Cálculo de la estructura electrónica por un método multinivel

Tesis
entregada a la
Universidad de Chile
en cumplimiento parcial de los requisitos
para optar al grado de
Magíster en Ciencias Físicas
Facultad de Ciencias

por

Xavier Iago Andrade Valencia

Diciembre, 2004

Director de Tesis: **Dr. José Rogan**
Co-Director de Tesis: **Dr. Aldo H. Romero**

FACULTAD DE CIENCIAS

UNIVERSIDAD DE CHILE

INFORME DE APROBACIÓN

TESIS DE MAGÍSTER

Se informa a la Escuela de Postgrado de la Facultad de Ciencias que la Tesis de Magíster presentada por el candidato

Xavier Iago Andrade Valencia

ha sido aprobada por la Comisión de Evaluación de la Tesis como requisito para optar al grado de Magíster en Ciencias Físicas, en el examen de Defensa de Tesis rendido el día 15 de Diciembre de 2004.

Director de Tesis

Dr. José Rogan

Co-Director de Tesis

Dr. Aldo H. Romero

Comisión de Evaluación de la Tesis

Dr. Patricio Fuentealba

Dr. Gonzalo Gutierrez

Dr. Miguel Kiwi

AGRADECIMIENTOS

Se agradece a FONDECYT por el financiamiento para realizar este trabajo, a través del proyecto No. 1030957.

Índice

RESUMEN

Se desarrolló un código para cálculos *ab-initio* en clusters y moléculas basado en la teoría del funcional de la densidad. Se usó una formulación en el espacio real para discretizar el problema y el método de *Rayleigh Quotient Multigrid* para encontrar la solución del problema de autovalores que aparece dentro de la iteración de autoconsistencia de las ecuaciones de Kohn-Sham.

En este trabajo se describe el proceso completo de solución, incluyendo la discretización del laplaciano usando un método de diferencias finitas de alto orden y el de algoritmo de diagonalización, basado, este último, en hacer una minimización multinivel del cuociente de Rayleigh.

El código fue probado en sistemas de pocos átomos, considerando todos los electrones. Los valores de energía obtenidos fueron comparados por los dados por otros programas y también con resultados de alta precisión que están disponibles en la literatura, obteniendo variaciones del orden de un 1 % para valores de energía total, y resultados aún más cercanos para diferencias de energía.

ABSTRACT

An *ab-initio* code is developed, based upon the density functional theory, for calculations of clusters and molecules. A real space formulation is used to map the problem onto a discrete grid. The Rayleigh Quotient Multigrid method is used to solve the eigenproblem associated to the the self consistent iteration of the Kohn-Sham equations.

I describe in this work the complete solution process, including the discretization of the Laplacian operator, on the basis of a high order finite difference formula and

a diagonalization algorithm which consists in performing a multilevel minimization of the Rayleigh quotient.

The code was tested on systems of a few atoms, including all the electrons in the calculations. The resulting energy values were compared with results obtained with other codes and other high accuracy results that are available in the literature. Differences of less than 1 % were obtained for the total energies and even better results for the energy differences.

Capítulo 1

Introducción

Un sólido o una molécula es un sistema compuesto por iones y electrones agrupados de manera que se forman enlaces entre ellos. Las propiedades que caracterizan al sistema en cuestión dependen de las diferentes interacciones. Si se hace una aproximación no relativista al problema, bastaría con resolver cuánticamente el sistema descrito por las interacciones ion-ion, ion-electrón y electrón-electrón, que son básicamente las responsables de todas sus propiedades físicas.

El primer grado de aproximación a este problema es la denominada aproximación de Born-Oppenheimer [1], que básicamente hace uso de que la masa electrónica es más de 2000 veces más pequeña que la masa del ion, lo que ayuda a separar las ecuaciones usando dos escalas de movimiento: una cuántica para los electrones y una clásica para los iones. Por lo tanto, desde este punto de vista, los electrones se encuentran en presencia de un campo producido por cargas fijas, los iones, mientras que los iones sienten la presencia de los electrones y al cambiar dinámicamente producen cambios instantáneos en su distribución electrónica.

Bajo esta aproximación es que la física de la materia condensada y la química cuántica se han desarrollado en los últimos 70 años, por una parte resolviendo la estructura electrónica, es decir, en el marco de la mecánica cuántica, encontrando la

función de onda que rige el comportamiento de los electrones y por otra suponiendo que las fuerzas que sienten los iones vienen de la estructura electrónica del sistema (lo que se relaciona a través de las fuerzas de Hellmann-Feynman [2] [3]).

Por lo tanto, los recursos de cálculo se han centrado en tratar de resolver las ecuaciones que rigen el comportamiento de los electrones. Entre las diferentes formas de tratar el problema, hay dos que han atraído la mayor atención. El primer método consiste en hacer uso de las ecuaciones de Hartree-Fock, incluyendo los fenómenos de correlación, mediante teoría de perturbaciones.

Como segunda opción está la denominada teoría del funcional de la densidad, desarrollada por el Profesor Walter Kohn, por la cual le fue otorgado el premio Nobel de Química en 1998. Esta teoría demuestra que la energía del sistema electrónico está determinada de manera unívoca por un funcional de la densidad electrónica, por lo que bajo el principio variacional se podría encontrar exactamente la densidad electrónica y la energía del estado fundamental. La forma funcional no es conocida y tienen que hacerse varias aproximaciones para poder obtener resultados físicos. La mayoría de estas aproximaciones parten de un desarrollo de la teoría que mapea el comportamiento de los electrones a unas nuevas partículas que están descritas por ecuaciones de un solo cuerpo, denominadas ecuaciones de Kohn-Sham. Aunque las funciones de onda de estas nuevas partículas no representan los electrones físicamente, la densidad electrónica construida a partir de las soluciones de las ecuaciones de Kohn-Sham es exactamente la densidad electrónica del problema original.

La manera como usualmente se resuelven las ecuaciones de Kohn-Sham es con métodos espectrales, donde la función de onda de estas nuevas partículas se representa como una expansión en términos de funciones base y lo que se busca son los coeficientes de la expansión.

Una de las bases más usadas, en especial por los físicos, es la base de ondas planas; en ella se usa la Transformada Rápida de Fourier (FFT) [13] para pasar el problema al espacio recíproco, donde el operador laplaciano es diagonal y la implementación es sencilla. Si bien es cierto que esta base es muy eficiente para tratar sistemas periódicos, es poco adecuada para sistemas finitos, ya que es necesario usar condiciones de borde periódicas, y porque el espacio vacío es ineficiente de representar en el espacio de Fourier. Hay muchos códigos desarrollados en este esquema, como *CPMD*, *AbInit*, *Wien2K* [41], etc.

En sistemas finitos, que tradicionalmente han sido de interés en la química, la base preferida es la formada por funciones gaussianas, que son muy eficientes porque muchos términos integrales pueden calcularse analíticamente. Sin embargo, al tratar sistemas muy grandes o sistemas periódicos se vuelven ineficientes porque las funciones de onda no son localizadas. Programas que usan este tipo de bases son *NWChem* y *Gaussian* [40].

En 1994 Chelikoswky, Troullier y Saad [14] presentaron un método de solución en el que las funciones de onda se representan directamente en el espacio real sobre una grilla. La parte principal de su trabajo fue el uso de diferencias finitas de alto orden para el cálculo del operador laplaciano, lo que permiten alcanzar la alta precisión necesaria para cálculos de estructura electrónica con grillas con relativamente pocos puntos.

Hay varias potenciales ventajas de la solución en el espacio real con respecto a otros métodos [17]. Por una parte no se tiene una base y es posible mejorar sistemáticamente la calidad de la solución disminuyendo la separación de la grilla, de forma análoga a la energía de corte en el método de ondas planas. Esto no sucede en otros métodos que usan bases localizadas, donde el manejo de la precisión de la base es

más complejo y no siempre se hace más preciso el resultado al aumentar la calidad de la base.

Aunque directamente en el espacio real el tiempo de cálculo para N electrones es $\mathcal{O}(N^3)$ al igual que con ondas planas, es posible implementar un algoritmo de funciones de onda localizadas con escalamiento lineal ($\mathcal{O}(N)$). Es posible usar cualquier tipo de condiciones de borde, tanto para sistemas periódicos o como para sistemas finitos y además la representación del espacio vacío es directa. Esto lo hace un método eficiente tanto como para sistemas finitos cono para sistemas periódicos, lo que es especialmente adecuado para tratar estructuras que son periódicas en una dirección y finitas en otra, como por ejemplo nanoestructuras.

La paralelización de códigos en el espacio real es simple y eficiente, ya que es posible descomponer el dominio de solución entre los procesadores, y solo la información sobre los bordes de la grilla en cada proceso es la que debe ser compartida. Esto implica que es posible llegar a tratar sistemas más grandes que con otros métodos.

Finalmente, en el espacio real existen métodos muy eficientes de solución de ecuaciones diferenciales de autovalores como Lanczos precondicionado [24] o *Multigrid* [27]. Este último fue desarrollado como un método con escalamiento lineal para resolver ecuaciones diferenciales elípticas, pero ha sido adaptado a problemas de autovalores como los que aparecen en cálculos de estructura electrónica, tanto como precondicionador [18], o para resolver directamente [19] [32] [33]. El método de *multigrid* se basa en el uso de una serie de grillas de diferente densidad de puntos, en los cuales se trata adaptativamente el proceso de solución, resolviendo eficientemente la escala adecuada a cada nivel.

Una aplicación del método de *multigrid* ha sido el cálculo del potencial de intercambio y correlación para funcionales dependientes de los orbitales. Estos funcionales

dan una forma exacta para la energía de intercambio y correlación en términos de los orbitales de Kohn-Sham, pero el potencial de intercambio y correlación debe ser calculado a partir de una ecuación integral, conocida como ecuación del potencial efectivo optimizado (OEP). Hasta ahora solo se habían obtenido soluciones basadas en aproximaciones de esta ecuación o soluciones exactas para casos simples, como sistemas con simetría esférica. Sin embargo, utilizando un formalismo en el espacio real combinado con *multigrid* se ha podido resolver de forma completa la ecuación OEP [6].

En este trabajo se desarrolló un código de solución de las ecuaciones de Kohn-Sham en el espacio real, basado en el esquema de iteración de autoconsistencia y usando el método de *Rayleigh Quotient Multigrid* para diagonalizar el problema de autovalores asociado.

La estructura de este trabajo es la siguiente: primero se introduce la teoría del funcional de la densidad, la cual provee el marco teórico para los cálculos de estructura electrónica. A continuación, se describe el método de solución de las ecuaciones de Kohn-Sham, incluyendo la iteración de autoconsistencia, y cómo se trata cada uno de los pasos dentro de ésta. Posteriormente se detalla como se realizó la implementación del código en términos de programación. Finalmente se analizan algunos resultados entregados por el código: en primer lugar se resuelve un problema de Schrödinger estándar, para probar el método de solución de autovalores; luego se calcula la densidad y la energía para átomos aislados y moléculas pequeñas, comparando con resultados de otros códigos.

Capítulo 2

Teoría del funcional de la densidad

En 1964 Hohenberg y Kohn [4] demostraron que la energía total de un sistema de electrones es un funcional de la densidad electrónica, que denominaremos ρ. A partir de esto, la energía del sistema se puede escribir entonces como [1]

$$E_{HK}[\rho] = F[\rho] + \int d^3r\, \rho(\vec{r})\, V_0(\vec{r}), \tag{2.1}$$

donde V_0 es el potencial externo y $F[\rho]$ un funcional universal que no depende del potencial externo.

Este funcional universal puede ser escrito como la suma del funcional asociado a la energía cinética T, la repulsión clásica electrón-electrón (o energía de Hartree) y la energía de intercambio y correlación E_{xc}:

$$F[\rho] = T[\rho] + \frac{1}{2}\int d^3r \int d^3r' \frac{\rho(\vec{r})\,\rho(\vec{r}')}{|\vec{r}-\vec{r}'|} + E_{xc}[\rho]. \tag{2.2}$$

La energía de intercambio está asociada a la repulsión entre electrones del mismo *spin* y al principio de exclusión de Pauli. La energía de correlación es la parte cuántica

[1] En este trabajo las ecuaciones se usaran unidades atómicas para las ecuaciones, en estas las unidades fundamentales son: la constante de Planck dividida por 2π $\hbar = h/2\pi$, la masa en reposo del electrón m_e, la carga del electrón e y 4π veces la permisividad del espacio vacio $4\pi\epsilon_0$. El resto de las unidades se derivan de estas: la energía es medida en Hartree que equivale a 1 [Ha] = 27.2113961 [eV] y la unidad de distancia es el radio de Bohr (o Bohr) $a_0 = 0.529177249$ [Å].

de la interacción coulombiana entre electrones.

De acuerdo al Teorema de Hohenberg y Kohn el valor mínimo del funcional de energía corresponde a la energía del estado fundamental, y la densidad que lo produce es la densidad real exacta, de manera que podemos encontrar la densidad minimizando el funcional de energía sujeto a la condición que el número de electrones N se conserve, lo que equivale a que

$$\int d^3r\, \rho(\vec{r}) = N. \tag{2.3}$$

Esto nos da la ecuación de Euler

$$\frac{\delta}{\delta\rho(\vec{r})}\left[E_{HK}[\rho] - \mu \int d^3r\, \rho(\vec{r})\right] = 0 \tag{2.4}$$

donde $\delta/\delta\rho$ es la derivada funcional con respecto a la densidad y μ es el multiplicador de Lagrange asociado a la constricción (2.3).

El problema con el teorema de Hohenberg y Kohn es que los funcionales de energía cinética, y de intercambio y correlación, son desconocidos, por lo que deben ser aproximados. Para la parte cinética la aproximación más básica es utilizar la energía cinética de un gas de electrones, la que puede ser calculada analíticamente pero que no produce buenos resultados físicos.

2.1. Ecuaciones de Kohn-Sham

Kohn y Sham [5] tuvieron la idea de aproximar la parte cinética por la energía cinética de electrones no interactuantes, para lo cual es necesario pasar de resolver la ecuación de Euler (2.4), a resolver una ecuación de Schrödinger para N electrones independientes sujetos a un cierto potencial efectivo que depende de la densidad.

Esto da lugar a las ecuaciones de Kohn-Sham

$$\left\{-\frac{1}{2}\nabla^2 + V_{ef}\left[\rho\right](r)\right\}\phi_i = \varepsilon_i\phi_i \tag{2.5}$$

a partir de cuyas autofunciones ϕ_i se calcula la densidad electrónica como

$$\rho = 2\sum_{i=0}^{N/2}\left|\phi_i\right|^2 . \tag{2.6}$$

Estas funciones de onda ϕ_i no tienen significado físico directo.

El potencial efectivo V_{ef} es la suma del potencial nuclear V_0, el potencial de Hartree

$$V_H\left[\rho\right](\vec{r}) = \int d^3r' \frac{\rho(\vec{r}')}{|\vec{r}-\vec{r}'|} , \tag{2.7}$$

y el potencial de intercambio y correlación V_{xc}, que es la derivada funcional de la energía de intercambio y correlación con respecto a la densidad

$$V_{xc}\left[\rho\right](\vec{r}) = \frac{\delta E_{xc}\left[\rho\right]}{\delta\rho(\vec{r})} . \tag{2.8}$$

Si consideramos la diferencia entre la energía cinética real y la de electrones no interactuantes dentro de la energía de intercambio y correlación, las ecuaciones de Kohn-Sham son en principio exactas, pero como se desconoce el funcional de intercambio y correlación es necesario usar aproximaciones.

2.2. Energías

Una vez encontrada la solución de la ecuacion (2.5) a partir de sus autovalores y la densidad podemos calcular los valores de energía para el sistema:

- La energía cinética:

$$T = 2\sum_{i=0}^{N-1} \varepsilon_i - \int V_{ef}\left[\rho\left(\vec{r}\right)\right]\rho\left(\vec{r}\right)\, dr \tag{2.9}$$

- La energía de Hartree:

$$E_H = \frac{1}{2}\int\int \frac{\rho\left(r\right)\rho\left(r'\right)}{|r - r'|}\, dr\, dr' \tag{2.10}$$

- Si tenemos p iones, de numero atómico z_i y posición $\vec{r}_i$, la energía de interacción ion electrón será:

$$E_{e-ion} = \int \sum_{i=0}^{p-1} \frac{z_i \rho\left(r\right)}{|\vec{r} - \vec{r}_i|}\, dr \tag{2.11}$$

- La energía de intercambio y correlación:

$$E_{xc} = E_{xc}\left[\rho\left(\vec{r}\right)\right] \tag{2.12}$$

- La energía de interacción entre los iones:

$$E_{ion-ion} = \sum_{i=0}^{p-1}\sum_{j=0}^{i-1} \frac{q_i q_j}{|\vec{r}_i - \vec{r}_j|} \tag{2.13}$$

- Y finalmente la energía total del sistema:

$$E = T + E_{xc} + E_h + E_{e-i} + E_{i-i} \tag{2.14}$$

2.3. Funcional de intercambio y correlación

La aproximación más usada para el funcional de intercambio y correlación se conoce como *Local Density Approximation* (LDA) [5]; la idea es suponer que

$$V_{xc}[\rho] = V_{xc}(\rho),$$

es decir, el potencial es sólo una función local de la densidad.

En este caso el funcional de energía de intercambio y correlación asociado es

$$E_{xc}[\rho] = \int d^3r\, \rho(\vec{r})\, \epsilon_{xc}(\vec{r}) \tag{2.15}$$

donde ϵ_{xc} es la densidad de energía de intercambio y correlación por electrón, de un gas homogéneo de electrones.

Existen otras aproximaciones que se obtienen para el funcional de intercambio y correlación si se consideran además términos que dependen de los gradientes de la densidad y de derivadas mayores; estas aproximaciones se conocen como *General Gradient Approximation* (GGA).

2.3.1. Densidades de *Spin*

Para incluir explícitamente el *spin* dentro de la teoría funcional de la densidad, se utilizan dos densidades: ρ^α y ρ^β, una para cada orientación de *spin* , y que cumplen

$$N^\alpha = \int \rho^\alpha(\vec{r})\, dr, \quad N^\alpha = \int \rho^\alpha(\vec{r})\, dr \quad \text{y} \quad N^\alpha + N^\beta = N\,. \tag{2.16}$$

donde N^α y N^β son los números de electrones para cada orientación.

En este caso existe una ecuación de Kohn-Sham para cada densidad de *spin*

$$\left\{-\nabla^2 + V_{ef}^{\alpha}\left[\rho^{\alpha}, \rho^{\beta}\right](\vec{r})\right\}\phi_i^{\alpha} = \varepsilon_i^{\alpha}\phi_i^{\alpha} \tag{2.17}$$

$$\left\{-\nabla^2 + V_{ef}^{\beta}\left[\rho^{\alpha}, \rho^{\beta}\right](\vec{r})\right\}\phi_i^{\beta} = \varepsilon_i^{\beta}\phi_i^{\beta}\ . \tag{2.18}$$

Las densidades se calculan como

$$\rho^{\alpha}(\vec{r}) = \sum_i \phi_i^{\alpha}(\vec{r})\ , \quad \rho^{\beta}(\vec{r}) = \sum_i \phi_i^{\beta}(\vec{r}) \quad \text{y} \quad \rho(\vec{r}) = \rho^{\alpha}(\vec{r}) + \rho^{\beta}(\vec{r}) \tag{2.19}$$

y los potenciales efectivos son

$$V_{ef}^{\alpha} = V(\vec{r}) + \int \frac{\rho(\vec{r}')}{|\vec{r} - \vec{r}'|}dr' + \frac{\delta E_{xc}\left[\rho^{\alpha}, \rho^{\beta}\right]}{\delta\rho^{\alpha}(\vec{r})} \tag{2.20}$$

$$V_{ef}^{\beta} = V(\vec{r}) + \int \frac{\rho(\vec{r}')}{|\vec{r} - \vec{r}'|}dr' + \frac{\delta E_{xc}\left[\rho^{\alpha}, \rho^{\beta}\right]}{\delta\rho^{\beta}(r)} \tag{2.21}$$

2.3.2. LDA

El funcional de intercambio y correlación que utilizamos en nuestro código corresponde a LDA en la forma dada por Vosko, Wilk, and Nusair [11], que detallamos a continuación.

Primero definimos dos magnitudes, el parámetro del gas de electrones, que es el radio de una esfera cuyo volumen es el volumen efectivo de un electron,

$$r_s = \left(\frac{3}{4\pi\rho}\right)^{1/3}\ , \tag{2.22}$$

y la polarización de *spin*

$$\zeta = (\rho^{\alpha} - \rho^{\beta})/\rho\ . \tag{2.23}$$

Término de intercambio

La densidad de energía de intercambio de un gas de electrones se puede calcular analíticamente y esta dada por

$$\epsilon_x(r_s,\zeta) = \epsilon_x^P(r_s) + \left[\epsilon_x^F(r_s) - \epsilon_x^P(r_s)\right] f(\zeta) \tag{2.24}$$

donde los superindices ϵ_x^P es la densidad de intercambio paramagnéticas y ϵ_x^F es la densidad ferromagnética, estas se calculan como

$$\epsilon_x^P(r_s) = 2^{-1/3}\epsilon_x^F(r_s) = -3\left(\frac{9}{32\pi^2}\right)^{1/3} r_s^{-1}\,. \tag{2.25}$$

y la función f es

$$f(\zeta) = \frac{(1+\zeta)^{4/3} + (1-\zeta)^{4/3} - 2}{2\left(2^{1/3}-1\right)} \tag{2.26}$$

Término de correlación

En cambio, el término de correlación no se conoce en forma analítica, de manera que se utilizan los valores para un gas de electrones obtenidos numéricamente mediante *Quantum Montecarlo* por Ceperley y Adler [9] y que fueron parametrizados por Vosko, Wilk y Nusair para el uso en cálculos numéricos.

Definimos la función

$$\begin{aligned} F(x;A,x_0,b,c) \;=\; & A\left\{\ln\frac{x^2}{X(x)} + \frac{2b}{Q}\tan^{-1}\frac{Q}{2x+b} + \right. \\ & \left. -\frac{bx_0}{X(x_0)}\left[\ln\frac{(x-x_0)^2}{X(x)} + \frac{2(b+2x_0)}{Q}\tan^{-1}\frac{Q}{2x+b}\right]\right\} \end{aligned}$$

con

$$\begin{aligned} X(x) &= x^2 + bx + c \\ Q &= (4c - b2)^{1/2} . \end{aligned}$$

A partir de F definimos tres funciones, ϵ_c^P, ϵ_c^F y α_c con los parámetros dados en la tabla 2.1.

	A	x_0	b	c
ϵ_c^P	0.0310907	-0.10498	3.72744	12.9352
ϵ_c^F	0.01554535	-0.32500	7.06042	18.0578
α_c	-1/(62)	-0.00475840	1.13107	13.0045

Tabla 2.1: Parámetros para definir las funciones ϵ_c^P, ϵ_c^F y α_c a partir de F

La densidad de energía de correlación está dada por

$$\epsilon_c(r_s, \zeta) = \epsilon_c^P(r_s) + \Delta\epsilon_c(r_s, \zeta) \tag{2.27}$$

con

$$\Delta\epsilon_c(r_s, \zeta) = \alpha_c(r_s)\frac{f(\zeta)}{f''(0)}\left[1 + \beta(r_s)\zeta^4\right] \tag{2.28}$$

usando la función f definida en 2.26. La función $\beta(r_s)$ está definida como

$$\beta(r_s) = \frac{f''(0)\Delta\epsilon_c(r_s, 1)}{\alpha_c(r_s)} - 1 \tag{2.29}$$

con

$$\Delta\epsilon_c(r_s, 1) = \epsilon_c(r_s, 1) - \epsilon_c(r_s, 0) = \epsilon_c^F(r_s) - \epsilon_c^P(r_s) . \tag{2.30}$$

Densidad de intercambio y correlación

Finalmente, la densidad total de energía de intercambio y correlación de LDA es la suma de los dos términos

$$\epsilon_{xc}(r_s,\zeta) = \epsilon_x(r_s,\zeta) + \epsilon_c(r_s,\zeta) \ . \tag{2.31}$$

La energía total de intercambio y correlación se obtiene de la integral (2.15).

Capítulo 3

Métodos numéricos

3.1. Solución numérica de las ecuaciones de Kohn-Sham

En esta sección se detallan los pasos generales llevados a cabo para resolver las ecuaciones de Kohn-Sham. Todas las operaciones se expresan en términos de funciones y operaciones continuas. Más adelante se detalla como se realizan estas operaciones en un espacio discreto.

La ecuación (2.5) no corresponde a un problema usual de autovalores, ya que el operador depende no en forma no lineal de las autofunciones. Sin embargo, es posible resolver esta ecuación iterativamente, resolviendo en cada paso un problema lineal de autovalores, cuyo potencial efectivo está fijo; una vez encontradas la funciones de onda, se generara una nueva densidad y un nuevo potencial efectivo. Esto se repite hasta encontrar una solución autoconsistente, es decir, aquella en que el potencial generado por las autofunciones sea igual al potencial con las que éstas fueron generadas. Un esquema de este proceso se detalla en la figura 3.1.

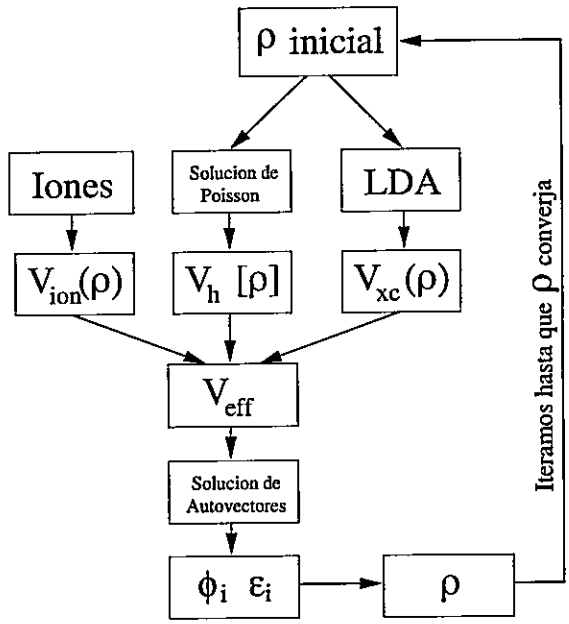

Figura 3.1: Esquema de la iteración de autoconsistencia

3.1.1. Iteración de autoconsistencia

En cada iteración de convergencia tenemos una densidad de entrada, generada en la iteración anterior, lo que implica que para la primera iteración tenemos que dar un valor de densidad inicial, que en este caso se toma constante en todo el dominio.

A partir de la densidad calculamos el potencial efectivo como la suma de las siguientes partes:

- El potencial de intercambio y correlación

$$V_{xc}(r) = \epsilon_{xc}(\rho(r)) + \rho(r)\frac{d\epsilon_{xc}}{d\rho}(\rho(r)) \ , \tag{3.1}$$

donde ϵ_{xc} es la densidad de energía de intercambio y correlación de LDA dada por la ecuación (2.31). En caso de usar GGA u otras aproximaciones para la energía de intercambio y correlación, sólo es necesario usar el funcional correspondiente en este paso.

- El potencial de Hartree y iónico, que calculamos resolviendo la ecuación de Poisson

$$\nabla^2 V(r) = -4\pi \left[\sum_{i=0}^{p-1} z_i \delta(r - r_i) - \rho(r) \right]. \tag{3.2}$$

Suponiendo que el sistema tiene p iones, de carga z_i ubicados en las posiciones r_i, la condición de borde (para un sistema aislado) es que $V|_{r=\infty} = 0$.

Una vez calculado el potencial efectivo para la nueva densidad, lo mezclamos con el que usamos en la iteración anterior, de acuerdo a la fórmula

$$V_{ef} = (1 - \lambda) V_{ef}^{old} + \lambda V_{ef}^{new}. \tag{3.3}$$

Ahora resolvemos la ecuación (2.5) pero con V_{ef} fijo, lo que la convierte en una ecuación de Schrödinger estándar. Una vez encontrados los primeros q autovectores ϕ_i, podemos calcular la nueva densidad de acuerdo a la ecuación (2.6).

3.1.2. Cálculo de energías

Una vez que hemos terminado el ciclo de autoconsistencia podemos calcular la energía del sistema a partir de las integrales de la densidad, dadas en la sección 2.2.

3.1.3. Caso con tratamiento de spin

En el caso de que consideremos spin no apareado se debe realizar el proceso de autoconsistencia para dos densidades, y en cada paso de autoconsistencia resolver dos problemas de autovalores. El proceso se detalla en la figura 3.2.

3.2. Discretización

Para poder tratar numéricamente las ecuaciones de Kohn-Sham es necesario discretizar el problema. Una forma es usar una base de funciones, típicamente ondas

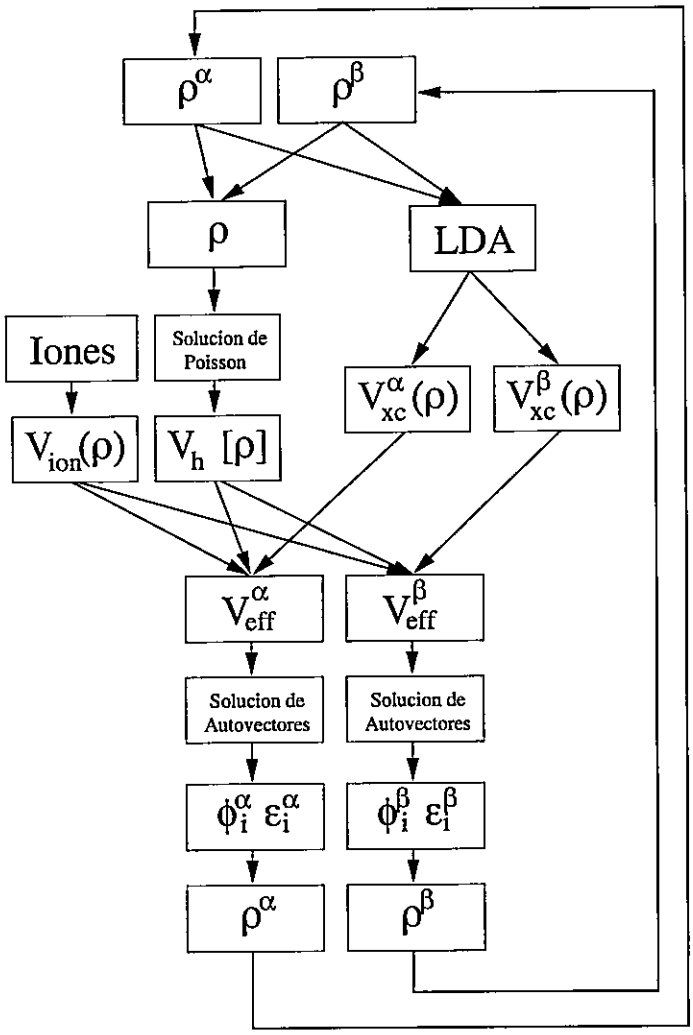

Figura 3.2: Esquema de la iteración de autoconsistencia para el caso de spin no apareado.

planas o funciones gaussianas, en donde se truncan los coeficientes de la expansión de las funciones para obtener un número finito de valores que representen la función.

En este caso usaremos una representación en el espacio real tomando los valores de la función en una serie finita de puntos.

3.2.1. Discretización de las funciones

Las funciones son representadas como una serie de valores reales dispuestos sobre una grilla de puntos en el espacio. La grilla es tridimensional y tiene un espaciado uniforme h. Cada punto de la grilla es designado por tres índices enteros (i, j, k) y representa el valor de la función en las coordenadas del punto

$$f_{ijk} = f\left(hi, hj, hk\right) \ . \tag{3.4}$$

Cuando la función varía muy rápido o es singular, en el punto se puede asignar el valor dado por el promedio de la función sobre el cubo de lado h centrado en (i, j, k),

$$f_{ijk} = \frac{1}{h^3} \int_{-h/2}^{h/2} \int_{-h/2}^{h/2} \int_{-h/2}^{h/2} f\left(hi + x, hj + y, hk + z\right) \, dx \, dy \, dz \ . \tag{3.5}$$

Con esta discretización conservamos la noción de las funciones como vectores, ya que las grillas son una serie de valores reales que pueden ser tratados como vectores de dimensión finita.

3.2.2. Dominio de solución

El uso de una grilla implica que sólo resolveremos en un dominio finito del espacio. Usaremos condiciones de borde tipo Dirichlet, fijando el valor de las soluciones sobre los puntos de los bordes de la grilla.

También es posible usar condiciones de borde periódicas, usando como vecinos de los puntos de los bordes los puntos del borde opuesto.

3.2.3. Integración y producto interno

La integral de una función sobre la grilla se calcula como la suma de los valores sobre la grilla, dados por f_{ijk}, multiplicados por el volumen de las celdas de la grilla

$$\int\int\int f(x,y,z)\, dxdydx = \sum_{i,j,k} h^3 c_{ijk} f_{ijk} \,. \tag{3.6}$$

Los c_{ijk} están dados por la fórmula de integración discreta. En este caso usaremos la fórmula de integración trapezoidal donde $c_{ijk} = 1$, salvo en los bordes donde valen 1/2, 1/4 y 1/8, dependiendo de si estamos cerca de uno, dos o tres bordes, respectivamente. De esta forma el producto interno entre dos funciones es

$$\langle f \mid g \rangle = \sum_{i,j,k} h^3 c_{ijk} f_{ijk} g_{ijk} \tag{3.7}$$

y la norma correspondiente es

$$\langle f \mid f \rangle = |f|^2 = \sum_{i,j,k} h^3 c_{ijk} f_{ijk}^2 \,. \tag{3.8}$$

3.2.4. Discretización del operador laplaciano

Existen varios metodos para calcular el operador Laplaciano sobre una grilla discreta. En este trabajo utilizamos 2 de ellos, el primero a través de una expansión de la alto orden de la segunda derivada y el segundo el metodo de *Mehrstellenverfahren.*

Expansión de la segunda derivada

Una forma de calcular el laplaciano de una función representada sobre la grilla es usar el método de la segunda derivada en diferencias finitas de alto orden. En general la segunda derivada de una función analítica se puede calcular de manera discreta por la fórmula

$$\frac{\partial^2 u_i}{\partial x^2} = \frac{1}{h^2} \sum_{k=-n/2}^{n/2} C_k^n u_{i+k} + \mathcal{O}\left(h^n\right), \tag{3.9}$$

que se conoce como la segunda derivada discreta de orden n (con n par), donde $\{C^n\}$ es el conjunto de coeficientes constantes que solo dependen de n y que se determinan a partir de las expansiones en serie de Taylor de la función sobre los puntos en la grilla. Los valores de los coeficientes para órdenes desde $n = 2$ a 8 se muestran en la tabla 3.1. Una manera simple de encontrar los coeficientes para un orden arbitrario se puede encontrar en [36].

Orden	Puntos	Prefactor	Coeficentes				
2	3	$1/h^2$	-2	1			
4	5	$1/12h^2$	-30	16	-1		
6	7	$1/180h^2$	-490	270	-27	2	
8	9	$1/5040h^2$	-14350	8064	-1008	128	-9

Tabla 3.1: Coeficientes de la expansión de la segunda derivada. El primer coeficiente corresponde al punto central (la expansión es simétrica respecto al punto central); cada término debe ser multiplicado por el prefactor.

Para calcular el laplaciano discreto simplemente se suman las expansiones correspondientes a cada dirección. Si en una coordenada estamos cerca de un borde de la grilla por lo que no hay puntos suficientes para calcular la segunda derivada a cierto orden, usamos una expresion de menor orden en esa dirección.

Discretización *Mehrstellenverfahren*

El método de *Mehrstellenverfahren* (MV) o método Hermítico se basa en la generalización de Hermite de las series de Taylor [28], de la cual se puede derivar la siguiente relación

$$\sum u_k + 2\sum u_s - 24u_m - \frac{1}{2}h^2\left(\sum \nabla^2 u_s + 6\nabla^2 u_m\right) = 0 + \mathcal{O}\left(h^6\right) , \qquad (3.10)$$

donde u_m es el valor de la función en el punto central, u_s son los valores de los primeros vecinos en la grilla y u_k los segundos vecinos. En este caso más que el operador,

discretizaremos de manera completa la ecuación diferencial, que supondremos tiene la forma

$$\nabla^2 u = F \ , \tag{3.11}$$

donde F puede contener operadores locales actuando sobre u.

Si reemplazamos la ecuación (3.11) en la fórmula (3.10) obtenemos la aproximación de Mehrstellenverfahren (MV) para la ecuación (3.11)

$$\frac{1}{6h^2}\left(\sum u_k + 2\sum u_s - 24u_m\right) = \frac{1}{12}\left(\sum F_s + 6F_m\right) + \mathcal{O}\left(h^4\right) \tag{3.12}$$

La ventaja de este método con respecto al anterior, es que a pesar de no tener un orden demasiado alto, en la expansión se usa información más local, no hay que hacer consideraciones especiales cerca de los bordes y es más simple de calcular. La desventaja es que hace que el término en el lado derecho de la ecuación discretizada no sea diagonal.

3.2.5. Discretización del potencial íonico

El potencial iónico en forma exacta es

$$V_0\left(\vec{r}\right) = \sum_{i=0}^{p-1} \frac{Z_i}{\left|\vec{r} - \vec{r}_i\right|} \ . \tag{3.13}$$

Notemos que esta fórmula, al ser representada directamente sobre los puntos de la grilla producirá problemas cuando un punto de la grilla, se encuentre muy cercano a la posición de un átomo; el valor del potencial será muy grande y además cambiará mucho con una pequeña variación de la distancia del ión al punto de la grilla, resultando, por ejemplo, que dos átomos que poseen el mismo z tengan valores muy distintos del potencial. Para solucionar este problema se calculó el potencial

iónico por medio de la ecuación de Poisson, junto con el potencial de Hartree, como se mostro en la ecuación 3.2.

Aun tenemos el problema de discretizar la densidad carga iónica, para esto se supone que toda la carga de este se encuentra sobre el punto de la grilla más cercano a la posición real, y en este punto de la grilla la densidad tiene un valor $\rho_i = Z_i/h^3$. Esto es equivalente a suponer que la carga del ion está distribuida sobre un cubo de tamaño h^3 con centro en la grilla, por lo que produce un potencial iónico que es finito sobre el punto.

3.3. Solución multinivel de la ecuación de Poisson

Como parte del proceso de solución de las ecuaciones de Kohn-Sham es necesario resolver la ecuación de Poisson (3.2)

$$\nabla^2 \varphi = f \ , \tag{3.14}$$

con condiciones de borde $\varphi|_{r=\infty} = 0$.

Para representar el problema sobre un dominio finito, usamos condiciones de borde tipo Dirichlet sobre el borde de la caja, y los valores que se asignan en ese borde son dados una expansión cuadrupolar del potencial.

La discretización de la ecuación (3.14) mediante los métodos descritos en la sección 3.2.4, da como resultado una ecuación algebraica lineal

$$Ax = b \ , \tag{3.15}$$

donde x y b son vectores de dimensión N, y A es una matriz de $N \times N$, siendo N el número de puntos en la grilla. Una particularidad de A es que muchos de sus

componentes son nulos, lo que se conoce como una matriz dispersa, por lo que no conviene representar la matriz por sus componentes sino que usarla aplicándola sobre un vector.

El método que utilizaremos para resolver la ecuación (3.15) se conoce como método *multigrid* o método multinivel, que se caracteriza por su eficiencia y escalabilidad. Además, es posible usarlo a una variedad de problemas más allá de la ecuación de Poisson: es muy común el uso de métodos de multigrid en simulaciones de dinámica de fluidos y otros sistemas descritos por ecuaciones elípticas; también es la base del método que usamos en este trabajo para resolver problemas de autovalores.

La base de este método la constituyen los esquemas iterativos o de relajación, que consisten en generar iterativamente una secuencia de aproximaciones a la solución $x^{(n)}$, que convergen al valor exacto x.

3.3.1. Métodos iterativos y de relajación

La estrategia de los métodos iterativos para resolver ecuaciones del tipo (3.15) es utilizar una matriz parecida a A, que llamaremos C, pero más simple de invertir. Esta matriz se itera de la siguiente forma

$$x^{(n+1)} = x^{(n)} + C^{-1}\left[b - Ax^{(n)}\right] . \tag{3.16}$$

La elección más simple es tomar C como la diagonal de A, lo que resulta en el método de Jacobi o Gauss-Jacobi, que aplicado a una ecuación de Poisson bidimensional discretizada a segundo orden resulta en la conocida fórmula

$$x_{ij}^{(n+1)} = \frac{1}{4}\left[x_{i+1\,j}^{(n)} + x_{i-1\,j}^{(n)} + x_{i\,j+1}^{(n)} + x_{i\,j-1}^{(n)} - h^2 b_{i\,j}\right] . \tag{3.17}$$

También es posible elegir C como la diagonal y la parte triangular inferior de A,

lo que se conoce como el método de Gauss-Seidel. Para el mismo caso de Poisson anterior la fórmula de iteración es

$$x_{ij}^{(n+1)} = \frac{1}{4}\left[x_{i+1\,j}^{(n)} + x_{i-1\,j}^{(n+1)} + x_{i\,j+1}^{(n)} + x_{i\,j-1}^{(n+1)} - h^2 b_{i\,j}\right] , \tag{3.18}$$

equivalente a realizar Gauss-Jacobi pero utilizando los nuevos puntos a medida que van siendo calculados. La convergencia de este método es algo mejor que la del método de Jacobi y requiere menos almacenamiento.

En tres dimensiones, y con un laplaciano expandido a un orden arbitrario l, la iteración de Gauss-Seidel es

$$x_{ijk}^{(n+1)} = x_{ijk}^{(n)} + \frac{1}{-3C_0}\left\{\sum_{m=-l/2}^{l/2} C_k\left[x_{i+m\,j\,k} + x_{i\,j+m\,k} + x_{i\,j\,k+m}\right] - h^2 b_{ijk}\right\} . \tag{3.19}$$

El problema de los métodos de relajación es que tienen una mala tasa de convergencia, lo que hace que sean muy ineficientes para grillas grandes. Esto sucede porque son efectivos en la remoción del error que varia en una distancia comparable a la separación de la grilla, pero resultan muy ineficientes en el momento de remover el error que varia en distancias mucho mayores a la separación de la grilla.

Este efecto, se puede apreciar claramente en la figura (3.3), donde se muestran las distintas etapas de un proceso de solución de la ecuación de Poisson por relajación. Luego de unas pocas iteraciones la aproximación a la solución es suave, pero dista mucho de la solución exacta que solo se consigue mediante muchas iteraciones más.

Si miramos como varía el error dentro del proceso de iteración, vemos que la relajación es muy efectiva en las primeras iteraciones pero la convergencia se vuelve muy lenta luego. Este hecho se puede apreciar en el gráfico en la figura (3.4), donde el error en el paso n de iteración se define como $\left|b - Ax^{(n)}\right|$.

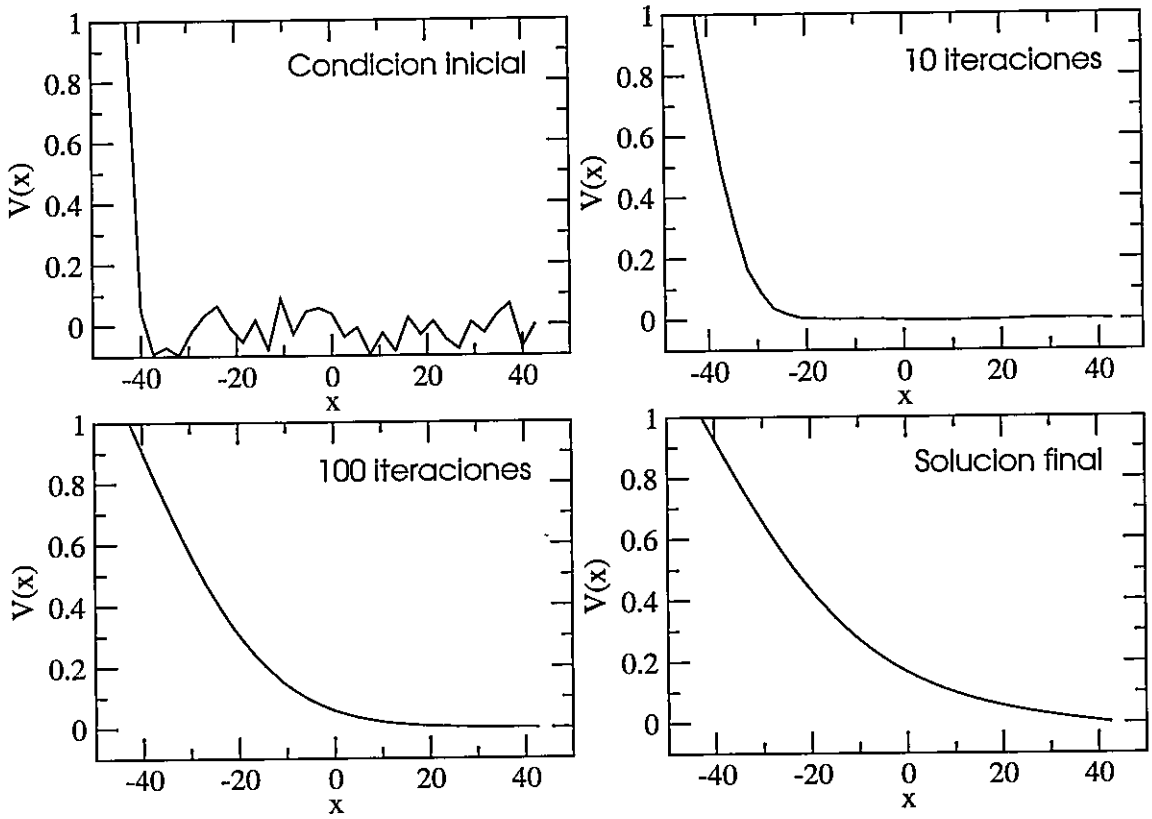

Figura 3.3: Varios pasos dentro de un proceso de solución por relajación para una caja con una pared a potencial 1, el resto a potencial 0, con grilla de 65^3 puntos.

El método de multigrid se basa en llevar el problema a grillas con un menor número de puntos (de un mayor grosor), donde la relajación es más efectiva. Para hacer esto se transfiere, no la ecuación en sí, sino la ecuación residual.

3.3.2. La ecuación residual

Si tenemos una ecuación lineal del tipo (3.15) y una aproximación a la solución x, podemos definir el error de aproximación como $\tau = b - Ax$, y a partir de este, definimos la ecuación residual como

$$Ad = \tau \ , \tag{3.20}$$

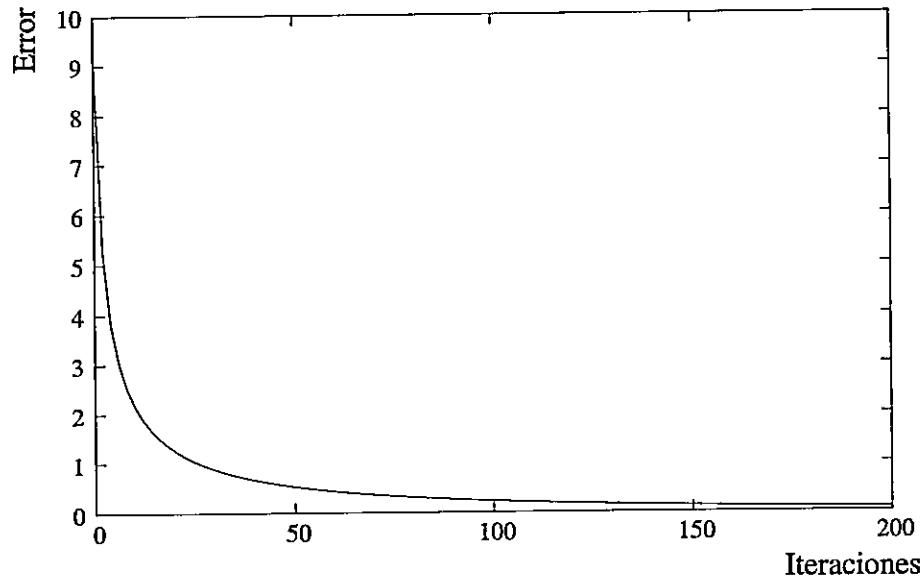

Figura 3.4: Convergencia de la relajación: Error en función del número de iteraciones para una relajación de Gauss-Seidel en caja con una pared a potencial 1, el resto a potencial 0, en una grilla de 65^3 puntos.

con la condición que $d = 0$ sobre los bordes del dominio. Una vez que tenemos la solución de la ecuación residual $\bar{d}$ (conocida como corrección), podemos encontrar la solución del problema original como

$$\bar{x} = x + \bar{d} \,. \tag{3.21}$$

Conviene notar que cuando la aproximación a la solución x es la solución exacta $\bar{x}$, la ecuación residual tiene solución nula.

3.3.3. El algoritmo multinivel

El principio de multigrid es el siguiente: si realizamos unos pocos pasos de relajación sobre una aproximación inicial cualquiera, sabemos que la aproximación de la solución que tendremos será suave (como se vio en la figura 3.3), esto es, solo tendrá error cuyas variaciones serán mucho mas grandes que la separación de la

grilla.

Este error es factible de ser representado en una grilla con un número menor de puntos, de modo que podemos llevar la ecuación residual del problema a una grilla más gruesa. En ésta la ecuación residual puede ser resuelta con menor costo, debido a que el número de puntos es menor y a que la relajación es más eficiente al ser mayor la separación de la grilla.

Luego tomamos la solución de la ecuación residual en la grilla gruesa y la llevamos a la grilla más fina, sumándola a la aproximación que teníamos antes. Esta nueva aproximación será muy cercana a la solución, y más que nada tendrá error de alta frecuencia, que es fácil remover con unos pocos pasos más de relajación.

Sin embargo, hay algunos detalles que deben ser considerados:

- Cómo elegir las grillas más gruesas.
- Cómo transferir las soluciones y el error entre grillas de distinto grosor.

En este caso consideraremos las grillas más gruesas conteniendo la mitad de puntos por cada dimensión de la manera en que se observa esquemáticamente en la figura (3.5).

Para transferir de una grilla gruesa a una fina utilizamos interpolación lineal, donde los puntos que coinciden en ambas grillas, toman en la grilla fina el valor del punto en la gruesa y los demás puntos se calculan como el promedio del valor de los primeros vecinos.

El operador que tranfiere de una grilla fina a una gruesa se conoce como operador de restricción. Este calcula el valor de cada punto de la grilla gruesa como el promedio de los valores vecinos a este en la grilla fina, incluyendo al punto coincidente. En este trabajo se usó la restricción de peso completo, *i.e* donde los pesos de cada punto son

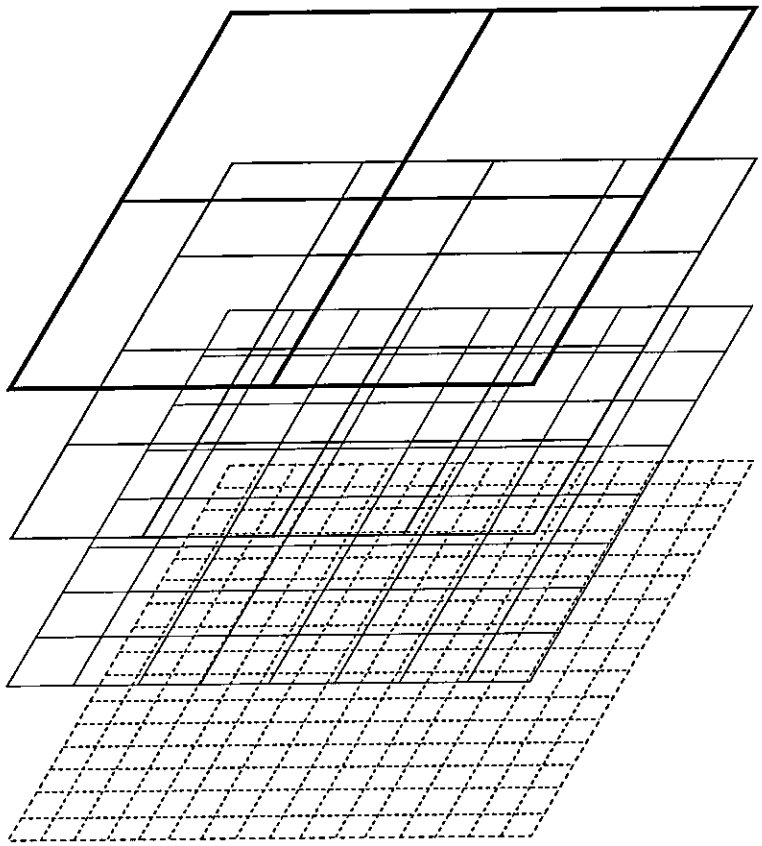

Figura 3.5: Esquema en 2D de la jerarquía de grillas utilizada en el algoritmo de Multigrid. Los puntos en que se representa la función corresponden a la intersección de las rectas.

1/8 para el punto coincidente, 1/16 para los primeros vecinos, 1/32 para los segundos vecinos, y 1/64 para los terceros.

Normalmente el método multigrid se aplica sucesivamente en cada grilla, llevando el proceso de relajación por los distintos niveles, llegando finalmente a la grilla más gruesa, donde la solución exacta es encontrada por unos pocos pasos de relajación.

En el algoritmo 1 detallamos el proceso de multigrid: se aplica sobre una serie de grillas de distinto grosor, denotadas por un superíndice l que toma valores desde L en la grilla más fina, hasta 0 en la grilla más gruesa. A^l será la discretización del operador en el nivel l, I_l^{l-1} es la interpolación y I_{l-1}^l el operador de restricción.

Algoritmo 1 Ciclo $V(\nu_1, \nu_2)$ de Multigrid

- Se elige una condición inicial x en el nivel más grueso L
- Definimos $d^L = x$ y $\tau^L = b$
- Para $l = L$ hasta $l = 0$:
  - Se realizan ν_1 pasos de relajación sobre la ecuación $A^l d^l = \tau^l$
  - Se calcula el error $e^l = b^l - A^l x^l$
  - Se transfiere a la grilla más gruesa, $\tau^{l-1} = I_l^{l-1} e^l$
- Resolvemos por relajación en el nivel 0
- Para $l = 0$ hasta $l = L$:
  - Tranferimos la corrección a la grilla fina, $d^l = d^l + I_{l-1}^l d^{l-1}$
  - Se realizan ν_2 pasos de relajación sobre la ecuación $A^l d^l = \tau^l$

3.3.4. Convergencia de multigrid

El método de multigrid tiene una tasa de convergencia muy superior a la obtenida usando únicamente relajación. En la figura 3.6 se observa como varía el error durante

un proceso de solución por multigrid. En cada paso de iteración que corresponde a un ciclo $V(2,2)$, sólo se realizan 4 pasos de relajación en la grilla más fina.

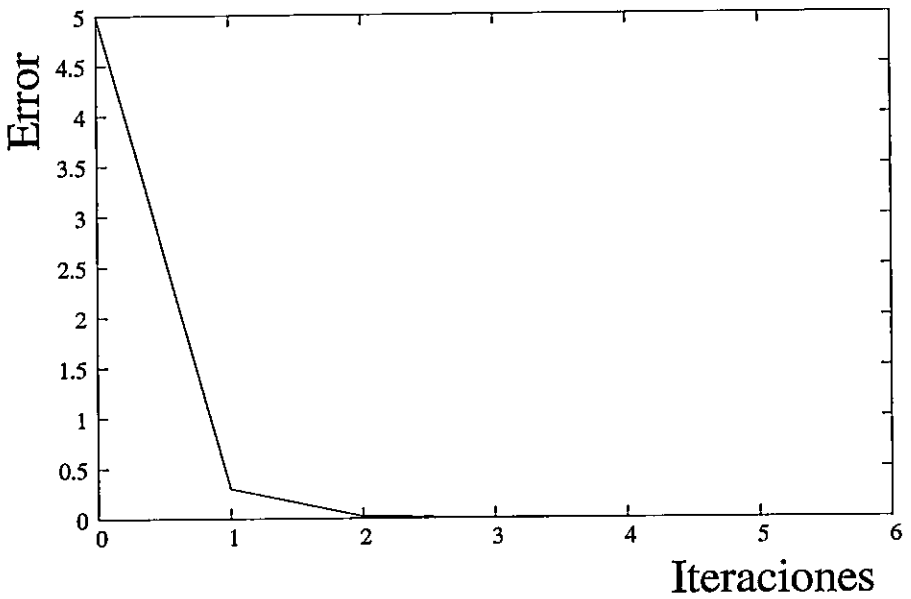

Figura 3.6: Convergencia de multigrid: Error en función del número de iteraciones, cada iteración corresponde a un ciclo multigrid $V(2,2)$. El problema es el mismo que para la figura 3.4

3.3.5. Proceso de solución de Poisson

Para resolver la ecuación de Poisson con una precisión deseada, digamos e_{min}, simplemente realizamos de manera consecutiva varios ciclos de multigrid hasta converger. El valor inicial se obtiene a partir de la solución de la iteración de autoconsistencia anterior, o simplemente se usa $x = 0$. El proceso se detalla en el algoritmo 2.

Algoritmo 2 Proceso de solución de la ecuación de Poisson

- Mientras $e > e_{min}$
  - Ciclo de $V(\nu_1, \nu_2)$ de Multigrid sobre $Ax - b = 0$
  - $e = |Ax - b|$

3.4. Solución de problemas de autovalores

La parte central del proceso de iteración de autoconsistencia es resolver la ecuación de autovalores (2.5) con un potencial fijo. Usando las técnicas de discretización de la sección 3.2.4 llegamos a la ecuación de autovalores discreta

$$H|\phi\rangle = \varepsilon M|\phi\rangle \ , \tag{3.22}$$

donde H y M son matrices $N \times N$. En el caso de la discretización en derivadas centradas M es la identidad, y la parte no diagonal de H sólo contiene contribuciones del laplaciano discreto. Para la discretización MV, M no es diagonal, pero sí definida positiva y H incluye términos no diagonales que provienen del potencial.

3.4.1. Aplicación de multigrid a problemas de autovalores

Existen varias ampliaciones del método de multigrid para resolver problemas de autovalores. En algunos casos se utiliza el método de multigrid para resolver las ecuaciones lineales que surgen dentro de un método iterativo [18], pero también se han diseñado métodos que tratan de manera directa el problema en un esquema multigrid [19].

El método que utilizamos se conoce como *Rayleigh Quotient Multigrid* (RQMG), y fue propuesto por Mandel y McCormick en 1989 [32] y adaptado a cálculos de estructura electrónica por Heiskanen, Torsti, Puska y Nieminen en 2001 [33]. Así co-

mo multigrid para la ecuación de Poisson se basa en un método de relajación como Gauss-Jacobi o Gauss-Seidel, esta técnica de multigrid se basa en una técnica de relajación conocida como relajación de coordenadas.

3.4.2. Relajación de coordenadas

La relajación de coordenadas es un método iterativo de solución de problemas que se basa en un principio minimal. Consiste en iterar sobre las diferentes componentes de la solución (los puntos de la grilla) y modificar su valor de manera que minimicen localmente el funcional de acción asociado. En el caso de la ecuación de Poisson la relajación de coordenadas es equivalente al método de Gauss-Seidel.

En el caso de un problema de autovalores hermítico generalizado, como la ecuación (3.22), el estado fundamental $|\phi_0\rangle$ minimiza el cuociente de Rayleigh

$$R = \frac{\langle\phi| H |\phi\rangle}{\langle\phi| M |\phi\rangle} \tag{3.23}$$

y para el mínimo, el valor del cuociente es igual al autovalor correspondiente al estado fundamental ε_0.

Dada un cierto estado $|\phi\rangle$, tomamos una dirección de corrección $|d\rangle$, en la que modificamos nuestra aproximación. El cuociente de Rayleigh cambia entonces a

$$R(s) = \frac{(\langle\phi| - s\langle d|) H (|\phi\rangle - s|d\rangle)}{(\langle\phi| - s\langle d|) M (|\phi\rangle - s|d\rangle)} \tag{3.24}$$

$$= \frac{\langle\phi| H |\phi\rangle - 2s\langle\phi| H |d\rangle + s^2\langle d| H |d\rangle}{\langle\phi| M |\phi\rangle - 2s\langle\phi| M |d\rangle + s^2\langle d| M |d\rangle} . \tag{3.25}$$

A continuación, buscamos el valor de s que minimize R, igualando a cero la derivada de $R(s)$. Esto nos da una ecuación cuadrática, cuya solución correspondiente al valor mínimo es

$$\sigma = \frac{-2C}{B + \sqrt{B^2 - 4AC}} \tag{3.26}$$

con los coeficientes

$$A = \langle d| M |d\rangle \langle \phi| H |d\rangle - \langle d| H |d\rangle \langle \phi| M |d\rangle \tag{3.27}$$

$$B = \langle \phi| M |\phi\rangle \langle d| H |d\rangle - \langle \phi| H |\phi\rangle \langle d| M |d\rangle \tag{3.28}$$

$$C = \langle \phi| H |\phi\rangle \langle \phi| M |d\rangle - \langle \phi| M |\phi\rangle \langle \phi| H |d\rangle \ . \tag{3.29}$$

Escogeremos la función $|d\rangle$ de manera que valga $1/h^3$ sobre un punto de la grilla y cero en el resto de los puntos. Con estos resultados, un paso de relajación de coordenadas sobre todos los puntos de la grilla puede ser descrito como se observa en el algoritmo 3.

3.4.3. Varios autovalores

Para encontrar varios autovalores usamos el mismo principio variacional, pero agregamos al cuociente de Rayleigh para cada autovector, un término de traslape con los autovectores de menor energía, el que garantiza que no converjan todos al estado fundamental.

$$R_n = \frac{\langle \phi_n| H |\phi_n\rangle}{\langle \phi_n| M |\phi_n\rangle} + \sum_{m=0}^{n-1} \omega_m \frac{\langle \phi_m \mid \phi_n\rangle^2}{\langle \phi_m \mid \phi_m\rangle} \tag{3.30}$$

El parámetro ω_m es un peso que regula la separación entre las funciones de onda. Mientras mayor sea ω_m (respecto del potencial), mayor precisión es posible lograr en las funciones de onda, pero más lenta es la convergencia. En este trabajo se usó un valor de ω_m fijo para todas las funciones de onda y para todas las iteraciones.

Algoritmo 3 Relajación de coordenadas

- Definimos $a = \langle\phi| H |\phi\rangle$ y $b = \langle\phi| M |\phi\rangle$
- Para cada punto i, j, k
  - $d_{ijk} = \delta_{ijk}/h^{3/2}$
  - $\alpha_{ijk} = h^{3/2} (H\phi)_{ijk} = \langle\phi| H |d\rangle$
  - $\beta_{ijk} = h^{3/2} (M\phi)_{ijk} = \langle\phi| M |d\rangle$
  - $D_H = \langle d| H |d\rangle$
  - $D_M = \langle d| M |d\rangle$
  - $A = \alpha_{ijk} D_M - D_H \beta_{ijk}$
  - $B = bD_H - aD_M$
  - $C = a\beta_{ijk} - b\alpha_{ijk}$
  - $\sigma = -2C/\left(B + \sqrt{B^2 - 4AC}\right)$
  - $\phi_{ijk} = \phi_{ijk} - \sigma/h^{3/2}$
  - $a = a - 2\sigma\alpha_{ijk} + \sigma^2 D_H$
  - $b = b - 2\sigma\beta_{ijk} + \sigma^2 D_M$

Si consideramos el término de separación en la minimización, tenemos la siguiente corrección a los coeficientes de la ecuación cuadrática para σ_m:

$$A' = A + \gamma_2 \langle d \mid d \rangle - \gamma_1 \langle \phi_n \mid d \rangle \tag{3.31}$$

$$B' = B + \gamma_1 \langle \phi_n \mid \phi_n \rangle - \gamma_3 \langle d \mid d \rangle \tag{3.32}$$

$$C' = C + \gamma_3 \langle \phi_n \mid d \rangle - \gamma_2 \tag{3.33}$$

con γ_1, γ_2 y γ_3 dados por

$$\begin{aligned}
\gamma_1 &= \sum_{m=0}^{n-1} \omega_m \frac{\langle \phi_m \mid d \rangle^2}{\langle \phi_m \mid \phi_m \rangle} \\
\gamma_2 &= \sum_{m=0}^{n-1} \omega_m \frac{\langle \phi_m \mid d \rangle \langle \phi_m \mid \phi_n \rangle}{\langle \phi_m \mid \phi_m \rangle} \\
\gamma_3 &= \sum_{m=0}^{n-1} \omega_m \frac{\langle \phi_m \mid \phi_n \rangle^2}{\langle \phi_m \mid \phi_m \rangle} .
\end{aligned}$$

3.4.4. Algoritmo multinivel

En la relajación de coordenadas, es necesario elegir un vector de soporte finito como dirección de minimización. Ello hace que la minimización sobre cada punto sea una operación local, de manera que el costo de iteración sobre toda la grilla es proporcional al número de puntos en ésta. La desventaja es que esta elección de base hace que la convergencia de la relajación de coordenadas de por sí sea lenta. Es en este punto donde el método de multigrid puede acelerar el proceso de convergencia sin aumentar el costo computacional.

Al igual que para la ecuación de Poisson, definimos una jerarquía de grillas cuyos niveles van desde $l = L$ (la más fina) a $l = 0$ (la más gruesa). Nuestro objetivo

es minimizar el cuociente de Rayleigh en la grilla más fina, pero elegiremos como dirección de minimización $\left|I_l^L d^l\right\rangle$, lo que nos da

$$R(s) = \frac{\left(\left\langle\phi^L\right| - s\left\langle I_l^L d^l\right|\right) H^L \left(\left|\phi^L\right\rangle - s\left|I_l^L d^l\right\rangle\right)}{\left(\left\langle\phi^L\right| - s\left\langle I_l^L d^l\right|\right) M^L \left(\left|\phi^L\right\rangle - s\left|I_l^L d^l\right\rangle\right)} \tag{3.34}$$

$$= \frac{\left\langle\phi^L\right| H^L \left|\phi^L\right\rangle - 2s\left\langle\phi^L\right| H^L \left|I_l^L d^l\right\rangle + s^2 \left\langle I_l^L d^l\right| H^L \left|I_l^L d^l\right\rangle}{\left\langle\phi^L\right| M^L \left|\phi^L\right\rangle - 2s\left\langle\phi^L\right| M^L \left|I_l^L d^l\right\rangle + s^2 \left\langle I_l^L d^l\right| M^L \left|I_l^L d^l\right\rangle} \tag{3.35}$$

Tanto en el numerador como en el denominador, el primer término se calcula en la grilla fina, pero los otros dos podemos reescribirlos en términos de una integral sobre la grilla más gruesa l

$$R(s) = \frac{\left\langle\phi^L\right| H^L \left|\phi^L\right\rangle - 2s\left\langle\phi^l\right| H^l \left|d^l\right\rangle + s^2 \left\langle d^l\right| H^l \left|d^l\right\rangle}{\left\langle\phi^L\right| M^L \left|\phi^L\right\rangle - 2s\left\langle\phi^l\right| M^l \left|d^l\right\rangle + s^2 \left\langle d^l\right| M^l \left|d^l\right\rangle} . \tag{3.36}$$

Lo que nos dice que podemos llevar el proceso de relajación de coordenadas a una grilla gruesa, pero minimizando el cuociente de la grilla más fina. Esto resulta en aumento de la efectividad del proceso junto con una disminución del costo computacional.

En el algoritmo 4 se muestra un ciclo $V(\nu_1, \nu_2)$ de RQMG para el conjunto $\left\{\left|\phi_n^L\right\rangle\right\}$ de los N primeros autovectores de la ecuación (3.22).

Dentro de este esquema cada paso de relajación de coordenadas para $\left\{\left|\phi_n^L\right\rangle\right\}$ en el nivel l se hace de la forma descrita en el algoritmo 5.

En el caso de que la matriz M de la ecuación (3.22) sea la identidad, no es necesario calcular b_n ni $\left|r_n^l\right\rangle$, ya que son iguales a c_{nn} y $\left|p_n^l\right\rangle$ respectivamente.

3.4.5. Ortogonalización

Para ortonormalizar los autovectores usamos el proceso de Grahm-Schmidt modificado, matemáticamente equivalente al método de ortogonalización de Grahm-

Algoritmo 4 Ciclo $V(\nu_1, \nu_2)$ de RQMG

- Hacemos ν_1 pasos de relajación de coordenadas sobre $\left\{\left|\phi_n^L\right\rangle\right\}$
- Ortonormalizamos $\left\{\left|\phi_n^L\right\rangle\right\}$
- Para n desde 0 hasta L:
  - $a_n = \left\langle \phi_n^L \right| H^L \left| \phi_n^L \right\rangle$
  - $b_n = \left\langle \phi_n^L \right| M^L \left| \phi_n^L \right\rangle$
  - Para cada m desde 0 hasta n:
    - $c_{mn} = \left\langle \phi_m^L \middle| \phi_n^L \right\rangle = \delta_{mn}$
  - $\left|q_n^L\right\rangle = 0$
  - $\left|r_n^L\right\rangle = 0$
  - $\left|p_m^L\right\rangle = 0$
- Para $l = L-1$ hasta $l = 1$:
  - Para n desde 0 hasta L:
    - $\left|r_n^l\right\rangle = I_{l+1}^l \left(\left|r_n^{l+1}\right\rangle + M^{l+1}\left|\phi_n^{l+1}\right\rangle\right)$
    - $\left|q_n^l\right\rangle = I_{l+1}^l \left(\left|q_n^{l+1}\right\rangle + H^{l+1}\left|\phi_n^{l+1}\right\rangle\right)$
    - $\left|p_n^l\right\rangle = I_{l+1}^l \left(\left|q_n^{l+1}\right\rangle + \left|\phi_n^{l+1}\right\rangle\right)$
  - Hacemos ν_1 pasos de relajación de coordenadas sobre $\left\{\left|\phi_n^l\right\rangle\right\}$
- Para $l = 1$ hasta $l = L-1$:
  - Hacemos ν_2 pasos de relajación de coordenadas sobre $\left\{\left|\phi_n^l\right\rangle\right\}$
  - Para n desde 0 hasta L:
    - $\left|\phi_n^{l+1}\right\rangle = \left|\phi_n^{l+1}\right\rangle + I_l^{l+1}\left|\phi_n^l\right\rangle$
- Ortonormalizamos $\left\{\left|\phi_n^L\right\rangle\right\}$
- Hacemos ν_2 pasos de relajación de coordenadas sobre $\left\{\left|\phi_n^l\right\rangle\right\}$
- Ortonormalizamos $\left\{\left|\phi_n^L\right\rangle\right\}$

Algoritmo 5 Relajación de coordenadas en nivel l para varios autovectores

- Para cada punto i, j, k
  - Para cada $n = 0$ hasta $n = N$
    - $d_{ijk} = 1/h^{3/2}$
    - $\alpha = h^{3/2}\left[(H\phi_n)_{ijk} + (q_n)_{ijk}\right] = \langle\phi_n| H |d\rangle + \langle q_n \mid d\rangle$
    - $\beta = h^{3/2}\left[(M\phi_n)_{ijk} + (r_n)_{ijk}\right] = \langle\phi_n| M |d\rangle + \langle r_n \mid d\rangle$
    - $\gamma_n = h^{3/2}\left[(\phi_n)_{ijk} + (p_n)_{ijk}\right] = \langle\phi_n \mid d\rangle + \langle p_n \mid d\rangle$
    - $D_H = \langle d| H |d\rangle$
    - $D_M = \langle d| M |d\rangle$
    - $s_1 = 0$
    - $s_2 = 0$
    - $s_3 = 0$
    - Para $m = 0$ hasta $m = n - 1$:
      - $s_1 = s_1 + \omega_m \gamma_m^2 / c_{mm}$
      - $s_2 = s_2 + \omega_m \gamma_m c_{mn} / c_{mm}$
      - $s_3 = s_3 + \omega_m c_{mn}^2 / c_{mm}$
    - $A = \alpha D_M - D_H \beta + s_2 - \gamma_n s_1$
    - $B = b_n D_H - a D_M + c_{mm} s_1 - s_3$
    - $C = a_n \beta - b_n \alpha + \gamma_n s_3 - s_2$
    - $\sigma_n = -2C / \left(B + \sqrt{B^2 - 4AC}\right)$
    - $(\phi_n)_{ijk} = (\phi_n)_{ijk} - \sigma_n / h^{3/2}$
    - $a_n = a_n - 2\sigma_n \alpha + \sigma_n^2 D_H$
    - $b_n = b_n - 2\sigma_n \beta + \sigma_n^2 D_M$
    - Para $m = 0$ hasta $m = n$:
      - $c_{mn} = c_{mn} - \sigma_m \gamma_n - \sigma_n \gamma_m + \sigma_m \sigma_n$

Schmidt, pero con mejores propiedades numéricas [25]. Este proceso se detalla en el algoritmo 6.

Algoritmo 6 Grahm-schmidt modificado

- $r_{00} = |\phi_0|$
- $|\phi_0\rangle = \frac{1}{r_{00}} |\phi_0\rangle$
- Para $n = 1$ hasta $N - 1$:
  - Para $m = 0$ hasta $n - 1$
    - $r_{nm} = \langle \phi_n \,|\, \phi_m \rangle$
    - $|\phi_n\rangle = |\phi\rangle - r_{nm} |\phi_m\rangle$
  - $r_{nn} = |\phi_n|$
  - $|\phi_n\rangle = \frac{1}{r_{nn}} |\phi_n\rangle$

3.4.6. Proyección de Ritz

Este procedimiento permite separar autovectores cuyos autovalores están muy cercanos, y a los que el proceso de iteración mantendrá mezclados. Lo importante para hacer una proyección de Ritz es que la base formada por la aproximación a los autovectores que tenemos, genere el mismo subespacio que los autovectores exactos. Si tomamos un conjunto $|\phi_n\rangle$ ortonormal de aproximaciones a los q primeros autovectores del operador H, y que subtienden el mismo subespacio formado por los q autovectores exactos $\left|\bar{\phi}_n\right\rangle$, la proyección de Ritz nos permitirá encontrar los autovectores exactos.

El procedimiento consiste en realizar una proyección de los operadores H y M en el subespacio generado por $|\phi_n\rangle$, del que resultan $\mathcal{H}$ y $\mathcal{M}$, matrices $q \times q$ dadas por

$$\mathcal{H}_{nm} = \langle \phi_n | \, H \, | \phi_m \rangle$$

$$\mathcal{M}_{nm} = \langle \phi_n | M | \phi_m \rangle$$

y luego resolver el problema de autovalores de dimensión q

$$\mathcal{H}y = \lambda \mathcal{M}y \ . \tag{3.37}$$

Dados los q autovectores y^k de la ecuación (3.37), podemos construir los autovectores exactos de la ecuación (3.22) como

$$\left|\bar{\phi}_k\right\rangle = \sum_{m=0}^{q-1} y_m^k \left|\phi_m\right\rangle \tag{3.38}$$

con autovalor $e_k = \lambda_k$.

La proyección de Ritz nos devolverá vectores que son ortogonales en la norma dada por M. En el caso en que $M \neq I$, como queremos vectores ortogonales en la norma I usamos la ecuación

$$\mathcal{A}y = \lambda y \tag{3.39}$$

con $\mathcal{A} = \mathcal{M}^{-1}\mathcal{H}$, que en la práctica aproximamos por

$$\mathcal{A}_{jk} \approx \frac{\mathcal{H}_{jk}}{\mathcal{M}_{jk}} \tag{3.40}$$

Como las matrices en la ecuación (3.37) son densas, el costo de encontrar los q autovectores y autovalores es $\mathcal{O}(q^3)$, por lo que para un número muy grande de partículas la proyección de Ritz puede volverse la parte más costosa del cálculo. Sin embargo, en estos casos es posible reducir el cálculo realizando proyecciones de Ritz en grupos de autovectores cuyos autovalores son muy cercanos entre sí [29].

3.4.7. Algoritmo completo de solución de autovalores

El proceso completo que utilizamos para resolver el problema de autovalores se detalla en el algoritmo 7.

Algoritmo 7 Solución del problema de autovalores

- Mientras, para algún n, $e_n > e_{min}$
  - Ciclo de $V(\nu_1, \nu_2)$ de RQMG sobre $\left\{\left|\phi_n^L\right\rangle\right\}$
  - Proyección de Ritz sobre $\left\{\left|\phi_n^L\right\rangle\right\}$
  - Para $n = 0$ hasta $n = q - 1$
    - $\epsilon_n = \langle\phi_n| H |\phi_n\rangle / \langle\phi_n| M |\phi_n\rangle$
    - $e_n = |H\phi_n - \epsilon_n M\phi_n|$

Si no hay una buena aproximación disponible, los autovalores iniciales son generados al azar. Esto es solo necesario al comienzo, ya que los autovalores del anterior paso de autoconsistencia son una buena aproximación inicial.

Para aumentar la efectividad del proceso se utilizó una técnica conocida como bloqueo de autovectores, que consiste en no modificar en el ciclo de multigrid aquellos autovectores cuyo error se encuentra bajo la tolerancia.

3.5. Algoritmo de autoconsistencia

En el marco del método multinivel, para acelerar el proceso de autoconsistencia este se hace en varios niveles. En cada nivel se hace completa la iteración de autoconsistencia; una vez encontrada la densidad, ésta se transfiere al nivel siguiente donde se usa para partir la iteración de ese nivel. También se transfieren los autovectores al nivel fino, ya que son buenos puntos de partida para el proceso de solución. A continuación detallamos el algoritmo que se utiliza: comenzando de un nivel L_i hasta

llegar al nivel más fino L_f.

Algoritmo 8 Algortimo multinivel de iteracion autoconsistente

- Damos densidad inicial ρ^{L_i}
- Para l desde L_i hasta L_f:
  - Mientras $\left|E_{ks} - E_{ks}^{old}\right| > \tau_{sci}$
    - Calculamos $V_{xc}\left[\rho^l\right]$ mediante LDA.
    - $\rho_{tot} = \rho^l + \rho_{ion}$
    - $V \longleftarrow$ Solución de $\nabla^2 V = 4\pi\rho_{tot}$
    - $V_{ef} = (1-\lambda)\, V_{ef} + \lambda\,(V + V_{xc})$
    - ϕ_n, $\epsilon_n \longleftarrow$ Solución de $\left[-\nabla^2/2 + V_{ef}\right]\phi_n = \epsilon_n\phi_n$
    - $E_{ks}^{old} = E_{ks}$
    - $E_{ks} = 2\sum_{n=0}^{q-1}\epsilon_n$
    - $\rho^l = 2\sum_{n=0}^{q-1}\left[\phi_n^l\right]^2$
  - $\rho^{l+1} = I_l^{l+1}\rho^l$
  - Para n entre 0 y $q-1$: $\phi_n^{l+1} = I_l^{l+1}\phi_n^l$

Capítulo 4

Implementación

4.1. Programación

El programa fue realizado en C++ y consta de aproximadamente 15 mil líneas de código. Se eligió este lenguaje de programación por varias razones:

- Es un lenguaje actual, estándar y muy popular en computación científica, disponible en la mayoría de los sistemas operativos, por lo que los programas son fácilmente transferibles.
- Al ser un lenguaje compilado genera código de máquina que es eficiente al ejecutarse. Además existen compiladores optimizados para todas las plataformas.
- Es posible acceder a rutinas externas y bibliotecas en C o Fortran.
- Es un lenguaje orientado al objeto, por lo que permite modularizar el código y reutilizar muchos de los componentes.

Una desventaja de la programación orientada al objeto es que puede resultar en código ineficiente y lento; por lo general este no es un factor importante para muchas aplicaciones, ya que este sacrificio en rendimiento se compensa con un desarrollo

más rápido y código más mantenible. Sin embargo, en este caso, el rendimiento es primordial, de modo que en el diseño se intentó evitar técnicas que pudieran resultar ineficientes.

Todas las funciones cortas se hicieron *inline* dentro de los *loops* internos. Ni en éstos, ni en las partes críticas del programa, se utilizaron técnicas de orientación al objeto que pudieran resultar demasiado perjudiciales para el rendimiento, como por ejemplo funciones virtuales. Esto evita el costo extra de llamadas a funciones, que resultan ineficientes en los procesadores actuales.

No se utilizaron sobrecargas de los operadores $=,+,-$ y $*$ en objetos que almacenan grandes cantidades de datos, ya que implican el uso de valores temporales. En algunos casos se programaron las funciones específicas para evitar el uso de temporales, por ejemplo en el caso del cálculo de elementos de matriz de los operadores. Sin embargo para algunas funciones fue necesario usar explícitamente valores temporales.

4.2. Estructuras de datos

4.2.1. Grillas

Las datos de las grillas se almacenan en objetos de tipo `EeX::Grid3`. Los datos pueden ser accesados a través de tres índices enteros, los cuales pueden tomar un rango de valores arbitrarios; la función de acceso es `EeX::Grid3::operator()` y está declarada *inline* por motivos de rendimiento.

Internamente la clase almacena todos los valores en un arreglo lineal, lo que permite realizar de manera más eficiente operaciones que no dependen de la posición de un dato en la grilla. Algunas de estas operaciones para mayor optimización, se realizan además mediante las subrutinas de la biblioteca BLAS [37].

4.2.2. Jerarquías de grillas

El algoritmo de *multigrid* está formulado en base a una jerarquía de grillas de distinto grosor; en el código éstas están representadas por los objetos del tipo `EeX::GridHier`, cada uno de los cuales contiene una serie de grillas, cada una con la mitad de puntos por lado que la anterior.

4.2.3. Base de grillas

Para manejar un grupo de autovectores se diseñó la clase `EeX::GridBasis` que contiene un arreglo de objetos `EeX::GridHier`. La proyección de Ritz y la ortogonalización de Grahm-Schmidt se encuentran entre los métodos de `EeX::GridBasis`.

4.2.4. Matrices

En algunos algoritmos se requiere operar con matrices densas cuadradas de dimensión arbitraria. Para esto se implementó la clase `EeX::Matrix`, que contiene las operaciones básicas entre matrices y ordena los elementos de acuerdo a la convención de Fortran, por lo que pueden ser directamente usadas en llamadas a las subrutinas de Lapack y Blas.

4.3. Procedimientos

La mayoría del los procedimientos dentro del código están representados por objetos; esto permite tener un código muy modular con componentes intercambiables. Los procedimientos de bajo nivel (como el cálculo de un laplaciano) se hacen intercambiables mediante el uso de *templates*, que no implican un sacrificio en rendimiento, pero que sólo pueden cambiarse a tiempo de compilación. Por otra parte los procedimientos de alto nivel (como la solución de la ecuación de Poisson) se basan

en el uso de herencia y funciones virtuales que permiten seleccionarlos a tiempo de compilación.

Todos los procedimientos derivan de la clase `EeX::Component`; ésta provee el método virtual `EeX::Component::info`, que devuelve una cadena de caracteres describiendo el componente. Esta cadena contiene una breve descripción del método y las referencias bibliográficas sobre éste.

4.3.1. Iteración de autoconsistencia

Esta es la parte principal del código, por lo tanto se encuentra dentro de la función `main`. A futuro, cuando el código permita modificar las posiciones de los átomos será necesario moverla a una rutina separada.

4.3.2. *Solvers*

Para resolver cada ecuación definimos un objeto de tipo *Solver*, al cual se le entrega a través del método `solve` la aproximación inicial a la solución, y éste devuelve la solución con cierta tolerancia previamente definida.

4.3.3. Poisson solver

El algoritmo para el *solver* de Poisson es *Multigrid* en un esquema de corrección, como se detalla en los algoritmos 1 y 2; el objeto correspondiente al *Solver* es `EeX::PoissonSolver`, un *template* que recibe dos objetos, correspondientes al hamiltoniano y al método de relajación. Esto permite que el mismo código sea usado por las dos aproximaciones al laplaciano descritas en la sección 3.2.4.

4.3.4. *Solver* de autovalores

Para resolver la ecuación de autovalores existen dos objetos distintos, correspondientes a las discretizaciones CDS y MV del laplaciano: `EeX::RQMGEigenSolver` y `EeX::RQMGEigenSolverMV`, respectivamente. Ambos están basados en el algoritmo 7, pero están separados porque para el caso de una discretización CDS el ciclo de RQMG (algoritmo 4) puede hacerse más eficientemente al ser diagonal la matriz M, lo que implica que $\left|r_m^L\right\rangle = \left|p_m^L\right\rangle$ y $b_n = c_{nn}$. Ambas clases derivan de la clase `EeX::EigenSolver`, cuyo método `EeX::EigenSolver::solve` es virtual, lo que permite a tiempo de ejecución elegir el método de solución de autovalores.

4.3.5. Laplaciano

Uno de los puntos centrales del código es el cálculo del laplaciano discreto de acuerdo a las fórmulas dadas en la sección 3.2.4.

Segunda Derivada Discreta (CDS)

Para el laplaciano CDS los coeficientes de la expansión (ecuación 3.9) se calculan dentro del código, usando la técnica de Hamming [36] para generar fórmulas de aproximación en diferencias finitas.

Es en la clase `EeX::LapCoeff`, donde los coeficientes se calculan y posteriormente se acceden. Esta clase también puede ser usada para calcular otros coeficientes, como por ejemplo los coeficientes en diferencias finitas para el gradiente que serán necesarios para la implementación de GGA.

Los cálculos de los coeficientes se hacen usando precisión extra, tipo `long double`, debido a que aparecen divisiones entre números muy grandes. Esto no representa una carga adicional debido a que el calculo sólo se hace una vez al principo del programa.

Para calcular el laplaciano CDS, primero debemos verificar si nos encontramos cerca de un borde; si es así reducimos el orden de la aproximación en la dirección correspondiente. Luego, para cada coordenada calculamos la suma 3.9 usando un *loop* y finalmente sumamos los tres resultados. Todas estas operaciones hacen que calcular el laplaciano CDS sea bastante ineficiente computacionalmente, a causa de las ramas y los *loops* adicionales.

La clase `EeX::LapBase` es la que se encarga de calcular el laplaciano CDS en un punto, mediante el método `EeX::LapBase::point_lap`. Los objetos que necesitan calcular el laplaciano CDS derivan de esta clase.

Mehrstellenverfehraren (MV)

El caso del laplaciano MV es más simple debido a que tiene un orden fijo y no hay que hacer consideraciones especiales cerca de los bordes. Entonces se puede incluir explícitamente la suma en el código, sin usar condicionales y ni *loops*. Esta aproximación del laplaciano es calculada por la clase `EeX::MVBase`, mediante la función miembro `EeX::LapBase::point_lap`. Los objetos que usan la aproximación MV derivan de esta clase.

4.3.6. Funcional de intercambio y correlación

El funcional de intercambio y correlación está representado por objetos que derivan de `EeX::XCFunctional`, cuyos métodos son: `EeX::XCFunctional::potential`, que calcula el potencial de intercambio y correlación para una densidad de carga, y `EeX::XCFunctional::energy` que devuelve la energía de intercambio y correlación para una densidad de carga.

Los funcionales implementados actualmente son los descritos en la sección 2.3.2: `EeX::XC_LDA_VWN`, que corresponde al caso con *spin* apareado, y `EeX::XC_LSDA_VWN`,

en el caso con polarización de *spin* .

Para detectar errores en la implementación de estos funcionales, se codificaron separadamente en C++ y en un programa de cálculo simbólico, comparando después los valores que entregaban. Además, para verificar la exactitud de las derivadas analíticas que se hicieron para el cálculo del potencial, se compararon los valores de éstas con derivadas numéricas.

4.4. Implementación de Multigrid

El método de multigrid está formulado en términos de varios operadores (relajación, interpolación, restricción, etc.) y fue implementado de esta misma forma en el código. Cada operación fue representada por un objeto, y usando herencia y funciones virtuales podemos crear algoritmos de *multigrid* generales en que cada operación puede ser intercambiada de manera independiente.

Existe otra formulación alternativa [39] que es más eficiente en el aprovechamiento de la estructura de memoria y cache de los computadores actuales, y que sería interesante considerar para futuras optimizaciones del código.

4.4.1. Operadores

Los operadores derivan de la clase `EeX::Operator`. La operación correspondiente la realizan mediante el método `EeX::Operator::operate`, que recibe 2 argumentos, que corresponden a la grilla de origen sobre la que se realiza la operación, y al destino donde se devuelve el resultado de la operación. También existen los métodos `EeX::Operator::matrix_element` que recibe dos grillas $|u\rangle$ y $|v\rangle$ y calcula el elemento de matriz $\langle u| A |v\rangle$, y `EeX::Operator::matrix_element_rhs`, que calcula el mismo elemento de matriz pero con el operador que aparece en el lado derecho de la

ecuación de autovalores.

Los operadores implementados son `EeX::Laplacian`, que calcula el laplaciano usando CDS a orden arbitrario (especificado en el constructor), y `EeX::MVLaplacian` que es igual al anterior pero que usa la aproximación MV. Para la ecuación de autovalores los operadores son `EeX::Hamiltonian` y `EeX::MVHamiltonian`. Dentro de la clase hay una referencia a la jerarquía de grillas que contiene el potencial asociado.

4.4.2. Transferencias

Existen dos tipos de operadores de transferencia: aquellos que llevan de una grilla con más puntos a una con menos, representados por la clase `EeX::Fine2Coarse`, y los que llevan de una grilla gruesa a una fina, que derivan de la clase `EeX::Coarse2Fine`. En el primer caso es `EeX::Fine2Coarse::restrict` quien realiza la transferencia, mientras que para el segundo es `EeX::Coarse2Fine::interpolate`.

Las operaciones implementadas que llevan a una grilla gruesa fueron: la que reemplaza cada punto de la grilla gruesa por el valor coincidente de la grilla fina: `EeX::Injection`, y `EeX::FullWeight` que realiza el proceso de restricción descrito en la sección 3.3.3.

4.4.3. Relajación

Para la ecuación de Poisson las operaciones de relajación derivan de la clase `EeX::Smoother`, cuyo método virtual `EeX::Smoother::smooth` recibe la aproximación actual a la solución y el lado derecho de la ecuación y realiza el número de pasos de relajación especificado.

En el caso de la relajación de coordenadas para la solución de autovalores la

función es más compleja, debido a que se hace de manera simultánea para todos los autovalores, por lo que la maneja el método `EeX::RQMgEigenSolver::crelax` o `EeX::RQMgEigenSolverMV::crelax`.

4.4.4. Almacenamiento de grillas en disco

Para almacenar las grillas en disco se generó un formato propio de archivo denominado `grd`. Cada archivo `grd` tiene un encabezamiento de texto, seguido por una cadena binaria que contiene los datos comprimidos de la grilla. El encabezamiento está compuesto de varios strings seguidos por su valor:

- La primera línea contiene la cadena `EEXGRD`.
- `VERSION` y un entero con el número de versión; en este caso 1.
- `GRIDSIZES` y tres enteros que corresponden a los tamaños de la grilla en las direcciones x, y y z.
- `TOTALSIZE` seguido por el número total de puntos en la grilla.
- `SPACING` y el valor de la separación de la grilla.
- `DATAFORMAT` el formato en que están almacenados los valores; puede ser `FLOAT`, `DOUBLE` o `INT` (actualmente sólo se usa `FLOAT`).
- `SCALEFACTOR` y un valor real de escalamiento, por el que deben ser multiplicados los valores para obtener los valores originales.
- `DATASIZE` El tamaño en *bytes* de la cadena de datos original.
- `COMPMETHOD` El método de compresión utilizado, que puede ser `RAW` (sin compresión), `GZ` o `BZ2` (de momento sólo está implementado este último).

- `COMPSIZE` El tamaño de la cadena de datos comprimida.
- `COMPFACTOR` El factor de compresión; 1.0 corresponde a sin compresión.

La línea siguiente es la cadena de datos, almacenados en forma binaria y comprimidos. Para reducir el espacio utilizado, los valores son almacenados como un arreglo de `float` (mientras que en el código los datos se manejan como `double`). Para minimizar el error asociado a la representación con menor precisión, todos los valores se reescalan, dividiéndolos por el valor promedio de la función sobre la grilla, lo que idealmente hará todos los valores cercanos a 1. Como los valores se guardan de forma binaria sin considerar su representación, estos podrían no tener sentido en otras plataformas. El objetivo de este formato es ser como base para el traspaso a otros formatos mas portable y el almacenamiento en disco de las grillas para que puedan ser releidas por el codigo, por ejemplo en el caso de implementar un sistema de *checkpointing* que permita al codigo retomar un calculo que fue interrumpido.

4.5. Ambiente de desarrollo

En esta sección se describen las herramientas utilizadas para desarrollar el código.

4.5.1. Compiladores

En el desarrollo se usó el compilador `G++ 3` sobre plataforma `x86`. Para garantizar una mayor portabilidad se usó el *flag* `-pedantic`, que hace que el compilador se apegue al estándar ANSI C++. Sin embargo, es posible que existan problemas con compiladores antiguos que no implementen el estándar actual de C++ o que no tengan soporte completo para *templates*.

4.5.2. Sistema Operativo

Como plataforma de desarrollo se usó el sistema operativo Debian GNU/Linux, de modo que se espera que el código compile y corra sin problemas en otros sistemas Linux y Unix.

Es posible también compilar una versión estática para Windows mediante el uso del compilador Mingw32.

4.5.3. Bibliotecas externas

Para hacer el código lo más portable posible, en su mayor parte se usan funciones estándares de C, C++; las excepciones son detalladas a continuación.

Blas y Lapack

BLAS (Basic Linear Algebra Subprograms) [37] es una serie de subrutinas de Fortran de operaciones entre vectores y matrices. Se utilizó para realizar algunas de las operaciones vectoriales entre grillas.

LAPACK (Linear Algebra PACKage) [38] son varias rutinas de álgebra lineal basadas en BLAS, pero que realizan operaciones más complejas que ésta; en el código se utilizó LAPACK para diagonalizar la matriz densa que aparece dentro de la proyección de Ritz.

La ventaja de estas bibliotecas es que son estándares y están disponibles en cualquier sistema, muchas veces en versiones optimizadas provistas por el fabricante del hardware. En este trabajo se usó una implementación de BLAS y parte de LAPACK llamada ATLAS (Automatically Tuned Linear Algebra Software), que está disponible libremente y que se optimiza automáticamente al sistema en que está siendo ejecutada, obteniendo a veces rendimientos superiores a las bibliotecas nativas.

Vis5D

Vis5D es un programa de visualización de grillas en tres dimensiones y que posee un formato propio de archivos llamado v5d. Para compilar el código es necesario tener las bibliotecas de Vis5D instaladas, que proveen una serie de funciones para generar este tipo de archivos.

Bzip2

Para almacenar las grillas de datos en disco, estas se comprimen usando el formato bzip2. Para esto es necesario la biblioteca libbzip2.

4.6. Uso del código

El programa al correr busca 2 subdirectorios en el directorio en que se está ejecutando: `input/` y `output`. La informacion de entrada es controlada por el archivo `input/eex.par`. Todos los archivos de salida son colocados en el directorio `output` se colocarán todos los archivos de salida.

Los detalles sobre el uso del codigo y el formato de `input/eex.par` se pueden encontrar en el manual del programa en el apéndice A.

Actualmente, como salida el código entrega los valores de las energía, la densidad y los autovalores. El calculo de otros observables que dependan de la densidad puede ser implementado a futuro.

4.6.1. Visualización

Para poder visualizar las grillas, tanto para fines de programación y depuración, como para después obtener resultados físicos es necesario poder visualizar los valores contenidos en una grilla, para esto se usaron varias formas.

La primera es hacer una proyección de la función sobre un plano o una línea, sobre el cual se puede visualizar la función como un gráfico 2D o 3D. Esto permite ver de forma detallada el comportamiento de la función pero no se la puede visualizar de forma completa. Para generar estas vistas, creamos un archivo de texto conteniendo los datos dispuestos como una matriz. Estos posteriormente pueden ser visualizados en `gnuplot` obteniendo un gráfico como en la figura 4.1.

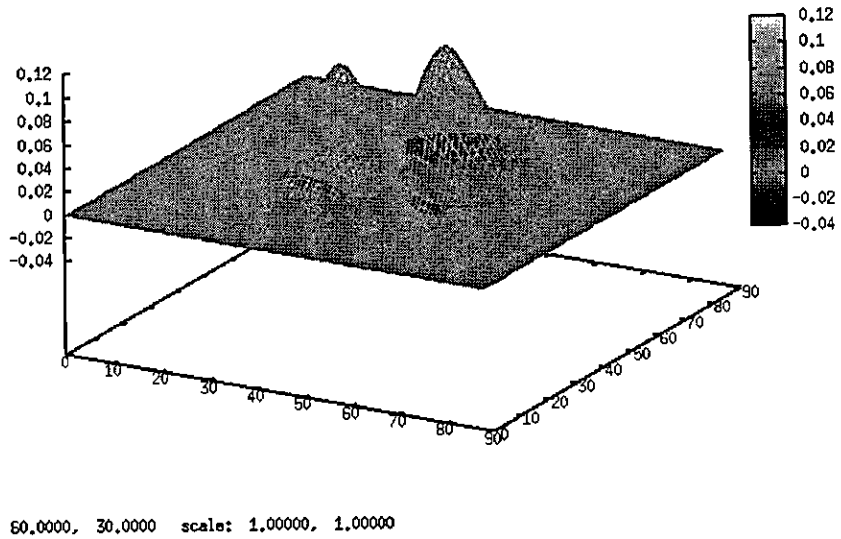

Figura 4.1: Visualizacion de una función sobre el corte dado por un plano

La visualizacion completa de la función tridimensional es mas compleja, para esto se uso el formato `v5d` del programa `Vis5D`, el que permite mostrar simultaneamente varias grillas, rotar, ver isosuperficies, cortes codificados por color, etc. Una captura de pantalla se muestra en la figura 4.2.

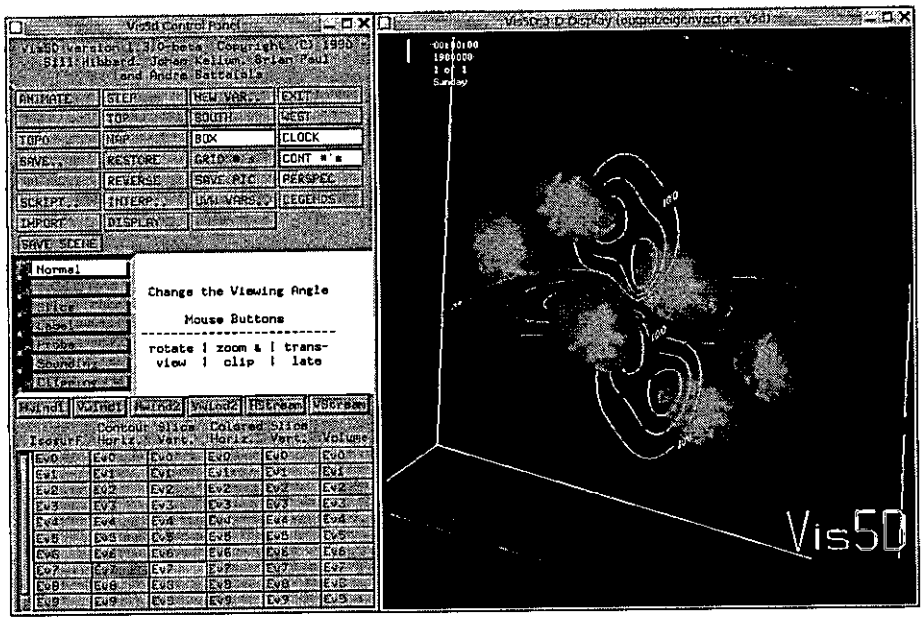

Figura 4.2: Captura de pantalla del programa Vis5D

Capítulo 5

Pruebas

5.1. Prueba del método de solución de autovalores

Antes de realizar cálculos autoconsistentes, se estudiaron la precisión y las propiedades del método de solución de autovalores. Para ello consideramos la ecuación de Schröndinger para un átomo de Hidrógeno, ya que este caso tiene la ventaja de que es un problema similar a los que tendremos que resolver dentro de un problema de Kohn-Sham y existe solución analítica para los valores de la energía y las funciones de onda.

5.1.1. Influencia de la condiciones de borde

Como condiciones de borde para las funciones de onda, se exige que se anulen sobre una caja de tamaño finito, a diferencia del caso analítico, donde las funciones se anulan en infinito. Para estudiar el efecto de esta aproximación, se calculó la energía total, manteniendo fijo el espaciado de la grilla, pero aumentando el tamaño de la caja. Los resultados para los orbitales 1s, 2s y 2p se pueden observar en la figura 5.1: hay un valor por sobre el cual el tamaño de la caja no influye, mientras menor sea la caja bajo este valor aumenta rápidamente la energía por sobre el valor analítico.

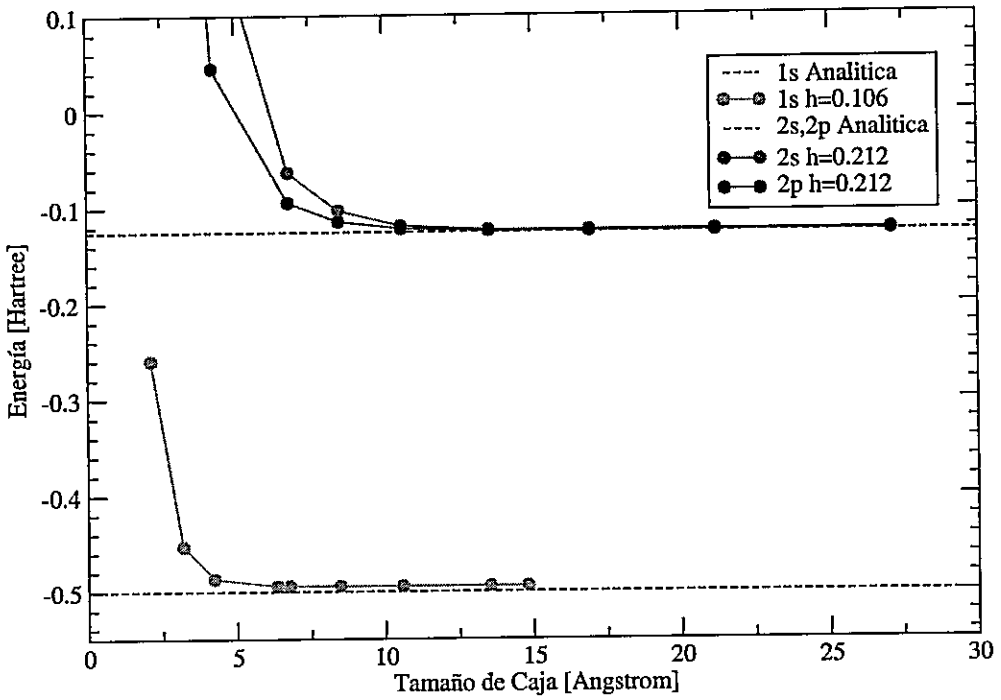

Figura 5.1: Valores de energía de los orbitales 1s, 2s y 2p para un átomo de Hidrógeno (no DFT) en función del tamaño de caja. Se usó la discretización MV con un espaciado fijo, pero distinto para cada orbital; los valores están indicados en el gráfico.

5.1.2. Calidad de la discretización

La precisión con que se discretiza la ecuación diferencial está determinada por la discretización del laplaciano y por la separación de la grilla. Para ver cuánto influye esta separación en el valor de energía, en el mismo problema anterior, se mantuvo fijo el tamaño de caja y se varió el número de puntos en la grilla. Los resultados para los orbitales 1s, 2s y 2p se encuentran en las figuras 5.2 y 5.3. Se puede ver que ambos métodos, MV y CDS 13 alcanzan valores similares de energía cuando la grilla es fina, sin embargo, al aumentar el espaciado, CDS 13 entrega valores mucho más cercanos al valor exacto que MV.

En la figura 5.4 se grafica un corte unidimensional de la función de onda de 1s para el estado fundamental y se comparan los valores para grillas de distinto grosor

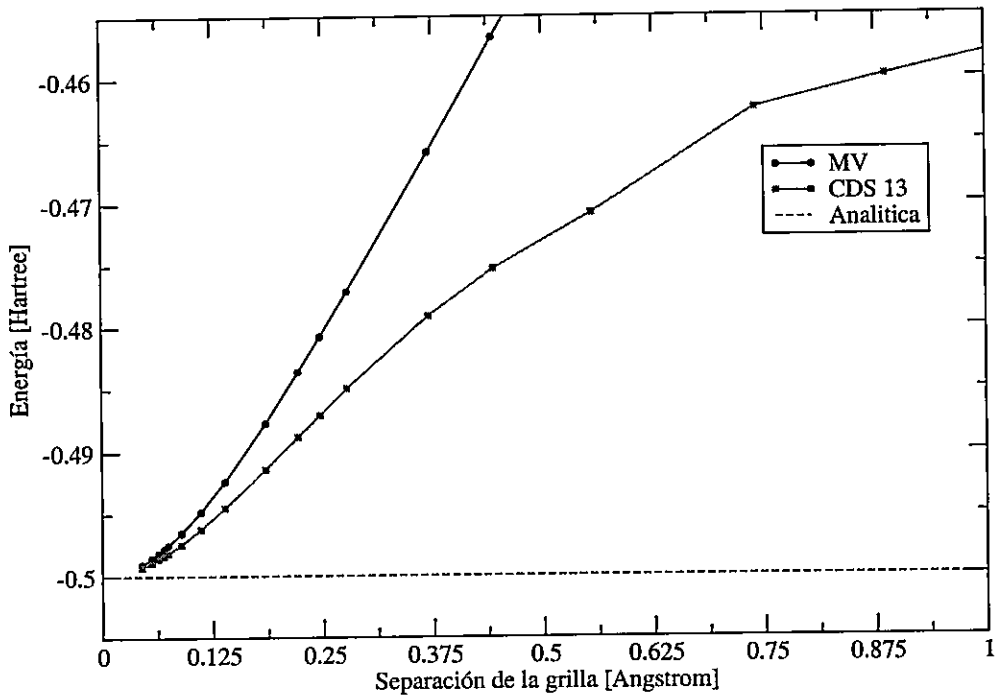

Figura 5.2: Energía de orbital 1s para un átomo de Hidrógeno (no DFT) en función de la separación de la grilla, para discretizaciones de Mehrstellenverfahren y segunda derivada discreta a orden 13. El valor analítico de la energía es -0.5 Hartree (linea segmentada). El tamaño de caja es de 7.938 Angstrom.

y el valor analítico. Se puede observar que es alrededor del centro de la grilla donde difieren las funciones de onda discretas con respecto al resultado exacto, y que la diferencia se hace mayor cuando la grilla tiene menos puntos.

5.2. Átomos individuales

La primera comprobación del código fueron los resultados para un átomo aislado. El esquema del código, basado en una grilla cúbica uniforme, es bastante ineficiente para este tipo de problemas, ya que en estos sería posible aprovechar la simetría radial del potencial y reducir el problema a una ecuación unidimensional con la parte angular descrita por armónicos esféricos.

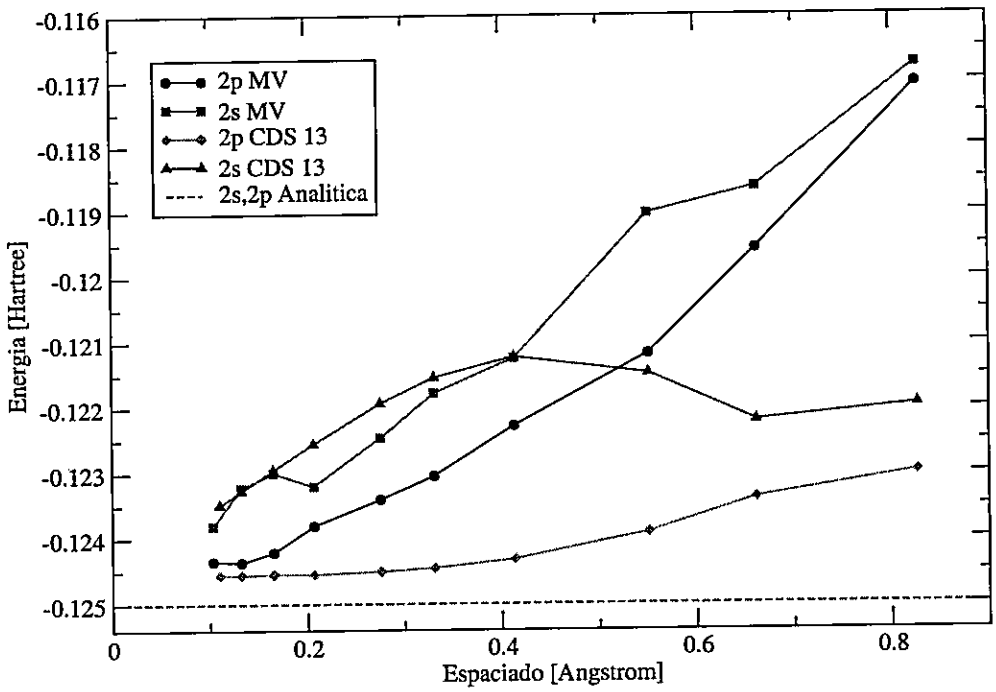

Figura 5.3: Energía del orbitales 2s y 2p para un átomo de Hidrógeno (no DFT) en función de la separación de la grilla, para discretizaciones de Mehrstellenverfahren y Segunda Derivada Discreta a orden 13. El valor analítico de la energía es -0.125 Hartree. El tamaño de caja es de 13.229431 Angstrom.

La ventaja de este caso es que existen valores de referencia muy precisos, con los que se puede verificar que el código esté funcionando de manera correcta. Los valores de referencia utilizados fueron obtenidos por Kotochigova, Levine, Shirley, Stiles, y Clark (KLSSC) [34] y pueden ser encontrados en [35].

Considerando los resultados de la sección 5.1, se debe encontrar el tamaño óptimo de la caja para realizar el calculo de manera precisa; para ello se fijó la separación de la grilla en 0.1 $\left[\text{Å}\right]$ y comenzó a aumentarse el tamaño de la caja, mientras se calcula la energía, hasta que ésta no cambia. Un tamaño de caja menor al óptimo fuerza las funciones de onda a cero, de forma no natural, haciendo mayor la energía, mientras que una caja más grande implica calcular espacio vacío que no influirá en

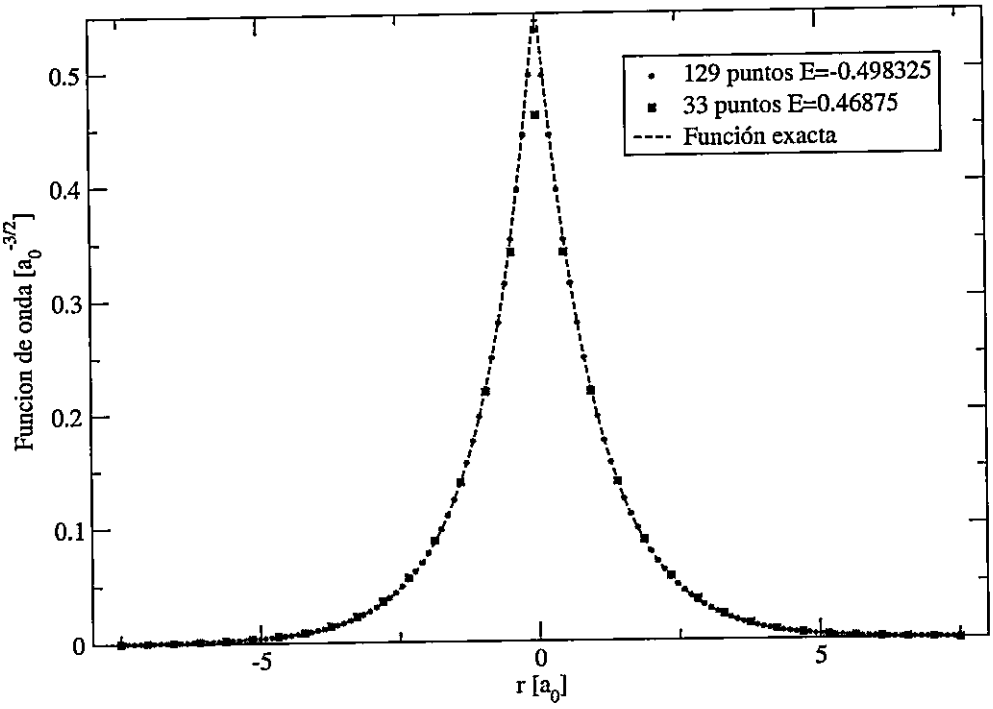

Figura 5.4: Corte en el eje x de la función de onda 1s de un átomo de Hidrógeno (no DFT), comparación de onda calculada para grillas de 129 y 33 puntos por lado con la función exacta $\phi(r,\theta,\phi) = \frac{1}{\sqrt{\pi}}e^{-r}$ (en unidades atómicas). Se usó una discretización CDS 13 y una caja de tamaño 7.93 Angstrom.

los resultados.

Teniendo el tamaño de caja óptimo y calculamos la energía total varias veces, disminuyendo en cada cálculo el espaciado de la grilla, hasta que la energía no cambie. Los resultados que obtuvimos se comparan con los valores de referencia en la tabla 5.1, en general los valores difieren en menos de un 1 %.

5.3. Moléculas

5.3.1. CO_2

Para una molécula de CO_2, se compararon los valores con los resultados al usar Gaussian [40]. Gaussian es un código que utiliza una base de funciones gaussianas,

Elemento	Z	Energía Obtenida [H]	Energía Referncia [H]	Diferencia %
H	1	-0.4786	-0.4787	0.009
He	2	-2.834	-2.835	0.04
Li	3	-7.352	-7.344	0.1
Be	4	-14.47	-14.45	0.1
B	5	-24.45	-24.35	0.4
C	6	-37.66	-37.47	0.5
N	7	-54.43	-54.14	0.5
O	8	-75.09	-74.53	0.7
Ne	10	-128.80	-128.23	0.4

Tabla 5.1: Comparación de los valores obtenidos por este trabajo y los valores de referencia para átomos individuales.

muy usado en Química cuántica. En ambos casos en el cálculo se consideraron todos los electrones, con espines apareados y se usó LDA para el funcional de intercambio y correlación.

Al buscar el mínimo de energía con respecto el largo del enlace C-O, usando Gaussian y suponiendo una configuración lineal, con el C al medio y los dos O equidistantes, se obtuvo como resultado 1.16 $\left[\text{Å}\right]$.

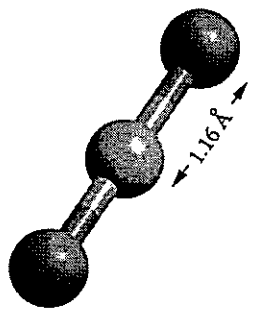

Figura 5.5: Geometría utilizada de la molécula de CO_2.

En nuestro código el tamaño de caja es 7.92 $\left[\text{Å}\right]$, en una grilla de 89 puntos por lado, con lo que el espaciado es de 0.9 $\left[\text{Å}\right]$. La molécula se colocó de forma diagonal

dentro de la caja, haciendo coincidir las posiciones de los 3 átomos con puntos de la grilla. El laplaciano fue aproximado con la fórmula CDS 13. En Gaussian se usó la base 6-311++g*.

En la tabla 5.2 se encuentra la comparación de las energías obtenidas mediante Gaussian y nuestro código, y en la tabla 5.3 se detallan los autovalores de Kohn-Sham. En la figura 5.6 se muestran los orbitales de Kohn-Sham obtenidos.

Energías	Este trabajo	Gaussian
Total	-188.8160	-187.6839
Cinética	186.1180	186.5932
Ion-Electrón	-561.7800	-559.3381
Hartree+XC	128.4524	126.6691

Tabla 5.2: Energías obtenidas para CO_2 por el código desarrollado en este trabajo y Gausssian.

Autovalores	Este trabajo	Gaussian
ε_0	-18.61410	-18.70624
ε_1	-18.61410	-18.70622
ε_2	-9.98016	-9.98172
ε_3	-1.11119	-1.09527
ε_4	-1.07726	-1.06107
ε_5	-0.53283	-0.52768
ε_6	-0.48546	-0.49909
ε_7	-0.48301	-0.49909
ε_8	-0.48240	-0.49239
ε_9	-0.34544	-0.36245
ε_{10}	-0.34540	-0.36245

Tabla 5.3: Comparación de los autovalores de Kohn-Sham para CO_2 por dos códigos: el de este trabajo y Gausssian.

Usando los dos códigos, calculamos la energía de formación del CO_2, restando a la energía obtenida las energías individuales de cada átomo. Los resultados se pueden ver en la tabla 5.4.

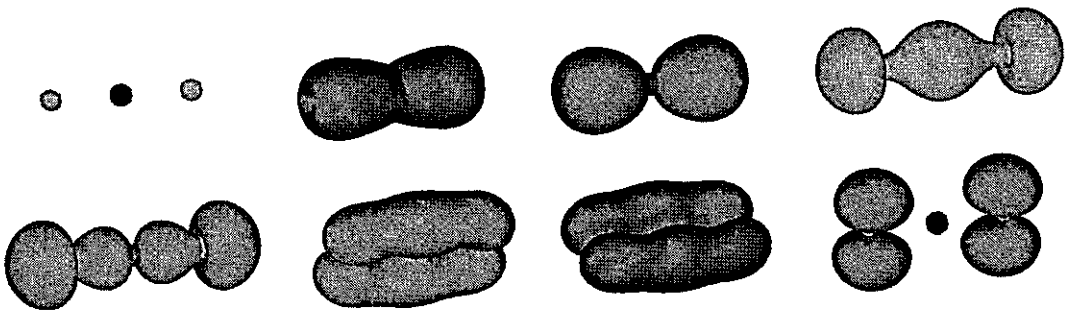

Figura 5.6: Superficies de densidad constante para los orbitales de la molécula de CO_2.

Energías	Este trabajo	Gaussian
$E(CO_2)$	-188.816	-187.684
E(C)	-37.676	-37.508
E(O)	-75.052	-74.571
$E(CO_2)$-2E(O)-E(C)	-1.0362	-1.0348

Tabla 5.4: Energía de formación para CO_2, calculada por el código de este trabajo y por Gausssian. Las energías de C y O en ambos códigos se calcularon con *spin* apareado.

5.3.2. Molécula de Hidrógeno

Se buscó el largo del enlace de la molécula de H_2. Se fijó el tamaño de caja y la separación de la grilla, $h = 0.07$, de acuerdo a los valores óptimos obtenidos para el hidrógeno. Después se calculó la energía para distintos largos de enlace, obteniendo la curva mostrada en figura 5.7; todos estos largos de enlace son múltiplos de h, para que los átomos calcen con puntos de la grilla. El valor mínimo se encontró interpolando un polinomio cúbico en los puntos alrededor de éste.

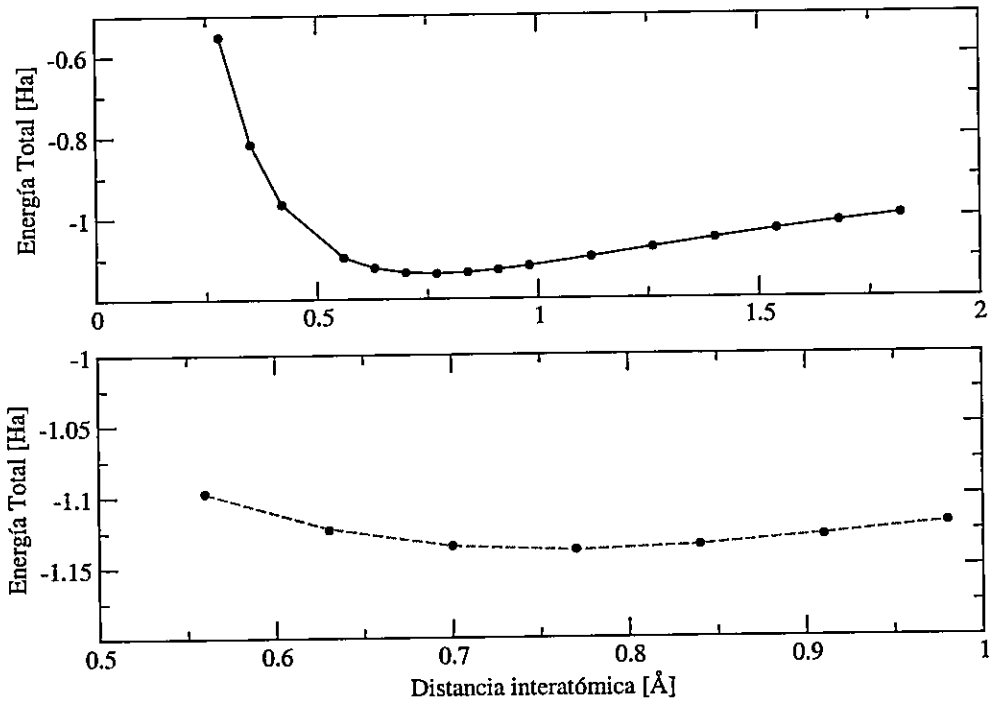

Figura 5.7: Energía de la molécula de H_2 en función del largo del enlace. En el segundo gráfico se detalla la interpolación cúbica usada para encontrar el mínimo.

Una vez obtenido el largo del enlace se calcula el valor de energía correspondiente. Estos valores se comparan en la tabla 5.5 con los valores experimentales. En la figura 5.8 se puede observar un gráfico de la densidad electrónica obtenida.

Estos valores se encuentran dentro de lo esperado para la aproximación LDA, que

	Experimental	Este trabajo
Largo del enlace	0.77 [Å]	0.764 [Å]
Energía de formación	-0.1742 [Ha]	-0.1804 [Ha]

Tabla 5.5: Comparación entre los valores obtenidos y los datos experimentales para H_2

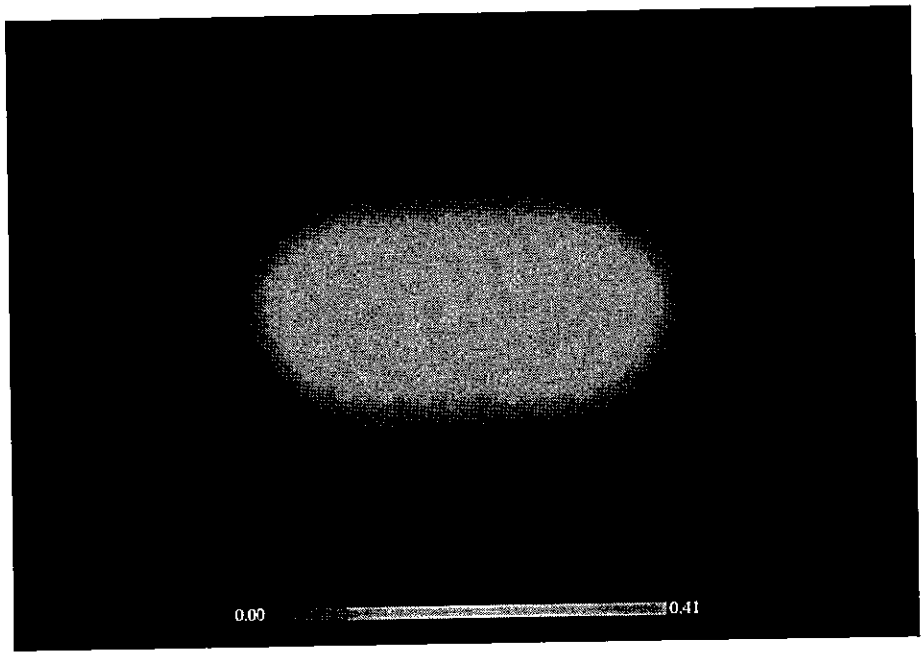

Figura 5.8: Densidad electrónica de la molécula de Hidrógeno

es sabido que entrega largos de enlace menores y energías de formación mayores con respecto a los valores experimentales [8].

5.3.3. B_2H_6

Se estudió el caso de una molécula de *Diborano*, cuya geometría se muestra en la figura 5.9.

En este caso el cálculo presenta una complejidad adicional, debido a su geometría más compleja, que impide hacer calzar todos los átomos con puntos de la grilla. En estos casos el código trata los átomos como si se encontraran en el punto de la grilla más cercano a su posición original. Para minimizar este efecto se buscaron los valores

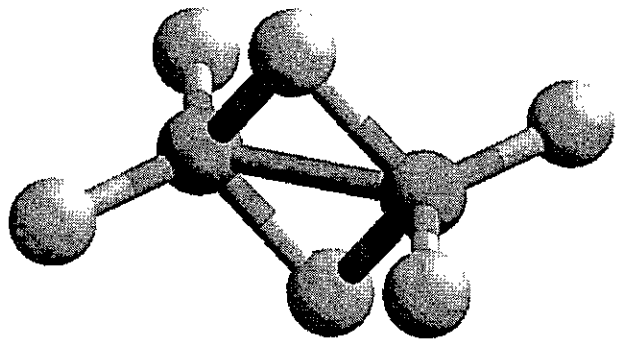

Figura 5.9: Molécula de B_2H_6

de h que producen el mínimo desplazamiento de los átomos. Para cada valor de h se sumaron los desplazamientos de cada átomo, obteniéndose el gráfico 5.10. En él se ve que el desplazamiento total oscila rápidamente, cambiando en hasta 10 veces su valor, por lo que es importante escoger un h óptimo. El valor escogido fue de $h = 0.0658482\,[\text{Å}]$, que produce un desplazamiento total de $0.053585\,[\text{Å}]$, lo que resulta en que el enlace B-B varía en $0.02\,[\text{Å}]$ y los enlaces B-H cambian en menos de $0.01\,[\text{Å}]$.

Al igual que para el CO_2 compararemos los resultados con Gaussian. En la tabla 5.6 se encuentrar los resultados de energías totales, en la tabla 5.7 se encuentras las energías de formación para esta molécula calculadas por ambos códigos. Nuevamente son muy similares los resultados obtenidos por el programa desarrollado en este trabajo con Gaussian en cuanto a diferencia de energía se refiere. En la figura 5.11 se muestras los orbitales de Kohn-Sham obtenidos.

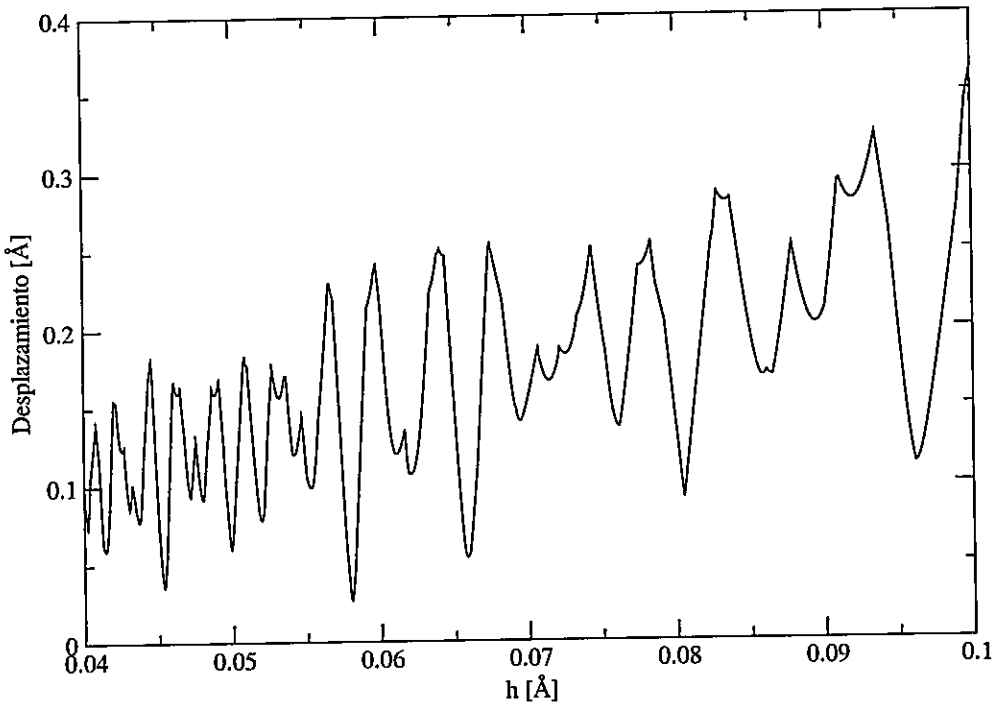

Figura 5.10: Desplazamiento total de los átomos en función de la separación de la grilla h.

Energías	Este trabajo	Gaussian
Total	-52.8091	-52.9387
Cinética	52.5227	52.1524
Ion-Electrón	-185.3774	-184.7183
Hartree+XC	47.8459	47.4282

Tabla 5.6: Energías obtenidas para B_2H_6 por el código desarrollado en este trabajo y Gausssian.

Energías	Este trabajo	Gaussian
$E(B_2H_6)$	-52.81	-52.94
E(H)	-0.504	-0.496
E(B)	-24.38	-24.44
$E(B_2H_6)$-6E(H)-2E(B)	-1.03	-1.07

Tabla 5.7: Energía de formación para B_2H_6, calculada por el código de este trabajo y por Gausssian.

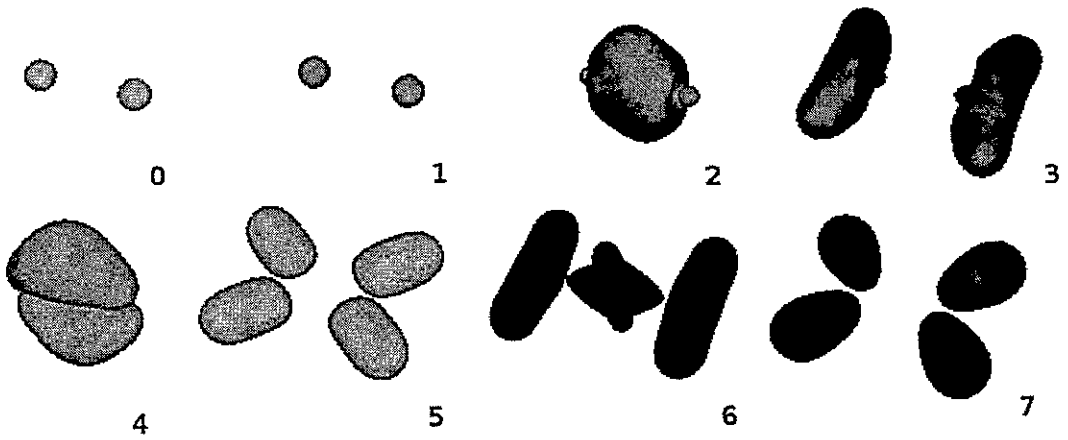

Figura 5.11: Superficies de densidad constante para los orbitales de la molécula de B_2H_6.

Capítulo 6

Conclusiones

Se diseñó una estrategia de solución de las ecuaciones de Kohn-Sham en el espacio real. Al igual que en otros métodos se usó un ciclo iterativo para encontrar soluciones autoconsistentes a las ecuaciones de Kohn-Sham. Se utilizó la aproximación LDA al funcional de intercambio y correlación, sin embargo el método es completamente general y permite la posible incorporación de otros funcionales más exactos.

Se implementaron los algoritmos necesarios para tratar las distintas partes del método de solución; para la parte central, encontrar los autovalores, se usó una versión de *multigrid* conocida como *Rayleigh Quotient Multigrid*, que resulta eficiente y exacta dentro de la precisión dada por la discretización.

Se programaron todos los algoritmos y el método de solución en un esquema de orientación al objeto, logrando una alta modularización del código, pero sin grandes sacrificios de rendimiento, evitando realizar muchas llamadas a funciones muy cortas mediante el uso de *templates* y funciones *inline*. Además muchos de los componentes del código son intercambiables, por lo que permiten la implementación por separado de componentes alternativos más eficientes o con otras características.

Finalmente se realizaron pruebas sobre el código para determinar sus características y limitaciones. Se encontró que existen varios parámetros que deben ser conside-

rados, como el tamaño de caja o el espaciado de la grilla, que hacen que el resultado varíe considerablemente para un mismo sistema físico, por lo que se diseñó una metodología sistemática para determinar los parámetros óptimos para una simulación, lo que permite obtener resultados más precisos y consistentes con el mínimo costo computacional.

Teniendo en consideración estos factores, fue posible conseguir resultados que concuerdan con los obtenidos por otros códigos ampliamente usados y con valores experimentales, sobre todo en términos de diferencias de energía, que al igual que en otros esquemas, resultan ser más precisos que los cálculos de energía total.

Esta primera versión del programa sienta las bases para el desarrollo de un código de nivel de producción, para esto es necesaria la incorporación de muchas mejoras e optimizaciones así como de varias características adicionales. Por ejemplo, se encontró que el programa tiene una limitación respecto al tamaño máximo de los sistemas que permite tratar debido a la grilla uniforme y a que se tratan todos los electrones del sistema. Por lo que un factor primordial para llevar el código a un nivel de producción es incorporar mejoras que permitan manejar sistemas más grandes y con más electrones de los que es posible manejar actualmente. Existen al menos dos estrategias para hacer esto. Por una parte sería interesante y novedoso el uso de grillas adaptativas o refinamientos locales que mejoren la resolución de la solución en las zonas cercanas a los núcleos, donde es necesaria una mayor precisión, sin aumentar considerablemente el costo computacional o el uso de memoria. Por otra parte una estrategia más simple es el uso de pseudopotenciales que reemplacen el potencial nuclear y los electrones del *core*, por un potencial suave permitiría aumentar el tamaño de los sistemas accesibles, al necesitarse un espaciado de grilla menor y disminuir el número de funciones de onda que es necesario manejar.

Otro aspecto importante es la incorporación al código de los funcionales de intercambio y correlación mas avanzados, conocidos como GGA, que en muchos casos entregan resultados físicos mas exactos que LDA. Estos nuevos funcionales puede ser facilmente incorporado en el código debido a su naturaleza modular.

Incluso una aplicación importante del código podría ser la incorporación del calculo de la energía de intercambio y correlación mediante funcionales dependientes de los orbitales que son mucho más precisos que los funcionales dependientes de la densidad. Sin embargo, el cálculo del potencial de intercambio y correlación con estos funcionales es más complejo, ya que es necesario resolver una ecuación integral, la cual puede ser resuelta de iterativamente de forma similar a los esquemas utilizados en este trabajo.

Apéndice A

Manual de EeX

A.1. Introducción

EeX es un código de cálculo de estructura *ab-initio* basado en la teoría del funcional de la densidad. Este utiliza un formalismo en el espacio real, que no usa una base, sino que representa las funciones sobre grillas, y que para resolver las ecuaciones de autovalores utiliza el método de *Multigrid.*

Aunque **EeX** es un código completamente funcional, es un trabajo en progreso, por lo que aún le faltan muchas características. Las funciones que actualmente realiza son:

- Puede hacer cálculos de energía y densidad electrónica para una configuración fija de átomos.
- Maneja los funcionales de intercambio y correlación LDA y LSDA.
- Realiza cálculos *all electron.*
- Resuelve sistemas con *spin* apareado o con polarización de *spin.*
- Sólo maneja sistemas aislados, sin usar superceldas.

A.2. Compilación

A.2.1. Requisitos

Arquitectura

El programa no contiene código dependiente de la arquitectura, por lo que se espera que pueda compilarse y ejecutarse sobre cualquier plataforma que implemente instrucciones de punto flotante IEEE. El programa ha sido probado tanto en x86 como en Alpha.

Compilador

Para compilar **EeX** se requiere un compilador ANSI C++. El código ha sido probado con los siguientes compiladores:

- Gnu C++ Compiler versiones 2.95 y 3.X (con X>2)
- Intel Compiler versiones 7 y 8

Se recomienda usar g++ 3.2 o superior.

Lapack y Blas

Para compilar el código se requiere de las bibliotecas Lapack y Blas. Sin embargo, no son necesarias versiones altamente optimizadas (aún). Recomendamos usar la version de referencia de Lapack (http://www.netlib.org/lapack) en conjunto con ATLAS (http://math-atlas.sourceforge.net).

Vis5d

Vis5D (http://vis5d.sourceforge.net) es un programa de visualización de grillas en tres dimensiones que tiene un formato propio de archivos llamado v5d. Para generar

este tipo de archivos Vis5D provee una serie de funciones, por lo que es necesario tener las bibliotecas de Vis5D instaladas para compilar **EeX** . El uso de Vis5D es controlado por el flag `-DEEX_HAVE_VIS5D`.

Bzip2

Se requiere la versión de desarrollo de la biblioteca `libzip2` para poder generar archivos de grillas comprimidos. Es una biblioteca disponible para casi todos los sistemas Unix.

A.2.2. Configuración

Archivo de configuración

Para construir código es necesario adecuar modificar los parámetros de compilación adecuados al sistema en que está ejecutando. Todos estos parámetros se encuentran en un archivo específico para cada máquina, cuyo nombre comienza con `Make.inc.` seguido de un *string* que identifica al sistema. Las opciones que vienen incluidas son:

- g++3-x86 G++ versiones 3.2 y superiores en Linux sobre x86.
- g++-2.95 G++ version 2.95 en Linux sobre x86.
- g++3-alpha G++ versiones 3.2 y superiores en Linux sobre Alpha.
- intel-7 Compilador Intel versión 7 en Linux sobre x86.
- intel-8 Compilador Intel versión 8 en Linux sobre x86.
- win32 Compilador mingw32 sobre Linux para producir un binario para Windows.

Es posible generar nuevos archivos, simplemente copiando uno de los existentes a un nuevo nombre, con la misma forma.

Macros de configuración

Existen varias macros que permiten adaptar el código al entorno de ejecución y/o compilación. Estas se detallan a continuación:

- EEX_HAVE_VIS5D Activa el soporte de formato v5d, requiere las bibliotecas de Vis5D.
- EEX_USE_GCC_BUILTINS Utiliza ciertas funciones de optimización de GCC; también pueden estar soportadas por otros compiladores.
- EEX_NO_STD_MAX_MIN El compilador no provee `std::max` ni `std::min`, por lo tanto se usa una definición alternativa.
- EEX_NORAND48 Usa `rand` en vez de `drand48`.
- EEX_CPUTIME Mide tiempo de cpu en vez de tiempo real, por lo que es útil en sistemas que no proveen `gettimeofday` pero si `clock`. Esta macro es necesaria para Win32.

A.2.3. Compilación

Para compilar el código basta dar `make CONF=plataforma`, donde plataforma es un *string* que coincide con el archivo `Make.inc.plataforma`, como fue descrito anteriormente. Si la compilación es exitosa se generará el ejecutable del programa llamado `eex`.

A.3. Uso del código

Al correr el programa busca 2 subdirectorios, llamados `input/` y `output/` en el directorio en que se está ejecutando: `input/` y `output`. En el primero busca el archivo de entrada `eex.par`, donde se le indican los parámetros para la simulación. En el directorio `output` colocará todos los archivos de salida.

A.3.1. Archivo de entrada

Al ejecutarse, **EeX** busca el archivo de `input/eex.par` que contiene los parámetros para la ejecución del programa. Este es un archivo de texto que en cada línea, contiene el nombre del parámetro seguido por su valor. En caso de que el código no encuentre el archivo de entrada, preguntará al usuario si desea generar un archivo con valores por defecto. En caso de que la respuesta sea afirmativa, se generará un archivo `input/eex.par` y el programa terminará.

Los parámetros tienen nombres en mayúsculas, y no pueden ser escritos en minúsculas. No necesitan estar en un orden específico; las líneas comenzadas con un signo # son comentarios y se ignoran todas las líneas que no coincidan con ningún parámetro esperado. Todos los parámetros de entrada usan las mismas unidades: Angstrom para distancias y Hartree para energías.

A continuación, se muestra un ejemplo de archivo de entrada, el cual corresponde al archivo generado por defecto por **EeX** .

```
#ION_POS_FILE: Archivo XYZ con las posiciones de los iones
ION_POS_FILE input/iones.xyz

#CHARGE: Carga electrica total del sistema (0 sistema neutro)
CHARGE 0

#SPIN_MULT: Spin multiplicity
SPIN_MULT 1

#EXTRA_EV: Numero de autovalores extra que calcular
EXTRAEV 0
```

```
#BOX_SIZE: Tamano del dominio de solucion (si es negativo se usa SPACING)
BOX_SIZE -1

#SPACING: Espaciado de la grilla en Angstrom (si es negativo se usa BOX_SIZE)
SPACING -1

#SCI_LEVELS: Niveles para la iteracion autoconsistente
SCI_LEVELS 2

#POINTS_PER_SIDE: Puntos por lado del grilla mas fina
POINTS_PER_SIDE 65

#LAP_TYPE: Tipo de la expansion del laplaciano, CDS o MV)
LAP_TYPE CDS

#LAP_POINTS: Numero de puntos en la expansion del laplaciano (numero impar)
LAP_POINTS 13

#SIGMA2: Factor de la gaussiana para la distribucion de carga
#(negativo se acerca al punto mas cercano)
SIGMA2 -1

#INT_FORMULA: Rule for numerical integration,TRAPEZOIDAL or SIMPSON
INT_FORMULA TRAPEZOIDAL

#SELF_MAX_STEPS: Numero maximo de pasos de iteracion de autoconsistencia
SELF_MAX_STEPS 30

#SELF_TOL: Tolerancia en la energia para terminar la iteracion de autoconsistencia
SELF_TOL 0.0001

#MIXING: Nivel de mezcla entre el antiguo y el nuevo potencial (1.0 es solo el nuevo)
MIXING 0.5

#POISSON_MAX_STEPS: Numero maximo de ciclos de multigrid para resolver la ecuacion de Poisson
POISSON_MAX_STEPS 20

#POISSON_TOL: Tolerancia de la ecuacion de Poisson
POISSON_TOL 0.0001

#POISSON_PRE_STEPS: Pasos de pre-relejacion por nivel para resolver Poisson
POISSON_PRE_STEPS 3

#POISSON_POST_STEPS: Pasos de post-relajacion por nivel para resolver Poisson
POISSON_POST_STEPS 3

#EIGEN_MAX_STEPS: Numero maximo de ciclos de multigrid para la ecuacion de autovalores
EIGEN_MAX_STEPS 2

#EIGEN_TOL: Tolerancia de la ecuacion de autovalores
EIGEN_TOL 0.0001

#EIGEN_PRE_STEPS: Pasos de pre-relejacion por nivel para los autovalores
EIGEN_PRE_STEPS 2

#EIGEN_POST_STEPS: Pasos de post-relajacion por nivel para los autovalores
EIGEN_POST_STEPS 2

#CR_WEIGHT: Peso para la separacion de autovectores
CR_WEIGHT 10
```

A.3.2. Parámetros de Entrada

Los parámetros de entrada que aparecen en `input/eex.par` son explicados en detalle a continuación. Cabe notar que no es necesario modificar todos los parámetros, para la mayoria basta dejar el valor por defecto.

Parámetros físicos

Estos parámetros regulan el sistema físico que se va a estudiar.

- ION_POS_FILE: Archivo XYZ con las posiciones de los iones. El formato es el siguiente: en la primera línea del archivo está el número de partículas, en la segunda una descripción (opcional) o una línea en blanco y a partir de la tercera línea las posiciones de los atomos, dadas por el símbolo del elemento seguidas de las coordenadas x, y y z. El valor por defecto es `input/iones.xyz`.

```
3
CO2 Bond Length = 1.16
C     0.0       0.0              0.0
O     0.54      0.63079315       0.81
O    -0.54     -0.63079315      -0.81
```

Figura A.1: Ejemplo de archivo XYZ.

- CHARGE: La carga total del sistema, a partir de la cual se calcula el número de electrones. Por defecto es 0, lo que implica un sistema neutro.

- SPIN_MULT: La multiplicidad de spin; en el caso que sea 1 se utiliza LDA y si es mayor se usa LSDA.

Parámetros de simulación

Estos parámetros regulan el nivel de la aproximación que se hará al sistema, y por consiguiente pueden variar la calidad del resultado y el tiempo de ejecución.

- EXTRA_EV: Número de autovectores extra que calcular por sobre los correspondientes al número de electrones. Por defecto es 0 autovectores extra. Por una parte esto permite reducir mejorar la convergencia de los últimos autovectores y/o conocer los autovalores de los estados no ocupados.

- BOX_SIZE: Tamaño en Angstrom del dominio de solución; este parámetro es excluyente con SPACING (sólo uno de los 2 puede estar determinado). Para deshabilitarlo debe tener un valor negativo o simplemente no aparecer en el archivo. Valor por defecto: -1.0.

- SPACING: Espaciado de la grilla en Angstrom, es excluyente con BOX_SIZE (sólo uno de los 2 puede estar determinado), para indeterminarlo debe tener valor negativo o no aparecer en el archivo. Por defecto tiene el valor de -1.0.

- POINTS_PER_SIDE: Puntos por cada lado del grilla más fina; el valor por defecto es 65. Este debe ser un número impar. Aumentar el número de puntos disminuye el espaciado o aumenta el tamaño de la caja (según lo que se haya especificado) pero aumenta cúbicamente el consumo de memoria y el tiempo de cálculo.

- LAP_TYPE: Tipo de la expansión del laplaciano a usar, CDS o MV (ver sección 3.2.4). Por defecto se usa CDS.

- LAP_POINTS: En el caso de usar un laplaciano CDS, el número de puntos en la expansión. El orden de la aproximación es LAP_POINTS−1. Por defecto el

valor es 13.

Parámetros de la iteración autoconsistente

Estos parámetros regulan la iteración de autoconsistencia. Se recomienda no modificar estos valores. En caso de que no haya convergencia puede ser necesario disminuir MIXING.

- SCI_LEVELS: Número de niveles en que se hace la iteración de autoconsistencia. El uso de varios niveles puede disminuir el tiempo de cálculo y modificarlo no varía el resultado. Por defecto son 2 niveles. En general usar más de 3 niveles no produce beneficios.

- SELF_MAX_STEPS: Número máximo de pasos antes de terminar la iteración de autoconsistencia si no se ha alcanzado la convergencia, el valor por defecto es de 30 pasos.

- SELF_TOL: Cuando la diferencia de autoenergías en una paso de autoconsistencia es menor a este valor, se considera que la iteración ha convergido. La diferencia entre energías se calcula como

$$\tau^{sc} = \left| \sum_{i=0}^{q-1} \epsilon_i^{new} - \sum_{i=0}^{q-1} \epsilon_i^{old} \right| , \tag{A.1}$$

  El valor por defecto de este parámetro es 1×10^{-4}.

- MIXING: Nivel de mezcla entre el antiguo y el nuevo potencial dentro de la iteración de autoconsistencia; un valor de 1.0 implica que sólo se usará el nuevo potencial. Por defecto el valor es 0.5, pero valores menores pueden necesitarse para asegurar la convergencia.

Parámetros de la solución de Poisson

Estos parámetros regulan el comportamiento del código de solución de la ecuación de Poisson por *multigrid*. Se recomienda dejar los valores por defecto.

- POISSON_MAX_STEPS: Número máximo de ciclos *multigrid* a realizar para resolver la ecuación de Poisson, en caso de que no se logre la convergencia. Por defecto 20.
- POISSON_TOL: Cuando el error de la ecuación de Poisson se hace menor que este valor, se considera que se ha encontrado la solución; el valor por defecto es 1×10^{-4}.
- POISSON_PRE_STEPS: Número de pasos de pre-relajación por nivel, dentro del ciclo de multigrid para resolver Poisson. Por defecto 3.
- POISSON_POST_STEPS: Número de pasos de post-relajación por nivel, dentro del ciclo de multigrid para resolver Poisson. Por defecto 3.

Parámetros de la solución de autovalores

Estos parámetros regulan el comportamiento del código de solución de la ecuación de autovalores por RQMG. Se recomienda mantener los valores puestos por defecto, salvo CR_WEIGHT que debe ser aumentado si no hay convergencia de los autovalores por sobre el estado fundamental.

- EIGEN_MAX_STEPS: Número máximo de ciclos de RQMG a realizar para resolver la ecuación de autovalores en caso de que no se logre la convergencia. Por defecto son sólo 2, debido a que la iteración de autoconsistencia no requiere autovectores totalmente convergidos, de modo que no conviene invertir mucho tiempo de cálculo en mejorarlos.

- EIGEN_TOL: Máximo valor del error para considerar un autovector como convergido; el valor por defecto es 1×10^{-4}.
- EIGEN_PRE_STEPS: Número de pasos de pre-relajación por nivel dentro del ciclo de RQMG. Por defecto 2.
- EIGEN_POST_STEPS: Número de pasos de post-relajación por nivel dentro del ciclo de RQMG. Por defecto 2.
- EIGEN_MAX_STEPS: Número máximo de ciclos de *multigrid* para la ecuación de autovalores. Valor por defecto -1.
- CR_WEIGHT: Peso ω para la separación de autovectores; por defecto el valor es 10 pero debe ser modificado de acuerdo a la magnitud del potencial iónico y la separación de la grilla. Por ejemplo, para Argon, que tiene $Z = 18$, se usó un valor de $\omega = 5000$.

A.3.3. Archivos de salida

El programa genera varios archivos de salida en el directorio `output/`, además de mostrar información en pantalla sobre la ejecución del programa.

`output/energy.log`

Este archivo contiene la información de la evolución de los autovalores de Kohn-Sham durante la iteración de autoconsistencia. La primera columna es el nivel en que se está resolviendo y la segunda el paso de autoconsistencia; en las filas siguientes están los autovalores de Kohn-Sham seguidos por el error que estos tienen, dados como potencias de 10.

`output/sciconv.log`

Este archivo contiene información acerca de la iteración de autoconsistencia a medida que se ejecuta el programa. La primera columna es el paso de la iteración, en las columnas siguientes está el cambio entre una iteración y la siguiente: en la segunda columna medido como el cambio en el potencial efectivo y en la tercera columna medido como el cambio en las autoenergías.

`output/results`

Este archivo contiene información sobre el cálculo y los resultados de energía. La estructura es la siguiente:

- Primero se detallan los componentes utilizados, los *solvers* de Poisson y de autovalores y el potencial de intercambio y correlación.
- A continuación se encuentra una sección por cada nivel de iteración autoconsistente: la más importante es la última sección, que corresponde a la mayor resolución. Cada nivel tiene las siguientes secciones:
  - Información sobre la grilla utilizada, la cantidad de puntos y el tamaño real.
  - Información sobre la iteración de autoconsistencia, el factor de mezcla, la cantidad de pasos realizados y cuánto cambiaron el potencial y los autovalores en la última iteración.
  - Las energías obtenidas: energía total, energía cinética, energía de Hartree, energía de la interacción con el potencial nuclear y la energía de intercambio y correlación.

- El cociente de la energía potencial respecto a la energía cinética (V/T), que de acuerdo al teorema del Virial debe valer -2.

- Finalmente se da el tiempo total de ejecución.

A continuación, se da un ejemplo de el archivo `output/results`:

```
POISSON EQUATION SOLVER:
Correction Scheme Multigrid Solver
* A. Brandt, Multi-level adaptive solutions to boundary value problems,
  Math. Comp. 31 333-390 (1977)

EIGENVALUE SOLVER:
Rayleigh Quotient MultiGrid
* Mandel, S. F. McCormick, A Multilevel Variational
  Method for $Au=\lambda Bu$ on Composite Grids,
  J. Comp.  Phys.80, pp. 442-452, (1989).
* Heiskanen, M., Torsti T., Puska, M. J., Nieminen, R. M.,
  A novel multigrid method for electronic structure calculations,
  xPhysical Review B 63, 245106 (2001).

EXCHANGE AND CORRELATION FUNCTIONAL:
LDA: Vosko Wilk Nusair parametrization of data from
Ceperley and Alder QM Calculations.
* S.H. Vosko, L. Wilk, and M. Nusair, Can. J. Phys. 58, 1200 (1980)
* S.H. Vosko and L. Wilk, Phys. Rev. B 22, 3812 (1980).


==================
  L E V E L  -  4
==================


GRID
----------------------
Size  33 x 33 x 33
TotalPoints = 35937
RealSize = 7
Spacing = 0.21875

SCI
----------------------
Mixing = 0.5
Steps = 13
EnergyDiff = 7.09365617229e-05
PotentialDiff = 0.000504336509236

ENERGY
----------------------
Etot  = -2.84448198234
Ekin  = 2.79009551967
Ecoul = 2.01258645948
Eenuc = -6.66591473042
Exc   = -0.981249231075

       -0.568340761717 -12
```

```
V/T    = -2.01949268844


==================
  L E V E L  -  5
==================


GRID
----------------------
Size  65 x 65 x 65
TotalPoints = 274625
RealSize = 7
Spacing = 0.109375

SCI
----------------------
Mixing = 0.5
Steps = 4
EnergyDiff = 3.35163486134e-05
PotentialDiff = 0.00742265514812

ENERGY
----------------------
Etot   = -2.83408261167
Ekin   = 2.77415331126
Ecoul = 1.99738132896
Eenuc = -6.6315602165
Exc    = -0.974057035379

        -0.569631230021 -7

V/T    = -2.02160273557

COMPUTATION
----------------------
TotalTime = 0h 1m 23.6585068703s
TotalSec = 83.6585068703
```

Archivos para visualización

EeX genera además varios archivos que permiten visualizar los resultados. Estos archivos vienen en 3 formatos [1]:

- `ldat` Contiene un corte unidimensional de la función en 2 columnas.
- `dat` Contiene un corte bidimensional de la grilla dispuesta como matriz. Estas pueden ser visualizados fácilmente con `gnuplot` usando el comando `splot`

[1] El formato cube de Gaussian será soportado dentro de poco.

`'archivo.dat' matrix`.

- `v5d` Estos archivos contienen la función 3D completa y pueden ser visualizados con el programa `Vis5D`.

Cuando el programa termina de correr, genera los siguientes archivos (en el caso con polarización de *spin* habra dos archivos de cada tipo, con sufijos `_up` y `_down`):

- `density` Contiene la densidad electrónica final.
- `ev`k Contiene el autovector k.
- `eigenvectors` Contiene todos los autovectores en un solo archivo v5d.

Referencias

[1] M. Born and R. Oppenheimer, *Ann. Phys.*, **84** 457 (1927)

[2] Hellmann, H., Emführung in die Quantumchemie (Deuticke, Leipzig) (1937).

[3] Feynman, R. P. *Phys. Rev.* **56** 340 (1939).

[4] P. Hohenberg and W. Kohn, *Phys. Rev. B* **136**, 864 (1964).

[5] W. Kohn and L. J. Sham, *Phys. Rev. A* **140**, 1133 (1965).

[6] S. Kümmel, Damped gradient iteration and multigrid relaxation: tools for electronic structure calculations using orbital density-functionals, *J. Comp. Phys.* **201**, 333-343 (2004).

[7] R. M. Dreizler and E. K. U. Gross, Density Functional Theory An Approach to the Many-Body Problem. *Springer-Verlag.* (1990).

[8] R. G. Parr and W. Yang, Density-Functional Theory of Atoms and Molecules. *Oxford University Press.* (1989).

[9] D. M. Ceperley and B. J. Adler *Phys. Rev. Lett.* **45** 556 (1980).

[10] J. P. Perdew and A. Zunger, *Phys. Rev. Lett.* **49** 1691 (1981).

[11] S. H. Vosko, L. Wilk, and M. Nusair, *Can. J. Phys.* **58**, 1200 (1980); S. H. Vosko and L. Wilk, *Phys. Rev. B* **22**, 3812 (1980).

[12] M. C. Payne, M. Teter, D. C. Allan, T. A. Arias and J. D. Joannopoulos, Iterative minimization techniques for *ab initio* total-energy calculations: molecular dynamics and conjugate gradients, *Rev. Mod. Phys.* **64**, 1045 (1992).

[13] J. W. Cooley and O. W. Tukey, "An Algorithm for the Machine Calculation of Complex Fourier Series." *Math. Comput.* **19**, 297-301, (1965).

[14] J. R. Chelikowsky, N. Troullier, and Y. Saad, Finite-difference-pseudopotential method: electronic structure calculations without a basis. *Phys. Rev. Lett.* **72** 1240 (1994).

[15] J. R. Chelikowsky, N. Troullier, K. Wu, and Y. Saad, 1994, Higher Order Finite Difference Pseudopotential Method: An Application to Diatomic Molecules, *Phys. Rev. B* **50** 11355-64 (1994).

[16] J. R. Chelikowsky, N. Troullier, X. Jing, D. Dean, N. Binggeli, K. Wu, and Y. Saad, Algorithms for the structural properties of clusters. *Computer Physics Communications*, **85**, 325-335 (1995).

[17] J. Bernholc, E.L. Briggs, C. Bungaro, M. Buongiorno Nardelli, J.-L. Fattebert, K. Rapcewicz, C. Roland, W.G. Schmidt, Q. Zhao Large-Scale Applications of Real-Space Multigrid Methods to Surfaces, Nanotubes, and Quantum Transport *phys. stat. sol. (b)* **217**, No. 1, 685-701 (2000).

[18] E.L. Briggs, D.J. Sullivan and J. Bernholc, *Phys. Rev. B* **54** 14362 (1996).

[19] A. Brandt, S. F. McCormick, and J. Ruge, *SIAM J. Sci. Stat. Comput* **4**, 244 (1983).

[20] W.L. Briggs and V.E. Henson, "Wavelets and Multigrid", *SIAM J. Stat. and Sci. Comp.*, 14 No. 2, pp. 506 10, (1993).

[21] I. Vasiliev, S. Öğüt, J. R. Chelikowsky, First-principles density-functional calculations for optical spectra of clusters and nanocrystals *Phys. Rev. B* **65**, 115416 (2002).

[22] S. Öğüt, J. R. Chelikowsky, and S. G. Louie, *Phys. Rev. Lett.* **79**, 1770 (1997).

[23] E. Tsuchida and M. Tsukada, *J. Phys. Soc. Jpn.* **67**, 3844 (1998).

[24] Y. Saad, A. Stathopoulos, J. Chelikowsky, K. Wu, and S. Öğüt, Solution of large eigenvalue problems in electronic structure calculations. *BIT*, **36**, pp. 563-578 (1995).

[25] Y. Saad, Numerical methods for large eigenvalue problems. *Manchester University Press, Machester.* (1992).

[26] T. Beck, Real-space multigrid Solution of Electrostatics Problems and the Kohn-Sham equations. *Intl. J. Quantum. Chem* **65**, 477 (1997).

[27] A. Brandt, Multi-level adaptive solutions to boundary value problems, *Math. Comp.* **31** 333-390 (1977).

[28] L. Collatz, The Numerical Treatment of Differential Equations, 3rd Ed. (Springer-Verlag, Berlin, 1964), p. 164.

[29] S. Costiner and S. Ta'asan *Phys. Rev. E* **51**, 3704 (1995).

[30] S. Costiner and S. Ta'asan *Phys. Rev. E* **52**, 1181 (1995).

[31] T. L. Beck, Real-space mesh techniques in density functional theory, *Rev. Mod. Phys.* **72**, 1041-1080 (2000).

[32] J. Mandel, S. F. McCormick, A Multilevel Variational Method for $Au = \lambda Bu$ on Composite Grids. *J. Comp. Phys.* **80**, pp. 442-452, (1989).

[33] Heiskanen, M., Torsti T., Puska, M. J., Nieminen, R. M., A novel multigrid method for electronic structure calculations *Physical Review B* **63**, 245106 (2001).

[34] S. Kotochigova, Z. H. Levine, E. L. Shirley, M. D. Stiles, and C. W. Clark, Local-density-functional calculations of the energy of atoms , *Phys. Rev. A* **55** , 191 (1997).

[35] S. Kotochigova, Z. H. Levine, E. L. Shirley, M. D Stiles, and C. W. Clark, website: http://physics.nist.gov/PhysRefData/DFTdata/contents.html

[36] R. W. Hamming, *Numerical Methods for Scientists and Engineers* (Dover, New York, 1962). Chapter 14 and 15.

[37] C. L. Lawson, R. J. Hanson, D. Kincaid, and F. T. Krogh, Basic Linear Algebra Subprograms for FORTRAN usage, `ACM Trans. Math. Soft.`, 5 pp. 308–323 (1979).

[38] Anderson, E. and Bai, Z. and Bischof, C. and Blackford, S. and Demmel, J. and Dongarra, J. and Du Croz, J. and Greenbaum, A. and Hammarling, S. and McKenney, A. and Sorensen, D., LAPACK Users' Guide, Third Edition (Society for Industrial and Applied Mathematics, Philadelphia, PA 1999).

[39] C. C. Douglas, Caching In with Multigrid Algorithms: Problems in Two Dimensions, *Parallel Algorithms and Applications* **9** 195–204, (1996).

[40] Gaussian 98, Revision A.11.3, M. J. Frisch, G. W. Trucks, H. B. Schlegel, G. E. Scuseria, M. A. Robb, J. R. Cheeseman, V. G. Zakrzewski, J. A. Montgomery, Jr., R. E. Stratmann, J. C. Burant, S. Dapprich, J. M. Millam, A. D. Daniels, K. N. Kudin, M. C. Strain, O. Farkas, J. Tomasi, V. Barone, M. Cossi, R. Cammi, B. Mennucci, C. Pomelli, C. Adamo, S. Clifford, J. Ochterski, G. A. Petersson, P. Y. Ayala, Q. Cui, K. Morokuma, N. Rega, P. Salvador, J. J. Dannenberg, D. K. Malick, A. D. Rabuck, K. Raghavachari, J. B. Foresman, J. Cioslowski, J. V. Ortiz, A. G. Baboul, B. B. Stefanov, G. Liu, A. Liashenko, P. Piskorz, I. Komaromi, R. Gomperts, R. L. Martin, D. J. Fox, T. Keith, M. A. Al-Laham, C. Y. Peng, A. Nanayakkara, M. Challacombe, P. M. W. Gill, B. Johnson, W. Chen, M. W. Wong, J. L. Andres, C. Gonzalez, M. Head-Gordon, E. S. Replogle, and J. A. Pople, Gaussian, Inc., Pittsburgh PA, 2002.

[41] P. Blaha, K. Schwarz, G. K. H. Madsen, D. Kvasnicka and J. Luitz, *WIEN2k, An Augmented Plane Wave + Local Orbitals Program for Calculating Crystal Properties* (Karlheinz Schwarz, Techn. Universität Wien, Austria), (2001). ISBN 3-9501031-1-2