Classe python VaspRun pour le post-traitement de calculs VASP

Germain Vallverdu

21 octobre 2010

La classe `VaspRun` est une classe python permettant de faire du post traitement de calculs VASP . Elle extrait les informations du fichier `vasprun.xml` créé par VASP à la fin du calcul. La raison du choix de lire les données sur le fichier `vasprun.xml` plutôt que sur les différents fichiers de sortie de VASP est que le fichier xml est un ensemble cohérent. Il contient à la fois les paramètres du calculs et les résultats. En utilisant ce fichier on est donc certain de la correspondance entre les résultats et les paramètres du calcul.

Il est inutile de connaître le langage python pour l'utiliser. La classe `VaspRun` peut être vue comme une simple boite à outils. Ce document décrit les possibilités offertes par cet outil.

Table des matières

1 Utilisation

1.1 Généralités :

Pour l'utiliser on procède en quatre étapes dans un terminal :

1. On lance python
2. On charge le fichier VASP.py qui contient les outils de post-traitement.
3. On crée un nouveau calcul VASP (pour parler technique : on crée une instance de la classe `VaspRun`).
4. On utilise un des outils (on appelle une méthode de la classe `VaspRun`).

1.2 Un exemple détaillé :

Un petit exemple, pour extraire les bandes d'énergie :

1. On lance python

```
[gvallver@kiwi] > python3
```

2. On charge le fichier VASP.py

```
>>> import VASP
```

3. On crée un calcul que l'on appelle `calcul`, le nom est sans importance. C'est ici que l'on appelle la classe `VaspRun`, comme elle est contenu dans le fichier VASP.py on l'appelle en indiquant : `VASP.VaspRun()`

```
>>> calcul = VASP.VaspRun()
```

4. on demande d'extraire les bandes d'énergie :

```
>>> calcul.extraireBandes()
```

Les étapes de 1 à 3 ne sont réalisées qu'une seule fois au départ. On va ensuite utiliser les outils que l'on souhaite de la même manière qu'à l'étape 4. On utilise la syntaxe :

```
mon_calcul . un_outil( options )
```

1.3 L'exemple complet, une vrai exécution :

Voilà ce qui se passe lorsqu'on entre la série de commande précédente. On peut voir que lors de la création du calcul (lorsqu'on instancie la classe `VaspRun`) des informations concernant le calcul s'affichent. On a entre autre le contenu du fichier `INCAR` puis le nombre et le type d'atome.

```
[gvallver@kiwi]> python3
Python 3.1.2 (r312:79147, Apr 15 2010, 12:35:07)
[GCC 4.4.3] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> import VASP
>>> calcul = VASP.VaspRun()

# Fichier xml du calcul : vasprun.xml
        * lecture du fichier xml
        * Lecture des mots clefs du calcul
        * Lecture des points K du calcul
        * Lecture des donnees sur les atomes du calcul

# fichier INCAR du calcul
     ENCUT = 400.0         ISTART = 0              NELM = 200
      PREC = normal          ISIF = 3            IBRION = 2
    EDIFFG = -0.01         ISMEAR = 0             LREAL =  auto
     EDIFF = 1e-06            NSW = 100           SIGMA = 0.05

# caracteristiques du systeme :
        * nombre d'atomes  : 8
        * nombre de types  : 1
        * liste des atomes : Si, Si, Si, Si, Si, Si, Si, Si

# Liste des types d'atomes du calcul et de leur caracteristiques :
Atome Si :
        masse                    : 28.085
        Nbre electron de valence : 4.0
        type                     : 1
        Pseudo potentiel         :  PAW_PBE Si 05Jan2001

>>> run.extraireBandes()
# Lecture des bandes d'energies
        * ISPIN           = 1
        * nombre de bandes = 8

# extraction des bandes d'energie
Fichier(s) cree(s) : 1
        bandes.dat

>>>
```

1.4 Autres syntaxes :

Il existe une autre syntaxe possible pour créer un calcul qui peut sembler plus simple. L'utilisation de l'une ou l'autre n'a pas d'importance.

```
[gvallver@kiwi]> python3
Python 3.1.2 (r312:79147, Apr 15 2010, 12:35:07)
[GCC 4.4.3] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> from VASP import *
>>> calcul = VaspRun()
...
```

Il est possible que le fichier xml du calcul ne s'appelle pas `vasprun.xml`, par exemple parce qu'il a été renommé. Dans ce cas, lors de la création du calcul, il suffit de préciser le nom (ou le chemin) du fichier xml entre guillemets dans les parenthèse. Voici un exemple où le fichier s'appelle `bandes.xml`.

```
[gvallver@kiwi]> python3
Python 3.1.2 (r312:79147, Apr 15 2010, 12:35:07)
[GCC 4.4.3] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> from VASP import *
>>> calcul = VaspRun(''bandes.xml'')

# Fichier xml du calcul : bandes.xml
        * lecture du fichier xml
        * Lecture des mots clefs du calcul
        * Lecture des points K du calcul
        * Lecture des donnees sur les atomes du calcu
...
```

1.5 Scripts :

Si on doit faire un traitement répétitif il est possible de placer l'ensemble des commandes dans un fichier à exécuter avec python. Reprenons l'exemple de l'affichage du fichier `INCAR` du calcul. On place dans un fichier `script.py` les lignes suivantes (les lignes commençant par # sont des commentaires) :

```
# on charge VASP
from VASP import *

# on cree le calcul
calcul = VaspRun(``bandes.xml'')

# on appelle la methode pour afficher le fichier INCAR
calcul.extraireBandes()
```

On exécute ensuite le script avec python dans un terminal :

```
[gvallver@kiwi]> python3 script.py

# Fichier xml du calcul : bandes.xml
        * lecture du fichier xml
        * Lecture des mots clefs du calcul
        * Lecture des points K du calcul
        * Lecture des donnees sur les atomes du calcul

# fichier INCAR du calcul
     ENCUT = 450.0         ISTART = 1            SYSTEM =
"Si bulk"
      PREC = medium        ICHARG = 11             ALGO =  FAST
    ISMEAR = 0              EDIFF = 1e-06          SIGMA = 0.05

# caracteristiques du systeme :
        * nombre d'atomes  : 2
        * nombre de types  : 1
        * liste des atomes : Si, Si

# Liste des types d'atomes du calcul et de leur caracteristiques :
Atome Si :
        masse                    : 28.085
        Nbre electron de valence : 4.0
        type                     : 1
        Pseudo potentiel         :  PAW_PBE Si 05Jan2001


# Lecture des bandes d'energies
        * ISPIN           = 1
        * nombre de bandes = 8

# extraction des bandes d'energie
Fichier(s) cree(s) : 1
        bandes.dat

[gvallver@kiwi]>
```

2 Liste des outils

En python les "outils" disponnibles dans une classe sont appelé des méthodes. Voici donc la liste des méthodes, ou outils, disponibles dans la classe `VaspRun` et leur utilisation. Les noms des méthodes sont assez explicites. On donne d'abord la liste des méthodes, leur utilisation est décrites plus loin.

On rappelle que pour utiliser un outil quel qu'il soit, on utilise la syntaxe :

```
mon_calcul . un_outil( options )
```

2.1 Liste des méthodes :

- Obtenir des informations sur les paramètres du calculs :
  - getINCAR()
  - getKPOINTS()

  - listerMotsClefs()
  - valeurMotClef("Mot Clef")
  - listerGroupesMotsClefs()
  - groupeMotsClefs("titre du groupe")
  - getInfoAtomes()

- Obtenir les bandes d'énergies :
  - extraireBandes(parDirection= True/False)
  - lectureBandes()

- Obtenir les densités d'états totale et projetées :

2.2 Obtenir de l'aide :

Pour obtenir de l'aide directement dans la consôle python (après avoir lancé python) on a deux possibilités :

- On utilise la fonction `help()` de python
- On appelle la méthode `help()` de la classe `VaspRun`.

Les deux possibilités rapporte les mêmes informations. La fonction `help()` de python contient cependant plus d'informations techniques alors que celle de la classe `VaspRun` contient uniquement le nom des méthodes, leurs but et leur syntaxe.

Voici les syntaxes possibles :

```
[gvallver@kiwi]> python3
Python 3.1.2 (r312:79147, Apr 15 2010, 12:35:07)
[GCC 4.4.3] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> from VASP import *
>>> VaspRun.help()
. . . affiche l'aide interne de VaspRun
>>>
>>> help(VaspRun)
. . . affiche l'aide python sur VaspRun
>>>
>>> import VASP
>>> VASP.VaspRun.help()
. . . affiche l'aide interne de VaspRun
>>>
>>> help(VASP.VaspRun)
. . . affiche l'aide python sur VaspRun
>>>
>>> calcul = VASP.VaspRun()
>>> help(calcul)
. . . affiche l'aide python sur VaspRun
>>>
```

2.3 Obtenir des informations sur les paramètres du calculs :

Ces méthodes permettent d'obtenir des informations sur les paramètres du calcul et le système étudié.

- getINCAR()

Cette méthode affiche l'ensemble des mots clefs présents dans le fichier INCAR du calcul.

- getKPOINTS()

  Affiche les informations du fichiers KPOINTS

- listerMotsClefs()

  Cette méthode interroge la base de données contenant l'ensemble des valeurs des mots clefs de VASP utilisés dans le calcul et affiche la liste des mots clefs. (voir `valeurMotClef( "Mot Clef" )`).

- valeurMotClef("Mot Clef")

  Cette méthode interroge la base de données contenant l'ensemble des valeurs des mots clefs de VASP utilisés dans le calcul et affiche la valeur du mot clef demandé (voir `listerMotsClefs()`).

- groupeMotsClefs("titre du groupe")

  Cette méthode interroge la base de données contenant les mots clefs du calcul et affiche la liste des mots clefs contenu dans le groupe dont le titre est donné en argument (voir `listerGroupesMotsClefs()`).

- listerGroupesMotsClefs()

  Cette méthode interroge la base de données contenant les mots clefs du calcul et affiche la liste des titres des groupes de mots clefs (voir `groupeMotsClefs( "titre" )`)

- getInfoAtomes()

  Affiche des informations sur le système : nombre d'atomes, types, pseudo ...

2.4 Obtenir les bandes d'énergies :

Ces méthodes permettent d'extraire les bandes d'énergie. Il n'est pas nécessaire d'appeler la méthode de lecture des bandes d'énergie avant celle pour extraire les bandes d'énergie.

- extraireBandes(parDirection= True/False)

  Méthode permettant d'extraire les bandes d'énergie du fichier xml et de les écrire dans un format pratique pour le tracer.

  - `calcul.extraireBandes()` ou `calcul.extraireBandes(parDirection=False )`

    extrait toutes les bandes dans le fichier "bandes.dat". Sur le fichier "bandes.dat", la colonne 1 donne l'indice du point k entre 0 et 1 (indice du points k / nbre de points k). Les colonnes de 2 à NBANDS (nombre de bandes) contiennent les valeurs des bandes d'énergie pour ce point k en eV.
    Si ISPIN = 2, un fichier est créé pour chaque spin.

  - `calcul.extraireBandes( parDirection=True )`

    extrait toutes les bandes par direction dans l'espace réciproque et crée un fichier par direction. Dans chaque fichier les 3 premières colonnes donnent les coordonnées du

points k. Les colonnes de 4 à NBANDS (nombre de bandes) contiennent les valeurs des bandes d'énergie pour ce point k en eV.

- lectureBandes()

  Méthode permettant de lire les bandes d'énergie sur le fichier xml du calcul. Après exécution de cette méthode, on dispose des bandes d'énergie dans la liste Bandes sous la forme :

```
from VASP import *
calcul = VaspRun()
calcul.bandes[ spin ][ point k ][ bandes ][ i ]
```

  spin : 0 ou 1, seule la valeur 0 est disponnible pour les calculs avec spin non polarisé (ISPIN = 1). Pour les calculs spin polarisé (ISPIN = 2), spin = 0 donne les bandes pour les spin alpha et spin = 1 pour les spin beta.

  point k : 0 -> nombre de point K
  le nombre de point k est calcul.nbrePointsK calcul.nbreDirectionsPointsK : nombre de directions de points k calcul.Ndivision : nombre de points k par direction Les points k sont listés direction après direction en donnant l'ensemble des points k sur chaque direction.

  bandes : 0 -> NBANDS-1

  i : 0 ou 1
  i = 0 -> valeurs de l'énergie i = 1 -> occupation de la bande

2.5 Obtenir les densités d'états totale et partielles :

- extraireDOS(partielles = True/False)

  Méthode permettant d'extraire la densité d'états totale et les densités d'états partielles projetées sur les atomes et de les écrire dans un format pratique pour les tracer.

  – `calcul.extraireDOS()` ou `calcul.extraireDOS( partielles=False )`

    extrait la DOS totale dans le fichier "dosTotale.dat". Sur le fichier "dosTotale.dat", la colonne 1 donne l'énergie, la colonne 2 donne la densité d'états et la colonne 3 donne l'intégrale de la densité d'états.
    Si ISPIN = 2, un fichier est créé pour chaque spin.

  – `calcul.extraireDOS( partielles = True )`

    extrait la DOS totale dans le fichier "dosTotale.dat" et les DOS partielles sur des fichiers séparés. Pour les DOS partielles, un fichier par atome est créé portant le nom de l'atome et son numéro correspondant à l'ordre dans lequel ils sont donnés dans le POSCAR. Les fichiers contenant les DOS partielles ont en colonne 1 l'énergie et en colonne 2 -> 9 les DOS partielles projetées sur les OA dans l'ordre : s py pz px dxy dyz dz2 dxz dx2.
    Si ISPIN = 2, un fichier est créé pour chaque spin.

- lectureDOS()

  Lecture de la densité d'états totale et des densités d'états partielles sur le fichier xml du calcul. Après exécution de cette méthode, on dispose de la densité d'états totale dans la liste dosTotale et des densités d'états partielles dans la liste dosPartielles. Elles sont de la forme :

```
from VASP import VaspRun
calcul = VaspRun()
calcul.dosTotale[ spin ][ E ][ i ]
```

spin : 0 ou 1, seule la valeur 0 est disponnible pour les calculs avec spin non polarisé (ISPIN = 1). Pour les calculs spin polarisé (ISPIN = 2), spin = 0 donne les bandes pour les spin alpha et spin = 1 pour les spin beta.

E : indice de parcours de la DOS pour un spin (valeurs de l'énergie)

i : 0 -> 2
i = 0 -> valeurs de l'énergie
i = 1 -> densité d'état
i = 2 -> intégrale de la DOS

```
calcul.dosPartielles[iat][ spin ][ E ][ i ]
```

iat : numéro de l'atome

spin : 0 ou 1, seule la valeur 0 est disponnible pour les calculs avec spin non polarisé (ISPIN = 1). Pour les calculs spin polarisé (ISPIN = 2), spin = 0 donne les bandes pour les spin alpha et spin = 1 pour les spin beta.

e : indice de parcours de la DOS pour un spin (valeurs de l'énergie)

i : 0 -> 8 valeurs de la DOS sur chaque OA
0 = s, 1 = p_y, 2 = p_z, 3 = p_x, 4 = d_{xy}, 5 = d_{yz}, 6 = d_{z^2}, 7 = d_{xz}, 8 = $d_{x^2-y^2}$

3 Liste des attributs

Cette section liste les attributs des objets de type `VaspRun`.

Attribut	***type***	**Définition**
self.fichierXML	*string*	nom du fichier xml
self.xml	*dom*	objet xml créé à partir du fichier xml
self.erreur	*bool*	vrai si le fichier xml est introuvable
self.DOSlue	*bool*	contrôle si la DOS totale a été lue
self.DOSPartiellesLues	*bool*	contrôle si les DOS partielles ont été lues
self.BandesLues	*bool*	contrôle si les bandes d'énergie ont été lues
self.allMotsClefs	*dic*	dictionnaire des mots clefs du calcul
self.INCAR	*dic*	dictionnaire des mots clefs du fichier INCAR
self.groupesMotsClefs	*dic*	dictionnaire des groupes de mots clefs, chaque groupe est repéré par le titre du groupe.
self.Natomes	*int*	Nombre d'atomes dans le calcul
self.Ntypes	*int*	Nombre de type d'atomes dans le calcul
self.atomesDuCalcul	*UnAtome*	Liste des atomes du calcul. Chaque élément de la liste est de type UnAtome et possède les attributs : nom, atomType, nElecValence, masse et pseudo.
self.eFermi	*float*	Energie en eV au niveau de Fermi
self.dosTotale	*list*	Tableau contenant la DOS totale : `dosTotale[spin][E][i]`
self.dosPartielles	*list*	Tableau contenant les DOS partielles : `dosPartielles[iat][spin][E][i]`
self.bandes	*list*	Tableau contenant les bandes d'énergie : `Bandes[spin][point k][bandes][i]`
self.typePointsK	*string*	type de points k : Gamma (grille régulière) ou listgenerated (selon des lignes de symétrie).
self.Ndivision	*int*	Nombre de points k le long de chaque ligne (listgenerated) ou le long de chaque direction (Gamma)
self.nbrePointsK	*int*	Nombre totale de points k
self.nbreDirectionsPointsK	*int*	Nombre de ligne de points k dans le cas listgenerated.
self.PointsK	*list*	Dans le cas où self.typePointsK = listgenerated, self.PointsK est la liste des points k entre lesquels sont construites les lignes de points k.
self.listePointsK	*list*	Liste de longueur self.nbrePointsK contenant les coordonnées des points k