RAPPORT DE STAGE

Amélioration du programme ROD : Introduction d'une possibilité de déformation des molécules.

EUROPEAN **S**YNCHROTRON **R**ADIATION **F**ACILITY

*(**I**nstallation **E**uropéenne de **R**ayonnement **S**ynchrotron)*

Doré Mikaël
IUP MAI 3

Stage réalisé du 07 Avril 2003 au 05 Septembre2003

Responsables ESRF : Me. Odile Robach
M. Olof Svensson
M. Claudio Ferrerro
Responsable IUP : M. Jean-Guillaume Dumas

version 18/09/2009

TABLE DES MATIERES

INTRODUCTION

En 1912, les travaux de Bravais sur la diffraction des rayons X ont permis de mettre en évidence la nature ordonnée et périodique de la structure cristalline. L'amélioration des outils d'étude de telles structures a conduit dans les années 90 à la nécessité d'avoir des outils de modélisation propres à la cristallographie. Le programme ROD a ainsi été écrit pour l'analyse de données expérimentales de diffraction X de surface. Ce programme contient différentes parties dont une spécifique à l'étude des groupes d'atomes à la surface du cristal.

Après avoir fait des rappels sur la cristallographie, le principe physique de la diffraction et les déformations, ce rapport portera sur l'introduction des problèmes de déformations des groupes d'atomes à la surface du cristal dans le programme ROD, ainsi qu'une documentation sur l'utilisation du programme.

Mais tout d'abord, ce rapport présentera l'ESRF (une présentation générale, un historique et son fonctionnement) où j'ai réalisé mon stage durant une période de cinq mois au sein du groupe « diffraction de surface ».

CHAPITRE 1 : PRESENTATION DE L'ESRF

1.1 Présentation générale

Située à Grenoble sur le polygone scientifique Louis Néel dans le delta formé par le Drac et l'Isère, L'ESRF (ref. [8]) (*European Synchrotron Radiation Facility ou Installation Européenne de Rayonnement Synchrotron*) est une organisation intergouvernementale, basée sur une convention signée par les représentants de 17 pays. L'ESRF est enregistrée comme compagnie française. Les 17 pays ont accepté de financer le projet pendant 20 ans.

Cette installation a pour mission de produire du rayonnement synchrotron et de favoriser son utilisation par les communautés scientifiques de ces pays.

Le rayonnement synchrotron, émis par les électrons de très grande énergie circulant dans un anneau (voir la photo ci-contre), est essentiellement composé de rayons X. Il sert à étudier les arrangements des atomes et des molécules dans la matière, un peu à la manière d'un «supermicroscope». Il peut également être utilisé dans des domaines très variés : physique, chimie, biologie, science des matériaux, géophysique, sciences de l'environnement ou médecine.

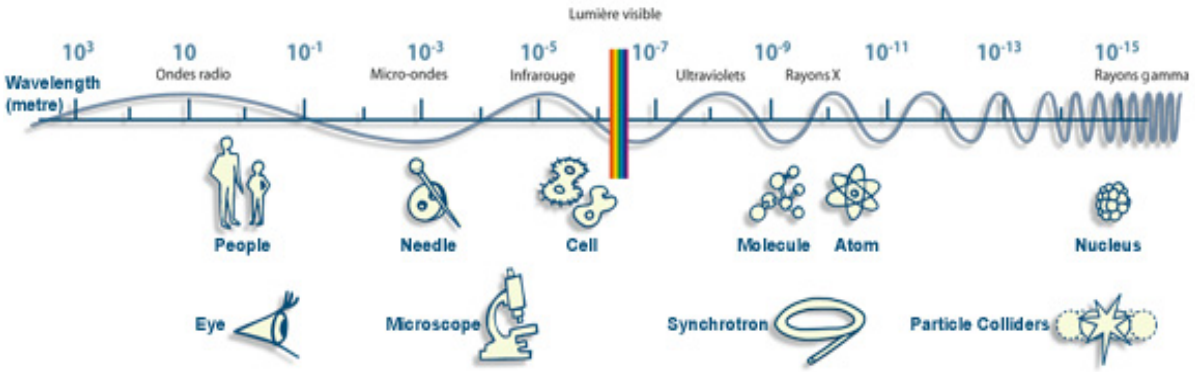
Ouvert à la recherche publique (fondamentale et appliquée), mais aussi à la recherche industrielle, l'ESRF offre à ses utilisateurs un outil extrêmement performant, des techniques en constante amélioration ainsi que les compétences d'une équipe interdisciplinaire de chercheurs, d'ingénieurs et de techniciens.

1.2 Historique

Les premiers accélérateurs (cyclotrons) ont été construits par les physiciens des particules dans les années 30. Il s'agissait de scinder le noyau de l'atome grâce à des collisions entre particules de très haute énergie, et en déduire les lois de la physique fondamentale qui régissent notre monde et l'ensemble de l'univers.

C’est dans un accélérateur d’un type différent (synchrotron) que le rayonnement synchrotron a été observé pour la première fois aux Etats-Unis en 1947. Considéré au début comme une nuisance car il faisait perdre de l’énergie aux particules, il a ensuite été reconnu dans les années 60 comme une lumière aux propriétés exceptionnelles.

Tout commence en 1975, où, lors d’une réunion de scientifiques européens, il est projeté de construire un synchrotron européen ayant l’énergie suffisante pour produire des rayons X durs d’une très grande puissance (voir le schéma avec les longueurs d’onde ci-dessous).



Il faudra attendre 1984 avant que la décision de le construire à Grenoble soit prise. Ce n’est que quatre ans plus tard, en 1988, que les travaux commencent, le budget nécessaire aura été de 550 millions d’euros puis 65 millions d’euros par an pour le faire fonctionner. Le projet donnera naissance à un anneau de 844 mètres de circonférence.

En 1992, le premier faisceau d’électrons « electron beam » circule dans l’anneau, et le premier faisceau de rayons X arrive dans un laboratoire « ligne de lumière » ou « beamline ». On peut compter 10 beamlines réparties tout autour de l’anneau, disponibles pour les chercheurs européens en 1994, puis 40 en 1998. Aujourd’hui, le nombre de beamlines est proche de son maximum qui s’élève à une cinquantaine de lignes de lumière. Ces lignes présentent des faisceaux de caractéristiques variées selon le type d’expérience qui y sont menées.

Un certain nombre de ces lignes appartiennent à l’ESRF et sert à des recherches effectuées par des scientifiques qui y sont employés. Cependant l’ESRF ayant pour vocation d’offrir une lumière synchrotron optimisée à des utilisateurs extérieurs, ces lignes peuvent être attribuées momentanément à des chercheurs ne faisant pas partie de l’entreprise. Le « visiteur » doit alors présenter un dossier sur la recherche qu’il souhaite effectuer. Une commission de l’ESRF étudie l’intérêt du projet et décide de l’attribution d’une ligne pendant un temps précisément déterminé.

1.3 Fonctionnement

Lorsque des électrons de très haute énergie sont contraints à suivre une trajectoire courbe, ils émettent des ondes électromagnétiques appelées rayonnement synchrotron. Les qualités du rayonnement synchrotron dépendent naturellement du faisceau d’électrons qui circule dans l’anneau principal, aussi appelé « anneau de stockage ». A l’ESRF, où l’on veut

obtenir des rayons X très durs, les électrons doivent posséder une très grande énergie (6 milliards d'électronvolts), ce qui requiert un anneau de stockage de grande taille.

Une autre caractéristique importante est la brillance des rayons X. Pour cela, il faut que le faisceau d'électrons soit le plus intense, le plus fin possible (l'épaisseur d'un cheveu) et très stable, ce qui demande une forte focalisation des électrons, un parfait contrôle de leur trajectoire, un vide extrême dans le tube où ils circulent et un alignement précis de tous les éléments de l'anneau de stockage.

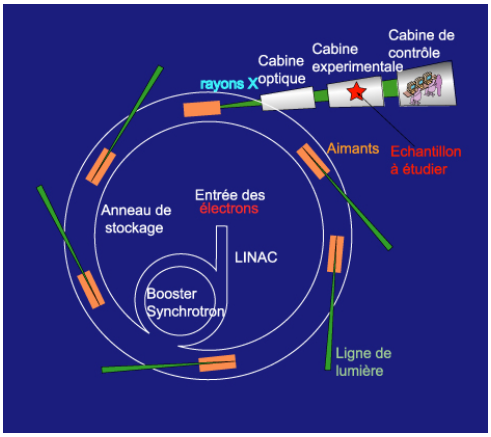
1.3.1 L'ensemble accélérateur

Les électrons sont émis par un canon à électrons puis accélérés dans un accélérateur linéaire (linac) jusqu'à une énergie de 200 millions d'électronvolts (200 MeV, grâce à des champs électriques pulsés circulant dans des cavités radiofréquence). Les électrons sont ensuite transférés dans un accélérateur circulaire (booster synchrotron), qui les amène à une énergie de 6 milliards d'électronvolts. Ils sont alors injectés dans l'anneau de stockage (storage ring), plus précisément à l'intérieur d'un tube à vide, où ils circulent à énergie constante pendant des heures, à raison de 350 000 tours par seconde.

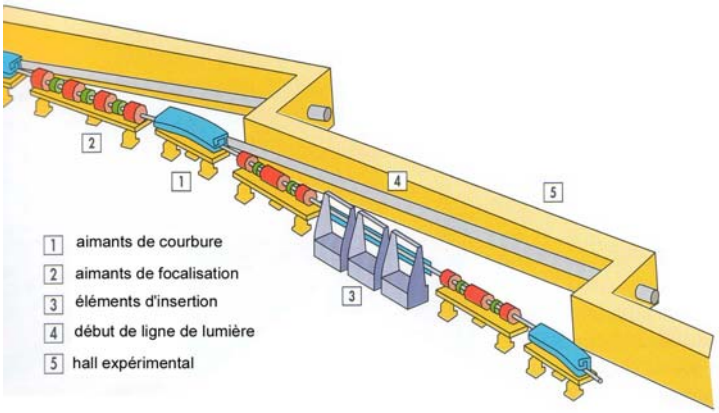
Bien que l'anneau de stockage ne soit pas à proprement parler un accélérateur, il est cependant nécessaire, là aussi, de faire passer les électrons dans des cavités radiofréquence. Il faut en effet compenser l'énergie que perdent les électrons sous forme de rayonnement synchrotron.

Dans l'anneau de stockage, les électrons sont regroupés en paquets. Il existe plusieurs modes de fonctionnement de la machine : le mode usuel ou multi-paquets (330 paquets sur un tiers de circonférence), le mode paquet unique ainsi que des modes intermédiaires. Le rayonnement émis sera pulsé avec des fréquences différentes selon le mode choisi.

Sur tout le trajet parcouru, le faisceau d'électrons circule dans une chambre à vide. La durée de vie du faisceau dépend bien sûr de la qualité du vide obtenu dans l'anneau de stockage (entre 10^{e-9} et 10^{e-10} mbar).



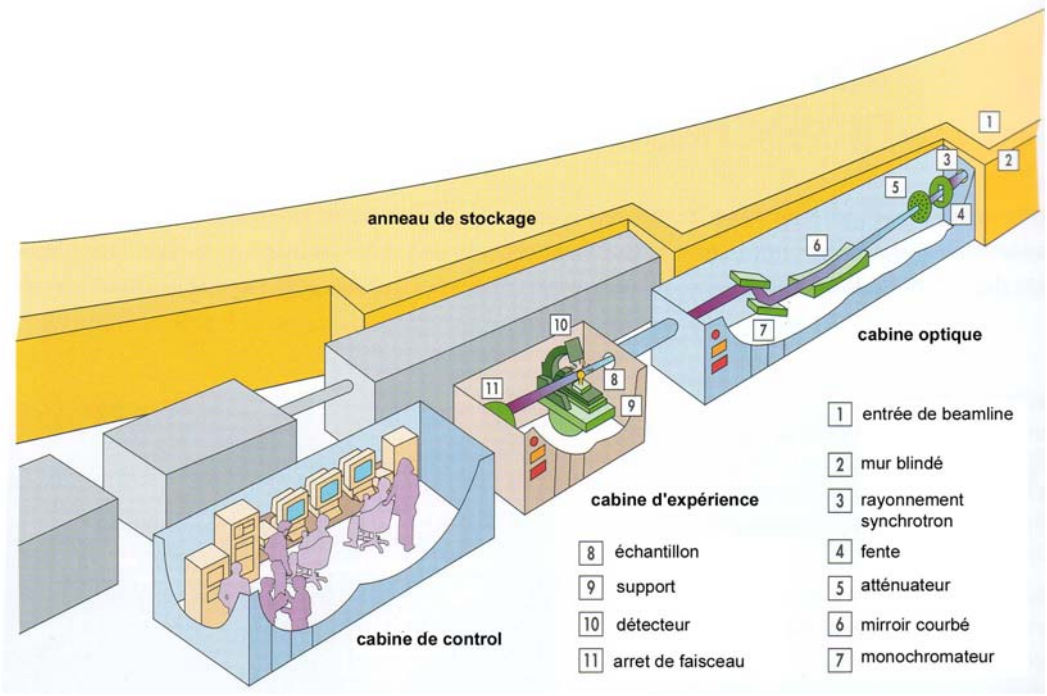
1.3.2 Dans l'anneau de stockage



Les **aimants de courbure** ont pour fonction première de forcer les électrons à suivre une trajectoire courbe, et c'est précisément en se déviant de leur trajectoire initiale que ceux-ci créent le rayonnement synchrotron.
Les **aimants de focalisation** ont pour but de concentrer les électrons dans un faisceau de très petite taille et de faible divergence, augmentant ainsi la qualité optique du faisceau lumineux.
Les **éléments d'insertion** font osciller les électrons pour que la lumière émise soit le plus intense possible.

La lumière issue de l'un de ces points sources couvre une gamme très large de longueurs d'onde, allant de l'infrarouge jusqu'au rayons X durs. De plus, elle est très intense et concentrée dans un faisceau extrêmement fin, un peu comme un rayon laser.

Chaque ligne de lumière comporte plusieurs cabines (1 à 3 cabines). Les murs des cabines dans lesquelles passe le faisceau sont renforcés de plomb afin de protéger les chercheurs des rayons X diffusés.



CHAPITRE 2 : LA CRISTALLOGRAPHIE

2.1 Introduction

La cristallographie est la science des cristaux. Elle concerne la forme extérieure, la structure interne, la croissance et les propriétés physiques des cristaux (ref. [1]).

La régularité des formes extérieures (présence de faces externes bien définies) ainsi que l'anisotropie des propriétés montrent l'existence d'une certaine régularité interne ou structure ordonnée jusqu'au niveau microscopique ou atomique. C'est cet arrangement régulier des atomes à grande distance qui distingue essentiellement les solides cristallins des solides amorphes.

Les premières expériences de diffraction des rayons X réalisées en 1912 par W. Friedrich et P. Knipping selon les idées de M. Von Laue, puis les travaux de W. L. Bragg sont venus confirmer la justesse du postulat de Bravais (selon Bravais si on considère un point P, quelconque dans un cristal, il existe dans le milieu, une infinité discrète, illimitée dans les trois directions de l'espace de points, autour desquels l'arrangement de la matière est le même qu'autour du point P). Les mesures de diffraction ont apporté la preuve expérimentale directe de la nature ordonnée et périodique de l'arrangement cristallin.

Le milieu cristallin est homogène car le comportement de plusieurs échantillons macroscopiques de mêmes dimensions et de même orientation taillée dans le cristal est identique. A l'échelle atomique où le milieu cristallin est discontinu, l'homogénéité subsiste au sens de Bravais : il existe dans le cristal suivant trois directions distinctes, une infinité discrète de points homologues d'un point quelconque, c'est-à-dire possédant le même environnement atomique.

2.2 La Géométrie

2.2.1 Le Réseau Bidimensionnel

Imaginons un plan xoy dans lequel les atomes sont répartis selon un ordre qui permet de retrouver tout atome à partir de l'origine en effectuant un certain nombre de translations élémentaires selon les axes ox et oy (fig.1).

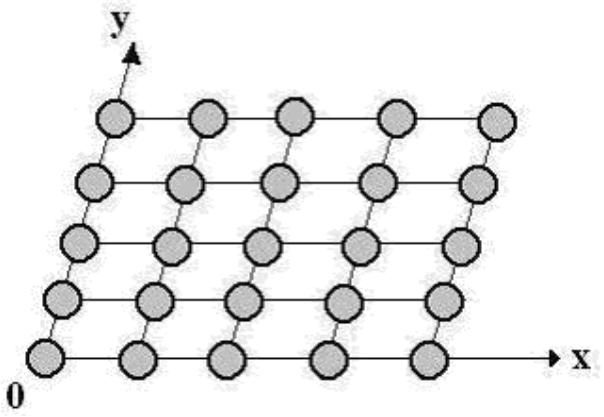


Fig.1 *Réseau bidimensionnel.*

Ce modèle bidimensionnel peut être défini à l'aide de deux paramètres qui sont le motif et le réseau (fig.2).

Motif :

- Ici atome
- Il peut être ion ou molécule

Ici :

$$\overrightarrow{OP} = 2\vec{a} + \vec{b}$$

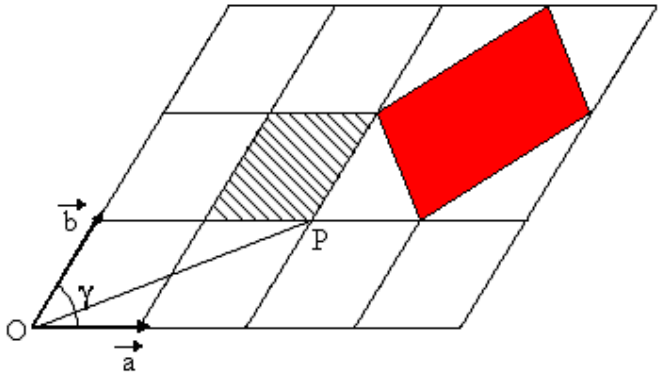


fig.2 *Réseau.*

Le réseau et le motif permettent de reconstituer le cristal bidimensionnel dans son intégralité.

Le milieu cristallin se caractérise par la présence d'une infinité de points géométriques équivalents à un point O quelconque du cristal. Tous ces points homologues ont le même environnement atomique et se déduisent les un des autres par une succession de translations élémentaires de trois vecteurs $\vec{a}, \vec{b}, \vec{c}$. Un homologue M de l'origine O est repéré par :

$$\overrightarrow{OM} = m\vec{a} + n\vec{b} + p\vec{c} ,$$

où m, n et p sont des entiers relatifs.

L'ensemble de ces points homologues, appelés nœuds, forme un réseau tridimensionnel qui traduit la périodicité du cristal dans toutes les directions (ref.[6]).

2.2.2 Le Réseau Tridimensionnel

Un cristal est tripériodique. Reprenons le réseau bidimensionnel et reproduisons le à une distance c selon un axe z extérieur au plan xoy (fig.3).

On peut répéter la même opération plusieurs fois (ref.[6], [11]).

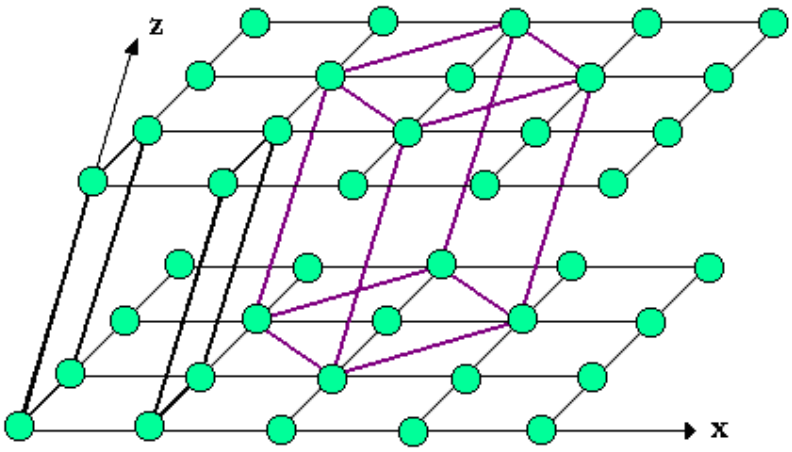


Fig.3 *Réseau tridimensionnel (ou tripériodique).*

2.2.3 La Maille

Le réseau est l'assemblage de parallélépipèdes identiques construits sur les vecteurs $\vec{a},\vec{b},\vec{c}$ mettant des faces en commun. Le parallélépipède construit avec les vecteurs $\vec{a},\vec{b},\vec{c}$ est appelé maille (ref.[4] & [6]).

Une maille est construite sur les vecteurs $\vec{a},\vec{b},\vec{c}$ qui par définition sont portés respectivement par les axes Ox, Oy, et Oz faisant entre eux les angles gamma , alpha , bêta.

Les grandeurs a, b, c, alpha, bêta et gamma sont les six paramètres définissant une maille. Elle est dite simple lorsqu'elle ne possède des noeuds qu'aux sommets. Dans le cas contraire la maille est multiple.

Donc un cristal est formé de mailles juxtaposées contenant un motif d'atomes qui se répète identiquement de l'une à l'autre. A un atome donné correspond une série d'homologues : l'ensemble de ces points forme le réseau cristallin. L'environnement de chaque atome de cet ensemble est identique, rien ne permet de les distinguer, s'ils sont suffisamment loin des bords du cristal. A partir de là il est possible de calculer la figure de diffraction d'un cristal de structure connue.

CHAPITRE 3 : PRINCIPE PHYSIQUE DE LA DIFFRACTION

Les rayons X sont une onde électromagnétique. Les rayons X interagissent avec le nuage électronique des atomes. Parmi les interactions possibles, il y a la « diffusion » ou « diffusion Rayleigh ». Lorsque les rayons X frappent un morceau de matière, ils sont donc diffusés par chacun des atomes de la cible. Ces rayons X diffusés interfèrent entre eux. Si les atomes sont ordonnés, c'est-à-dire placés à des intervalles réguliers (ce qui caractérise les cristaux), alors ces interférences vont être constructrices dans certaines directions (les ondes s'additionnent), destructrices dans d'autres (les ondes s'annulent). Ces interférences d'ondes diffusées forment le phénomène de diffraction (ref. [10]).

Pour un rayon incident de longueur d'onde donnée, on prévoit comment il faut orienter le cristal pour qu'il y ait diffraction, et quelle sera la direction du rayon diffracté. Les intensités dépendent de la structure du motif de la maille, tandis que les orientations des rayons diffractés ne dépendent que de la géométrie de la maille.

Le problème expérimental est inverse, il s'agit de trouver, à partir de l'observation des positions et intensités des pics de diffraction, la structure du cristal. Dans un premier temps il faut déterminer les paramètres de la maille cristalline. Et ensuite la disposition des atomes à l'intérieur de cette maille, c'est là que se trouve toute la difficulté.

Pour se faire différentes méthodes ont été essayées et perfectionnées, même si aucune n'apporte de solution sûre dans tous les cas, elles ont permis de résoudre un nombre considérable de problèmes de structure avec de plus en plus de précision et de rapidité.

Si l'on calcule les directions dans lesquelles on a du signal, on s'aperçoit que l'on obtient une loi simple : si l'on trace des plans imaginaires parallèles passant par les atomes, et si on appelle d la distance entre ces plans (ou « distance interréticulaire »), alors les interférences sont constructrices si : **2.d.sin(θ) = n.λ** où θ est la moitié de la déviation, n est un nombre entier appelé « ordre de diffraction », et λ est la longueur d'onde des rayons X, ceci est la loi de Bragg (fig.4).

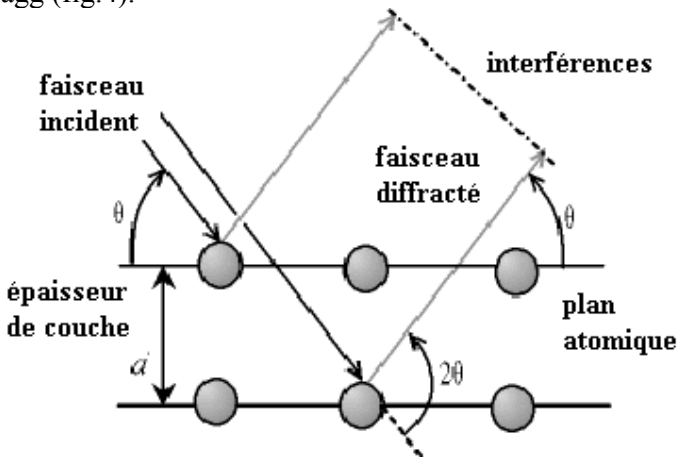


fig.4 *Loi de Bragg, réflexion par des plans réticulaires.*

La réflexion du rayonnement par les plans atomiques fournit une image moins abstraite que la réflexion par les plans réticulaires. La répartition des atomes représente cependant aussi bien le motif de la structure que la périodicité. Parallèle à la série de plans réticulaires (***hkl***), il y a des plans formés par les différentes espèces atomiques. L'amplitude

des rayons réfléchis dépend de l'espèce atomique. Les rayons provenant de plans équivalents par translation doivent être en phase, d'où la loi de Bragg. L'interférence des rayons réfléchis par différents types d'atomes donne lieu à ce que l'on appelle facteur de structure.

3.1 Diffusion Thomson

Le rayonnement incident, caractérisé par un vecteur d'onde $\mathbf{s_0}$ et un champ électrique $\mathbf{E_0}$, soumet l'électron à une accélération γ : cet électron émet un rayonnement secondaire qui, à grande distance de la source, possède une structure d'onde plane polarisée dans le plan $\mathbf{s_0}$, $\mathbf{E_0}$.

L'accélération de l'électron est donc : $\gamma = \frac{e}{m} E_0 . \exp(i\omega t)$

L'amplitude diffusée en P est égale à :

$$E_P = \frac{\mu_0}{4.\pi} \frac{e^2}{m} \frac{\sin\varphi}{r} E_0 \qquad \text{avec,} \quad \mu_0 = \frac{1}{\varepsilon_0 . c^2}$$

Si on considère un repère dans lequel l'axe Ox est confondu avec le vecteur d'onde incident s_0 et tel que le plan xOy contienne le point d'observation P. Le champ électrique incident peut s'écrire $\mathbf{E = E_y + E_z}$. Le rayonnement incident est la somme d'un grand nombre de vibrations incohérentes dont l'effet s'obtient en faisant la somme des intensités (voir fig.5).

Si le rayonnement incident n'est pas polarisé les valeurs moyennes de $\mathbf{E_y}$ et de $\mathbf{E_z}$ sont égales et donc : $\mathbf{I_y = I_z = I/2}$. la contribution de la composante suivant Oy de l'onde incidente à l'onde diffusée est :

$$I_y^P = \left(\frac{\mu_0 e^2 \sin\varphi_1}{4.\pi m r} \right)^2 .I \quad \text{de même : } I_z^P = \left(\frac{\mu_0 e^2 \sin\varphi_2}{4.\pi m r} \right)^2 .I$$

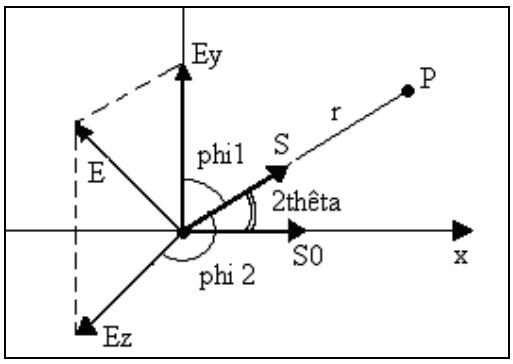


***Fig.5** Représentation dans le repère xOy.*

Or $\varphi_2 = \frac{\pi}{2}$ et $2\theta + \varphi_1 = \frac{\pi}{2}$. On en déduit, en sommant les intensités, la formule de Thomson :

$$\frac{I_{diffusé}}{I_{incident}} = \left(\frac{\mu_0}{4\pi} \right)^2 \frac{e^4}{m^2 . r^2} \frac{1 + \cos^2 2\theta}{2}$$

En posant : $r_e = \frac{1}{4.\pi.\varepsilon_0}\frac{e^2}{m.c^2} = 2{,}818.10^{-15}$ et $P(\theta) = \frac{1+\cos^2 2\theta}{2}$,

où $\mathbf{r_e}$ est le rayon classique de l'atome et $\mathbf{P(\theta)}$ qui est le facteur de polarisation. On peut exprimer l'intensité diffusée par un électron sous la forme :

$$I_{él} = I_0.\left(\frac{r_e}{r}\right)^2.P(\theta)$$

3.2 Conditions de diffraction

Pour que *F* soit maximal (c'est à dire que l'on ait un pic), il faut que les rayons diffusés soient en phase, donc que leur déphasage soit le même à 2π radians près, soit $\mathbf{K}.\mathbf{r}_j = \mathbf{K}.\mathbf{r}_l$ [2π] pour tous les atomes de la maille deux à deux. Si ($\mathbf{a_1}$, $\mathbf{a_3}$, $\mathbf{a_3}$) sont les vecteurs de base de la maille (c'est-à-dire les vecteurs des arêtes non coplanaires), les positions des atomes s'écrivent

$\mathbf{r}_j = x_j.\mathbf{a_1} + y_j.\mathbf{a_2} + z_j.\mathbf{a_3}$

x, *y* et *z* sont des réels positifs inférieurs à 1. De ceci, on déduit la condition de diffraction de Laue : il y a un maximum d'intensité si **K** est une combinaison linéaire entière des vecteurs de la base conjuguée ($\mathbf{a^*_1}$, $\mathbf{a^*_3}$, $\mathbf{a^*_3}$) définie par :

$\mathbf{a^*_1} = 2\pi.\mathbf{a_2}\hat{}\mathbf{a_3}/V$
$\mathbf{a^*_2} = 2\pi.\mathbf{a_3}\hat{}\mathbf{a_1}/V$
$\mathbf{a^*_3} = 2\pi.\mathbf{a_1}\hat{}\mathbf{a_2}/V$

$V = \mathbf{a_1}.(\mathbf{a_2}\hat{}\mathbf{a_3})$ étant le volume d'une maille élémentaire ; on a $\mathbf{a}_i.\mathbf{a^*}_i = 2\pi$, et $\mathbf{a}_i.\mathbf{a^*}_j = 0$ si $i \neq j$. (si l'on prend $k = 1/\lambda$, alors il n'y a pas de facteur 2π dans la définition de la base réciproque, et l'on a $\mathbf{a}_i.\mathbf{a^*}_i = 1$).

On peut donc écrire $\mathbf{K} = \mathbf{k_1} + \mathbf{k_2} + \mathbf{k_3} = h.\mathbf{a^*_1} + k.\mathbf{a^*_2} + l.\mathbf{a^*_3}$, *h*, *k* et *l* étant des entiers. On peut ainsi indicer les vecteurs de diffraction donnant un maximum d'intensité par ces indices et écrire $\mathbf{K}_{hkl}$. On a donc un pic de diffraction si le détecteur se trouve dans une direction $\mathbf{k'} = \mathbf{k}+\mathbf{K}_{hkl}$. Et *h*, *k* et *l* sont les composantes du vecteur K dans la base de l'espace réciproque ($\mathbf{a^*_1}$, $\mathbf{a^*_3}$, $\mathbf{a^*_3}$).

Lorsque **K** vaut $\mathbf{K}_{hkl}$ (pour *h*, *k* et *l* donnés), cela signifie que les rayons diffusés dans la direction du détecteur par tous les atomes d'un même plan cristallographique (*hkl*) dans le monocristal ont la même phase, on peut donc dire que le pic de diffraction est généré par ce plan (*hkl*) ; on retrouve ainsi la loi de Bragg.

3.3 Réseau réciproque et espace des phases

Prenons un espace à trois dimensions, avec la base vectorielle ($\mathbf{a^*_1}$, $\mathbf{a^*_3}$, $\mathbf{a^*_3}$), et traçons-y le vecteur de diffraction **K** (partant de l'origine) - rappel : **K** est imposé par les positions du tube et du détecteur par rapport au cristallite. Il n'y a diffraction que si l'extrémité de **K** est sur un point dont les coordonnées dans le repère sont entières. On a ainsi, dans cet espace, un réseau en 3D de points de coordonnées entières correspondant aux conditions de diffraction.

Les vecteurs du réseau réciproque sont parallèles (dans le cas où on n'a que des angles droits) aux vecteurs de la base cristalline, mais les rapports des longueurs sont inversés (fig.5) : si $\mathbf{a_3}$ est le vecteur le plus long de l'espace direct, $\mathbf{a^*_3}$ sera le vecteur le plus court du réseau réciproque.

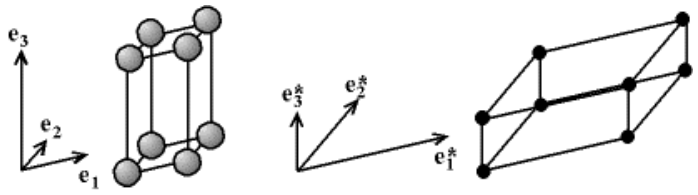


Fig.6 *Comparaison entre les réseaux direct et réciproque*

Lorsque l'extrémité du vecteur **K** est sur le point de coordonnées (*h*,*k*,*l*), les rayons X diffusés dans la direction du détecteur par les atomes des plans (*hkl*) sont en phase, on peut donc dire que ce point représente le plan (*hkl*).

3.4 Facteur de structure

Considérons les atomes présents dans une seule maille cristalline. La somme des ondes diffractées par les atomes (*j*) de cette maille vaut :

$$\psi'_1(t,r) = A.\exp[i.(\omega.t + k'.r)].F(K)$$

avec $\mathbf{F(K)} = \sum_j f_j \exp(i K r_j)$ ou encore, $\mathbf{F_{hkl}} = \sum_j f_j \exp(2\pi i (hx_j + ky_j + lz_j))$

F est appelé « facteur de structure », car il est caractéristique de la structure cristallographique.

3.5 Intensité intégrale

L'intensité intégrale I_{hkl}, c'est-à-dire la surface d'un pic de diffraction, correspond à l'énergie diffractée par le plan (*hkl*). Si on a respectivement N1, N2, N3 mailles dans les trois directions, alors on a l'intensité totale qui vaut :

$$\mathbf{I}_{hkl} = \left| \sum_{n1=1}^{n1=N1} \exp(2i\pi n_1 h) \sum_{n2=1}^{n2=N2} \exp(2i\pi n_2 k) \sum_{n3=1}^{n3=N3} \exp(2i\pi n_3 l) \right|^2 \quad |\mathbf{F}_{hkl}|^2$$

Ou encore, $$\mathbf{I}_{hkl} = \frac{\sin^2(N_1 h\pi)}{\sin^2(h\pi)} \quad \frac{\sin^2(N_2 k\pi)}{\sin^2(k\pi)} \quad \frac{\sin^2(N_3 l\pi)}{\sin^2(l\pi)} \quad |\mathbf{F}_{hkl}|^2$$

Dans le cas de la diffraction de surface le terme n_3 disparaît dans la première formule, la dernière somme disparaît.

Dans le paragraphe suivant nous allons introduire la possibilité de petites déformations d'un groupe d'atomes, ceci afin de simuler les déformations des molécules induites par l'interaction molécule-substrat ou l'interaction entre molécules voisines. Et ainsi de manière à se rapprocher le plus possible du modèle recherché (des données expérimentales).

CHAPITRE 4 : LES DEFORMATIONS

L'idée de composantes de déformation a été introduite pour représenter l'ensemble des petites déformations d'un corps (ref. [5]). Avant de définir ces composantes, définissons d'abord le vecteur déplacement.

Par la suite de la déformation, l'élément de coordonnées initiales *x, y, z* prend pour coordonnées *x+U, y+V, z+W*. C'est le vecteur *(U, V, W)* que l'on va désigner par « vecteur déplacement », et ses composantes sont fonctions de *x, y, z*.

Considérons alors un petit élément de volume et supposons l'origine des coordonnées à l'intérieur de cet élément avant la déformation. Le déplacement de tout point de cet élément s'exprime au moyen de douze constantes sous la forme suivante :

- $U = U_0 + \frac{\partial U}{\partial x} x + \frac{\partial U}{\partial y} y + \frac{\partial U}{\partial z} z$
- $V = V_0 + \frac{\partial V}{\partial x} x + \frac{\partial V}{\partial y} y + \frac{\partial V}{\partial z} z$
- $W = W_0 + \frac{\partial W}{\partial x} x + \frac{\partial W}{\partial y} y + \frac{\partial W}{\partial z} z$

où les dérivées sont prises à l'origine.

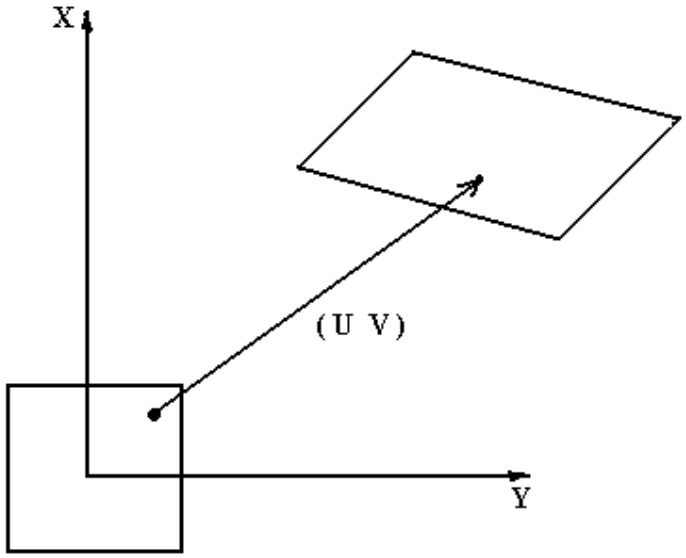


Fig.7 *représentation du vecteur déplacement dans le plan.*

Le vecteur *(U_0, V_0, W_0)* définit le déplacement du point se trouvant à l'origine avant la déformation. Dans notre cas ici, nous n'allons pas traiter ce vecteur, car il a déjà été traité auparavant, dans le cas des déplacements du groupe, par Olof Svensson. Le déplacement de tout point du corps par rapport à l'origine est donc défini par neuf constantes qui sont les neufs dérivées partielles.

On peut obtenir une interprétation physique plus directe par neuf autres constantes qui sont des fonctions linéaires des précédentes.

$$e_{xx} = \frac{\partial U}{\partial x}, \qquad e_{yy} = \frac{\partial V}{\partial y}, \qquad e_{zz} = \frac{\partial W}{\partial z}$$

$$e_{yz} = \frac{\partial W}{\partial y} + \frac{\partial V}{\partial z}, \qquad e_{zx} = \frac{\partial U}{\partial z} + \frac{\partial W}{\partial x}, \qquad e_{xy} = \frac{\partial V}{\partial x} + \frac{\partial U}{\partial y}$$

$$\omega_x = \frac{\partial W}{\partial y} - \frac{\partial V}{\partial z}, \qquad \omega_y = \frac{\partial U}{\partial z} - \frac{\partial W}{\partial x}, \qquad \omega_z = \frac{\partial V}{\partial x} - \frac{\partial U}{\partial y}$$

On peut obtenir une signification immédiate de ces neuf constantes si leur ordre de grandeur est petit (c'est-à-dire si elles sont négligeables devant l'unité, car on traite des déformations infinitésimales). Dans ce cas, on peut décomposer en deux la déformation par rapport à l'origine (ref.[5]&[7]). La première déformation a pour vecteur déplacement :

$$U_S = e_{xx}\ x + \frac{1}{2} e_{xy}\ y + \frac{1}{2} e_{xz}\ z$$

$$V_S = \frac{1}{2} e_{xy}\ x + e_{yy}\ y + \frac{1}{2} e_{yz}\ z$$

$$W_S = \frac{1}{2} e_{xz}\ x + \frac{1}{2} e_{yz}\ y + e_{zz}\ z$$

dont les coefficients forment une matrice symétrique.

La deuxième déformation a pour vecteur déplacement :

$$U_A = \qquad - \frac{1}{2}\ \omega_z\ y\ + \frac{1}{2} \omega_y\ z$$

$$V_A = \frac{1}{2} \omega_z\ x \qquad\qquad - \frac{1}{2} \omega_x\ z$$

$$W_A = - \frac{1}{2} \omega_y\ x + \frac{1}{2} \omega_x\ y$$

dont les coefficients forment une matrice antisymétrique.

On voit facilement que le premier vecteur déplacement *(U_S, V_S, W_S)* représente une déformation dans laquelle des axes de coordonnées liés au solide ne changent pas de direction. On appelle ces axes « axes principaux » de la déformation.

Les six quantités e_{xx}, e_{yy}, e_{zz}, e_{xz}, e_{yz}, e_{xz} sont appelées « composantes de déformation ». On peut considérer la première composante e_{xx} comme la variation de longueur relative d'un segment initialement parallèle à l'axe des x, une interprétation analogue s'applique à la deuxième et la troisième composante (voir **5.5.1 La dilatation / compression**). On peut considérer e_{yz} comme la variation de l'angle Oy, Oz pendant la déformation, ces axes restant liés au solide; e_{zx} et e_{xy} s'interprètent enfin de façon analogue (voir **5.5.2 Le cisaillement**).

Et on voit facilement que le vecteur déplacement (U_A, V_A, W_A) représente une déformation dans laquelle le corps subit une rotation d'ensemble autour de l'origine. Comme pour le déplacement du groupe d'atome, nous ne traiterons pas la rotation car celle-ci a été traité auparavant par Olof Svensson.

Le paragraphe suivant présente le programme ROD, ses applications, son utilisation avec les fichiers utilisés ainsi que les différentes améliorations apportées.

CHAPITRE 5 : LE PROGRAMME ROD

5.1 Introduction

Le programme ROD est l'outil de base à l'ESRF pour l'analyse des données expérimentales de diffraction X de surface. Le but des mesures est d'élucider la structure cristallographique au voisinage de la surface d'un substrat monocristallin. La surface est typiquement soit propre soit recouverte d'atomes étrangers.

Le programme ROD contient une partie de gestion du modèle atomique, une partie de calcul des intensités diffractées correspondant à ce modèle atomique, et une partie d'ajustement des paramètres structuraux du modèle ainsi qu'une partie graphique permettant de représenter le modèle en 3D et de tracer des courbes représentant différents résultats. Cette dernière partie permet de trouver quels modèles atomiques peuvent correspondre à un certain jeu de mesures des intensités diffractées.

Les programmes ANA/AVE/ROD ont été écrits au départ par Elias Vlieg dans le début des années 90. Les programmes ont été à l'origine écrits pour être exécutés sur les PCs compatibles MSDOS. Le package commercial "GraphiC" a été utilisé pour les traçages des courbes. Le code source a été rendu accessible pour tous, ceci à favoriser le développement en parallèle par un certain nombre de laboratoires et équipements, par exemple BNL et Daresbury (synchrotrons). Paul Howes (université de Leicester) a mis en communication le code sous UNIX, il a changé la routine de traçage pour utiliser la bibliothèque graphique de PGPLOT et a changé la routine de menu pour utiliser la bibliothèque de readline de GNU. Le projet d'ANA/ROD à l'ESRF a été lancé par Detlef Smilgies (ID10B) et Olof Svensson, le but de ce projet était le suivant : utiliser un compilateur libre, afin de permettre à des utilisateurs de modifier facilement le code source, les programmes devaient fonctionner sous MS WINDOWS 95/98/NT et UNIX. Il fallut remplacer la bibliothèque "GraphiC" commerciale par une bibliothèque graphique gratuite. Prendre les meilleures parties des différentes versions et les utiliser pour développer des versions d'ANA et de ROD adaptées aux besoins des utilisateurs de l'ESRF.

Par la suite Olof Svensson a introduit la possibilité de regrouper les atomes à la surface et ainsi pouvoir déplacer la molécule ou changer l'orientation de la molécule, en agissant sur le groupe et non sur chaque atome (car tous ces atomes sont liés), ce qui va simplifier la gestion d'un grand nombre d'atomes. Et ceci va permettre par la suite d'introduire des déformations de la molécule due à l'interaction avec la surface de substrat.

5.2 Fonctionnement de ROD

5.2.1 Utilisation de ROD

ROD utilise des commandes en lignes, à l'exécution de ROD on accède au menu ROD> et dans la suite on a plusieurs menus (qui sont écrits en majuscule avec un historique des menus ex. : ROD.MENU1.MENU2>). Pour chaque menu, on accède à une aide en utilisant la commande « help » ou « ? » ceci va afficher une liste de commandes accessibles avec une courte description. Pour exécuter une commande, l'utilisateur doit saisir la partie en majuscule de la commande pour que l'interprète de commande-ligne puisse l'accepter et ainsi l'exécuter. Sur une seule ligne on peut écrire autant de commandes qu'on le souhaite à la suite les unes des autres, celles-ci sont exécuter après le <return>.

L'échantillon est typiquement un monocristal sous forme de disque possédant une face polie et c'est la diffraction par la partie du cristal au voisinage de cette face que l'on décrit. Avec ROD, pour décrire le cristal et sa surface, on divise le modèle atomique en deux parties :

- une partie « bulk » (ou substrat) qui est un cristal infini en x, y et semi infini dans la direction z, de maille et positions atomiques connues, et dont les atomes ne bougent pas.
- Une partie « surface » dont la maille est identique à celle du « bulk » (sinon on a une erreur), mais dont les atomes peuvent bouger et être regroupés dans différents groupes.

```
*** Structure analysis program, version experimental-1-3 Jun 23 2003 ***
CLI: Opening macro file "plotinit.mac"
CLI: Closing macro file "plotinit.mac"
CLI: Opening macro file "rod_init.mac"
CLI: Closing macro file "rod_init.mac"
ROD>help
********************************** MAIN MENU **********************************
 Read      : Read data/model files
 List      : List data/model
 RESet     : Reset all parameters
 Calculate : Calculate structure factors
 Plot      : Goto plotting menu
 Set       : Set parameters
 Fit       : Fit experimental structure factors
 Energy    : Goto lattice energy menu
 Extensions: Goto extensions menu
 Macro     : Run macro file
 !         : Execute an operating system command
 Help      : Display menu
 QUIT      : Quit program
ROD>ex sv se
ROD,EXT,SVE,SET>h
********************************** SET MENU ***********************************
 Group     : Model par. for fitting group(s)
 CYlinder  : Model par. for fitting cylinder(s)
 Ellipsoid : Model par. for fitting ellipsoid(s)
 PArameters: Values of fit parameters
 Deform    : Deformation
 Help      : Display menu
 Return    : Return to main menu
ROD,EXT,SVE,SET>
```

Fig.8 *Exemples de menus.*

Dans la suite nous allons travailler avec des groupes d'atomes qui se trouvent à la surface du cristal, nous allons décrire comment sont définis les groupes d'atomes ainsi que les différents fichiers utilisés. Et par la suite je décrirai quelles modifications j'ai apporté pour introduire la possibilité de déformation du groupe d'atomes, ceci afin de simuler les déformations des molécules induites par l'interaction molécule-substrat ou l'interaction entre molécules voisines.

5.2.2 Groupe d'atome à la surface du cristal

Seul le modèle de la surface peut contenir un groupe d'atome et celui-ci est défini comme un groupe rigide, c'est avec ce groupe d'atome que je vais travailler par la suite. Un groupe contient différentes données :

- Le nom du groupe.
- Une liste des atomes le constituant, avec pour chaque atome ses coordonnées (dans un repère orthogonal) en Ångström (Å, 1Å = 10^{-10} m).
- Les coordonnées du centre de rotation du groupe d'atome (en Ångström) dans le repère orthogonal (O, x, y, z).
- Les coordonnées du centre de rotation dans la maille de surface et trois angles de rotation phi, chi et thêta. La rotation se fait dans le repère orthogonal (O, x, y, z), on effectue une rotation de χ autour de l'axe x, une rotation de θ autour de l'axe z et une rotation de ϕ degrés autour de l'axe z.

Ce groupe d'atome peut être saisi à la main par l'utilisateur via le menu ROD.EXT.SVE.SET.GROUP> ou via un fichier d'entrée (un fichier *.sur ou *.fit qui est préparé à l'extérieur et lu directement par ROD) définissant le modèle de la surface.

5.2.3 Saisir un groupe d'atomes avec ROD

Dans cette partie nous allons voir comment l'utilisateur peut saisir un groupe d'atomes défini comme ci-dessus. Ci-dessous (fig.8 & 9) un exemple de molécule formée de 7 atomes, donnés par leurs coordonnées (en Ångström) dans le repère orthogonal (O, x, y, z).

Atome	x	y	z
C1	0.0	0.0	0.0
C2	1.204	0.0	0.695
C3	-1.204	0.0	0.695
C4	1.204	0.0	2.085
C5	-1.204	0.0	2.085
C6	0.0	0.0	2.780
C7	0.0	0.0	4.690

Fig.9 *Tableau correspondant aux coordonnées.*

Ici la molécule est de forme plane et hexagonale.

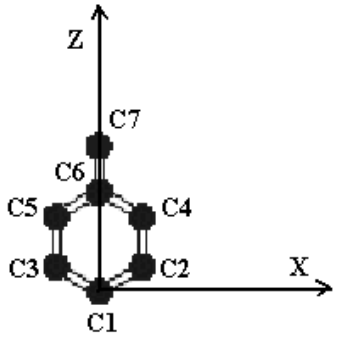


Fig.10
Représentation de la molécule dans le plan (Ox,Oz).

Menu ROD.EXT.SVE.SET.GROUP>

Le menu ROD.EXT.SVE.GROUP> contient ces fonctionnalités.

```
ROD.EXT.SVE.SET.GROUP>help
************************** GROUP MODEL PARAMETERS ***********************
Name : Group name
Xstart : Start x-position
XDisplace : Serial number of x-displacement parameter
Ystart : Start y-position
YDisplace : Serial number of y-displacement parameter
Zstart : Start z-position
ZDisplace : Serial number of z-displacement parameter
Phistart : Start phi-position
PHIDispl : Serial number of phi-displacement par.
Chistart : Start chi-position
CHIDispl : Serial number of chi-displacement par.
Thstart : Start th-position
THDispl : Serial number of th-displacement par.
Origin : Origin of group atoms
ADDGroup : Add a group
DELGroup : Delete a group
NGroups1 : No groups in the first surface cell
Element : Element type of group atom
XAtom : X-coordinate of group atom
YAtom : Y-coordinate of group atom
ZAtom : Z-coordinate of group atom
B : Serial # of group atom Debye-Waller par
ADDAtom : Add an atom to group
DELAtom : Delete an atom from group
List : List atoms
Help : Display menu
Return : Return to main menu

ROD.EXT.SVE.SET.GROUP>list
No groups defined.
```

Comment ajouter un groupe à la surface.

```
ROD.EXT.SVE.SET.GROUP>addg
New group added, group no = 1
```

Nous venons d'ajouter un groupe mais celui-ci ne contient encore rien.

```
ROD.EXT.SVE.SET.GROUP>list
# group name x + d y + d z + d phi + d chi + d th + d
1 0.0000 0 0.0000 0 0.0000 0 0.0000 0 0.0000 0 0.0000 0
Origin of atoms: 0.0000 0.0000 0.0000
# el x + y + z + B
```

Entrer les données de ce groupe.

```
ROD.EXT.SVE.SET.GROUP>name
Group number: 1
Group name[ ]: Example
ROD.EXT.SVE.SET.GROUP>
Then we add the first atom:
ROD.EXT.SVE.SET.GROUP>adda
Group number:1
New atom added in group # 1, atom no = 1
ROD.EXT.SVE.SET.GROUP>element
Group number:1
Atom number: 1
Element type: C

ROD.EXT.SVE.SET.GROUP>list
# group name x + d y + d z + d phi + d chi + d th + d
```

```
1 Example 0.0000 0 0.0000 0 0.0000 0 0.0000 0 0.0000 0 0.0000 0
Origin of atoms: 0.0000 0.0000 0.0000
# el x + y + z + B
1 C 0.0000 0.0000 0.0000 0
```

Nous venons d'ajouter un atome il reste à entrer les 6 autres.

Pour entrer les 6 autres atomes, on procède comme au dessus et en plus on va entrer les coordonnées de ces atomes (dans le repère orthogonal). Pour pouvoir entrer les coordonnées des atomes on va utiliser les fonctionnalités « xatom », « yatom » et « zatom » puis on saisie le numéro de groupe ainsi que le numéro d'atome et on peut alors saisir la valeur de la coordonnée. De même si on veut que le groupe d'atome ait une certaine orientation, alors on peut utiliser les fonctionnalités « phistart », « chistart » et « thstart » pour orienter le groupe dans le repère orthogonal.

```
ROD.EXT.SVE.SET.GROUP>adda 1 element 1 2 C xatom 1 2 1.204 zatom 1 2 0.695
New atom added in group # 1, atom no = 2
ROD.EXT.SVE.SET.GROUP>adda 1 element 1 3 C xatom 1 3 -1.204 zatom 1 3 0.695
New atom added in group # 1, atom no = 3
ROD.EXT.SVE.SET.GROUP>adda 1 element 1 4 C xatom 1 4 1.204 zatom 1 4 2.085
New atom added in group # 1, atom no = 4
ROD.EXT.SVE.SET.GROUP>adda 1 element 1 5 C xatom 1 5 -1.204 zatom 1 5 2.085
New atom added in group # 1, atom no = 5
ROD.EXT.SVE.SET.GROUP>adda 1 element 1 6 C zatom 1 6 2.780
New atom added in group # 1, atom no = 6
ROD.EXT.SVE.SET.GROUP>adda 1 element 1 7 C zatom 1 7 4.690
New atom added in group # 1, atom no = 7
```

Les 6 atomes ont été entrés nous pouvons faire afficher la liste :

```
ROD.EXT.SVE.SET.GROUP>list
# group name x + d y + d z + d phi + d chi + d th + d
1 Example 0.0000 0 0.0000 0 0.0000 0 0.0000 0 0.0000 0 0.0000 0
Origin of atoms: 0.0000 0.0000 0.0000
# el x + y + z + B
1 C 0.0000 0.0000 0.0000 0
2 C 1.2040 0.0000 0.6950 0
3 C -1.2040 0.0000 0.6950 0
4 C 1.2040 0.0000 2.0850 0
5 C -1.2040 0.0000 2.0850 0
6 C 0.0000 0.0000 2.7800 0
7 C 0.0000 0.0000 4.6900 0
```

5.3 Les fichiers d'entrée

On peut simplifier la construction du modèle atomique en lisant divers types de fichiers. Avec ROD, ces fichiers peuvent être produits en utilisant un éditeur standard ou être sauvés depuis ROD en utilisant la commande de LIST. La commande LIST permet d'avoir l'affichage de cette liste à l'écran, mais si on indique un nom de fichier, alors le modèle est écrit dans un fichier au format approprié. Dans la suite nous allons voir les différents formats de fichiers utilisés.

5.3.1 Fichiers (*.sur) contenant le model de la surface

Une autre manière de construire un groupe d'atome, c'est de lire le fichier *.sur (voir exemples annexe 3) qui représente ce modèle, voici comment se présente ce fichier :

```
Première ligne : commentaire

Seconde ligne : a   b   c   alpha  bêta  gamma   (paramètres de maille)*(1)
```

```
Définition des groupes :

Group « nom du groupe »     [ origine-x     origine-y     origine-z  ]
Xstart     Ystart    Zstart     phi     chi     th

Type d'atome 1      x-atome1  y-atome1 z-atome1  [DebyWaller serial number1]
Type d'atome 2      x-atome2  y-atome2 z-atome2  [DebyWaller serial number2]
……
endgroup
```

*** (1).** Les paramètres de maille sont ceux décrits plus haut (cf. **2.2.3 La Maille**), ces paramètres doivent être les mêmes pour le substrat (fichiers *.bul voir exemple annexe 3) et la surface (fichiers *.sur ou *.fit).

Si on reprend l'exemple de ci-dessus on a (dans un fichier *.sur) :

```
Example of grouping of atoms
1.0000 1.0000 1.0000 90.0 90.0 90.0
group Example 0.0000 0.0000 0.0000
0.0000 0.0000 0.0000 0.0000 0.0000 0.0000
C 0.00000 0.00000 0.00000
C 1.20400 0.00000 0.69500
C -1.20400 0.00000 0.69500
C 1.20400 0.00000 2.08500
C -1.20400 0.00000 2.08500
C 0.00000 0.00000 2.78000
C 0.00000 0.00000 4.69000
endgroup
```

Les fichiers (*.bul) qui contiennent le modèle du substrat, ont le même format que les fichiers de surface (à la différence que le fichier *.bul ne possède pas de groupes).

5.3.2 Les fichiers fit (*.fit)

Un fichier modèle extérieur contient des valeurs fixes pour les coordonnées des atomes. En optimisant une structure extérieure, on doit pouvoir changer les positions des atomes. En raison des contraintes de symétrie, quand un atome est déplacé, d'autres atomes doivent être déplacés. Le logiciel en bloc de cristallographie contient souvent cette information de symétrie et seulement un atome d'un ensemble symétrie-connexe doit être donné par l'utilisateur. Ce n'est pas le cas dans ROD et toutes ces relations doivent être données à la main. Ceci est fait dans ROD en utilisant des paramètres de déplacement.

Les fichiers fit (*.fit) sont presque les mêmes que les fichiers de surface (*.sur, vus au dessus), tout en contenant d'autres paramètres, ici de déplacement, et après, de déformation. Voici une présentation des fichiers fit :

```
Première ligne : commentaire

Seconde ligne : a   b   c   alpha   bêta   gamma       (pamètres de maille)

Définission des groupes :

Group « nom du groupe »     [ origine-x     origine-y     origine-z  ]

Xstart   SN-déplx   Ystart   SN-déply   Zstart   SN-déplz      phi   SN-rotph
chi   SN-rotch   th   SN-rotth

Type d'atome 1      x-atome1  y-atome1 z-atome1  [DebyWaller serial number1]
Type d'atome 2      x-atome2  y-atome2 z-atome2  [DebyWaller serial number2]
……
endgroup
```

SN-déplx, SN-déply, SN-déplz sont respectivement les « serial number » des déplacements en x, y et z du groupe (translation du groupe). Ainsi que SN-rotph, SN-rotch et SN-rotth sont les « serial number » des rotations du groupe (phi, chi et theta).

5.3.3 Les « serial number ».

La notion de « serial number » est utilisée dans tout le programme (pour les déplacements d'un groupe, les rotations …) ceci pour éviter d'avoir plusieurs indices et permettre des liens entre les déplacements dans plusieurs directions (c'est-à-dire que si on a un serial number égal pour deux déplacements dans deux directions différentes alors la quantité de déplacement dans les deux directions sera la même). Durant l'optimisation pour trouver le « bon modèle » si on a un « serial number » égal à zéro alors on n'a pas de variation de grandeur de la variable désignée et dans le cas où le « serial number » est positif alors on a une variation de la variable concernée.

ROD optimise les valeurs de tous les paramètres variables énumérés ci-dessus (ceux avec un numéro de série « serial number »). L'optimisation se fait en minimisant l'écart entre les intensités mesurées et les intensités calculées pour les différents *h, k, l*. Il est possible de changer ces paramètres (ou de les éditer) dans ROD (dans le menu ROD.EXT.SVE.SET.GROUP>). En utilisant un fichier fit, un fichier surface (*.sur) n'est plus nécessaire, puisque le fichier fit contient toutes les informations du fichier *.sur. Il est néanmoins souvent plus rapide de commencer par un fichier surface, en lisant celui-ci dans ROD et en le sauvant comme fichier fit en utilisant la commande de liste (par exemple, ROD>list, test.fit convenable). Ceci produira un fichier fit qui pourra être utilisé plus tard.

5.3.4 Les fichiers de données (*.dat)

Les fichiers de données (*.dat) ont le format suivant :

Première ligne :	`comments`
Les lignes suivantes :	`h k l` f_{data} `sigma [lbragg]`

C'est une liste d'index de diffraction (*hkl*) suivis du module de facteur de structure (f_{data}) mesuré et l'erreur type (sigma) du facteur de structure. Les valeurs « lBragg » sont facultatives. Ce sont ces fichiers qui vont nous permettre de retrouver la structure du cristal. Ces fichiers peuvent être générés par l'utilisateur à l'aide de ROD (pour fabriquer des fichiers de données « simulées », afin de tester les procédures de fit) ou alors édités à la main. Les fichiers en question sont générés par le programme ANA à partir du fichier contenant toutes les mesures de l'expérience. Après nous verrons comment générer ce type de fichier avec ROD pour pouvoir faire des tests.

5.3.5 Autres formats de fichiers

Dans la suite de notre travail nous allons utiliser deux autres types de fichiers (les fichiers *.par et *.mac) qui ont tous deux le même format. Ces fichiers vont contenir tous les deux des commandes en lignes qui seront lues par ROD. Les fichiers macro (*.mac) sont des fichiers de commandes en lignes qui vont permettre d'exécuter automatiquement la plupart des commandes que l'on rentre normalement « à la main » dans ROD et donc permettre entre autre de lire des fichiers, activer des extensions ou encore entrer des atomes, les fichiers *.par contiennent aussi des commandes en lignes mais par contre ils vont servir à entrer des paramètres (ex. : des paramètres pour le « fitting »).

5.4 Géométrie et définitions des différentes variables dans ROD

5.4.1 Les paramètres de maille

Voici les six paramètres de maille ainsi que les variables (**Maille**) :

a = DLAT[0] (direct lattice parameter)
b = DLAT[1]
c = DLAT[2]
alpha = DLAT[3]
beta = DLAT[4]
gamma = DLAT[5]

Ces données représentent les paramètres de la maille (voir au dessus **2.2.3 La Maille**), ainsi on va avoir besoin de matrices de passage qui vont nous permettre de passer d'un système orthogonal (dans lequel sont définies les positions des atomes du groupe) à un système du cristal et vice versa. Pour ce faire on doit calculer les paramètres réciproques (ref.[4]) de la maille (RLAT[] , reciprocal lattice parameter) en utilisant la fonction « calc_rlat(dlat[] , rlat[]) » (dans **read.c**). On va pouvoir calculer les deux matrices en appelant la fonction « calculate_transformation_matrices( ) » (dans **svensson.c**), celle-ci va retourner les matrices de passages d'un système à un autre (matrice de changement de repère). La matrice **MMATRIX[][]** va nous permettre de passer de coordonnées orthogonales à des coordonnées du cristal et la matrice **MINVMATRIX[][]** qui va calculer des coordonnées orthogonales (par ortho normalisation du repère ($\vec{a},\vec{b},\vec{c}$)) à partir de coordonnées du cristal.

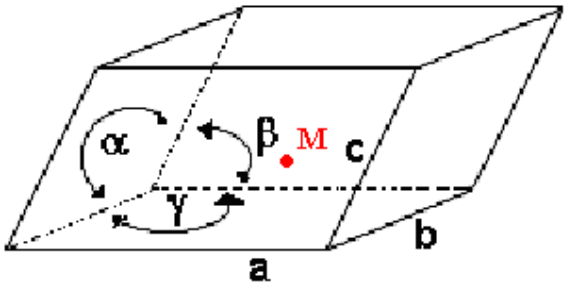


fig.11 *Maille de surface.*

$$\vec{OM} = \begin{pmatrix} XGROUPS[groupe] \\ YGROUPS[groupe] \\ ZGROUPS[groupe] \end{pmatrix},$$ les coordonnées du centre de rotation du groupe dans la maille de surface, dans le repère (O, $\vec{a},\vec{b},\vec{c}$).

5.4.2 Le groupe d'atomes à la surface du cristal

$$O'\vec{M} = \begin{pmatrix} ORIGINOFATOMS[0][groupe] \\ ORIGINOFATOMS[1][groupe] \\ ORIGINOFATOMS[2][groupe] \end{pmatrix},$$ les coordonnées du centre de rotation du groupe d'atomes dans le repère (O', x, y, z)

$$O'\vec{A} = \begin{pmatrix} XATOMINGROUPS[groupe][atomeA1] \\ YATOMINGROUPS[groupe][atomeA1] \\ ZATOMINGROUPS[groupe][atomeA1] \end{pmatrix},$$ les coordonnées des atomes (A1, A2, …) du groupe dans le repère orthogonal (O', x, y, z).

Et les trois angles de rotation phi, chi et thêta qui sont représentés par : PHIGROUPS[groupe], CHIGROUPS[groupe] et THGROUPS[groupe]. Ces trois valeurs donnent l'orientation du groupe d'atome à la surface du cristal (voir la figure ci-dessous). Pour le calcul des coordonnées après rotation, on utilise la matrice de rotation « **RMATRIX[][]** ».

La rotation est décomposée en trois rotations :

- Rotation d'angle phi autour de l'axe des z.
- Rotation d'angle chi autour de l'axe des x.
- Rotation d'angle thêta autour de l'axe des z.

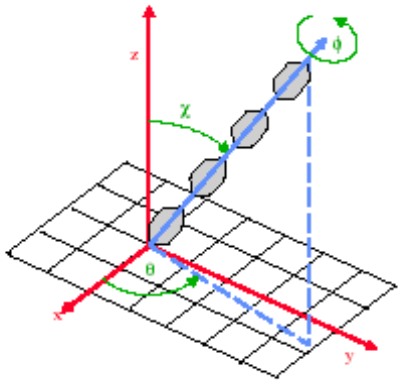


fig.12 *Orientation du groupe d'atomes.*

On a :

$$R_z(th) = \begin{pmatrix} \cos(th) & -\sin(th) & 0 \\ \sin(th) & \cos(th) & 0 \\ 0 & 0 & 1 \end{pmatrix},$$ la matrice de rotation d'angle th autour de l'axe z.

$$R_x(chi) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos(chi) & -\sin(chi) \\ 0 & \sin(chi) & \cos(chi) \end{pmatrix}$$, la matrice de rotation d'angle chi autour de l'axe x.

$$R_z(phi) = \begin{pmatrix} \cos(phi) & -\sin(phi) & 0 \\ \sin(phi) & \cos(phi) & 0 \\ 0 & 0 & 1 \end{pmatrix}$$, la matrice de rotation d'angle phi autour de l'axe z.

Ce qui va nous donner la matrice de rotation : **RMATRIX[][]** = $R_z(th)R_x(chi)R_z(phi)$.
Donc si on note **A1'** l'image de **A1** par la rotation, on a :

$\vec{MA1'}$ = RMTRIX[][] × $\vec{MA1}$, dans le repère (O', x, y, z).

Dans la suite, nous allons traiter des deux déformations vues au dessus, tout d'abord la dilatation avec la compression et ensuite le cisaillement. Nous allons voir que dans chacun des cas, nous avons plusieurs paramètres (soient fixes, soient fixés par les utilisateurs) et des contraintes que nous définirons dans chaque partie.

Tout comme pour les déplacements « en bloc » du groupe, on va utiliser des « serial number » pour les déformations, ceux-ci seront différents.

5.5 Les déformations

5.5.1 La dilatation / compression

Dans un premier temps, il faut définir le centre de la déformation (on note C le centre de la déformation) celui-ci peut être :

- Soit le centre géométrique du groupe d'atomes (soit N le nombre d'atomes dans le groupe traité).
- Soit le centre de la rotation du groupe d'atome (le centre O).
- Soit un centre saisi par l'utilisateur.

Par défaut ce centre de déformation est le centre du groupe d'atomes.

Dans un deuxième temps, si on veut faire une dilatation ou une compression il faut entrer une quantité de déformation (en %), cette quantité est soit positive dans le cas de la dilatation soit négative dans le cas d'une compression.

Par la suite on notera :

- dx = % de déformation sur x (variation relative d'un segment initialement parallèle à l'axe des x).
- dy = % de déformation sur y.
- dz = % de déformation sur z.

donc on va avoir :

A1 = $\begin{pmatrix} x_{A1} \\ y_{A1} \\ z_{A1} \end{pmatrix}$ les coordonnées d'un atome du groupe (dans le repère orthogonal), et

C = $\begin{pmatrix} Cx \\ Cy \\ Cz \end{pmatrix}$ le centre de déformation.

Alors, si on a **A1'** issu de **A1** après la déformation on a :

A1' = $\begin{pmatrix} x_{A1} + dx/100 \times (x_{A1} - Cx) \\ y_{A1} + dy/100 \times (y_{A1} - Cy) \\ z_{A1} + dz/100 \times (z_{A1} - Cz) \end{pmatrix}$

Comme c'est préciser au-dessus nous allons utiliser des « serial number » (voir **5.3.3** Les « serial number ») pour les déformations. Dans le cas de la dilatation/compression nous utiliserons les tableaux suivants :

- `SNEXPX[groupe]` = numéro de série de la déformation sur x (pour un **groupe** donné).
- `SNEXPY[groupe]` = numéro de série de la déformation sur y.
- `SNEXPZ[groupe]` = numéro de série de la déformation sur z.

Si pour un même groupe on a plusieurs numéros de série qui sont positifs, alors on va avoir des déformations sur les axes correspondants, et si ces numéros sont différents on peut avoir une quantité de déformation différente sur x, y et z.

Pour une dilatation/compression, l'utilisateur peut décider de conserver le volume du groupe d'atomes pendant la déformation. Mais dans ce cas, l'utilisateur doit entrer deux numéros de série positifs pour ce groupe (sinon on a un message d'erreur), ensuite il faut faire attention à ne pas faire de conservation de volume dans le cas d'un groupe d'atomes plan.

Une fois les deux quantités voulues saisies (dans le cas de la conservation du volume), le calcul de la troisième quantité de déformation se fait avec l'appel de la fonction « **calc_comp_th** (*choix de la composante recherchée, le numéro de groupe, première quantité, deuxième quantité*) » (voir **annexe 2.5**). Cette fonction retourne la troisième quantité recherchée. Avant la déformation, la fonction va calculer tout d'abord le centre géométrique du groupe d'atomes, puis les distances maximums entre ce centre et les atomes les plus éloignés (du centre) sur les trois axes (x, y, z), en appelant la fonction « **max_dist** (*le centre, le groupe, l'axe*) » (voir **annexe 2.6**). On va donc obtenir trois longueurs (L1, L2, L3) et on fait de même après la déformation (avec les deux quantités données) ainsi on va avoir deux nouvelles longueurs ce qui nous permettra de trouver la troisième et donc dans le même temps la troisième quantité.

Exemple avec un parallélépipède.

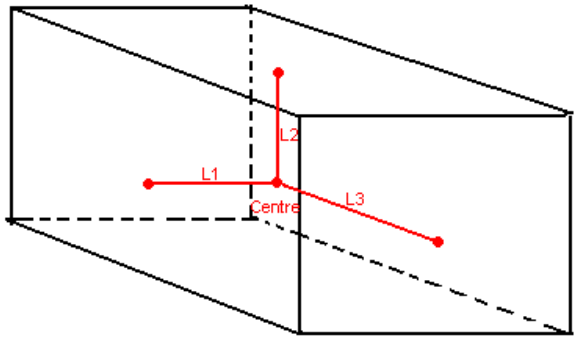


Fig.13 *Avant déformation.*

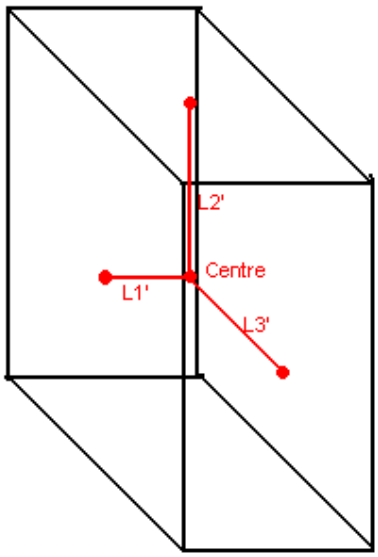


Fig.14 *Après déformation.*

En effet, si on a conservation du volume alors on impose : **L1**×**L2**×**L3** = **L1'**×**L2'**×**L3'** (fig.12 & 13), nous n'avons pas d'outils pour calculer le volume d'un groupe d'atomes (sachant que celui-ci peut avoir des formes totalement différentes) mais nous pouvons trouver une approximation de celui-ci. Cette formule est exacte pour des molécules en forme d'ellipsoïde (ou formes sphériques).

5.5.2 Le cisaillement

Si on considère la matrice S de déformation, les six termes non diagonaux de cette matrice vont correspondre aux composantes de cisaillement.

On a donc la matrice [S] suivante :

$$[S] = \begin{bmatrix} 1 & \frac{1}{2}e_{xy} & \frac{1}{2}e_{xz} \\ \frac{1}{2}e_{xy} & 1 & \frac{1}{2}e_{yz} \\ \frac{1}{2}e_{xz} & \frac{1}{2}e_{yz} & 1 \end{bmatrix}$$

On va considérer un atome de coordonnées (dans le repère orthogonal) $\begin{pmatrix} x \\ y \\ z \end{pmatrix}$ avant la déformation et $\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix}$ après le cisaillement (dans le même repère), alors :

$$\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} = \begin{bmatrix} 1 & \frac{1}{2}e_{xy} & \frac{1}{2}e_{xz} \\ \frac{1}{2}e_{xy} & 1 & \frac{1}{2}e_{yz} \\ \frac{1}{2}e_{xz} & \frac{1}{2}e_{yz} & 1 \end{bmatrix} \cdot \begin{pmatrix} x \\ y \\ z \end{pmatrix}$$

Comme on a pu voir au-dessus, on peut interpréter e_{yz} comme la variation de l'angle (Oy, Oz) de -2α (où $\alpha = \tan^{-1}(\frac{1}{2} \cdot e_{yz})$) pendant la déformation, ces axes restant liés au solide; e_{zx} et e_{xy} s'interprètent enfin de façon analogue.

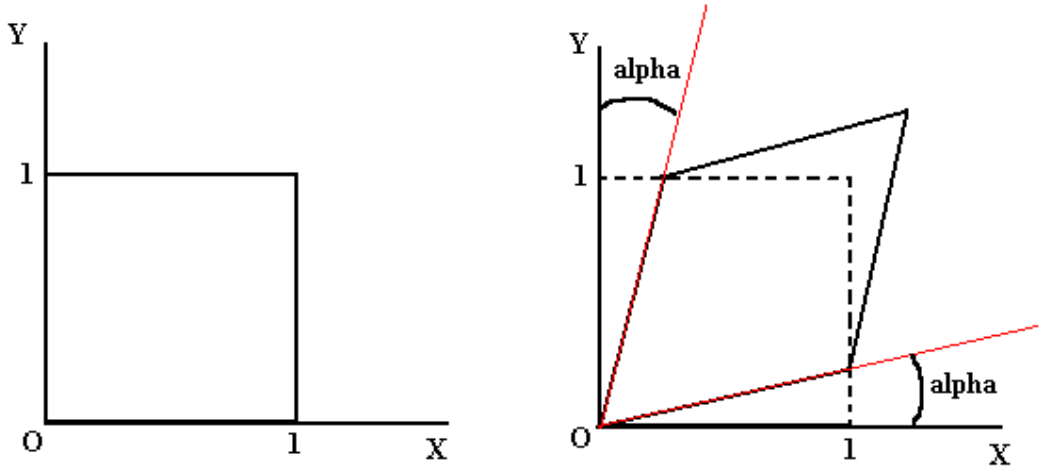


Fig.15 *Exemple de déformation dans le plan (O, x, y) avec* $e_{xy} = 2 \tan \alpha$.

Comme pour la dilatation/compression on va utiliser des « serial number » (voir **5.3.3**) pour le cisaillement. On va avoir trois numéros de série correspondant aux trois composantes de cisaillement explicitées au-dessus :

`SNSHEARXY [groupe]` = numéro de série du paramètre de cisaillement e_{xy}.
`SNSHEARXZ [groupe]` = numéro de série du paramètre de cisaillement e_{xz}.
`SNSHEARYZ [groupe]` = numéro de série du paramètre de cisaillement e_{yz}.

Exemples de cisaillement.

Par la suite on va considérer un cube d'arête 2 centré en O (fig.15), que l'on va déformer en utilisant la matrice de déformation [S] (voir au-dessus) pour différentes valeurs des paramètres de déformation (e_{xy}, e_{yz} et e_{xz} seront pris << 1 car on traite des petites déformations).

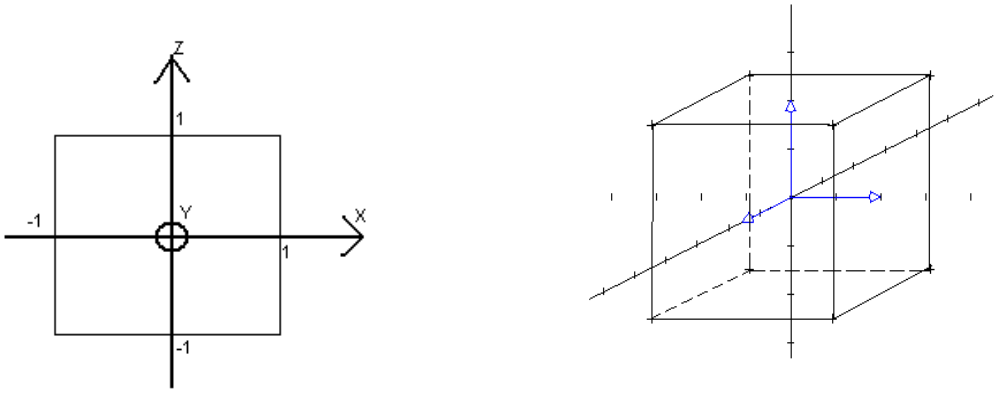


Fig.16 *Représentation du cube d'arête 2 dans le plan et dans le repère (O, x, y, z).*

Exemple 1 :

Ici on a : $e_{xy} = 0$, $e_{xz} = \frac{1}{4}$, $e_{yz} = -\frac{1}{2}$

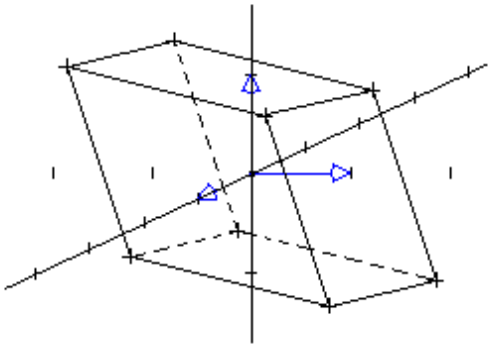


Fig.17 *Représentation dans le repère après déformation.*

Exemple 2 :

Ici on a : $e_{xy} = -\frac{1}{2}$, $e_{xz} = -\frac{1}{4}$, $e_{yz} = \frac{1}{2}$

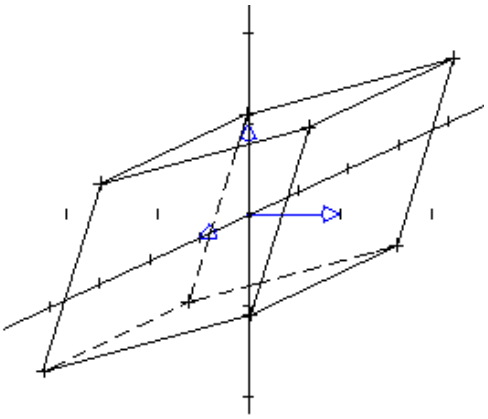


Fig.18 *Représentation dans le repère après déformation.*

Exemple 3 :

Ici on va prendre : $e_{xy} = \frac{1}{2}$, $e_{xz} = \frac{1}{2}$, $e_{yz} = \frac{1}{2}$

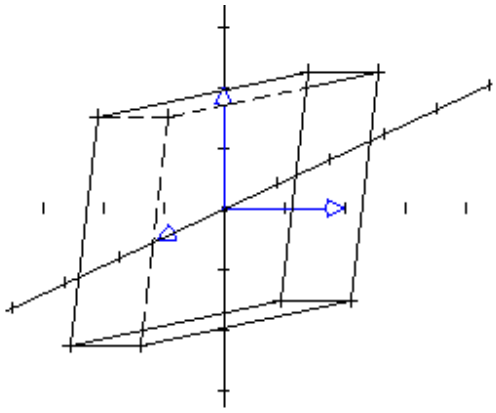


Fig.19 *Représentation dans le repère après déformation.*

5.6 Utilisation des déformations dans le programme ROD

Nous venons de voir comment sont faites les différentes déformations, maintenant nous allons voir comment utiliser ces déformations dans le programme ROD et comment faire la saisie des différentes variables. Nous allons également observer les modifications apportées au fichier fit (*.fit). Dans ROD, on va pouvoir trouver le menu de déformation dans le menu ROD.EXT.SVE.DEF> (fig.20).

```
ROD.EXT.SVE.SET.DEF>h
********************************** MIK MENU ****************************
 Parameter : Model par. for fitting group(s)
 Expansion : Value of group expansion parameter
 Shearing  : Value of group shearing parameter
 LParam    : List parameter
 LAtoms    : List atoms
 Help      : Help
 Return    : Close menu/no deformation
ROD.EXT.SVE.SET.DEF>
```

Fig.20 *Menu des déformations.*

Donc, si on a entré un groupe au préalable (sinon impossible) on va pouvoir saisir les paramètres de déformations. Cette opération se fait tout d'abord dans le menu ROD.EXT.SVE.SET.DEF.PAR> (fig.21).

```
****************************** DEFORMATION PARAMETERS ******************
 CEnterexp : Center of expansion
 COnservol : Conserving of the volume for the expan
 XExpan    : Serial number of x-expansion parameter
 YExpan    : Serial number of y-expansion parameter
 ZExpan    : Serial number of z-expansion parameter
 XYshear   : Serial number of xy-shear parameter
 ZXshear   : Serial number of xz-shear parameter
 YZshear   : Serial number of yz-shear parameter
 Help      : Display menu
 Return    : Return to main menu
ROD.EXT.SVE.SET.DEF.PAR>
```

Fig.21 *Menu des paramètres de la déformation.*

Dans ce menu, on va pouvoir entrer les « serial numbers » (voir **5.3.3**) des différentes déformations, choisir le centre de la dilatation/compression et demander à l'utilisateur si celui-ci veut conserver le volume du groupe d'atome. Une fois entrés les numéros de série on peut saisir les valeurs des déformations dans le menu ROD.EXT.SVE.DEF> avec les commandes « expansion » et « shearing » (fig.22), on peut aussi entrer un intervalle avec les commandes « lower limit » et « upper limit ». Et on a une variable « range checking » qui va permettre, si on la met à « yes », de faire une recherche des valeurs dans l'intervalle donné.

```
********************************** MIK MENU ***********************************
 Parameter : Model par. for fitting group(s)
 Expansion : Value of group expansion parameter
 Shearing  : Value of group shearing parameter
 LParam    : List parameter
 LAtoms    : List atoms
 Help      : Help
 Return    : Close menu/no deformation
ROD.EXT.SVE.SET.DEF>ex
Serial number of group expansion parameter: 1
Value of expansion parameter 1 [ 0.000]: 5
Lower limit of expansion 1 [ 0.000]:
Upper limit of expansion 1 [ 0.000]: 10
Range checking on expansion 1 [NO]:
ROD.EXT.SVE.SET.DEF>sh
Serial number of group shearing parameter: 1
Value of shearing parameter 1 [ 0.000]: 0.5
Lower limit of shearing 1 [ 0.000]:
Upper limit of shearing 1 [ 0.000]: 1
Range checking on shearing 1 [NO]:
```

Fig.22 *Exemple de déformation.*

Liste des paramètres de déformation avec la commande « lpar » (fig.23).

```
ROD.EXT.SVE.SET.DEF>lpar
Group number : 1
 1 Group Expan =   5.0000 [  0.0000 ,  10.0000]
 2 Group Expan =   0.0000 [  0.0000 ,   0.0000]
 3 Group Expan =   0.0000 [  0.0000 ,   0.0000]
 4 Group Shear =   0.5000 [  0.0000 ,   1.0000]
 5 Group Shear =   0.0000 [  0.0000 ,   0.0000]
 6 Group Shear =   0.0000 [  0.0000 ,   0.0000]
ROD.EXT.SVE.SET.DEF>
```

Fig.23 *Liste des paramètres.*

Pour la suite, et pour simplifier on va modifier les fichiers fit (*.fit) en ajoutant une ligne dans ces derniers avec les six « serial number » des déformations. Voici comment vont se présenter maintenant les fichiers fit :

```
Première ligne : commentaire

Seconde ligne : a   b   c   alpha   bêta   gamma       (pamètres de maille)

Définition des groupes :

Group « nom du groupe »    [ origine-x     origine-y    origine-z  ]

Xstart   SN-dépx   Ystart   SN-dépy   Zstart   SN-dépz    phi-start   SN-rotp
chi-start  SN-rotc   th-start  SN-rott

SNEXPX SNEXPY SNEXPZ SNSHEARXY SNSHEARZX SNSHEARYZ

Type d'atome 1    x-atome1  y-atome1 z-atome1 [Debye Waller serial number1]
Type d'atome 2    x-atome2  y-atome2 z-atome2 [Debye Waller serial number2]
……

endgroup
```

De cette manière, l'utilisateur n'a plus qu'à saisir les différentes valeurs des déformations dans ROD ou à lire un fichier de paramètres (*.par).

5.7 Amélioration du menu « plot »

Le menu plot se présente comme suit :

```
ROD>plot h
********************************** PLOT MENU **********************
 Bulk      : Bulk contribution (against l)
 SURface   : Surface contribution (against l)
 Sum       : Interference sum of bulk and surface
 All       : Above three curves (against l)
 Data      : Rod data (against l)
 BOth      : Data plus calculated interference sum
 MULTIBoth : Command both (multi)
 FTheory   : Theoretical structure factors
 FData     : Experimental structure factors
 FBoth     : Theoretical and experimental f's
 MOriginal : Original structure model
 MRefined  : Refined structure model
 MBoth     : Original + refined structure model
 MLarge    : Several unit cells of refined model
 M3d       : Refined model, PLOT3D output file
 CUt3d     : Cut trough scattering plane, PLOT3D
 MSI       : Refined model, MSI output file
 DPatterson: Patterson function of experimental data
 TPatterson: Patterson of theoretical structure f's
 DIfference: Electron density difference map
 ELectron  : Electron density map of model
 Errors    : Set error bar plotting on/off
More, press <return> to continue
********************************** PLOT MENU **********************
 Unit      : Set drawing of unit cell on/off
 Help      : Display menu
 Return    : Return to main menu
ROD.PLOT>
```

Fig.24 *Menu « plot ».*

La commande « both » permet de tracer pour chaque index de diffraction « *l* » la valeur du facteur de structure f_{data} (par des points) expérimental ainsi que le facteur de structure calculé f_{calc} (par une courbe) (fig.25).

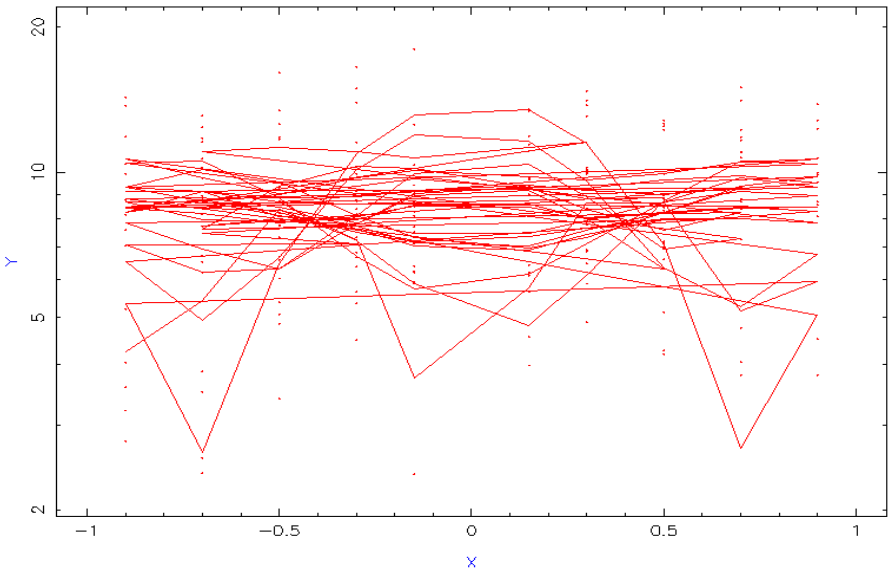


Fig.25 *Exemple de la commande « both ».*

Comme on peut le voir sur la figure au-dessus les points représentent les facteurs de structure donnés (mesurés) et les courbes représentent les facteurs de structures calculés à partir du modèle. Dans une même fenêtre on a les facteurs de structures pour tous les index de diffraction « *h et k* ». Ce que l'on cherche à obtenir, c'est une représentation séparée pour chaque index de diffraction (pour chaque valeur de *h* et *k*) dans une même fenêtre. Pour ce faire j'ai utilisé la fonctionnalité « multiplot » déjà présente dans ROD de manière à obtenir ce que l'on cherche. Ainsi l'utilisateur va pouvoir utiliser la commande « multiboth ».

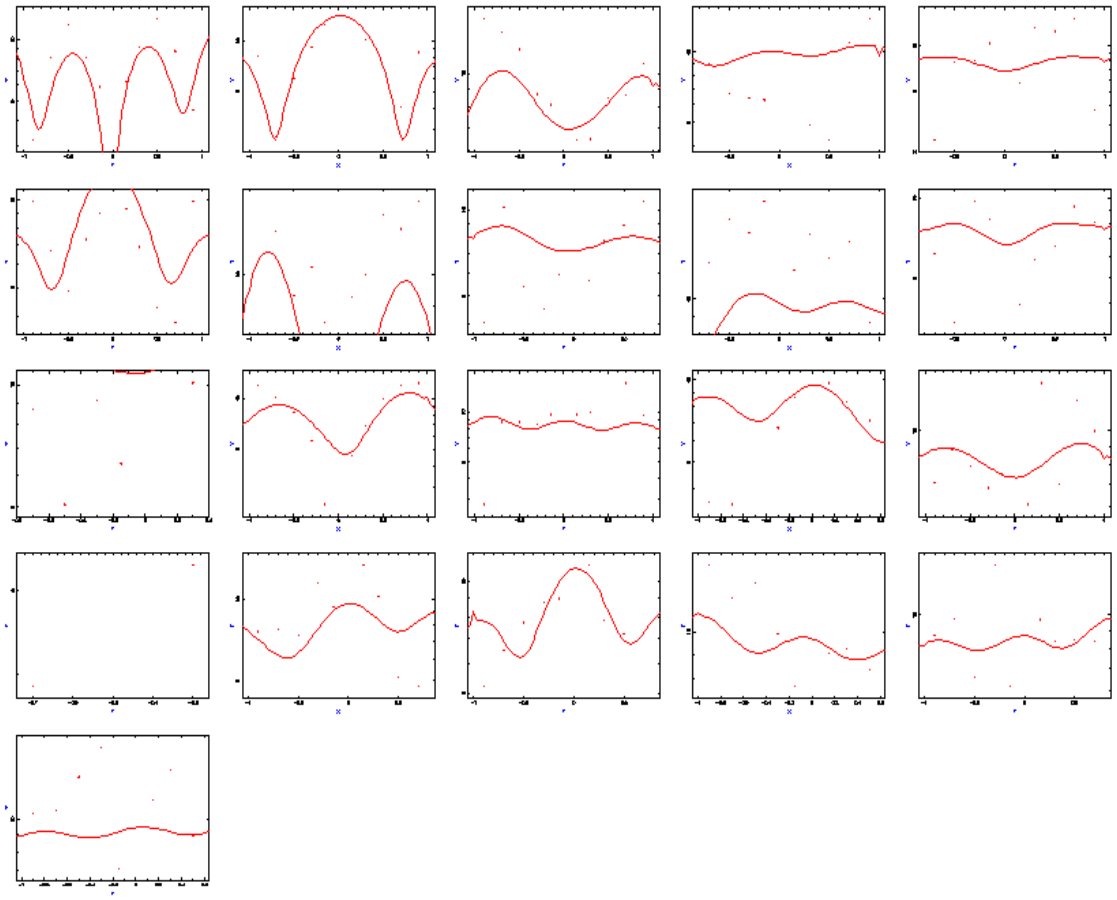


Fig.26 *Même exemple qu'au-dessus avec la commande « multiboth ».*

On voit bien que pour chaque couple (*h, k*) de diffraction on a une fenêtre différente sur l'écran.

CHAPITRE 6 : LES TESTS

Dans cette partie, je vais faire plusieurs tests, avec des *données simulées*. Et expliquer comment obtenir des données simulées avec les différents exemples.

6.1 Simulation de données

Comme on a pu voir auparavant les données peuvent être lues dans un fichier de données (*.dat) ou comme on va voir maintenant peuvent être simulées avec ROD. Tout d'abord on va lire un fichier surface (*.sur) représentant une molécule non déformé, puis on va faire des déformations avec ROD, ensuite dans le menu ROD.LIST > on va utiliser la commande « surface model » pour pouvoir enregistrer un nouveau fichier (de surface) représentant la molécule déformée. Pour simuler des données, on va lire ce nouveau fichier de surface et dans un deuxième temps on va utiliser la commande « Range » dans le menu ROD.CALC> qui va calculer le facteur de structure pour une gamme d'index de diffraction *h* et *k* donnés et pour *l* fixé.

```
******************************** CALCULATION MENU ************************
 ROd       : Calculate rod profile
 RAnge     : Calculate f's for range of h and k
 Qrange    : Calculate f's within q-max
 Data      : Calculate f's for all data points
 DIstance  : Calculate the distance between two atoms
 Angle     : Calculate bond angle between three atoms
 LEffective: Calculate effective l from slit width
 ROUghness : Calculate roughness in atomic layers
 Help      : Display menu
 Return    : Return to main menu
ROD.CALC>range
Start value of diffraction index h [0.0]: -1
End value of diffraction index h [0.0]: 1
Step size of diffraction index h [0.0]: 1
Start value of diffraction index k [0.0]: -1
End value of diffraction index k [0.0]: 1
Step size of diffraction index k [0.0]: 1
Diffraction index l [0.0]: 0.2
ROD>list sum t
     h      k      l       f-sum    phase
-1.000 -1.000  0.200    11.27918     0.00
-1.000  0.000  0.200    13.34289     0.00
-1.000  1.000  0.200    11.27918     0.00
 0.000 -1.000  0.200    13.34289     0.00
 0.000  1.000  0.200    13.34289     0.00
 1.000 -1.000  0.200    11.27918     0.00
 1.000  0.000  0.200    13.34289     0.00
 1.000  1.000  0.200    11.27918     0.00
```

Après avoir calculé les facteurs de structure pour des index de diffraction donnés, on peut enregistrer ces données simulées dans un fichier de données (*.dat) dans le menu ROD.EXT.SVE > et avec la commande « Simdata ». Cette commande va créer un fichier *.dat, du nom que l'utilisateur lui aura donné et dans lequel seront écrites toutes les données simulées.

```
********************************* SVENSSON MENU ************************
 Activate  : activate/deactivate Svensson's extension
 Set       : Set parameters for Svensson's extension
 Bulkmult  : Expand bulk unit cell
 Overlap   : Check for overlapping atoms
 SImdata   : Sum output for simulation
 Help      : Display menu
 Return    : Return to main menu
ROD.EXT.SVE>sim
Filename (.dat) (type 't' or <return> for terminal): example
Comments: example of simulation data
ROD.EXT.SVE>
ROD.EXT>
```

```
ROD>r data example
example of simulation data
ROD>list dat t
     h      k      l       f-dat       sigma lbr
-1.000 -1.000  0.200    11.27918     3.36000   0
-1.000  0.000  0.200    13.34289     3.65000   0
-1.000  1.000  0.200    11.27918     3.36000   0
 0.000 -1.000  0.200    13.34289     3.65000   0
 0.000  1.000  0.200    13.34289     3.65000   0
 1.000 -1.000  0.200    11.27918     3.36000   0
 1.000  0.000  0.200    13.34289     3.65000   0
 1.000  1.000  0.200    11.27918     3.36000   0
```

Pour chaque test, je vais lire un fichier surface (*.sur) correspondant à une molécule (ou une figure géométrique) sans déformation, puis je vais lire un fichier de données simulées (*.dat) obtenu après la lecture du fichier surface (*.sur) correspondant à la molécule déformée. Par la suite, pour pouvoir tester les déformations dans le menu ROD.FIT >, je vais lancer la commande « run » qui va effectuer des calculs et retourner les valeurs « quisqr » et « normalized ». Ces deux valeurs représentent l'erreur entre le modèle actuel et celui recherché, donc si on veut se rapprocher le plus possible du modèle recherché, on doit minimiser les deux quantités « quisqr » et « normalized ». Dans le cas de données simulées, on doit obtenir les deux valeurs égales à zéro.

6.2 Test 1 : Simulation d'un cube

On va considérer un cube d'arête 2 centré en 0 (fig.27) que l'on va déformer. Dans cet exemple on va considérer seulement les atomes à la surface qui seront entrés dans un fichier surface (voir **annexe 3.1**).

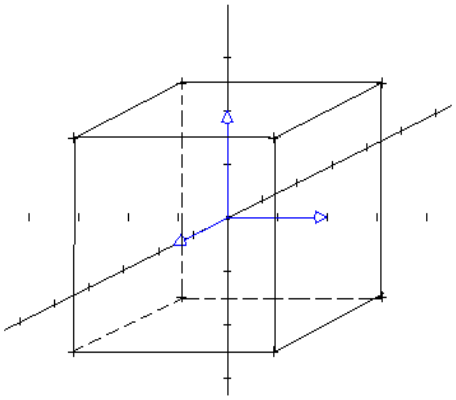


fig.27 *Représentation du cube dans le repère orthogonal.*

6.2.1 Dilatation/compression avec conservation du volume

Ici j'ai introduit une déformation de type dilatation/compression avec une conservation du volume. On a donc dx = - 33,333 % et dy = 50 % (voir le fichier surface obtenu après déformation, **annexe 3.1**).

```
ROD>fit run
Fit results:
              Scale =          1       <FIXED>
             Scale2 =          1       <FIXED>
               Beta =          0       <FIXED>
    Surface fraction =          1       <FIXED>

chisqr = 105.5352, normalized = 2.1987, Q = 0.00000
```

On voit bien que si on lance la commande « run » après avoir lu les deux fichiers sans avoir entrer de déformation, on obtient des valeurs de « quisqr » et « normalized » non égales à zéro. Donc dans le menu ROD.EXT.SVE.SET.DEF.PAR >, on va entrer les différents paramètres de la déformation, les « serial number » et la « conservation du volume » (voir **5.3.3**).

```
ROD>ex sv se de pa
ROD.EXT.SVE.SET.DEF.PAR>xe
Group number: 1
Serial number of x-expansion parameter [0]: 1
ROD.EXT.SVE.SET.DEF.PAR>ye
Group number: 1
Serial number of y-expansion parameter [0]: 2
ROD.EXT.SVE.SET.DEF.PAR>con
Group number: 1
Conserving of the volume ? [NO]: y
```

Ensuite, pour chaque « serial number », on va entrer les différentes valeurs de chaque déformation, et si on veut que la recherche se fasse dans un certain intervalle de valeurs il suffit de saisir les champs « lower limit » (borne inférieure) et « upper limit » (borne supérieure). Pour qu'il n'y ait pas pendant la recherche de valeurs qui sortent des bornes, il suffit de fixer « range checking » à « yes ».

```
ROD.EXT.SVE.SET.DEF>ex
Serial number of group expansion parameter: 1
Value of expansion parameter 1 [ 0.000]: -30
Lower limit of expansion 1 [ 0.000]: -40
Upper limit of expansion 1 [ 0.000]: -30
Range checking on expansion 1 [NO]: y
ROD.EXT.SVE.SET.DEF>ex
Serial number of group expansion parameter: 2
Value of expansion parameter 2 [ 0.000]: 45
Lower limit of expansion 2 [ 0.000]: 45
Upper limit of expansion 2 [ 0.000]: 55
Range checking on expansion 2 [NO]: y
ROD.EXT.SVE.SET.DEF>lpar
Group number : 1
 1 Group Expan = -30.0000 [-40.0000 , -30.0000]  cheked
 2 Group Expan =  45.0000 [ 45.0000 ,  55.0000]  cheked
```

Puis dans le menu ROD.FIT >, on utilise la commande « loose » pour permettre aux paramètres de déformation de varier pendant le fit (la procédure de fit fait varier ces paramètres jusqu'à obtention d'un minimum pour « chisqr »).

« Chisqr » représente la somme des distances au carré entre f_{data} et f_{calc}, « normalized » est calculé pareil mais divisé par le nombre de points.

```
ROD.FIT>run
Iteration stopped afer converging.

Chi2 increase for error estimate (0 = use covariance matrix): [1.0]
Fit results:
               Scale =          1       <FIXED>
              Scale2 =          1       <FIXED>
                Beta =          0       <FIXED>
    Surface fraction =          1       <FIXED>
      Group expan  1 =     -33.33 +/- 0.6044        [       -40 , -30       ]
      Group expan  2 =      49.82 +/- 3.014         [        45 , 55        ]

chisqr = 0.0000, normalized = 0.0000, Q = 1.00000
```

Après avoir entrer les déformations, on trouve « chisqr » et « normalized » égaux à zéro, donc on a bien retrouvé le modèle recherché. Et on retrouve bien les valeurs des déformations.

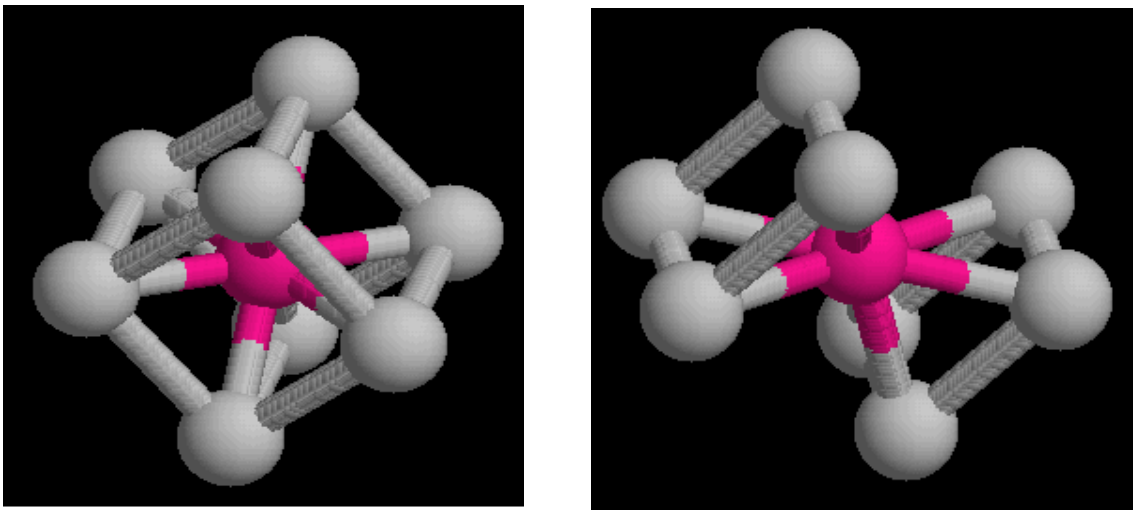


Fig.28 *Représentation du cube sans déformation (à gauche), puis avec déformations (à droite).*

Dans l'exemple que nous venons de voir et dans ceux qui vont suivre, j'ai choisi d'entrer des intervalles petits pour les déformations, pour pouvoir retrouver le modèle recherché. En effet, la méthode d'optimisation ne permet pas de retrouver le minimum global, il arrive que l'on trouve des minimums locaux. On peut retrouver le modèle mais en faisant plusieurs tests sur les valeurs ou en ne laissant « libre » qu'un seul paramètre à la fois.

6.2.2 Cisaillement

On prend la même figure et on fait un cisaillement avec : $e_{xy} = -0.2$, $e_{xz} = 0.1$ et $e_{yz} = 0.4$. (voir le fichier surface obtenu après déformation, **annexe 3.1**).

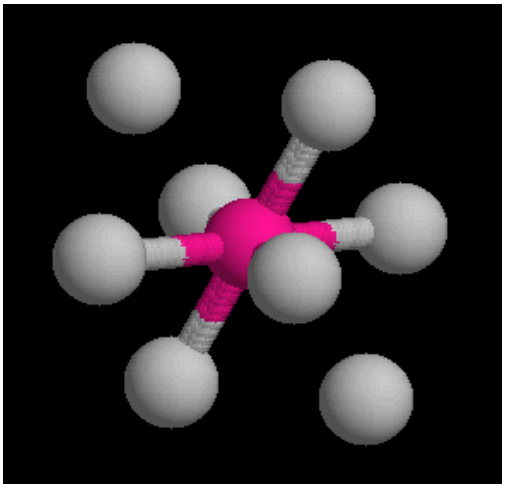


Fig.29 *Représentation du cube après le cisaillement.*

```
ROD>fit run
Fit results:
               Scale =      0.602       <FIXED>
              Scale2 =          1       <FIXED>
                Beta =          0       <FIXED>
    Surface fraction =          1       <FIXED>

chisqr = 492.2580, normalized = 10.2554, Q = 0.00000
```

On n'a pas introduit de déformation donc on a « chisqr » et « normalized » qui sont grands.

```
ROD>fit run
Iteration stopped with diverging lambda.

Chi2 increase for error estimate (0 = use covariance matrix): [1.0]
Fit results:
               Scale =      0.602 +/- 0.07635
              Scale2 =          1       <FIXED>
                Beta =          0       <FIXED>
    Surface fraction =          1       <FIXED>
      Group shear  1 =    -0.1398 +/- 0.008276      [      -0.3 , -0.1      ]
      Group shear  2 =    0.09458 +/- 0.02558       [        0 , 0.2        ]
      Group shear  3 =     0.3618 +/- 0.02536       [      0.3 , 0.5        ]

chisqr = 8.9936, normalized = 0.2044, Q = 1.00000
```

Ici on introduit la déformation (cisaillement) avec des intervalles assez grands.

```
ROD.FIT>run
Iteration stopped with diverging lambda.

Chi2 increase for error estimate (0 = use covariance matrix): [1.0]
Fit results:
               Scale =      0.602       <FIXED>
              Scale2 =          1       <FIXED>
                Beta =          0       <FIXED>
    Surface fraction =          1       <FIXED>
      Group shear  1 =       -0.2 +/- 0.006564      [     -0.25 , -0.15     ]
      Group shear  2 =        0.1 +/- 0.03342       [      0.05 , 0.15      ]
      Group shear  3 =        0.4 +/- 0.02943       [      0.35 , 0.45      ]

chisqr = 0.0000, normalized = 0.0000, Q = 1.00000
```

On remarque que si l'on réduit l'intervalle, on retrouve de meilleures valeurs de la déformation.

6.2.3 Dilatation/compression et cisaillement

Dans ce test, j'ai pris : dx = 5 %, dy = 2 %, dz = -3 %, e_{xy} = 0.2 et e_{xz} = 0.1.

```
ROD>fit run
Fit results:
              Scale =      1.045       <FIXED>
             Scale2 =          1       <FIXED>
               Beta =          0       <FIXED>
   Surface fraction =          1       <FIXED>

chisqr = 2251.7695, normalized = 46.9119, Q = 0.00000
```

Sans déformation, on voit bien que les deux valeurs à minimiser sont grandes.

```
ROD.FIT>run
Fit results:
              Scale =      1.045       <FIXED>
             Scale2 =          1       <FIXED>
               Beta =          0       <FIXED>
   Surface fraction =          1       <FIXED>
     Group expan  1 =          5       <FIXED>
     Group expan  2 =          2       <FIXED>
     Group expan  3 =         -3       <FIXED>
     Group shear  1 =        0.2       <FIXED>
     Group shear  2 =        0.1       <FIXED>

chisqr = 0.2170, normalized = 0.0045, Q = 1.00000
```

Alors qu'avec une déformation, on obtient une bonne approximation.

6.3 Test 2 : Simulation d'une molécule sphérique (C60 fulrène)

Dans cet exemple, je vais travailler avec une molécule de C60 (molécule de carbone avec une géométrie type « ballon de football » fulrène) en simulant les données. Dans cette partie, comme pour le premier exemple, on va lire un fichier (*.sur) qui contient les coordonnées des atomes du C60 et que l'on va déformer. Par contre, un test sera effectuer avec un substrat.

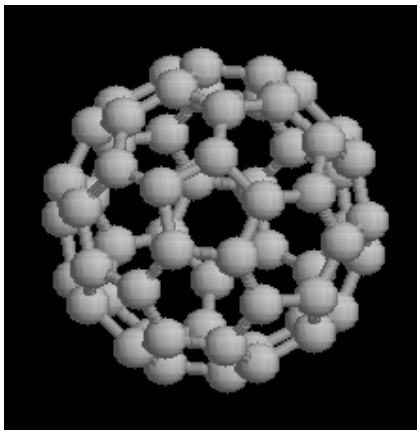


Fig.30 *Représentation de la molécule de C60.*

6.3.1 Dilatation/compression avec conservation du volume

Dans ce test j'ai pris : dx = 5 et dy = -2, avec une conservation du volume.

```
ROD.FIT>run
Iteration stopped afer converging.

Chi2 increase for error estimate (0 = use covariance matrix): [1.0]
Fit results:
              Scale =      3.408       <FIXED>
             Scale2 =          1       <FIXED>
               Beta =   -5.5e-05       <FIXED>
   Surface fraction =          1       <FIXED>
     Group expan  1 =          5 +/- 1.156
     Group expan  2 =         -2 +/- 0.9374

chisqr = 0.0000, normalized = 0.0000, Q = 1.00000
```

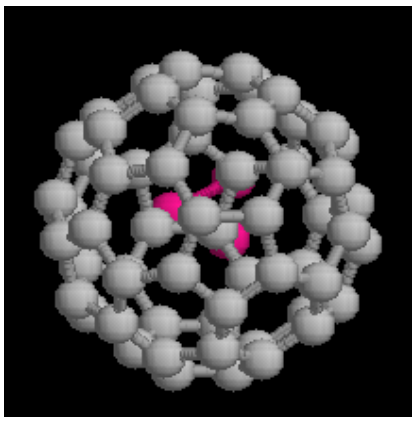


Fig.31 *Représentation du substrat et de la molécule de C60 déformée à la surface.*

En passant les deux paramètres de déformation en paramètres libres, on obtient directement le bon modèle.

```
Iteration stopped afer converging.

Chi2 increase for error estimate (0 = use covariance matrix): [1.0]
Fit results:
              Scale =      9.063       <FIXED>
             Scale2 =          1       <FIXED>
               Beta =   -5.5e-05       <FIXED>
   Surface fraction =          1       <FIXED>
     Group expan  1 =          5 +/- 0.6735
     Group expan  2 =         -2 +/- 0.5419
     Group shear  1 = -4.424e-09 +/- 0.008711
     Group shear  2 = -1.141e-08 +/- 0.003188
     Group shear  3 = -2.841e-08 +/- 0.00344

chisqr = 0.0000, normalized = 0.0000, Q = 1.00000
```

Dans le cas où j'entre plusieurs variables de déformation avec le même test et que je les passe en paramètres libres, alors on obtient quand même le bon modèle.

6.3.2 Cisaillement

Pour ce test, j'ai pris : $e_{xy} = 0.1$ et $e_{zx} = e_{yz} = 0.2$. Donc on a deux paramètres qui sont égaux, alors on va pouvoir prendre les deux « serial number » égaux.

```
ROD.FIT>run
Iteration stopped afer converging.

Chi2 increase for error estimate (0 = use covariance matrix): [1.0]
Fit results:
               Scale =      3.408       <FIXED>
              Scale2 =          1       <FIXED>
                Beta =   -5.5e-05       <FIXED>
    Surface fraction =          1       <FIXED>
      Group shear  1 =    0.04695 +/- 0.008171
      Group shear  2 =    -0.1176 +/- 0.004331

chisqr = 229.8978, normalized = 4.9978, Q = 0.00000
```

Dans ce cas si on passe les paramètres de déformation en paramètres libres, on a une mauvaise approximation : en optimisant, le programme ROD a du trouver un minimum local.

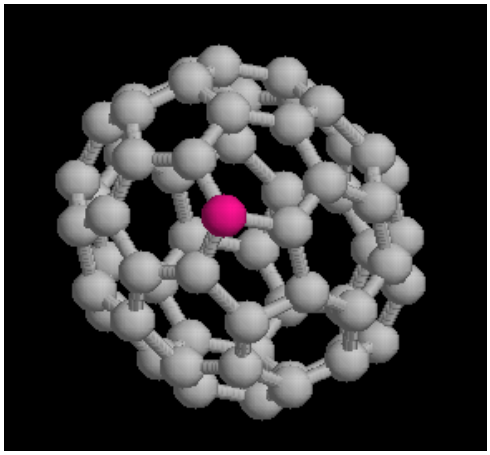


Fig.32 *Représentation de la molécule de C60 après la déformation.*

Par contre, si on entre un intervalle assez petit on s'approche de la solution (voir le test ci-après).

```
ROD.FIT>run
Iteration stopped afer converging.

Chi2 increase for error estimate (0 = use covariance matrix): [1.0]
Fit results:
               Scale =      3.408       <FIXED>
              Scale2 =          1       <FIXED>
                Beta =   -5.5e-05       <FIXED>
    Surface fraction =          1       <FIXED>
      Group shear  1 =    0.09508 +/- 0.01241       [         0 , 0.15      ]
      Group shear  2 =     0.1676 +/- 0.002406      [       0.1 , 0.25      ]

chisqr = 58.9805, normalized = 1.2822, Q = 0.06498
```

6.3.3 Dilatation/compression

Dans ce test, j'ai introduit un substrat, et pris : dx = 3, dy = -5 et dz = 2.

```
ROD.FIT>run
Iteration stopped afer converging.

Chi2 increase for error estimate (0 = use covariance matrix): [1.0]
Fit results:
              Scale =      3.408       <FIXED>
             Scale2 =          1       <FIXED>
               Beta =   -5.5e-05       <FIXED>
    Surface fraction =          1       <FIXED>
      In-plane DW  1 =          0       <FIXED>
      Group expan  1 =          3 +/- 1.231
      Group expan  2 =         -5 +/- 0.7052
      Group expan  3 =          2 +/- 10.61

chisqr = 0.0000, normalized = 0.0000, Q = 1.00000
```

Comme ci-dessus, en passant les paramètres de déformation en paramètres libres, on retrouve le bon modèle.

6.4 Test 3 : Simulation d'une molécule plane (thiol)

Dans cet exemple, j'ai travaillé avec un fichier (*.sur) représentant une molécule de thiol, et un substrat (voir annexe 11). Comme déformation, j'ai fait : dx = dy = 2 et $e_{xz} = 0.06$.

```
ROD.FIT>run
Iteration stopped afer converging.

Chi2 increase for error estimate (0 = use covariance matrix): [1.0]
Fit results:
              Scale =          1       <FIXED>
             Scale2 =          1       <FIXED>
               Beta =          0       <FIXED>
    Surface fraction =          1       <FIXED>
      Group expan  1 =      2.173 +/- 0.4068
      Group shear  1 =    0.05649 +/- 0.009974

chisqr = 0.0334, normalized = 0.0007, Q = 1.00000
```

En passant les paramètres de déformation en paramètres libres, on a une bonne approximation du modèle.

6.5 Test 4 : Comparaison avec des données expérimentales (C60 fulrène)

Ici, j'ai testé des données expérimentales, il s'agit de molécules de C60 déposées sous vide sur un substrat de GaAs, sans déformation on obtient « chisqr = 401.675 » et « normalized = 2.1830 » et si on introduit des déformations alors on a « chisqr = 392.1042 » et « normalized = 2.1310 ». On a donc une meilleure approximation du modèle.

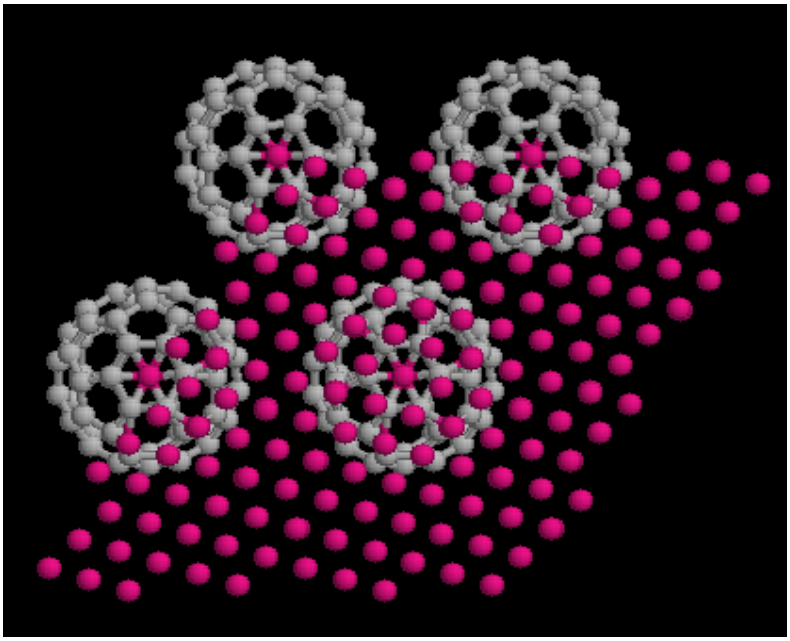


Fig.33 *Représentation des molécules de C60 (en gris) et du substrat (en rose).*

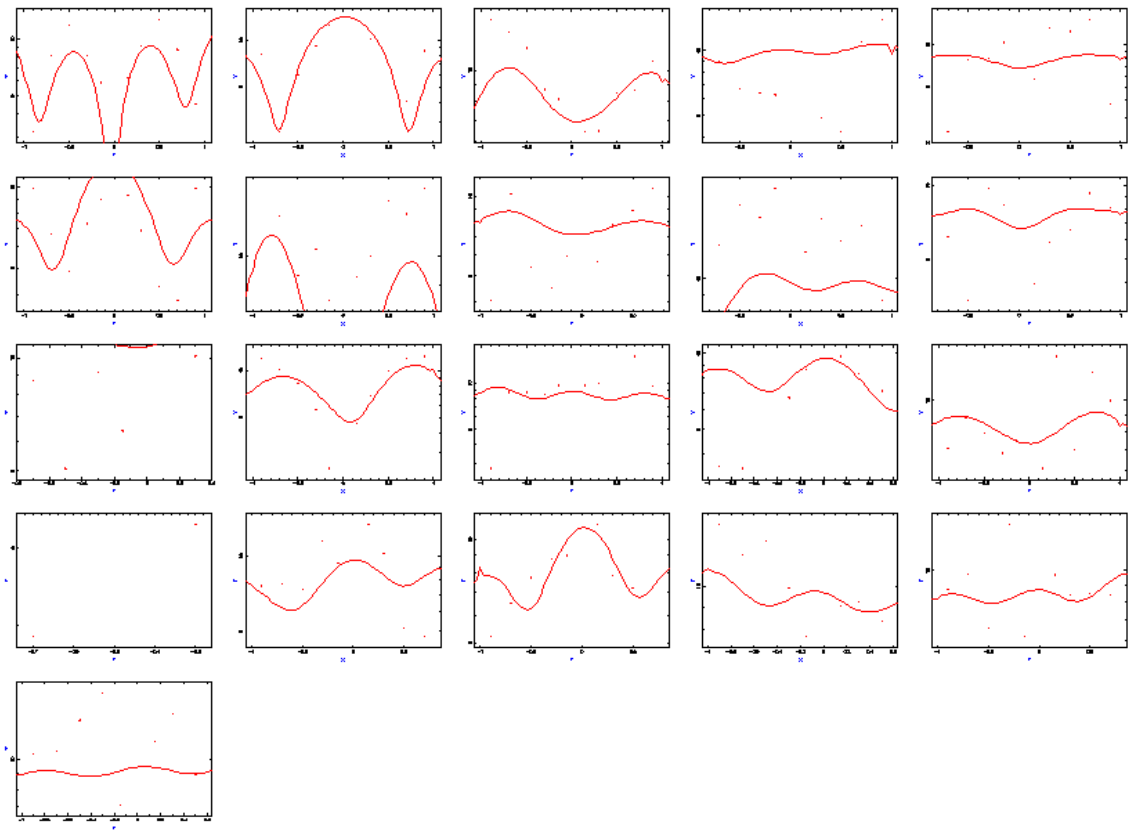


Fig.34 *Représentation de f_{data} et f_{calc} avant la déformation.*

```
      Group expan  1 =          0       <FIXED>
      Group expan  2 =          0       <FIXED>
      Group expan  3 =          0       <FIXED>
      Group shear  1 =          0       <FIXED>
      Group shear  2 =          0       <FIXED>
      Group shear  3 =          0       <FIXED>

chisqr = 401.6715, normalized = 2.1830, Q = 0.00000
```

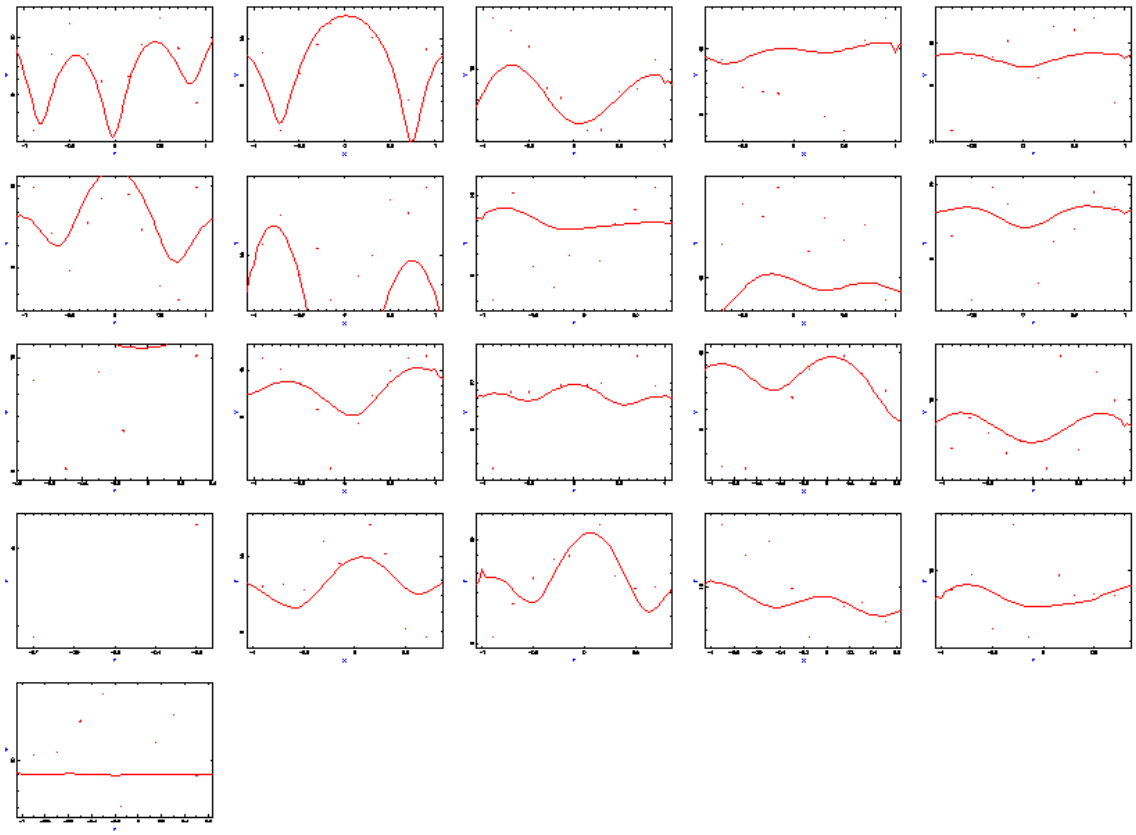


Fig.35 *Représentation de f_{data} et f_{calc} après la déformation.*

CONCLUSION

Le programme ROD permet l'analyse de données expérimentales de diffraction X de surface, le but des mesures étant de retrouver la structure cristallographique au voisinage de la surface d'un substrat monocristallin. Comme on a pu le voir ROD contient une partie de gestion du modèle atomique, une partie de calcul et une partie d'ajustement des paramètres.

Pour traiter le cas de molécules complexes absorbées à la surface du substrat, il a été introduit la possibilité de regrouper des atomes, et de tourner ou de translater un groupe d'atomes. Durant ce stage, j'ai introduit la possibilité d'avoir de petites déformations du groupe d'atomes, ceci afin de simuler les déformations des molécules induites par l'interaction molécules-substrat ou l'interaction entre molécules.

Les différents tests réalisés m'ont permis de vérifier le bon fonctionnement des déformations dans ROD, et de l'importance des déformations dans les différents calculs. D'autres tests expérimentaux seraient utiles pour pouvoir montrer l'intérêt des petites déformations des groupes d'atomes à la surface du cristal dans le programme ROD.

ANNEXES

Annexe 1 : Déclaration des variables et des fonctions dans « svensson.h »

```
/*************************  mikaël part  ****************************/


SET float XATOMINGROUPSFIT[MAXGROUPS][MAXATOMSPERGROUP];
/* fitted x-coordinates of atom */
SET float YATOMINGROUPSFIT[MAXGROUPS][MAXATOMSPERGROUP];
/* fitted y-coordinates of atom */
SET float ZATOMINGROUPSFIT[MAXGROUPS][MAXATOMSPERGROUP];
/* fitted z-coordinates of atom */
SET float ORIGINOFATOMSINGROUPSFIT[MAXGROUPS][3];
/* fitted coordinates of the origin of rotation */

SET int SNEXPX[MAXGROUPS];    /* serial number of x-expansion of group */
SET int SNEXPY[MAXGROUPS];    /* serial number of y-expansion of group */
SET int SNEXPZ[MAXGROUPS];    /* serial number of z-expansion of group */
SET int SNSHEARXY[MAXGROUPS]; /* serial number of xy-shearing of group */
SET int SNSHEARZX[MAXGROUPS]; /* serial number of zx-shearing of group */
SET int SNSHEARYZ[MAXGROUPS]; /* serial number of yz-shearing of group */

/****************** fonctions utilisées ***************************/
void mik(void);   /* annexe 2.1 */

void mik_deform_group(void);  /* annexe 2.2 */
void mik_update_model(void);  /* annexe 2.3 */

float center_atoms_group(int ngroup,int comp);  /* annexe 2.4 */
float calc_comp_th(int choice,int ngroup,float d1,float d2); /* annexe 2.5
*/
float max_dist(float center,int ngroup,int comp); /* annexe 2.6 */

/*************** fonction non utilisée***************/
void replace_originofatoms(int ngroup); /* annexe 2.7 */

SET int consvolexpan[MAXGROUPS];
SET int choicecenterexpan;
SET float exp_center[MAXGROUPS][3]; /* center of the expansion */

FIT int NEXPANGROUPSTOT;  /* total number of expansion */
FIT int NSHEARGROUPSTOT;  /* total number of shearing */
```

Ici, j'ai introduit une structure pour pouvoir ajouter d'autres déformations ultérieurement sans avoir trop de variables.

```
FIT struct
 {
  float defgroups[MAXPAR];
  float deflimgroups[MAXPAR][2];
  int defpengroups[MAXPAR];
  }
  func_deform[2];

/*
func_deform[0] = expansion
func_deform[1] = shearing
*/
```

Annexe 2 : Code source des fonctions utilisées dans « svensson.c »

Annexe 2.1 : La fonction mik()

Ici, on fait la saisie des valeurs des paramètres pour les deux déformations, le cisaillement (« shearing ») et la dilatation/compression (« expansion ») après avoir saisi les différents paramètres (« parameter ») correspondants.

```
/*********************************************************************/
void     mik(void)
/*********************************************************************/

/*
set values of the deformation fit parameters
*/

     /* mik's menu*/
{
  /* define mik menu */

static struct MENU mik_menu[]=

      {"parameter", 1, 1,"Model par. for fitting group(s)",
      "expansion",1, 2, "Value of group expansion parameter",
      "shearing", 1, 3, "Value of group shearing parameter",
      "lparam"  , 2, 18,"List parameter",
      "latoms"  , 2, 19,"List atoms",
      "help"    , 1, 20,"Help",
      "return"  , 1, 21,"Close menu/no deformation"
      };

 /* number of commands in mik  menu */

    int mik_length = sizeof(mik_menu)/sizeof(mik_menu[0]);

    int     stop = FALSE;
    char    token[100];
    int i,j,npar,ngroup;

    while (!stop)

      {
      if (!get_token(token,"ROD.EXT.SVE.SET.DEF>")) break;
      switch (cmnd_match(token,mik_menu,mik_length))
          {
          case 1:
            mik_deform_group();
            break;

          case 2:
            npar = get_int(1,
                "Serial number of group expansion parameter: ");
            if ((npar < 1) || (npar > NEXPANGROUPSTOT))
                {
                errtype("ERROR, no such serial number in model");
                break;
                }
            sprintf(STRING,
                "Value of expansion parameter %1d [%6.3f]: ",
                npar,func_deform[0].defgroups[npar-1]);
```

```
        func_deform[0].defgroups[npar-1] =
  get_real(func_deform[0].defgroups[npar-1],STRING);
  sprintf(STRING,
      "Lower limit of expansion %1d [%6.3f]: ",
      npar,func_deform[0].deflimgroups[npar-1][0]);
  func_deform[0].deflimgroups[npar-1][0] =
    get_real(func_deform[0].deflimgroups[npar-1][0],STRING);
  sprintf(STRING,
      "Upper limit of expansion %1d [%6.3f]: ",
      npar,func_deform[0].deflimgroups[npar-1][1]);
  func_deform[0].deflimgroups[npar-1][1] =
    get_real(func_deform[0].deflimgroups[npar-1][1],STRING);
  sprintf(STRING,
      "Range checking on expansion %1d [%s]: ",
      npar,yesnostr(func_deform[0].defpengroups[npar-1]));
  func_deform[0].defpengroups[npar-1] =
  yesno(func_deform[0].defpengroups[npar-1],STRING);
  break;

case 3:
  npar = get_int(1,
      "Serial number of group shearing parameter: ");
  if ((npar < 1) || (npar > NSHEARGROUPSTOT))
      {
      errtype("ERROR, no such serial number in model");
      break;
      }
  sprintf(STRING,
      "Value of shearing parameter %1d [%6.3f]: ",
      npar,func_deform[1].defgroups[npar-1]);
  func_deform[1].defgroups[npar-1] =
  get_real(func_deform[1].defgroups[npar-1],STRING);
  sprintf(STRING,
      "Lower limit of shearing %1d [%6.3f]: ",
      npar,func_deform[1].deflimgroups[npar-1][0]);
  func_deform[1].deflimgroups[npar-1][0] =
    get_real(func_deform[1].deflimgroups[npar-1][0],STRING);
  sprintf(STRING,
      "Upper limit of shearing %1d [%6.3f]: ",
      npar,func_deform[1].deflimgroups[npar-1][1]);
  func_deform[1].deflimgroups[npar-1][1] =
    get_real(func_deform[1].deflimgroups[npar-1][1],STRING);
  sprintf(STRING,
      "Range checking on shearing %1d [%s]: ",
      npar,yesnostr(func_deform[1].defpengroups[npar-1]));
  func_deform[1].defpengroups[npar-1] =
  yesno(func_deform[1].defpengroups[npar-1],STRING);
  break;

case 18:
  ngroup = get_ngroup("Group number : ");
  if (ngroup)
        {
        if ((NEXPANGROUPSTOT==0) &&  (NSHEARGROUPSTOT==0))
              {
              sprintf(STRING,"Not deformation !!\n");
              type_line(STRING);
              }
        else
        {
        for (i = 0; i < NEXPANGROUPSTOT; i++)
              {
```

```
                        sprintf(STRING,"%2d Group Expan = %8.4f [%8.4f ,
%8.4f]",i+1,func_deform[0].defgroups[i],func_deform[0].deflimgroups[i][0],f
unc_deform[0].deflimgroups[i][1]);
                        type_line(STRING);

                        if (func_deform[0].defpengroups[i]==1)
                              type_line("  cheked ");
                        else
                              type_line("         ");
                        type_list("\n",ROWS);
                        }

                  for (i = 0; i < NSHEARGROUPSTOT; i++)
                        {
                        sprintf(STRING,"%2d Group Shear = %8.4f [%8.4f ,
%8.4f]",i+1+NEXPANGROUPSTOT,func_deform[1].defgroups[i],func_deform[1].defl
imgroups[i][0],func_deform[1].deflimgroups[i][1]);
                        type_line(STRING);

                        if (func_deform[1].defpengroups[i]==1)
                              type_line("  cheked ");
                        else
                              type_line("         ");
                        type_list("\n",ROWS);
                        }
                  }
            }
            break;

          case 19:
            if (!NGROUPSS) type_line("No groups defined.\n");
            else
                {
                clear_screen();
                for (i = 0; i < NGROUPSS; i++)
                    {
                  if (NGROUPSS2 && (i == 0))
                      type_list("First surface cell:\n",ROWS);
                  if (NGROUPSS2 && (i == NGROUPSS-NGROUPSS2))
                      {
                      type_list("----------------------------------------
----------------------------------\n",ROWS);
                      type_list("Second surface cell:\n",ROWS);
                      }
                  sprintf(STRING,
                        "%2s %10s %7s %2s %7s %2s %7s %2s %7s %2s %7s %2s
%7s %2s\n",
                        "#","group name",
                        "x +","d",
                        "y +","d",
                        "z +","d",
                        "phi +","d",
                        "chi +","d",
                        "th +","d");
                  type_list(STRING,ROWS);
                  sprintf(STRING,
                        "%2d %10s %7.4f %2d %7.4f %2d %7.4f %2d %7.4f %2d
%7.4f %2d %7.4f %2d\n",
                        i+1, NAMEOFGROUPS[i],
                        XGROUPS[i], NXDISGROUPS[i],
                        YGROUPS[i], NYDISGROUPS[i],
                        ZGROUPS[i], NZDISGROUPS[i],
                        PHIGROUPS[i]*RAD, NPHIDISGROUPS[i],
```

```
                        CHIGROUPS[i]*RAD, NCHIDISGROUPS[i],
                        THGROUPS[i]*RAD, NTHDISGROUPS[i]);
                  type_list(STRING,ROWS);
                  sprintf(STRING, "Origin of atoms: %7.4f %7.4f %7.4f
Occupancy: %7.4f\n",

                        ORIGINOFATOMSINGROUPSFIT[i][0],
                        ORIGINOFATOMSINGROUPSFIT[i][1],
                        ORIGINOFATOMSINGROUPSFIT[i][2],
                        OCCUPANCYOFGROUPS[i]);
                  type_list(STRING,ROWS);
                  sprintf(STRING, "%2s %2s %7s %7s %7s %2s\n",
                        "#","el", "x +", "y +", "z +", "B");
                  type_list(STRING,ROWS);
                  for ( j = 0; j < NATOMSINGROUPS[i]; j++)
                      {
                      sprintf(STRING, "%2d %2s %7.4f %7.4f %7.4f %2d\n",
                            j+1, ELEMENT[TATOMINGROUPS[i][j]],
                            XATOMINGROUPSFIT[i][j],
                            YATOMINGROUPSFIT[i][j],
                            ZATOMINGROUPSFIT[i][j],
                            NDWSATOMINGROUPS[i][j]);
                      type_list(STRING,ROWS);
                      }
                  }
               }
            break;

         case 20:
            list_menu("MIK MENU",mik_menu,mik_length);
            break;
         case 21:
            stop=TRUE;
         }
      }
update_model();
}
```

Annexe 2.2 : La fonction mik_deform_group()

Saisie des différents paramètres de la dilatation/compression (centre de la dilatation/compression, conservation de volume) et saisie des « serial number » pour les déformations.

```
/*************************************************************************
*/
      void mik_deform_group(void)
/*************************************************************************
*/
{
/*
Set fit parameters of group deformation
*/

    /* define mik_deform_group_menu */

    static struct   MENU mik_deform_group_menu[] =
      {
      "centerexp",  2,  1,  "Center of expansion",
      "conservol",  2,  11, "Conserving of the volume for the expan",
      "xexpan",     2,  2,  "Serial number of x-expansion parameter",
      "yexpan",     2,  3,  "Serial number of y-expansion parameter",
      "zexpan",     2,  4,  "Serial number of z-expansion parameter",
      "xyshear",    2,  5,  "Serial number of xy-shear first parameter",
      "zxshear",    2,  6,  "Serial number of xz-shear second parameter",
      "yzshear",    2,  7,  "Serial number of yz-shear first parameter",
      "help",       1,  31, "Display menu",
      "return",     1,  32, "Return to main menu"
      };

    int mik_deform_group_length = sizeof(mik_deform_group_menu) /
sizeof(mik_deform_group_menu[0]);

    int     stop = FALSE;
    char    token[30];
    int     ngroup;

    while (!stop)
      {
      if (!get_token(token,"ROD.EXT.SVE.SET.DEF.PAR>")) break;
      switch
(cmnd_match(token,mik_deform_group_menu,mik_deform_group_length))
          {
          case -1:
            break;
          case 0:
            break;
          case 1:
            ngroup = get_ngroup("Group number: ");
            if (ngroup)
                {
                sprintf(STRING,"The expansion's center : \n\n");
                type_list(STRING,ROWS);
                sprintf(STRING,"\t 1/ Center of the atoms group.\n");
                type_list(STRING,ROWS);
                sprintf(STRING,"\t 2/ Origin of rotation.\n");
                type_list(STRING,ROWS);
                sprintf(STRING,"\t 3/ Enter the center.\n\n");
                type_list(STRING,ROWS);
```

```
                choicecenterexpan++;
                sprintf(STRING,"Enter your choice [%d] :
",choicecenterexpan);
                choicecenterexpan=get_int(choicecenterexpan,STRING)-1;

            if (choicecenterexpan==0)
                  {
                  exp_center[ngroup-1][0]=center_atoms_group(ngroup-1,0);
                  exp_center[ngroup-1][1]=center_atoms_group(ngroup-1,1);
                  exp_center[ngroup-1][2]=center_atoms_group(ngroup-1,2);
                  }
            else if (choicecenterexpan==1)
                  {
                  exp_center[ngroup-1][0]=ORIGINOFATOMSINGROUPS[ngroup-
1][0];
                  exp_center[ngroup-1][1]=ORIGINOFATOMSINGROUPS[ngroup-
1][1];
                  exp_center[ngroup-1][2]=ORIGINOFATOMSINGROUPS[ngroup-
1][2];
                  }
            else if (choicecenterexpan==2)
                  {
                  sprintf(STRING,"Enter the x center of the expansion
                  [%5.3f]: ",exp_center[ngroup-1][0]);
                  exp_center[ngroup-1][0]=get_real(exp_center[ngroup-
1][0],STRING);
                  sprintf(STRING,"Enter the y center of the expansion
                  [%5.3f]: ",exp_center[ngroup-1][1]);
                  exp_center[ngroup-1][1]=get_real(exp_center[ngroup-
1][1],STRING);
                  sprintf(STRING,"Enter the z center of the expansion
                  [%5.3f]: ",exp_center[ngroup-1][2]);
                  exp_center[ngroup-1][2]=get_real(exp_center[ngroup-
1][2],STRING);
                  }
            else return;

                }
            break;

          case 11:
              ngroup = get_ngroup("Group number: ");
            if (ngroup)
                {
                  sprintf(STRING,"Conserving of the volume ? [%s]:
                  ",yesnostr(consvolexpan[ngroup-1]));
                consvolexpan[ngroup-1]=yesno(consvolexpan[ngroup-
1],STRING);

                if (consvolexpan[ngroup-1]==1)
                  {
                  if ((SNEXPX[ngroup-1]>0)&&(SNEXPY[ngroup-
1]>0)&&(SNEXPZ[ngroup-1]>0))
                        {
                        sprintf(STRING,"Error, impossible to conserve the
volume.\n");
                        type_list(STRING,ROWS);
                        consvolexpan[ngroup-1]=0;
                        }
                  }
                }
            break;
```

```
case 2:
  ngroup = get_ngroup("Group number: ");
  if (ngroup)
      {
      sprintf(STRING,
            "Serial number of x-expansion parameter [%1d]: ",
            SNEXPX[ngroup-1]);
      SNEXPX[ngroup-1] = get_int(SNEXPX[ngroup-1],STRING);
      if (SNEXPX[ngroup-1] > NEXPANGROUPSTOT)
        NEXPANGROUPSTOT = SNEXPX[ngroup-1];
      }
  break;

case 3:
  ngroup = get_ngroup("Group number: ");
  if (ngroup)
      {
      sprintf(STRING,
            "Serial number of y-expansion parameter [%1d]: ",
            SNEXPY[ngroup-1]);
      SNEXPY[ngroup-1] = get_int(SNEXPY[ngroup-1],STRING);
      if (SNEXPY[ngroup-1] > NEXPANGROUPSTOT)
        NEXPANGROUPSTOT = SNEXPY[ngroup-1];
      }
  break;

case 4:
  ngroup = get_ngroup("Group number: ");
  if (ngroup)
      {
      sprintf(STRING,
            "Serial number of z-expansion parameter [%1d]: ",
            SNEXPZ[ngroup-1]);
      SNEXPZ[ngroup-1] = get_int(SNEXPZ[ngroup-1],STRING);
      if (SNEXPZ[ngroup-1] > NEXPANGROUPSTOT)
        NEXPANGROUPSTOT = SNEXPZ[ngroup-1];
      }
  break;

case 5:
  ngroup = get_ngroup("Group number: ");
  if (ngroup)
      {
      sprintf(STRING,
            "Serial number xy-shear parameter [%1d]: ",
            SNSHEARXY[ngroup-1]);
      SNSHEARXY[ngroup-1] = get_int(SNSHEARXY[ngroup-1],STRING);
      if (SNSHEARXY[ngroup-1] > NSHEARGROUPSTOT)
        NSHEARGROUPSTOT = SNSHEARXY[ngroup-1];
      }
  break;

case 6:
    ngroup = get_ngroup("Group number: ");
  if (ngroup)
      {
      sprintf(STRING,
            "Serial number zx-shear parameter [%1d]: ",
            SNSHEARZX[ngroup-1]);
      SNSHEARZX[ngroup-1] = get_int(SNSHEARZX[ngroup-1],STRING);
      if (SNSHEARZX[ngroup-1] > NSHEARGROUPSTOT)
        NSHEARGROUPSTOT = SNSHEARZX[ngroup-1];
      }
```

```
            break;

          case 7:
            ngroup = get_ngroup("Group number: ");
            if (ngroup)
                {
                sprintf(STRING,
                      "Serial number y-shear parameter [%1d]: ",
                      SNSHEARYZ[ngroup-1]);
                SNSHEARYZ[ngroup-1] = get_int(SNSHEARYZ[ngroup-1],STRING);
                if (SNSHEARYZ[ngroup-1] > NSHEARGROUPSTOT)
                  NSHEARGROUPSTOT = SNSHEARYZ[ngroup-1];
                }
            break;

          case 31:
            list_menu("DEFORMATION PARAMETERS",
mik_deform_group_menu,mik_deform_group_length);
            break;

          case 32:
            stop = TRUE;
          }
      }
    update_model();
    }
```

Annexe 2.3 : La fonction mik_update_model()

La fonction suivante permet de faire une mise à jour de toutes les coordonnées des atomes dans le repère orthogonal ainsi que les coordonnées du centre de rotation du groupe d'atomes dans le repère orthogonal, après avoir entré les différents paramètres de la déformation.

```
/*************************************************************************
*/
      void mik_update_model(void)
/*************************************************************************
*/

{
int i,j;
float exp_temp;

for (i=0 ; i<NGROUPSS ; i++)
      {
      ORIGINOFATOMSINGROUPSFIT[i][0]=ORIGINOFATOMSINGROUPS[i][0];
      ORIGINOFATOMSINGROUPSFIT[i][1]=ORIGINOFATOMSINGROUPS[i][1];
      ORIGINOFATOMSINGROUPSFIT[i][2]=ORIGINOFATOMSINGROUPS[i][2];

      for (j=0 ; j<NATOMSINGROUPS[i] ; j++)
            {
            XATOMINGROUPSFIT[i][j]=XATOMINGROUPS[i][j];
            YATOMINGROUPSFIT[i][j]=YATOMINGROUPS[i][j];
            ZATOMINGROUPSFIT[i][j]=ZATOMINGROUPS[i][j];
            }

      if ((SNEXPX[i]>0)&&(SNEXPY[i]==0)&&(SNEXPZ[i]==0))
            {
            for (j=0 ; j<NATOMSINGROUPS[i] ; j++)
                  {
                  XATOMINGROUPSFIT[i][j]+=
                  func_deform[0].defgroups[SNEXPX[i]-
                  1]/100*(XATOMINGROUPS[i][j]-exp_center[i][0]);
                  }
            ORIGINOFATOMSINGROUPSFIT[i][0]+=
            func_deform[0].defgroups[SNEXPX[i]-
            1]/100*(ORIGINOFATOMSINGROUPS[i][0]-exp_center[i][0]);

            }

      if ((SNEXPX[i]==0)&&(SNEXPY[i]>0)&&(SNEXPZ[i]==0))
            {
            for (j=0 ; j<NATOMSINGROUPS[i] ; j++)
                  {
                  YATOMINGROUPSFIT[i][j]+=
                  func_deform[0].defgroups[SNEXPY[i]-
                  1]/100*(YATOMINGROUPS[i][j]-exp_center[i][1]);
                  }
            ORIGINOFATOMSINGROUPSFIT[i][1]+=
            func_deform[0].defgroups[SNEXPY[i]-
            1]/100*(ORIGINOFATOMSINGROUPS[i][1]-exp_center[i][1]);

            }

      if ((SNEXPX[i]==0)&&(SNEXPY[i]==0)&&(SNEXPZ[i]>0))
            {
            for (j=0 ; j<NATOMSINGROUPS[i] ; j++)
```

```
                {
                ZATOMINGROUPSFIT[i][j]+=
                func_deform[0].defgroups[SNEXPZ[i]-
                1]/100*(ZATOMINGROUPS[i][j]-exp_center[i][2]);
                }
          ORIGINOFATOMSINGROUPSFIT[i][2]+=
          func_deform[0].defgroups[SNEXPZ[i]-
          1]/100*(ORIGINOFATOMSINGROUPS[i][2]-exp_center[i][2]);

          }

     if ((SNEXPX[i]>0)&&(SNEXPY[i]>0)&&(SNEXPZ[i]>0))
          {
          for (j=0 ; j<NATOMSINGROUPS[i] ; j++)
                {
                XATOMINGROUPSFIT[i][j]+=
                func_deform[0].defgroups[SNEXPX[i]-
                1]/100*(XATOMINGROUPS[i][j]-exp_center[i][0]);
                YATOMINGROUPSFIT[i][j]+=
                func_deform[0].defgroups[SNEXPY[i]-
                1]/100*(YATOMINGROUPS[i][j]-exp_center[i][1]);
                ZATOMINGROUPSFIT[i][j]+=
                func_deform[0].defgroups[SNEXPZ[i]-
                1]/100*(ZATOMINGROUPS[i][j]-exp_center[i][2]);
                }
          ORIGINOFATOMSINGROUPSFIT[i][0]+=
          func_deform[0].defgroups[SNEXPX[i]-
          1]/100*(ORIGINOFATOMSINGROUPS[i][0]-exp_center[i][0]);
          ORIGINOFATOMSINGROUPSFIT[i][1]+=
          func_deform[0].defgroups[SNEXPY[i]-
          1]/100*(ORIGINOFATOMSINGROUPS[i][1]-exp_center[i][1]);
          ORIGINOFATOMSINGROUPSFIT[i][2]+=
          func_deform[0].defgroups[SNEXPZ[i]-
          1]/100*(ORIGINOFATOMSINGROUPS[i][2]-exp_center[i][2]);

          }

     if
((SNEXPX[i]>0)&&(SNEXPY[i]>0)&&(SNEXPZ[i]==0)&&(consvolexpan[i]==0))
          {
          for (j=0 ; j<NATOMSINGROUPS[i] ; j++)
                {
                YATOMINGROUPSFIT[i][j]+=
                func_deform[0].defgroups[SNEXPY[i]-
                1]/100*(YATOMINGROUPS[i][j]-exp_center[i][1]);
                XATOMINGROUPSFIT[i][j]+=
                func_deform[0].defgroups[SNEXPX[i]-
                1]/100*(XATOMINGROUPS[i][j]-exp_center[i][0]);
                }
          ORIGINOFATOMSINGROUPSFIT[i][0]+=
          func_deform[0].defgroups[SNEXPX[i]-
          1]/100*(ORIGINOFATOMSINGROUPS[i][0]-exp_center[i][0]);
          ORIGINOFATOMSINGROUPSFIT[i][1]+=
          func_deform[0].defgroups[SNEXPY[i]-
          1]/100*(ORIGINOFATOMSINGROUPS[i][1]-exp_center[i][1]);

          }

     if
((SNEXPX[i]==0)&&(SNEXPY[i]>0)&&(SNEXPZ[i]>0)&&(consvolexpan[i]==0))
          {
          for (j=0 ; j<NATOMSINGROUPS[i] ; j++)
                {
```

```
                  YATOMINGROUPSFIT[i][j]+=
                  func_deform[0].defgroups[SNEXPY[i]-
                  1]/100*(YATOMINGROUPS[i][j]-exp_center[i][1]);
                  ZATOMINGROUPSFIT[i][j]+=
                  func_deform[0].defgroups[SNEXPZ[i]-
                  1]/100*(ZATOMINGROUPS[i][j]-exp_center[i][2]);
                  }
            ORIGINOFATOMSINGROUPSFIT[i][2]+=
            func_deform[0].defgroups[SNEXPZ[i]-
            1]/100*(ORIGINOFATOMSINGROUPS[i][2]-exp_center[i][2]);
            ORIGINOFATOMSINGROUPSFIT[i][1]+=
            func_deform[0].defgroups[SNEXPY[i]-
            1]/100*(ORIGINOFATOMSINGROUPS[i][1]-exp_center[i][1]);

            }

      if
((SNEXPX[i]>0)&&(SNEXPY[i]==0)&&(SNEXPZ[i]>0)&&(consvolexpan[i]==0))
            {
            for (j=0 ; j<NATOMSINGROUPS[i] ; j++)
                  {
                  ZATOMINGROUPSFIT[i][j]+=
                  func_deform[0].defgroups[SNEXPZ[i]-
                  1]/100*(ZATOMINGROUPS[i][j]-exp_center[i][2]);
                  XATOMINGROUPSFIT[i][j]+=
                  func_deform[0].defgroups[SNEXPX[i]-
                  1]/100*(XATOMINGROUPS[i][j]-exp_center[i][0]);
            }
            ORIGINOFATOMSINGROUPSFIT[i][0]+=
            func_deform[0].defgroups[SNEXPX[i]-
            1]/100*(ORIGINOFATOMSINGROUPS[i][0]-exp_center[i][0]);
            ORIGINOFATOMSINGROUPSFIT[i][2]+=
            func_deform[0].defgroups[SNEXPZ[i]-
            1]/100*(ORIGINOFATOMSINGROUPS[i][2]-exp_center[i][2]);

            }

      if
((SNEXPX[i]>0)&&(SNEXPY[i]>0)&&(SNEXPZ[i]==0)&&(consvolexpan[i]==1))
            {
            exp_temp=calc_comp_th(1,i,func_deform[0].defgroups[SNEXPX[i]-
            1],func_deform[0].defgroups[SNEXPY[i]-1]);

            for (j=0 ; j<NATOMSINGROUPS[i] ; j++)
                  {
                  YATOMINGROUPSFIT[i][j]+=
                  func_deform[0].defgroups[SNEXPY[i]-
                  1]/100*(YATOMINGROUPS[i][j]-exp_center[i][1]);
                  XATOMINGROUPSFIT[i][j]+=
                  func_deform[0].defgroups[SNEXPX[i]-
                  1]/100*(XATOMINGROUPS[i][j]-exp_center[i][0]);
                  ZATOMINGROUPSFIT[i][j]+=exp_temp/100*(ZATOMINGROUPS[i][j]
                  -exp_center[i][2]);
                  }

            ORIGINOFATOMSINGROUPSFIT[i][0]+=
            func_deform[0].defgroups[SNEXPX[i]-
            1]/100*(ORIGINOFATOMSINGROUPS[i][0]-exp_center[i][0]);
            ORIGINOFATOMSINGROUPSFIT[i][1]+=
            func_deform[0].defgroups[SNEXPY[i]-
            1]/100*(ORIGINOFATOMSINGROUPS[i][1]-exp_center[i][1]);
            ORIGINOFATOMSINGROUPSFIT[i][2]+=exp_temp*(ORIGINOFATOMSINGROUPS
            [i][2]-exp_center[i][2]);
```

```
            }

      if
((SNEXPX[i]==0)&&(SNEXPY[i]>0)&&(SNEXPZ[i]>0)&&(consvolexpan[i]==1))
            {
            exp_temp=calc_comp_th(2,i,func_deform[0].defgroups[SNEXPZ[i]-
            1],func_deform[0].defgroups[SNEXPY[i]-1]);

            for (j=0 ; j<NATOMSINGROUPS[i] ; j++)
                  {
                  YATOMINGROUPSFIT[i][j]+=
                  func_deform[0].defgroups[SNEXPY[i]-
                  1]/100*(YATOMINGROUPS[i][j]-exp_center[i][1]);
                  ZATOMINGROUPSFIT[i][j]+=
                  func_deform[0].defgroups[SNEXPZ[i]-
                  1]/100*(ZATOMINGROUPS[i][j]-exp_center[i][2]);
                  XATOMINGROUPSFIT[i][j]+=exp_temp/100*(XATOMINGROUPS[i][j]
                  -exp_center[i][0]);
                  }

            ORIGINOFATOMSINGROUPSFIT[i][2]+=
            func_deform[0].defgroups[SNEXPZ[i]-
            1]/100*(ORIGINOFATOMSINGROUPS[i][2]-exp_center[i][2]);
            ORIGINOFATOMSINGROUPSFIT[i][1]+=
            func_deform[0].defgroups[SNEXPY[i]-
            1]/100*(ORIGINOFATOMSINGROUPS[i][1]-exp_center[i][1]);
            ORIGINOFATOMSINGROUPSFIT[i][0]+=exp_temp*(ORIGINOFATOMSINGROUPS
            [i][0]-exp_center[i][0]);

            }

      if
((SNEXPX[i]>0)&&(SNEXPY[i]==0)&&(SNEXPZ[i]>0)&&(consvolexpan[i]==1))
            {
            exp_temp=calc_comp_th(3,i,func_deform[0].defgroups[SNEXPX[i]-
            1],func_deform[0].defgroups[SNEXPZ[i]-1]);

            for (j=0 ; j<NATOMSINGROUPS[i] ; j++)
                  {
                  ZATOMINGROUPSFIT[i][j]+=
                  func_deform[0].defgroups[SNEXPZ[i]-
                  1]/100*(ZATOMINGROUPS[i][j]-exp_center[i][2]);
                  XATOMINGROUPSFIT[i][j]+=
                  func_deform[0].defgroups[SNEXPX[i]-
                  1]/100*(XATOMINGROUPS[i][j]-exp_center[i][0]);
                  YATOMINGROUPSFIT[i][j]+=exp_temp/100*(YATOMINGROUPS[i][j]
                  -exp_center[i][1]);
                  }

            ORIGINOFATOMSINGROUPSFIT[i][0]+=
            func_deform[0].defgroups[SNEXPX[i]-
            1]/100*(ORIGINOFATOMSINGROUPS[i][0]-exp_center[i][0]);
            ORIGINOFATOMSINGROUPSFIT[i][2]+=
            func_deform[0].defgroups[SNEXPZ[i]-
            1]/100*(ORIGINOFATOMSINGROUPS[i][2]-exp_center[i][2]);
            ORIGINOFATOMSINGROUPSFIT[i][1]+=exp_temp*(ORIGINOFATOMSINGROUPS
            [i][1]-exp_center[i][1]);

            }

      if (SNSHEARXY[i]>0)
      {
```

```
            for (j=0 ; j<NATOMSINGROUPS[i] ; j++)
                  {
                  XATOMINGROUPSFIT[i][j]+=
                  0.5*func_deform[1].defgroups[SNSHEARXY[i]-
                  1]*YATOMINGROUPS[i][j];
                  YATOMINGROUPSFIT[i][j]+=
                  0.5*func_deform[1].defgroups[SNSHEARXY[i]-
                  1]*XATOMINGROUPS[i][j];
                  }
            ORIGINOFATOMSINGROUPSFIT[i][0]+=
            0.5*func_deform[1].defgroups[SNSHEARXY[i]-
            1]*ORIGINOFATOMSINGROUPS[i][1];
            ORIGINOFATOMSINGROUPSFIT[i][1]+=
            0.5*func_deform[1].defgroups[SNSHEARXY[i]-
            1]*ORIGINOFATOMSINGROUPS[i][0];

      }

      if (SNSHEARZX[i]>0)
      {
            for (j=0 ; j<NATOMSINGROUPS[i] ; j++)
                  {
                  XATOMINGROUPSFIT[i][j]+=
                  0.5*func_deform[1].defgroups[SNSHEARZX[i]-
                  1]*ZATOMINGROUPS[i][j];
                  ZATOMINGROUPSFIT[i][j]+=
                  0.5*func_deform[1].defgroups[SNSHEARZX[i]-
                  1]*XATOMINGROUPS[i][j];
                  }
            ORIGINOFATOMSINGROUPSFIT[i][0]+=
            0.5*func_deform[1].defgroups[SNSHEARZX[i]-
            1]*ORIGINOFATOMSINGROUPS[i][2];
            ORIGINOFATOMSINGROUPSFIT[i][2]+=
            0.5*func_deform[1].defgroups[SNSHEARZX[i]-
            1]*ORIGINOFATOMSINGROUPS[i][0];

      }

      if (SNSHEARYZ[i]>0)
      {
            for (j=0 ; j<NATOMSINGROUPS[i] ; j++)
                  {
                  YATOMINGROUPSFIT[i][j]+=
                  0.5*func_deform[1].defgroups[SNSHEARYZ[i]-
                  1]*ZATOMINGROUPS[i][j];
                  ZATOMINGROUPSFIT[i][j]+=
                  0.5*func_deform[1].defgroups[SNSHEARYZ[i]-
                  1]*YATOMINGROUPS[i][j];
                  }
            ORIGINOFATOMSINGROUPSFIT[i][2]+=
            0.5*func_deform[1].defgroups[SNSHEARYZ[i]-
            1]*ORIGINOFATOMSINGROUPS[i][1];
            ORIGINOFATOMSINGROUPSFIT[i][1]+=
            0.5*func_deform[1].defgroups[SNSHEARYZ[i]-
            1]*ORIGINOFATOMSINGROUPS[i][2];
      }
}
}
```

Annexe 2.4 : La fonction center_atoms_group()

La fonction ci-dessous calcul les coordonnées du centre géométrique du goupe d'atome (« ngroup »).

```
/**************************************************************************
*/
     float    center_atoms_group(int ngroup, int comp)
/**************************************************************************
*/

{
/*
Compute the center of the atoms
*/

int j;
float sum;

sum = 0;

if (comp==0)
      {
      for (j=0 ; j<NATOMSINGROUPS[ngroup] ; j++)
            {
            sum +=XATOMINGROUPS[ngroup][j];
            }
      }
if (comp==1)
      {
      for (j=0 ; j<NATOMSINGROUPS[ngroup] ; j++)
            {
            sum +=YATOMINGROUPS[ngroup][j];
            }
      }
if (comp==2)
      {
      for (j=0 ; j<NATOMSINGROUPS[ngroup] ; j++)
            {
            sum +=ZATOMINGROUPS[ngroup][j];
            }
      }

return (sum/NATOMSINGROUPS[ngroup]);
}
```

Annexe 2.5 : La fonction calc_comp_th()

La fonction qui suit retourne la valeur (en pourcentage) de la quantité de déformation (dilatation/compression) suivant un axe, lorsque l'on a conservation du volume et que l'on a saisi deux quantités de déformation dans les deux autres directions (sinon impossible).

```
/*************************************************************************
*/
     float     calc_comp_th(int choice, int ngroup, float d1, float d2)
/*************************************************************************
*/

{

float Oxmax,Oymax,Ozmax,Oxmax2,Oymax2,Ozmax2,xcenter,ycenter,zcenter,comp;

      xcenter=center_atoms_group(ngroup,0);
      ycenter=center_atoms_group(ngroup,1);
      zcenter=center_atoms_group(ngroup,2);

      Oxmax=max_dist(xcenter,ngroup,0);
      Oymax=max_dist(ycenter,ngroup,1);
      Ozmax=max_dist(zcenter,ngroup,2);

if (choice==1)
      {

      Oxmax2=(d1/100*((Oxmax+xcenter)-exp_center[ngroup][0]))+Oxmax;
      Oymax2=(d2/100*((Oymax+ycenter)-exp_center[ngroup][1]))+Oymax;
      Ozmax2=(Oxmax*Oymax*Ozmax)/(Oxmax2*Oymax2);

      comp=(Ozmax2-Ozmax)*100/Ozmax;
      }
if (choice==2)
      {

      Ozmax2=(d1/100*((Ozmax+zcenter)-exp_center[ngroup][2]))+Ozmax;
      Oymax2=(d2/100*((Oymax+ycenter)-exp_center[ngroup][1]))+Oymax;
      Oxmax2=(Oxmax*Oymax*Ozmax)/(Ozmax2*Oymax2);

      comp=(Oxmax2-Oxmax)*100/Oxmax;
      }
if (choice==3)
      {

      Oxmax2=(d1/100*((Oxmax+xcenter)-exp_center[ngroup][0]))+Oxmax;
      Ozmax2=(d2/100*((Ozmax+zcenter)-exp_center[ngroup][2]))+Ozmax;
      Oymax2=(Oxmax*Oymax*Ozmax)/(Oxmax2*Ozmax2);

      comp=(Oymax2-Oymax)*100/Oymax;
      }

return comp;

}
```

Annexe 2.6 : La fonction max_dist()

Cette fonction calcule et renvoie la distance maximum dans un groupe (« ngroup ») entre le centre (« center ») des atomes et les autres atomes suivant un axe (x, y ou z, respectivement pour « comp » égal à 0, 1 ou 2). Cette fonction est appelée dans « calc_comp_th ( ) » (voir annexe 2.5).

```
/*************************************************************************
*/     float max_dist(float center, int ngroup, int comp)
/*************************************************************************
*/
{
/*
compute the maximum distance between the center and the
atoms of the group on the axis x, y or z
*/
float maxdist,tmp;
int j;
maxdist=0;

if (comp==0)
      {
      for (j=0 ; j<NATOMSINGROUPS[ngroup] ; j++)
            {
            tmp=fabs(XATOMINGROUPS[ngroup][j]-center);

            if (tmp > maxdist)
                  {
                  maxdist=tmp;
                  }
            }
      }
if (comp==1)
      {
      for (j=0 ; j<NATOMSINGROUPS[ngroup] ; j++)
            {
            tmp=fabs(YATOMINGROUPS[ngroup][j]-center);

            if (tmp > maxdist)
                  {
                  maxdist=tmp;
                  }
            }
      }
if (comp==2)
      {
      for (j=0 ; j<NATOMSINGROUPS[ngroup] ; j++)
            {
            tmp=fabs(ZATOMINGROUPS[ngroup][j]-center);

            if (tmp > maxdist)
                  {
                  maxdist=tmp;
                  }
            }
      }
return maxdist;
}
```

Annexe 2.7 : La fonction replace_originofatoms()

La fonction qui suit n'a pas été utilisée dans le programme, celle-ci après une déformation va replacer le centre de la rotation (« ORIGINOFATOMSINGROUPS ») à sa place d'origine en faisant une translation du centre ainsi que de tous les autres atomes du groupe (« ngroup »).

```
/*************************************************************************
*/
     void     replace_originofatoms(int ngroup)
/*************************************************************************
*/

{

int j;
float xtranslate,ztranslate,ytranslate;

xtranslate=ORIGINOFATOMSINGROUPS[ngroup][0]-
ORIGINOFATOMSINGROUPSFIT[ngroup][0];
ytranslate=ORIGINOFATOMSINGROUPS[ngroup][1]-
ORIGINOFATOMSINGROUPSFIT[ngroup][1];
ztranslate=ORIGINOFATOMSINGROUPS[ngroup][2]-
ORIGINOFATOMSINGROUPSFIT[ngroup][2];

      for (j=0 ; j<NATOMSINGROUPS[ngroup] ; j++)
                  {
                  XATOMINGROUPSFIT[ngroup][j]+=xtranslate;
                  YATOMINGROUPSFIT[ngroup][j]+=ytranslate;
                  ZATOMINGROUPSFIT[ngroup][j]+=ztranslate;
                  }

ORIGINOFATOMSINGROUPSFIT[ngroup][0]=ORIGINOFATOMSINGROUPS[ngroup][0];
ORIGINOFATOMSINGROUPSFIT[ngroup][1]=ORIGINOFATOMSINGROUPS[ngroup][1];
ORIGINOFATOMSINGROUPSFIT[ngroup][2]=ORIGINOFATOMSINGROUPS[ngroup][2];

}
```

Annexe 3 : Fichiers d'entrée utilisés dans les tests

Annexe 3.1 : Simulation d'un cube

Fichier (*.sur) représentant le cube d'arête 2 centré en 0 et non déformé.

```
not dilated
 1.0000 1.0000 1.0000  90.0  90.0  90.0
el      XS          YS          ZS debw_in deb_out  occtot
group ex1  0.0000  0.0000  0.0000  1.0000
0.0000 0.0000 0.0000 0.0000 0.0000 0.0000
 C    -1.00000    -1.00000    -1.00000   0 0.00
 C    -1.00000    -1.00000     1.00000   0 0.00
 C    -1.00000     1.00000     1.00000   0 0.00
 C    -1.00000     1.00000    -1.00000   0 0.00
 C     1.00000     1.00000    -1.00000   0 0.00
 C     1.00000     1.00000     1.00000   0 0.00
 C     1.00000    -1.00000     1.00000   0 0.00
 C     1.00000    -1.00000    -1.00000   0 0.00
Endgroup
```

Fichier (*.sur) représentant le cube dilaté et comprimé avec une conservation du volume.

```
dilated
 1.0000 1.0000 1.0000  90.0  90.0  90.0
el      XS          YS          ZS debw_in deb_out  occtot
group ex1  0.0000  0.0000  0.0000  1.0000
0.0000 0.0000 0.0000 0.0000 0.0000 0.0000
 C    -0.66666    -1.50000    -1.00000   0 0.00
 C    -0.66666    -1.50000     1.00000   0 0.00
 C    -0.66666     1.50000     1.00000   0 0.00
 C    -0.66666     1.50000    -1.00000   0 0.00
 C     0.66666     1.50000    -1.00000   0 0.00
 C     0.66666     1.50000     1.00000   0 0.00
 C     0.66666    -1.50000     1.00000   0 0.00
 C     0.66666    -1.50000    -1.00000   0 0.00
endgroup
```

Fichier (*.sur) représentant le cube cisaillé.

```
sheared
 1.0000 1.0000 1.0000  90.0  90.0  90.0
el      XS          YS          ZS debw_in deb_out  occtot
group ex1  0.0000  0.0000  0.0000  1.0000
0.0000 0.0000 0.0000 0.0000 0.0000 0.0000
 C    -0.95000    -1.10000    -1.25000   0 0.00
 C    -0.85000    -0.70000     0.75000   0 0.00
 C    -1.05000     1.30000     1.15000   0 0.00
 C    -1.15000     0.90000    -0.85000   0 0.00
 C     0.85000     0.70000    -0.75000   0 0.00
 C     0.95000     1.10000     1.25000   0 0.00
 C     1.15000    -0.90000     0.85000   0 0.00
 C     1.05000    -1.30000    -1.15000   0 0.00
endgroup
```

Annexe 3.2 : Simulation d'une molécule sphérique (C60 fulrène)

Fichier (*.sur) définissant la molécule de c60 à la surface du cristal.

```
c60 with not expansion and not shearing
 2.7744 2.7744 6.7959  90.0  90.0 120.0
Pt     0.00000     0.00000     0.00000   0 0.00 0.00
group c60  0.0000  0.0000  0.0000  1.0000
0.0000 0.0000 1.0000 0.0000 0.0000 0.0000
 C     0.00000     0.00000     0.00000   0 0.00
 C    -0.63662    -1.20386     0.27886   0 0.00
 C    -0.66558     1.25829     0.29145   0 0.00
 C     1.41322     0.15991     0.29688   0 0.00
 C     0.33653     2.19590     0.76833   0 0.00
 C     1.62124     1.51712     0.77162   0 0.00
 C    -1.93907     1.25833     0.84894   0 0.00
 C     2.12873    -0.89078     0.85990   0 0.00
 C    -1.96754    -1.20383     0.86159   0 0.00
 C     0.11120    -2.30188     0.86723   0 0.00
 C    -2.60447     0.00000     1.14027   0 0.00
 C     1.46334    -2.14884     1.15146   0 0.00
 C     0.02185     3.09317     1.78233   0 0.00
 C     2.53599     1.76469     1.78867   0 0.00
 C    -2.04210    -2.30194     1.81004   0 0.00
 C    -0.75737    -2.98048     1.81368   0 0.00
 C    -2.26807     2.19608     1.90881   0 0.00
 C     3.08488    -0.63202     1.92288   0 0.00
 C    -1.30917     3.09320     2.36521   0 0.00
 C     3.28409     0.66653     2.37692   0 0.00
 C    -3.34479     0.16005     2.38011   0 0.00
 C     2.00796    -2.66733     2.39484   0 0.00
 C     0.97789     3.35201     2.84526   0 0.00
 C     2.20717     2.70252     2.84834   0 0.00
 C    -3.13686     1.51732     2.85514   0 0.00
 C     3.01025    -1.72985     2.87165   0 0.00
 C    -2.75030    -2.14877     2.99636   0 0.00
 C    -0.23631    -3.47635     3.00341   0 0.00
 C    -3.41595    -0.89063     3.28762   0 0.00
 C     1.17688    -3.31639     3.30052   0 0.00
 C    -1.17561     3.35226     3.78848   0 0.00
 C     3.41785     0.92572     3.80019   0 0.00
 C     0.23777     3.51231     4.08508   0 0.00
 C     2.75226     2.18411     4.09136   0 0.00
 C    -3.00904     1.76517     4.21703   0 0.00
 C     3.13829    -1.48189     4.23353   0 0.00
 C    -2.20564    -2.66700     4.23970   0 0.00
 C    -0.97636    -3.31605     4.24326   0 0.00
 C    -2.00682     2.70297     4.69397   0 0.00
 C     3.34674    -0.12465     4.70811   0 0.00
 C    -3.28251    -0.63142     4.71110   0 0.00
 C     1.31044    -3.05708     4.72397   0 0.00
 C    -3.08347     0.66716     5.16547   0 0.00
 C     2.26953    -2.16015     5.18026   0 0.00
 C     0.75922     3.01624     5.27435   0 0.00
 C     2.04418     2.33763     5.27763   0 0.00
 C    -2.53441    -1.72916     5.29944   0 0.00
 C    -0.02047    -3.05682     5.30657   0 0.00
 C    -1.46187     2.18457     5.93696   0 0.00
 C     2.60668     0.03601     5.94811   0 0.00
 C    -0.10948     2.33781     6.22075   0 0.00
 C     1.96981     1.23992     6.22642   0 0.00
 C    -2.12742     0.92649     6.22844   0 0.00
 C     1.94093    -1.22203     6.23993   0 0.00
 C    -1.61970    -1.48093     6.31660   0 0.00
 C    -0.33494    -2.15941     6.32043   0 0.00
 C    -1.41159    -0.12377     6.79134   0 0.00
 C     0.66736    -1.22167     6.79730   0 0.00
 C     0.63869     1.24016     6.80920   0 0.00
 C     0.00184     0.03653     7.08824   0 0.00
endgroup
```

Fichier (*.bul) définissant les atomes du substrat.

```
#Pt (111)
2.7744 2.7744 6.7959 90 90 120
Pt   0.000000   0.000000   0.000000   1   0
Pt   0.333333   0.666667  -0.333333   1   0
Pt   0.666667   0.333333  -0.666667   1   0
```

Annexe 3.3 : Simulation d'une molécule plane (thiol)

Fichier (*.sur) définissant la molécule de thiol à la surface du cristal.

```
c18 thiol on GaAs(001):As
 5.6500 5.6500 5.6500  90.0  90.0  90.0
As    0     0     0
As    0.5   0.5   0
Ga    0.25  0.25  -0.25
Ga    0.75  0.75  -0.25
group thiol  0.0000  0.0000  1.5000
0.0000 0.0000 0.3500 -90.0000 57.0000 45.0000
 S     0.00000     0.00000     1.50000   0 0.00
 C     0.80000     0.00000     2.80000   0 0.00
 C     0.00000     0.00000     4.10000   0 0.00
 C     0.80000     0.00000     5.40000   0 0.00
 C     0.00000     0.00000     6.70000   0 0.00
 C     0.80000     0.00000     8.00000   0 0.00
 C     0.00000     0.00000     9.30000   0 0.00
 C     0.80000     0.00000    10.60000   0 0.00
 C     0.00000     0.00000    11.90000   0 0.00
 C     0.80000     0.00000    13.20000   0 0.00
 C     0.00000     0.00000    14.50000   0 0.00
 C     0.80000     0.00000    15.80000   0 0.00
 C     0.00000     0.00000    17.10000   0 0.00
 C     0.80000     0.00000    18.40000   0 0.00
 C     0.00000     0.00000    19.70000   0 0.00
 C     0.80000     0.00000    21.00000   0 0.00
 C     0.00000     0.00000    22.30000   0 0.00
 C     0.80000     0.00000    23.60000   0 0.00
 C     0.00000     0.00000    24.90000   0 0.00
Endgroup
```

Fichier (*.bul) définissant les atomes du substrat.

```
GaAs(001) c(2x2) unit cell (=bulk cell); As terminated
5.65  5.65  5.65  90    90    90
As    0.5   0     -0.5
As    0     0.5   -0.5
Ga    0.75  0.25  -0.75
Ga    0.25  0.75  -0.75
As    0     0     -1.00
As    0.5   0.5   -1.00
Ga    0.25  0.25  -1.25
Ga    0.75  0.75  -1.25
```

BIBLIOGRAPHIE

Livres

[1] **Les rayons X -** André Guinier - Edition Que sais-je ?.1984. Chapitre 7.

[2] **Theory of elasticity** - S. P. Timoshenko, J. N. Goodier - Third Edition.1970. Chapitres 1 & 2.

[3] **A treatise on the mathematical theory of elasticity** - A. E. H. Love - Dover Edition. 1944. Chapitre 1.

[4] **Fundamentals of crystallography** – C. Giacovazzo, H. L. Monaco, D. Viterbo, F. Scordari, G. Gilli, G. Zanotti, M. Catti – Edited by C. Giacovazzo. 1992.

[5] **Elasticité et anélasticité des métaux** – C. Zener – DUNOD. 1955. Chapitre 1.

[6] **Cristallographie géométrique et radiocristalographie** – J-J. Rousseau – DUNOD. 1999.

[7] **Elasticité linéaire** – D. Dartus – Edité par Cépaduès. 1995. Chapitres 1 & 2.

Liens internet

[8] http://www.esrf.fr/AboutUs/Documentation/

[9] http://www.esrf.fr/computing/scientific/joint_projects/ANA-ROD/index.html

[10] http://www.chez.com/deuns/sciences/drx/drx.html

[11] http://www.chez.com/deuns/sciences/cristallo/cristallo23.html

[12] http://v.favrenicolin.free.fr/recherche/these/manuscript/4-Chapitre1.pdf

[13] http://www.cem2.univ-montp2.fr/cours/ProjetsIUP2/A4/progHTML/Debut1.html

[14] http://esm2.imt-mrs.fr/gar/gdhtml/gd.html

[15] http://www.sciences.ch/htmlfr/mecanalytique/mecanamecfluides.php#solides

Communications privées avec M. Francesco Borgatti, mail : borgatti@tasc.infm.it.