Université de Bordeaux 1
Rapport de stage en laboratoire

Maître de stage : Foerster Dietrich

Mise en oeuvre d'une extention en $O(N^3)$ de la méthode GW de Hedin sur des Cristaux périodiques

Barbry Marc

Bordeaux, du 15 mai au 15 juillet

Remerciements

Je tiens à remercier Mr Foerster, Mr Sanchez et Mr Koval pour m'avoir accepté dans leur projet. Je les remercie également pour le temps qu'ils ont passé pour m'expliquer les problémes liés à leur projet.
Je remercie Mr Koval et Mr Sanchez pour leur hospitalité lors de mon séjour à San Sebastian qui fut très agréable et très instructif.
Je remercie particulierement Mr Foerster pour son aide durant le stage, mais également avant, en particulier pour son soutiens dans mes projets à l'international.

Sommaire

Introduction

J'ai effectué mon stage au LOMA (laboratoire ondes et matières Aquitaine) du 15 mai au 15 juillet, sous la tutelle de Mr Foester. Mr Foester travail actuellement avec Koval Peter et Sanchez Daniel, checheurs à l'université de San Sebastian, sur un programme dont le but est de réduire le temps de calcul de N^4 à N^3 de l'aproximation GW de Hedin (N est le nombre d'atomes) pour des cristaux organiques. Car aujourd'hui, pour déterminer les caractéristiques d'un matériau organique, on utilise la méthode DFT/TDDFT de Kohn-Shame-Gross. Mais cette méthode ne permet pas de déterminer le gap du système. Or la méthode GW de Hedin est utilisée pour calculer la gap des systèmes inorganiques, et il serait intéressant de pouvoir appliquer cette méthode à des matériaux organiques. Mais pour ce faire, il faut tout d'abord réduire les temps de calcul de cette méthode, car les molécules des semi-conducteurs organiques sont bien plus complexes que leurs homologues inorganiques. En particulier il serait envisageable de calculer le gap pour des semi-conducteurs organiques, pour pouvoir par la suite déterminer les meilleurs matériaux pour faire des panneaux solaires avec des coûts relativement bas et des rendements comparables aux semi-conducteurs inorganiques.
Dans ce stage, j'ai étudié les réseaux des cristaux périodiques et les semi-conducteurs organiques, afin de pouvoir comprendre comment on peut réduire le temps de calcul pour déterminer les caractéristiques de ces matériaux, et ensuite j'ai participé à l'élaboration du code dans mes modestes moyens.

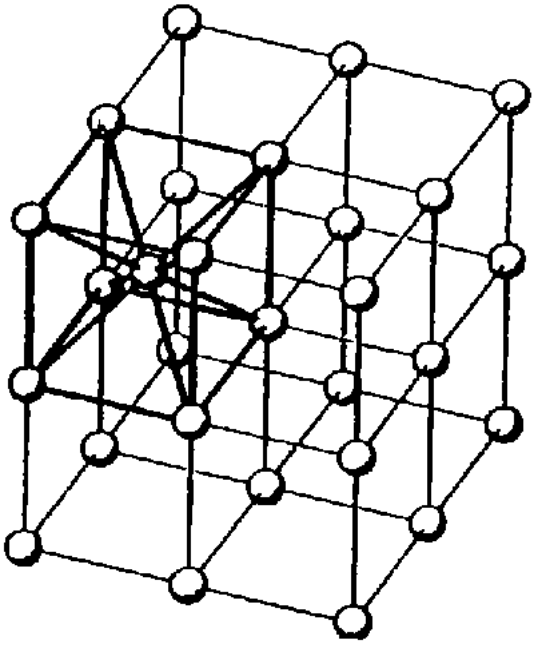

FIGURE 1 – Réseaux cristallin

1 Présentation du Laboratoire

Le laboratoire ondes et matières d'Aquitaine (LOMA) est un des centres de recherche en physique de l'université de Bordeaux 1. Ce laboratoire est articulé autour de deux départements :

- Le département physique de la matière condensée, dont les thèmes principaux sont :
  - Matière molle
  - Écoulement
  - Matériaux et nanostructures
- Le département photonique et matériaux :

– Génération et manipulations de la lumière
– Matériaux dynamiques et nanostructures
– Méthodes innovantes en photoniques

Le laboratoire est constitué d'une centaine de membres dont les 2/3 sont des chercheurs et des enseignants chercheurs. Le laboratoire acceuille également une trentaine de doctorants et post-doctorants. Les membres du laboratoire effectuent, pour la majorité, également des enseignements à l'université en plus de leurs travaux de recherche.
Mr Foerster fait parti du département de physique de la matière condensée, en particulier, son travail fait parti de la thématique Matériaux et nanostructures.
Mon travail était théorique, mes activités lors de ce stage étaient principalement la lecture des concepts que je devais savoir maîtriser dans le but de participer à l'écriture du programme.

2 Bref présentation des semi-conducteurs organiques

2.1 La liason hybride sp_2 du carbone

Les semi-conducteurs organiques sont basés sur les propriétés inhabituelles de l'atome de carbone : entre-autre, il peut former ce que l'on appelle l'hybridation sp_2, où l'orbital sp_2 forme un triangle dans un plan et les orbitales pz sont dans le plan perpendiculaire. Une liaison s entre deux atomes de carbones peut-être créé par la superposition d'une orbitale avec deux orbitales sp_2.
La différence d'énergie entre les niveaux occupés et les niveaux innocupés est assez importante et va bien au delà du spectre visible. Par conséquent, plus les chaînes de carbone liées ont un gap important entre l'orbital moléculaire la plus haute en énergie occupée par un électron (HOMO) et l'orbital moléculaire la plus basse en énergie inoccupé (LUMO), plus le semi-conducteur aura les propriétés d'un isolant.
Cependant, dans l'hybridation sp_2, les orbitales p_z forment des liaisons p. Ces liaisons ont alors une différence d'énergie plus petite entre HOMO et LUMO, ce qui conduit à une forte absorption et à un spectre proche de celui du visible et donc les propriétés du système tend vers celui d'un semi-conducteur.

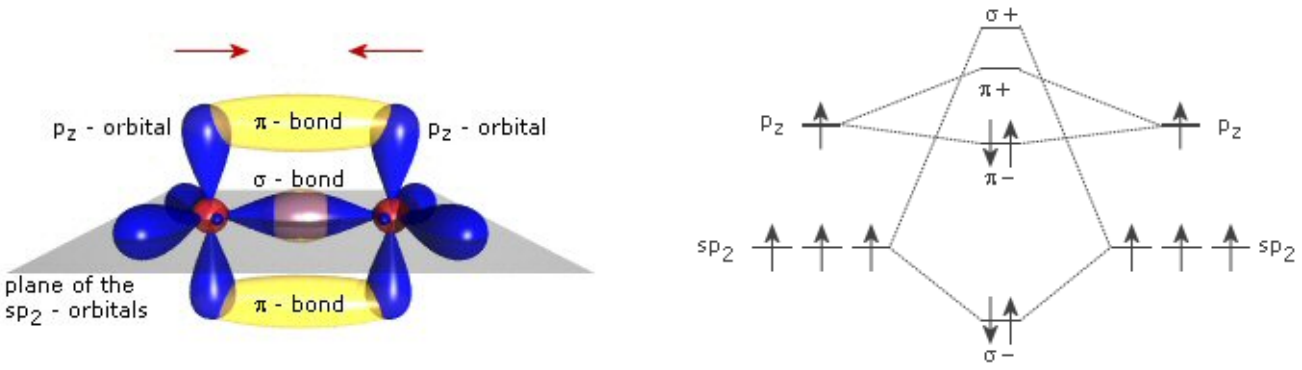

FIGURE 2 – Schémas des orbitales et des liasons pour deux hybridations sp_2 d'atomes de carbone

2.2 Les polymères

Si une longue chaîne de carbone est formé, les liasons p deviennent délocalisé le long de la chaine et forme un système électronique à une dimension La bande à une dimension qui en résulte a une très grande largeur de bande (à l'échelle de l'électron volt). C'est à dire, que nous avons un semi-conducteur à une dimension avec une bande de valence remplie qui correspond à HOMO et une bande de conduction vide qui correspond à LUMO. Les propriétés de transport d'un tel polymère sont habituellement déterminé par des défauts dans les chaînes 1-D ou en sautant d'une chaîne à l'autre. Par conséquent, les échantillons ne présentent pas de transport de bande mais sont thermalement activé par saut. Les polymères sont habituellement déposés par des procédés humides, comme le spin-coating ou l'impression.

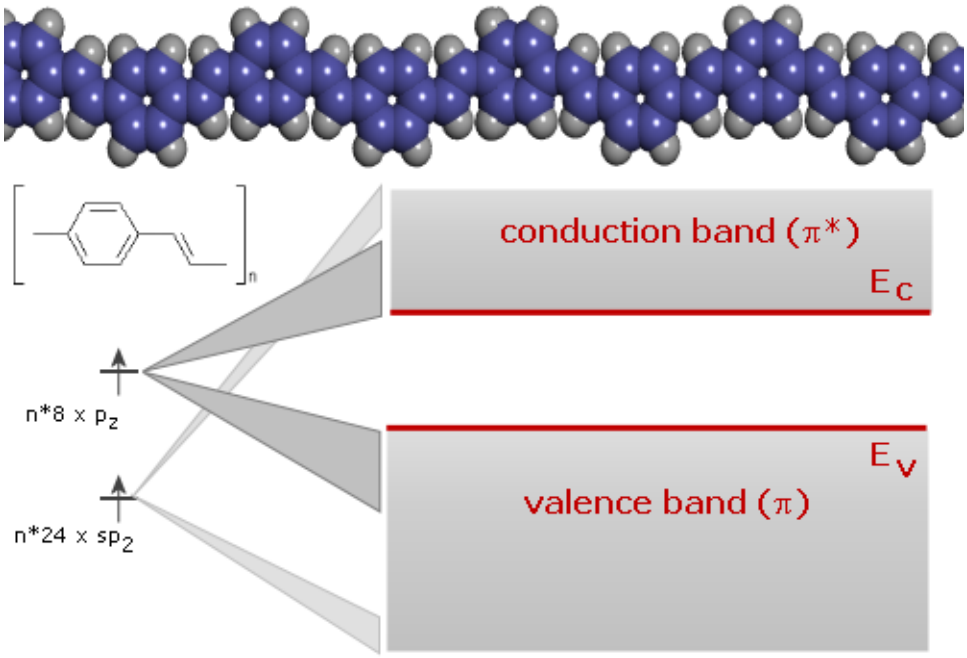

FIGURE 3 – Schéma d'un polymère et de son énergie de structure

3 L'approximation GW de Hedin

Dans cette partie je vais faire une courte présentation de l'approximation GW de Hedin. Car il s'agit du théme central du projet, bien que je n'ai pu l'étudier en détail à cause de sa complexité.
L'approximation GW (GWA) de Hedin est une approximation dont le but est de calculer la self-energy Σ d'un systeme. La self-energy représente la contribution de l'énergie de la particule ou de sa masse effective dû à l'interaction entre la particule et le système.

$$\Sigma = GW \tag{1}$$

Où :

- G sont les fonctions de Green [1]
- W est l'interaction dynamique écrantée

$$G(rt, r't') = -i < \psi_0 | T \left[\psi(rt) \psi^\dagger(r't') \right] | \psi_0 > \tag{2}$$

1. Pour plus de détail, voir l'annexe sur les fonctions de Green.

Avec :

- $|\psi_0>$ qui représente l'état fondamental du système
- ψ et $\psi^\dagger$ sont les opérateurs annihilateurs et créateurs d'électrons.
-

$$\begin{aligned} T\left[A(t)B(t')\right] &= \pm A(t)B(t') \qquad si \qquad t > t' \\ &= \pm B(t')A(t) \qquad si \qquad t < t' \end{aligned}$$

- si bosons
+ si fermions

$$W(r,r',\omega) = \int dr" \epsilon^{-1}(r,r",\omega) v(r,r") \tag{3}$$

Où ϵ^{-1} est l'inverse de la fonction diélectrique.

4 Construction des produits dominants de Bloch

Comme ce projet est très complexe, il est basé sur les travaux précédents de Mr Foester, Mr Koval et Mr Sanchez et a demandé plusieurs années de travail et de collaboration, il m'est tout simplement impossible d'en comprendre tous les aspects en seulement deux mois, et je me bornerai bien entendu à décrire ma modeste contribution.
Mon premier travail à consisté à mettre au point un programme qui permet de construire les produits dominants de Bloch.
Je vais donc faire d'abord une présentation théorique des fonctions de Bloch qui nous permettra ensuite d'aborder la construction du programme.

4.1 Présentation théorique :

4.1.1 Définition des orbitales de Bloch

Nous supposons les translations invariantes : $\overrightarrow{a_i}$ avec $i = 1, 2, 3$.
Les orbitales de Bloch sont définies par [8] [6] :

$$\widetilde{f}^a_{\overrightarrow{p}}(\overrightarrow{r}) = \sum_{n_i=-\infty}^{+\infty} f^a\left(\overrightarrow{r} - n_1\overrightarrow{a_1} - n_2\overrightarrow{a_2} - n_3\overrightarrow{a_3}\right) e^{-i(p_1n_1+p_2n_2+p_3n_3)} \tag{4}$$

Où :

- $f^a(\overrightarrow{r})$ est l'orbitale atomique de la cellule unité.
- les $\overrightarrow{a_i}$ sont les translations dans les différentes directions de l'espace.
- les $\overrightarrow{p_i}$ sont des facteurs de phase.

Par construction les orbitales de Bloch sont périodique par translation :

$$\widetilde{f}^a_{\overrightarrow{p}}(\overrightarrow{r} + \overrightarrow{a}_i) = e^{-ip_i}\widetilde{f}^a_{\overrightarrow{p}}(\overrightarrow{r}) \tag{5}$$

Les orbitales de Bloch n'ont pas de sens physique, mais elles sont nécessaires à la construction de la théorie. On peut montrer [7] que les orbitales de Bloch $\widetilde{f}^{a*}_{\overrightarrow{p}}$, $\widetilde{f}^b_{\overrightarrow{q}}$ pour différentes phases ($\overrightarrow{p} \neq \overrightarrow{q}$) sont mutuellement orthogonales :

$$\int d^3r \widetilde{f}^{a*}_{\overrightarrow{p}}(\overrightarrow{r})\widetilde{f}^b_{\overrightarrow{q}}(\overrightarrow{r}) = (2\pi)^3 \prod_{i=1}^{3} \delta_{circle}(p_i - q_i)\gamma^{ab}(\overrightarrow{p}) \tag{6}$$

Les orbitales de Bloch de même moment ont une métrique :

$$\gamma^{ab}(\overrightarrow{p}) = \sum_{n_i=-\infty}^{+\infty} e^{-i(p_1n_1+p_2n_2+p_3n_3)} \int f^{*a}(\overrightarrow{r}) f^b\left(\overrightarrow{r} - n_1\overrightarrow{a_1} - n_2\overrightarrow{a_2} - n_3\overrightarrow{a_3}\right) dr \tag{7}$$

4.1.2 Construction des orbitales de Bloch

Dans le but d'appliquer la méthode LCAO [2] à un cristal [1] [4] [5], nous voulons exprimer l'opérateur du champ d'un électron en terme des opérateurs d'électrons de Bloch comme $\psi(\overrightarrow{r},t) = \int dp \widetilde{f}^{a}_{\overrightarrow{p}}(\overrightarrow{r}) c_a(\overrightarrow{p},t)$, Ce qui conduit à la densité électronique :

$$n(r,t) \;=\; \psi^{\dagger}(r,t)\psi(r,t) = \int dp_1 dp_2 \widetilde{f}^{a*}_{\overrightarrow{p1}}(\overrightarrow{r}) \widetilde{f}^{b}_{\overrightarrow{p2}}(\overrightarrow{r}) c^{\dagger}_a(p_1,t) c_b(p_2,t) \tag{8}$$

Comme pour les résaux, il y a beaucoup trop de produits $\widetilde{f}^{a*}_{\overrightarrow{p1}}(\overrightarrow{r}) \widetilde{f}^{b}_{\overrightarrow{p2}}(\overrightarrow{r})$, nous pourrions chercher la dépendance linéaire de ces produits directement dans l'espace des impulsions mais il est moins difficile d'exprimer les produits des orbitales de Bloch avec l'expression :

$$\widetilde{f}^{a}_{\overrightarrow{p}}(\overrightarrow{r}) \;=\; \sum_{n_i=-\infty}^{+\infty} f^a(\overrightarrow{r} - n_i \overrightarrow{a}_i) e^{-i\overrightarrow{p}n} \tag{9}$$

$$\Longrightarrow \widetilde{f}^{a*}_{\overrightarrow{q}_1}(\overrightarrow{r}) \widetilde{f}^{b}_{\overrightarrow{q}_2}(\overrightarrow{r}) \;=\; \sum_{m_i,n_i=-\infty}^{+\infty} e^{-i(\overrightarrow{q}_1\overrightarrow{m} + \overrightarrow{q}_2\overrightarrow{n})} f^a(\overrightarrow{r} - m_i\overrightarrow{a}_i) f^b(\overrightarrow{r} - n_i\overrightarrow{a}_i) \tag{10}$$

Ici, $e^{-i\overrightarrow{p}.\overrightarrow{n}}$ signifie $exp(-i(p_1n_1 + p_2n_2 + p_3n_3))$. Dans leurs précédents travaux [2], Mr Foerster, Mr Coulaud et Mr Koval ont montré que :

$$f^a(\overrightarrow{r} - m_i\overrightarrow{a}_i) f^b(\overrightarrow{r} - n_i\overrightarrow{a}_i) \;=\; \sum_{\mu,k} V\left(a,\overrightarrow{m}|b,\overrightarrow{n}|\mu,\overrightarrow{k}\right) F^{\mu}(\overrightarrow{r} - k_i\overrightarrow{a}_i) \tag{11}$$

Ici les orbitales a, b et μ appartiennent généralement à différentes cellules élémentaires. Parce que les translations sont invariantes, il est suffisant pour connaître les coefficients $V\left(a,\overrightarrow{m}|b,\overrightarrow{n}|\mu,\overrightarrow{k}\right)$ pour le centre de la fonction $F^{\mu}(\overrightarrow{r} - k_i\overrightarrow{a}_i)$, de travailler dans la cellule $\overrightarrow{k} = 0$. Car, en utilisant l'invariance par translation, nous pouvons trouver tous les coefficients grâce au sous-ensemble $\overrightarrow{k} = 0$

$$V\left(a,\overrightarrow{m}|b,\overrightarrow{n}|\mu,\overrightarrow{k}\right) \;=\; V\left(a,\overrightarrow{m} - \overrightarrow{k}|b,\overrightarrow{n} - \overrightarrow{k}|\mu,0\right) \tag{12}$$

Nous pouvons construire les coefficients $V\left(a,\overrightarrow{m}|b,\overrightarrow{n}|\mu,\overrightarrow{k}\right)$, ou leurs équivalents $V(a,\overrightarrow{m}|b,\overrightarrow{n}|\mu,0)$ en utilisant les techniques dévelloppées dans l'article [2]. En combinant les équations (9) et (11), nous trouvons

$$\widetilde{f}^{a}_{\overrightarrow{q1}}(\overrightarrow{r}) \widetilde{f}^{b}_{\overrightarrow{q2}}(\overrightarrow{r}) \;=\; \sum_{\overrightarrow{m}_1,\overrightarrow{m}_2=-\infty}^{+\infty} \sum_{\overrightarrow{k}} e^{-i(\overrightarrow{q}_1\overrightarrow{m}_1 + \overrightarrow{q}_2\overrightarrow{m}_2)} V\left(a,\overrightarrow{m}_1|b,\overrightarrow{m}_2|\mu,\overrightarrow{k}\right) F^{\mu}(\overrightarrow{r} - k_i\overrightarrow{a}_i) \tag{13}$$

En utilisant l'invariance par translation des coefficients (12), nous pouvons réecrire l'équation (13) comme

$$\widetilde{f}^{a}_{\overrightarrow{q1}}(\overrightarrow{r}) \widetilde{f}^{b}_{\overrightarrow{q2}}(\overrightarrow{r}) \;=\; \sum_{\overrightarrow{m}_1,\overrightarrow{m}_2=-\infty}^{+\infty} \sum_{\overrightarrow{k}} e^{-i(\overrightarrow{q}_1\overrightarrow{m}_1 + \overrightarrow{q}_2\overrightarrow{m}_2)} V\left(a,\overrightarrow{m}_1 - \overrightarrow{k}|b,\overrightarrow{m}_2 - \overrightarrow{k}|\mu,0\right) F^{\mu}(\overrightarrow{r} - k_i\overrightarrow{a}_i) \tag{14}$$

En effectuant les changements de variables $m_{1,2} \longrightarrow m_{1,2} + k$, on obtient

$$\begin{aligned} \widetilde{f}^{a}_{\overrightarrow{q1}}(\overrightarrow{r}) \widetilde{f}^{b}_{\overrightarrow{q2}}(\overrightarrow{r}) = & \sum_{\overrightarrow{m}_1,\overrightarrow{m}_2=-\infty}^{+\infty} e^{-i(\overrightarrow{q}_1\overrightarrow{m}_1 + \overrightarrow{q}_2\overrightarrow{m}_2)} V(a,\overrightarrow{m}_1|b,\overrightarrow{m}_2|\mu,0) \\ & \times \sum_{\overrightarrow{k}} e^{-i(\overrightarrow{q}_1 + \overrightarrow{q}_2)\overrightarrow{k}} F^{\mu}(\overrightarrow{r} - k_i\overrightarrow{a}_i) \end{aligned} \tag{15}$$

2. Voir annexe LCAO.

On reconnait dans (15) la forme d'un produit de Bloch périodique :

$$\widetilde{F}^{\mu}_{\overrightarrow{q}}(\overrightarrow{r}) \quad = \quad \sum_{\overrightarrow{n}} e^{-i\overrightarrow{q}\,\overrightarrow{n}} F^{\mu}(\overrightarrow{r} - \overrightarrow{n}\,\overrightarrow{a}) \tag{16}$$

Où $\overrightarrow{n}\,\overrightarrow{a} = n_1\overrightarrow{a}_1 + n_2\overrightarrow{a}_2 + n_3\overrightarrow{a}_3$. Les vecteurs $\overrightarrow{a}_i$ correspondent aux vecteurs élémentaires du réseau et $(n_1, n_2, n_3) \in \mathbb{N}$. Avec $\widetilde{F}^{*\mu}_{\overrightarrow{q}}(\overrightarrow{r}) = \widetilde{F}^{\mu}_{-\overrightarrow{q}}(\overrightarrow{r})$. En d'autres termes, nous avons trouvé la décomposition des fonctions de Bloch :

$$\widetilde{f}^{a}_{\overrightarrow{q}_1}(\overrightarrow{r})\widetilde{f}^{b}_{\overrightarrow{q}_2}(\overrightarrow{r}) \quad = \quad \sum_{\mu} \widetilde{V}^{a,b}_{\mu}(\overrightarrow{q}_1, \overrightarrow{q}_2)\, \widetilde{F}^{\mu}_{\overrightarrow{q}_1+\overrightarrow{q}_2}(\overrightarrow{r}) \tag{17}$$

Où $\widetilde{V}^{a,b}_{\mu}(\overrightarrow{q}_1, \overrightarrow{q}_2)$ est la transformée de Fourier des coefficients locaux du vertex :

$$\widetilde{V}^{a,b}_{\mu}(\overrightarrow{q}_1, \overrightarrow{q}_2) \quad = \quad \sum_{\overrightarrow{m}_1, \overrightarrow{m}_2 = -\infty}^{+\infty} e^{-i(\overrightarrow{q}_1\overrightarrow{m}_1 + \overrightarrow{q}_2\overrightarrow{m}_2)} V(a, \overrightarrow{m}_1 | b, \overrightarrow{m}_2 | \mu, 0) \tag{18}$$

Où :

- a, b etμ sont des indices pour repérer les orbitales de chaque atome des cellules concernées.
- a est associé à la première cellule.
- b est associé à la seconde cellule.
- μ est associé à la cellule 0.($\overrightarrow{k} = 0$)
- $\overrightarrow{q}_1$ et $\overrightarrow{q}_2$ sont des impulsions.
- $\overrightarrow{m}_1 = k_1\overrightarrow{a}_1 + k_2\overrightarrow{a}_2 + k_3\overrightarrow{a}_3$, où $(k_1, k_2, k_3) \in \mathbb{N}$
- $\overrightarrow{m}_1 = k'_1\overrightarrow{a}_1 + k'_2\overrightarrow{a}_2 + k'_3\overrightarrow{a}_3$, où $(k'_1, k'_2, k'_3) \in \mathbb{N}$

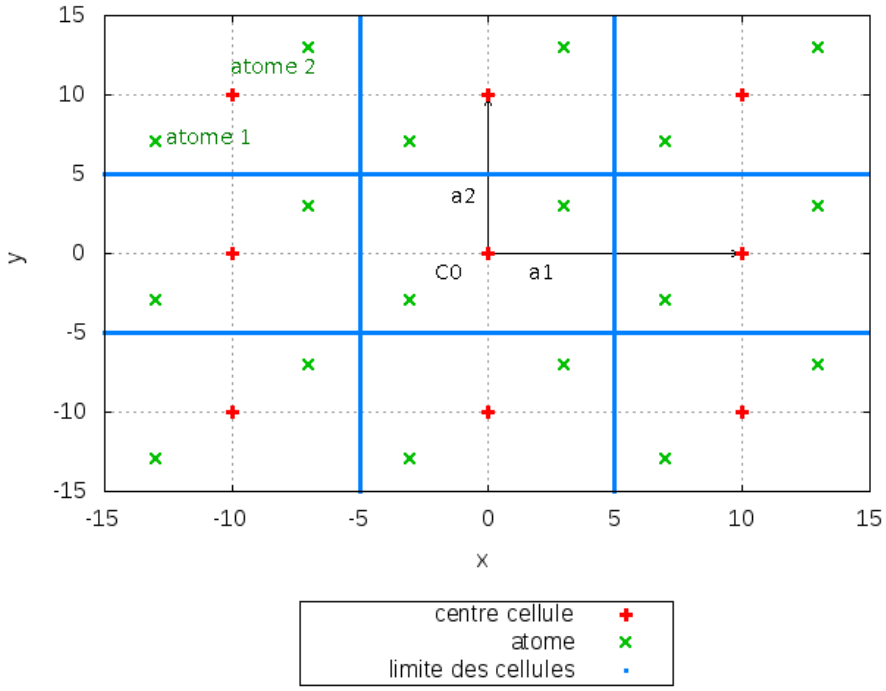

FIGURE 4 – Réseau à deux dimensions, en bleue apparait les limites des cellules

Avec

$$\widetilde{V}_{\mu}^{a,b}\left(\overrightarrow{q}_1;\overrightarrow{q}_2\right)^* = \widetilde{V}_{\mu}^{a,b}\left(-\overrightarrow{q}_1,-\overrightarrow{q}_2\right)$$

Or on a :

$$\widetilde{f}_{\overrightarrow{q}}^{a*}\left(\overrightarrow{r}\right) = \widetilde{f}_{-\overrightarrow{q}}^{a}\left(\overrightarrow{r}\right)$$

On obtient alors :

$$\widetilde{f}_{\overrightarrow{q}_1}^{a*}\left(\overrightarrow{r}\right)\widetilde{f}_{\overrightarrow{q}_2}^{b}\left(\overrightarrow{r}\right) = \widetilde{f}_{-\overrightarrow{q}_1}^{a}\left(\overrightarrow{r}\right)\widetilde{f}_{\overrightarrow{q}_2}^{b}\left(\overrightarrow{r}\right) \quad = \quad \sum_{\mu}\widetilde{V}_{\mu}^{a,b}\left(-\overrightarrow{q}_1,\overrightarrow{q}_2\right)\widetilde{F}_{-\overrightarrow{q_1}+\overrightarrow{q_2}}^{\mu}(\overrightarrow{r}) \tag{19}$$

4.1.3 L'expression de la densité électronique

Il est intéressant de voir comment cette décomposition peut être utilisée pour la densité électronique que l'on avait exprimée avec un produit des fonctions de Bloch :

$$n(r,t) \quad = \quad \psi^{\dagger}(r,t)\psi(r,t) = \int dp_1 dp_2 \widetilde{f}_{\overrightarrow{p}_1}^{a*}(\overrightarrow{r})\widetilde{f}_{\overrightarrow{p}_2}^{b}(\overrightarrow{r})c_a^{\dagger}(p_1,t)c_b(p_2,t) \tag{20}$$

En utilisant l'équation (19) on obtient :

$$n(\overrightarrow{r};t) = \int dq_1 dq_2 \sum_{\mu}\widetilde{V}_{\mu}^{a,b}\left(-\overrightarrow{q_1};\overrightarrow{q_2}\right)\widetilde{F}_{-\overrightarrow{q_1}+\overrightarrow{q_2}}^{\mu}(\overrightarrow{r})c_a^{\dagger}(\overrightarrow{q}_1,t)c_b(\overrightarrow{q}_2,t) \tag{21}$$

Ceci montre que la décomposition des fonctions de Bloch peut-être utilisée pour exprimer la densité électronique. Plus généralement, les fonctions $\widetilde{F}_{\overrightarrow{p}}^{\mu}(\overrightarrow{r})$ nous assurent une base de tenseurs pour la réponse en densité d'un système périodique. Comme les fonctions de Bloch originales, les fonctions $\widetilde{F}_{-\overrightarrow{p_1}+\overrightarrow{p_2}}^{\mu}(\overrightarrow{r})$ n'ont pas de sens physique, mais elles sont nécessaires avec les coefficients $\widetilde{V}_{\mu}^{a,b}\left(-\overrightarrow{p_1};\overrightarrow{p_2}\right)$, pour écrire la densité.

4.2 Construction de la base des fonctions de Bloch

4.2.1 Description du système

Les systèmes étudiés sont des réseaux de cristaux périodiques, le système est donc constitué d'un motif qui ce répète un très grand nombre de fois. Nous désignerons ce motif sous le terme de cellule. Chaque cellule posséde un certain nombre d'atomes (nombre qui est identique pour chaque cellule, puisque les cellules sont identiques). La cellule au centre du système est appelé cellule 0. On positionera l'origine de notre repére au centre de la cellule 0. Chaque centre de cellule sera repéré par un numéro, 3 indices (i, j, k), ainsi que par ses coordonnées (x, y, z).

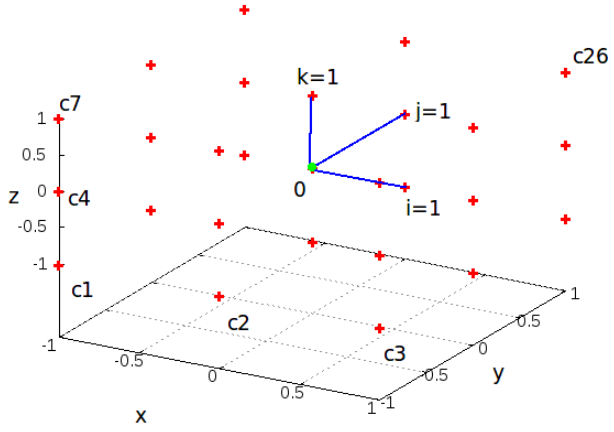

FIGURE 5 – Réseau cubique

Toutes les caractéristiques du système sont enregistrées dans un fichier dont l'extension est .fdf dont la forme est :
BN_chain.fdf :

```
#NB chain

NumberOfAtoms              2
NumberOfSpecies            2
AtomicCoordinatesFormat  Fractional

#pas du reseaux
LatticeConstant            3.43 Ang

#choix des donnees en sorties
COOP.write                 T
SolutionMethod             diagon
Use.New.Diagk              T
WriteEigenvalues           T
WriteWaveFunctions         T

#description des espesces presente
%block ChemicalSpeciesLabel
1        5        B
2        7        N
%endblock ChemicalSpeciesLabel
```

```
%block AtomicCoordinatesAndAtomicSpecies
0.00     0.00     0.00     1
0.025    0.025    0.025    2
%endblock AtomicCoordinatesAndAtomicSpecies

%block LatticeVectors
0.0      0.5      0.5
0.5      0.0      0.5
0.5      0.5      0.0
%endblock LatticeVectors

%block BandPoints
0.00     0.00     0.00
1.00     0.00     0.00
0.50     0.50     0.50
%endblock BandPoints
```

Ce fichier est alors exploité par le programme Siesta [3], qui nous donne en sortie toutes les informations dont nous avons besoin pour traiter le système, et en particulier ici, déterminer la base des fonctions de Bloch.

4.2.2 Première étape vers un code pour construire $\widetilde{V}_{\mu}^{a,b}(p_1,p_2)$ et $\widetilde{F}_{p}^{\mu}(r)$

Les fonctions dominantes de Bloch $\widetilde{F}_{\overrightarrow{p}}^{\mu}(\overrightarrow{r})$ et les coefficients $\widetilde{V}_{\mu}^{a,b}(\overrightarrow{p_1};\overrightarrow{p_2})$ sont définis comme suit :

$$\widetilde{V}_{\mu}^{a,b}(\overrightarrow{p_1},\overrightarrow{p_2}) = \sum_{\overrightarrow{m}_1,\overrightarrow{m}_2=-\infty}^{+\infty} e^{-i(\overrightarrow{p}_1\overrightarrow{m}_1+\overrightarrow{p}_2\overrightarrow{m}_2)} V(a,\overrightarrow{m}_1|b,\overrightarrow{m}_2|\mu,0) \quad (22)$$

$$\widetilde{F}_{\overrightarrow{p}}^{\mu}(\overrightarrow{r}) = \sum_{n} e^{-ipn} F^{\mu}(\overrightarrow{r}-n_i\overrightarrow{a}_i) \quad (23)$$

La première étape pour construire la base des fonctions de Bloch consiste à établir une liste des cellules qui participent à la construction de la base. En effet, dans la pluspart des cas, beaucoup de cellules ne participent pas à la construction de la base des fonctions de Bloch.
Tout d'abord, les centres des cellules doivent appartenir à une sphère dont le centre est confondu avec l'origine du repére (centre de la cellule zéro) et le rayon est égal au rayon de l'orbitale maximal des atomes, plus deux fois la distance maximale caractéristique des cellules (cette distance est par exemple la diagonale du cube dans le cas d'un réseaux cubique). Si le centre des cellules est en dehors de cette sphère, alors le coefficient $\widetilde{V}_{\mu}^{a,b}(-\overrightarrow{p_1};\overrightarrow{p_2})$ peut-être négligé.
Ensuite, il faut tester pour tous les atomes de chaque cellule appartenant à la sphère :
Pour M le milieu de la droite AB, où :
A est la position de $atome_i \in C_q$
B est la position de $atome_j \in C_p$, avec $p \neq q$.
La cellule participe à la construction de la sphère si :

$$M \in C_0$$

Le programme que j'ai écrit construit la liste des cellules qui respecte ces conditions.

4.2.3 Description du programme

Dans cette partie, je vais présenter les parties importantes du programme.
Tout d'abord mon programme est divisé en différents modules, mais il utilise également des modules écrits par Peter Koval. Voici le programme principal :
minimal_cell.F90 :

```
1 program minimal_cell
2
3    use modul_freq_vars, only : initialize_freq_vars
4    use modul_import, only : import_from_siesta
5    use modul_input, only : input_iv, input_calc_param_from_file
6    use modul_log, only : init_logs, ilog, log_timing_note
7    use modul_parallel, only : mynode, end_parallel, init_parallel
8    use modul_init_dft_data, only : init_dft_data
9    use modul_globals
10   use modul_append
11   use modul_minimal_cell
12   use modul_diag_hamiltonien
13   use modul_dipoles
14
15   implicit none
16   integer :: iv = 1
17   real(8) :: start_time, finish_time
18
19   call cputime(start_time);
20   call init_parallel(iv);                        ! parallel initialization
21   call init_logs(-1, '.MINI.txt', mynode, iv); ! initialize log file, times file, etc
22   call input_calc_param_from_file(iv);      ! read input parameters from ./tddft_lr.inp
23   iv = input_iv
24   call import_from_siesta(iv);                   ! import data from SIESTA
25   call init_dft_data(iv);                        ! initializes some internal data
26
27   call construct_minimal_cell(iv);
28   call diag_hamil(iv);
29
30   call cputime(finish_time);
31   call log_timing_note('minimal total ', finish_time-start_time, 0, iv);
32   if(iv>0)write(ilog,*) 'minimal: Voila :-)';
33
34
35 end program !minimal_cell
```

Dans ce fichier, seules les subroutines construct_minimal_cell et diag_hamil, ainsi que les modules modul_globals, modul_minimal_cell, modul_append et modul_diag_hamiltonien, ont été écrites de ma main, toutes les autres parties ont été écrites par Peter Koval et servent principalement ici à récupérer les données calculés par le logiciel Siesta. Ce programme se contente d'appeler les modules qui effectuent les calculs, seules les lignes 27 et 28 nous intéresseront donc.
La ligne 27 permet d'appeler le module qui construit la liste des cellules qui nous intéressent.
modul_minimal_cell.f90 :
Le programme a besoin des modules suivants :

```
1 !modul initialiser par peter et completer par marc afin de calculer les produits
   des fonctions de Bloch
2 module modul_minimal_cell
3 contains
4
5 subroutine construct_minimal_cell(iv)
6   use modul_orbital_vars
7   use modul_globals
8   use modul_append
9   use modul_dipoles
10  use modul_general_vertex
```

Le module modul_orbital_vars permet de récupérer des données calculées par Siesta comme le pas du réseau, l'agencement des cellules entre elles, les coordonnées des atomes dans les cellules ...

Le module modul_globals définit les constantes générales utilisées pour les calculs comme la valeur de π.
Le module modul_append appelle la subroutine append_tabular qui permet d'ajouter un élement à un tableau dont on ne connait pas à l'avance le nombre d'élements qu'il contiendra, cette fonction est inspirée de la commande LIST.append() du langage Python.
Les modules modul_dipoles et modul_general_vertex sont des modules écrits par P.Koval et qui servent à calculer les coefficients $\widetilde{V}_{\mu}^{a,b}(\overrightarrow{p_1},\overrightarrow{p_2})$ et les fonctions élementaires $\widetilde{F}_{\overrightarrow{p}}^{\mu}(\overrightarrow{r})$
Comme le programme est très long, je ne présenterai que les parties principales. La partie qui suit permet de construire l'ensemble du réseau et commence le tri des cellules :

```
164   do i = dimen, -dimen, -1
165        do j = -dimen, dimen
166             do k = -dimen, dimen
167                  !calcul des numeros des cellules
168                  if(compt <= ((dimen*2 +1)**3)/2) then
169                          plan(i, j, k)%numero = compt
170                  elseif(compt > ((dimen*2 + 1)**3)/2 + 1) then
171                          plan(i, j, k)%numero = compt-1
172                   else
173                           plan(i, j, k)%numero = 0
174                   endif
175                   compt = compt + 1
176                   !coordonnees des cellules
177                   plan(i, j, k)%x = j*A(1) + i*A(2) + k*A(3)
178                   plan(i, j, k)%y = j*B(1) + i*B(2) + k*B(3)
179                   plan(i, j, k)%z = j*C(1) + i*C(2) + k*C(3)
180                   !coordonnee des atomes
181                   allocate(plan(i, j, k)%pos_atoms_x(1:natoms))
182                   allocate(plan(i, j, k)%pos_atoms_y(1:natoms))
183                   allocate(plan(i, j, k)%pos_atoms_z(1:natoms))
184                   allocate(plan(i, j, k)%pos_atoms_norme(1:natoms))
185                   !coordonnees des atomes par rapport a la cellule 0
186                   do l = 1, natoms
187                         plan(i, j , k)%pos_atoms_x(l) = plan(i, j, k)%x + list_atoms(l)%x
188                         plan(i, j , k)%pos_atoms_y(l) = plan(i, j, k)%y + list_atoms(l)%y
189                         plan(i, j , k)%pos_atoms_z(l) = plan(i, j, k)%z + list_atoms(l)%z
190                         plan(i, j, k)%pos_atoms_norme(l) = sqrt(plan(i, j , k)%pos_atoms_z(l)**2 + &
191                                        &plan(i, j , k)%pos_atoms_y(l)**2 + plan(i, j , k)%pos_atoms_x(l)**2)
192                   enddo
193                   !print*, 168
194                   !distances des cellules a la cellule 0
195                   plan(i, j, k)%norme = sqrt((plan(i, j, k)%x)**2 + (plan(i, j, k)%y)**2 + plan(i, j, k)%z**2)
196                   !determinations des cellules appartenants a la sphere ou non
197                   if((plan(i, j, k)%norme-r_sph>0).and.(plan(i, j, k)%norme-r_sph<(pas1+pas2+pas3)/2)) then
198                             call append_tabular(list_non_sphere, plan(i, j, k)%x, plan(i, j, k)%y, plan(i, j, k)%z,&
199                         & i, j, k, plan(i, j, k)%norme, plan(i, j, k)%numero, compt1)
200                             compt1 = compt1 + 1
201                    endif
202                    if ((plan(i, j, k)%norme-r_sph)<0) then
203                             call append_tabular(list_sphere, plan(i, j, k)%x, plan(i, j, k)%y, plan(i, j, k)%z,&
204                         & i, j, k, plan(i, j, k)%norme, plan(i, j, k)%numero, compt2)
205                             compt2 = compt2 + 1
206                     endif
207             enddo
208        enddo
209   enddo
```

Ici on a donc une triple boucle pour pouvoir se déplacer dans les trois directions de l'espace. Le premier if permet de numéroter les cellules sachant que la cellule C_1 à pour indices $i = dimen$, $j = -dimen$, $k = -dimen$ où dimen donne la dimension du système. Par exemple, si $dimen = 3$, on aura 7 cellules selon chaque axe (les indices variants de $-dimen$ à $+dimen$), ce qui nous fera un total de $7 \times 7 \times 7 = 343$ cellules. Il faut aussi noter que la cellule 0 doit être au centre du réseau.
Ensuite $plan(i, j, k)$ est un type dérivé correspondant à une matrice de dimension 3 qui permet de décrire les cellules, c'est à dire qu'a chaque case du tableau plan correspond :

- un numéro de cellule

- 3 coordonnées x, y, z
- la norme du centre de la cellule
- 3 listes pos_atom_x, y, z qui donnent les coordonnées de chaque atome de la cellule
- une liste de la norme de chaque atome

Donc on donne au centre de la cellule ses coordonnées, puis les coordonnées de chaque atome de la cellule, puis sa norme, puis la norme de chaque atome. Enfin on effectue un test pour savoir si la cellule en question appartient à la sphère (on l'ajoute alors à la liste list_sphere() grâce à la subroutine append_tabular) ou non (on l'ajoute à la liste list_non_sphere()).
À partir de la liste list_sphere(), on construit 3 nouvelles listes (une pour chaque indice), qui donnent le numéro des indices contenus dans la sphere. Puis grâce au code suivant, on détemine le numéro des indices qui délimite la sphère (mais qui sont contenus dans la sphère).

```
224         !determinations des numeros de colonne et de ligne qui sont
            !dans la sphere mais pres de la surface
225         val_max_j = maxval(list_j_sphere)
226         val_max_i = maxval(list_i_sphere)
227         val_max_k = maxval(list_k_sphere)
```

Après on détermine les indices des cellules qui sont à l'extérieur de la sphère, grâce à la condition suivante :

```
230         compt = 0
231         !determination de toutes les cellule entourant la sphere
232         do i = 0, 900
233                 if(abs(list_non_sphere(i, 6))>val_max_j) then
234                         call append_tabular_2(list_colonne_OK,
                        list_non_sphere(i, 1), list_non_sphere(i, 6), compt)
235                         compt = compt + 1
236                 endif
237         enddo
238         k1 = compt
```

On répète cet algorithme pour chaque indice.
Enfin, on détermine les indices maximaux des cellules qui délimitent la sphère.

```
260         !indices entourant la sphere
261         j_droit = list_colonne_OK(0, 2)
262         if (j_droit>0) then
263                 j_gauche = -j_droit
264         else
265                 j_gauche = j_droit
266                 j_droit = - j_droit
267         endif
```

On répète également ceci pour chaque indice. On a donc ici les indices extrèmes qui entourent la sphère, il ne reste alors plus qu'à construire le cube qui délimite la sphère en prennant les cellules dont deux indices sont maximaux et le troisième varit entre ces deux valeurs extrêmes.
Prenons par exemple, le cas des cellules tels que $k = k\_der$ et $i = i\_haut$. Alors j varie entre j_droit et j_gauche. On a alors construit une droite. Après en faisant varier par exemple i, on obtient un plan etc..
La liste list_cellule_lim correspond à la liste des cellules qui délimitent la sphère, et list_cellule_complete correspond à la liste de toutes les cellules contenues dans le cube.
Voici l'algorithme qui permet de réaliser ceci :

```
295         !remplissage de list_cellule
296         do j = j_gauche, j_droit
297                 do i = i_bas, i_haut
298                         do k = k_der, k_dev
299                                 call append_tabular_3(list_cellule_complete, plan(i, j, k)%x, plan(i, j, k)%y,&
300                                 & plan(i, j, k)%z, plan(i, j, k)%numero, compt1)
301                                 compt1 = compt1 + 1
```

```
302                             if(j == j_gauche.Or.j == j_droit) then
303                             !if((i == i_haut.or.i == i_bas).and.(k/=k_dev.or.k/=k_der)) then
304                                     call append_tabular_3(list_cellule_lim, plan(i, j, k)%x, plan(i, j, k)%y,&
305                                     & plan(i, j, k)%z, plan(i, j, k)%numero, compt)
306                                     compt = compt + 1
307                             endif
308                     enddo
309             enddo
310     enddo
311     !print*, 258
312     do i = i_bas, i_haut
313             do j = j_gauche, j_droit
314                     do k = k_der, k_dev
315                             if((i == i_haut.or. i == i_bas).and.(j/=j_droit.or.j/=j_gauche)) then
316                                      !print*, "?2"
317                                     call append_tabular_3(list_cellule_lim, plan(i, j, k)%x, plan(i, j, k)%y,&
318                                     & plan(i, j, k)%z, plan(i, j, k)%numero, compt)
319                                     compt = compt + 1
320                             endif
321                     enddo
322             enddo
323     enddo
324
325     do k = k_der, k_dev
326             do j = j_gauche, j_droit
327                     do i = i_bas, i_haut
328                             if((k == k_dev.or. k == k_der).and.(j/=j_droit.or.j/=j_gauche)) then
329                                     call append_tabular_3(list_cellule_lim, plan(i, j, k)%x, plan(i, j, k)%y,&
330                                     & plan(i, j, k)%z, plan(i, j, k)%numero, compt)
331                                     compt = compt + 1
332                             endif
333                     enddo
334             enddo
335     enddo
```

Dans le suite c'est avec list_cellule_complete que nous travaillerons, puisqu'on s'intéresse à la liste des cellules appartenants à la sphère donc au cube. Voici ce que cela donne en 2 dimensions :

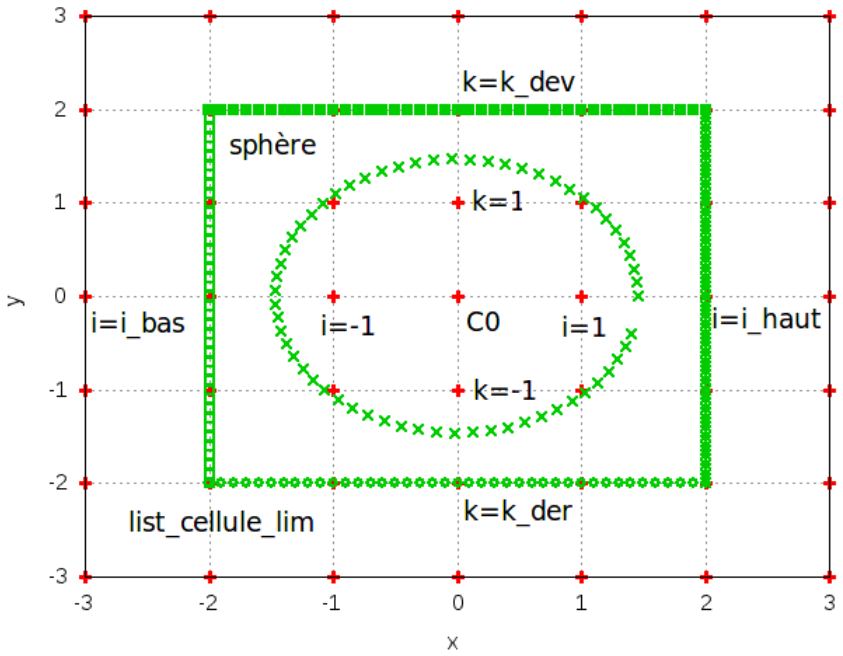

FIGURE 6 – Représentation à deux dimensions du réseau, de la sphère et de list cellule lim

Il ne reste alors plus qu'à vérifier la derniére condition avec l'algorithme suivant :

```
338         !reamplissage de list_finale
339         compt = 0
340         m=0
341         l=0
342         n=0
343         open(100, file='bob.dat')
344         do j = j_gauche, j_droit
345                 do i = i_bas, i_haut
346                         do k = k_der, k_dev
347                                 m = j_gauche
348                                 l = i_bas
349                                 n = k_der
350                                 o = 1
351                                 do while(o <= natoms)
352                                         m = j_gauche
353                                         do while(m <= j_droit)
354                                                 l = i_bas
355                                                 do while (l <= i_haut)
356                                                         n = k_der
357                                                         do while(n <= k_dev)
358 p = 1
359 do while(p<=natoms)
360         milieu_x = (plan(i, j, k)%pos_atoms_x(o) + plan(l, m, n)%pos_atoms_x(p))/2
361         milieu_y = (plan(i, j, k)%pos_atoms_y(o) + plan(l, m, n)%pos_atoms_y(p))/2
362         milieu_z = (plan(i, j, k)%pos_atoms_z(o) + plan(l, m, n)%pos_atoms_z(p))/2
363         write(100, *) plan(i, j, k)%numero, plan(l, m, n)%numero, milieu_x, milieu_y, milieu_z
364         if((milieu_x<pas1/2.and.milieu_x>-pas1/2).and.(milieu_y<pas2/2.and.milieu_y>-pas2/2)&
365         &.and.(milieu_z<pas3/2.and.milieu_z>-pas3/2)) then
366
367                 call append_tabular_4(list_finale, plan(i, j, k)%numero, o, plan(l, m, n)%numero, p, compt)
368                 compt = compt + 1
369                 p = natoms+1
370                 n = k_dev+1
371                 l = i_haut +1
372                 m = j_droit +1
373                 o = natoms + 1
374         endif
375         p = p +1
376 enddo
377 n = n + 1
378                                                         enddo
379                                                         l = l + 1
380                                                 enddo
381                                                 m = m + 1
382                                         enddo
383                                         o = o + 1
384                                 enddo
385                         enddo
386                 enddo
387         enddo
```

Après ceci on posséde donc la liste de toutes les cellules qui participent à la construction de la base des fonctions de Bloch.

Donc toute les cellules qui appartiennent à ce parrallépipéde rectangle, et qui respecte la condition donné par le code ci-dessus, participent à la construction de la base des fonctions de Bloch.

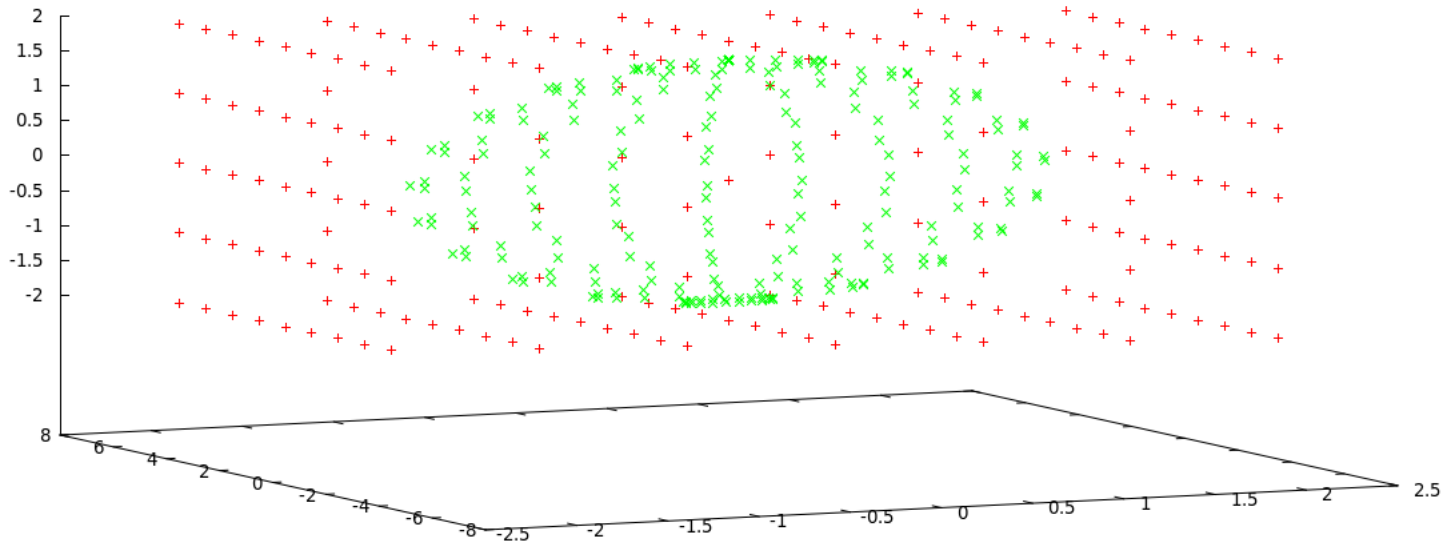

FIGURE 7 – Graphique en 3 dimensions de la sphère et des cellules qui la délimitent

Conclusion :

Donc durant mon stage mon principal travail fut la construction de ce programme, mais bien sûr la base des fonctions de Bloch n'est pas encore construite et je suis actuellement en train d'essayer de calculer les coefficients $\widetilde{V}_{\mu}^{a,b}(\overrightarrow{p_1},\overrightarrow{p_2})$ et les fonctions élementaires $\widetilde{F}_{\overrightarrow{p}}^{\mu}(\overrightarrow{r})$ à partir de la liste des cellules que j'ai construite.
Malheureusement, ce travail n'est pas terminé et je ne peux donc pas en parler dans ce rapport.
Après, lorsque les coefficients seront calculés, il restera à P.Koval à corriger et à intégrer mon programme au projet, et j'espère ainsi avoir été util dans leur projet et que mon programme ne soit pas hors contexte et inutilisable.
Bien sur Mr Foerster, Mr Koval et Mr Sanchez ont encore beaucoup de travail avant d'avoir fini leur projet et j'espère qu'ils réussiront.
Pour ma part, je ressort avec une très bonne impression de ce stage, durant lequel j'ai appris beaucoup de choses en physique des matériaux et en programmation. J'ai également pu voir et participer à la construction d'un programme complet destiné à la physique des matériaux ce qui fut une expérience très enrichissante.

Annexes

Les fonctions de Grenn

Voici un exemple d'utilisation des fonctions de Green pour la résolution de l'équation d'onde en électromagnétisme [9] [10] :

$$\left(\overrightarrow{\nabla}^2 - \frac{1}{c^2}\frac{\partial^2}{\partial t^2}\right)\Phi(\overrightarrow{r};t) = -\frac{\rho(\overrightarrow{r};t)}{\varepsilon_0}$$

$$\Longleftrightarrow D\Phi = \frac{\rho}{\varepsilon_0}$$

Avec : $D = \overrightarrow{\nabla}^2 - \frac{1}{c^2}\frac{\partial^2}{\partial t^2}$
Nous résolvons en premier l'équation suivante :

$$DG(\overrightarrow{r};t) = \delta(t-t_0)\delta((\overrightarrow{r} - \overrightarrow{r}_0)$$

Nous utilisons la transformé de Fourier temporelle :

$$TF\left(\frac{\partial^2 G}{\partial t^2}\right) = \omega^2 G(\omega) \quad (24)$$

$$TF\left(\delta(t-t_0)\right) = 1 \quad (25)$$

Nous trouvons alors :

$$\left(\overrightarrow{\nabla}^2 + \gamma^2\right) G_\omega\left(\overrightarrow{r};\overrightarrow{r}_0;\omega\right) = \delta(\overrightarrow{r} - \overrightarrow{r}_0) \qquad (\star)$$

Avec : $\gamma = \frac{\omega}{c}$
On utilise la tranformée de Fourier spatiale dans $(\star)$. Ce qui nous donne :

$$-k^2 G_{\omega k} + \gamma^2 G_{\omega k} = 1$$

$$\Longleftrightarrow G_{\omega k} = \frac{1}{\gamma^2 - k^2}$$

Puis,

$$G\left(\overrightarrow{r};\overrightarrow{r}_0;\omega\right) = \left(\frac{1}{2\pi}\right)^3 \int_{-\infty}^{+\infty} d^3k \frac{e^{i\overrightarrow{k}\overrightarrow{R}}}{\gamma^2 - k^2}$$

Avec $\overrightarrow{R} = \overrightarrow{r} - \overrightarrow{r}_0$
On passe en coordonnée polaire :

$$\Longrightarrow G\left(\overrightarrow{r};\overrightarrow{r}_0;\omega\right) = -\left(\frac{1}{2\pi}\right)^3 \int_{-\infty}^{+\infty} \frac{e^{ikRcos(\theta)}}{\gamma^2 - k^2} k^{2dkdcos(\theta)d\phi}$$

$$\Longleftrightarrow G\left(\overrightarrow{r};\overrightarrow{r}_0;\omega\right) = -\frac{1}{4\pi^2}\int_{-\infty}^{+\infty} \frac{e^{ikRcos(\theta)}}{\gamma^2 - k^2} k^2 dkdcos(\theta)$$

$$\Longleftrightarrow G\left(\overrightarrow{r};\overrightarrow{r}_0;\omega\right) = \frac{1}{4\pi^2}\int_{-\infty}^{+\infty}\int_{-1}^{+1} \frac{1}{1-\frac{\gamma^2}{k^2}} e^{ikRcos(\theta)} dcos\theta dk$$

$$\Longleftrightarrow G\left(\overrightarrow{r};\overrightarrow{r}_0;\omega\right) = \frac{1}{4\pi^2}\int_{-\infty}^{+\infty} \frac{1}{1-\frac{\gamma^2}{k^2}} \frac{1}{ikR}\left(e^{ikR} - e^{-ikR}\right) dk$$

$$\Longleftrightarrow G\left(\overrightarrow{r};\overrightarrow{r}_0;\omega\right) = \frac{1}{4\pi^2 Ri}\int_{-\infty}^{+\infty} \frac{1}{k^2 - \gamma^2}\left(e^{ikR} - e{-ikR}\right) dk$$

$$\Longleftrightarrow G\left(\overrightarrow{r};\overrightarrow{r}_0;\omega\right) = I_1 + I_2$$

Avec :

$$I_1 = \int_{-infty}^{+\infty} \frac{k}{k^2-\gamma^2} e^{ikR} dk$$

$$I_2 = -\int_{-infty}^{+\infty} \frac{k}{k^2-\gamma^2} e^{-ikR} dk$$

On utilise la méthode des résidus :
Calcule pour I_1 : on ce place dans le demi-plan supérieur

$$\oint f(z) e^{izR} dz$$

avec $R > 0$, les pôles de $f(z)$ sont donnés par :

$$z^2 - \gamma^2 = 0 \Longrightarrow z_{\pm} = \pm\gamma \in \mathbb{R}$$

$$\Longrightarrow \oint f(z) e^{izR} dz = \sum_i Res\left(f(z)e^{izR}; z_i Im(z_i) \neq 0\right) = 0$$

$$\Longrightarrow \int_{-r}^{-\gamma-\varepsilon} f(z)e^{izR}dz + \int_{-\gamma+\varepsilon}^{\gamma-\varepsilon} f(z)e^{izR}dz + \int_{\gamma+\varepsilon}^{r} f(z)e^{izR}dz + \oint_{\Gamma_1} f(z)e^{izR}dz + \oint_{\Gamma_2} f(z)e^{izR}dz + \oint_{C_r} f(z)e^{izR}dz = 0$$

Les théorèmes de Jordans permettent d'écrire :

$$\int_{-\infty}^{+\infty} f(z)e^{izR}dz - i\pi\left(Res\left(f(z)e^{izR}; -\gamma\right) + Res\left(f(z)e^{izR}; \gamma\right)\right) = 0$$

$$\begin{aligned} Res\left(f(z)e^{izR}; -\gamma\right) &= lim_{z\to-\gamma}\left(\frac{k}{k-\gamma}e^{izR}\right) = \frac{1}{2}e^{i\gamma R} \\ Res\left(f(z)e^{izR}; \gamma\right) &= lim_{z\to\gamma}\left(\frac{k}{k+\gamma}e^{izR}\right) = \frac{1}{2}e^{-i\gamma R} \end{aligned}$$

$$\Longrightarrow I_1 = \frac{i\pi}{2}\left(e^{i\gamma R} + e^{-i\gamma R}\right)$$

un calcul similaire nous donne :

$$I_2 = \frac{i\pi}{2}\left(e^{i\gamma R} - e^{-i\gamma R}\right)$$

$$\Longrightarrow G\left(\overrightarrow{r}; \overrightarrow{r}_0; \omega\right) = \frac{1}{4\pi R} e^{i\gamma R}$$

On retourne dans l'espace temporel

$$\Longrightarrow G\left(\overrightarrow{r}; \overrightarrow{r}_0; t; t_0\right) = \frac{1}{2\pi}\int_{-\infty}^{+\infty} \frac{1}{4\pi} e^{i\gamma R} e^{-i\omega(t-t_0)} d\omega$$

mais : $\gamma = \frac{\omega}{c}$

$$\Longrightarrow G\left(\overrightarrow{r}; \overrightarrow{r}_0; t; t_0\right) = \frac{1}{4\pi R}\frac{1}{2\pi}\int_{-\infty}^{+\infty} e^{-i\omega\left(t_0 - \left(t + \frac{R}{c}\right)\right)} d\omega$$

$$\Longrightarrow G\left(\overrightarrow{r}; \overrightarrow{r}_0; t; t_0\right) = \frac{1}{4\pi R}\delta\left(t_0 - \left(t + \frac{R}{c}\right)\right)$$

Le potentiel électrique est alors donné par :

$$\Phi(r,t) = \frac{1}{\varepsilon_0}\int dt' \int d^3r' G(r', r_0; t', t_0)\rho(r', t')$$

LCAO

La combinaison lineaire des orbitales atomiques ou LCAO est une superposition quantique d'orbitales atomique et une méthode de calcul moléculaire en chimie quantique. En mécanique quantique, la configuration électronique des atomes est décrite comme une fonction d'onde. Mathématiquement, ces fonctions d'ondes forment une base de fonctions qui décrivent les électrons d'un atome donné. Pour les réactions chimiques, les orbitales des fonctions d'ondes sont modifié, c'est à dire que la forme du nuage électronique est changé, selon les types d'atomes qui participent à la réaction.
LCAO a été introduite en 1929 par Sir John Lennard-Jones avec la description des liasons des molécules diatomiques.
La première hypothése est que le nombre d'orbitales moléculaires est égal au nombre d'orbitales atomiques incluse dans le develloppement linéaire. Dans le sens où, n orbitales atomiques se combinent pour former n orbitales moléculaires, lesquelles peuvent être numérotés de $i = 1$ à n et elles ne peuvent pas être identiques.
L'expression pour la i^{eme} orbitale moléculaire est :

$$\Phi_i = \sum_r C_{ri}\chi_{ri} \tag{26}$$

Où Φ_i est l'orbital moléculaire représenté comme la somme de n orbital atomique χ_r, chacune multipliée par un coefficient C_{ri} et r (numéroté de 1 à n) représente l'orbitale atomique qui est comniné dans le terme. Les coefficients représentent le poids de la contribution des n orbitales atomiques pour l'orbitale moléculaire. La méthode de Hartree-Fock est utilisé pour calculer ces coefficients.
En minimisant l'énergie totale du système, une famille appropriée de coefficients de la combinaison linéaire est déterminée.

Références

[1] P.M. Chaikin and T.C. Lubensky. *Principles of condensed matter physics.* Cambridge university press, 1995.

[2] P. Koval D. Foerster and O.coulaud. Fast construction of the kohn sham response function for molecules. *physica status solidi B*, 2010.

[3] D. Sanchez E. Artacho. *User's Guide : Siesta 3.1*, 2011.

[4] Gabriele F. Giuliani and Giovanni Vignale. *Quantum theory of the electron liquid.* Cambridge University Press, 2005.

[5] Giuseppe Grosso and Giuseppe Pastori Parravicinni. *Solid state physics.* Academic Press, 2000.

[6] Hartmut Haug and Stephan W.Koch. *Quantum theory of the optical and electronic properties os semiconductors.* World Scientific, 2004.

[7] H.Haken. *Quantum field theory of solids.* North-Holland, 1976.

[8] Richard M. Martin. *Electronic structure, basic theory and practical methods.* Cambridge University Press, 2004.

[9] G. Rickayzen. *Green's functions and condensed matter.* Academic Press, 1984.

[10] E. H. Sondheimer S. doniach. *Grenn's functions for solid state physicits.* Advanced Book Program, 1974.