(ji  ,jj  ,jk)  )
          END DO
        END DO
```

Le tableau *zwh* est ensuite utilisé pour le calcul des variables *zua* et *zva*.

4.1.2 racines carrées instables

Les racines carrées instables correspondent à un appel à la fonction *sqrt* avec un argument non significatif. Lors des 1500 itérations en temps, surviennent 6679 racines carrées instables.

Les racines carrées instables sont générées par deux instructions :

1. Dans le fichier *bsfpcg.F*:

```
res=sqrt(rnorme)
```

où *rnorme* est le second produit scalaire des tableaux *gcr* et *zgwgt*. La valeur *rnorme* est alors 344 fois non significative.

2. Dans le fichier *zdfmix.tke.h*:

```
    DO 1000 jj=ktask,jpj,kcpu (1 a 40)
      DO jk=2,jpkm1 (2 a 20)
         DO ji=1,jpi (1 a 40)
            emxl(ji,jj,jk) = sqrt(2.D0*en(ji,jj,jk)/max(bn2n(ji,jj,jk),z0cray))
         END DO
      END DO
1000 CONTINUE
```

où $z0cray$ vaut 10^{-53}.

La valeur $emxl(ji, jj, jk)$ a cependant toujours au moins un chiffre significatif exact.

Le nombre de racines carrées instables survenues toutes les 100 itérations a été déterminé. La figure 7.2 en annexe présente l'évolution du nombre de racines carrées instables jusqu'à 1500 itérations.

4.1.3 divisions instables

Lors des 1500 premières itérations en temps, surviennent 8380 divisions instables. Ces divisions instables sont dues aux deux lignes suivantes du fichier *bsfpcg.F*:

```
alph=rr/radd
beta=rnorme/rr
```

Dans ces deux lignes de code, la variable rr a la même valeur : le produit scalaire des tableaux gcr et $zgwgt$.

Le nombre de divisions instables survenues toutes les 100 itérations a été déterminé. La figure 7.3 en annexe présente l'évolution du nombre de divisions instables au cours des 1500 premières itérations en temps.

4.1.4 tests instables

Lors des 1500 premières itérations en temps, surviennent 3994 tests instables. Ces instabilités sont dues au test **rnorme.lt.epsr**, test de convergence pour la méthode du gradient conjugué préconditionné utilisée dans la procédure $bsfpcg$.
Les lignes de code correspondantes dans le fichier *bsfpcg.F* sont :

```
IF(rnorme.lt.epsr.or.jn.eq.nmax) THEN
   res=sqrt(rnorme)
   niter=jn
   ncut=999
ENDIF
```

- $rnorme$, calculé dans la procédure *bsfpcg*, est le second produit scalaire des tableaux gcr et $zgwgt$. La valeur $rnorme$ est alors 344 fois non significative.

- $epsr$ est calculé dans la procédure *dynspg*: $epsr = eps * eps * rnorme$, où $eps = 10^{-2}$ et $rnorme$ est alors le produit scalaire des tableaux gcb et $zgwgt$.

Le nombre de tests instables survenus toutes les 100 itérations a été déterminé. La figure 7.4 en annexe présente l'évolution du nombre de tests instables au cours des 1500 premières itérations en temps.

Remarque :

Afin de réduire le nombre d'instabilités numériques, nous avons remplacé dans le test d'arrêt de la méthode du gradient conjugué l'inégalité stricte par une inégalité large.
Le test devient alors : IF(rnorme.**le**.epsr.or.jn.eq.nmax) .
Avec ce nouveau test, sont survenus au cours des 1500 premières itérations seulement 360 divisions instables, 227 tests instables et 1361 racines carrées instables.

4.2 Evolution de la qualité numérique des variables

Grâce à la bibliothèque CADNA, le nombre de chiffres significatifs exacts des variables peut être déterminé à tout moment durant l'exécution du code.
La précision de divers tableaux a été contrôlée au cours des 1500 premières itérations en temps. Pour chaque tableau considéré, toutes les 100 itérations, le nombre de chiffres significatifs exacts de ses éléments a été enregistré. Plus précisément, pour toutes les itérations multiples de 100, ont été déterminées :

- la répartition des éléments du tableau selon leur nombre de chiffres significatifs exacts. Ce nombre, pour des variables caculées en double précision, est un entier compris entre 0 et 15.

- la moyenne du nombre de chiffres significatifs exacts des éléments du tableau

Dans le code qui nous a été fourni, les conditions imposées au modèle, sont relativement simples. Pour divers tableaux, certains éléments, ne sont pas modifiés lors de l'exécution du code et gardent la précision maximale (15 chiffres significatifs exacts). Pour ces tableaux, la moyenne du nombre de chiffres significatifs exacts des éléments a donc été calculée sans tenir compte des éléments qui ne sont pas recalculés après leur initialisation.

Les tableaux traités sont bidimensionnels ou tridimensionnels. La plage de variation de leurs indices est alors (40,40) ou (40,40,20).

Dans ce qui suit, nous présentons l'évolution dans le temps du nombre de chiffres significatifs exacts de divers tableaux. Les figures correspondantes sont toutes rassemblées en annexe.

4.2.1 tn : température

(en degrès Celsius, tableau 3D)
La moyenne du nombre de chiffres significatifs exacts pour les éléments du tableau recalculés à chaque itération varie entre 10.6 et 11.4.
Le nombre de chiffres significatifs exacts varie entre 9 et 15.

4.2.2 un : vitesse horizontale du courant

(en $m.s^{-1}$, tableau 3D)
La moyenne du nombre de chiffres significatifs exacts pour les éléments du tableau recalculés à chaque itération varie entre 5.4 et 7.2.
Le nombre de chiffres significatifs exacts varie entre 0 et 15.
Toutes les 100 itérations, a été calculé le nombre de chiffres significatifs exacts de tous les éléments du tableau *un*. Au cours des 1500 premières itérations en temps, seuls deux éléments du tableau n'ont aucun chiffre significatif exact. Ces deux résultats non significatifs, obtenus à l'itération 400, sont de l'ordre de 10^{-9}. Ces deux résultats sont négligeables par rapport à la moyenne de tous les éléments du tableau à l'itération 400 qui est $-5.1\,10^{-4}$.

4.2.3 vn : vitesse horizontale du courant

(en $m.s^{-1}$, tableau 3D)
La moyenne du nombre de chiffres significatifs exacts pour les éléments du tableau recalculés à chaque itération varie entre 5.6 et 7.3.
Le nombre de chiffres significatifs exacts varie entre 1 et 15.

4.2.4 wn : vitesse verticale du courant

(en $m.s^{-1}$, tableau 3D)
La moyenne du nombre de chiffres significatifs exacts pour les éléments du tableau recalculés à chaque itération varie entre 6.0 et 7.3.
Le nombre de chiffres significatifs exacts varie entre 1 et 15.

4.2.5 bsfn : fonction barotropique

(en $m^3.s^{-1}$, tableau 2D)
La moyenne du nombre de chiffres significatifs exacts pour les éléments du tableau recalculés à chaque itération varie entre 4.9 et 6.8.
Le nombre de chiffres significatifs exacts varie entre 3 et 15.
En fait, seul un élément du tableau à l'itération 1500 n'a que 3 chiffres significatifs exacts.

4.2.6 spgu : gradient de pression de surface horizontal

(tableau 2D)
La moyenne du nombre de chiffres significatifs exacts pour les éléments du tableau recalculés à chaque itération varie entre 5.4 et 7.0.
Le nombre de chiffres significatifs exacts varie entre 5 et 15.

4.2.7 spgv : gradient de pression de surface horizontal

(tableau 2D)
La moyenne du nombre de chiffres significatifs exacts pour les éléments du tableau recalculés à chaque itération varie entre 4.7 et 6.3.
Le nombre de chiffres significatifs exacts varie entre 1 et 15.
En fait, seul un élément du tableau à l'itération 400 n'a qu'un chiffre significatif exact.

4.2.8 q : condition de flux de chaleur à la surface

(em $w.m^{-2}$, tableau 2D)
La moyenne du nombre de chiffres significatifs exacts pour les éléments du tableau recalculés à chaque itération varie entre 11.7 et 13.5.
Le nombre de chiffres significatifs exacts varie entre 11 et 15.

4.2.9 rotn : vorticité relative

(en s^{-1}, tableau 3D)
La moyenne du nombre de chiffres significatifs exacts pour les éléments du tableau recalculés à chaque itération varie entre 4.6 et 5.9.
Le nombre de chiffres significatifs exacts varie entre 0 et 15.
Au cours des 1500 premières itérations en temps, 8 éléments du tableau *rotn* n'ont aucun chiffre significatif exact. Deux résultats non significatifs sont obtenus à l'itération 100 et un résultat non significatif est obtenu aux itérations 400, 500, 1000, 1100, 1400 et 1500. Ces résultats non significatifs correspondent à des éléments du tableau *rotn* de coordonnées distinctes. Ces résultats non significatifs, au maximum de l'ordre de $3.\,10^{-3}$ sont négligeables par rapport à la moyenne de tous les éléments du tableau qui à chaque itération est $-8.6\,10^{-10}$.

4.2.10 hdivn : divergence horizontale

(en s^{-1}, tableau 3D)
La moyenne du nombre de chiffres significatifs exacts pour les éléments du tableau recalculés à chaque itération varie entre 5.7 et 7.0.
Le nombre de chiffres significatifs exacts varie entre 0 et 15.
Seul un élément non significatif est obtenu à l'itération 400. Sa valeur ($5.6\,10^{-15}$) est négligeable par rapport à la moyenne de tous les éléments du tableau à l'itération 400 qui est $1.6\,10^{-10}$.

4.2.11 bsfd : dérivée par rapport au temps de la fonction barotropique

(en $m^3.s^{-2}$, tableau 2D)
La moyenne du nombre de chiffres significatifs exacts pour les éléments du tableau recalculés à chaque itération varie entre 1.8 et 3.9.
Le nombre de chiffres significatifs exacts varie entre 0 et 15.
A chaque itération, quelques éléments du tableau *bsfd* n'ont aucun chiffre significatif exact. Le nombre maximal d'éléments non significatifs, obtenu à l'itération 400, est 40.
Avec la bibliothèque CADNA, pour contrôler la précision d'une variable réelle, on change le type de celle-ci (real ou double precision) en un type stochastique. Une variable de type stochastique correspond à plusieurs variables réelles subissant des perturbations différentes lors de chaque opération arithmétique. La comparaison de ces différents résultats permet de déterminer, à tout moment lors de l'exécution du code, le nombre de chiffres significatifs exacts de la variable à contrôler. Lorsque cette variable n'a aucun significatif exact, la moyenne des variables réelles qui lui correspondent fournit l'ordre de grandeur du résultat. L'ordre de grandeur de chaque élément non significatif du tableau *bsfd* a ainsi été déterminé. Au vu du tableau 4.1, pour chaque itération multiple de 100, la valeur absolue de la moyenne des éléments du tableau *bsfd* est supérieure à l'ordre de grandeur maximum (en valeur absolue) des éléments non significatifs.

itération	moyenne de tous les éléments	maximum des éléments non significatifs
100	2.10^{-2}	1.10^{-5}
200	9.10^{-4}	9.10^{-6}
300	7.10^{-4}	1.10^{-4}
400	5.10^{-4}	8.10^{-5}
500	3.10^{-4}	1.10^{-4}
600	9.10^{-5}	7.10^{-5}
700	2.10^{-4}	6.10^{-5}
800	5.10^{-4}	4.10^{-5}
900	8.10^{-4}	9.10^{-5}
100	1.10^{-3}	6.10^{-5}
1100	2.10^{-3}	6.10^{-5}
1200	2.10^{-3}	3.10^{-7}
1300	3.10^{-3}	3.10^{-5}
1400	4.10^{-3}	1.10^{-4}
1500	5.10^{-3}	1.10^{-4}

TAB. 4.1 – *bsfd: comparaison des éléments non significatifs avec la moyenne des éléments*

4.2.12 en : énergie cinétique turbulente

(tableau 3D)
La moyenne du nombre de chiffres significatifs exacts pour les éléments du tableau recalculés à chaque itération varie entre 9.9 et 11.5.
Le nombre de chiffres significatifs exacts varie entre 8 et 15.

4.2.13 hprn : pression hydrostatique

(en *pa*, tableau 3D)
La moyenne du nombre de chiffres significatifs exacts pour les éléments du tableau recalculés à chaque itération varie entre 14.4 et 15.0.
Le nombre de chiffres significatifs exacts est 14 ou 15.

4.2.14 rhon : densité in situ

(en $kg.m^{-3}$, tableau 3D)
La moyenne du nombre de chiffres significatifs exacts pour les éléments du tableau recalculés à chaque itération varie entre 14.8 et 15.0.
Le nombre de chiffres significatifs exacts est 14 ou 15.

4.2.15 bn2n : fréquence de Brunt-Vaisala

(en s^{-2}, tableau 3D)
La moyenne du nombre de chiffres significatifs exacts pour les éléments du tableau recalculés à chaque itération varie entre 9.1 et 11.2.
Le nombre de chiffres significatifs exacts varie entre 8 et 15.

4.2.16 sn : salinité

(en *psu*, tableau 3D)
Dans le code qui nous a été fourni, la salinité ne subit aucune variation. Les éléments du tableau *sn* restent nuls au cours des 1500 itérations en temps et gardent la précision maximale (15 chiffres significatifs exacts).

4.2.17 qsr : radiation solaire

(en $w.m^{-2}$, tableau 2D)
De même que pour le tableau *sn*, les éléments du tableau *qsr* restent nuls au cours des 1500 itérations en temps et gardent la précision maximale (15 chiffres significatifs exacts).

4.2.18 Evolution de la qualité numérique de ua et va lors de l'appel aux procédures de préfixe dyn

Les éléments des tableaux *ua* et *va* sont initialisés à zéro dans la procédure *inidtr*, puis sont modifiés à chaque itération en temps lors des appels aux procédures de préfixe *dyn*: successivement *dynber*, *dynvor*, *dynhdf*, *dynzad*, *dynzdf*, *dynspg* et *dynnxt*. Dans la procédure *dynnxt*, le tableau *ua* (respectivement *va*) est affecté au tableau *un* (respectivement *vn*) ; puis certains éléments des tableaux *ua* et *va* sont fixés à zéro.

Les tableaux *ua* et *va* sont tridimensionnels. La plage de variation de leurs indices est (40,40,20). Les éléments modifiés dans les procédures de préfixe *dyn* sont : $ua(ji, jj, jk)$ et $va(ji, jj, jk)$, les

indices ji et jj variant de 2 à 39, l'indice jk variant de 1 à 19. Cependant deux procédures présentent une exception. Dans la procédure *dynzdf*, l'indice jj varie de 1 à 40. Dans la procédure *dynnxt*, l'indice jk varie de 1 à 20.

La moyenne du nombre de chiffres significatifs exacts des éléments des tableaux *ua* et *va* a été calculée à chaque itération en temps, après chaque appel à une procédure de préfixe *dyn*. Cette moyenne a été déterminée pour les éléments $ua(ji, jj, jk)$ et $va(ji, jj, jk)$, les indices ji et jj variant de 2 à 39 et l'indice jk de 1 à 19.

Soit *ua_dynber* (respectivement *va_dynber*) la moyenne du nombre de chiffres significatifs exacts des éléments du tableau *ua* (respectivement *va*) après l'appel à la procédure *dynber*. De même, soient *ua_dynvor*, *ua_dynhdf*, *ua_dynzad*, *ua_dynzdf* et *ua_dynspg* (respectivement *va_dynvor*, *va_dynhdf*, *va_dynzad*, *va_dynzdf* et *va_dynspg*) les moyennes calculées pour le tableau *ua* (respectivement *va*) après l'appel aux procédures *dynvor*, *dynhdf*, *dynzad*, *dynzdf* et *dynspg*. Enfin *ua_dynnxt* et *va_dynnxt* sont les moyennes calculées pour les tableaux *ua* et *va*, dans la procédure *dynnxt*, juste après qu'ils soient affectés à *un* et *vn*.

Influence de *dynber* sur la qualité numérique de *ua* et *va*

La différence *ua_dynber* – *ua_dynnxt* montre l'influence de la procédure *dynber* sur la qualité numérique de *ua*. Cette différence est négative et vaut en moyenne -1 au cours des 1500 itérations en temps : le tableau *ua* perd en moyenne 1 chiffre significatif à l'exécution de la procédure *dynber*. Après environ 60 itérations en temps, la différence *ua_dynber* – *ua_dynnxt* est au pire -1.9.

De même, la différence *va_dynber* – *va_dynnxt* montre l'influence de la procédure *dynber* sur la qualité numérique de *va*. Cette différence est négative et vaut en moyenne -0.7 au cours des 1500 itérations en temps : le tableau *va* perd en moyenne 0.7 chiffre significatif à l'exécution de la procédure *dynber*. Après environ 80 itérations en temps, la différence *ua_dynber* – *ua_dynnxt* est au pire -1.6.

Influence de *dynvor* sur la qualité numérique de *ua* et *va*

La différence *ua_dynvor* – *ua_dynber* montre l'influence de la procédure *dynvor* sur la qualité numérique de *ua*. Cette différence est positive et vaut en moyenne 1.3 au cours des 1500 itérations en temps : le tableau *ua* gagne en moyenne 1.3 chiffre significatif à l'exécution de la procédure *dynvor*. Après environ 75 itérations en temps, la différence *ua_dynber* – *ua_dynnxt* est au mieux 2.6.

De même, la différence *va_dynvor* – *va_dynber* montre l'influence de la procédure *dynvor* sur la qualité numérique de *va*. Cette différence vaut en moyenne 0.16 au cours des 1500 itérations en temps et après 100 itérations varie entre -0.7 et 1.4. Selon l'itération en cours, la procédure *dynvor* peut avoir un effet positif ou négatif sur la qualité numérique de *va*.

Influence de *dynhdf* sur la qualité numérique de *ua* et *va*

La procédure *dynhdf* a peu d'influence sur la qualité numérique de *ua* et de *va*.

La différence *ua_dynhdf* – *ua_dynvor* est négative, mais vaut en moyenne -0.08 au cours des 1500 itérations en temps. Après environ 120 itérations, la différence *ua_dynhdf* – *ua_dynvor* est au pire -0.7.

De même, la différence *va_dynhdf* – *va_dynvor* est négative, mais vaut en moyenne -0.04 au cours des 1500 itérations en temps. Après environ 70 itérations, la différence *va_dynhdf* – *va_dynvor* est au pire -0.5.

Influence de *dynzad* sur la qualité numérique de *ua* et *va*

La procédure *dynzad* a très peu d'influence sur la qualité numérique de *ua* et de *va*.

La différence *ua_dynzad* – *ua_dynhdf* est en valeur absolue au maximum de l'ordre de 10^{-3}.

Après 5 itérations en temps, la différence *va_dynzad* – *va_dynhdf* est en valeur absolue au maximum de l'ordre de 10^{-2}.

Influence de *dynzdf* sur la qualité numérique de *ua* et *va*

La procédure *dynzdf* a peu d'influence sur la qualité numérique de *ua* et de *va*.

La différence *ua_dynzdf* – *ua_dynzad* est négative, mais vaut en moyenne -0.1 au cours des 1500 itérations en temps. La différence *ua_dynzdf* – *ua_dynzad* est au pire -0.4.

De même, la différence *va_dynzdf* – *va_dynzad* est négative, mais vaut en moyenne -0.43 au cours des 1500 itérations en temps. La différence *va_dynzdf* – *va_dynzad* est au pire -0.6.

Influence de *dynspg* sur la qualité numérique de *ua* et *va*

La différence *ua_dynspg* – *ua_dynzdf* montre l'influence de la procédure *dynspg* sur la qualité numérique de *ua*. Cette différence est négative et vaut en moyenne -1.8 au cours des 1500 itérations en temps : le tableau *ua* perd en moyenne 1.8 chiffre significatif à l'exécution de la procédure *dynspg*. La différence *ua_dynspg* – *ua_dynzdf* est au pire -3.5.

De même, la différence *va_dynspg* – *va_dynzdf* montre l'influence de la procédure *dynspg* sur la qualité numérique de *va*. Cette différence est négative et vaut en moyenne -1.1 au cours des 1500 itérations en temps : le tableau *va* perd en moyenne 1.1 chiffre significatif à l'exécution de la procédure *dynspg*. La différence *va_dynspg* – *va_dynzdf* est au pire -2.9.

Influence de *dynnxt* sur la qualité numérique de *ua* et *va*

La différence *ua_dynnxt* – *ua_dynspg* montre l'influence de la procédure *dynnxt* sur la qualité numérique de *ua*. Cette différence est positive et vaut en moyenne 1.8 au cours des 1500 itérations en temps : le tableau *ua* gagne en moyenne 1.8 chiffre significatif à l'exécution de la procédure *dynnxt*. La différence *ua_dynnxt* – *ua_dynspg* est au mieux 3.5.

De même, la différence *va_dynnxt* – *va_dynspg* montre l'influence de la procédure *dynnxt* sur la qualité numérique de *va*. Cette différence est positive et vaut en moyenne 2.1 au cours des 1500 itérations en temps : le tableau *va* gagne en moyenne 2.1 chiffre significatif à l'exécution de la procédure *dynspg*. La différence *va_dynnxt* – *va_dynspg* est au mieux 3.7.

En conclusion, nous pouvons classer les procédures de préfixe *dyn* selon leur influence sur *ua* et *va*.
Les procédures *dynber* et *dynspg* ont une influence négative sur la qualité numérique de *ua* et *va*.
La procédure *dynnxt* a une influence positive sur la qualité numérique de *ua* et *va*. La procédure *dynvor* a également une influence positive sur la qualité numérique de *ua*. Lors de l'exécution de la procédure *dynvor*, *va* peut gagner ou perdre quelques chiffres significatifs.
Enfin les procédures *dynhdf*, *dynzad* et *dynzdf* ont peu d'influence sur la qualité numérique de *ua* et *va*.

4.2.19 Analyse du fichier barotropic.stat

Une partie des variables précédemment analysées sont régulièrement sauvegardées lors de l'exécution du code dans des fichiers dont le nom dépend de l'itération en cours.
Le fichier *barotropic.stat* contient des informations relatives à chaque itération. Les valeurs $niter$, res et $sqrt(epsr)/eps$, calculées dans la procédure *bsfpcg*, sont sauvegardées dans le fichier *barotropic.stat* par la procédure *stpctl*.

$niter$ désigne le nombre d'itérations nécessaires à la convergence de la méthode du gradient conjugué.

res désigne le résidu du solveur : $res = sqrt(rnorme)$ où $rnorme$ est le produit scalaire des tableaux gcr et $zgwgt$.

eps est fixé dans *parlec.F* à 10^{-2} et $epsr = eps * eps * rnorme$.

Au cours des 1500 premières itérations en temps, $niter$ varie de 1 à 137.
A la première itération, res et $sqrt(epsr)/eps$ ont respectivement 12 et 15 chiffres significatifs exacts. Puis la précision de res et de $sqrt(epsr)/eps$ décroît régulièrement jusqu'à l'itération 87. Ensuite jusqu'à l'itération 1500, le nombre de chiffres significatifs exacts de res (respectivement de $sqrt(epsr)/eps$) varie de 0 à 3 (respectivement de 3 à 6).

Chapitre 5

Conclusions et perspectives

Les instabilités numériques détectées sont bien souvent liées les unes aux autres.

Par exemple, le calcul de $rnorme$ dans le fichier *bsfpcg.F* provoque

- des multiplications instables : $rnorme = sdot(jpi * jpj, gcr, 1, zgwgt, 1)$
- des racines carrées instables : $res = sqrt(rnorme)$
- des tests instables : $rnorme.lt.epsr$
- des divisions instables : $beta = rnorme/rr$, où la valeur $rnorme$ est affectée à rr avant que celle-ci ne soit modifiée.

De même le calcul de $radd$ dans le fichier *bsfpcg.F* provoque

- des multiplications instables : $radd = sdot(jpi * jpj, gcdes, 1, zgwgt, 1)$
- des divisions instables : $alph = rr/radd$.

Beaucoup d'instabilités proviennent du gradient conjugué préconditionné utilisé dans le fichier *bsfpcg.F* et peuvent être évitées en changeant le test d'arrêt de la méthode.

L'évolution de la précision de certains tableaux a été analysée. Pour chaque tableau étudié, la moyenne du nombre de chiffres significatifs exacts de ses éléments a été calculée toutes les 100 itérations, jusqu'à l'itération 1500. Cette moyenne ne tient pas compte des éléments jamais recalculés qui gardent toujours 15 chiffres significatifs exacts. Cette moyenne est toujours minimale à l'itération 1500, sauf pour le tableau *bsfd* où elle est minimale à l'itération 400.

Les tableaux étudiés peuvent être classés en différentes catégories selon leur précision.

- Seul le tableau *bsfd* a une mauvaise précision : 1.8 chiffres significatifs exacts en moyenne à l'itération 400.
- Les tableaux $bsfn$, $spgu$, $spgv$, un, vn, wn, $rotn$ et $hdivn$ ont une précision moyenne. A l'itération 1500, leur nombre moyen de chiffres significatifs exacts varie de 4.6 à 6.0.
- Les tableaux tn, q, en et $bn2n$ ont une bonne précision. A l'itération 1500, leur nombre moyen de chiffres significatifs exacts varie de 9.1 à 11.7.
- Les tableaux $hprn$ et $rhon$ ont une excellente précision. Au cours des 1500 itérations, leurs éléments ont toujours 14 ou 15 chiffres significatifs exacts.

Certains éléments des tableaux étudiés n'ont aucun chiffre significatif exact. Cependant leur ordre de grandeur est toujours inférieur à la valeur absolue de la moyenne de tous les éléments du tableau.

Les résultats obtenus en double précision (64 bits) permettent de prévoir ceux obtenus en utilisant une autre précision, telle que la quadruple précision (128 bits). En effet, le nombre de chiffres significatifs perdus au cours d'un calcul en arithmétique virgule flottante ne dépend pas de la précision utilisée.

Les perspectives à cette étude sont nombreuses. Il serait intéressant d'étudier l'impact sur la validité des résultats de certaines modifications du code, telles que le changement des dimensions du bassin ou le calcul du champ de vent avec rotationnel. L'impact des erreurs de données sur les résultats pourrait également être analysé grâce à la bibliothèque CADNA. Une autre prepective est l'étude de la validité numérique des calculs inhérents à la version 8.1.

Chapitre 6

Annexe 1 : Liste des procédures appelées

La liste ci-dessous présente les procédures nécessaires à l'exécution du code qui nous a été fourni. Chaque fichier de nom "nom.F" contient la procédure "nom". Les procédures sont classées dans l'ordre dans lequel elles sont appelées.

```
opa
  inipar
     parcst
     parlec
     parctl
  inimpp
  inidom
     dommba
     dommsk
     domhgr
     domzgr
        domzgr.z.h
     domstp
     domwri
  inispg
     bsfmat
  inidta
     dtacof
  inidtr
    eos
    bn2
    prh
    wzv
  inimix
  step
    tau
    flx
    forkjoin
    zdfmix
       zdfmix.tke.h
       mpplnk2
    dynber
    dynvor
```

```
      dynvor.enstrophy.h
   dynhdf
      dynhdf.laplacian.h
   trahad
   hdfslp
   dynzad
   dynzdf
      dynzdf.implicit.h
         zdf.matrixsolver.h
   trazad
   trazdf
   tranxt
   eos
   bn2
   prh
   dynspg
       bsfpcg
          mpprsum
       mpprsum
   dynnxt
   div
   cur
   wzv
   stpctl
   rstwri
      write2
      write3
   diawri
     ctlopn
```

Sont également nécessaires les fichiers suivants :

```
allcray.f
parameter.h
common.h
stafun.h
```

Chapitre 7

Annexe 2 : Figures

7.1 Evolution des instabilités numériques

Le nombre de multiplications instables survenues à chaque itération en temps a été déterminé. La figure 7.1 présente l'évolution du nombre de multiplications instables jusqu'à 185 itérations en temps. Au cours des 185 premières itérations, plus de 500000 multiplications instables ont été détectées.

Le nombre de racines carrées, de divisions et de tests instables survenus toutes les 100 itérations en temps a été déterminé. Les figures 7.2 (respectivement 7.3 et 7.4) présentent l'évolution du nombre de racines carrées (respectivement divisions et tests) instables jusqu'à 1500 itérations en temps.

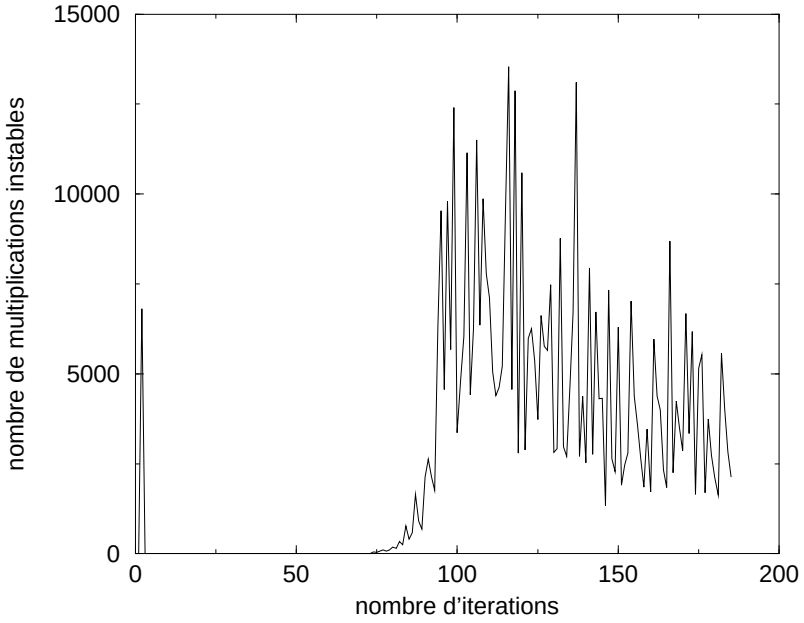

FIG. 7.1 – *Evolution du nombre de multiplications instables jusqu'à 185 itérations*

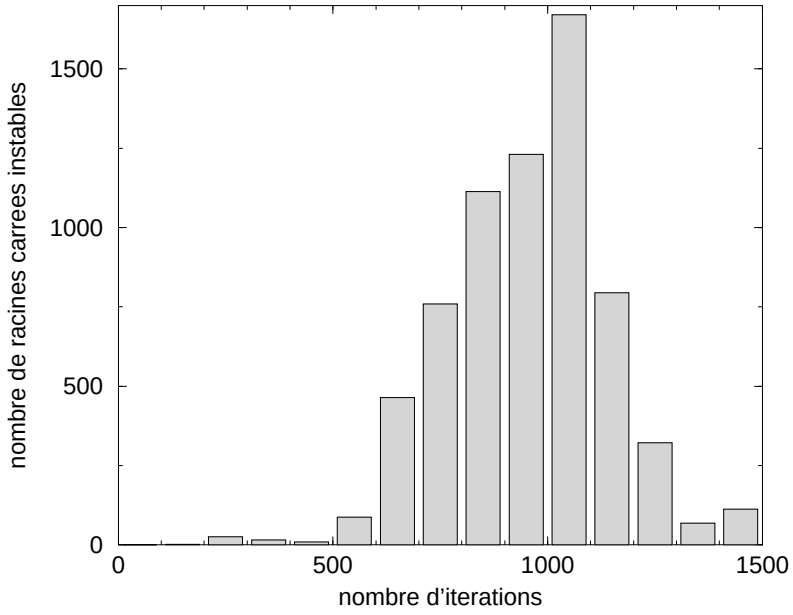

FIG. 7.2 – *Evolution du nombre de racines carrées instables jusqu'à 1500 itérations*

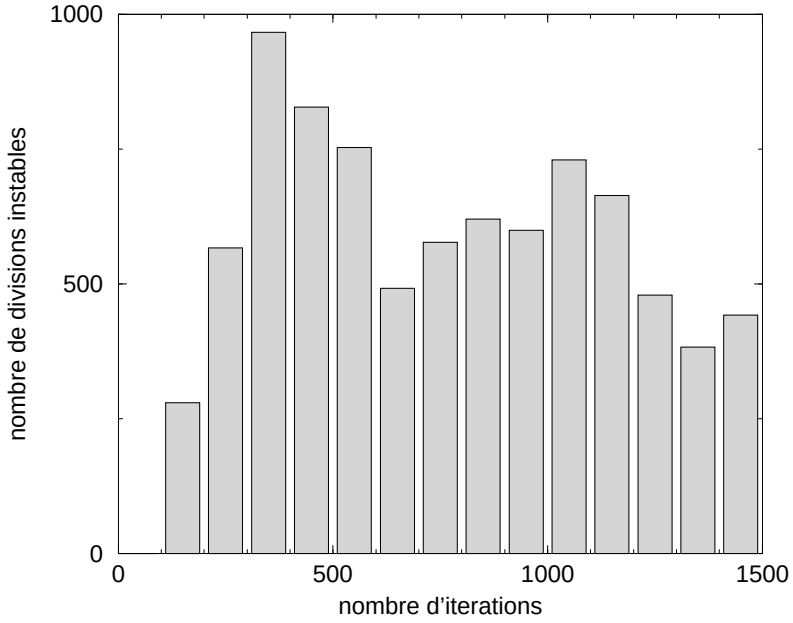

FIG. 7.3 – *Evolution du nombre de divisions instables jusqu'à 1500 itérations*

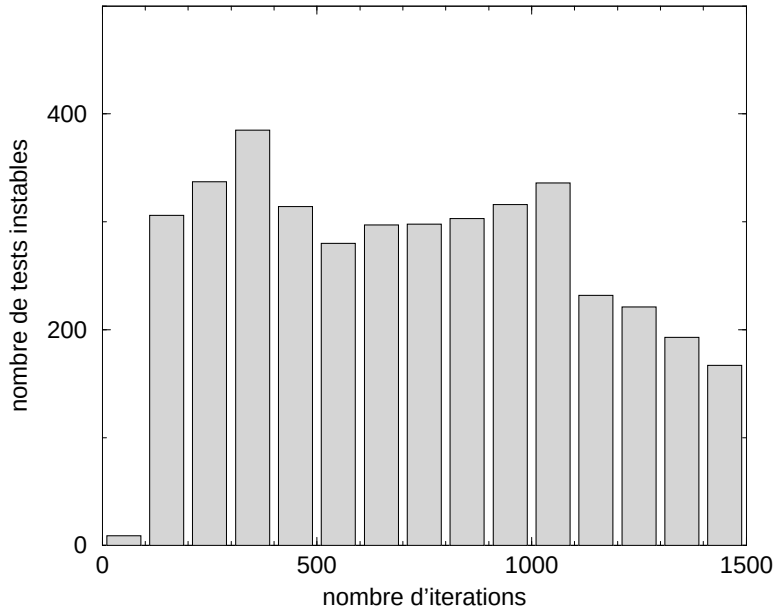

FIG. 7.4 – *Evolution du nombre de tests instables jusqu'à 1500 itérations*

7.2 Evolution de la précision des variables

Chaque variable analysée au chapitre 4, est un tableau bidimensionnel ou tridimensionnel. Certains éléments des tableaux ne sont pas modifiés au cours du temps et gardent la précision maximale (15 chiffres significatifs exacts en double précision).

Pour chaque tableau, nous présentons deux figures.

- La première décrit l'évolution au cours des 1500 premières itérations en temps de la moyenne du nombre de chiffres significatifs exacts des éléments du tableau. Cette moyenne a été déterminée, sans tenir compte des éléments du tableau jamais recalculés.
- La seconde figure représente la répartion des éléments du tableau selon leur nombre de chiffres significatifs exacts à l'itération 1500.

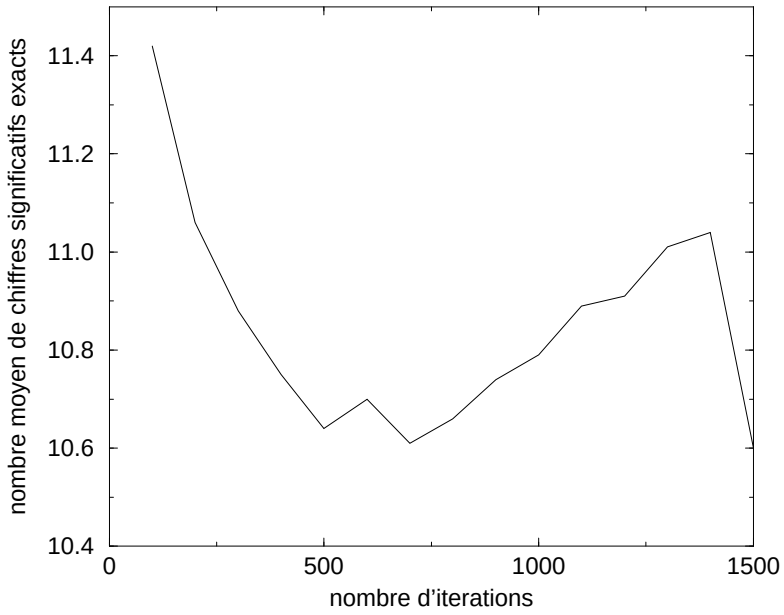

FIG. 7.5 – *Evolution de la précision des résultats au cours des 1500 itérations*

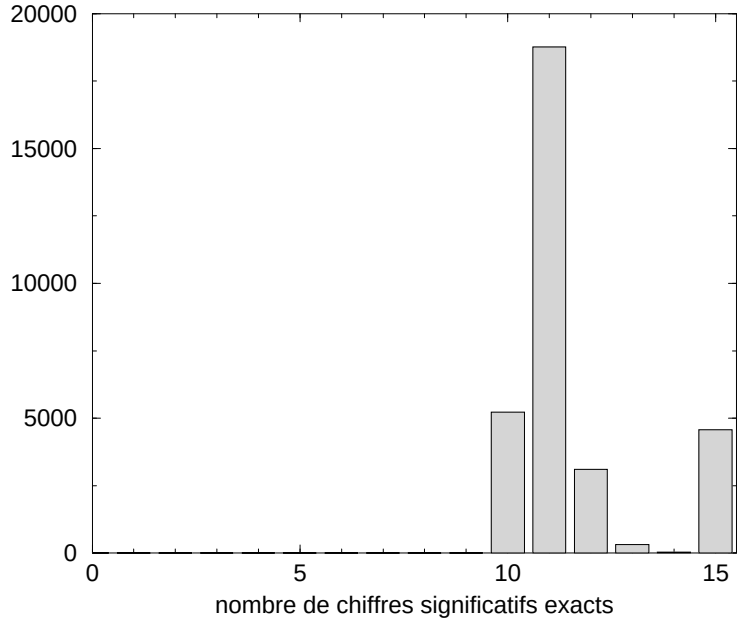

FIG. 7.6 – *Répartion des résultats selon leur précision à l'itération 1500*

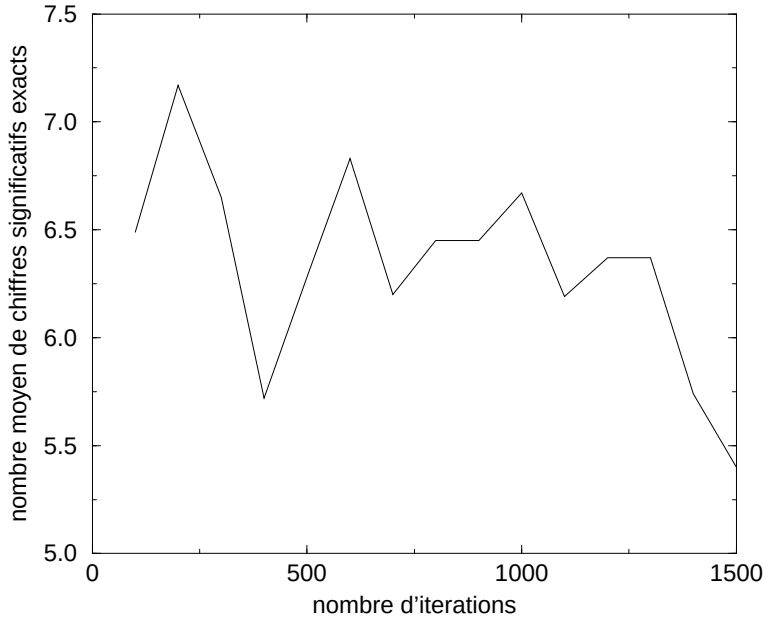

FIG. 7.7 – *Evolution de la précision des résultats au cours des 1500 itérations*

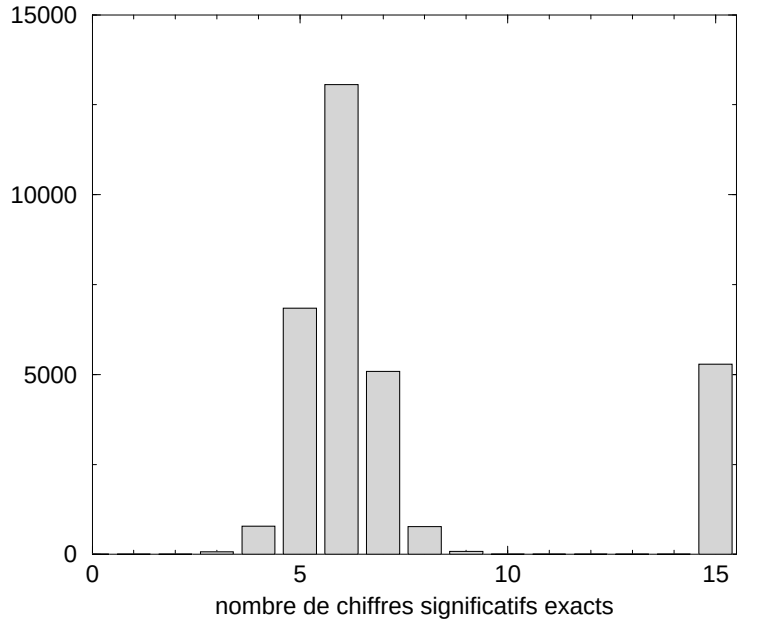

FIG. 7.8 – *Répartion des résultats selon leur précision à l'itération 1500*

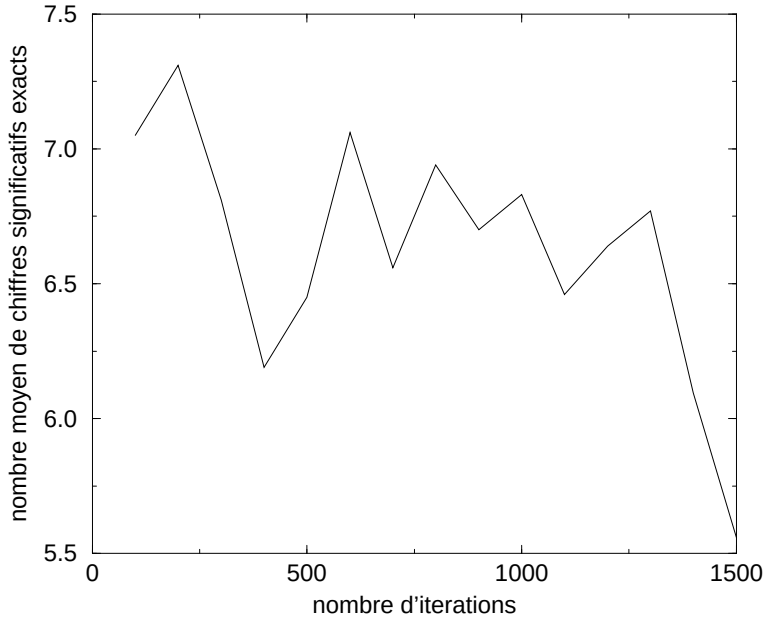

FIG. 7.9 – *Evolution de la précision des résultats au cours des 1500 itérations*

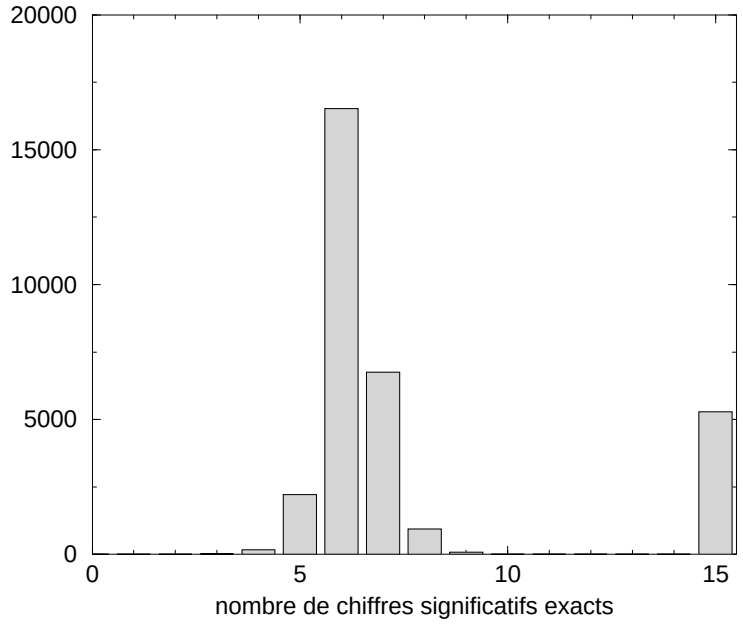

FIG. 7.10 – *Répartion des résultats selon leur précision à l'itération 1500*

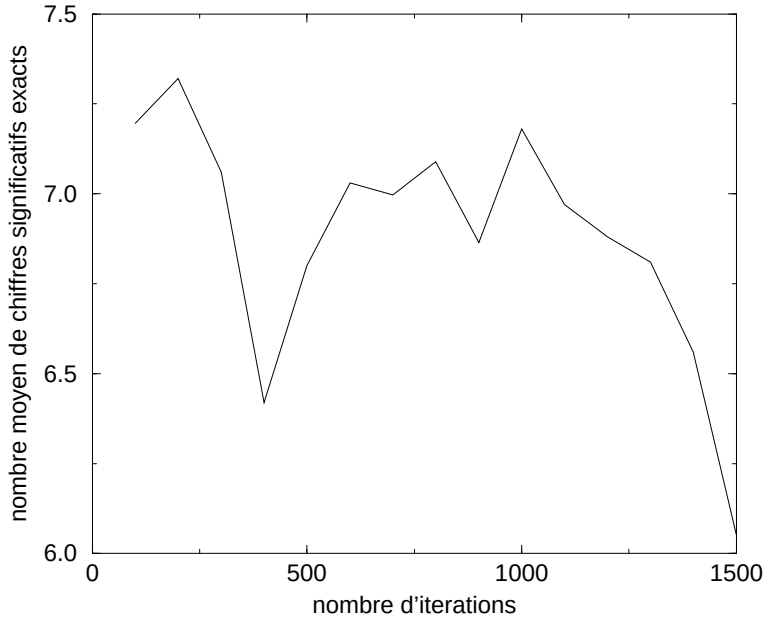

FIG. 7.11 – *Evolution de la précision des résultats au cours des 1500 itérations*

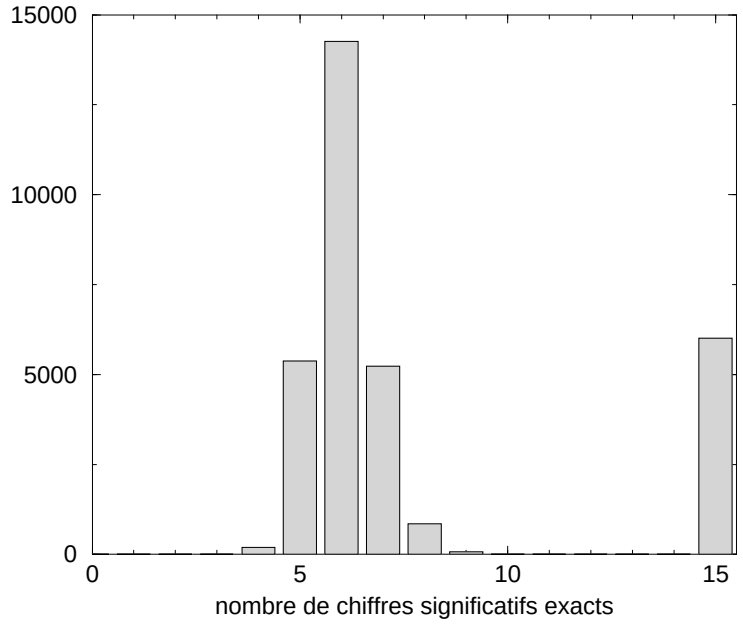

FIG. 7.12 – *Répartion des résultats selon leur précision à l'itération 1500*

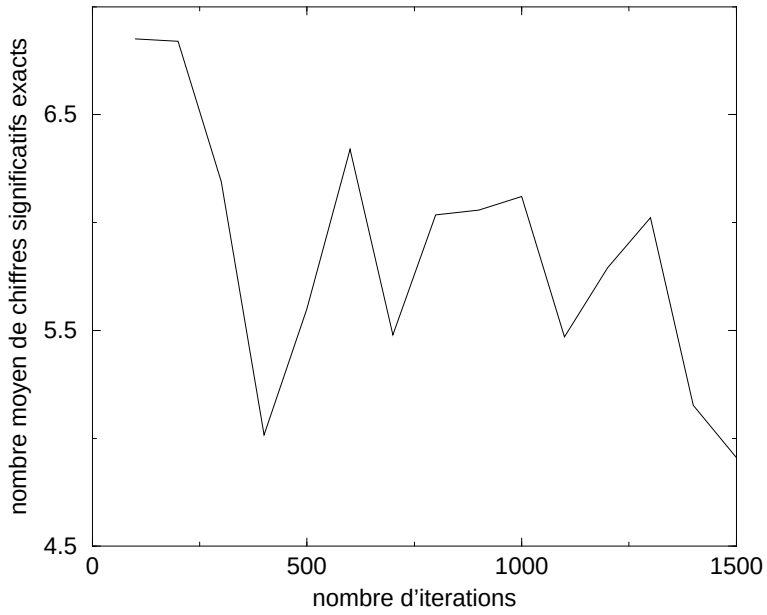

FIG. 7.13 – *Evolution de la précision des résultats au cours des 1500 itérations*

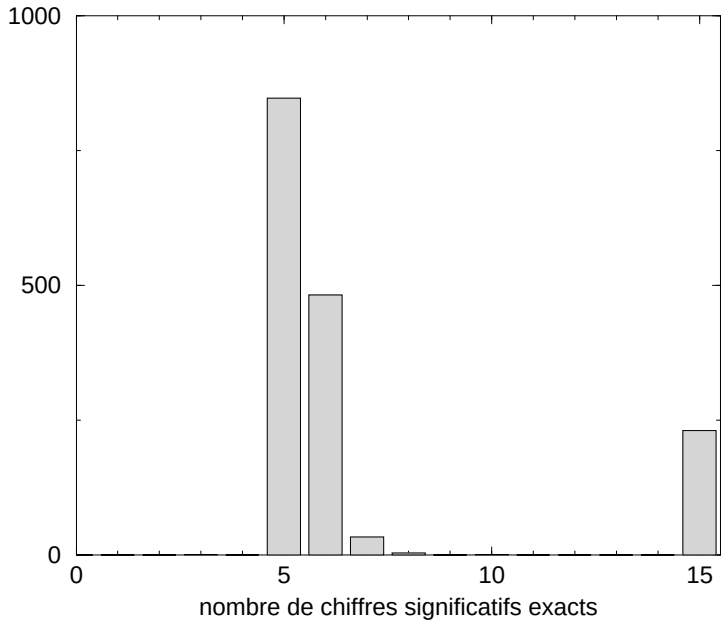

FIG. 7.14 – *Répartion des résultats selon leur précision à l'itération 1500*

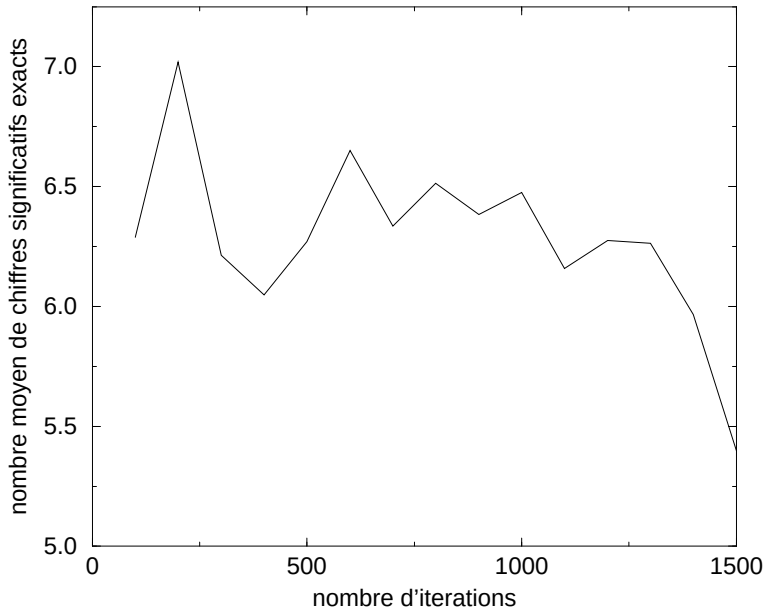

FIG. 7.15 – *Evolution de la précision des résultats au cours des 1500 itérations*

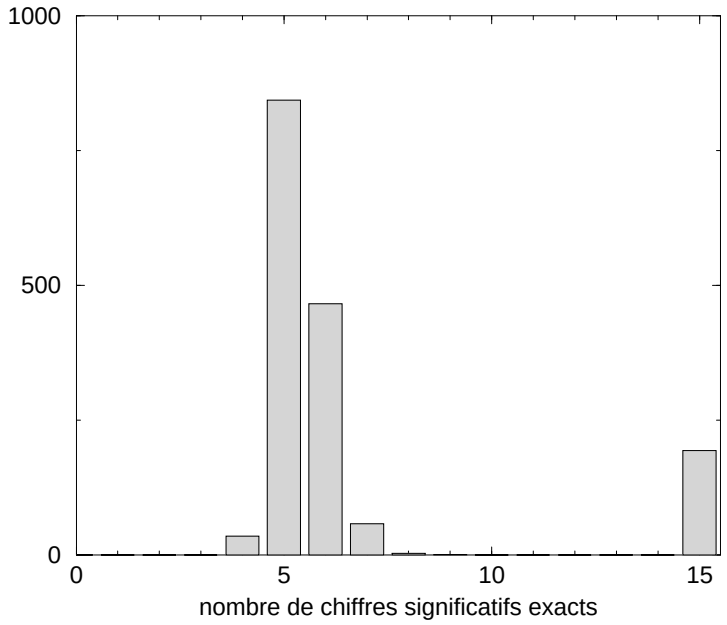

FIG. 7.16 – *Répartion des résultats selon leur précision à l'itération 1500*

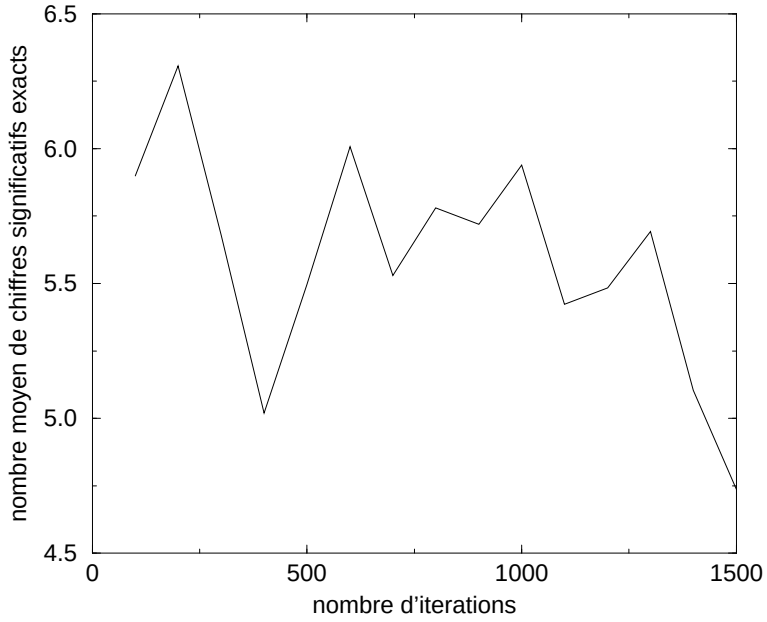

FIG. 7.17 – *Evolution de la précision des résultats au cours des 1500 itérations*

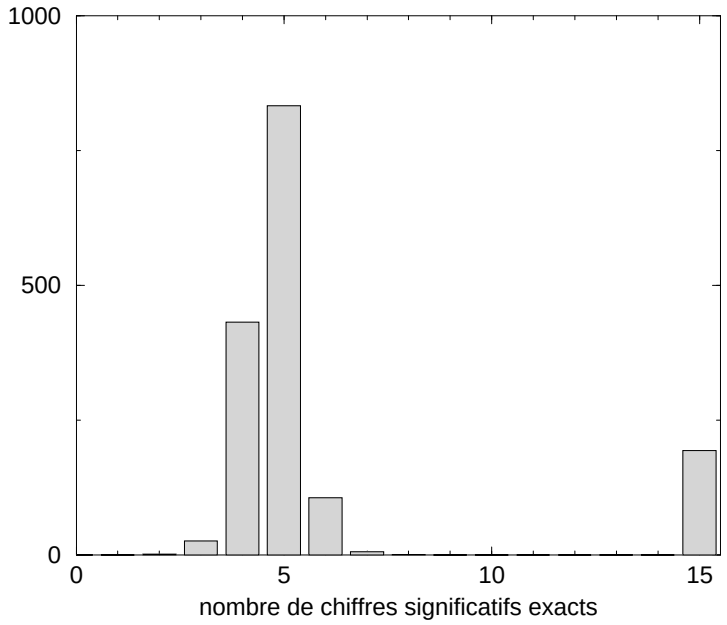

FIG. 7.18 – *Répartion des résultats selon leur précision à l'itération 1500*

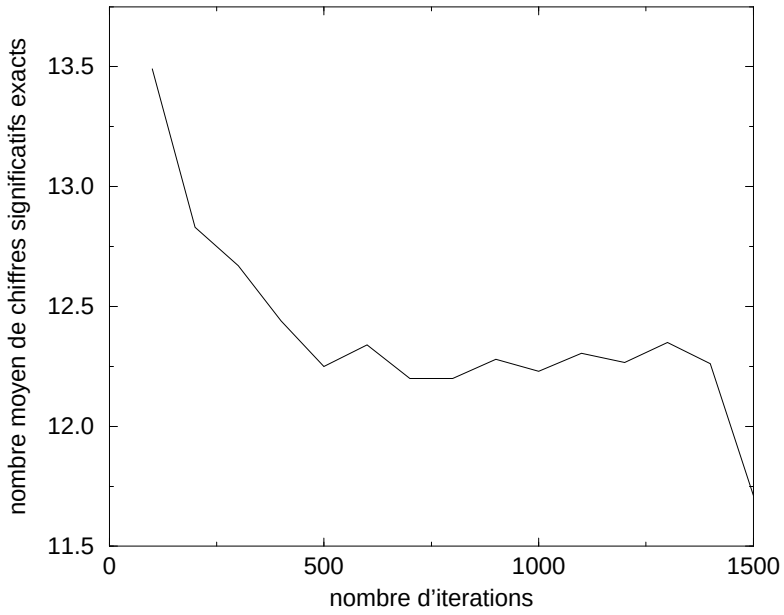

FIG. 7.19 – *Evolution de la précision des résultats au cours des 1500 itérations*

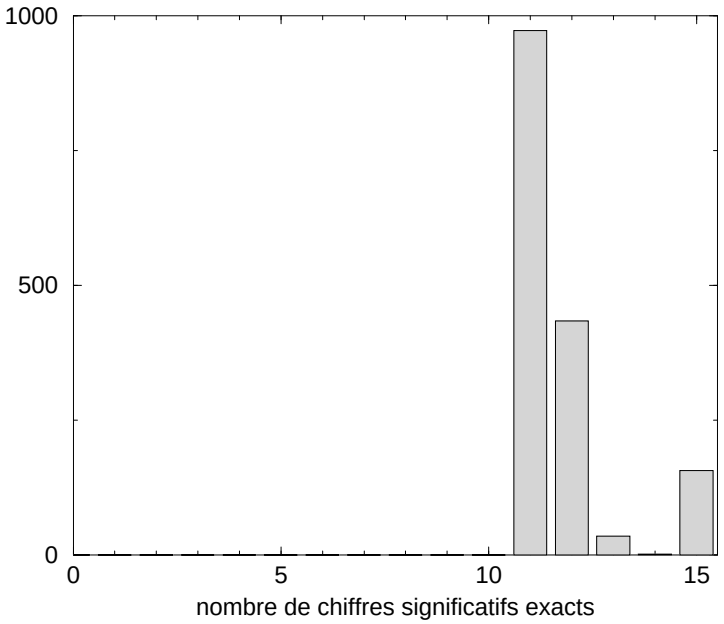

FIG. 7.20 – *Répartion des résultats selon leur précision à l'itération 1500*

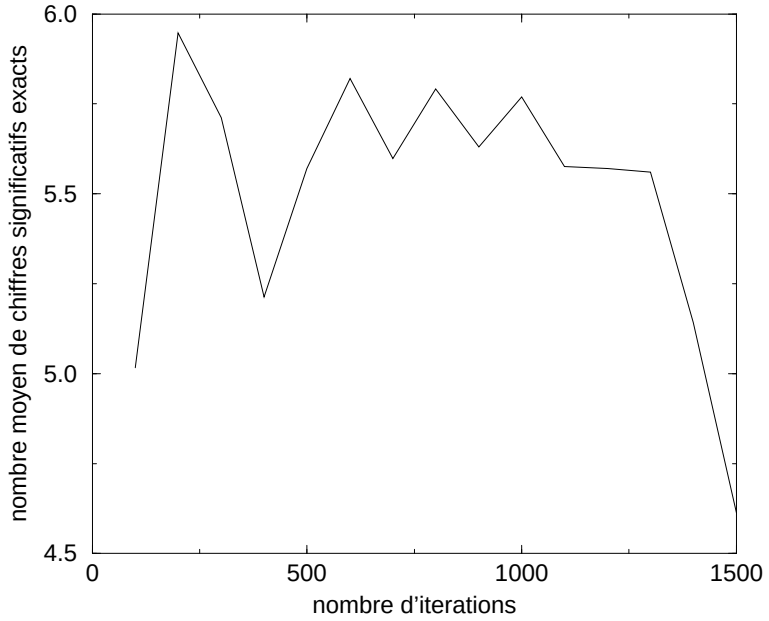

FIG. 7.21 – *Evolution de la précision des résultats au cours des 1500 itérations*

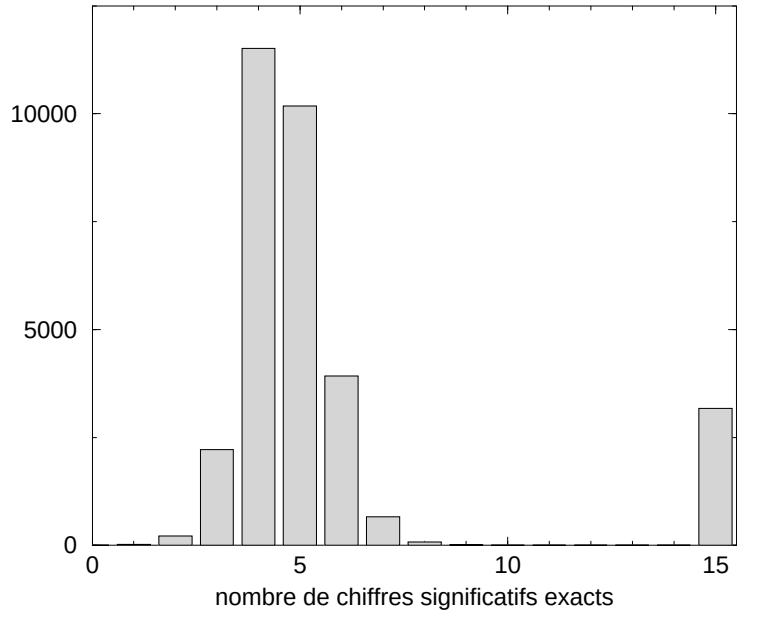

FIG. 7.22 – *Répartion des résultats selon leur précision à l'itération 1500*

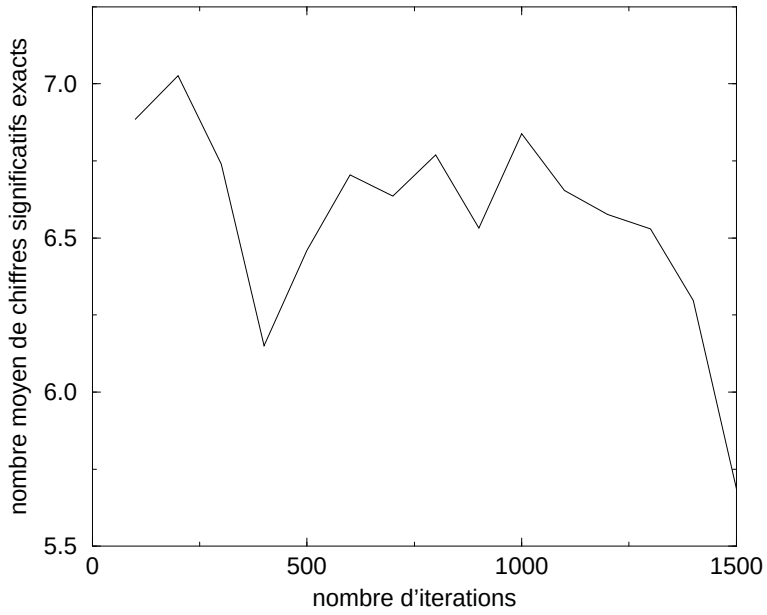

FIG. 7.23 – *Evolution de la précision des résultats au cours des 1500 itérations*

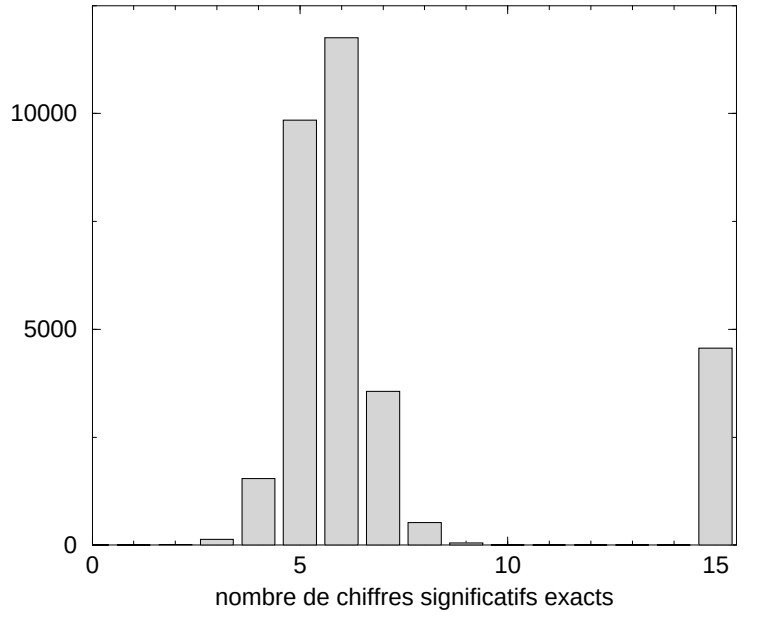

FIG. 7.24 – *Répartion des résultats selon leur précision à l'itération 1500*

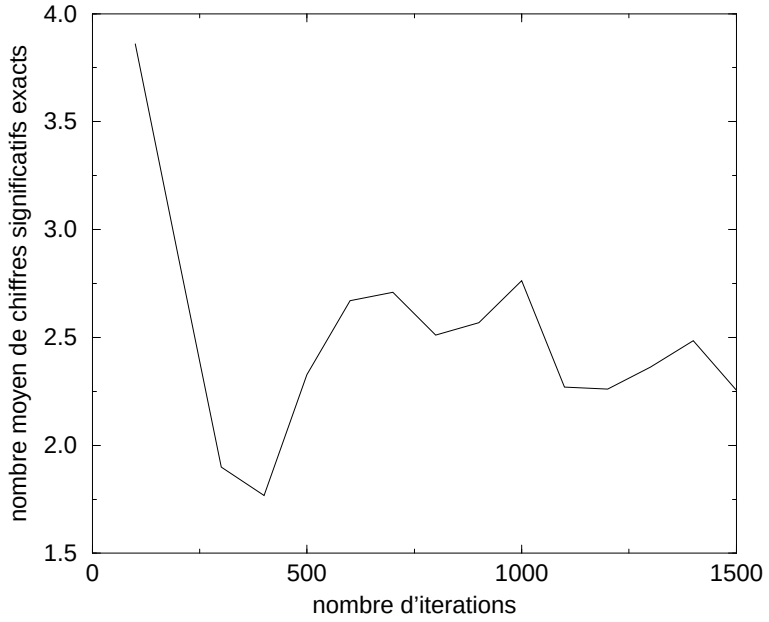

FIG. 7.25 – *Evolution de la précision des résultats au cours des 1500 itérations*

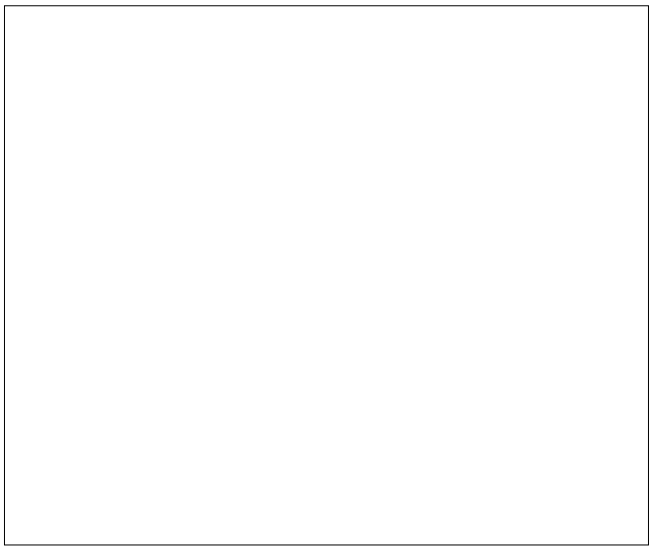

FIG. 7.26 – *Répartion des résultats selon leur précision à l'itération 1500*

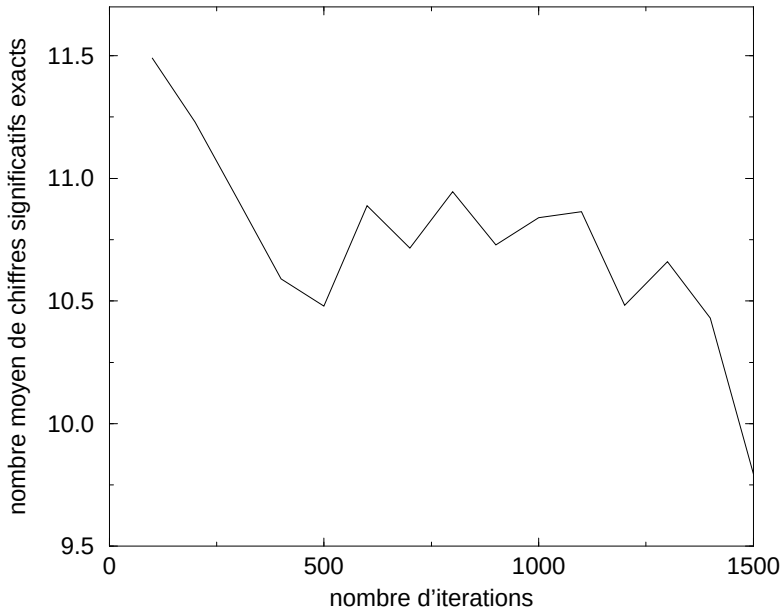

FIG. 7.27 – *Evolution de la précision des résultats au cours des 1500 itérations*

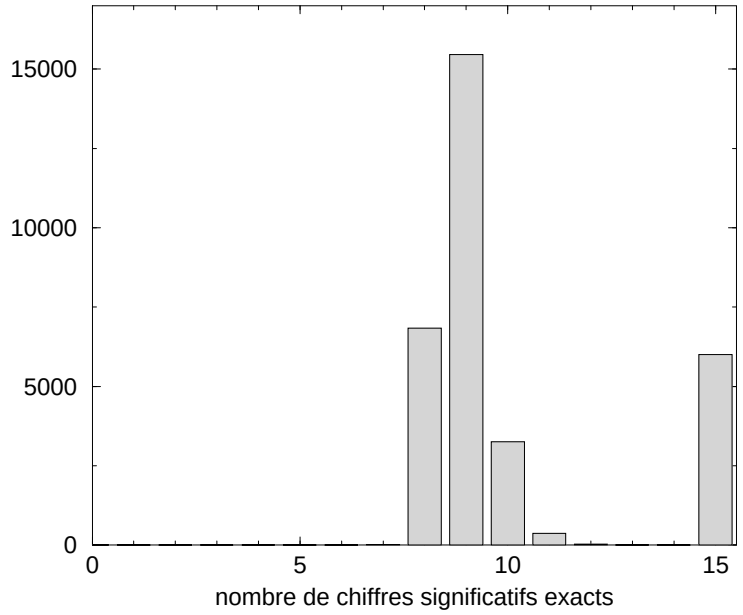

FIG. 7.28 – *Répartion des résultats selon leur précision à l'itération 1500*

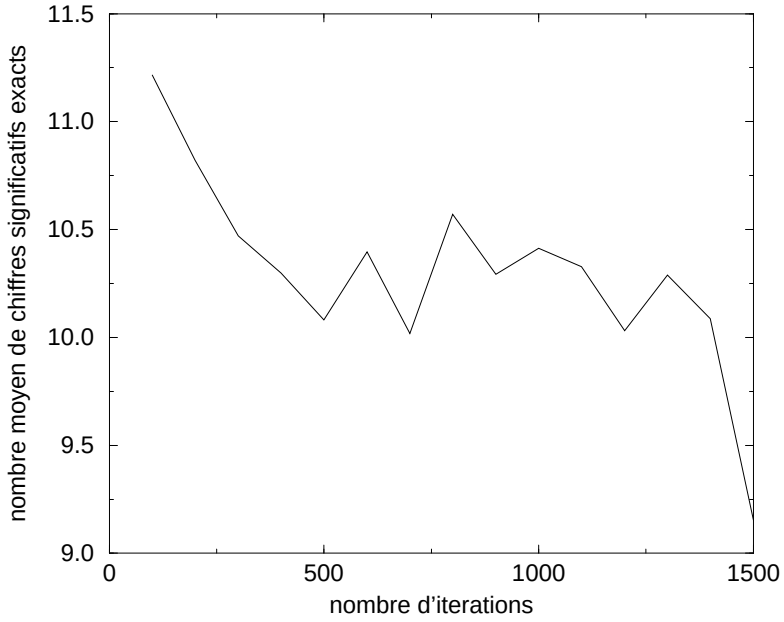

FIG. 7.29 – *Evolution de la précision des résultats au cours des 1500 itérations*

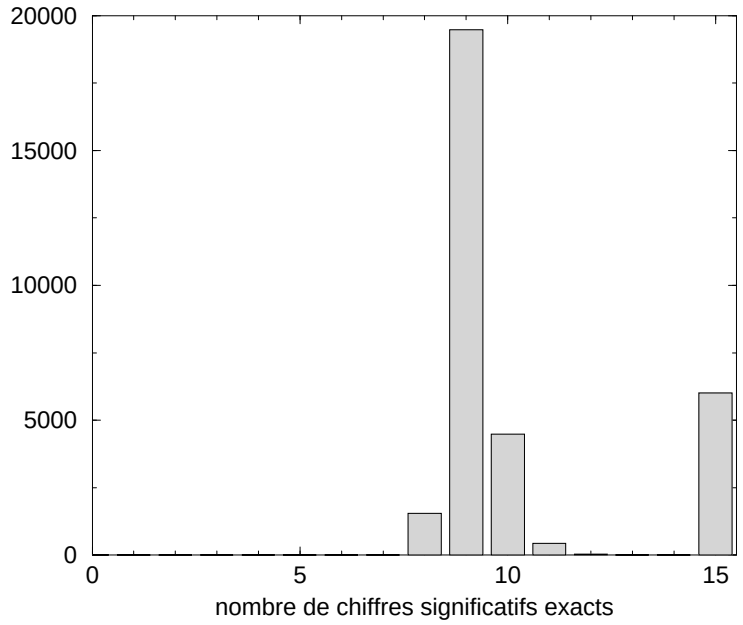

FIG. 7.30 – *Répartion des résultats selon leur précision à l'itération 1500*

Remerciements

Nous adressons nos plus vifs remerciements aux membres de l'Institut Pierre-Simon Laplace (IPSL) et du Laboratoire d'Océanographie DYnamique et de Climatologie (LODYC) pour leur convivialité et l'intérêt qu'ils ont porté à cette étude.
Nous tenons à remercier particulièrement Marie-Alice Foujols et Maurice Imbard pour leur aide précieuse.

Bibliographie

[1] URL address : http://www-anp.lip6.fr/cadna/.

[2] Qualité des calculs sur odinateur. rédaction collégiale, Masson, Paris, 1997.

[3] Blanke B. & Delecluse P. Variability of the tropical atlantic ocean simulated by a general circulation model with two different mixed layer physics. *J. Phys. Oceanogr.*, Vol. 23:pages 1363–1388, 1993.

[4] Braconnot P., Marti O., & Joussaume S. Adjustment and feedbacks in a global coupled ocean-atmosphere model. *Climate Dyn.*, Vol. 13:pages 507–519, 1997.

[5] Cassou C., Noyret P., Sevault E., Thual O., Terray L., Beaucourt D. & Imbard M. Distributed ocean-atmosphere modelling and sensitivity to the coupling flux precision: the cathode project. *Mon. Wea. Rev.*, in press.

[6] Chesneaux J.M. Modélisation théorique et conditions de validité de la méthode CESTAC. *C.R.A.S., Paris*, série 1(tome 307):pages 417–422, 1988.

[7] Chesneaux J.M. L'arithmétique stochastique et le logiciel CADNA. *Habilitation à diriger des recherches, Laboratoire MASI-IBP, Université Pierre et Marie Curie*, 1995.

[8] Chesneaux J.M. & Vignes J. Sur la robustesse de la méthode CESTAC. *C.R.A.S., Paris*, série 1(tome 307):pages 855–860, 1988.

[9] Chesneaux J.M. & Vignes J. Les fondements de l'arithmétique stochastique. *C.R.A.S., Paris*, série 1(tome 307):pages 1435–1440, 1992.

[10] Guilyardi E. & Madec G. Performance of the opa/arpege-t21 global ocean-atmosphere coupled model. *Climate Dyn.*, Vol. 13:pages 149–165, 1997.

[11] Madec G. & Imbard M. A global ocean mesh to overcome the north pole singularity. *Climate Dyn.*, Vol. 12:pages 381–388, 1996.

[12] Maes C., Madec G. & Delecluse P. Sensitivity of an equatorial pacific ogcm to the lateral diffusion. *Mon. Wea. Rev.*, 1996.

[13] Vignes J. Estimation de la précision des résultats de logiciels numériques. *La Vie des Sciences, Comptes Rendus, série générale*, Vol. 7:pages 93–145, 1990.

[14] Vignes J. A stochastic arithmetic for reliable scientific computation. *Math. Comp. Simul.*, Vol. 35:pages 233–261, 1993.

[15] Vintzileos A. & Sadourny R. A general interface between an atmospheric general circulation model and underlying ocean and land surface models: delocalized physics scheme. *Mon. Wea. Rev.*, in press.