Variables de Abinit

9.6.2

https://docs.abinit.org/variables

ÉQUIPE PÉDAGOGIQUE

Jean-Pierre TCHAPET NJAFA, PhD

Table des matières

Abinit

Contenu du chapitre

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 **abinit**.

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.

accuracy
ACCURACY

Type de variable : entière ; **Dimensions** : scalaire ;
Valeur par défaut : 0

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 ; **Dimensions** : (3) ;
Valeur par défaut : 3*1

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 ; **Dimensions** : (3) ;
Valeur par défaut : aucune

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 choisi tel que la symétrie trigonale qui les échange est le long de l'axe cartésien z :

```
R1 = (a,     0,            c)
R2 = (-a/2,  sqrt(3)/2*a,  c)
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 **CUT**off

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

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 **G** dans l'espace réciproque :

$$\frac{1}{2}(2\pi)^2(\mathbf{k}+\mathbf{G})^2 < \texttt{ecut}. \tag{1.1.1}$$

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, 1Ha = 27.211 384 5 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.

einterp
Electron bands **INTERP**olation

Type de variable : réelle ; **Dimensions** : (4) ;
Valeur par défaut : `[0,0,0,0]`

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.

iscf
Integer for **S**elf-**C**onsistent-**F**ield cycles

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

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

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

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 **D**a**T**a**SET**s

Type de variable : entière ; **Dimensions** : `ndtset` ;
Valeur par défaut : `[{'start': 1, 'stop':'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 permisse. 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.

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.

kptnrm
K-PoinTs NoRMalization
Type de variable : réelle ; **Dimensions** : scalaire ;
Valeur par défaut : 1

Elle établi 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.

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 prisent par `kptopt` sont 0, 1, 2, 3, 4 et des valeurs négatives.

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**).

nband
Number of **BAND**s

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

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 fourni pour permettre que le nombre de bandes traité varie d'un k point à un autre. Pour les valeurs de occopt non égalent 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**.

nbandhf
Number of **BAND**s for (**H**artree)-**F**ock exact exchange

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

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 **DaTaSET**s

Type de variable : entière ; **Dimensions** : scalaire ;
Valeur par défaut : 0

Elle donne le nombre de jeux de données à traiter. Si 0, cela signifie que le traitement du jeux 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 **G**rid points for **K** **P**oin**T**s generation

Type de variable : entière ; **Dimensions** : 3 ;
Valeur par défaut : [0,0,0]

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 essayera 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**.

nkpath
Number of **K**-points defining the **PATH**

Type de variable : entière ; **Dimensions** : scalaire ;
Valeur par défaut : 0

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**.

nkpt
Number of **K**-**P**oints

Type de variable : entière ; **Dimensions** : scalaire ;
Valeur par défaut : 1 if kptopt == 0, 0 if not

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)$= 40$.

nkpthf
Number of **K**-**P**oints for (**H**artree) **F**ock exact exchange

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

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**.

nshiftk
Number of **SHIFT**s for **K** point grids

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

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.

nsppol
Number of **SP**in **POL**arization

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

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.

nstep
Number of (non-)self-consistent field **STEP**s

Type de variable : entière ; **Dimensions** : scalaire ;
Valeur par défaut : 30

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 gape 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.

nsym
Number of **SYM**metry operations

Type de variable : entière ; **Dimensions** : scalaire ;
Valeur par défaut : 0

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 ; **Dimensions** : scalaire ;
Valeur par défaut : 1

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 pseudopotentiels. 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 **mixalch**, **npsp**, **ntypalch** et **npspalch**.

occopt
OCCupation **OPT**ion

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

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 redimensionnées 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)×rprim(i,1)×acell(1)`
- `R2p(i) = scalecart(i)×rprim(i,2)×acell(2)`
- `R3p(i) = scalecart(i)×rprim(i,3)×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∧R2).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

```
rprim  0.86602  0.5  0.0
      -0.86602  0.5  0.0
       0.0      0.0  1.0
```

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

```
angdeg 90 90 120
```

Remarquons que ce qui suit doit aussi bien fonctionner :

```
rprim  sqrt(0.75)  0.5  0.0
      -sqrt(0.75)  0.5  0.0
       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) :

```
  rprim  "a"      0        0
          0      "a"      0
         "a/2"   "a/2"    "c/2"
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 :

```
  rprim   1        0       0
          0        1       0
          1/2      1/2     1/2
scalecart  "a"  "a"  "c"
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** :

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

En fait, cette dernière correspondrait à :

```
  rprim  "a"      0       0
          0      "a"     0
         "c/2"   "c/2"   "c/2"
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

rprimd
Real space **PRIM**itive translations, **D**imensional

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

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)×rprim(i,1)×acell(1)
- R2p(i) = rprimd(i,2) = scalecart(i)×rprim(i,2)×acell(2)
- R3p(i) = rprimd(i,3) = scalecart(i)×rprim(i,3)×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 CARTesian coordinates

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

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 tétragonales, 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.

»»»

shiftk
SHIFT for **K** points

Type de variable : réelle ; **Dimensions** : (3,**nshiftk**) ;
Valeur par défaut : 3*1

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 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.

»»»

structure
initialize the crystalline **STRUCTURE** from …

Type de variable : chaîne de caractères ; **Dimensions** : scalaire ;
Valeur par défaut : aucune

Cette variable fourni une interface simplifiée pour construire une structure crystalline 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 (15 février 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 :

```
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.

```
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 :

```
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.

```
# Structure de maille du MgB2

acell   2*3.086  3.523 Angstrom

rprim   0.866025403784439  0.5  0.0
       -0.866025403784439  0.5  0.0
        0.0                0.0  1.0

natom   3

# Positions réduites suivies par les symboles des éléments.

xred_symbols
 0.0  0.0  0.0  Mg
 1/3  2/3  0.5  B
 2/3  1/3  0.5  B
```

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

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

Un fichier POSCAR typique pour le MgB_2 ressemble à ceci (ignorez les commentaires) :

```
# Titre (ignoré par Abinit)

Mg1 B2

# Facteur d'échelle pour des vecteurs de mailles.

1.0

# Vecteurs de maille en  Angstrom (Rappel: Abinit utilise le BOHR par défaut)

2.672554  1.543000 0.000000
-2.672554 1.543000 0.000000
0.000000  0.000000 3.523000

# Liste des symboles des éléments

Mg B

# Nombre d'atomes pour chaque symbole

1 2

# "direct" pour les coordonnées réduites ou "cartesian".

direct

# Coordonnées suivie par le symbole chimique de l'atome.

0.000000 0.000000 0.0 Mg
0.333333 0.666667 0.5 B
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 $Å^3$. 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 :

```
ntypat 2
typat 1 2 2
znucl  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 **IMPORTANTS**

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'entré. 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 **znucl** 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 **znucl** 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 **znucl** 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.

symrel
SYMmetry in **RE**a**L** 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`.

tnons
Translation **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 **DiF**ference of total **E**nergy

Type de variable : réelle ; **Dimensions** : scalaire ;
Valeur par défaut : `0.0`

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.

»»»

toldff
TOLerance on the **DiF**ference of **F**orces

Type de variable : réelle ; **Dimensions** : scalaire ;
Valeur par défaut : `0.0`

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.

»»»

tolvrs
TOLerance on the potential **V**(r) **R**e**S**idual

Type de variable : réelle ; **Dimensions** : scalaire ;
Valeur par défaut : `0.0`

Elle paramètre une tolérance pour le potentiel résiduel qui, lorqu'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.

»»»

tolwfr
TOLerance on **W**ave**F**unction squared **R**esidual

Type de variable : réelle ; **Dimensions** : scalaire ;
Valeur par défaut : `0.0`

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 :

$$\langle n\mathbf{k}| \left(H - \epsilon_{n\mathbf{k}}\right)^2 |n\mathbf{k}\rangle \,, \text{ avec } \epsilon_{n\mathbf{k}} = \langle n\mathbf{k}| H |n\mathbf{k}\rangle \,, \tag{1.1.2}$$

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.

»»»

typat
TYPe of **AT**oms

Type de variable : entière ; **Dimensions** : `[3,'natrd'] if natrd < natom, [3, 'natom'] otherwise` ;
Valeur par défaut : `1 if natom == 1, None otherwise`

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 pseudopotentiels. 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 :

```
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

```
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.

udtset
Upper limit on a **D**a**T**a **SET**s

Type de variable : entière ; **Dimensions** : (2) ;
Valeur par défaut : aucune

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 :

```
do i1=1,intarr(1)
 do i2=1,intarr(2)
  idtset=idtset+1
  dtsets(idtset)%jdtset=i1*10+i2
 end do
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.

usewvl
Use **W**a**V**e**L**et basis set

Type de variable : entière ; **Dimensions** : scalaire ;
Valeur par défaut : 0

Utilisée pour définir si le calcul est fini sur une base d'ondelette paramétrée oou 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.

wtk
Weigh**T**s for **K** points

Type de variable : réelle ; **Dimensions** : (**nkpt**) ;
Valeur par défaut : **nkpt***1.0

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 ; **Dimensions** : scalaire ;
Valeur par défaut : 0.5

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 ; **Dimensions** : (3,min(natom,natrd)) ;
Valeur par défaut : aucune

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 8Å), 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 ; **Dimensions** : (3,min(natom,natrd)) ;
Valeur par défaut : 0.0

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

`znucl`
charge -**Z**- of the **NUCL**eus

Type de variable : réelle ; **Dimensions** : (**`npsp`**) ;
Valeur par défaut : aucune

Elle donne la charge nucléaire de chaque type de pseudopotentiel, en ordre. Si `znucl` 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, `znucl` est appelée "`zatom`".

Pour un atome "factice", avec `znucl = 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)

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

»»» Pertinente uniquement si : `optdriver == 99`

bs_algorithm
Bethe-**S**alpeter **ALGORITHM**

Type de variable : entière ; **Dimensions** : scalaire ;
Valeur par défaut : 2

Cette variable d'entrée défini l'algorithme employé pour calculer la fonction diélectrique macroscopique. Les valeurs possible sont `[1,2,3]` :

- 1 → Le diélectrique macroscopique est obtenu en réalisant une diagonalisation directe du hamiltonien excitonique. L'avantage est qu'elle donne un accès direct aux valeurs propres excitoniques aussi bien qu'aux forces d'oscillateur. Mais l'inconvénient est que c'est une approche gourmande en ressource mémoire et CPU puisque la taille du hamiltonien est de l'ordre de $(n_k * n_c * n_v)^2$, avec n_k le nombre de k points dans la zone de Brillouin complète, et n_c et n_v sont respectivement les nombres d'états de conduction et de valence. L'autre avantage est qu'elle peut être utilisé à la fois pour des calculs de résonance uniquement et de résonance+couplage.

- 2 → Méthode itérative de Haydock. La fonction diélectrique macroscopique est obtenue par des applications itératives du hamiltonien sur un ensemble de vecteurs dans l'espace électron-trou. Lest avantages sont qu'elle est moins gourmande en ressource mémoire et est généralement plus rapide que la diagonalisation directe pourvu que **zcut** soit plus grande que l'espacement d'énergie typique des valeurs propres. Comme inconvénient c'est une méthode itérative et par conséquent la convergence en respect avec **bs_haydock_niter** doit être vérifiée. Il n'est pas possible d'avoir l'information directe sur la spectre d'exciton, des forces d'oscillateur et des fonctions d'ondes excitoniques. À l'heure actuelle `bs_algorithm = 2` ne peut pas être utilisé pour les calculs dans lesquels le terme de couplage est inclus (approximation de Tamm-Dancoff).

- 3 → Méthode de gradient conjugué. Cette méthode nous permet de rechercher les premières petites valeurs propres excitoniques. Disponible uniquement dans le cas de calculs de résonance (approximation de Tamm-Dancoff).

»»» Pertinente uniquement si : `optdriver == 99`

bs_calctype
Bethe-**S**alpeter **CALC**ulation **TYPE**

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

Les valeurs possible sont `[1,2,3]` :

- 1 → Utilise les valeurs propres KS et les fonctions d'onde stockée dans le fichier WFK pour construire l'espace de transition.

- 2 → L'espace de transition est construit avec des orbitales de Kohn-Sham mais les énergies sont lu à partir d'un fichier GW externe.

- 3 → Des amplitudes QP et des énergies seront lu du fichier QPS et utilisées pour construire H_{ex}. Non encodé pour l'instant parce que $\langle\psi|\, r\, |\psi\rangle^{QP}$ doit être calculé en tenant compte de la non localité de l'énergie propre dans le commutateur $[H, r]$.

»»» Pertinente uniquement si : `optdriver == 99`

bs_coulomb_term
Bethe-**S**alpeter **COULOMB** **TERM**

Type de variable : entière ; **Dimensions** : scalaire ;
Valeur par défaut : 11

Cette variable gouverne le choix parmi les différentes options disponible pour le traitement du terme de Coulomb du hamiltonien Bethe-Salpeter. `bs_coulomb_term` est la concaténation de deux chiffres, étiquetés (A) et (B).

Le premier chiffre (A) peut prendre les valeurs `0, 1, 2` :

- `0` → Le terme de Coulomb n'est pas calculé. Ce choix est équivalent à calculer le spectre RPA mais utilisant la représentation dans l'espace de transition au lieu de l'approche la plus efficiente basée sur la somme sur les états.
- `1` → Le terme de Coulomb est calculé en utilisant l'interaction filtrée lu à partir d'un fichier SCR externe (standard excitonic calculation).
- `2` → Le terme de Coulomb est calculé en utilisant un modèle de fonction de filtrage (utile pour des études de convergence ou pour reproduire des résultats publiés).

Le second chiffre (B) peut prendre les valeurs `0, 1` :

- `0` → Utilise une approximation diagonale pour $W_{\mathbf{GG'}}$ (principalement utilisé pour accélérer des études de convergence).
- `1` → Le terme de Coulomb est correctement évalué en utilisant le vrai filtrage non local $W(\mathbf{r}, \mathbf{r}')$.

»»» Pertinente uniquement si : `optdriver == 99`

`bs_coupling`
Bethe-Salpeter COUPLING

Type de variable : entière ; **Dimensions** : scalaire ;
Valeur par défaut : `0`

La variable d'entrée `bs_coupling` définie le traitement du bloc de couplage du hamiltonien Bethe-Salpeter. Les valeurs possibles sont `0, 1`.

- `0` → Le bloc de couplage est négligé (c'est l'approximation de Tamm-Dancoff). Le code s'exécute rapidement et la matrice du hamiltonien requière moins de mémoire (facteur 4). C'est une bonne approximation pour le spectre d'absorption qui requière uniquement la connaissance de $\Im(\epsilon)$. La fiabilité de cette approximation doit être testée dans le cas de calculs EEFL.
- `1` → Le terme de couplage est inclus (non approximation de Tamm-Dancoff).

»»» Pertinente uniquement si : `optdriver == 99`

`bs_eh_cutoff`
Bethe-Salpeter Electron-Hole CUTOFF

Type de variable : entière ; **Dimensions** : (2) ;
Valeur par défaut : `['-inf','inf']`

Utilisée pour définir une coupure dans l'ensemble de base e-h. Seules les transitions dont l'énergie est située entre `bs_eh_cutoff(1)` et `bs_eh_cutoff(2)` seront considérées dans dans la construction du hamiltonien e-h.

»»» Pertinente uniquement si : `optdriver == 99`

`bs_exchange_term`
Bethe-Salpeter EXCHANGE TERM

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

- `0` → Le terme d'échange n'est pas calculé. Ceci est équivalent à négliger des effets de champ local dans la fonction diélectrique macroscopique.
- `1` → Le terme d'échange est calculé et ajouté au hamiltonien excitonique.

»»» Caractéristique(s) : ENERGY ; Pertinente uniquement si : `optdriver == 99`

`bs_freq_mesh`
Bethe-Salpeter FREQuency MESH

Type de variable : réelle ; **Dimensions** : (3) ;
Valeur par défaut : `[0.0,0.0,0.01]`

- `bs_freq_mesh(1)` définit la première fréquence pour le calcul de la fonction diélectrique macroscopique.
- `bs_freq_mesh(2)` donne la dernière fréquence pour le calcul de la fonction diélectrique macroscopique. Si zéro, `bs_freq_mesh(2)` est automatiquement paramétré à `MAX(resonant_energy) + 10` %.
- `bs_freq_mesh(3)` donne l'étape du *mesh* linéaire utilisé pour l'évaluation de la fonction diélectrique macroscopique.

»»» Pertinente uniquement si : `optdriver == 99` et `bs_algorithm == 2`

`bs_hayd_term`
Bethe-**S**alpeter **HAY**dock
TERMinator

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

Définit comment terminer l'expression de la fraction continue pour la fonction diélectrique. Le terminateur réduit le nombre d'itérations nécessaire pour converger en régularisant l'oscillation dans la partie la plus haute en énergie du spectre.

- `0` → Aucun terminateur. La contribution donnée par les termes manquant dans la chaine de Lanczos sont paramétrés à zéro.
- `1` → Utilise la fonction terminatrice. L'expression particulière dépend du type de calcul : dans le cas de la résonance uniquement, les coefficients a_i et b_i pour `i > niter`, sont remplacés par leurs valeurs à `i = niter`. Si le bloc de couplage est inclus, la fonction terminatrice est celle décrite dans [Rocca *et al.*, 2008].

»»» Pertinente uniquement si : `optdriver == 99` et `bs_algorithm == 2`

`bs_haydock_niter`
Bethe-**S**alpeter **HAYDOCK**
Number of **ITER**ations

Type de variable : entière ; **Dimensions** : scalaire ;
Valeur par défaut : `100`

`bs_haydock_niter` définit le nombre maximum d'itérations utilisé pour calculer la fonction diélectrique macroscopique. L'algorithme itératif s'arrête quand la différence entre deux évaluations consécutives du spectre optique est plus petite que **`bs_haydock_tol`**.

»»» Pertinente uniquement si : `optdriver == 99` et `bs_algorithm == 2`

`bs_haydock_tol`
Bethe-**S**alpeter **HAYDOCK**
TOLerance

Type de variable : réelle ; **Dimensions** : (2) ;
Valeur par défaut : `[0.02,0.0]`

Définit le critère de convergence pour la méthode itérative de Haydock. L'algorithme itératif s'arrête lorsque la différence entre deux évaluations consécutives de la fonction diélectrique macroscopique est plus petite que `bs_haydock_tol(1)`. Le signe de `bs_haydock_tol(1)` définit la manière d'estimer l'erreur de convergence.

Une valeur négative signale que la convergence doit être atteinte pour chaque fréquence (critère stricte), alors qu'une valeur positive indique que l'erreur de convergence est estimée en moyennant par rapport à la gamme de fréquence entière (critère léger).

`bs_haydock_tol(2)` définit la quantité qui sera vérifiée pour la convergence :

- `0.0` → La partie réelle et la partie imaginaire doivent converger toutes les deux.
- `1.0` → Seule la partie réelle.
- `2.0` → Seule la partie imaginaire.

(Les derniers sont des nombres réels, la tolérance est 10^{-6}).

»»» Pertinente uniquement si : bs_interp_mode > 0 et bs_algorithm == 2 et bs_coupling == 0

bs_interp_kmult
Bethe-**S**alpeter **INTERP**olation **K**-point **MULTI**iplication factors

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

Cette variable définit le nombre de divisions utilisé pour générer le mesh dense dans une interpolation. ngkpt du mesh dense = bs_interp_kmult(:)*ngkpt du mesh vulgaire.

»»» Pertinente uniquement si : bs_interp_mode == 3 et bs_algorithm == 2 et bs_coupling == 0

bs_interp_m3_width
Bethe-**S**alpeter **INTERP**olation **M**ethod**3 WIDTH**

Type de variable : réelle ; **Dimensions** : scalaire ;
Valeur par défaut : 1.0

Définit la largeur de la région où le traitement de la divergence est appliqué pour une interpolation BSE.

»»» Pertinente uniquement si : bs_interp_mode > 0 et bs_algorithm == 2 et bs_coupling == 0

bs_interp_method
Bethe-**S**alpeter **INTERP**olation **METHOD**

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

bs_interp_method sélectionne la méthode d'interpolation :

- 0 → Interpolation utilisant la technique de Y. Gillet avec 8 voisins (voir [Gillet *et al.*, 2016]).
- 1 → Interpolation utilisant la technique de Rohlfing & Louie (voir l'article mentionné ci-dessus et [Rohlfing et Louie, 2000])

»»» Pertinente uniquement si : bs_interp_mode > 0 et bs_algorithm == 2 et bs_coupling == 0

bs_interp_mode
Bethe-**S**alpeter **INTERP**olation **MODE**

Type de variable : entière ; **Dimensions** : scalaire ;
Valeur par défaut : 0

bs_interp_mode sélectionne le mode d'interpolation :

- 0 → Pas d'interpolation. Un calcul Bethe-Salpeter standard est exécuté.
- 1 → Interpolation simple.
- 2 → Traitement de la divergence sur tout l'ensemble des k points dense.
- 3 → Traitement de la divergence avec la diagonale dans le k espace et interpolation simple ailleurs.

»»» Pertinente uniquement si : bs_interp_mode > 0 et bs_algorithm == 2 et bs_coupling == 0

bs_interp_prep
Bethe-**S**alpeter **INTERP**olation **PREP**aration

Type de variable : entière ; **Dimensions** : scalaire ;
Valeur par défaut : 0

Cette variable nous permet de déclencher la préparation de l'interpolation avec la méthode 2 ou la méthode 3. Elle génère la décomposition du BSR dans des coefficients a, b, c utilisés pour l'interpolation.

»»» Pertinente uniquement si : bs_interp_mode > 0 et bs_algorithm == 2 et bs_interp_method == 1 et bs_coupling == 0

bs_interp_rl_nb
Bethe-**S**alpeter **INTERP**olation **R**ohlfing & **L**ouie **N**eigh**B**our

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

Elle donne l'index du voisin qui est utilisé dans la méthode de Rohlfing et Louie [Rohlfing et Louie, 2000].

»»» Pertinente uniquement si : optdriver == 99

bs_loband
Bethe-**S**alpeter **L**owest **O**ccupied **BAND**

Type de variable : entière ; **Dimensions** : (nsppol) ;
Valeur par défaut : 0

Cette variable définit l'index de la bande occupée la plus basse utilisée pour la construction de l'ensemble de base électron-trou. Pour des calculs de spin polarisé, on doit fournir deux indices séparés pour le spin *up* et le spin *down*. Une énergie de coupure additionnelle peut être appliquée au moyen de la variable d'entrée **bs_eh_cutoff**.

Pertinente uniquement si : optdriver == 99 et bs_algorithm in [2,3]

bs_nstates
Bethe-**S**alpeter **N**umber of **STATES**

Type de variable : entière ; **Dimensions** : scalaire ;
Valeur par défaut : 0

bs_nstates définit le nombre maximum d'états excitoniques calculés dans la diagonalisation directe de la matrice excitonique ou dans la méthode de gradient conjugué. Le nombre d'états doit être suffisamment grand pour une description correcte des propriétés optiques dans la bande de fréquence à laquelle on s'intéresse.

1.3 Variables d'entrée gstate (Ground State)

Cette section liste et donne la description des noms (*keywords*) des variables d'entrée gstate (état fondamental) à utiliser dans le fichier d'entrée pour l'exécutable **abinit**.

algalch
ALGorithm for generating **ALCH**emical pseudopotentials

Type de variable : entière ; **Dimensions** : (ntypalch) ;
Valeur par défaut : ntypalch*1

Utilisé pour la génération de pseudopotentiels alchimiques, c'est-à-dire, quand **ntypalch** est différent de zéro.

Donner l'algorithme à utiliser pour générer les potentiels alchimiques **ntypalch** à partir de différents pseudopotentiels **npspalch** dédier à cet usage.

Dans la version actuelle de Abinit, algalch ne peut qu'avoir la valeur 1, c'est-à-dire :

- le potentiel local est mélangé, grâce aux coefficients de mélange **mixalch** ;
- les facteurs de forme des projecteurs non locaux sont tous préservés, et tous considérés pour générer le potentiel alchimique ;
- les coefficients scalaires des projecteurs non locaux sont multipliés par la proportion du type d'atome correspondant présent dans **mixalch** ;
- le rayon caractéristique pour la charge de cœur est une combinaison linéaire des rayons caractéristiques des charges de cœur, construits avec le coefficient de mélange **mixalch** ;
- la fonction de charge de cœur $f(r/r_c)$ est une combinaison linéaire des fonctions de charge de cœur, construites avec le coefficient de mélange **mixalch**.

D'autre algorithmes de mélange seront insérés plus tard.

Note 1.1

Notons que ce mélange alchimique ne peut être utilisé avec PAW.

auxc_ixc
AUxiliary **XC** functional for hybrid functional, **IXC** number

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

Spécification d'une fonctionnelle auxiliaire d'échange-correlation, grâce à sa valeur **ixc**, pour possiblement remplacer la lourde évaluation d'une fonctionnelle hybride à des occasions spécifiques, par exemple quand l'opérateur de Fock est gelé durant le cycle auto-cohérent, grâce à **fockoptmix** == 11, ou quand on évalue les forces de correction dû à la densité résiduelle. Cet fonctionnelle auxiliaire d'échange-correlation doit être redimensionnée, grâce à **auxc_scal** quand **fockoptmix** == 11. Si **qwcalctyp** == 5, 15 or 25, auxc_ixc renvoie à **ixc_sigma** au lieu de **ixc**.

auxc_scal
AUxiliary **XC** functional for hybrid functional-**SCAL**ing factor

Type de variable : réelle ; **Dimensions** : scalaire ;
Valeur par défaut : 1.0

Possible facteur d'échelle pour la fonctionnelle auxiliaire d'échange-correlation définit par **auxc_ixc** qui a pour but de remplacer l'opérateur de Fock ou la fonctionnelle hybride quand **fockoptmix** == 11.

La valeur par défaut 1.0 correspond à la fonctionnelle xc non modifiée. Quand la fonctionnelle auxiliaire est utilisée pour remplacer la fonctionnelle hybride dans des boucles SCF, une valeur de 1.5 a été observée pour accélérer d'une manière ou d'une autre la convergence.

boxcenter
BOX CENTER

Type de variable : réelle ; **Dimensions** : (3) ;
Valeur par défaut : [0.5,0.5,0.5]

Cette variable définit le centre de la boîte, dans des coordonnées réduites. Présentement, cette information est uniquement utilisée dans le cas d'un calcul DFT dépendant du temps de la force d'oscillateur. On doit prendre boxcenter tel qu'il soit rigoureusement le centre du cluster ou de la molécule. Par défaut est sensible quand l'espace entourant le cluster ou la molécule a **xred** 0 ou 1. Au contraire, quand le cluster ou la molécule est proche de l'origine, il est préférable de prendre boxcenter=[0.0,0.0,0.0].

boxcutmin
BOX CUT-off **MIN**imum

Type de variable : réelle ; **Dimensions** : scalaire ;
Valeur par défaut : 2.0

Le ratio de boîte de coupure est le ratio entre le rayon de la sphère G qui peut être insérée dans la boîte FFT (voir **ngfft**) et la sphère G utilisée pour présenter les fonctions d'onde tels que calculées de la variable d'entrée **ecut**. Pour que la densité soit exact (dans le cas de la partie d'onde plane, pas le PAW situé sur les termes),ce ratio doit **être au moins deux**.

Alerte 1.2

Si on décide d'utiliser un ratio plus petit (par exemple 1.5), on gagnera en rapidité au prix de la précision. Gardons à l'esprit que l'utilisation d'une valeur de boxcutmin très proche de 1 conduira à des erreurs d'exécution dans les routines (programmes) FFT, donc une valeur plus grande que 1.1 est fortement recommandée.

Il est aussi à noter que la qualité des forces est affectée par la valeur de boxcutmin. La valeur par défaut (2.0) est correcte pour des calculs de l'état fondamental et des relaxations structurales. Au contraire, des calculs de fréquences vibrationnelles basés sur des méthodes de différence fini comme celle implémentée par phonopy sont plutôt sensibles à la qualité des forces.

cellcharge
CELL CHARGE

Type de variable : réelle ; **Dimensions** : (**nimage**) ;
Valeur par défaut : 0

Utilisée pour établir la charge excédent entre le nombre d'électrons remplissant les bandes et la charge nominale associée avec les cœurs atomiques. Le code ajoute le nombre d'électrons de valence fourni par les pseudopotentiels de chaque type (appelé "zval"), puis ajoute cellcharge pour obtenir le nombre par unité cellulaire, **nelect**. Puis, si **iscf** est positive, le code ajoute les bande d'occupation (données dans le tableau **occ**) pour toutes les bandes à chaque k point, ensuite multiplie par le poids du k point **wtk** à chaque k point. On appelle cette somme "**nelect_occ**" (pour nombre d'électrons à

partir des nombres d'occupation). Il est également requis que : `nelect_occ=nelect`. Pour traiter un système neutre, ce qui est désiré dans à peu près tous les cas, on doit utiliser `cellcharge = 0`. Pour traiter un système auquel il manque un électron par cellule unitaire, utilisé `cellcharge = +1`.

charge
CHARGE
Type de variable : réelle ; **Dimensions** : scalaire ;
Valeur par défaut : 0

Cette variable est obsolète, et est remplacée par `cellcharge`.

chkexit
CHec**K** whether the user want to **EXIT**
Type de variable : entière ; **Dimensions** : scalaire ;
Valeur par défaut : 0

Si `chkexit` vaut 1 ou 2, Abinit vérifiera si l'utilisateur veut interrompre l'exécution (en utilisant le mot-clé "*exit*" au début du fichier d'entrée en créant un fichier nommé "`abinit.exit`") :

- 0 → la vérification n'est pas du tout exécutée ;
- 1 → la vérification n'est pas exécutée fréquemment (après chaque étape SCF) ;
- 2 → la vérification est faite fréquemment (après une petite bande, à chaque k point).

Dans tous les cas, la vérification est exécutée au maximum toutes les 2 secondes du temps CPU.

chkprim
CHec**K** whether the cell is **PRIM**itive
Type de variable : entière ; **Dimensions** : scalaire ;
Valeur par défaut : 1

Si le découvreur de symétrie est utilisé (voir `nsym`), une valeur différente de zéro de `chkprim` sera s'arrêter le code si une cellule non primitive est utilisée. Si `chkprim = 0`, une alerte est émise, mais l'exécution ne s'arrête pas.

Si vous générez la géométrie atomique et cellulaire en utilisant `spgroup`, vous devez générer une cellule primitive en utilisant `brvltt = -1`.

Caractéristique(s) : INPUT_ONLY

chksymbreak
CHec**K SYM**metry **BREAK**ing
Type de variable : entière ; **Dimensions** : scalaire ;
Valeur par défaut : 1

Cette variable gouverne le comportement du code quand il y a une potentielle source de brisure de symétrie reliée à la grille de k point.

Quand `chksymbreak = 1`, le code s'arrête si la grille de k point est non-symétrique, dans le cas `kptopt = 1, 2 or 4`. De plus, le code s'arrête si `nshiftk` n'est pas `1, 2 or 4`.

Notons que la vérification est désactivée quand le nombre de k points dans le BZ est plus grand que 40^3.

Quand `chksymbreak = 0`, il n'y a pas une telle vérification.

Caractéristique(s) : INPUT_ONLY

chksymtnons
CHec**K SYM**metry of **TNONS**
Type de variable : entière ; **Dimensions** : scalaire ;
Valeur par défaut : 1

Cette variable gouverne le comportement du code quand il y a une potentielle brisure de symétrie, reliée à la présence des translations non-symmorphiques ne laissant pas la grille d'échange-correlation FFT invariante.

Quand `chksymtnons = 1`, le code s'arrête si la partie de la translation non-symmorphique des opérations de symétrie a des composants qui sont différents de zéro, ou des simples fractions avec 2, 3, 4, 5, 6, 8, 9, 10 ou 12 comme dénominateur. De plus, des suggestions pour contourner le problème sont fournis dans le fichier de sorti.

Quand `chksymtnons = 2`, le code fait une vérification similaire, mais ne s'arrête pas après avoir fourni dans le fichier de sortie des suggestions pour contourner le problème.

Quand chksymtnons = 0, le code saute la vérification.

chrgat
CHARGE of the **AT**oms

Type de variable : réelle ; **Dimensions** : [natrd] if natrd<natom, [natom] otherwise ;
Valeur par défaut : 0.0

Cette variable donne la charge cible intégrée dans le cas d'un calcul DFT contraint, voir constraint_kind. Donné en unité de charge atomique (=moins la charge de l'électron). Notons que ce nombre est la charge positive nette à l'intérieur de la sphère : on soustrait de la charge nucléaire ziontypat la densité de valence électronique intégrée dans la sphère défini par ratsph. La dernière a en réalité une valeur négative. Notons que si le rayon de la sphère ratsph n'est pas suffisamment grand, la quantité d'électrons sera plus petite que prévu sur la base de l'intuition chimique. Ceci signifie qu'il y a dans ce cas un biais vers des charges intégrées trop positives. Par contre, si le rayon de la sphère est trop grand, les sphères dépasseront, et les électrons dans la région interatomique seront comptés doublement.

»»» Pertinente uniquement si : iscf > 1 and iscf < 10 and iomov /= 4

constraint_kind
CONSTRAINT KIND in constrained DFT

Type de variable : entière ; **Dimensions** : (ntypat) ;
Valeur par défaut : 0

Si constraint_kind est différente pour au moins un type d'atome, l'algorithme de DFT contraint est activé. constraint_kind défini, pour chaque type d'atome, le type de contrainte(s) imposé par la DFT contrainte. Quand constraint_kind est zéro pour un type d'atome, il n'y a pas de contrainte appliquée à ce type d'atome. Sinon, des contraintes différentes peuvent être imposées sur la charge totale (ionique+électronique) et/ou la magnétisation, calculée dans une sphère de rayon ratsph, possiblement enduit dans une largeur ratsm. Une telle charge ionique+électronique est imposée à être égale à chrgat, alors que la magnétisation doit être comparée à spinat. Le premier nombre de constraint_kind défini la contrainte sur la charge, tandis que le second défini la contrainte sur le magnétisme.

Quand constraint_kind est 10 ou plus, la contrainte de charge sera imposée.

»»» Caractéristique(s) : NO_MULTI, INPUT_ONLY

cpuh
CPU time limit **H**ours

Type de variable : réelle ; **Dimensions** : scalaire ;
Valeur par défaut : 0.0

Seule un des trois paramètres réels cpus, cpum et cpuh peut être défini dans le fichier d'entrée pour paramétrer une limite de temps CPU. Quand la tâche atteint cette limite, elle essayera de s'arrêter doucement. Toutefois, notons que ceci peut pourtant prendre un peu de temps. Si l'utilisateur veut une limite de temps CPU solide, le présent paramètre doit être réduit suffisamment. L'intuition à propos de la marge actuelle à prendre en compte doit venir de l'expérience. Une valeur de zéro n'as aucune action sur la tâche.

»»» Caractéristique(s) : NO_MULTI, INPUT_ONLY

cpum
CPU time limit **M**inutes

Type de variable : réelle ; **Dimensions** : scalaire ;
Valeur par défaut : 0.0

Seule un des trois paramètres réels cpus, cpum et cpuh peut être défini dans le fichier d'entrée pour paramétrer une limite de temps CPU. Quand la tâche atteint cette limite, elle essayera de s'arrêter doucement. Toutefois, notons que ceci peut pourtant prendre un peu de temps. Si l'utilisateur veut une limite de temps CPU solide, le présent paramètre doit être réduit suffisamment. L'intuition à propos de la marge actuelle à prendre en compte doit venir de l'expérience. Une valeur de zéro n'as aucune action sur la tâche.

»»» Caractéristique(s) : NO_MULTI, INPUT_ONLY

cpus
CPU time limit **S**econds

Type de variable : réelle ; **Dimensions** : scalaire ;
Valeur par défaut : 0.0

Seule un des trois paramètres réels cpus, cpum et cpuh peut être défini dans le fichier d'entrée pour paramétrer une limite de temps CPU. Quand la tâche atteint cette limite, elle essayera de s'arrêter doucement. Toutefois, notons que ceci peut pourtant prendre un peu de temps. Si l'utilisateur veut une

limite de temps CPU solide, le présent paramètre doit être réduit suffisamment. L'intuition à propos de la marge actuelle à prendre en compte doit venir de l'expérience. Une valeur de zéro n'as aucune action sur la tâche.

»»» Caractéristique(s) : ENERGY

`diecut` **DIE**lectric matrix energy **CUT**off	**Type de variable** : réelle ; **Valeur par défaut** : `2.2`	**Dimensions** : scalaire ;

C'est l'énergie cinétique de coupure qui contrôle le nombre d'ondes planes utilisées pour représenter la matrice diélectrique : $(1/2)[2\pi \mathbf{G}_{diel,max}]^2 =$ `diecut` avec $\mathbf{G}_{diel,max}$ la longueur maximum de l'espace réciproque des vecteurs d'onde d'onde plane pour la matrice diélectrique. Elle peut être spécifiée en Ha (par défaut, 1Ha = 27.211 384 5 eV), Ry, eV ou en K, puisque `diecut` a les caractéristiques `ENERGY`. Toutes les ondes planes dans cette sphère de base centrée en $\mathbf{G} = 0$ sont inclus dans la base. Ceci est utile seulement quand `iprcel >= 21`, ce qui signifie qu'un arrangement de pré-conditionnement basé sur la matrice diélectrique est utilisée.

Remarque 1.3.1

Une `diecut` négative définira la même sphère de base diélectrique que la valeur positive correspondante, mais la grille FFT sera identique à celle utilisée pour les fonctions d'onde. La grille FFT plus petite, utilisée quand `diecut` est positive, donne exactement les mêmes résultats.

Aucune signification pour des calculs RF pour l'instant.

»»» Caractéristique(s) : ENERGY

`diegap` **DIE**lectric matrix **GAP**	**Type de variable** : réelle ; **Valeur par défaut** : `0.1`	**Dimensions** : scalaire ;

Elle donne une estimation approximative du gap diélectrique entre le niveau d'énergie le plus haut calculé dans l'exécution, et le paramètre des bandes non représentés. Elle est utilisée pour extrapoler la matrice diélectrique quand `iprcel >= 21`. Elle peut être spécifiée en Ha (par défaut, 1Ha = 27.211 384 5 eV), Ry, eV ou en K, puisque `diegap` a les caractéristiques `ENERGY`.

Aucune signification pour des calculs RF pour l'instant.

»»» Pertinente uniquement si : `iprcel >= 21`

`dielam` **DIE**lectric matrix **LAM**bda	**Type de variable** : réelle ; **Valeur par défaut** : `0.5`	**Dimensions** : scalaire ;

Elle donne la quantité des états occupés avec l'énergie moyenne donnée par le niveau le plus haut calculé dans l'exécution, incluse dans l'extrapolation de la matrice diélectrique.

Aucune signification pour des calculs RF pour l'instant.

Anaddb

Multibinit

Optic

a-TDEP

Aim

Paramètres externes

Statistique

Index

T

U

W

X

Z

Bibliographie

[Gillet *et al.*, 2016] Gillet, Y., Giantomassi, M. et Gonze, X. (2016). Efficient on-the-fly interpolation technique for Bethe–Salpeter calculations of optical spectra. *Comput. Phys. Commun.*, 203 :83–93.

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

[Rocca *et al.*, 2008] Rocca, D., Gebauer, R., Saad, Y. et Baroni, S. (2008). Turbo charging time-dependent density-functional theory with lanczos chains. *J. Chem. Phys.*, 128(15) :154105.

[Rohlfing et Louie, 2000] Rohlfing, M. et Louie, S. G. (2000). Electron-hole excitations and optical spectra from first principles. *Phys. Rev. B*, 62(8) :4927–4944.