

Variables de Abinit

9.6.2

<https://docs.abinit.org/variables>

ÉQUIPE PÉDAGOGIQUE

Jean-Pierre TCHAPET NJAFA, PhD



Table des matières

1	<i>Abinit</i>	<i>1</i>
1.1	Variables d'entrée basiques	1
1.2	Variables d'entrée bse (Bethe-Salpeter)	17
2	<i>Anaddb</i>	<i>18</i>
3	<i>Multibinit</i>	<i>19</i>
4	<i>Optic</i>	<i>20</i>
5	<i>α-TDEP</i>	<i>21</i>
6	<i>Aim</i>	<i>22</i>
7	<i>Paramètres externes</i>	<i>23</i>
8	<i>Statistique</i>	<i>24</i>
	<i>Index</i>	<i>25</i>

Abinit

Contenu du chapitre



1.1 Variables d'entrée basiques

1.2 Variables d'entrée bse (Bethe-Salpeter)

1.1 Variables d'entrée basiques

Cette section liste et donne une description des noms (*keywords*) des variables d'entrée de base à utiliser dans le fichier d'entrée pour l'exécutable de **abinit**.

»»» Caractéristique(s) : –

accuracy
ACCURACY

Type de variable : entière ;
Valeur par défaut : 0

Dimensions : scalaire ;

Elle permet d'ajuster l'exactitude du ground-state ou du calcul DFPT **optdriver** = 0 ou 1, en paramétrant automatiquement les variables conformément au Tableau 1.1.1.

accuracy = 4 correspond au paramétrage par défaut de Abinit.

»»» Caractéristique(s) : EVOLVING, LENGTH

acell
CELL lattice vector scalling

Type de variable : réelle ;
Valeur par défaut : 3*1

Dimensions : (3) ;

Elle donne les échelles de longueur par lesquelles les translations primitives sans dimension (**rprim**) doivent être multipliées. Par défaut, elle est donnée en unités atomiques de Bohr (1 Bohr = 0.529 177 210 8Å), bien que l'Angstrom puisse être spécifié.

»»» Caractéristique(s) : INPUT_ONLY

angdeg
ANGLES in **DEG**rees

Type de variable : réelle ;
Valeur par défaut : aucune

Dimensions : (3) ;

Elle donne les angles entre les directions des tableaux primitifs d'une cellule unitaire (en degrés), comme une alternative au vecteur d'entrée **rprim**. Elle devra être utilisée pour paramétrer **rprim**, qui, avec le tableau **acell**, seront utilisés pour définir des vecteurs primitifs.

- **angdeg(1)** est l'angle entre le second et le troisième vecteur ;
- **angdeg(2)** est l'angle entre le premier et le troisième vecteur ;
- **angdeg(3)** est l'angle entre le premier et le second vecteur.

Si les trois angles sont égaux à 10^{-12} près (sauf s'ils sont exactement égaux à 90°), les trois vecteurs primitifs sont choisis tel que la symétrie trigonale qui les échange est le long de l'axe cartésien z :

accuracy	1	2	3	4	5	6
ecut	E_min	E_med	E_med	E_max	E_max	E_max
pawecutdg	ecut	ecut	1.2*ecut	1.5*ecut	2*ecut	2*ecut
fband	0.5	0.5	0.5	0.5	0.75	0.75
boxcutmin	1.5	1.8	1.8	2.0	2.0	2.0
bxctmindg	1.5	1.8	1.8	2.0	2.0	2.0
pawxcdev	1	1	1	1	2	2
pawmixdg	0	0	0	0	1	1
pawovlp	10	7	7	5	5	5
pawnhatxc	0	1	1	1	1	1
tolvrs	10 ⁻³	10 ⁻⁵	10 ⁻⁷	10 ⁻⁹	10 ⁻¹⁰	10 ⁻¹²
tolmxf	10 ⁻³	5.0 × 10 ⁻⁴	10 ⁻⁴	5.0 × 10 ⁻⁵	10 ⁻⁶	10 ⁻⁶
optforces	1	1	2	2	2	2
timopt	0	0	1	1	1	1
npulayit	4	7	7	7	15	15
nstep	30	30	30	30	50	50
prteig	0	0	1	1	1	1
prtden	0	0	1	1	1	1

TABLE 1.1.1 : Variables paramétrées par accuracy.

```

1 R1 = (a,      0,      c)
2 R2 = (-a/2,  sqrt(3)/2*a, c)
3 R3 = (-a/2, -sqrt(3)/2*a, c)

```

où $a/2 + c/2 = 1.0$. Si les angles ne sont pas égaux (ou s'ils valent tous 90°), on aura la forme générique suivante :

- $R1 = (1, 0, 0)$
- $R1 = (a, b, 0)$
- $R1 = (c, d, e)$

où chaque vecteur est normalisé, et forme les angles souhaités avec les autres.

»»» Caractéristique(s) : ENERGY

ecut

Energy CUTOFF

Type de variable : réelle ;

Valeur par défaut : aucune

Dimensions : scalaire ;

Elle est utilisée pour définir l'énergie cinétique de coupure qui contrôle le nombre d'ondes planes en un k point donné. Les ondes planes permises sont celles pour lesquelles l'énergie cinétique est inférieur à $ecut$, qui se traduit par la contrainte suivante sur le vecteur \vec{G} dans l'espace réciproque :

$$\frac{1}{2}(2\pi)^2(\vec{k} + \vec{G})^2 < ecut. \quad (1.1.1)$$

Toutes les ondes planes dans cette sphère de base centrée en k sont incluses dans la base (sauf si **dilatmx** est défini). L'énergie de coupure peut être spécifiée dans les unités suivantes : Ha (par défaut, $1\text{Ha} = 27.211\,384\,5\text{ eV}$), Ry, eV ou en K.

Ce seul paramètre peut avoir un impact énorme sur la qualité des calculs ; basiquement plus `ecut` est large, meilleur est la convergence du calcul. Pour une géométrie fixée, l'énergie totale **doit** toujours décroître à mesure que `ecut` augmente à cause de la nature variationnelle du problème.

Généralement on exécute plusieurs calculs avec différentes valeur de `ecut` pour investiguer sur la convergence nécessaire pour des résultats acceptables.

Caractéristique(s) : –

einterp
Electron bands **INTER**Polation

Type de variable : réelle ;
Valeur par défaut : [0, 0, 0, 0]

Dimensions : (4) ;

Cette variable active l'interpolation des valeurs propres électroniques. Elle peut être utilisée pour interpoler KS valeurs propres à la fin d'une exécution GS ou pour interpoler GW énergies dans des calculs sigma (`optdriver` = 4). Le k-path peut être spécifié avec `kptbounds` et `nkpath`. `einterp` consiste en quatre entrées. Le premier élément spécifie la méthode d'interpolation :

- 0 → pas d'interpolation (par défaut) ;
- 1 → interpolation star-function (voir [Pickett et al., 1988]).

La signification des autres entrées dépend de la technique d'interpolation sélectionnée.

Caractéristique(s) : –

iscf
Integer for **Self-Consistent-Field**
cycles

Type de variable : entière ;
Valeur par défaut :

Dimensions : scalaire ;

17 if `usepaw` == 1, 0 if `usewvl` == 1, 7 if not

Elle contrôle l'algorithme d'auto-cohérence (self-consistency).

Les valeurs positives correspondent au choix usuel pour des calculs du ground state (GS) ou pour des relaxations structurales, où le potentiel doit être déterminé de manière auto-cohérente alors que les valeurs négatives correspondent aux calculs non auto-cohérents.

Les différentes valeurs positives (> 0) vont de 1 à 17 et les valeur négatives vont de -3 à -1.

Alerte 1.1

Toutes autres valeurs positives, y compris 0, ne sont pas permises ; de même que des valeurs négatives en dessous de -3.

Caractéristique(s) : –

ixc
Index of e**X**change-**C**orrelation
fonctional

Type de variable : entière ;
Valeur par défaut : 1

Dimensions : scalaire ;

Elle contrôle le choix d'échange et corrélation (xc). La liste des fonctionnelles xc est donnée dans la documentation en ligne (<https://docs.abinit.org/variables/basic/#ixc>). Les valeurs positives correspondent aux valeurs natives de la librairie des fonctionnelles xc de Abinit (de 0 à 42), tandis que les négatives sont celles de la librairie ETSF LibXC (pour les utiliser il faut compiler Abinit avec le plug-in LibXC).

Caractéristique(s) : NO_MULTI

jdtset
index **-J-** for **DaTaSETs**

Type de variable : entière ;
Valeur par défaut : [{"start": 1, "stop": "ndtset"}];

Dimensions : `ndtset` ;

Elle donne la base de données d'index de chaque base de données. Ces index seront utilisés pour :

- déterminer quelles variables d'entrée sont spécifiques pour chaque base de données, sachant que les noms de variables pour cette base de données seront construites à partir de nom de variable dévoilée concaténée avec cet index, et seulement si ce nom de variable composite n'existe pas, le code considérera le nom de variable dévoilé, ou même, celle par défaut ;

- caractériser des noms de variables de sortie, si leurs contenus diffèrent d'une base de données à l'autre ;
- caractériser des fichiers de sortie (noms racine joint à `_DSx` où 'x' est l'index de la base de données).

Les index permis sont compris entre 1 et 9999. Une variable d'entrée joint à 0 n'est pas permise. Lorsque `ndtset` == 0, ce tableau n'est pas utilisé, et de plus, aucun nom de variable d'entrée n'est joint à un numéro permis. Ce tableau doit être initialisé grâce à l'utilisation de la variable d'entrée `udtset`. Dans ce cas, `jdtset` ne peut être utilisée.

»»» Caractéristique(s) : -

kpt
K-PoinTs

Type de variable : réelle ; **Dimensions** : (3,**nkpt**) ;
Valeur par défaut : [0,0,0]

Elle contient les k points en termes de translations d'espace réciproque primitif (pas en coordonnées cartésiennes). Utile seulement si `kptopt` = 0, sinon déduit d'autres variables.

Elle contient des nombres sans dimension tel que les coordonnées doivent être : $k_cartesian = k_1 * G_1 + k_2 * G_2 + k_3 * G_3$ où (k_1, k_2, k_3) représentent les "coordonnées réduites" sans dimension et G_1, G_2, G_3 sont les coordonnées des vecteurs de translation primitifs. G_1, G_2, G_3 sont reliés au choix l'espace de translation primitif direct construit en `rprim`. Notons que la norme de tous les k points est fourni par `kptnrm`. Ceci permet d'éviter la fourniture de plusieurs chiffres pour les k points pour représenter ces points tels que (1, 1, 1)/3.

Remarque 1.1.1

1. L'un des algorithmes utilisés pour paramétrer la sphère des vecteurs G pour la base a besoin des composants des k points dans l'intervalle $[-1, 1]$, ainsi la création d'une nouvelle carte est facilité par ajout ou soustraction de 1 de chaque composant jusqu'à ce que cela se retrouve dans l'intervalle $[-1, 1]$. C'est-à-dire, soit le k point de normalisation `kptnrm` décrit ci-dessous, chaque composant doit se trouver dans $[-kptnrm, kptnrm]$.
2. Un décalage global peut être fourni par `qptn`, pas lu si `kptopt` /= 0.

»»» Caractéristique(s) : -

kptnrm
K-PoinTs NoRMalization

Type de variable : réelle ; **Dimensions** : scalaire ;
Valeur par défaut : 1

Elle établit un dénominateur de normalisation pour chaque k point. Utile seulement si `kptopt` <= 0, sinon est déduit d'autres variables. Les coordonnées du k point comme fractions des translations de la maille réciproque sont par conséquent `kpt(mu, ikpt) / kptnrm`. Par défaut `kptnrm` vaut 1 et peut être ignoré par l'utilisateur. Elle a été introduite pour éviter le besoin de plusieurs chiffres dans la représentation des nombres tels que 1/3. Elle ne peut pas être plus petite que 1.0.

»»» Caractéristique(s) : -

kptopt
KPoinT OPTion

Type de variable : entière ; **Dimensions** : scalaire ;
Valeur par défaut : 4 **if** `nspden` == 4, 1 **if** **not**

Elle contrôle le paramétrage de la liste des k points. L'objectif sera d'initialiser, par lecture directe ou par une approche par prétraitement basée sur les variables d'entrée, les variables d'entrée suivantes, donnant les k points, leur nombre, et leur poids : `kpt`, `kptnrm`, `nkpt` et, pour `iscf` /= 2, `wtk`.

L'utilisation des symétries (spatiale et/ou inversion du temps) est cruciale pour déterminer l'action de `kptopt`. Les valeurs prises par `kptopt` sont 0, 1, 2, 3, 4 et des valeurs négatives.

»»» Caractéristique(s) : -

natom
Number of ATOMs

Type de variable : entière ; **Dimensions** : scalaire ;
Valeur par défaut : 1

Elle donne le nombre total d'atomes dans une cellule unitaire. La valeur par défaut est 1, mais nous devons inclure cette valeur explicitement. Notons que `natom` se réfère à tous les atomes d'une cellule unitaire, pas seulement au système irréductible d'atomes dans une cellule unitaire (utilisant des opérations de symétrie, ce système permet de retrouver tous les atomes). Si nous voulons spécifier seulement le système d'atomes irréductible, il faut utiliser le symétriseur (voir la variable d'entrée `natrd`).

»»» Caractéristique(s) : –

nband Number of BANDs	Type de variable : entière ; Valeur par défaut : aucune	Dimensions : scalaire ;
--	--	--------------------------------

Elle donne le nombre de bandes, occupées plus possiblement celles inoccupées, pour lesquelles des fonctions d'onde ont été programmées avec des valeurs propres.

Remarque 1.1.2

Si le paramètre `occopt` (voir plus loin) n'est pas paramétré à 2, `nband` est un scalaire entier, mais si le paramètre `occopt` est paramétré à 2, alors `nband` doit être un tableau `nband(nkpt*nsppol)` donnant explicitement le nombre de bandes de chaque k point. Cette option est fournie pour permettre que le nombre de bandes traité varie d'un k point à un autre. Pour les valeurs de `occopt` non égales à 0 ou 2, `nband` peut être omis. Le nombre de bandes sera paramétré grâce à l'utilisation de la variable `fband`. La valeur par défaut présente ne sera pas utilisée.

Si `nspinor` est 2, `nband` devra être pour tous les k points.

Dans le cas d'un calcul GW (`optdriver` = 3 or 4), `nband` donne le nombre de bandes à traiter pour générer criblage (susceptibilité et matrice diélectrique), aussi bien que l'énergie propre. Cependant, pour générer le fichier `_KSS` (voir `kssform`) le nombre de bandes important est donné par `nbandkss`.

»»» Caractéristique(s) : –

nbandhf Number of BANDs for (Hartree)-Fock exact exchange	Type de variable : entière ; Valeur par défaut : aucune	Dimensions : scalaire ;
---	--	--------------------------------

Elle donne le nombre maximum de bandes occupées avec lesquelles l'échange exact a été calculé pour les fonctions d'onde.

»»» Caractéristique(s) : NO_MULTI

ndtset Number of DaTaSETs	Type de variable : entière ; Valeur par défaut : 0	Dimensions : scalaire ;
--	---	--------------------------------

Elle donne le nombre de jeux de données à traiter. Si 0, cela signifie que le traitement du jeu de multi-données n'est pas utilisé, donc les noms des fichiers racine ne seront pas joint à `_DSx` où 'x' est l'index de la base de données défini par la variable d'entrée `jdtset`, et aussi ces noms d'entrée avec un index de base de données ne sont pas permis. Sinon, `ndtset` = 0 est équivalent à `ndtset` = 1.

»»» Caractéristique(s) : INPUT_ONLY

ngkpt Number of Grid points for K PoinTs generation	Type de variable : entière ; Valeur par défaut : [0,0,0]	Dimensions : 3 ;
---	---	-------------------------

Elle est utilisée quand `kptopt` >= 0, si `kptrlatt` n'est pas défini (`kptrlatt` et `ngkpt` s'excluent mutuellement). Ses trois composantes positives donnent le nombre de k points des grilles de Monkhorst-Pack (défini en respect avec l'axe primitif dans l'espace réciproque) dans chacune des trois dimensions. `ngkpt` doit être utilisé pour générer la variable d'entrée `kptrlatt` correspondante. L'utilisation de `nshiftk` et `shiftk` permet de générer des grilles décalées, ou des grilles de Monkhorst-Pack défini en respectant des cellules unitaires conventionnelles.

Lorsque `nshiftk` = 1, `kptrlatt` est initialisé comme une matrice (3 × 3), dont les éléments de la diagonale sont les trois valeurs `ngkpt` (1:3). Lorsque `nshiftk` est plus grand que 1, Abinit essaiera

de générer `kptrlatt` dans la base des vecteurs primitifs des k mailles : le nombre de décalage doit être réduit, au quel cas `kptrlatt` ne sera plus diagonal.

Des grilles de Monkhorst-Pack sont généralement les plus efficaces quand leurs entiers de définition sont pairs. Pour une mesure de l'efficacité, voir la variable d'entrée `kptrlen`.

»»» Caractéristique(s) : –		
nkpath Number of K -points defining the PATH	Type de variable : entière ; Valeur par défaut : 0	Dimensions : scalaire ;

Cette variable est utilisée pour définir le nombre de k points de haute symétrie dans le tableau `kptbounds` quand `kptopt` > 0. Historiquement, `kptbounds` est utilisé en conjonction avec une valeur négative de `kptopt` quand un calcul de structure de bande NSCF est fait. Dans ce cas, le nombre de k points dans `kptbounds` est donné par `abs(kptbounds)+1`. Il y a cependant, d'autres cas dans lesquels on a à spécifier un chemin k dans le fichier d'entrée dans le but d'activer certains outils de post-traitement.

Remarque 1.1.3

Contrairement à `kptopt`, `nkpath` représente le nombre total de points dans le tableau `kptbounds`.

»»» Caractéristique(s) : –		
nkpt Number of K -Points	Type de variable : entière ; Valeur par défaut : 1 if <code>kptopt</code> == 0, 0 if not	Dimensions : scalaire ;

Si différente de zéro, `nkpt` donne le nombre de k points dans le tableau k point `kpt`. Ces points sont utilisés soit comme échantillon de la zone de Brillouin, ou pour construire une structure de bande avec des lignes spécifiques.

Si `kptopt` est positive, `nkpt` doit être cohérent avec les valeurs `kptrlatt`, `nshiftk` et `shiftk`. Pour des calculs du ground state, on doit sélectionner le k point dans la zone irréductible de Brillouin (obtenue en tenant compte des symétrie de point et de la symétrie d'inversion du temps). Pour une réponse de fonction des calculs, on doit sélectionner k points dans la zone de Brillouin totale, si le vecteur d'onde de la perturbation ne s'évanouit pas, ou si dans une moitié de la zone de Brillouin $q = 0$. Le code décrémentera automatiquement le nombre de k points à l'ensemble minimal pour chaque perturbation particulière.

Si `kptopt` est négative, `nkpt` sera la somme du nombre des points sur les différentes lignes de la structure de bande. Par exemple, si `kptopt` = - 3, on aura trois segments ; en supposant que `ndivk` est [10, 12, 17], le nombre total de k points du circuit sera 10 + 12 + 17 + 1 (pour le point final) = 40.

»»» Caractéristique(s) : –		
nkpthf Number of K -Points for (H artree) F ock exact exchange	Type de variable : entière ; Valeur par défaut : aucune	Dimensions : scalaire ;

Elle donne le nombre de k points utilisés pour échantillonner la zone de Brillouin totale pour la contribution de l'échange exact de Fock. Elle s'obtient à partir de la spécification de la fonction d'onde de la grille de k point (voir `kptopt`), éventuellement replié comme spécifié par `fockdownsampling`.

»»» Caractéristique(s) : –		
nshiftk Number of SHIFT s for K point grids	Type de variable : entière ; Valeur par défaut : 1	Dimensions : scalaire ;

Ce paramètre donne le nombre de grilles décalées qui doivent être actuellement utilisées pour générer la grille totale des k points. Il peut être utilisé avec des grilles primitives définis soit de `ngkpt` ou de `kptrlatt`. La valeur maximale permise de `nshiftk` est 8. Les valeurs des décalages sont données par `shiftk`.

L'utilisation de `nshiftk = 1, 2, or 4` est très commune, voir les valeurs suggérées dans la description de `shiftk`. Les autres valeurs servent au débogage par des experts, ou peuvent indiquer une erreur. Ces autres valeurs ne sont permises que si `chksymbreak = 0`.

»»» Caractéristique(s) : –		
nsppol Number of SP in POL arization	Type de variable : entière ; Valeur par défaut : 1	Dimensions : scalaire ;

Elle donne nombre de polarisations de spin **indépendants**, pour lesquels il n'y a pas de fonctions d'onde relative. Elle peut prendre les valeurs 1 ou 2.

»»» Caractéristique(s) : –		
nstep Number of (non-)self-consistent field STEP s	Type de variable : entière ; Valeur par défaut : 30	Dimensions : scalaire ;

Elle donne le nombre maximum de cycles (ou “itérations”) dans une exécution SCF ou non-SCF. La convergence totale pour des nombres aléatoire est généralement atteinte en 12-20 itérations SCF. Chacune pouvant aller de quelques minutes à des heures. Dans certains cas difficile, souvent reliés à une bande de gap nulle ou au magnétisme, la performance de convergence peut être mauvaise. Quand la tolérance de convergence `tolwfr` sur les fonctions d'onde est satisfaite, des itérations s'arrêteront, donc pour de bons calculs convergents nous devons paramétrer `nstep` à une valeur plus grande que nous pensons avoir besoin pour une convergence complète. Exemple, si l'utilisation de 20 étapes fait généralement converger le système, paramétrez `nstep` à 30. Pour des exécutions non auto-cohérents (`iscf < 0`) `nstep` gouverne le nombre de cycles de convergence pour les fonctions d'onde pour une densité fixée et un hamiltonien.

»»» Caractéristique(s) : –		
nsym Number of SYM metry operations	Type de variable : entière ; Valeur par défaut : 0	Dimensions : scalaire ;

Elle donne le nombre de symétries de groupe d'espace à appliquer dans ce problème. Des symétries seront insérées dans le tableau `symrel` et des vecteurs de translation (nonsymmorphic) seront insérés dans le tableau `trons`. S'il n'y a pas de symétrie dans le problème alors paramétrez `nsym` à 1, parce que l'identité est toujours une symétrie. Dans le cas d'un calcul RF, le code est capable d'utiliser les symétries du système pour diminuer le nombre de perturbations à calculer, et pour diminuer le nombre de k points spéciaux à utiliser pour l'échantillonnage de la zone de Brillouin. Après que la réponse aux perturbations ait été calculé, les symétries sont utilisées pour générer au tant que possible d'éléments de 2DTE de ceux déjà calculés.

»»» Caractéristique(s) : NO_MULTI		
ntypat Number of TY pes of AT oms	Type de variable : entière ; Valeur par défaut : 1	Dimensions : scalaire ;

Elle donne le nombre de types d'atomes dans une cellule unitaire. Par exemple pour un système homopolaire (e.g. Si pure) `ntypat` est 1.

Le code essaye de lire le même nombre dans les fichiers de pseudopotentiel. Le premier pseudopotentiel est assigné au numéro de type 1, ainsi de suite.

Il y a une exception dans le cas d'un mélange alchimique de potentiels. Dans ce cas, le nombre de types atomiques différera du nombre de pseudopotentiels. Voir `mixa1ch`, `npssp`, `ntypalch` et `npspalch`.

»»» Caractéristique(s) : –		
occopt OCC upation OPT ion	Type de variable : entière ; Valeur par défaut : 1	Dimensions : scalaire ;

Elle contrôle comment les paramètres d'entrée `nband`, `occ` et `wtk` sont manipulés. Les valeurs possibles vont de 0 à 9. Pour des matériaux avec gap (semi-conducteurs, molécules, ...), `occopt = 1` est le choix favori pour la plupart des usages. Pour des situations métallique (aussi des molécules avec niveaux dégénéré à l'énergie de Fermi), `occopt = 7` est le choix favori pour la plupart des usages,

et on doit aussi contrôler la variable d'entrée **tsmear**. Utiliser `occopt = 9` pour des calculs d'énergie quasi-fermi pour les états excités dans les matériaux à gap (pour les autres valeurs voir <https://docs.abinit.org/variables/basic/#occopt>).

»»» Caractéristique(s) : EVOLVING

rprim
Real space **PRIM**itive translations

Type de variable : réelle ; **Dimensions** : (3, 3);
Valeur par défaut : `[[1,0,0], [0,1,0], [0,0,1]]`

Elle donne les trois translations primitives sans dimension dans l'espace réel, qui doivent être re-dimensionnées par **acell** et **scalecart**. Les trois premiers nombres sont les coordonnées du premier vecteur, les suivantes celles du second et les trois dernières celle du troisième. C'est un **EVOLVING** seulement si **ionmov** == 2 or 22 et **optcell** /= 0, sinon elle est fixe.

Si la valeur par défaut est utilisée, c'est-à-dire que **rprim** est une matrice unitaire, les trois vecteurs primitifs sans dimension sont trois vecteurs unitaires en coordonnées cartésiennes. Les coordonnées (et par conséquent la longueur) de chaque vecteurs seront (possiblement) multipliées par la valeur de **acell** correspondante, alors (possiblement) elles s'étendraient en direction des coordonnées cartésiennes par la valeur correspondante **scalecart**, pour donner les vecteurs primitifs dimensionnés, appelé **rprim**.

Dans le cas général, les coordonnées cartésiennes dimensionnelles des translations du cristal primitif **R1p**, **R2p** et **R3p**, voir **rprimd**, sont :

- $R1p(i) = scalecart(i) \times rprim(i,1) \times acell(1)$
- $R2p(i) = scalecart(i) \times rprim(i,2) \times acell(2)$
- $R3p(i) = scalecart(i) \times rprim(i,3) \times acell(3)$

où ($i = 1,2,3$) est la composante de la translation primitive (i.e x , y et z).

La variable **rprim**, appelée par **scalecart**, est alors utilisée pour définir des directions des vecteurs primitifs, qui seront multipliés (tout en gardant la direction inchangée) par la longueur appropriée **acell(1)**, **acell(2)**, ou **acell(3)**, respectivement pour donner des translations primitives dimensionnelles dans l'espace réel en coordonnées cartésiennes. Présentement, il est requis que le produit mixte $(R1 \wedge R2) \cdot R3$ soit positif. Si ce n'est pas le cas, simplement échanger une pair de vecteurs.

Pour être plus spécifique, en laissant la valeur de **scalecart** = 1 pour simplifier les chose, **rprim 1 2 3 4 5 6 7 8 9** correspond à l'entrée des trois translations primitives $R1 = (1,2,3)$ (à multiplier par **acell(1)**), $R2 = (4,5,6)$ (à multiplier par **acell(2)**) et $R3 = (7,8,9)$ (à multiplier par **acell(3)**). Remarquons attentivement que les trois premiers nombres représentent la première colonne de **rprim**, les trois suivants la seconde, et les trois derniers la troisième. Ceci correspond à l'ordre usuel des tableaux de Fortran. La matrice dont les colonnes sont des translations primitives de l'espace réciproque est la transposée inverse de la matrice dont les colonnes sont des translations primitives de l'espace direct.

Alternativement à **rprim**, des directions des vecteurs primitifs sans dimension peuvent être spécifié en utilisant la variable d'entrée **angdeg**. Ceci est spécialement utile pour des mailles hexagonales (avec des angles de 120° ou 60°). En fait, dans le but que des symétries soient reconnues, **rprim** doit être symétrique jusqu'à **tolsym** (10^{-5} par défaut), incluant une spécification telle que

```
1 rprim  0.86602  0.5  0.0
2      -0.86602  0.5  0.0
3         0.0    0.0  1.0
```

ce qui peut être évité grâce à **angdeg** :

```
1 angdeg 90 90 120
```

Remarquons que ce qui suit doit aussi bien fonctionner :

```

1 rprim  sqrt(0.75)  0.5  0.0
2        -sqrt(0.75)  0.5  0.0
3          0.0        0.0  1.0

```

Bien que l'utilisation de **scalecart** ou **acell** soit plutôt équivalent quand les vecteurs primitifs sont alignés avec les directions cartésiennes, ce n'est pas le cas pour de vecteurs primitifs non-orthogonaux. En particulier, des débutants font l'erreur d'essayer d'utiliser **acell** pour définir des vecteurs primitifs dans une maille à face centrée tétragonale, ou une maille à corps centrée tétragonale, ou similairement dans des mailles à face ou corps centrée orthorhombique. Prenons l'exemple d'une maille à corps centrée tétragonale, qui soit être défini en utilisant ce qui suit ("**a**" et "**c**" doivent être remplacés par la longueur appropriée du vecteur de cellule conventionnel) :

```

1 rprim  "a"      0      0
2        0      "a"    0
3        "a/2"   "a/2"  "c/2"
4 acell 3*1      scalecart 3*1  ! (Celles-ci sont les valeurs par défaut)

```

Ce qui suit est un un chemin alternatif valide pour définir les mêmes vecteurs primitifs :

```

1 rprim  1      0      0
2        0      1      0
3        1/2    1/2    1/2
4 scalecart "a"  "a"  "c"
5 acell 3*1  ! (Celles-ci sont les valeurs par défaut)

```

En fait, la cellule a été étendue le long des coordonnées cartésiennes, par des facteurs "**a**", "**a**" et "**c**".

Une variante, comme ce qui suit est **fausse** :

```

1 rprim  1      0      0
2        0      1      0
3        1/2    1/2    1/2
4 acell  "a"  "a"  "c"  ! CECI EST FAUT
5 scalecart 3*1  ! (Celles-ci sont les valeurs par défaut)

```

En fait, cette dernière correspondrait à :

```

1 rprim  "a"      0      0
2        0      "a"    0
3        "c/2"   "c/2"  "c/2"
4 acell 3*1      scalecart 3*1  ! (Celles-ci sont les valeurs par défaut)

```

C'est-à-dire, le troisième vecteur a été redimensionné par "**c**". Il n'est pas du tout au centre de la cellule tétragonale dont des vecteurs de bases sont défini par le facteur d'échelle "**a**". Comme autre différence entre **scalecart** et **acell** (notons que **scalecart** est un **INPUT_ONLY**) est que son contenu sera immédiatement appliquer à **rprim**, au moment de l'analyse, et ainsi **scalecart** sera paramétrée aux valeurs par défaut (3*1). Donc, dans ce cas **scalecart** est utilisée, la répétition de **rprim** dans le fichier de sortie n'est pas la valeur contenue dans le fichier d'entrée, mais celle redimensionnée par **scalecart**.

>>> Caractéristique(s) : INTERNAL_ONLY, EVOLVING

rprimdReal space **PRIM**itive translations, **D**imensional**Type de variable** : réelle;
Valeur par défaut : aucune**Dimensions** : (3, 3);

Cette variable interne donne les vecteurs primitifs de l'espace réel dimensionnel, calculés de **acell**, **scalecart** et **rprim**.

- $R1p(i) = rprimd(i,1) = scalecart(i) \otimes rprim(i,1) \otimes acell(1)$
- $R2p(i) = rprimd(i,2) = scalecart(i) \otimes rprim(i,2) \otimes acell(2)$
- $R3p(i) = rprimd(i,3) = scalecart(i) \otimes rprim(i,3) \otimes acell(3)$

où (i = 1,2,3) est la composante de la translation primitive (i.e x, y et z).

C'est un **EVOLVING** seulement si **ionmov** == 2 **or** 22 et **optcell** /= 0, sinon elle est fixe.

>>> Caractéristique(s) : INPUT_ONLY

scalecart**SCALE CART**esian coordinates**Type de variable** : réelle;
Valeur par défaut : 3*1**Dimensions** : (3);

Elle donne les facteurs d'échelle des coordonnées cartésiennes par lesquelles des translations primitives sans dimension (dans **rprim**) doivent être multipliés. Elle est spécialement utile pour des mailles à corps centrée et à face centrée tétraogonales, aussi bien que pour des mailles à corps centrée et à face centrée orthorhombiques (voir **rprimd**). Notons que **scalecart** est un **INPUT_ONLY** : son contenu sera immédiatement appliquer à **rprim**, au moment de l'analyse, et ainsi **scalecart** sera paramétrée aux valeurs par défaut. Donc, il ne sera pas répété.

>>> Caractéristique(s) : -

shiftk**SHIFT** for **K** points**Type de variable** : réelle;
Valeur par défaut : 3*1**Dimensions** : (3,**nshiftk**);

Elle est utilisée unique quand **kptopt** >= 0, et doit être défini si **nshiftk** est plus grand que 1. **shift(1:3,1:nshiftk)** définit des décalages **nshiftk** de la grille homogène des **k** points basés sur **ngkpt** ou **kptrlatt**. Les décalages induits par **shiftk** correspondent aux coordonnées réduites dans le système de coordonnées de la maille **k** point. Par exemple, si la maille **k** point est défini en utilisant **ngkpt**, le point dont les coordonnées de l'espace réciproque réduit sont (**shiftk(1,ii)/ngkpt(1)**, **shiftk(2,ii)/ngkpt(2)**, **shiftk(3,ii)/ngkpt(3)**) appartient au nombre de grille décalé **ii**.

>>> Caractéristique(s) : -

structureinitialize the crystalline **STRUC-**
TURE from ...**Type de variable** : chaîne de caractères;
scalaire;
Valeur par défaut : aucune**Dimensions** :

Cette variable fourni une interface simplifiée pour construire une structure cristalline à partir d'un fichier externe. L'idée est de laisser séparé l'**information de géométrie** séparée du fichier d'entrée tel qu'on puisse faire plusieurs calculs avec différents fichiers d'entrée partageant la même structure sans faire un copier-coller de la description des cellule unitaire dans le fichier d'entré. La source unique de vérité est donnée par un fichier externe qui peut facilement être distribué. Comme effet secondaire, on peut facilement redémarrer les relaxations de structure en place en lisant la structure à partir du fichier de sortie d'une exécution précédente.

La variable **structure** est une chaîne de caractères au format **typeFichier:cheminFichier** où :

- **typeFichier** spécifie le format du fichier externe;
- **cheminFichier** donne le chemin vers le fichier *relatif* au répertoire où est localisé le fichier d'entré.

Des variables telles que **natom**, **ntypat**, **typat** et **znucl** sont automatiquement initialisées à partir du fichier externe et n'ont pas besoin d'être spécifiées dans l'entrée Abinit.

À cette date (19 janvier 2023), les valeurs de `typeFichier` permises sont :

- **abifile** : un fichier de sorti produit par Abinit (seuls les fichiers `netcdf` sont supportés en ce moment);
- **abivars** : un fichier texte (`.txt`) d'entrée avec des variables Abinit;
- **poscar** : des fichiers POSCAR au format VASP-5 (le symbole d'élément après la position atomique est requis).

Certains exemples aident à clarifier la compréhension.

Pour lire la structure à partir d'un fichier externe `netcdf` produit par Abinit (par exemple `out_GSR.nc`) utiliser le préfixe **abifile** et la syntax :

```
1 structure "abifile:out_GSR.nc"
```

D'autres fichiers de sortie de Abinit tels que le `WFK.nc`, le `DEN.nc` et le `HIST.nc` sont également bien supportés par exemple.

```
1 structure "abifile:out_HIST.nc"
```

Dans le cas des relaxations de structure, ces fichiers contiennent la géométrie finale (pas nécessairement relaxée dans la tolérance donnée) par conséquent `structure` peut être utilisé pour effectuer un redémarrage sur place en lisant la sortie d'une exécution précédente.

Pour lire la structure à partir d'un fichier externe avec une structure au format Abinit, utiliser :

```
1 structure "abivar:mon_fichier_texte"
```

où `mon_fichier_texte` spécifie une maille en termes de `[[acell], (rprim or angdeg)]` alors que les positions atomiques sont spécifiées avec **natom** et la variable spéciale `xred_symbols` qui peut être utilisée seulement dans ce fichier externe.

```
1 # Structure de maille du MgB2
2
3 acell    2*3.086  3.523 Angstrom
4
5 rprim    0.866025403784439  0.5  0.0
6          -0.866025403784439  0.5  0.0
7          0.0                0.0  1.0
8
9 natom    3
10
11 # Positions réduites suivies par les symboles des éléments.
12
13 xred_symbols
14 0.0 0.0 0.0 Mg
15 1/3 2/3 0.5 B
16 2/3 1/3 0.5 B
```

Pour lire la structure à partir du fichier POSCAR externe, utiliser :

```
1 structure "poscar:t04_POSCAR"
```

Un fichier POSCAR typique pour le MgB₂ ressemble à ceci (ignorez les commentaires) :

```
1 # Titre (ignoré par Abinit)
2
3 Mg1 B2
4
5 # Facteur d'échelle pour des vecteurs de mailles.
6
7 1.0
8
9 # Vecteurs de maille en Angstrom (Rappel: Abinit utilise le BOHR par défaut)
10
11 2.672554 1.543000 0.000000
12 -2.672554 1.543000 0.000000
13 0.000000 0.000000 3.523000
14
15 # Liste des symboles des éléments
16
17 Mg B
18
19 # Nombre d'atomes pour chaque symbole
20
21 1 2
22
23 # "direct" pour les coordonnées réduites ou "cartesian".
24
25 direct
26
27 # Coordonnées suivie par le symbole chimique de l'atome.
28
29 0.000000 0.000000 0.0 Mg
30 0.333333 0.666667 0.5 B
31 0.666667 0.333333 0.5 B
```

Un facteur d'échelle positif peut être utilisé pour redimensionner les vecteurs de maille, alors qu'une valeur négative est interprétée comme le volume d'une unité cellulaire en Å³. La valeur 0 n'est pas permise.

La variable **typat** est automatiquement initialisée à partir de la liste des symboles chimiques en accord avec leurs positions. Dans cet exemple, Mg est de type 1, tandis que B est de type 2.

Les variables Abinit associées à ce POSCAR sont donc :

```
1 ntypat 2
2 typat 1 2 2
3 znuc1 12.0 5.0
```

Ce sont les *seules quantités* qui sont initialisées à partir d'un POSCAR externe, donc nous devons nous assurer que notre POSCAR ressemble à l'exemple donné ci-dessus et ne pas espérer qu'Abinit comprenne d'autres entrées telles que Selective dynamics ou velocities.

Remarque 1.1.4**IMPORTANT**

1. La structure est initialisée par un analyseur au tout début du calcul, donc les fichiers externes doivent exister quand Abinit commence à analyser le fichier d'entrée. Pour résumer, les variables structure ne peuvent pas être utilisées pour faire passer la géométrie de sortie d'une base de données à la suivante.
2. Des bases de données multiples sont supportées mais gardez à l'esprit que certaines variables telles que **ntypat**, **typat** et **znuc1** sont taguées comme **NO_MULTI**. En d'autres termes, on peut lire différents fichiers via structure et la syntaxe des bases de données multiples fournie par ces quantités ne change pas. Des syntaxes Abinit telle que **xred+** sont, évidemment, non supportées.
3. Les valeurs de **typat** et **znuc1** données dans le fichier d'entrée (s'il y en a) sont ignorées par l'analyseur. La valeur de **natom**, **ntypat** est contrôlée pour l'uniformité.
4. Comme règle d'or, ne mélangez pas les deux approches : soit vous utilisez structure ou l'approche *standard* (plus verbeuse) basée sur **ntypat**, **typat** et **znuc1** pour définir une cellule unitaire.

Limitations

1. La spécification des structures pour des calculs avec des images n'est pas supporté.
2. Le mélange alchimique n'est pas supporté.
3. La lecture de structure à partir d'un fichier Fortran n'est pas encore implémenté. C'est juste un problème technique que les concepteurs de Abinit espèrent résoudre dans les prochaine version.

Dans tous les cas où la variable structure n'est pas supportée, on doit recourir à l'approche standard pour définir la liste d'atomes et leurs types.

»»» Caractéristique(s) : -

symrel**SYM**metry in **REaL** space

Type de variable : entière ; **Dimensions** : (3,3,**nsym**);
Valeur par défaut :
 [[1,0,0],[0,1,0],[0,0,1]] **if** nsym == 1,
 aucune **if not**

Elle donne des matrices 3×3 **nsym** exprimant des symétries de groupe d'espace en termes de leur action sur l'espace direct (ou réel) des translations primitives. Il s'avère que ceux-ci peuvent toujours être exprimés sous forme de nombres entiers. Toujours donner la matrice identité même si d'autres symétries sont prises, par exemple **symrel** 1 0 0 0 1 0 0 0 1.

»»» Caractéristique(s) : -

tnons**T**ranslation **NON-S**ymmorphic
vectors

Type de variable : réelle ; **Dimensions** : (3,**nsym**);
Valeur par défaut : aucune

Elle donne les vecteurs de translation (non symmorphiques) associés avec les symétries exprimés dans **symrel**. Ceux-ci doivent être tous 0, ou doivent être des translations fractionnelles (non primitive) exprimées relativement à l'espace réel des translations primitives (donc, utilisant le système de coordonnées réduit, voir **xred**). Si tous les éléments du groupe d'espace laissent invariant 0 0 0, alors ils sont tous 0. Lorsque le chercheur de symétrie est utilisé (voir **nsym**), **tnons** est calculée automatiquement.

Pour les drivers ground-state et DFPT de Abinit, la valeur de **tnons** n'est pas restreinte. Cependant,

pour GW et BSE, les opérations de symétrie doivent laisser la grille FFT invariante. Des exécutions préparatoire (état fondamental) doivent aussi utiliser la même géométrie atomique, par conséquent la même valeur de `tnons`. Puisque Abinit ignore ce si l'utilisateur fera un GW ou un BSE après une exécution GS, une approche conservative est implémentée, nécessitant une telle correspondance des opérations de symétrie et de la grille FFT également dans le cas GS.

»»» Caractéristique(s) : ENERGY

toldfe**TOLerance on the DiFference of total E**nergy**Type de variable** : réelle ;
Valeur par défaut : 0.0**Dimensions** : scalaire ;

Elle paramètre la tolérance des différences absolues de l'énergie totale qui, atteintes deux fois successivement, causeront l'arrêt d'un cycle SCF (et le retrait d'ions). Elle peut être spécifiée en Ha (par défaut), Ry, eV ou K puisque `tolfe` a la caractéristique **ENERGY**. Si elle est paramétrée à 0, cette condition d'arrêt est ignorée. Efficace uniquement lorsque les cycle SCF sont terminés (`iscf` > 0). En raison de la précision de la machine, il ne vaut pas la peine d'essayer d'obtenir des différences d'énergie inférieures à environ 1.0×10^{-12} de l'énergie totale.

»»» Caractéristique(s) : -

toldff**TOLerance on the DiFference of F**orces**Type de variable** : réelle ;
Valeur par défaut : 0.0**Dimensions** : scalaire ;

Elle paramètre une tolérance pour des différences de forces (en Ha/Bohr) qui, atteintes deux fois successivement, causeront l'arrêt d'un cycle SCF (et le retrait d'ions). Elle est effective seulement quand des cycles SCF sont terminés (`iscf` > 0). Cette tolérance s'applique à n'importe quel composant cartésien particulier de n'importe quel atome, y compris les composants fixes.

»»» Caractéristique(s) : -

tolvrs**TOLerance on the potential V(r) R**esidual**Type de variable** : réelle ;
Valeur par défaut : 0.0**Dimensions** : scalaire ;

Elle paramètre une tolérance pour le potentiel résiduel qui, lorsqu'atteinte causera l'arrêt d'un cycle SCF (et le retrait d'ion). Si paramétré à zéro, cette condition d'arrêt est ignorée. Efficace uniquement lorsque les cycle SCF sont terminés (`iscf` > 0). Obtenir des contraintes précises peut être assez exigeant. Pour des matériaux simples avec des positions internes déterminées par des symétries, une valeur de `tolvrs` = 10^{-12} conduit empiriquement à une précision très approximative de 10^{-6} unités atomiques pour le paramètre de réseau optimisé.

»»» Caractéristique(s) : -

tolwfr**TOLerance on W**aveFunction squared **R**esidual**Type de variable** : réelle ;
Valeur par défaut : 0.0**Dimensions** : scalaire ;

La signification de cette tolérance dépend de l'ensemble de base. Dans des ondes planes, elle donne la tolérance pour le plus grand résiduel carré (défini ci-dessous) pour n'importe quel bande. Le résiduel carré est :

$$\langle nk | (H - \epsilon_{nk})^2 | nk \rangle, \text{ avec } \epsilon_{nk} = \langle nk | H | nk \rangle, \quad (1.1.2)$$

qui est clairement non négatif et va vers 0 lorsque les itérations convergent vers un état propre. Avec le résiduel carré exprimé en Ha², le plus grand résiduel carré (appelé `residm` dans le code) rencontré sur toutes les bandes et les k points doit être plus petit que `tolwfr` pour que des itérations s'arrêtent à cause d'une convergence réussit.

»»» Caractéristique(s) : -

typat**TYPE of A**Toms**Type de variable** : entière ;
Dimensions : [3,`natrd`] si `natrd` < `natom`, [3,`natom`] sinon ;
Valeur par défaut : 1 if `natom` == 1, aucune sinon

C'est un tableau donnant un label entier à chaque atome dans une cellule unitaire pour symboliser leurs types. Les différents types d'atomes sont construits à partir de fichiers de pseudopotentiel. Il y a au plus **ntypat** types d'atomes. Comme exemple, pour le BaTiO_3 , où le pseudopotentiel de Ba est le nombre 1, celui de Ti est le nombre 2, et celui de O est le nombre 3, la valeur actuelle du tableau **typat** doit être :

```
1 typat 1 2 3 3 3
```

Le tableau **typat** doit être en accord avec les locations actuelles des atomes données dans **xred** ou **xcart**, et l'entrée des pseudopotentiels doit être ordonnée conformément avec les atomes identifiés dans **typat**. La charge nucléaire des éléments, donnée par le tableau **znucl**, doit aussi être conforme avec le type d'atomes désigné dans **typat**. Le tableau **typat** n'est pas contraint d'être écrit de manière croissante, donc

```
1 typat 3 3 2 3 1
```

est permis. Une représentation interne de la liste d'atomes, profondément dans le code (tableau **atindx**), groupe les atomes de même type ensemble. Cela doit être transparent pour l'utilisateur, tout en gardant l'efficacité.

»»» Caractéristique(s) : –

udtset

Upper limit on a **DaTa SETs**

Type de variable : entière ;

Valeur par défaut : aucune

Dimensions : (2) ;

Utilisée pour définir l'ensemble des indices dans le mode de multi-base de données, quand une double boucle est nécessaire (voir plus loin). Les valeurs de **udtset**(1) doivent être entre 1 et 999, les valeurs de **udtset**(2) doivent être entre 1 et 9, leur produit doit être égal à **ndtset**(2). Les valeurs de **jdtset** sont obtenues faisant des boucles sur les deux indices définis par **udtset**(1) et **udtset**(2) comme ci-dessous :

```
1 do i1=1,intarr(1)
2 do i2=1,intarr(2)
3   idtset=idtset+1
4   dtsets(idtset)%jdtset=i1*10+i2
5 end do
6 end do
```

Donc, **udtset**(2) paramètre la plus grande valeur pour le chiffre de l'unité, qui varie entre 1 et **udtset**(2). Si **udtset** est utilisée, la variable d'entrée **jdtset** ne peut pas être utilisée.

»»» Caractéristique(s) : –

usewvl

Use **WaVeLet** basis set

Type de variable : entière ;

Valeur par défaut : 0

Dimensions : scalaire ;

Utilisée pour définir si le calcul est fini sur une base d'ondelette paramétrée ou non. Les valeurs de **usewvl** doit être 0 ou 1. En mettant **usewvl** à 1, rend **icoulomb** obligatoire à 1. Le nombre de bandes (**nband**) doit être paramétré manuellement au stricte nombre nécessaire pour un système isolateur (c-à-d nombre d'électrons sur deux). La coupure n'est pas pertinente dans le cas des ondelettes, utiliser **wvl_hgrid** à la place. Dans le cas d'ondelette, le système doit être des systèmes isolés (molécules ou clusters). Toutes les optimisations de géométrie sont disponibles (voir **ionmov**, spécialement l'optimisation de géométrie et les dynamiques moléculaires). Le calcul de spin n'est présentement pas possible avec des ondelettes et des systèmes métalliques peuvent être lents à converger.

»»» Caractéristique(s) : -		
wtk Weigh T s for K points	Type de variable : réelle ; Valeur par défaut : nkpt *1.0	Dimensions : (nkpt) ;

Elle donne les poids de k point. Les poids de k point auront leur somme (re)normalisée à 1 (à moins que **occopt** = 2 et **kptopt** = 0 ; voir la description de **occopt**) dans le programme et peut donc être entré avec n'importe quelle normalisation arbitraire. Cette fonctionnalité permet d'éviter d'avoir à utiliser de nombreux chiffres pour représenter des poids fractionnaires tels que 1/3. **wtf** est ignorée si **iscf** est positive, excepté si **iscf** = -3.

»»» Caractéristique(s) : LENGTH		
wvl_hgrid WaVeLet H step GRID	Type de variable : réelle ; Valeur par défaut : 0.5	Dimensions : scalaire ;

Elle donne la taille du pas dans l'espace réel pour la résolution de grille dans l'ensemble de la base des ondelettes. Cette valeur est grandement responsable de l'occupation de mémoire dans le calcul d'ondelette. La valeur est une longueur en unités atomiques.

»»» Caractéristique(s) : EVOLVING, LENGTH		
xcart vectors X of atom position in CART esian coordinates	Type de variable : réelle ; (3,min(natom , natrd)) ; Valeur par défaut : aucune	Dimensions :

Elle donne les coordonnées cartésiennes des atomes dans les cellules unitaires. Cette information est redondante avec celle fourni par le tableau **xred**. Par défaut, **xcart** est donné en unités atomiques de Bohr (1 Bohr = 0.529 177 210 8 Å), bien que l'angstrom puisse être spécifié, si préféré, sachant que **xcart** a les caractéristique **LENGTH**. Si **xred** est absent du fichier d'entrée et que **xcart** est fourni, alors la valeur de **xred** sera calculée à partir de **xcart** fourni (c-à-d l'utilisateur doit utiliser **xcart** au lieu de **xred** pour fournir les coordonnées de départ). Une et une seule variable entre **xred** ou **xcart** doit être fournie. Les positions atomiques évoluent si **ionmov** /= 0.

»»» Caractéristique(s) : EVOLVING		
xred vectors (X) of atom positions in RED uced coordinates	Type de variable : réelle ; (3,min(natom , natrd)) ; Valeur par défaut : 0.0	Dimensions :

Elle donne les locations atomiques dans la cellule unitaire en coordonnées par rapport aux translations d'espace primitif (pas en coordonnées cartésiennes). Ainsi, ce sont des nombres fractionnaires généralement compris entre 0 et 1 et sans dimension. Les coordonnées cartésiennes des atomes (en Bohr) sont données par : $R_cartesian = xred1 * rprimd1 + xred2 * rprimd2 + xred3 * rprimd3$, où (**xred1**, **xred2**, **xred3**) sont les coordonnées réduites données dans les colonnes de **xred**, (**rprimd1**, **rprimd2**, **rprimd3**) sont les colonnes du tableau des vecteurs primitifs **rprimd** en Bohr.

Si vous préférez travailler uniquement avec les coordonnées cartésiennes, vous devez entièrement travailler avec **xcart** et ignorer **xred**, dans ce cas **xred** doit être absente du fichier d'entrée. Une et une seule variable entre **xred** ou **xcart** doit être fournie. Les positions atomiques évoluent si **ionmov** /= 0.

La répétition de **xcart** dans le fichier de sortie principal est accompagné par sa répétition en Angstrom, appelée **xangst**.

»»» Caractéristique(s) : NO_MULTI		
znuc1 charge - Z - of the NUCL eus	Type de variable : réelle ; Valeur par défaut : aucune	Dimensions : (np sp) ;

Elle donne la charge nucléaire de chaque type de pseudopotentiel, en ordre. Si **znuc1** n'est pas en accord avec la charge nucléaire, tel que donné dans les fichiers de pseudopotentiels, le programme écrit un message d'erreur et s'arrête.

Remarque 1.1.5

Dans les fichiers de pseudopotentiels, **znuc1** est appelée "zatom".

Pour un atome “factice”, avec `znuc1 = 0`, tel qu'utilisé dans le cas des calculs avec seulement une surface de jellium, Abinit fixe arbitrairement le rayon covalent à un.

1.2 Variables d'entrée bse (Bethe-Salpeter)

Anaddb



Multibinit



Optic



α -TDEP



Aim



Paramètres externes



Statistique



Index

A

accuracy 1
 acell 1, 8–11
 angdeg 1, 8, 11

B

boxcutmin 2
 bxctmindg 2

C

chksymbreak 7

D

dilatmx 2

E

ecut 2
 einterp 3
 ENERGY 14
 EVOLVING 8, 10

F

fband 2, 5
 fockdownsampling 6

I

icoulomb 15

INPUT_ONLY 9, 10
 ionmov 8, 10, 15, 16
 iscf 3, 4, 7, 14, 16
 ixc 3

J

jdtset 3, 5, 15

K

kpt 4, 6
 kptbounds 6
 kptnrm 4
 kptopt 4–6, 10, 16
 kptrlatt 5, 6, 10
 kptrlen 6
 kssform 5

L

LENGTH 16

M

mixalch 7

N

natom 4, 11, 13, 14, 16
 natrd 5, 14, 16
 nband 5, 7, 15
 nbandhf 5
 nbandkss 5
 ndivk 6

ndtset 3–5, 15
 ngkpt 5, 6, 10
 nkpath 6
 nkpt 4, 6, 16
 nkpthf 6
 NO_MULTI 13
 npsp 7, 16
 npspalch 7
 npulayit 2
 nshiftk 5, 6, 10
 nspden 4
 nspinor 5
 nsppol 7
 nstep 2, 7
 nsym 7, 13
 ntypalch 7
 ntypat 7, 11, 13, 15

O

occ 7
 occopt 5, 7, 16
 optcell 8, 10
 optdriver 1, 3, 5
 optforces 2

P

pawecutdg 2
 pawmixdg 2
 pawnhatxc 2
 pawovlp 2
 pawxcdev 2
 prtden 2
 prteig 2

Q

qpnt 4

R

rprim 1, 4, 8, 10, 11

rprimd 8, 10, 16

S

scalecart 8–10
 shiftk 5, 6, 10
 structure 10
 symrel 7, 13

T

timopt 2
 tnons 13
 toldfe 14
 toldff 14
 tolmxfr 2
 tolsym 8
 tolvr 2, 14
 tolwfr 7, 14
 trons 7
 tsmear 8
 typat 11–14

U

udtset 4, 15
 usepaw 3
 usewvl 3, 15

W

wtk 4, 7, 16
 wvl_hgrid 15, 16

X

xcart 15, 16
 xred 13, 15, 16

Z

znuc1 11, 13, 15, 16

Bibliographie

[Pickett et al., 1988] Pickett, W. E., Krakauer, H., and Allen, P. B. (1988). Smooth Fourier interpolation of periodic functions. *Phys. Rev. B*, 38(4):2721–2726.