2

UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

FACULTAD DE QUIMICA

EXAMENES PROFESIONALES
FACULTAD DE QUIMICA

MONTE CARLO CUANTICO: DESARROLLO DE UN PAQUETE DE SOFTWARE EDUCATIVO Y DE INVESTIGACION

T E S I S

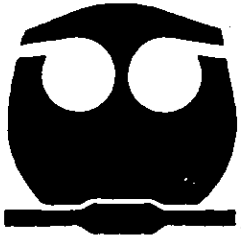
QUE PARA OBTENER EL GRADO DE:

Q U I M I C O

PRESENTAN:

ALAN ASPURU GUZIK

RAUL PERUSQUIA FLORES



CIUDAD UNIVERSITARIA, D.F.

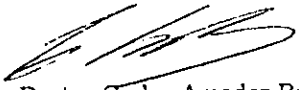
2000

Jurado Asignado

Presidente	Prof. Carlos Amador Bedolla
Vocal	Prof. Karl Mario García Ruiz
Secretario	Prof. Carlos Federico Bunge Molina
Primer Suplente	Prof. Emilio Orgaz Baqué
Segundo Suplente	Prof. Eugenia Corvera Poiré

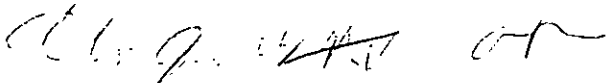
Esta tesis fue desarrollada en el departamento de Física y Química Teórica de la Facultad de Química de la UNAM.

Asesor



Doctor Carlos Amador Bedolla

Sustentantes



Alán Aspuru Guzik Raúl Alejandro Perusquía Flores

A mis papás,
Raúl

A mi familia,
Alán

Agradecimientos

Agradecemos a Dario Bressanini, por las interminables pláticas por correo electrónico, el código de las funciones de prueba correlacionadas, y todas las sugerencias que nos dio a lo largo de un año. A los abuelos: Juan Carlos (entomo) Martínez , Enrique (gordito) Hernández , Karl (karmelo) García y Rubén (ese) Zárate por las pláticas que tuvimos con respecto a la tesis. A Arturo (pupilo) Espinosa y Federico (quartic) Mena por enseñarnos trucos de C y GTK. También agradecemos a Miguel de Icaza, Fernando (la mancha) Magariños, los hermanos Daniel y Adrian Kornhauser, Germán (bizcocho) Salazar y Jorge (el_oso) Vieyra por el tiempo de máquina que nos prestaron sabiéndolo o sin saberlo. El desarrollo de la interfase gráfica no hubiera sido posible sin la invaluable ayuda de Francisco (bit) Bustamente, quien hizo la tabla periódica y el árbol de niveles de teoría. Muchas gracias a Daniel (roadmaster) Manrique por las horas que pasó con nosotros jugando con PVM. Por supuesto, sin el apoyo de Carlos Amador en todo, tal vez esta tesis nunca se hubiera realizado.

Raúl

A los Q'95 (Al Germán[2], a las Claudias, al Marquitos, la Licha[3], Gina, etc.) , a los QFBs (Bola, Andrea, Vero, Richi, David). A Mayra y los rayas, a las otras chavas y obviametne a Estela. Y absolutamente a nadie más[4]. A Dulce, Anita, Tibus y Vic32.

Alán

A todos mis amigos y amigas: El Ham, David, El Pupi, Miguel, Alejandra, Paula, Paulina, Andrea, El Mix, Roadmaster, Ruben, Juan Carlos, El Oso, Hiroko, El Tyco, etc.

[2]Por prestarme su casa.

[3]Caroza de fuego.

[4]No es cierto.

Índice General

Índice de Figuras

Prefacio

Two roads diverged in a wood, and
I took the one less traveled by,
And that has made all the difference.
Robert Frost, The Road Not Taken.

0.1. La revolución de la información y la ciencia

En las primeras décadas de este siglo, se formularon las bases teóricas de la mecánica cuántica. Científicos de la talla de Dirac creían que las ecuaciones estaban planteadas, y que sólo faltaba resolverlas para poder predecir cualquier fenómeno químico. Sin embargo, la solución de estas relaciones fundamentales era posible sólo realizando aproximaciones drásticas.

Durante la segunda guerra mundial, se empezaron a construír inmensas máquinas de cálculo con las que fue posible encontrar soluciones de sistemas relativamente pequeños. Estas máquinas han cambiado a nuestra sociedad en todos los ámbitos, desde la manera en la que nos comunicamos y percibimos el mundo, la forma en la que producimos e intercambiamos bienes materiales y nuestra capacidad de *simular* los distintos niveles de nuestra realidad: desde el nivel atómico hasta el nivel cosmológico. De hecho, varios historiadores consideran que hemos dejado la revolución industrial, y nos encontramos en la revolución de la infomración.

La evolución del poder de cómputo es asombrosa. En 1965, Gordon E. Moore notó que el número de componentes en un circuito integrado[1] se duplicaba cada año. Tan increíble como pueda oírse, esta predicción se ha cumplido hasta la fecha. Esto implica que cercanos al año 2000, tenemos un poder de cómputo 1.7×10^{10} mayor que en en 1965. A esta observación se le conoce como la *ley de Moore*. (ver figura 0.1).

La simulación de la realidad en el ámbito de la química se ha realizado en los más diversos niveles: desde el cálculo de potenciales de ionización hasta la predicción de propiedades de materiales. Podemos pensar en una *escalera de la complejidad* de la realidad. (ver figura 0.1).

[1] Cantidad directamente relacionada al poder de cómputo.

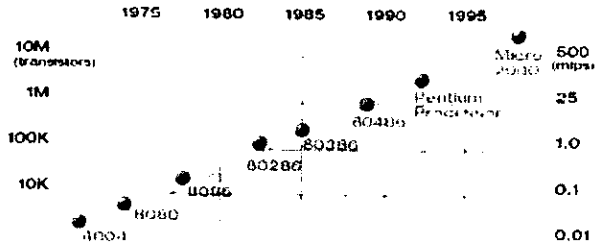


Figura 0.1.: La ley de Moore: Cada año se duplica el poder de cómputo, tanto en las computadoras personales, como en las supercomputadoras. (gráfica tomada de [1]).

Niveles		Ejemplos
3°	Nivel macroscópico	Puentes, turbinas, medicinas, biología
2°	Nivel mesoscópico	Fracturas,defectos,reacciones químicas
1°	Nivel cuántico	Átomos, moléculas y solidos

Tabla 0.1.: Para poder estudiar un sistema complejo, se requieren resultados de sistemas más sencillos.

Es válido hacerse la siguiente pregunta: ¿Qué sucede al ir aumentando la complejidad de los sistemas estudiados? ¿Podemos obtener resultados precisos conforme escalamos la escalera de la complejidad? La respuesta es no. Al hacer aproximaciones en cada escalón los errores se vuelven cada vez más difíciles de controlar.

Los caminos menos transitados en la simulación son aquellos en los que se efectuan menos aproximaciones. Generalmente, el prescindir de las aproximaciones implica un gasto computacional mayor; pero si confiamos en la Ley de Moore, los métodos que hoy día parecen costosos por el consumo de tiempo de computadora pueden convertirse en cálculos de rutina en 10 años.

El método de Monte Carlo cuántico está situado en el primer escalón de la complejidad. Su ventaja es que involucra muy pocas aproximaciones, por lo que los resultados que se obtienen son muy precisos. En la figura 0.1 podemos ver la tendencia en los errores de cálculo para la energía de pequeños hidrocarburos. El método que nosotros estudiamos tiene desviaciones muy pequeñas en comparación con la tendencia de los métodos tradicionales. Otro ejemplo promisorio es el cálculo de la energía de cohesión por átomo del Germanio sólido. Un cálculo con funcionales de la densidad da como resultado 4.59 eV/átomo, mientras que la técnica de Monte Carlo cuántico obtiene el valor confirmado experimental de 3.85 eV/átomo [32].

Nuestro principal objetivo es el desarrollaar un programa de Monte Carlo cuántico

gratuito y difundir el uso de esta poderosa técnica de cálculo. Esta técnica permitirá a los investgadores reducir los errores asociados a los cálculos en los primeros escalones, acercándonos a un mejor entendimiento de la realidad.

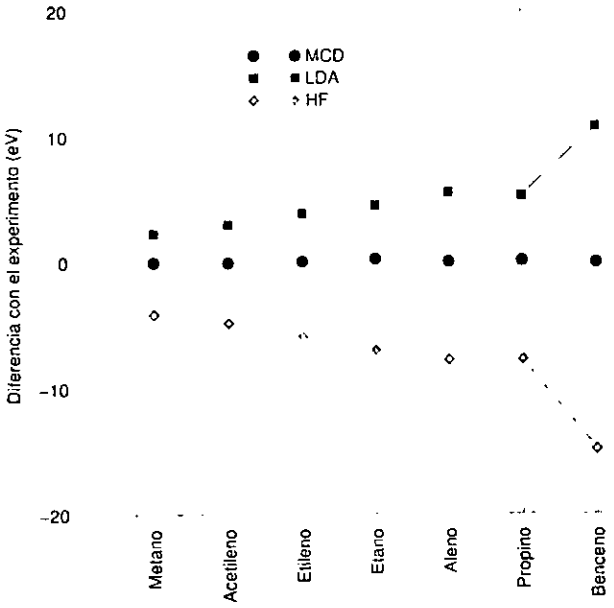


Figura 0.2.: El error en el cálculo de la energía de hidrocarburos con respecto a la energía experimental: Los métodos de Hartree-Fock subestiman la energía un promedio de 30%, mientras que los métodos de funcionales de la densidad la sobreestiman en un 20%. Los métodos de Monte Carlo cuántico tienen un error máximo del 1%. (resultados y gráfica tomados de [40]).

La figura (resume nuestro interés en MCC: obt resultados realistas.

El cálculo de estados propios para la ecuación de Schrödinger independiente del tiempo de sistemas complejos es uno de los principales retos para la química cuántica asi como para la física atómica y nuclear. Los métodos variacionales, en especial los de interacción de configuraciones (IC) han resultado los mas eficientes hasta la fecha.

El método de IC proporciona buenos resultados para sistemas pequeños, pero su eficiencia disminuye cuando el número de electrones (N) aumenta. De hecho, el esfuerzo computacional crece en un factor N^{α} donde α es del orden de 6 a 10 [77], haciendo que este método sea impráctico para un número grande de electrones. Este método es exacto en el límite en el que se evalúa un gran número de matrices. Éstas sólo pueden ser llevado a cabo completamente en casos relativamente pequeños (*Full CI*). A su vez, los cálculos tradicionales usan la aproximación de Born-Oppenheimer, haciendo que la optimización geométrica de una molécula requiera una gran canti-

dad de cálculos independientes para encontrar el mínimo de una superficie de energía potencial multidimensional.

El método de Monte Carlo cuántico (MCC) es una alternativa que permite resolver numéricamente la ecuación de Schrödinger para potenciales arbitrarios. Esto significa que esencialmente el mismo programa de cómputo puede ser empleado para resolver problemas tan distintos como la caja de potencial en una dimensión, el rotor rígido en tres dimensiones o el átomo de hidrógeno, todos ellos de gran interés didáctico; pero además, el mismo programa de MCC puede resolver sistemas más complejos tales como moléculas, iones, el estado basal de un cúmulo de muchos átomos, etcétera. Esta versatilidad hace de MCC un método de cálculo *ab initio* útil, competitivo y afortunadamente para nosotros, con muchas vetas teóricas en las que trabajar.

El MCC se basa en transformar la representación diferencial de la ecuación de Schrödinger a una representación integral. Al hacer esta transformación, cambiamos de un problema en el que el operador Hamiltoniano $\hat{\mathcal{H}}$ se especifica de manera local, a un problema en el que el operador inverso $G(\mathbf{R}, R'; \delta\tau)$ relaciona a cada punto de $\Psi(\mathbf{R}, \tau)$ con los demás. Esta ecuación integral puede ser resuelta por medio de series de Neumann y estas series, a su vez, pueden simularse por medio de una caminata aleatoria. El MCC tiene las siguientes ventajas [23, 53, 69]:

- El error del método puede ser calculado en la misma simulación a un bajo costo.
- No se depende de expansiones de conjuntos de base, y por lo tanto, no se tiene que trabajar con los mismos truncados o incompletos.
- En una sola corrida se pueden evaluar varias propiedades, desde la energía, la función de onda o la densidad electrónica, hasta propiedades de respuesta como dipolos, cuadrupolos, octupolos, etc.
- Las barras de error pueden ser estimadas rigurosamente.
- En cuanto a requerimientos computacionales, la memoria para una simulación de MCC es acotada *a priori*.
- No se calculan ni guardan costosas integrales.
- El código es fácilmente paralelizable.

En los primeros capítulos de la tesis se hará una introducción a algunos fundamentos matemáticos relacionados con la química cuántica y el método de Monte Carlo. Posteriormente, analizaremos con detalle los algoritmos de Monte Carlo cuántico variacional (MCV) y Monte Carlo cuántico de difusión (MCD). Después, describiremos al programa MOISS, que fue la pieza de software que creamos para realizar cálculos de MCC. A continuación, revisaremos los resultados de nuestra simulacion de algunos

sistemas cuánticos. La tesis termina con una descripción del software libre, presentando las razones por las cuales decidimos liberar nuestro programa completo bajo la licencia pública general (LPG).

En el apéndice A introducimos conceptos matemáticos útiles para el tratamiento de los temas que estudiamos.

Raúl y Alán

Notación y unidades empleadas

La notación que usaremos a lo largo del texto es:

$\hat{O}$	Operador.
$\hat{\mathcal{H}}$	Operador Hamiltoniano.
$\hat{P}$	Operador permutación.
Ψ	Función de onda dependiente del tiempo.
ψ	Función de onda independiente del tiempo .
ϕ	Función de prueba.
Φ	Función base.
χ	Orbital espín.
r	Posición en el espacio n dimensional de una partícula, $\mathrm{r} \equiv \{x_1, x_2, \cdots, x_n\}$
R	Posición de todas las partículas que componen a un sistema, $\mathrm{R} \equiv \{\mathrm{r}_1, \mathrm{r}_2, \cdots, \mathrm{r}_n\} = \{x_{1,1}, x_{2,2}, \cdots, x_{1,n}, x_{2,1}, \cdots, x_{2,n}, \cdots, x_{n,n}\}$
ω	Coordenada de espín.
x	Coordenada espín-espacial. $\mathrm{x} \equiv \{\mathrm{r}, \omega\}$
η, ξ	Variables aleatorias.
$\mathcal{M}(\eta)$	Esperanza matemática de η.
$\mathcal{D}(\eta)$	Varianza de η.
$\sigma(\eta)$	Desviación estándar η.
$\mathcal{U}_{a,b}$	Número aleatorio uniforme con intervalo (a, b).
δ_p	Error probable.
δ_r	Error real.
$\mathcal{G}_A$	Número aleatorio con distribución Gaussiana y varianza A.
floor(a)	Redondear al valor a al entero inferior.
$\overline{\mathrm{A}}$	Si A es una matriz, $\overline{\mathrm{A}}$ es su matríz transpuesta.

0.2. Unidades atómicas

En esta tesis se utilizarán unidades atómicas[5]. Éstas se presentan a continuación:

Unidad	*Definición*	*Símbolo*	*Valor en SI*
Carga	carga del electrón	e	$1.60219 \times 10^{-19} C$
Masa	masa del electrón	m_e	$9.10953 \times 10^{-31} kg$
Longitud	radio de Bohr	$\frac{4\pi\epsilon_0\hbar^2}{m_e e^2}$	$5.29177 \times 10^{-11} m$
Energía	$2PI_H$	$\frac{m_e e^4}{\hbar^2}$	$4.3598 \times 10^{-18} J$
Momento angular		$\hbar$	$1.0546 \times 10^{-34} Js$

donde PI_H es el potencial de ionización del hidrógeno, considerando la masa del protón como infinita ($m_p \to \infty$). A esta unidad de energía se le conoce como Hartree.

[5] Las unidades atómicas fueron introducidas por D.H. Hartree en 1928

1. Introducción: ¿Cómo hacer química cuántica con conejos?

Antes de comenzar a hablar de mecánica cuántica, funciones de Green y Monte Carlo, vale la pena pensar en una analogía que ayudará a entender que es lo que queremos hacer en las siguientes páginas.

Imagínate un valle completamente obscuro lleno de conejos. Los conejos son bastante tontos, y como no pueden ver nada, caminan al azar. Estos animales se reproducen... como conejos. Si el terreno del valle es bajo, hay comida y ausencia de depredadores, las condiciones les permiten dedicarse a las faenas reproductivas. En los terrenos altos y montañosos, hay varios cazadores y poca comida, por lo que los conejos que se adentran por ahí tienen una tasa de muerte alta. La cantidad de comida en el valle es limitada, por lo que los conejos no se pueden reproducir sin ton ni son, por lo que existirá un momento en el que algunos morirán por falta de alimento.

Si esperamos un tiempo suficiente, la población de conejos se estabilizará; tanto en número de conejos, como en la *distribución* espacial de estos en el valle.

Este proceso esta esquematizado en las figuras 1.1, 1.2 y 1.3.

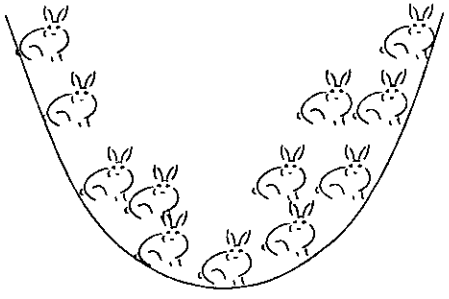


Figura 1.1.: Distribución inicial aleatoria de conejos en el valle.

Ahora imagínate que queremos salvar a los conejos de las garras de los cazadores, y por tanto de su inminente muerte en las zonas altas de las montañas. Podríamos hacer que saliera el sol, o dotarlos de olfato, de manera que los pequeños conejitos pudieran dirigirse hacia las zonas en las que tuvieran más probabilidad de sobrevivir. Haciendo

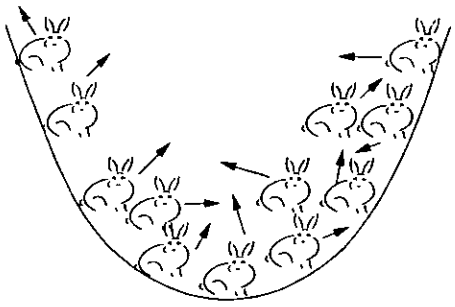


Figura 1.2.: Los conejos se mueven en el valle al azar; la reproducción depende de la altura del valle.

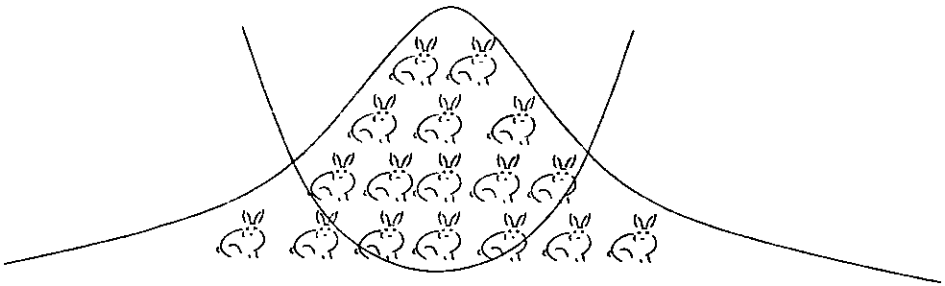


Figura 1.3.: Un *histograma* de la distribución final de los conejos al equilibrio.

esto, lograríamos que la población de conejos se equilibrara más rápido. EStaríamos *induciendo* a que la población llegara a un *estado estacionario* en menos tiempo.

Si quiseramos estudiar la distribución espacial de conejos en este mundo unidimensional, bastaría con apilarlos unos sobre otros, haciendo un *histograma* de su población. Si normalizamos el área bajo la curva, podemos encontrar la *distribución de probabilidad* de los conejos en el valle a cada instante. Y si esperamos lo suficiente, podemos encontrar su distribución en el equilibrio.

Si nos vemos hábiles, podríamos obtener información de la tasa de muerte y nacimiento de los conejos, quizá podríamos saber cuánta comida tenía el valle originalmente.

¡Las técnicas de MCC son muy parecidas a este juego de conejos!

Además, si fueramos unos grandes estudiosos de la fauna en el valle, podríamos cambiarle forma, cambiar la comida disponible en éste, o pensar en nuevas maneras de ayudarle a los conejos a encontrar una distribución estacionaria.

En esta tesis, nos enfrentamos a un problema similar. En vez de conejos, tenemos unos entes a los que llamamos *caminantes*, pequeños pedacitos de la ecuación de Schrödinger, que apilaremos en histogramas para conocer la función de onda asociada a ese valle de potencial, así como la cantidad de comida (energía) disponible en el valle.

Cuando hablemos de *difusión*, simplemente piensa en el caminar al azar del conejito. Su andar es parecido al de un borracho que no sabe en dónde está la siguiente cantina.

Por último, cuando hablemos de *reproducción y muerte*, piensa en cada vez que los conejitos procrean o se convierten en asado.

2. Mecánica cuántica

2.1. ¿Cuál es el problema?

El problema que ha mantenido entretenidos a muchos químicos cuánticos en los últimos sesenta años ha sido el de resolver la ecuación de Schrödinger y la ecuación de Dirac lo más preciso posible y cada vez para sistemas más grandes.

En el primer párrafo de su artículo *Quantisierung als Eigenwertproblem*, publicado en la revista *Annalen der Physik* el 27 de enero de 1926, Erwin Schrödinger menciona [72]:

> En este comunicado quisiera demostrar, primero para el caso simple del átomo de hidrógeno no relativista y sin perturbaciones, que las reglas usuales de la cuantización pueden ser reemplazadas por otro postulado, en el que no se mencionan números enteros. En vez de ésto, la introducción de números enteros aparece de un modo natural, como aparecen en una cuerda vibrante para la cual el número de nodos es entero. Esta nueva concepción puede ser generalizada, y creo que penetra profundamente en la verdadera naturaleza de las reglas cuánticas.

En este y en otros cuatro artículos publicados a lo largo de seis meses, Schrödinger aplica su ecuación de onda a problemas como el átomo de hidrógeno, el oscilador armónico, el rotor rígido, la molécula diatómica y el átomo de hidrógeno en un campo eléctrico (efecto Stark).

Sus métodos fueron adoptados rápidamente, y ahora no hay una rama de la física que no haya sido relacionada con esta ecuación.

El sistema dinámico propuesto por Schrödinger difiere de la mecánica clásica de Newton, Lagrange y Hamilton tanto en sus objetivos como en sus métodos. En vez de intentar encontrar ecuaciones que nos ayuden a predecir la posición (r) y momento (p) exactos de un sistema, en la mecánica cuántica se calcula una función de las coordenadas del sistema (R) y del tiempo (t), cuyo cuadrado nos proporciona valores probables para las coordenadas y otras propiedades dinámicas del sistema.

Posteriormente se encontró que el aceptar ecuaciones dinámicas del tipo de la ecuación de Schrödinger implica renunciar a la esperanza de describir en detalle el

comportamiento del sistema. El grado de certidumbre con el que podemos describir un sistema con los métodos de la mecánica cuántica está relacionado con el *principio de incertidumbre de Heisenberg*.

La ecuación de Schrödinger y sus postulados auxiliares nos permiten determinar ciertas funciones $\Psi(\mathbf{R}, t)$ llamadas *funciones de onda* o *funciones de amplitud de probabilidad*. El cuadrado de dicha función se interpreta como una *distribución de probabilidad* para las coordenadas del sistema representados por dicha función de onda.

La ecuación de Shchrödinger nos permite calcular las funciones de amplitud de probabilidad, $\Psi(\mathbf{R}, t)$, y la energía de los estados estacionarios asociados al sistema, entre otros observables de interés.

Para los motivos de esta tesis, la ecuación de Schrödinger y sus postulados auxiliares, las restricciones sobre la función de onda y la interpretación de la función de onda, se toman como postulados *a priori*.

Todas estas ideas son tratadas con más detalle en otras fuentes [13, 57, 58], y no nos detendremos a analizarlas en este trabajo.

La ecuación de Schrödinger dependiente del tiempo es:

$$\frac{1}{i}\frac{\partial \Psi(\mathbf{R}, t)}{\partial t} + \hat{\mathcal{H}}\Psi(\mathbf{R}, t) = 0, \tag{2.1}$$

donde $\hat{\mathcal{H}}$ es el operador **Hamiltoniano** del sistema, el cual se describe más adelante y con detalle en los capítulos 4 y 5. Podemos intentar resolver la ecuación utilizando el método de separación de variables [29], obteniendo que:

$$\Psi(\mathbf{R}, t) = \sum_{n}^{\infty} a_n \psi_n(\mathbf{R})\varphi_n(t); \tag{2.2}$$

donde la solución particular dependiente del tiempo tiene la forma:

$$\varphi_n(t) = \exp\left\{-2\pi i \frac{E_n}{h} t\right\}, \tag{2.3}$$

mientras que $\psi(\mathbf{R})$, es la solución de la siguiente ecuación de valores propios:

$$\hat{\mathcal{H}}\psi(\mathbf{R}) = E\Psi(\mathbf{R}) \tag{2.4}$$

conocida como ecuación de Schrödinger no relativista independiente del tiempo [1].

Las soluciones de la ecuación de Schrödinger para sistemas atomicos y moleculares no pueden predecir varias lineas espectrales encontradas experimentalmente. Científicos como Sommerfeld, Landé y Dirac las pudieron explicar tomando en cuenta efectos relativistas. En 1925, Uhlenbeck y Goudsmit [2] propusieron que para describir completamente al electrón, es necesario atribuirle una propiedad conocida como

[1] La llamaremos simplemente ecuación de Schrödinger.

[2] El articulo original es el siguiente: G.E. Uhlenberk & S. Goudsmit, *Naturwissenschften* 13, 953 (1985). Un año después, se tradujo al inglés en el volumen 117 de Nature.

espín[3]. Podemos describir el espín de un electrón por medio de 2 funciones ortonormales, $\alpha(\omega)$ y $\beta(\omega)$, las cuales representan a un espín hacia arriba y un espín hacia abajo respectivamente. Entonces, las funciones de onda no dependen solamente de las 3 coordenadas espaciales sino también de una coordenada de espín ω. A este conjunto de coordenadas lo representaremos como:

$$x = \{r, \omega\}.$$

Por último, para poder describir correctamente sistemas electrónicos, se debe cumplir el principio de exclusión de Pauli, que enuncia que la función de onda total de un sistema de fermiones debe ser antisimétrica ante el intercambio de partículas. Ésto implica que si dos argumentos son cambiados de lugar en la función de onda, ésta debe cambiar de signo:

$$\psi(x_1, x_2, \ldots, x_i, x_j, \ldots, x_n) = -\psi(x_1, x_2, \ldots, x_j, x_i, \ldots, x_n) \tag{2.5}$$

El siginficado físico del principio de exclusión es que dos electrones con el mismo espín no pueden ocupar el mismo estado simultáneamente.

El reto asociado a resolver esta ecuación exactamente, es que nos enfrentamos a un problema de muchos cuerpos, específicamente muchos fermiones[4]. Aun no se conocen soluciones exactas para sistemas de muchos cuerpos, por lo que para casos realistas, se tienen que hacer aproximaciones.

A manera de revisión somera, presentaremos las formas tradicionales de resolver la ecuación de Schrödinger.

2.2. Métodos de un solo electrón

Se le llama *orbital* a la función de onda de una sola partícula. Un orbital espacial, $\Phi_i(r)$, es una función del vector posición r que describe la distribución espacial del electrón. Siendo $|\Phi_i(r)|^2 dr$ la probabilidad de encontrar a dicho electrón en el elemento de volumen dr centrado en r. Por cada orbital espacial podemos formar 2 *orbitales espín*:

$$\chi(x) = \begin{cases} \Phi(r)\,\alpha(\omega) \\ \text{ó} \\ \Phi(r)\,\beta(\omega) \end{cases}, \tag{2.6}$$

multiplicando el orbital espacial por las 2 funciones de espín.

[3]Presenta este nombre debido a que originalmente, por analogía con la mecánica clásica, se pensaba que esta propiedad surgía del giro (*spin* en ingles) de electrón sobre un eje.

[4]Son partículas que presentan espín fraccionario y siguen la estadística de Fermi. Las funciones de onda asociadas a éstas son antisimétricas con respecto al intercambio.

Para resolver un sistema con N electrones, podemos como primera aproximación, asumir que cada electrón se mueve independientemente. De esta manera, podemos separar variables y separar el problema en N ecuaciones independientes de un solo electrón:

$$-\frac{1}{2}\nabla^2\chi_i(\mathbf{x}) = \epsilon_i\chi_i(\mathbf{x}). \tag{2.7}$$

La solución al problema de muchos cuerpos es dada por el producto de las N soluciones de la ecuación (2.7):

$$\phi(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N) = \chi_i(\mathbf{x}_1)\chi_j(\mathbf{x}_2)\cdots\chi_k(\mathbf{x}_N). \tag{2.8}$$

El valor propio asociado al operador Hamiltoniano es la Energía (E). En este caso, ésta se obtiene sumando la energía de cada electrón:

$$E = \epsilon_i + \epsilon_j + \cdots + \epsilon_k. \tag{2.9}$$

2.2.1. La aproximación de Hartree

En el caso anterior, las funciones de onda de cada electrón no estaban *correlacionadas*[5]. En esta segunda aproximación, partimos de las ecuaciones para un solo electrón (2.7). Pero ahora, se propone un potencial total $U(\mathbf{r})$ para modelar la interacción electrón-electrón y electrón-ión[6].

Los iones contribuyen al potencial total como:

$$U_{iones}(\mathbf{r}) = -\sum_{\alpha}\frac{Z_\alpha}{\mathbf{r}-\mathbf{d}_\alpha}. \tag{2.10}$$

Donde Z_α es la carga del ión α, y d_α es la distancia de electrón al ión[7].

Mientras que la contribución de cada uno de los electrones al potencial total se aproxima como una interacción electrostática **promedio** con todos los demás. Ésta se puede escribir como función de la densidad electrónica, $\rho(\mathbf{r})$, como:

$$U_H(\mathbf{r}) = \int[\rho(\mathbf{r}') - \rho_i(\mathbf{r}')]\frac{1}{|\mathbf{r}-\mathbf{r}'|}d\mathbf{r}', \tag{2.11}$$

donde no se considera el potencial de interacción del electrón i consigo mismo.

[5] Por correlación entendemos el fenómeno físico de interacción entre electrones. Cuando un electrón se acerca a otro, es repelido por éste. En el caso de la solución 2.8, los electrones no interactuan entre sí.

[6] En esta tesis, nos referiremos a los núcleos y a los iones positivos indistintamente.

[7] Estamos empleando la aproximación de Born-Oppenheimer, en la cual los núcleos se consideran fijos.

Para calcular el potencial de Hartree es necesario conocer la distribución electrónica del sistema. Si se asume que los electrones son independientes unos de otros, es sencillo construir la densidad electrónica a partir de las funciones propias monoelectrónicas:

$$\rho(\mathbf{r}) = \sum_i |\chi_i(\mathbf{x})|^2, \tag{2.12}$$

donde la suma sobre i corre sobre todos los estados ocupados. Usando esta densidad electrónica, el potencial total para un sólo electrón es:

$$U_i(\mathbf{r}) = U_{iones}(\mathbf{r}) + \sum_{j \neq i} \int |\chi_j(\mathbf{x}')|^2 \frac{1}{|\mathbf{r} - \mathbf{r}'|} d\mathbf{r}'. \tag{2.13}$$

Si sustituimos (2.13) en (2.7), obtenemos el conjunto de ecuaciones conocidas como *ecuaciones de Hartree*:

$$(-\frac{1}{2}\nabla^2 + U_{iones}(\mathbf{r}))\chi_i(\mathbf{x}) + \sum_{j \neq i} \int |\chi_j(\mathbf{r}')|^2 \frac{1}{|\mathbf{r} - \mathbf{r}'|} d\mathbf{r}' \chi_i(\mathbf{x}) = \epsilon_i \chi_i(\mathbf{x}). \tag{2.14}$$

Las ecuaciones del tipo (2.14) también pueden ser obtenidas por argumentos variacionales [90].

2.2.2. La aproximación de Hartree-Fock

La aproximación de Hartree-Fock es una extensión de la idea original de Hartree en la que se incluye la simetría de permutación de la función de onda, que lleva a la interacción de intercambio. El intercambio se debe al principio de exclusión de Pauli mencionado anteriormente. Un tipo de función de onda que satisface los requisitos de antisimetría es un determinante de Slater:

$$\phi = \begin{vmatrix} \chi_1(\mathbf{x}_1) & \chi_1(\mathbf{x}_2) & \cdots & \chi_i(\mathbf{x}_N) \\ \chi_2(\mathbf{x}_1) & \chi_2(\mathbf{x}_2) & \cdots & \chi_2(\mathbf{x}_N) \\ \vdots & \vdots & \ddots & \vdots \\ \chi_N(\mathbf{x}_1) & \chi_N(\mathbf{x}_2) & \cdots & \chi_N(\mathbf{x}_N) \end{vmatrix} \tag{2.15}$$

donde las entradas $\chi_i(\mathbf{x}_j)$ son funciones de onda de un solo electrón.

Al minimizar los valores esperados del operador $\tilde{\mathcal{H}}$ con respecto a las funciones de onda de un electrón, las ecuaciones obtenidas son llamadas de *Hartree-Fock*:

$$\begin{aligned}(-\frac{1}{2}\nabla^2 + U_{iones}(\mathbf{r}))\chi_i(\mathbf{r}) + \sum_{j \neq i} \int |\chi_j(\mathbf{r}')|^2 \frac{1}{|\mathbf{r} - \mathbf{r}'|} d\mathbf{r}' \chi_i(\mathbf{r}) \\ - \sum_j \delta_{s_i s_j} \int \frac{\chi_j(\mathbf{r}')\chi_i(\mathbf{r}')}{|\mathbf{r} - \mathbf{r}'|} \chi_j(\mathbf{r}) = \epsilon_i \chi_i(\mathbf{r}),\end{aligned} \tag{2.16}$$

donde s_i es el espín de la partícula i. El término extra en estas ecuaciones con respecto a la ecuación (2.14) se conoce como la parte de ésta relacionada con el intercambio. El valor numérico del intercambio es distinto de cero cuando consideramos electrones del mismo espín[8]. El efecto físico del intercambio es que los electrones del mismo espín tienden a evitarse mutuamente. Como resultado de ésto hay un *hueco* asociado a esta prohibición llamado *hueco de intercambio* o hueco de Fermi. Este es un volumen pequeño alrededor de los electrones que evitan las partículas con el mismo espín. La densidad de carga en el hueco de intercambio es positiva y equivalente a la ausencia de un electrón.

El término de intercambio es un operador integral no local que hace a las ecuaciones de Hartree-Fock difíciles de resolver. Aunque en algunos casos simples la solución es practicable.

2.3. Interacción de configuraciones

En este método, la integral $\langle\chi|\hat{\mathcal{H}}|\chi\rangle$ es minimizado buscando todas las funciones χ que sean combinaciones lineales de determinantes N-electrónicos hechos de hasta M estados base ($M >> N$). Ésto no resulta practico para un número grande de electrones, ya que existen

$$^{M}C_{N} = \frac{M!}{N!(M-N)!} \tag{2.17}$$

determinantes de N-electrones. Por lo que la matriz a diagonalizar tiene una dimensión $^{M}C_{N}$. Lo que usualmente se acostumbra hacer es usar determinantes cercanos al estado basal de *Hartree-Fock*. Los métodos de IC son sólo prácticos para moléculas pequeñas por requerir mucho tiempo de cómputo.

2.4. Funcionales de la densidad

La teoría de funcionales de la densidad (DFT por sus siglas en inglés) [71] es una teoría formalmente exacta basada en la densidad de carga del sistema. La teoría de funcionales de la densidad de Kohn y Sham es una teoría de un electrón formalmente exacta.

La ecuación de Schrödinger es reemplazada por un conjunto de N ecuaciones de la forma:

$$(-\frac{1}{2}\nabla^2 + V(\mathbf{r}))\Phi_i(\mathbf{r}) = \epsilon\Phi_i(\mathbf{r}), \tag{2.18}$$

[8]Para mayor información referirse a [84].

donde $\psi_i(\mathbf{r})$ es una función de onda monoelectrónica. Éstas funciones contienen un potencial $V(\mathbf{R})$ producido por todos los iones y electrones. La teoría de funcionales de la densidad incluye todas las partes de la interacción electrónica (el potencial de Hartree):

$$V_H(\mathbf{r}) = \int \frac{\rho(\mathbf{r}')}{|\mathbf{r} - \mathbf{r}'|} d\mathbf{r}, \tag{2.19}$$

donde ρ es la densidad de carga de todos los electrones, un potencial relacionado a los efectos de intercambio y correlación $V_{IC}(\mathbf{r})$, y el potencial externo debido a los iones.

Hohenberg y Kohn originalmente desarrollaron la teoría de DFT para el estado basal de un sistema de fermiones sin espín. En un sistema de ese tipo, la densidad está dada por:

$$\rho(\mathbf{r}) = N \int |\psi(\mathbf{R})|^2 d\mathbf{R}, \tag{2.20}$$

,donde R es un vector que corresponde a todas las coordenadas electrónicas:

$$\mathbf{R} = (\mathbf{r}, \mathbf{r}_2, \ldots, \mathbf{r}_N). \tag{2.21}$$

Se puede demostrar que el estado basal del sistema es un funcional de la densidad $E[\rho(\mathbf{r})]$, y que si la energía correspondiente a las interacciones electrón-ión es excluida, el resto es un función universal de la densidad $F[\rho(\mathbf{r})]$[9]. El recuadro 2.1 tiene una prueba de la teoría de DFT.

2.4.1. Las ecuaciones de Kohn y Sham

Kohn y Sham propusieron un método basado en el teorema de Hohenberg-Kohn que permite minimizar el funcional $E[\rho(\mathbf{r})]$ variando $\rho(\mathbf{r})$ a lo largo de todas las densidades que contengan N electrones. Esta restricción se logra por medio del multiplicador de Lagrange μ, que es escogido de manera que $\int \rho(\mathbf{r})d\mathbf{r} = N$, es decir,

$$\frac{\delta}{\delta\rho(\mathbf{r})}\left[E[\rho(\mathbf{r})] - \mu \int \rho(\mathbf{r})d\mathbf{r}\right] = 0 \rightarrow \frac{\delta E[\rho(\mathbf{r})]}{\delta\rho(\mathbf{r})} = \mu. \tag{2.22}$$

Kohn y Sham escogieron separar a $F[\rho(\mathbf{r})]$ en tres partes, de manera que $E[\rho(\mathbf{r})]$ se convierte en :

$$E[\rho(\mathbf{r})] = T_s[\rho(\mathbf{r})] + \frac{1}{2}\int\int \frac{\rho(\mathbf{r})\rho(\mathbf{r}')}{|\mathbf{r} - \mathbf{r}'|} d\mathbf{r}d\mathbf{r}' + E_{IC}[\rho(\mathbf{r})] + \int \rho(\mathbf{r})V_{ext}(\mathbf{r})d\mathbf{r}, \tag{2.23}$$

[9]Es decir,$F[\rho(\mathbf{r})]$ no depende de la forma del potencial generado por los iones.

donde $T_s[\rho(\mathrm{r})]$ es definido como el operador de energía cinética de un gas de electrones libres (con partículas *independientes entre sí*) con densidad $\rho(\mathrm{r})$,

$$T_s[\rho(\mathrm{r})] = \frac{1}{2}\sum_{i=1}^{N}\int \psi_i^*(\mathrm{r})\psi_i(\mathrm{r})d\mathrm{r}. \tag{2.24}$$

La ecuación (2.23) también puede ser vista como una definición del *funcional de la energía de intercambio y correlación*, $E_{IC}[\rho(\mathrm{r})]$. Podemos reescribir a la ecuación (2.22) en términos de un potencial efectivo, $V_{ef}(\mathrm{r})$ de la siguiente manera:

$$\frac{\delta T_s[\rho(\mathrm{r})]}{\delta\rho(\mathrm{r})} + V_{ef}(\mathrm{r}) = \mu, \tag{2.25}$$

donde:

$$V_{ef}(\mathrm{r}) = V_{ext}(\mathrm{r}) + \int \frac{\rho(\mathrm{r}')}{|\mathrm{r}-\mathrm{r}'|}d\mathrm{r}' + V_{IC}(\mathrm{r}), \tag{2.26}$$

y

$$V_{IC}(\mathrm{r}) = \frac{\delta E_{IC[\rho(\mathrm{R})]}}{\delta\rho(\mathrm{r})}. \tag{2.27}$$

Para encontrar la energía del estado basal y la densidad electrónica, basta resolver las ecuaciones de un electrón:

$$(-\frac{1}{2}\nabla_i^2 + V_{ef}(\mathrm{r}) - \epsilon_i)\psi_i(\mathrm{r}) = 0, \tag{2.28}$$

y la densidad se construye de acuerdo a:

$$\rho(\mathrm{r}) = \sum_{i=1}^{N}|\psi_i(\mathrm{r})|^2 \tag{2.29}$$

Al resolver las ecuaciones (2.26)-(2.29) autoconsistentemente, se encuentra la solución al problema de funcionales de la densidad.

Existen varias formas funcionales para el intercambio y correlación. Entre ellas, la aproximación de densidad local (LDA, por sus siglas en inglés), en la que el funcional de IC se aproxima a una simple función de la densidad en cualquier posición r. El valor de esta función es la energía de intercambio y correlación para un gas de electrones homogéneo de densidad $n(\mathrm{r})$. Cabe mencionar que estos datos de densidad del gas de electrones fueron obtenidos por las primeras simulaciones de Monte Carlo Cuántico.

2.5. Otra manera de resolver la caja de potencial

El problema de la caja de potencial es un *clásico* de la mecánica cuántica, tiene varias cualidades que lo hacen un excelente problema didáctico:

- tiene solución exacta,
- es un modelo sencillo para describir a los electrones π de un alqueno, a un protón en un núcleo atómico, a una partícula de gas encerrada en una caja completamente aislante, etc.
- Existen varias maneras de resolverlo.

La manera tradicional de estudiarlo puede consultarse en varios libros de química cuántica [19]. En este caso, mataremos moscas con cañonazos, y lo resolveremos usando la técnica de las funciones de Green.

Debido a que en los siguientes capítulos abordaremos a la solución general de la ecuación de Schrödinger con funciones de Green, vale la pena adentrarse en un problema sencillo primero.

Si el lector no se encuentra familiarizado con las funciones de Green, puede revisar el apéndice A, o la bibliografía al respecto [9, 52].

El tratamiento que haremos aquí es una pequeña variante del originalmente propuesto por Kalos y Whitlock [54], ya que no usaremos ningún método de Monte Carlo[10].

El formalismo de la ecuación de Schrödinger y su representación con funciones de Green se explica con detalle en el capítulo 5; por lo pronto nos interesaremos en las propiedades *cualitativas* de dicha forma de estudiar esta ecuación.

En la caja de una dimensión, el valor del potencial tiene la forma:

$$V(x) = \begin{cases} 0, \text{ si } b > x > a \\ \infty, \text{ si } b < x \text{ ó } x < a \end{cases} \tag{2.30}$$

Y la ecuación de Schrödinger del sistema determina al mismo tiempo la función de onda (función propia),ψ, y la energía (valor propio), E, del *operador hamiltoniano:*

$$\hat{\mathcal{H}} = \frac{-1}{2}\frac{d^2\psi}{dx^2} + V(x). \tag{2.31}$$

Estamos interesados en obtener las propiedades el estado *basal* [11] de la ecuación de Schrödinger para la caja de potencial, E_0, ψ_0:

$$\hat{\mathcal{H}}\psi_0 = E_0\psi_0. \tag{2.32}$$

Para poder definir a la ecuación anterior, basta proponer unas condiciones a la frontera, que definan a los parámetros a y b del potencial (2.30) como -1 y 1 respectivamente. Esto equivale a:

$$\psi(-1) = 0, \ \psi(1) = 0. \tag{2.33}$$

[10] Debido a la baja dimensionalidad del problema, integragremos por el método de Simpson. Además, no aprenderemos a integrar por Monte Carlo sino hasta el siguiente capítulo.

[11] Entendemos como estado basal a la solución de la ecuación que tenga la menor energía.

Estas condiciones implican que la función de onda debe de ser cero en los puntos en los que el potencial diverge al infinito. Reduciendo así la ecuación anterior a una ecuación en la que el término $V(\mathrm{r})$ es cero cuando nos encontramos en la región $(-1, 1)$, por lo tanto:

$$\frac{-1}{2}\frac{d^2\psi}{dx^2}\psi_0 = E\psi_0. \tag{2.34}$$

Donde las soluciones de la ecuación, la función de onda y la energía, son:

$$\psi_0(\mathrm{r}) = cos\left(\frac{\pi}{2}x\right), \quad E_0 = \left(\frac{\pi}{2}\right)^2. \tag{2.35}$$

La función de onda se puede ver en la figura 2.1.

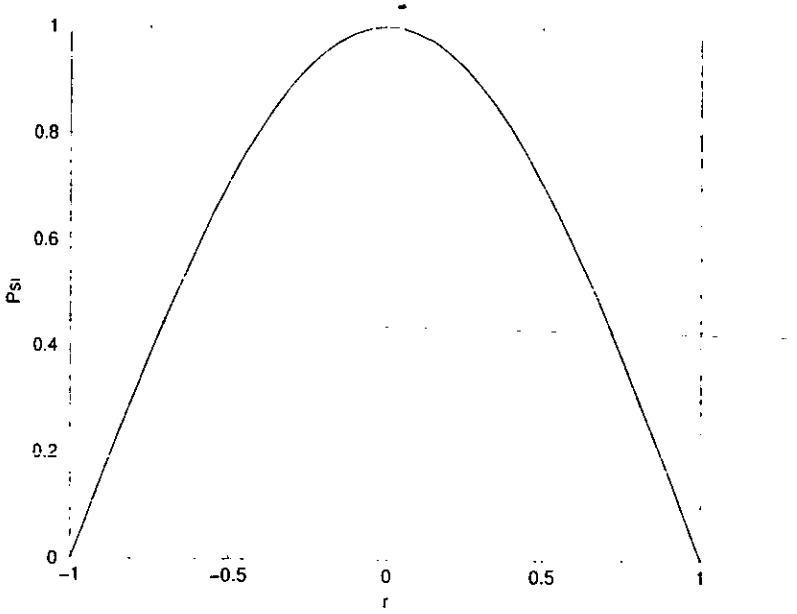


Figura 2.1.: Función de onda del estado basal de la caja de potencial con condiciones a la frontera $\psi(-1) = 0, \quad \psi(1) = 0$.

Como analizaremos con detalle en los subsecuentes capítulos, es posible transformar la ecuación diferencial (2.32) a su forma integral:

$$\psi_0(\mathrm{r}) = E_0 \int G(\mathrm{r}, \mathrm{r}')\psi_0(\mathrm{r}')d\mathrm{r}'. \tag{2.36}$$

Analicemos la ecuación anterior con cuidado. Lo primero que vemos es que aparecen dos variables de posición: r y r'. Por lo pronto, nos interesa que ambas sean intercambiables. Además aparece una función que depende de ambas, $G(\mathrm{r}, \mathrm{r}')$, a la que

llamamos *función de Green*. Al parecer no hemos avanzado mucho, de una ecuación dependiente de una variable, r, hemos pasado a una que depende de dos variables, r y r', introduciendo una función auxiliar extraña $G(\mathrm{r},\mathrm{r}')$, y además hemos hecho que la función de onda en un punto $\psi(\mathrm{r})$ dependa de una integral sobre todos los puntos, $\int d\mathrm{r}'$.

Por ahora supondremos que podemos resolver la ecuación anterior por iteración:

$$\psi^{(n+1)}(\mathrm{r}) = E_R \int G(\mathrm{r},\mathrm{r}')\psi^{(n)}(\mathrm{r}')d\mathrm{r}'. \tag{2.37}$$

Puesto que en la ecuación (2.36) no conocemos de antemano ni a ψ_0 ni a E_0, proponemos que iterando la ecuación anterior, nos iremos acercando al valor de ellos. Así, proponemos una $\psi^{(n)}$ y una E_R, para esperar esperar que:

- multiplicando a la función $G(\mathrm{r},\mathrm{r}')$ por $\psi^{(n)}$ en todos los puntos, obtengamos a la función de onda de la siguiente iteración, $\psi^{(n+1)}$.
- dado que tenemos un nuevo conjunto de puntos, $\psi^{(n+1)}$, entonces podemos repetir la operación hasta el infinito.

Por el momento asumiremos que la misteriosa función de Green tiene la forma:

$$G(\mathrm{r},\mathrm{r}') = \begin{cases} \frac{1}{2}(1-\mathrm{r}')(1+\mathrm{r}), & \text{si } \mathrm{r} \leq \mathrm{r}' \\ \frac{1}{2}(1+\mathrm{r}')(1-\mathrm{r}), & \text{si } \mathrm{r} \geq \mathrm{r}' \end{cases} \tag{2.38}$$

Si graficámos la función anterior, vemos que tiene la forma de un triángulo con un *vértice* en el punto $\mathrm{r} = \mathrm{r}'$, y que también cumple las mismas condiciones a la frontera que la ecuación de Schrödinger, $G(-1,-1) = 0$ y $G(1,1) = 0$ (ver la figura 2.2).

2.5.1. Resolviendo la ecuación de Schrödinger para la caja de potencial por medio de iteraciones.

Propongamos una función de onda ψ^0 conformada de un conjunto r' de 40 números aleatorios entre 0 y 1 espaciados uniformemente:

$$\psi^0(\mathrm{r}') = \{(-1.0, 0.0343), (-0.9, 0.682), \ldots, 0.9, 0.462, 1.0, 0.973\} \tag{2.39}$$

Iteremos la ecuación (2.37) siguiendo el siguiente algoritmo:

- Escoja el valor de $\mathrm{r} = -1.0$.
- Escoja el primer punto de $\psi^0(\mathrm{r}')$, y aplique la función de Green, $G(\mathrm{r},\mathrm{r}')$.
- Repita el paso anterior para todos los puntos de $\psi^0(\mathrm{r}')$.

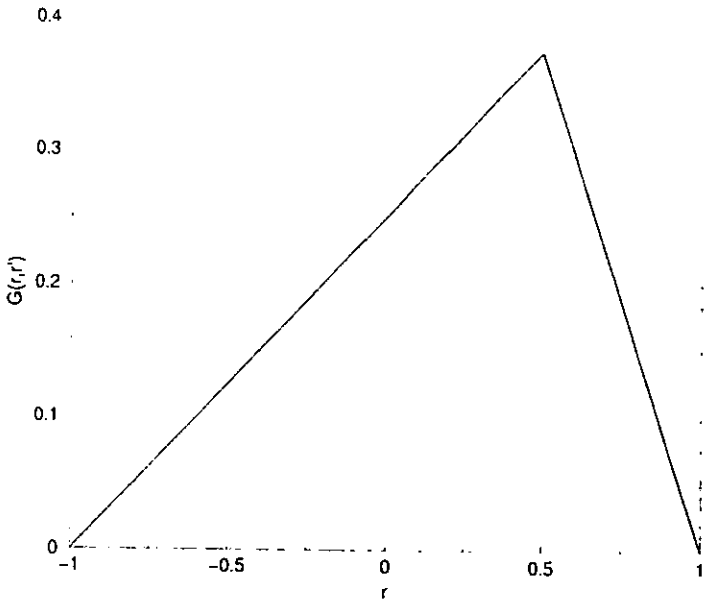


Figura 2.2.: Función de Green para la caja de potencial con condiciones a la frontera $\psi(-1) = 0$, $\psi(1) = 0$, y r = 0.5.

- Sume el resultado de todos los valores obtenidos en los dos pasos anteriores y multiplique por dr. Éste es una aproximación a la integral:

$$I = \int G(\mathbf{r}, \mathbf{r}')\psi^{(n)}(\mathbf{r}')d\mathbf{r}' \tag{2.40}$$

  Estamos integrando numéricamente por el método de Simpson, en el cual la integral se aproxima por la suma $I = \sum_i f(i)\Delta x$. Debido a que este método de integración es bastante burdo, perderemos área cada vez que se haga una iteración.

- Asígnelo al primer punto de $\psi^1 = (-1.0, I)$.
- Repita los cuatro pasos anteriores hasta obtener todos los puntos de ψ^1.
- Ahora ψ^1 es la función de onda dentro de la integral y ψ^2 será la nueva función que se obtendrá usando la función de Green.
- Repita los pasos anteriores cuatro o cinco veces.

Los resultados de aplicar el algoritmo anterior son impresionantes, de un conjunto de puntos aleatorios, obtenemos la función de onda en muy pocas iteraciones. De hecho, a la primera ya tenemos una función casi exacta.

Vale la pena mencionar que en la iteración anterior, no ajustamos la energía de referencia E_R, por lo que la función de onda obtenida no está normalizada. Basta normalizarla para encontrar a ψ_0 y E_0. (ver figura 2.3).

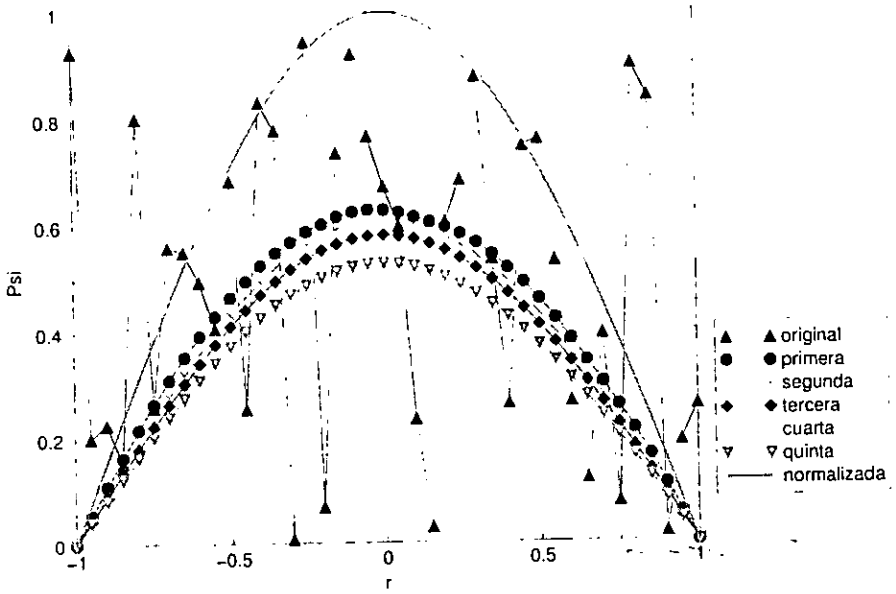


Figura 2.3.: Solución por medio de funciones de Green para la caja de potencial con condiciones a la frontera $\psi(-1) = 0, \quad \psi(1) = 0$. Se muestra la función de onda después de varias iteraciones.

En los capítulos subsecuentes estudiaremos casos más complejos de la ecuación de Schrödinger. Como veremos con detalle, realizar una integración de ψ por el método de mallas[12] se vuelve no costeable computacionalmente conforme el número de dimensiones crece , por lo que tendremos que resolver la integral (2.36) por métodos de Monte Carlo.

Regresaremos a los detalles de este ejemplo en la sección 3.8.

[12]La integral que acabamos de realizar es una integral de *malla*, pues el eje de las posiciones r fue dividido en m subdivisiones. Si se generalizara ésto a más dimensiones n, el costo computacional de realizar la integral de malla crece proporcionalmente a n.

- Para cualquier densidad dada por una función N-electrónica antisimétrica, se puede definir un funcional de la densidad correspondiente a cualquier operador $\hat{O}$ como:

$$O[\rho(\mathbf{r})] = \min_{|\psi\rangle \to \rho(\mathbf{r})} \langle \psi | \hat{O} | \psi \rangle$$

El lado derecho se evalúa proponiendo funciones ψ y buscando cuál de ellas minimiza a O.

- Podemos definir a $F[\rho(\mathbf{R})]$ de la misma manera, donde:

$$\hat{\mathcal{F}} = \sum_i -\frac{1}{2}\nabla_i^2 + \frac{1}{2}\sum_{i \neq j} \frac{1}{|\mathbf{r}_i - \mathbf{r}_j|} \quad , \quad F[\rho(\mathbf{r})] = \min_{|\psi\rangle \to \rho(\mathbf{r})} \langle \psi | \hat{\mathcal{F}} | \psi \rangle.$$

- Sea ψ_0 el estado basal de un sistema N-electrónico y ψ un estado que nos da una densidad $\rho(\mathbf{r})$ y minimiza $\langle \psi | \hat{\mathcal{F}} | \psi \rangle$. Entonces, de la definición de $E[\rho(\mathbf{r})]$:

$$E[\rho(\mathbf{r})] = F[\rho(\mathbf{r})] + \int \rho(\mathbf{r}) V_{ext}(\mathbf{r}) d\mathbf{r} = \langle \psi | \hat{\mathcal{F}} + V_{ext} | \psi \rangle$$

- Ahora, la función $\hat{\mathcal{F}} + V_{ext}$ no es otra cosa más que el Hamiltoniano de la ecuación de Schrödinger, por lo que $E[\rho(\mathbf{r})]$ debe obedecer el teorema variacional (ver la sección 4.1):

$$E[\rho(\mathbf{r})] \geq E_0.$$

- De la definición previa de $F[\rho(\mathbf{r})]$, tenemos:

$$F[\rho(\mathbf{r})] \leq \langle \psi_0 | \hat{\mathcal{F}} | \psi_0 \rangle,$$

debido a que ψ_0 es sólo una de las funciones de prueba de las que se obtiene $\rho_0(\mathbf{r})$.

- Si sumamos $\int \rho(\mathbf{r}) V_{ext}(\mathbf{r}) d\mathbf{r}$ a la ecuación anterior, obtenemos:

$$E[\rho(\mathbf{r})] \leq E_0,$$

- que en combinación con la expresión previa $E[\rho(\mathbf{r})] \geq E_0$, nos da:

$$E[\rho(\mathbf{r})] = E_0$$

QED.

Tabla 2.1.: Prueba de que en la teoría de los funcionales de la densidad, la energía es un funcional de la densidad.

3. El Método de Monte Carlo

3.1. Introducción

El método de Monte Carlo provee de soluciones aproximadas a una gran variedad de problemas matemáticos haciendo muestreos estadísticos en una computadora [31]. Vale la pena hacer notar que este método se aplica a problemas de naturaleza no probabilística (haciendo uso de hábiles trucos) y también a problemas con naturaleza estadística intrínseca.

Esto por sí solo no le da ninguna ventaja al método de Monte Carlo sobre algún otro método de aproximación. Sin embargo, de entre todos los métodos numéricos que requieren de una evaluación de n puntos en un espacio m dimensional para producir una solución aproximada, el método de Monte Carlo tiene un error absoluto que decrece como $n^{1/2}$ mientras que, en ausencia de una estructura especial explotable (por ejemplo, simetrías), los demás métodos decrecen como $n^{1/m}$ cuando mucho.

Esta propiedad le da al método de Monte Carlo una ventaja considerable en eficiencia computacional cuando es aplicado a problemas donde el número de variables m es grande.

El método de Monte Carlo incorpora cuatro desarrollos históricos distintos:

- Primero, los juegos de azar motivaron a los matemáticos del siglo XVII y XVIII a desarrollar la teoría de probabilidad, asignando una posibilidad de gansar un juego como el resultado de secuencias sucesivas de eventos aleatorios.

- Observando que el promedio de una función de variables aleatorias continuas tomaba la forma de una integral, los matemáticos del siglo XIX y de principios del siglo XX reconocieron, que en principio, uno puede tomar números al azar, transformarlos de acuerdo a reglas específicas y obtener una aproximación a la integral que resolvía un problema carente de contenido probabilístico. Lord Rayleigh en 1899 mostró que una caminata al azar en una dimensión daba la solución a una ecuación diferencial parabólica. Courant, en 1928, Kolmogorov en 1931 y Petrowski en 1933, contribuyeron a la conexión de las caminatas al azar y la resolución de ecuaciones diferenciales.

- Durante el desarrollo de la energía atómica en la posguerra, los científicos necesitaban resolver problemas de difusión de neutrones, y en general de transporte en un medio isotrópico. Debido a que se había establecido previamente que las ecuaciones diferenciales que describen a estos fenómenos tienen análogos en procesos estocásticos, John von Neumann y Stalinslaw Ulam sugirieron experimentos de muestreo en los que los números aleatorios generados en las computadoras digitales recientemente creadas, encontraron soluciones a varios problemas relacionados con las bombas atómicas.

El uso de los métodos de Monte Carlo se extendió posteriormente a varias disciplinas, entre ellas la química, la ingeniería, la investigación de operaciones y la física.

La simplicidad del método es la base para varios refinamientos interesantes. En particular, von Neumann, Ulam y otros se dieron cuenta que se podía modificar el método de Monte Carlo estándar de manera que produjera una solución al problema original con barras de error especificadas previamente. Muchas de estas *técnicas de reducción de la varianza* nacieron del desarrollo de los cálculos de Monte Carlo.

Este método fue bautizado "Monte Carlo" por Metrópolis y Ulam [63], en referencia a los juegos de azar jugados en los casinos de la ciudad de Monte Carlo.

A continuación definiremos los conceptos arriba mencionados y daremos las bases para poder responder la siguiente pregunta: ¿Qué es un método Monte Carlo?

3.2. Variables aleatorias

Al tirar un dado, sabemos que pueden aparecer los números 1,2,3,4,5 ó 6, pero no podemos determinar que número saldrá después de una tirada, debido a que la predicción del evento requeriría el conocimiento de una infinidad de condiciones iniciales y relaciones dinámicas que desconocemos *a priori*, en este sentido el número de puntos que saquemos al arrojar un dado es una variable aleatoria, y 1,2,3,4,5 ó 6 son los valores posibles, cada uno de éstos resultados posibles tiene una probabilidad de ocurrencia de $\frac{1}{6}$.

Definición 3.1. *Se llama aleatoria o estocástica a la magnitud que como resultado de un experimento toma uno y solamente un valor posible, de antemano desconocido y dependiente de causas fortuitas que previamente no se pueden tomar en cuenta. (informal)*

3.2.1. Variables aleatorias discretas

Cuando un evento aleatorio ξ puede tener un valor restringido a un conjunto finito de valores S, podemos definir completamente este conjunto mediante una tabla con la

siguiente forma:

$$\xi = \begin{bmatrix} x_1 & x_2 & x_3 & \dots & x_n \\ \Uparrow & \Uparrow & \Uparrow & & \Uparrow \\ p_1 & p_2 & p_3 & \dots & p_n \end{bmatrix} \tag{3.1}$$

donde p_i son las probabilidades de que la variable ξ obtenga el valor x_i. La tabla anterior es conocida como la *distribución* de la variable aleatoria *discreta*. Podemos escribir esto también de la siguiente manera:

$$p_i \leftarrow \mathcal{P}(\xi = x_i) \tag{3.2}$$

donde las probabilidades p_i deben de cumplir con dos condiciones:

$$p_i > 0 \quad \forall x_i \in S \tag{3.3}$$

y la que llamaremos condición de *normalización* de la probabilidad:

$$\sum_{i=1}^{n} p_i = 1 \tag{3.4}$$

Esperanza matemática y varianza

Bueno, sabemos que podemos definir completamente una variable aleatoria conociendo su distribución (3.1), pero en muchos casos ésta es desconocida, y debemos conformarnos con otras cantidades que la describan parcialmente, entre las mas importantes están la esperanza matemática y la varianza.

Se denomina esperanza matemática de la variable x_i a:

$$\mathcal{M}(\xi) = \sum_{i=1}^{n} x_i p_i \tag{3.5}$$

la esperanza matemática es el al valor medio de la variable aleatoria y tiene las siguientes propiedades:

Propiedades 3.1. *Dadas las variables aleatorias ξ y η, y una constante c, se cumple que:*

$$\mathcal{M}(\xi + c) = \mathcal{M}(\xi) + c$$

$$\mathcal{M}(c\xi) = c\mathcal{M}(\xi) \tag{3.6}$$

$$\mathcal{M}(\xi + \eta) = M(\xi) + \mathcal{M}(\eta)$$

y si ξ y η son independientes:

$$\mathcal{M}(\xi\eta) = \mathcal{M}(\xi)\mathcal{M}(\eta) \tag{3.7}$$

La varianza es una propiedad importante en toda distribución aleatoria, la cual nos indica el grado de dispersión de los valores posibles de una magnitud aleatoria alrededor de su valor medio.

Definición 3.2. *La varianza de una distribución aleatoria es la esperanza matemática del cuadrado de la desviación de la variable aleatoria ξ con respecto a su valor medio $\mathcal{M}(\xi)$:*

$$\mathcal{D}(\xi) = \mathcal{M}[\xi - \mathcal{M}(\xi)]^2 \tag{3.8}$$

Otra forma de expresar la varianza es:

$$\mathcal{D}(\xi) = \mathcal{M}(\xi^2) - (\mathcal{M}(\xi))^2 \tag{3.9}$$

Si consideramos la suma de dos variables independientes, la varianza de ésta será:

$$\mathcal{D}(\xi + \eta) = \mathcal{D}(\xi) + \mathcal{D}(\eta). \tag{3.10}$$

3.2.2. Variables aleatorias continuas

Si una variable aleatoria ξ puede tomar cualquier valor en $\Re$, se dice que ξ es una variable aleatoria continua. En este caso, la distribución de probabilidad será:

$$F(x) = \mathcal{P}(\xi \leq x) \qquad x \in \Re \tag{3.11}$$

donde F es una función no decreciente de x que cumple con:

$$F(-\infty) = 0, \qquad F(\infty) = 1$$

Si existe una función no negativa $f(x)$ tal que pueda escribirse de la forma:

$$F(x) = \int_{-\infty}^{x} f(x')\,dx' \tag{3.12}$$

entonces,

$$f(x) = \frac{dF(x)}{dx} \tag{3.13}$$

podemos interpretar a $f(x)$ como una probabilidad si la escribimos de la siguiente manera:

$$f(x)\,dx = \mathcal{P}[\xi \in (x, x + dx)] \tag{3.14}$$

Basados en lo anterior, podemos decir que las variables aleatorias continuas quedan definidas si se conoce su intervalo (a,b) y una función $f(x)$ que llamamos *densidad de probabilidad* [1] de la variable aleatoria ξ. Por analogía con las variables aleatorias discretas $f(x)$ debe de cumplir con:

[1] A $f(x)$ también se le conoce como la densidad de la distribución de ξ.

- La condición de no negatividad:

$$f(x) \geq 0 \tag{3.15}$$

- La condición de normalización:

$$\int_a^b f(x)\, dx = 1 \tag{3.16}$$

La esperanza matemática para este caso es:

$$\mathcal{M}(\xi) = \int_a^b x\, f(x)\, dx \tag{3.17}$$

Cabe mencionar que las propiedades (3.1) se cumplen también para las variables continuas. Sólo nos queda una pregunta que contestar, ¿Cómo sería la esperanza matemática para una función de ξ? La respuesta es sencilla tomando en cuenta que $g(\xi)$ es a su vez un número aleatorio. La esperanza matemática para una función $g(x)$ es:

$$\mathcal{M}(g(\xi)) = \int_a^b g(\xi)\, f(x)\, dx, \tag{3.18}$$

donde ξ tiene como densidad de probabilidad a $f(x)$. Regresaremos a la ecuación anterior cuando estudiemos métodos de integración por Monte Carlo.

La distribución gaussiana

Un ejemplo muy importante de densidad de probabilidad continua es la distribución gaussiana , la cual se define de la siguiente manera:

Definición 3.3. *Una variable aleatoria ξ tiene una distribución gaussiana si su intervalo posible de valores es ξ $(-\infty, \infty)$ y su densidad de probabilidad es:*

$$P_\xi(x) = \mathcal{P}(\xi = x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left[-\frac{(x-\mu)^2}{2\sigma^2}\right] \tag{3.19}$$

donde:

$$\mu = \mathcal{M}(\xi) \tag{3.20}$$

y:

$$\sigma^2 = \mathcal{D}(\xi). \tag{3.21}$$

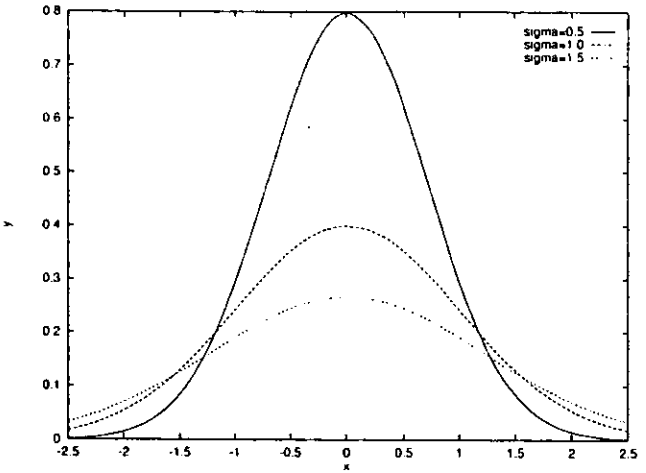


Figura 3.1.: Distribución gaussiana con distintos valores de σ.

El parámetro μ desplaza la gráfica en el eje de las x, mientras que σ conocida como *desviación cuadrática media*[2], hace que la curva tienda a aplanarse. (ver figura 3.1).

Esta densidad la encontramos en muchas aplicaciones físicas y matemáticas y además juega un papel importante en la estimación de los errores en los métodos Monte Carlo, pero ¿por qué?

La respuesta la dio el matemático ruso A. M. Liapunov, en el teorema del límite central de la teoría de probabilidades [79]:

Teorema 3.1. *Sean n variables aleatorias $\xi_1, \xi_2, \ldots, \xi_n$, independientes y con la misma distribución de probabilidades, y proponiendo:*

$$\mathcal{M}(\xi_1) = \mathcal{M}(\xi_2) = \ldots = \mathcal{M}(\xi_n) = m$$
$$\mathcal{D}(\xi_1) = \mathcal{D}(\xi_2) = \ldots = \mathcal{D}(\xi_n) = b^2$$

$$\chi = \xi_1 + \xi_2 + \ldots + \xi_n$$

entonces si la influencia de cada una de ellas en la suma es despreciable, ν, es una variable aleatoria con distribución gaussiana ($\mu = Nm, \sigma^2 = Nb^2$) y n es grande :

$$\mathcal{P}(a < \chi < b) \approx \int_a^b P_\nu(x)\,dx$$

[2]También conocida como desviación estándar

El teorema anterior nos dice que si una magnitud aleatoria (χ), es la suma de un número muy grande de magnitudes aleatorias mutuamente independientes, esta tendrá una distribución próxima a la gaussiana.

3.3. Generación de variables aleatorias uniformes

Una variable aleatoria es uniforme cuando el evento tiene la misma probabilidad en cualquier punto de un intervalo (a,b). Es decir, la densidad de probabilidad tiene la forma:

$$f(x) = \begin{cases} \frac{1}{(b-a)} & \text{si } a \leq x \leq b, \\ 0 & \text{en el resto.} \end{cases} \tag{3.22}$$

a este tipo de variables aleatorias las vamos a denotar como $\mathcal{U}_{a,b}$.

3.3.1. Métodos para generar variables aleatorias uniformes

Una manera *natural* de generar variables aleatorias es realizar un evento azaroso en la vida real; por ejemplo, tirar un dado, o meter 10 bolitas numeradas del 0 al 9 y sacarlas de una bolsa. La tabla de distribuciones para el ejemplo de la bolita sería:

$$\xi = \begin{bmatrix} 0 & 1 & 2 & \dots & 9 \\ \Uparrow & \Uparrow & \Uparrow & & \Uparrow \\ 0.1 & 0.1 & 0.1 & \dots & 0.1 \end{bmatrix} \tag{3.23}$$

Podemos repetir el experimento anterior y crear tablas de números aleatorios[3]. Los primeros estudios estocásticos se realizaban de esa manera.

Tiempo después se crearon máquinas especializadas que utilizaban el ruido de lámparas electrónicas para generar cifras en sistema binario, que después se agrupaban y se cambiaban a base decimal. Pero el construir máquinas especializadas para estudios estocásticos no es muy rentable. Hoy en día existen periféricos para computadoras que generan números aleatorios.

Cuando la velocidad de las computadoras aumentó se estudió la posibilidad de generar números *pseudoaleatorios* a partir de una fórmula que imitara las propiedades de las variables aleatorias *reales*. Pero el crear variables aleatorias en un instrumento determinista como lo es una computadora, no es una tarea fácil.

Cada experimento de Monte Carlo depende de un método que lo provea de secuencias de números aleatorios o en su defecto *pseudoaleatorios* que cumplan con las siguientes propiedades:

[3]Una de las mas grandes que ha existido fue realizada en 1955 por la *RAND Corporation* con ayuda de una ruleta electrónica. Tenía 1,000,000 de dígitos con una densidad de probabilidad como la de la ecuación (3.23) y 100,000 con distribución gaussiana.

- Al seleccionar arbitrariamente subconjuntos de dicha secuencia, éstos *aparenten* ser secuencias independientes estadísticamente hablando.

- Además, dichas subsecuencias deben de *aparentar* ser tomadas de una distribución uniforme $\mathcal{U}$.

La palabra *aparentar* toma una importancia extrema, pues del error de los generadores de números aleatorios puede depender la validez de algunas simulaciones.

Los factores que se comparan entre distintos algoritmos para la generación de números pseudoaleatorios [31] son:

La calidad de los números generados: Para una muestra de tamaño finito, uno desea que el error en la generación de los números *pseudoaleatorios*, $\epsilon_{\mathcal{U}}$ sea mucho menor al error en la simulación ϵ_S:

$$\epsilon_{\mathcal{U}} << \epsilon_S. \tag{3.24}$$

Existen muchas pruebas para encontrar la calidad numérica y estimar el error $\epsilon_{\mathcal{U}}$ de los generadores de números aleatorios [31]. Entre ellas se encuentran la distancia entre distintos puntos, el número de hiperplanos generados[4], pruebas de discrepancia, pruebas espectrales, etc.

El tiempo de cómputo: En general queremos un compromiso entre la cantidad de números aleatorios generados por unidad de tiempo y la calidad de nuestro algoritmo. No sirve de nada tener un excelente generador de números aleatorios con respecto a la eficiencia numérica, si su velocidad es limitada.

La facilidad de uso: Un generador de números aleatorios debe de tener una interfase fácil de usar. El usuario debe de tener control sobre el generador, pero estar aislado de su funcionamiento interno[5].

La portabilidad: El generador de números aleatorios que escojamos debe poder ser utilizable en distintas arquitecturas de cómputo sin modificaciones.

En el programa MOISS, descrito en el capítulo 7, usamos la biblioteca de generadores de números aleatorios GNU Scientific Library [34], que provee de una veintena de generadores de distintas calidades numéricas y velocidades computacionales. La documentación incluida con la biblioteca especifica las características particulares de cada generador.

[4] En general, los números aleatorios quedan alineados en hiperplanos en un espacio k dimensional.

[5] A diferencia de la creencia de que es mejor "programar" nuestro propio generador de números aleatorios sin basarnos en la experiencia previa de otras personas que ya probaron y optimizaron otro previamente

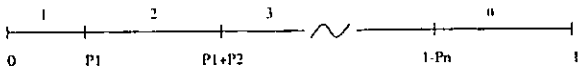


Figura 3.2.: Conjunto de variables aleatorias discretas con distribución arbitraria sobre un eje.

3.4. Generación de variables aleatorias no uniformes

Ya discutidas las herramientas para obtener números aleatorios continuos, ahora trataremos los distintos métodos para muestrear una variable aleatoria continua de una densidad de probabilidades arbitraria. Primero definamos lo que es muestrear una distribución.

Definición 3.4. *Dada una densidad de probabilidades $f(x)$, muestrear dicha distribución significa producir mediante un algoritmo ciertas variables aleatorias $\eta_1, \eta_2, \eta_3, \ldots$ tales que sus probabilidades sean:*

$$\mathcal{P}(a < \eta < b) = \int_a^b f(x)dx \leq 1 \qquad x \in (a, b)$$

Al muestrear una variable partiremos de las variables aleatorias generadas por los algoritmos de la sección anterior, es decir partiremos de una distribución uniforme entre cero y uno, $\mathcal{U}_{0,1}$.

3.4.1. Técnicas de inversión

Si tenemos 3 eventos $\{a, b, c\}$ con probabilidades 0.2, 0.3 y 0.5 respectivamente, se puede definir la siguiente distribución:

$$f(x) = \begin{cases} a & 0 \leq x \leq 0.2 \\ b & 0.2 < x \leq 0.5 \\ c & 0.5 < x \leq 1 \end{cases}$$

Es evidente que para muestrear esta distribución debemos hacer lo siguiente: Si $\mathcal{U}_{0,1}$ es menor que 0.2 entonces a, si $\mathcal{U}_{0,1}$ es mayor que 0.2 y menor que 0.5 entonces b, y así sucesivamente. Si generalizamos lo anterior, el evento i ocurrirá cuando:

$$\sum_{j=0}^{i-1} p_j \leq \mathcal{U}_{0,1} \leq \sum_{j=0}^{i} p_j. \tag{3.25}$$

Extendiendo lo anterior a variables continuas η con densidades de distribución $f(x)$ y en un intervalo (a, b), la distribución de probabilidad se convierte en:

$$F(x') = \int_a^{x'} f(x)\, dx.$$

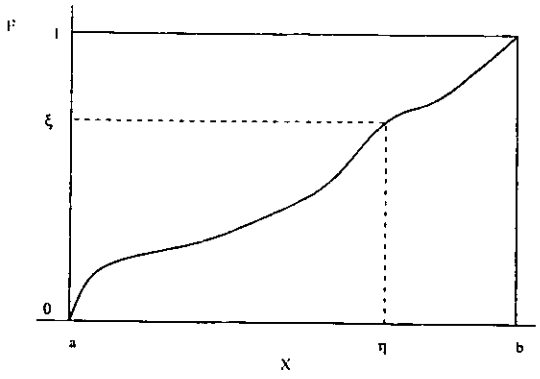


Figura 3.3.: Conjunto de variables aleatorias discretas con distribución arbitraria sobre un eje

en la cual como se puede ver en la figura 3.3, cualquier recta $y = \xi$ donde $0 < \xi < 1$, corta a F en un punto único, es decir, la ecuación:

$$\nu = \int_a^{\xi} f(x)\,dx. \tag{3.26}$$

tiene una solución única para ν,la cual queda en función de ξ y tiene la densidad de probabilidad de $f(x)$.Debemos recordar que ξ tiene densidad de probabilidad $\mathcal{U}_{0,1}$.

Por ejemplo, supongamos que queremos muestrear la densidad de distribución $p = x^2$ en el intervalo $(\frac{1}{2}, \frac{1}{2} \cdot 5^{(2/3)})$, donde se cumplen las condiciones (3.15) y (3.16):

$$\int_{\frac{1}{2}}^{\frac{1}{2}\cdot 5^{(2/3)}} x^2 dx = 1.$$

El primer paso es hacer un cambio de variable e integrar:

$$\int_{\frac{1}{2}}^{x'} x^2\,dx = \frac{1}{3}x'^3 - \frac{1}{24} = \mathcal{U}_{0,1}.$$

El resultado anterior es igualado a $\mathcal{U}_{0,1}$,y se despeja x' para quedarse sólo con la raíz real. Esta raíz x' es nuestra variable aleatoria η muestreada de $f(x)$:

$$\eta_p = \frac{1}{2}(1 + 24\,\mathcal{U}_{0,1})^{\frac{1}{3}}. \tag{3.27}$$

A este método de muestreo se le conoce como de transformadas inversas, y es el camino más directo para generar un muestreo, podemos resumirlo en:

Teorema 3.2. *Sea* $\{f(x); a \leq x \leq b\}$, *una distribución, con una densidad de distribución inversa:*

$$f^{-1}(u) = \inf\{x \in [a,b] : f(x) \leq u; 0 \leq u \leq 1\}.$$

Entonces $\eta = f^{-1}(\mathcal{U}_{0,1})$ *tiene la misma densidad de probabilidad que* f.

3.4.2. Métodos de aceptación-rechazo

Aunque cada distribución de probabilidad tiene una representación inversa, y por lo tanto podemos usar el método de inversión descrito previamente, muchas veces esta inversa no existe en una forma conveniente para calcularla.

El método de aceptación y rechazo nos permite muestrear una función sin conocer su función inversa aprovechando las propiedades de la probabilidad condicional.

Supongamos que queremos conocer una densidad de probabilidades $w(y)$, y que conocemos otra $h(y)$ para la que si conocemos un generador. Además, se cumple que para todo punto, $h(y) \geq w(y)$ (ver la figura 3.4).

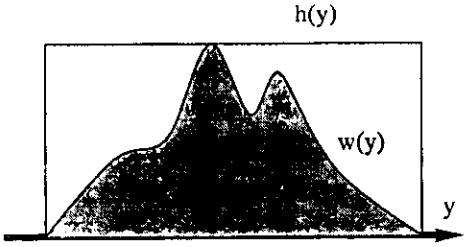


Figura 3.4.: Esquematización de las distribuciones $h(y)$ y $w(y)$ en el algoritmo de rechazo.

Si generamos un número aleatorio y a partir de la distribución uniforme U_d donde d es el dominio de la distribución, y hacemos que:

$$p(y) = \frac{w(y)}{h(y)} \leq 1. \tag{3.28}$$

Aceptamos al valor y como miembro de la distribución $w(y)$ si $p > \xi$, donde ξ es un número aleatorio proveniente de $U_{0,1}$.

Una desventaja de este método es que si $w(y)$ y $h(y)$ son muy distintas, su diferencia de áreas (que es proporcional al número de intentos rechazados) es considerable, disminuyendo la eficiencia del método.

3.5. Integración por métodos de Monte Carlo

Supongamos que queremos calcular el valor de la siguiente integral:

$$I = \int_a^b g(x)dx. \tag{3.29}$$

Geométricamente, el valor de I es el área bajo la curva, en este hecho se basan muchos métodos numéricos como la aproximación del trapezoide o la regla de Simpson.

Definamos otra función $f(x)$ tal que

$$f(x) = \begin{cases} \frac{1}{(b-a)} & \text{si } a \leq x \leq b, \\ 0 & \text{en el resto.} \end{cases} \tag{3.30}$$

Por tanto, podemos representar a la ecuación anterior como:

$$I = \frac{1}{(b-a)} \int_a^b g(x) f(x)\, dx. \tag{3.31}$$

Como $f(x)$ cumple con las condiciones para ser una distribución de probabilidad, usamos la ecuación (3.18) para obtener:

$$\int_a^b g(x) f(x)\, dx = \mathcal{M}[g(f \mapsto \xi)],$$

donde el signo $\mapsto$ significa *mapeado de*.

Entonces podemos encontrar el valor aproximado de I calculando el promedio de $g(x)$ en el intervalo [a,b]:

$$I \approx (b-a) \frac{1}{N} \sum_{i=1}^{N} g(\xi_i) \tag{3.32}$$

donde ξ_i es un número aleatorio uniforme definido en el intervalo (a,b) $(\mathcal{U}_{a,b})$ y N es el número de iteraciones.

Los métodos de integración por Monte Carlo son eficientes comparados con otros métodos numéricos cuando se evalúan integrales multidimensionales. Si queremos evaluar una integral n-dimensional de la forma:

$$I = \int_\Omega g(\mathbf{R})\, d\mathbf{R}, \tag{3.33}$$

y los límites de integración Ω no dependen de R, entonces la ecuación (3.32) se representa como:

$$I \approx \prod_{i=1}^{n} (b_i - a_i) \frac{1}{N} \sum_{i=1}^{N} g(\Xi_i) \tag{3.34}$$

donde Ξ es un vector n-dimensional de variables aleatorias independientes:

$$\Xi = \{\xi_1, \xi_2, \ldots, \xi_n\} \tag{3.35}$$

Un ejemplo ilustrativo es el cálculo de π por medio de integración por métodos de Monte Carlo. La primera aplicación del método de Monte Carlo de la que se tiene noticia fue realizada por el conde G. N. N. Buffon en 1777. (ver la tabla 3.1). En esta aplicación, la fuente de números aleatorios fue *natural*, ya que el lanzaba agujas a su parquet y su servidumbre evaluaba el resultado.

Algoritmo 1 Integra I. Integración por métodos Monte Carlo

Salida: $I = \int_a^b g(x)\,dx$
  N {Número de pasos}
  a,b {Límites de integración}
  for i = 1, ..., N do
    $\xi_i \leftarrow a + \mathcal{U}_{0,1}(b-a)$
    sum $\leftarrow$ sum + $g(\xi_i)$
  end for
  I $\leftarrow$ (b − a)sum/N

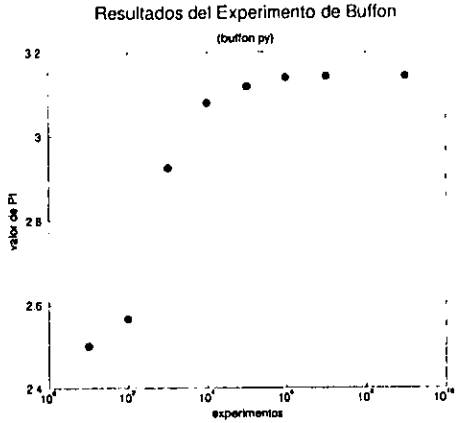


Figura 3.5.: Resultados del experimento de Buffon. Se grafica el valor de π obtenido a partir de la simulación por computadora con respecto al logaritmo del número de pasos calculados.

3.5.1. Disminución de la varianza

Podemos disminuir la varianza del algoritmo anterior mediante una técnica conocida como muestreo inducido (*importance sampling*). La idea consiste en concentrar la cantidad de muestras aleatorias en lugares donde la distribución sea grande o tenga una forma complicada.

Supongamos que tenemos una distribución de probabilidad arbitraria $P_\xi(x)$ definida en el intervalo (a,b). A esta distribución le llamaremos distribución de muestreo inducido. Podemos crear otra variable aleatoria (η) de la forma:

$$\eta = \frac{g(\xi)}{P_\xi(\xi)} \tag{3.37}$$

y recordando la ecuación (3.18), sabemos que la esperanza matemática de η será igual

al valor de la integral (3.29):

$$\mathcal{M}(\eta) = \mathcal{M}\left[\frac{g(x)}{P_t(x)}\right] = I$$

Integrando:

$$\mathcal{M}(\eta) = \int_a^b \left[\frac{g(x)}{P_t(x)}\right] P_t(x)\, dx = I \tag{3.38}$$

La varianza tiene la forma:

$$\mathcal{D}(\eta) = \mathcal{M}(\eta^2) - I^2 = \int_a^b \left[\frac{g(x)^2}{P_t(x)}\right] dx - I^2 \tag{3.39}$$

¿Cómo debemos escoger la distribución de muestreo inducido para que la integración por el método de Monte Carlo tenga una varianza menor y sea mas eficiente? Para responder esto partiremos de la desigualdad de Cauchy-Schwarz (ver (A.1)):

$$\left[\int_a^b |u(x)v(x)|dx\right]^2 \leq \int_a^b u(x)^2 dx \int_a^b v(x)^2 dx \tag{3.40}$$

Si se hacen las siguientes sustituciones:

$$u = \frac{g(x)}{\sqrt{P_t(x)}}$$

$$v = \sqrt{P_t(x)}$$

y se aplica la condición de normalización (3.16), se obtiene:

$$\left[\int_a^b |g(x)|dx\right]^2 \leq \int_a^b \frac{g(x)^2}{P_t(x)} dx. \tag{3.41}$$

Las ecuaciones (3.39) y (3.41) son la base del siguiente teorema:

Teorema 3.3. *Al estimar el valor de la integral definida* (3.29) *se tiene una varianza mínima:*

$$\mathcal{D}(\eta)_{\min} = \left(\int_a^b |g(x)|\, dx\right)^2 - I^2 \tag{3.42}$$

cuando la distribución de muestreo inducido es:

$$P_t(x) = \frac{|g(x)|}{\int_a^b |g(x)|dx} \tag{3.43}$$

Entre mas parecida sea $P_t(x)$ a la función que queremos integrar, la varianza será menor. Encontramos dos corolarios interesantes:

- La varianza disminuye cuadráticamente conforme $P_t(x)$ se vuelve mas parecido a $P(x)$.
- En el límite en que ambas funciones son iguales, la varianza es cero.

Para evaluar la integral (3.29) muestreamos ξ de la distribución P_t,ya sea por el método de inversión o por el método de aceptación-rechazo, y calculamos la media de (3.37):

$$I \approx \frac{1}{N}\sum_{i=1}^{N}\eta_i = \frac{1}{N}\sum_{i=1}^{N}\frac{g(\xi_i)}{P_t(\xi_i)}. \tag{3.44}$$

Algoritmo 2 Integra II. Integración por métodos MC con muestreo inducido

Salida: $I = \int_a^b g(x)\,dx$
  N {Número de pasos}
  h() {Densidad de Dist. Inv. de $f()$; Con intervalo a, b}
  f() {Función de prueba del muestreo inducido}
  sum $\leftarrow$ 0
  for i = 1, ..., N do
    $\xi \leftarrow$ h($\mathcal{U}_{0,1}$)
    sum $\leftarrow$ sum + g(ξ)/f(ξ)
  end for
  I $\leftarrow$ sum/N

Un buen ejemplo es la evaluación de la siguiente integral:

$$I = \int_0^{\frac{\pi}{2}} \text{sen}(x)dx = -\cos(x)\Big|_0^{\frac{\pi}{2}} = 1,$$

propondremos 3 distribuciones de muestreo inducido:

$$\begin{aligned} P_t^a(x) &= \frac{2}{\pi}, \\ P_t^b(x) &= 8\frac{x}{\pi^2}, \\ P_t^c(x) &= 3\frac{\sqrt{2}\sqrt{x}}{\pi^{3/2}}. \end{aligned}$$

Mediante la técnica de inversión podemos muestrear las siguientes distribuciones a partir de $\mathcal{U}_{0,1}$,

$$\begin{aligned} \xi^a &= \frac{1}{2}\mathcal{U}_{0,1}\,\pi \\ \xi^b &= \frac{1}{2}\,\pi\,\sqrt{\mathcal{U}_{0,1}} \\ \xi^c &= \frac{1}{4}\,2^{2/3}\left(\sqrt{2}\pi^{3/2}\mathcal{U}_{0,1}\right)^{2/3} \end{aligned}$$

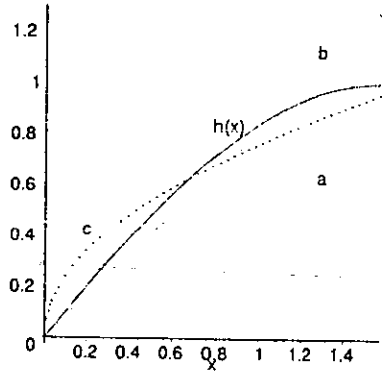


Figura 3.6.: Comparación de $g(x) = \mathrm{sen}(x)$ y las distintas distribuciones de prueba $\{a, b, c\}$.

Como puede verse en la tabla 3.2, las funciones de muestreo inducido b y c resultaron ser mas eficientes, dando un error real (δ_r) menor con menos pasos (N), y teniendo una varianza mucho menor que la funcion de muestreo inducido a.

3.6. Estimación de errores

Los métodos Monte Carlo nos dan respuestas aproximadas, cuya precisión depende de varios factores, los cuales incluyen el tiempo de la simulación, la desviación de inicialización y la autocorrelación al equilibrio[80].

Int(sen(x),x=0..1); por Monte Carlo con muestreo inducido.
(Realizada en MAPLE)

```
> a:=x->stats[random,uniform[0,1]](1);
```

$$a := x \rightarrow stats_{random,\, uniform_{0,1}}(1)$$

```
> Pt:=x->8*x/Pi**2;
> eta:=x->(Pi*sqrt(x)/2);
```

$$Pt := x \rightarrow 8\,\frac{x}{\pi^2}$$

$$\eta := x \rightarrow \frac{1}{2}\,\pi\,\sqrt{x}$$

```
> st := time():
> N:=100;suma:=0:suma2:=0:
> for i from 1 to N do rnd:=a():
t1:=sin(eta(rnd))/Pt(eta(rnd)); suma :=suma+t1:
suma2:=suma2+t1**2:  od:
> resultado:=evalf(suma/N);
> var:=evalf((1/N*(suma2-(suma*suma/N))));
> sd:=sqrt(var);
> errPROB:=0.675*sqrt(var/N);
> errREAL:=abs(1-resultado);tiempo:=time() -
st;
```

$$N := 100$$
$$resultado := .9959766477$$
$$var := .01712905076$$
$$sd := .1308779995$$
$$errPROB := .008834264966$$
$$errREAL := .0040233523$$
$$tiempo := .901$$

Despues de realizar una simulación, disponemos solamente de un conjunto de datos. Sabemos por el teorema del valor medio, (sección 3.1) que si estos datos son independientes, seguirán una distribución gaussiana. Por lo tanto, para poder estudiar el carácter cuantitativo de este conjunto hay que estimar la esperanza matemática (μ) y la desviación cuadrática media (σ); debido a que ambos parámetros determinan a la distribución gaussiana.

¿Cómo le hacemos para estimar estos parámetros? Para poder contestar debemos definir a la *media general*:

Definición 3.5. *Se llama media general* $\overline{s}$ *de un conjunto* $S = \{x_1, x_1, \ldots, x_n\}$, *a la media aritmética de los valores de* S:

$$\overline{s} = \frac{x_1 + x_1 + \cdots + x_n}{N}$$

donde N es el numero de elementos de S.

Ahora supongamos que de nuestro conjunto de datos (S) de tamaño n, con valores $\{x_1, x_1, \ldots, x_n\}$, extraemos al azar un objeto, la probabilidad de obtener un dato con valor x_i es $\frac{1}{n}$. Entonces:

$$\mathcal{M}_m(S) = \sum_i^n x_i \cdot \frac{1}{n} \equiv \frac{x_1 + x_1 + \cdots + x_n}{N} = \overline{x}. \tag{3.45}$$

De este modo, si al conjunto de valores posibles S se le considera como una magnitud aleatoria, la esperanza matemática de la muestra es igual a la media general del conjunto.

De la definición 3.2 y utilizando la ecuación anterior encontramos que la varianza muestral de un conjunto S es:

$$\mathcal{D}_m(S) = \overline{x^2} - \overline{x}^2 \tag{3.46}$$

donde:

$$\overline{x} = \frac{\sum_i^n x_i}{n}, \quad \overline{x^2} = \frac{\sum_i^n x_i^2}{n}.$$

A la raíz cuadrada de la varianza muestral se le conoce como desviación cuadrática media muestral o desviación estándar.

$$\sigma_m(S) = \sqrt{\mathcal{D}_m(S)}. \tag{3.47}$$

3.6.1. Error probable

Se puede comprobar que para cualquier σ y μ,

$$\int_{a-r}^{a+r} f_\xi(x)\, dx = 0.5, \tag{3.48}$$

Cuando el valor de r es 0.675 σ, en el dominio de las probabilidades:

$$\mathcal{P}(|\xi - a| < r) = \mathcal{P}(|\xi - a| > r) = 0.5. \tag{3.49}$$

Es decir, las desviaciones mayores que r y las desviaciones menores que r son igualmente probables. A la magnitud r se le conoce como error probable y generalmente es del mismo orden de magnitud que el error real[6].

[6] Ver resultados de la tabla 3.2.

3.7. Caminatas aleatorias

¿Cómo simular una partícula coloidal suspendida en un líquido? Al parecer esta partícula presenta un movimiento completamente aleatorio[7], es decir, la trayectoria que tomará esta partícula en un instante dado no depende de las posiciones anteriores. Matemáticamente este proceso se modela con las *cadenas de Markov*. En una cadena de Markov, el sistema cambia de estado aleatoriamente siguiendo una *regla de transición*, $\mathcal{P}_{s\to s+1}$, describiendo lo que conocemos como una caminata aleatoria (*random walk*) en el espacio de las configuraciones S=$\{s_0, s_1, \ldots, s_n\}$.

Definición 3.6 (Cadena de Markov). *La secuencia de variables aleatorias en* $S = (X_n)_{n\in\mathrm{N}}$, *es llamada cadena de Markov homogénea , si para cada una de las* $n \in \mathrm{N}$ *se cumple que*

$$P[X_{n+1} = j | X_0 = i_0, \ldots, X_n = i_n] = P[X_{s+1} = j | X_n = i_n] = p_{i_n j} \tag{3.50}$$

para todas las $(i_0, \ldots, i_n, j) \in S^{n+2}$, *y además:*

$$P[X_0 = i_0, \ldots, X_n = i_n] > 0.$$

La primera identidad en la ecuación (3.50) es llamada "Propiedad de Markov", y define la *memoria* o el *orden* de la cadena. En este caso, el orden es igual a uno porque las probabilidades de transición a un nuevo estado j están totalmente determinadas por el estado actual i [8].

La segunda identidad es la llamada condición de homogeneidad. Ésta asegura que las probabilidades de transición al avanzar las iteraciones es independiente de n [9]. Por lo tanto para construir una cadena de Markov se requiere conocer su distribución inicial y su regla de transición. Esta regla se puede representar por una matriz. Para definir la *matriz de transición* primero asignaremos a cada par de estados posibles $(i, j) \in S^2$, un número real con las siguientes propiedades:

$$p_{ij} \geq 0 \qquad \forall (i,j) \in S^2, \tag{3.51}$$

$$\sum_{j\in S} p_{ij} = 1 \qquad \forall i \in S. \tag{3.52}$$

[7] A este tipo de movimiento se le llama Browniano, por el botánico ingles Robert Brown que en 1827 lo caracterizó por primera vez observando una suspensión de polen.

[8] Por ejemplo: El estado del tiempo del dia de hoy (soleado,nublado, etc.) afecta al clima de mañana, pero el clima de ayer es irrelevante para el promostico de mañana

[9] En nuestra analogia climatica esto significaría que no existieran estaciones.

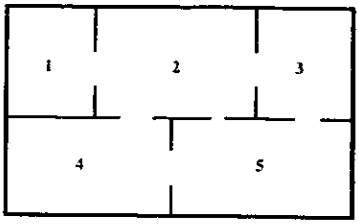


Figura 3.7.: Laberinto de la rata.

Entonces, podemos escribir la matriz de transición [10] como $\mathbf{P} = \overline{\{p_{ij}\}}$.

$$\begin{pmatrix} p_{11} & p_{21} & \cdots & p_{n1} \\ p_{12} & p_{22} & \cdots & p_{n2} \\ \vdots & \vdots & \ddots & \vdots \\ p_{1n} & p_{2n} & \cdots & p_{nn} \end{pmatrix} \tag{3.53}$$

Un ejemplo de caminata de Markov es la evolución de un juego de mesa como el *Turista*.

Por ejemplo supóngase que una rata se encuentra en el laberinto esquematizado por la figura 3.7. La rata cambia de cuarto aleatoriamente. Tomando en cuenta que es más probable que cambie de cuarto si la puerta es más grande, podemos escribir una matriz que represente la probabilidad de que la rata se mueva de un cuarto a otro. La secuencia $S = (X_n)_{n\in\mathbb{N}}$, donde X_n representa el cuarto donde se encuentra la rata, es determinada por la siguiente *matriz de transición*:

$$\mathbf{P}_{\text{rata}} = \begin{pmatrix} 0 & \frac{2}{7} & 0 & 0 & 0 \\ 1 & 0 & \frac{1}{2} & \frac{1}{2} & \frac{1}{5} \\ 0 & \frac{2}{7} & 0 & 0 & \frac{2}{5} \\ 0 & \frac{2}{7} & 0 & 0 & \frac{2}{5} \\ 0 & \frac{1}{7} & \frac{1}{2} & \frac{1}{2} & 0 \end{pmatrix} \tag{3.54}$$

A este tipo de matrices se les conoce como matrices de probabilidad de transición.

Ahora examinemos la forma de encontrar la probabilidad de estar en algún estado dado. En cualquier paso (n) de la caminata $p_{ij}^{(n)}$, la probabilidad es:

$$P[X_{n+m} = j | X_m = i] = p_{ij}^{(n)} \tag{3.55}$$

Además existe una condición de *balance detallado* o reversibilidad. La probabilidad de que la rata se mueva del cuarto i al cuarto j es la misma que la probabilidad de retorno del cuarto j al cuarto i:

$$p_{ij}^{(1)} = p_{ij}$$

[10] Este tipo de matrices, en las que todos sus elementos son positivos y la suma de sus columnas es uno, se conocen como matrices estocásticas. El uso de la forma transpuesta es algebraícamente mas conveniente.

Si representamos al producto de probabilidades como un vector $\mathbf{p}^{(n)}$, entonces la distribución en $n = 1$ será:

$$p_j^{(1)} = \sum_i p_{i,j} p_i^{(0)} \tag{3.56}$$

La expresión anterior es una multiplicación de matrices:

$$\mathbf{p}^{(1)} = \mathbf{P}\mathbf{p}^{(0)}, \tag{3.57}$$

por lo que podemos continuar realizando el producto:

$$\mathbf{p}^{(2)} = \mathbf{P}\mathbf{p}^{(1)} = \mathbf{P}^2\mathbf{p}^{(0)}, \tag{3.58}$$

y en general:

$$\mathbf{p}^{(n)} = \mathbf{P}^n\mathbf{p}^{(0)} \tag{3.59}$$

Si la rata del ejemplo anterior se encontraba inicialmente en el cuarto uno, podemos encontrar las probabilidades de que se encuentre en algún cuarto al paso n, realizando la siguiente operación:

$$\mathbf{p}_{\text{rata}}^{(n)} = \begin{pmatrix} 0 & \frac{2}{7} & 0 & 0 & 0 \\ 1 & 0 & \frac{1}{2} & \frac{1}{2} & \frac{1}{5} \\ 0 & \frac{2}{7} & 0 & 0 & \frac{2}{5} \\ 0 & \frac{2}{7} & 0 & 0 & \frac{2}{5} \\ 0 & \frac{1}{7} & \frac{1}{2} & \frac{1}{2} & 0 \end{pmatrix}^n \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} \tag{3.60}$$

Al aplicar la regla anterior con distintas distribuciones iniciales encontramos que, $\mathbf{p}^{(n)}$, llega a un estado de *equilibrio* (ver figura 3.8):

$$\mathbf{p}_{\text{rata}}^{(\infty)} \approx \begin{pmatrix} 0.90909091 \\ 0.31818182 \\ 0.18181818 \\ 0.18181818 \\ 0.22727272 \end{pmatrix} \tag{3.61}$$

Decimos que una caminata es *ergódica* cuando $\mathbf{p}^{(n)}$ converge a una distribución límite independientemente de la distribución inicial $\mathbf{p}^{(0)}$

$$\lim_{n\to\infty} \mathbf{p}^{(n)} = \mathbf{p} \tag{3.62}$$

Otras características de una caminata ergódica son:

1. Podemos pasar de un estado a otro del sistema, en un número finito de pasos.

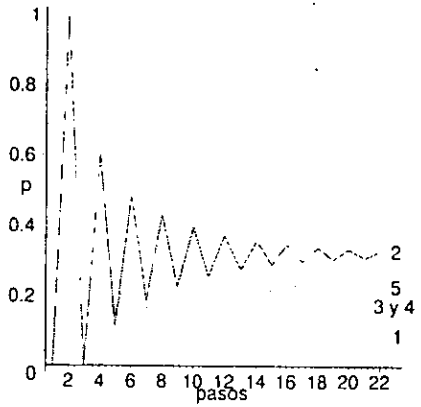


Figura 3.8.: Evolución de las probabilidades de encontrar a la rata en cada uno de los cuartos.

2. No es periódica.

3. El tiempo que tarda el sistema en regresar a un estado i es finito, pero sus periodos no son iguales [11]

4. (Teorema de Foster) Se puede encontrar una solución no nula de $\mathbf{x} = \mathbf{P}\mathbf{x}$ con $\sum |x_i| < \infty$

3.7.1. Evaluación de $\mathbf{P}^n$

Supongamos que tenemos una cadena de Markov con k posibles estados representada por la matriz de transición $\mathbf{P}$. Los valores propios de $\mathbf{P}$ son un conjunto de números (λ) que son raíces del polinomio característico:

$$p(\lambda) = \det(\mathbf{P} - \lambda \mathbf{I}) = 0. \tag{3.63}$$

Los vectores propios satisfacen las siguientes ecuaciones:

$$\mathbf{P}\mathbf{x}_i = \lambda_i \mathbf{x}_i \tag{3.64}$$

para los vectores propios por la izquierda $\mathbf{x}$, ó:

$$\mathbf{y}_i^t \mathbf{P} = \lambda_i \mathbf{y}_i^t \tag{3.65}$$

para los vectores propios por la derecha $\mathbf{y}$.

[11] Siempre y cuando el sistema sea finito, por ejemplo cuando tenemos condiciones a la frontera periódicas.

Si los k valores propios de P son distintos, entonces P es diagonalizable, es decir, existe una matriz diagonal D:

$$\mathbf{D} = \begin{pmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \lambda_k \end{pmatrix}, \tag{3.66}$$

tal que P es equivalente a D:

$$\begin{aligned} \mathbf{D} &= \mathbf{X}^{-1}\mathbf{P}\mathbf{X} \\ \mathbf{X}\mathbf{D} &= \mathbf{X}\mathbf{X}^{-1}\mathbf{P}\mathbf{X} \\ \mathbf{X}\mathbf{D}\mathbf{X}^{-1} &= \mathbf{P}\mathbf{X}\mathbf{X}^{-1} \\ \mathbf{X}\mathbf{D}\mathbf{X}^{-1} &= \mathbf{P}, \end{aligned} \tag{3.67}$$

donde X es una matriz cuyas columnas son los vectores característicos de P:

$$\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_k].$$

En el caso de los vectores propios por la derecha, tendríamos en lugar de la matriz X, una matriz $\mathbf{Y} = [\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_k]$, por tanto:

$$\mathbf{P} = (\mathbf{Y}^t)^{-1}\mathbf{D}\mathbf{Y}^t = \mathbf{X}\mathbf{D}\mathbf{X}^{-1}. \tag{3.68}$$

Relacionando las ecuaciones (3.64) y (3.65) obtenemos:

$$\mathbf{y}_j^t\mathbf{P}\mathbf{x}_i = \mathbf{y}_j^t(\lambda_i\mathbf{x}_i) = (\mathbf{y}_j^t\lambda_j)\mathbf{x}_i. \tag{3.69}$$

Si todos los valores propios son distintos, podemos escoger $\mathbf{x}_i$ y $\mathbf{y}_j^t$ de tal manera que:

$$\mathbf{y}_j^t \cdot \mathbf{x}_i = \begin{cases} 0 & \text{si } i \neq j \\ 1 & \text{si } i = j \end{cases} \tag{3.70}$$

la ecuación anterior la podemos escribir de la siguiente manera:

$$\mathbf{Y}^t\mathbf{X} = \mathbf{I}. \tag{3.71}$$

Partiendo de la ecuación (3.67):

$$\begin{aligned} \mathbf{P} &= \mathbf{X}\mathbf{D}\mathbf{X}^{-1} \\ \mathbf{P} &= \mathbf{X}\mathbf{D}\mathbf{I}\mathbf{X}^{-1} \\ \mathbf{P} &= \mathbf{X}\mathbf{D}\mathbf{Y}^t\mathbf{X}\mathbf{X}^{-1} \\ \mathbf{P} &= \mathbf{X}\mathbf{D}\mathbf{Y}^t. \end{aligned} \tag{3.72}$$

Siempre y cuando los vectores propios de P sean ortonormales, se puede ver que:

$$\mathbf{P} = \sum_{i=1}^{k} \lambda_i \mathbf{C}_i \tag{3.73}$$

donde:

$$\mathbf{C}_i = \mathbf{x}_i \mathbf{y}_i^t$$

es una matriz de $k \times k$ que satisface las siguientes relaciones:

$$\begin{aligned} &\mathbf{C}_i\mathbf{C}_j = 0 \quad \text{si } i \neq j \\ &\mathbf{C}_i\mathbf{C}_j = 1 \quad \text{si } i = j \\ &\sum_{i=i}^{k} \mathbf{C}_i = \mathbf{I} \end{aligned} \tag{3.74}$$

Partiendo de la ecuación (3.73) y tomando en cuenta las propiedades de C (ec. (3.74)), podemos escribir las potencias de P de la siguiente manera:

$$\mathbf{P}^n = \sum_{i=i}^{n} \lambda^n \mathbf{C}_i. \tag{3.75}$$

En forma más general si los vectores propios no están normalizados la ecuación anterior se transforma en:

$$\mathbf{P}^n = \sum_{i=i}^{n} c_j^{(-1)} \lambda^n \mathbf{C}_i \tag{3.76}$$

donde

$$c_j = \mathbf{y}_j^t \cdot \mathbf{x}_i \tag{3.77}$$

Esta expresión nos sirve para encontrar $\mathbf{P}^n$, a un precio computacionalmente más barato, siempre y cuando los valores propios no estén degenerados. Cuando los valores propios de la matriz de transición se encuentran degenerados la expresión para $\mathbf{P}^n$ se vuelve mas complicada:

$$\mathbf{P}^n = \sum_j \frac{1}{(r_j - 1)!} \left[\frac{d^{r_j-1}}{d\lambda^{r_j-1}} \frac{\lambda^n \text{adj}(\lambda \mathbf{I} - \mathbf{P})}{\phi_j(\lambda)} \right]_{\lambda=\lambda_j} \tag{3.78}$$

$$\phi_j(\lambda) = \prod_{i \neq j} (\lambda - \lambda_i)_i^r.$$

Para más información sobre esta expresión consultar [33].

3.7.2. Caminatas y ecuaciones integrales

Los métodos de Monte Carlo pueden ser clasificados en *estáticos* o *dinámicos*. Para resolver integrales de una dimensión mediana, los métodos estáticos (sección 3.5) son buenos. Al pasar a sistemas de dimensión grande [80], como las aplicaciones de mecánica cuántica o termodinámica estadística, se debe pensar en métodos *dinámicos* de integración. La idea de los métodos dinámicos es la de inventar un *proceso estocástico* con un espacio de estados S que tenga una única distribución de equilibrio π. Luego simulamos este proceso estocástico en la computadora, empezando de una configuración inicial arbitraria; ya que el sistema llega al equilibrio, medimos promedios "temporales", que convergen asintóticamente a los promedios de la distribución π. En términos físicos, estamos inventando una evolución estocástica del sistema. Hay que darse cuenta que esta evolución estocástica no corresponde a ninguna dinámica física real. Este proceso estocástico puede ser visto como una cadena de Markov, y las ecuaciones que se pueden integrar de esta manera son las ecuaciones integrales de Fredholm de segunda especie.

Considere la ecuación integral integral de Fredholm de segunda especie (A.20) (sección A.1.5):

$$y(x) - \lambda_n \int_a^b K(x,x')y(x')\,dx' = f(x)\lambda(\mathrm{x}) = v(\mathrm{x}) + \nu \int_\omega K(\mathrm{x},\mathrm{x}')\lambda(\mathrm{x}')d\mathrm{x}' \qquad (3.79)$$

donde la función $v(\mathbf{r})$ y el núcleo $K(\mathrm{x},\mathrm{x}')$ son funciones continuas y acotadas. El problema a resolver es encontrar los valores de una cantidad desconocida $\lambda(\mathrm{x})$ para un conjunto de valores $x \in \mathcal{R}$.

Esta representación y su análogo discreto aparecen en muchas aplicaciones de la física y la mecánica estadística. Por ejemplo, en la formulación del problema del transporte de neutrones, $v(\mathbf{r})$ representa la probabilidad de que un neutrón de energía inicial E_0, teniendo su primera colisión en x_0, pase del punto x_0 hacia afuera del contenedor. El núcleo $K(\mathrm{x},\mathrm{x}')$ representa la distribución de probabilidad condicional de que el neutrón dispersado en el punto x, tenga su siguiente colisión en el punto x', y ν denota la probabilidad de no ser absorbido.

La naturaleza recursiva de la expresión (3.79) y su carácter multidimensional hacen muy difícil la obtención de resultados analíticos. Sin embargo, el método de Monte Carlo provee una manera numérica y directa de evaluar este tipo de integrales.

Para un valor $\mathrm{w} \in \mathcal{R}$, definimos:

$$H_0(\mathrm{w}) = v(\mathrm{w}) \qquad (3.80)$$

Aplicando a la ecuación de Fredholm, (3.79):

$$\begin{aligned} H_1(\mathrm{w}) &= \int_\omega v(\mathrm{w})K(\mathrm{w},\mathrm{s}_1)d\mathrm{s}_1 \\ H_1(\mathrm{w}) &= \int_\omega H_0(\mathrm{w})K(\mathrm{w},\mathrm{s}_1)d\mathrm{s}_1 \\ &\vdots \\ H_i(\mathrm{w}) &= \int_\omega v(\mathrm{s}_i)K(\mathrm{w},\mathrm{s}_i)\left[\prod_{j=2}^{i} K(\mathrm{s}_{j-1},\mathrm{s}_j)\right] d\mathrm{s}_1 \ldots d\mathrm{s}_i \\ H_i(\mathrm{w}) &= \int_\omega H_{i-1}(\mathrm{z})K(\mathrm{w},\mathrm{z})d\mathrm{z} \quad , i \geq 2 \end{aligned} \tag{3.81}$$

Repitiendo la recursión llegamos a la expresión:

$$\lambda \mathrm{s}_0 = \sum_{i=0}^{\infty} \nu^i H_i(\mathrm{s}_0) \tag{3.82}$$

que converge si se cumple la siguiente relación para los valores ν:

$$|\nu| < \frac{1}{sup_\omega \int_\omega |K(\mathrm{w},\mathrm{z})|d\mathrm{z}} \tag{3.83}$$

a la serie descrita se le conoce como una *serie convergente de Neumann*.

La expresión (3.82) nos indica que la integral de Fredholm (3.79) es la suma de un número infinito de integrales multidimensionales. Debido a que no existen formas analíticas para representar dicha suma, lo que se llega a hacer es truncar la serie hasta un número de términos $\tau + 1$, donde τ es pequeño y después evaluar:

$$\lambda \mathrm{s}_0 = \sum_{i=0}^{\tau} \nu^i H_i(\mathrm{s}_0). \tag{3.84}$$

Sin embargo, la convergencia lenta de esta serie nos lleva a pensar en la idea de usar cadenas de Markov para realizar esta integración. Si reescribimos la ecuación (3.82) introduciendo un término p_I que denota a un entero aleatorio:

$$\lambda \mathrm{s}_0 = \left[\sum_{i=0}^{\infty} \frac{\nu^i H_i(\mathrm{s}_0)}{p_I}\right] p_I \tag{3.85}$$

Lo que nos hace ver a la ecuación anterior como el valor esperado de una cadena de Markov formada por puntos w propagados por el núcleo de la integral de Fredholm.

El uso de formas adecuadas de las expresiones para minimizar la varianza, así como técnicas de muestreo inducido puede mejorar de manera considerable esta integración por cadenas de Markov.

3.7.3. Algoritmo de Metropolis

El algoritmo de Metropolis [12] es un método de muestreo de variables aleatorias con distribuciones arbitrarias basado en una caminata aleatoria.

Si se quiere muestrear una distribución de probabilidad f por medio de una caminata al azar, la matriz de transición debe cumplir con la condición de balance detallado.

Supongamos que tenemos un sistema en el estado i y que posteriormente pasa a otro estado j. En el equilibrio, la condición de *balance detallado* (ver la sección 3.7). enuncia que el flujo de $i \to j$ debe ser igual al flujo $j \to i$. Lo anterior puede ser expresado de la siguiente manera:

$$T_{i\to j} f_i = T_{j\to i} f_j \tag{3.86}$$

donde p_i es la probabilidad de encontrar el sistema en el estado i y T es la matriz de transición del sistema.

El algoritmo 3 garantiza que la condición de balance detallado se cumpla.

Algoritmo 3 Muestreo por el método de Metropolis

Salida: $S = \{x_i, x_2, \cdots, x_N\}$ con distribución $f()$

```
repeat
  for i = 1, ..., N do
    t2 ← x[i] + δ · U_{-1,1}
    q = f(t2)/f(x[i])
    A ← min(q, 1.0)
    if A > U_{0,1} then
      x[i] ← t2
    end if
  end for
until { Se llege al equilibrio}
```

Una de las características más atractivas del algoritmo de Metropolis es que la implementación sólo se basa en la razón de dos distribuciones $\frac{f}{f}$. Por lo tanto, no es necesario conocer sus propiedades globales. Sólo basta poderla evaluarla en algunos puntos del espacio. Ésta propiedad la usaremos como base para los algortimos de Monte Carlo cuántico.

[12] También se le conoce como método M(RT)^2, por las siglas de sus autores: Metrópolis, Rosenbluth, Rosenbiuth, Teller y Teller[62].

3.8. Regresando a la caja de potencial

En el capítulo anterior, describimos la solución de la caja de potencial usando funciones de Green, y fuerza bruta (integrando por métodos de malla). Ahora estamos en condiciones de proponer una solución a la ecuación de Schrödinger usando cadenas de Markov.

Recordemos que la ecuación de Schrödinger para un número arbitrario de partículas, tiene la forma:

$$\hat{\mathcal{H}}\psi(\mathrm{R}) = E\psi(\mathrm{R}) \tag{3.87}$$

La función de Green es definida por la ecuación:

$$\hat{\mathcal{H}}G(\mathrm{R},\mathrm{R}') = \delta(\mathrm{R},\mathrm{R}') \tag{3.88}$$

con las mismas condiciones de frontera que rigen a la ecuación 3.87, tanto para $\psi(\mathrm{R})$ como para R y R'.

Podemos racionalizar esta definición relacionando los siguientes hechos: La ecuación anterior significa que la función de Green al ser operada por el Hamiltoniano debe comportarse como una función delta en el punto en el que las coordenadas R y R' son las mismas. El operador Hamiltoniano posee un término con una segunda derivada (el laplaciano correspondiente a la energía cinética). Si uno se refiere a la jerarquía de singularidades en la sección A.1.3, específicamente en la figura A.1.3, se puede dar cuenta que la función valor absoluto es el resultado de antiderivar la función delta dos veces.

La función de Green que intuitivamente nos dió el resultado correcto en el ejemplo de la caja de potencial, tiene un *vértice* parecida a la que presenta la función valor absoluto. Ésta se puede ver en la figura 2.2.

Por lo que una función que cumple las condiciones a la frontera de nulidad de la función de onda en -1 y 1, así como las condiciones de vértice en los puntos en los que las coordenadas coalescen[13] son los requisitos para una función de Green.

Una solución formal a las condiciones que solicitamos para la función de Green es:

$$G(\mathrm{R},\mathrm{R}') = \sum_{\alpha} E_{\alpha}^{-1}\psi_{\alpha}(\mathrm{R})\psi_{\alpha}(\mathrm{R}'), \tag{3.89}$$

donde $\psi_{\alpha}(\mathrm{R})$ son las funciones propias del problema. Con esta función como núcleo, podemos transformar a la ecuación de Schrödinger en una ecuación integral de Fredholm de segunda especie:

$$\psi_0(\mathrm{R}) = E_0 \int G(\mathrm{R},\mathrm{R}')\psi_0(\mathrm{R}')d\mathrm{R}', \tag{3.90}$$

[13] La coalescencia implica que coinciden en el mismo punto.

que fue la ecuación que resolvimos por iteración en el capítulo previo:

$$\psi^{(n+1)}(\mathrm{R}) = E_R \int G(\mathrm{R},\mathrm{R}')\psi^{(n)}(\mathrm{R}')d\mathrm{R}'. \tag{3.91}$$

Ahora expandamos a nuestra distribución inicial de puntos de $\psi(\mathrm{R})$ en la base infinita de funciones propias:

$$\psi^{(0)}(\mathrm{R}) = \sum_{\alpha} C_\alpha \psi_\alpha(\mathrm{R}). \tag{3.92}$$

Ahora, sustituyamos (3.92) y (3.89) en (3.79):

$$\psi^{(1)}(\mathrm{R}') = E_T \sum_{\alpha\alpha'} E_{\alpha'}^{-1} \int \psi_{\alpha'}(\mathrm{R})\psi_{\alpha'}(\mathrm{R}')C_\alpha\psi_\alpha(\mathrm{R})d\mathrm{R}. \tag{3.93}$$

Lo que se convierte en una suma sobre los dos índices que se cancelará por las propiedades de la delta de Kronecker:

$$\psi^{(1)}(\mathrm{R}) = E_T \sum_{\alpha\alpha'} E_{\alpha'}^{-1}\delta_{\alpha\alpha'}\psi_{\alpha'}(\mathrm{R}')C_\alpha, \tag{3.94}$$

que se simplifica a:

$$\psi^{(n)}(\mathrm{R}) = \sum_{\alpha} \left(\frac{E_T}{E_\alpha}\right)^n C_\alpha\psi_\alpha(\mathrm{R}). \tag{3.95}$$

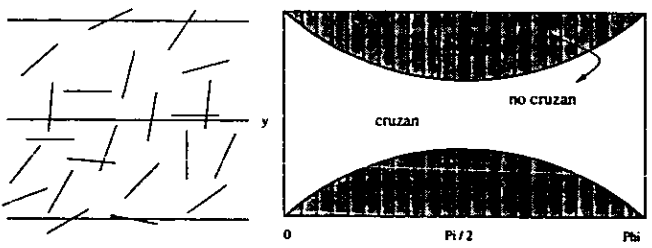
Para una n, lo suficientemente grande, el término $\alpha = 0$ domina en la suma:

$$\psi^{(n)}(\mathrm{R}) \approx \sum_{\alpha} \left(\frac{E_T}{E_\alpha}\right)^n C_0\psi_0(\mathrm{R}). \tag{3.96}$$

Si $E_T = E_0$, el único término que sobrevive es el del estado basal, y obtenemos la solución de la ecuación de Schrödinger para este estado.

En los capítulos 4 y 5 analizaremos con detalle los métodos de integración de (3.90) por cadenas de Markov.

- Suponga usted que lanza n varitas rectas de longitud l en un piso con líneas periódicas separadas por una distancia h.
- Las varitas pueden caer con un ángulo ϕ relativo a la vertical en la malla y en cualquier punto y que las separe de las líneas.



- Cuente las varitas que cruzan y cuente las varitas que no cruzan.
- Dividamos las áreas $a_{cruzan} = 2\int_0^{\pi} \frac{l}{2} sen\phi d\phi = 2l$ y $a_{nocruzan} = \pi y$, para obtener la probabilidad de que una varita cruce las líneas del parquet:

$$p = \frac{2l}{\pi h} \tag{3.36}$$

- Si la varita es de longitud $h/2$, la probabilidad es $p = 1/\pi$. Por lo que basta lanzar n varitas para irse aproximando al valor de π.
- Los resultados del experimento están en la figura 3.5

Tabla 3.1.: Descripción del método que usó el conde de Buffon para calcular el número π.

P_t	Iteraciones	I	$\mathcal{D}(\eta)$	σ	δ_p	δ_r
a	10	1.084697	**0.180632**	0.425008	0.090719	0.084697
a	100	1.034063	**0.250531**	0.500530	0.033785	0.034063
b	10	0.932534	**0.013006**	0.114043	0.024343	0.067465
b	100	1.002002	**0.017248**	0.131332	0.008864	0.002002
c	10	1.029628	**0.014683**	0.121174	0.025865	0.029628
c	100	0.998072	**0.027343**	0.165359	0.011161	0.001927

Tabla 3.2.: Resultados de la integración por Monte Carlo.

4. Monte Carlo Variacional

El método de Monte Carlo Variacional (MCV) es una extensión directa de las ideas presentadas en el capítulo anterior. La idea principal es aplicar los métodos de integración de ecuaciones de Fredholm de segunda especie por cadenas de Markov a la resolución de una integral variacional.

4.1. Métodos Variacionales

Los métodos variacionales utilizados en mecánica cuántica se basan en el teorema variacional, el cual dice lo siguiente:

Teorema 4.1. *El valor esperado $E[\phi]$ asociado a un operador Hamiltoniano independiente del tiempo $\hat{\mathcal{H}}$ usando cualquier función de prueba normalizada es una* **cota superior** *al eigenvalor de menor energía.*

$$E[\phi] = \int \phi^* \hat{\mathcal{H}} \phi \, d\Omega \geq E_0, \tag{4.1}$$

donde ϕ es una función bien portada[1].

Para una función de prueba no normalizada, la ecuación (4.1) se convierte en:

$$\frac{\int \phi^* \hat{\mathcal{H}} \phi \, d\Omega}{\int \phi^* \phi \, d\Omega} \geq E_0 \tag{4.2}$$

Si se desea ahondar en las consecuencias del teorema variacional se pueden consultar textos básicos de mecánica cuántica [57].

Comúnmente se utilizan como funciones de prueba, la suma de varias funciones independientes *pesadas* por coeficientes indeterminados (c_i),

$$\phi = c_1\chi_1 + c_2\chi_2 + \cdots + c_n\chi_n,$$

y se buscan los valores de c_i que hagan a $E[\phi]$ un mínimo.

[1] Por bien portada nos basta que la función sea continua, univaluada y finita dentro del espacio de configuración del sistema.

4.2. El método de Monte Carlo Variacional

4.2.1. La motivación para usar métodos de Monte Carlo en un problema Variacional

Alan D. Sokal [80] afirma:

> El método de Monte Carlo es extremadamente malo; sólo debe ser usado cuando todos los demás métodos disponibles son peores. ¿A qué se debe esto? Primero, *todos* los métodos numéricos son potencialmente peligrosos, comparados a los métodos analíticos; hay más formas de cometer errores. Segundo, con respecto a los métodos numéricos, el de Monte Carlo es uno de los menos eficientes; sólo debe ser usado en aquellos problemas difíciles en los que todos los demás métodos numéricos son menos eficientes. Déjenme ser más preciso al respecto. Virtualmente todos los métodos de Monte Carlo tienen la propiedad de que el error estadístico se comporta como:
>
> $$error \sim \frac{1}{\sqrt{\text{esfuerzo computacional}}} \tag{4.3}$$
>
> (o peor); ésta es una consecuencia universal del teorema del límite central. Es posible incrementar la constante de proporcionalidad en esta relación por un factor de 10^6 o más ... , pero el comportamiento $1/\sqrt{n}$ es ineludible. Esto debe ser comparado con los métodos numéricos determinísticos tradicionales cuya tasa de convergencia es típicamente algo como $1/n^4$ o e^{-n} o e^{-2n}. Por lo tanto, los métodos de Monte Carlo deberían ser usados solo en aquellos problemas extremadamente difíciles en los que todos los demás métodos numéricos se comportan *peor* que $1/\sqrt{n}$.
>
> [...] Si la integración numérica es difícil en muchas dimensiones; aunque el método de Monte Carlo sea malo, todos los demás métodos son peores.
>
> Concluyendo, los métodos de Monte Carlo sólo deberían de ser usados cuando ni los métodos analíticos ni los métodos numéricos determinísticos son usables (o eficientes). Un dominio general de aplicación para los métodos de Monte Carlo será, por lo tanto, el de los sistemas con *varios grados de libertad, lejos del régimen perturbativo.* ¡Y justo esos sistemas son los de mayor interés en la mecánica estadística y la teoría de campo cuántica!

Después de leer esta cita no debemos descorazonarnos. Los métodos numéricos variacionales convencionales son muy buenos, su desventaja consiste en la incapacidad de evaluar funciones de prueba arbitrarias (sección 2.2). Las funciones de prueba que pueden evaluar dichos métodos son monoelectrónicas y expansiones de funciones orbitales. En cambio, en Monte Carlo Cuántico la forma de las funciones de prueba

puede ser arbitraria. Por lo que se pueden incluir términos que traten la correlación de manera directa y en una forma muy compacta. Se ahondará más al respecto en el capítulo 6.

4.2.2. Formalismo del método de Monte Carlo Variacional

Podemos rearreglar la ecuación (4.1) de la siguiente forma:

$$E[\phi] = \int \frac{\hat{\mathcal{H}}\phi}{\phi}\mathcal{F}d\Omega \equiv \langle E_L \rangle, \tag{4.4}$$

donde:

$$\mathcal{F} = \frac{|\phi|^2}{\int \phi^*\phi d\Omega} \tag{4.5}$$

El operador de peso, $\mathcal{F}$, no es otra cosa que la densidad de probabilidad normalizada para los electrones.

La expresión que nos servirá para estimar la energía es:

$$\frac{1}{\phi_T}\hat{\mathcal{H}}\phi_T \equiv E_L, \tag{4.6}$$

que es conocida como energía local (E_L).

Según la ecuación (3.44), el valor de la integral (4.4) es el promedio de una serie E_L evaluadas a partir de configuraciones R muestreadas a partir de $\mathcal{F}$. Es decir podemos estimar el valor esperado de la energía mediante:

$$E_{MCV} = \lim_{N\to\infty}\frac{1}{N}\sum_{i=1}^{N}\frac{1}{\phi_T(\mathrm{R}_i)}\hat{\mathcal{H}}\phi_T(\mathrm{R}_i) \tag{4.7}$$

Si asumimos un muestreo no correlacionado, la varianza del valor medio está dada por:

$$\sigma^2(E[\phi]) = \frac{\langle E_L^2\rangle - \langle E_L\rangle^2}{M-1} \tag{4.8}$$

Debido a que la distribución ϕ^2 en la mayoría de los casos no puede ser muestreada directamente, tendremos que usar el algoritmo de Metropolis (sección 3.7.3)., el cual sólo necesita conocer las razones de las densidades de probabilidad en diferentes puntos , por lo que no es necesario calcular la integral de normalización:

$$\int \phi_T^*\phi_T d\Omega$$

¿Qué es un *caminante*?

A partir de aquí, hablaremos en términos de *caminantes*[2]. En los algoritmos de cadenas de Markov, nada nos impide promediar más de una caminata aleatoria. En el método de Monte Carlo cuántico proponemos varias configuraciones de partículas en el espacio[3] e ir evolucionándolas con una cadena de Markov. Posteriormente, los resultados *locales* de cada uno de dichos caminantes son promediados para obtener las propiedades *globales* del sistema. (ver figura 4.1)

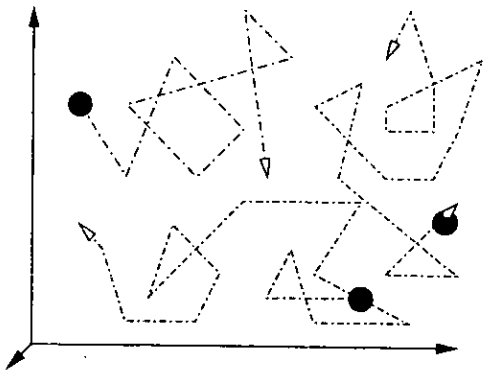


Figura 4.1.: Los caminantes se mueven siguiendo las reglas de transición de una cadena de Markov. Las propiedades del ensamble son promediadas para obtener la información del sistema.

Los caminantes son análogos a los conejos descritos en la introducción de esta tesis.

4.3. Muestreo Inducido por el formalismo de Fokker Planck

En el caso anterior, los caminantes se movían al azar, sin tomar en cuenta previamente algunas de las propiedades de la distribución ψ^2, provocando que varios movimientos fueran rechazados innecesariamente. En esta sección, introduciremos una aproximación que no involucra rechazos, y que incluye muestreo inducido a partir del formalismo de Fokker-Planck [42]. Este formalismo, sin embargo, tiene un error asociado al tamaño de paso $\delta\tau$, por lo que posteriormente hablaremos de un algoritmo que combina los mejores aspectos del muestreo con el algoritmo de Metropolis y el muestreo de Fokker-Planck.

[2]Llamado *walker*, o *psi-particle*.

[3]Análogas a puntos en el espacio fase.

Algoritmo 4 Monte Carlo Variacional-Metropolis

Salida: Distribución de equilibrio y una estimación de la energía promedio (E y E_σ) de un conjunto de caminantes (W) usando una función de prueba ϕ_t muestreada por el algoritmo de Metropolis.

{Inicializando posiciones al azar}
for C = 1, ..., Ncaminates do
  for i = 1, ..., dim · N_{elec} do
    W[C, i] ← $\mathcal{U}_{-1,1}$
  end for
end for
{Tomar promedios hasta el equilibrio}
for P = 1, ..., N_{pasos} do
  for C = 1, ..., $N_{caminates}$ do
    Psi1 ← psit(W[C])
    for i = 1, ..., dim · N_{elec} do
      W'[C, i] ← R(C, i] + $\mathcal{U}_{-1,1} \cdot \tau$
    end for
    Psi2 ← psit(W'[C])
    A = min($\frac{(psi2)^2}{(psi1)^2}$, 1.0)
    if A > $\mathcal{U}_{0,1}$ then
      copia (W' → W)
      Aceptados ← Aceptados + A
    end if
    E ← Elocal(R)
    E_{prom} ← E_{prom} + E
    E^2 ← E^2 + E * E
  end for
end for
Aceptados ← $\frac{Aceptados}{N_{cam} N_{pasos}}$
E_{final} ← $\frac{Eprom}{N_{cam} N_{pasos}}$
E_σ ← $\frac{\sqrt{(\frac{E^2}{N_{cam} N_{pasos}} - E_{prom} * E_{prom})}}{N_{cam} Npasos - 1}$

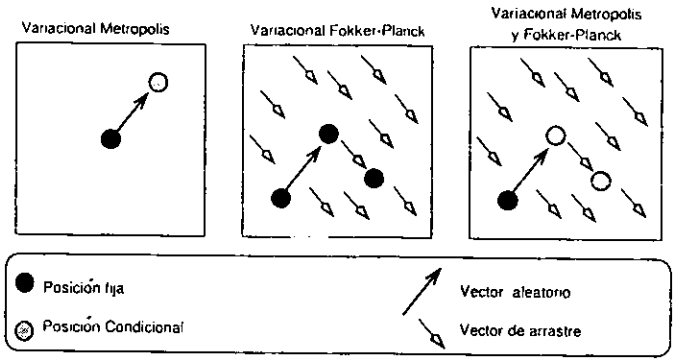


Figura 4.2.: Los tres algoritmos de MCV discutidos. El primero, MCV con Metropolis, mueve a los caminantes al azar, el aceptarlos o no depende de pasar la prueba de Metropolis. El segundo, MCV Fokker-Planck, primero mueve a los caminantes al azar y posteriormente los *arrastra* hacia zonas de mayor densidad electrónica. El tercer algoritmo combina ambos métodos; mueve a los caminantes como en el caso de Fokker-Planck, pero el movimiento puede ser rechazado si falla la prueba de Metropolis.

Considérese un proceso de difusión caracterizado por una densidad dependiente del tiempo $f(\mathbf{r},t)$. Los procesos de difusión isotrópicos obedecen a la ecuación de Fokker-Planck:

$$\frac{\partial f}{\partial \tau} = \sum_i D \frac{\partial}{\partial \mathrm{x}_i}\left(\frac{\partial}{\partial \mathrm{x}_i} - F_i(\mathbf{x})\right) f, \tag{4.9}$$

Donde D es una constante de difusión, y F_i es la componte i de una velocidad de arrastre causada por un potencial externo. Como antes, queremos converger a la densidad estacionaria $f = \phi^2 / \int \phi^2 d\mathrm{x}$. Un estado estacionario puede obtenerse haciendo cero la parte izquierda de la ecuación, es decir:

$$\sum_i D\left(\frac{\partial^2 f}{\partial^2 \mathrm{x}_i} - \frac{\partial}{\partial \mathrm{x}_i}(F_i f)\right) = 0. \tag{4.10}$$

La ecuación (4.10) puede ser satisfecha si cada término de la suma es igualado a cero:

$$\frac{\partial^2 f}{\partial \mathrm{x}_i^2} = f \frac{\partial}{\partial \mathrm{x}_i} F_i + F_i \frac{\partial}{\partial \mathrm{x}_i} f. \tag{4.11}$$

El vector de arrastre $\vec{\mathbf{F}}$, debe ser de la forma $F_i = g(x)\frac{\delta f}{\delta x}$ para poder asegurar la existencia de la segunda derivada. Si sustituimos esta forma de $\vec{\mathbf{F}}$ en la ecuación (4.11) obtenemos:

$$\frac{\partial^2 f}{\partial \mathrm{x}_i^2} = f \frac{\partial g}{\partial f}\left(\frac{\partial f}{\partial \mathrm{x}_i}\right)^2 + f g \frac{\partial^2 f}{\partial \mathrm{x}_i^2} + g\left(\frac{\partial f}{\partial \mathrm{x}_i}\right)^2. \tag{4.12}$$

La cancelación de los términos con segundas y primeras derivadas requiere que se cumpla la igualdad $g = \frac{1}{f}$. Por lo que la densidad estacionaria $f = \phi^2 / \int \phi^2 dx$ puede ser obtenida si se escoge que el vector de arraste sea de la forma:

$$\vec{F} = 2\frac{1}{\phi}\nabla\phi. \tag{4.13}$$

Ahora tenemos una ecuación de difusión con arrastre que genera la distribución estacionaria que necesitamos. El problema es ahora el de la implementación en un algoritmo de Monte Carlo. En mecánica estadística, las trayectorias resultantes de la ecuación de Fokker-Planck se generan a partir de la ecuación de Langevin:

$$\frac{\partial x(t)}{\partial t} = D\vec{F}(x(t)) + \eta. \tag{4.14}$$

Donde η es una *fuerza estocástica*[4] distribuida en una Gaussiana multidimensional con una media de cero y una varianza de $2D$. Al integrarla en un intervalo pequeño de tiempo δt, obtenemos una forma discreta que se puede usar facilmente en una simulación de Monte Carlo. La partícula se mueve de x a x' de acuerdo a la relación:

$$x' = x + D\vec{F}\delta t + \xi, \tag{4.15}$$

donde ξ es una variable Gaussiana aleatoria con una media de cero y una varianza de $2D\delta t$. Al usar esta forma discreta, introducimos un error en la simulación que depende de δt. Este error se puede corregir implementando un paso de aceptación de Metropolis después de mover la partícula.

En la figura 4.2 se muestra una comparación de los tres algoritmos mencionados.

[4]Una fuerza estocástica es una fuerza que fluctúa aleatoriamente.

5. Monte Carlo Cuántico de difusión

En el capítulo anterior analizamos el Método de Monte Carlo cuántico variacional (MCV), en este procedimiento la función de prueba ϕ determina la calidad de los resultados obtenidos. Si se requieren resultados de alta calidad el problema que se presenta es el de encontrar mejores alternativas. Las formas de funciones de prueba tradicionales, como las usadas en los métodos post-Hartree-Fock, requieren un número muy grande de determinantes (alrededor de 10^5 o 10^6) para obtener un 99% de la energía de correlación. Otras formas más generales de funciones de prueba que toman en cuenta las posiciones de las partículas convergen mucho más rápido[1]. Sin embargo, las expresiones que incluyen las coordenadas de las partículas son difíciles de incluir en un formalismo *a-la-Hartree-Fock*.

Los métodos de Monte Carlo cuántico resuelven la ecuación de Schrödinger transformándola en una ecuación integral. Para poder hacer esta transformación, se debe determinar la función de Green adecuada. En este capítulo estudiaremos las maneras de formular e implementar la evaluación de dichas funciones de Green.

Comenzaremos discutiendo la aplicación de las funciones de Green a la ecuación de Schrödinger, después discutiremos la aplicación de las técnicas de Muestreo Inducido y los algoritmos asociados a este método.

5.1. El formalismo de las funciones de Green

La ecuación de Schrödinger independiente del tiempo para el estado basal puede ser escrita como:

$$\hat{\mathcal{H}}|\psi_0\rangle = E_0|\psi_0\rangle, \tag{5.1}$$

donde $\hat{\mathcal{H}}$ es el operador Hamiltoniano descrito en la ecuación (5.12) de la sección 5.2.

Como ya estudiamos en el capítulo 3, los métodos de Monte Carlo son útiles para:

- crear una cadena de Markov iterativa,

[1] El ejemplo más sencillo es el de las funciones de prueba propuestas por Hylleraas para el átomo de Helio.

- y estimar el valor de integrales.

Para aprovechar estas cualidades, es conveniente transformar la expresión (5.1) en una ecuación integral iterativa [54].

La solución formal a la relación (5.1) se obtiene operando por la izquierda con el operador $\hat{\mathcal{H}}^{-1}$:

$$|\psi_0\rangle = \hat{\mathcal{H}}^{-1} E_0 |\psi_0\rangle, \tag{5.2}$$

Debido a que $\hat{\mathcal{H}}$ es un operador diferencial, $\hat{\mathcal{H}}^{-1}$ es un operador integral. Para poder deducir la forma del operador, podemos multiplicar por un conjunto de estados $\int_{-\infty}^{\infty} |\mathrm{R}\rangle \mathrm{R}\langle| d\mathrm{R}$ entre el operador $\hat{\mathcal{H}}^{-1}$ y ψ_0. Además se debe multiplicar por la izquierda por el conjunto de posiciones $|\mathrm{R}'\rangle$. Esto implica aplicar un operador de proyección a ψ de manera que cambiemos de la base $\{\psi_0, psi_1, \ldots, psi_n\}$ a una nueva representación $\{\mathrm{r}_1, \mathrm{r}_2, \ldots, \mathrm{r}_n\} \equiv \mathrm{R}$ obteniendo:

$$\langle \mathrm{R}'|\psi_0\rangle = E_0 \int_{-\infty}^{\infty} \langle \mathrm{R}'|\hat{\mathcal{H}}^{-1}|\mathrm{R}\rangle \langle \mathrm{R}|\psi_0\rangle d\mathrm{R} \tag{5.3}$$

Ahora definamos a la función de Green asociada con el operador $\hat{\mathcal{H}}$ como:

$$G(\mathrm{R}', \mathrm{R}) = \langle \mathrm{R}'|\hat{\mathcal{H}}^{-1}|\mathrm{R}\rangle \tag{5.4}$$

Por lo tanto, la ecuación (5.3) se convierte en la ecuación integral:

$$\psi_0(\mathrm{R}') = E_0 \int_{-\infty}^{\infty} G(\mathrm{R}', \mathrm{R}) \psi_0(\mathrm{R}) d\mathrm{R} \tag{5.5}$$

La ecuación anterior puede ser resuelta iterativamente como una cadena de Markov. Debemos escoger un estado inicial $\psi^{(0)}$ y una energía de referencia E_R que aproximen a ψ_0 y E_0 respectivamente. Posteriormente se puede construir la serie:

$$\psi^{(n+1)}(\mathrm{R}') = E_R \int_{-\infty}^{\infty} G(\mathrm{R}', \mathrm{R}) \psi^{(n)}(\mathrm{R}) d\mathrm{R} \tag{5.6}$$

Cuando $E_R = E_0$ la ecuación anterior está normalizada (ver la sección 3.8).

Antes de estudiar las propiedades de convergencia de la serie anterior, analicemos las propiedades de la función de Green. Recordando lo mencionado en la sección A.2.1, las condiciones de discontinuidad de la función de Green hacen que ésta por definición cumpla que:

$$\hat{\mathcal{H}} G(\mathrm{R}', \mathrm{R}) \equiv \delta(\mathrm{R} - \mathrm{R}') \tag{5.7}$$

Lo que en lenguaje de operadores se traduce como:

$$\hat{\mathcal{H}}\hat{\mathcal{H}}^{-1}\psi_0 = \psi_0 \tag{5.8}$$

Usando la definición anterior se puede construir la siguiente relación:

$$\int_{-\infty}^{\infty} \psi^*(\mathrm{R})\hat{\mathcal{H}}G(\mathrm{R}', \mathrm{R})d\mathrm{R} = \int_{-\infty}^{\infty} \psi^*(\mathrm{R})\delta(\mathrm{R} - \mathrm{R}')d\mathrm{R} \tag{5.9}$$

que nos lleva directamente a la ecuación integral obtenida previamente (5.5):

$$E_0 \int_{-\infty}^{\infty} G(\mathrm{R}', \mathrm{R})\psi(\mathrm{R})d\mathrm{R} = \psi(\mathrm{R}')\, d\mathrm{R} \tag{5.10}$$

Si tomamos el resultado de la ecuación anterior, $\psi(\mathrm{R}')$, y usamos la definición (5.8), se puede ver que la función de Green es simétrica con respecto a R y a R', esto es que $G(\mathrm{R}', \mathrm{R}) = G(\mathrm{R}, \mathrm{R}')$ para los estados reales $\psi_0 = \psi_0^*$.

Para poder usar métodos de Monte Carlo, la función de Green obtenida aquí debe cumplir las reglas de las matrices de transición para procesos Markovianos (definidas en la sección 3.7). Las matrices de transición deben tener elementos positivos, por tanto, la función de Green debe ser positiva en todos los puntos y normalizable.

Si se cumple la condición anterior, la ecuación (5.6) se puede integrar usando una caminata aleatoria en donde la probabilidad de moverse de un punto R a un punto R' es dada por la función de Green $G(\mathrm{R}', \mathrm{R})$.

Los métodos de solución que analizaremos a continuación son muy diferentes a los métodos clásicos debido a que:

- En la forma diferencial de la ecuación de Schrödinger (5.1), la información de la función de onda en un punto es determinada por el potencial y la curvatura *locales*.
- En la forma integral de la misma ecuación (5.5), la información de la función de onda en un punto está relacionada *globalmente* con la función de onda en todos los puntos restantes del espacio.

5.2. Funciones de Green dependientes del tiempo

Como siguiente paso, estudiaremos las funciones de Green para la ecuación de Schrödinger dependiente del tiempo. Sus soluciones serán el punto de partida para el algoritmo de Monte Carlo cuántico de difusión (MCD) La ecuación de Schrödinger dependiente del tiempo es

$$-\frac{\partial \Psi_,(\mathrm{R}, t)}{i\partial t} = \hat{\mathcal{H}}\Psi_,(\mathrm{R}, t) \tag{5.11}$$

En la ecuación anterior, $\hat{\mathcal{H}}$ es el operador Hamiltoniano y:

$$\mathrm{R} = (\mathrm{r}_1, \mathrm{r}_2, \ldots, \mathrm{r}_n),$$

es un vector 3N dimensional que denota las posiciones de las N partículas en el espacio continuo. Y definiremos a D como una *constante de difusión* con valor de $\hbar^2/2m_e$, es decir $\frac{1}{2}$ en unidades atómicas. El Hamiltoniano está dado por:

$$\hat{\mathcal{H}} = \hat{T} + \hat{\mathcal{U}} = \hat{T} + \hat{\mathcal{V}_{ee}} + \hat{\mathcal{V}_{eN}} + \hat{\mathcal{E}_{NN}} \tag{5.12}$$

donde $\hat{T}, \hat{\mathcal{V}_{ee}}$ y $\hat{\mathcal{V}_{eN}}$ son los operadores para la energía cinética y las interacciones electrón-electrón y electrón-núcleo. Por simpleza consideramos un sistema adiabático [69], en el cual los núcleos o iones son estáticos y contribuyen con una constante E_{NN} a la energía. Sin embargo, este formalismo puede ser fácilmente extendido a sistemas no adiabáticos en donde se considera el carácter cuántico de los núcleos [15].

La solución formal a la ecuación de Schrödinger dependiente del tiempo es:

$$\Psi(\mathrm{R}, t) = \sum_{k=0}^{\infty} C_k \psi_k(\mathrm{R}) e^{-i(E_k - E_R)t} \tag{5.13}$$

En esta ecuación, los coeficientes C_k sólo dependen de las condiciones iniciales:

$$C_k = \langle \psi_k | \Psi(t = 0) \rangle. \tag{5.14}$$

Si tomamos a la ecuación (5.11) y realizamos el siguiente cambio de variable $\tau = it$, y restamos una energía de referencia E_R a ambos lados de la expresión,

$$it \rightarrow \tau$$

$$V(x) \rightarrow V(x) - E_R$$

$$E_i \rightarrow E_i - E_R$$

obtenemos la relación:

$$\frac{\partial \Psi_i(\mathrm{R}, \tau)}{\partial \tau} = \frac{1}{2}\nabla^2 - [V(\mathrm{R}) - E_R]\Psi_i(\mathrm{R}, \tau) \tag{5.15}$$

La expresión (5.11) tiene un comportamiento temporal oscilatorio, el resolver esta ecuación nos lleva a los métodos conocidos como los **métodos de integrales de camino**, *Path Integral Methods* [60]. Éstos métodos se usan par estudiar fenómenos relacionados con la dinámica cuántica.

Sin embargo, nosotros no estamos interesados en el comportamiento oscilatorio, sino más bien en el decaimiento exponencial en el tiempo imaginario (5.15) y la extracción de propiedades del estado basal a una temperatura cero. Estamos *inventando* un proceso estocástico sin sentido físico, es decir, los resultados dinámicos de nuestra simulación no tienen que ver con ningún observable. Lo que al final nos va a importar son promedios a lo largo de la simulación (ver la sección 3.7.2). Existen otros métodos basados en las integrales de camino de Feynman y la evolución en tiempo imaginario que estudian casos cuánticos en temperaturas finitas, [26] [11] [89], pero estos métodos presentan dificultades para obtener la estructura electrónica de moléculas y sólidos.

Las funciones propias $|\Psi_i\rangle$ pueden ser expresadas como una combinación lineal de los estados propios del sistema independientes del tiempo ψ:

$$|\Psi_i(\mathbf{R},\tau)\rangle = \sum_i c_i(\tau)|\psi_i(\mathbf{R})\rangle \tag{5.16}$$

y la solución formal para el *tiempo imaginario* τ es:

$$|\Psi_i(\mathbf{R},\tau)\rangle = \sum_i C_i e^{-(E_i - E_R)\tau}|\psi_i\rangle \tag{5.17}$$

Para un tiempo imaginario τ suficientemente largo, sólo una función propia contribuye a Ψ, de hecho, la función que tenga el valor propio más negativo - el estado basal. Debe notarse, sin embargo, que aunque hablemos de *movimiento de partículas*, en realidad no podemos obtener soluciones a la dinámica en tiempo real.

Si hacemos que E_R sea E_0 cuando $E_i < E_0$, la función anterior diverge exponencialmente a ∞, mientras que para $E_i > E_0$ la función tiende a cero, por tanto, a tiempos imaginarios τ suficientemente grandes, obtenemos el estado basal [2]:

$$\lim_{\tau\to\infty} |\Psi_i(\mathbf{R},\tau)\rangle = C_0|\psi_0(\mathbf{R})\rangle \tag{5.18}$$

Ahora intentemos resolver el caso dependiente del tiempo (5.15) por medio del método de funciones de Green. Al igual que hicimos con la ecuación (5.5) el primer paso sería proponer una ecuación integral de la forma:

$$\Psi(\mathbf{R}',\tau_2) = \int_{-\infty}^{\infty} G(\mathbf{R}',\tau_2|\mathbf{R},\tau_1)\Psi(\mathbf{R},\tau_1)d\mathbf{R} \tag{5.19}$$

La función de Green $G(R',\tau_2|R,\tau_1)$ satisface la ecuación de Schrödinger (como se demuestra la tabla 5.1):

[2] Es posible obtener la función de onda de un estado excitado, pero es un procedimiento más elaborado y fuera de los alcances de esta tesis.

- Tomamos a la ecuación (5.19):

$$\Psi(\mathbf{R}', \tau_2) = \int_{-\infty}^{\infty} G(\mathbf{R}', \tau_2|\mathbf{R}, \tau_1)\Psi(\mathbf{R}, \tau_1)d\mathbf{R}$$

- Aplicamos el operador $(\hat{\mathcal{H}} - E_R)$ a ambos lados de la ecuación anterior:

$$(\hat{\mathcal{H}} - E_R)\Psi(\mathbf{R}', \tau_2) = \int (\hat{\mathcal{H}} - E_R)G(\mathbf{R}', \tau_2|\mathbf{R}, \tau_1)\Psi(\mathbf{R}, \tau_1)$$

- Por otro lado, derivamos con respecto a τ_2 ambos lados de la ecuación original:

$$-\frac{\partial\Psi(\mathbf{R}', \tau_2)}{\partial\tau_2} = -\int \frac{\partial G(\mathbf{R}', \tau_2|\mathbf{R}, \tau_1)}{\partial\tau_2}\Psi(\mathbf{R}, \tau_1)d\mathbf{R}$$

- Los lados izquierdos de las ecuaciones anteriores son iguales (de hecho son la ecuación de Schrödinger dependiente del tiempo). Para que los lados derechos lo sean, se debe de cumplir:

$$(\hat{\mathcal{H}} - E_R)G(\mathbf{R}', \tau_2|\mathbf{R}, \tau_1) = -\frac{\partial G(\mathbf{R}', \tau_2|\mathbf{R}, \tau_1)}{\partial\tau_2}$$

QED.

Tabla 5.1.: Demostración de que la función de Green cumple con la misma ecuación diferencial que la función de onda dependiente del tiempo Ψ.

$$-\frac{\partial G(\mathbf{R}', \tau_2|\mathbf{R}, \tau_1)}{\partial\tau_2} = (\hat{\mathcal{H}} - E_R)G(\mathbf{R}', \tau_2|\mathbf{R}, \tau_1) \tag{5.20}$$

Además podemos ver que a $\tau_2 = \tau_1$, tenemos:

$$\Psi(\mathbf{R}', \tau_1) = \int G(\mathbf{R}', \tau_1|\mathbf{R}, \tau_1)\Psi(\mathbf{R}, \tau_1)d\mathbf{R} \tag{5.21}$$

Que por la definición de la delta de Dirac (sección A.1.3) satisface:

$$G(\mathbf{R}', \tau_1|\mathbf{R}, \tau_1) = \delta(\mathbf{R} - \mathbf{R}') \tag{5.22}$$

5.2.1. Forma de la función de Green

La solución de la ecuación (5.15) de un tiempo τ_1 a un tiempo τ_2 es:

$$|\Psi(\tau_2)\rangle = \exp\{(\hat{\mathcal{H}} - E_R)(\tau_2 - \tau_1)\}\Psi(\tau_1)\rangle \tag{5.23}$$

El operador del lado derecho $e^{(\hat{\mathcal{H}}-E_R)(\tau_2-\tau_1)}$ es el operador de evolución temporal para $(\hat{\mathcal{H}} - E_R)$. Si insertamos un conjunto completo de estados $|\mathrm{R}\rangle$ entre el operador y Ψ, y multiplicamos por la izquierda por los estados $\langle \mathrm{R}'|$, obtenemos:

$$\Psi(\mathrm{R}', \tau_2) = \int \langle \mathrm{R}'| \exp\{(\hat{\mathcal{H}} - E_R)(\tau_2 - \tau_1)\}|\mathrm{R}\rangle \Psi(\mathrm{R}, \tau_1) d\mathrm{R} \tag{5.24}$$

Comparando con la ecuación (5.19) encontramos que la forma de la función de Green es:

$$G(\mathrm{R}', \tau_2|\mathrm{R}, \tau_1) = \langle \mathrm{R}'| \exp\{(\hat{\mathcal{H}} - E_R)(\tau_2 - \tau_1)\}|\mathrm{R}\rangle \tag{5.25}$$

La ecuación anterior sólo depende de la diferencia de tiempos $\tau_2 - \tau_1$, por lo que podemos reescribir la ecuación (5.19) como:

$$\Psi(\mathrm{R}', \tau + \delta\tau) = \int_{-\infty}^{\infty} G(\mathrm{R}', \mathrm{R}; \delta\tau) \Psi(\mathrm{R}, \tau) d\mathrm{R} \tag{5.26}$$

Como en el caso de la ecuación (5.10) hemos obtenido una relación que se puede resolver iterativamente.

La convergencia de la serie producida por iterar la relación (5.26) expandiendo a la función de Green en las funciones propias de $\hat{\mathcal{H}}$. Esto se logra insertando dos conjuntos completos de estados en la expresión (5.25):

$$G(\mathrm{R}', \mathrm{R}; \delta\tau) = \sum_{i=0}^{\infty} e^{-(E_I - E_R)\delta\tau} \psi_i(\mathrm{R}') \psi_i(\mathrm{R}) \tag{5.27}$$

Si sustituimos la ecuación anterior en (5.26) expandiendo al mismo tiempo:

$$\Psi(\mathrm{R}, 0) = \sum_k C_k \psi_k, \tag{5.28}$$

producimos la primera iteración:

$$\Psi(\mathrm{R}', \delta\tau) = \int \sum_{i=0}^{\infty} e^{-(E_I - E_R)\delta\tau} \psi_i(\mathrm{R}') \psi_i(\mathrm{R}) \sum_{k=0}^{\infty} C_k \psi_k(\mathrm{R}) d\mathrm{R} \tag{5.29}$$

$$= t \sum_{k=0}^{\infty} C_k \psi_k(\mathrm{R}') e^{-(E_I - E_R)\delta\tau} \tag{5.30}$$

Que muestra de nuevo que los estados excitados decaen exponencialmente.

5.3. Relación de la Ecuación de Schrödinger en el tiempo imaginario con un proceso de difusión y reproducción

Hasta ahora, hemos transformado el problema de resolver la ecuación de Schrödinger sin introducir ninguna aproximación. Si se conociera la función de Green exacta, no se

requeriría realizar ningún cálculo de Monte Carlo, pues la solución se podría obtener analíticamente.

Por un momento ignoremos el término del potencial en el Hamiltoniano (5.12) de la ecuación (5.11). Esta ecuación se convierte en la **ecuación de difusión en un espacio 3-N dimensional**:

$$-\frac{\partial \Psi(\mathrm{R},\tau)}{\partial \tau} = -\frac{1}{2}\sum_{i=1}^{N}\nabla_i^2\Psi(\mathrm{R},\tau) \tag{5.31}$$

La función de Green para este problema es exacta. Es una curva gaussiana con una varianza proporcional a τ:

$$G(\mathrm{R}',\mathrm{R},\tau) = (2\pi\tau)^{3N/2}\exp\left\{\frac{-(\mathrm{R}-\mathrm{R}')^2}{2\tau}\right\} \tag{5.32}$$

La ecuación de difusión describe una gran variedad de fenómenos físicos:

- la transferencia de calor,
- el movimiento browniano de partículas en disolución,
- el mezclado de dos gases,
- la penetración de átomos dopantes de la superficie de un sólido al interior, etc.

5.3.1. Representación de la función de onda por caminantes aleatorios

Los procesos de difusión se asocian a la aleatoriedad. La relación anterior se puede interpretar como una **ecuación maestra** de un sistema de partículas moviéndose en un *proceso estocástico de difusión*. Esta interpretación nos permite representar a la función de onda Ψ por un conjunto de *partículas brownianas*, es decir como un conjunto de puntos de muestreo discretos también conocidos como **caminantes aleatorios**.

La manera de interpretar a la ecuación maestra de difusión (5.32) es que representa el caso continuo de la matriz de transición (3.53) que rige la evolución de Ψ en el tiempo imaginario. Por lo anterior, podemos establecer la siguiente regla de correspondencia:

$$\Psi(\mathrm{R},\tau) \Longrightarrow \sum_k \delta(\mathrm{R}-\mathrm{R}_k) \tag{5.33}$$

Sustituyendo la ecuación anterior en (5.15) logramos la evolución de los caminantes para un intervalo de tiempo $\delta\tau$:

$$\Psi(\mathrm{R},\tau+\delta\tau) \Longrightarrow \sum_k G(\mathrm{R},\mathrm{R}_k;\delta\tau) \tag{5.34}$$

El resultado es un conjunto de gaussianas con una varianza igual a $3N\tau$. Para volver a la representación de funciones delta, las gaussianas en (5.34) son muestreadas de nuevo por otro conjunto de funciones delta.

El procedimiento anterior es repetido hasta que se han llevado a cabo un número suficiente de iteraciones con las que podamos calcular promedios de cantidades de interés. Llamaremos *función de Green de difusión*, $G_{difusion}$ al propagador G de la ecuación (5.34).

Las simulaciones por métodos de Monte Carlo de procesos de difusión son comunes en el ámbito de la ingeniería, ciencias atmosféricas y en la economía. Un ejemplo económico es un simulación por Cadenas de Markov y Monte Carlo (MCMC) de la volatilidad del mercado cambiario de Estados Unidos [30].

5.4. Funciones de Green para partículas en un potencial arbitrario

Ahora tratemos un caso más complicado, el caso en el que el operador Hamiltoniano $\hat{\mathcal{H}}$ incluye también los términos de energía potencial. El problema se vuelve no trivial. No existen expresiones explícitas para la función de Green por lo que tenemos que usar algunas aproximaciones.

Ahora olvidemos por un momento los términos de energía cinética y pensemos en los términos de energía potencial. La ecuación de Schrödinger se convierte en un proceso cinético de primer orden:

$$-\frac{\partial\Psi(\mathrm{R},\tau)}{\partial\tau} = (E_R - V(\mathrm{R}))\Psi(\mathrm{R},\tau) \tag{5.35}$$

La función de Green que satisface la condición $-\frac{\partial G}{\partial\tau} = (E_R - V)G$ es:

$$G(\mathrm{R}',\mathrm{R};\tau) = \exp\left\{-(\frac{1}{2}(V(\mathrm{R}) + V(\mathrm{R}')) - E_R\tau)\right\} \tag{5.36}$$

Algunos de los problemas físicos que siguen una ley cinética de primer orden son:

- el decaimiento radiactivo,
- muchas reacciones químicas
- la reproducción de poblaciones bacterianas

La justificación matemática de esta "separación" en dos procesos consiste en usar la fórmula de Trotter, que para dos operadores dados A y B está dada por:

$$e^{-\delta\tau(A+B)} = e^{-\delta\tau A/2}e^{-\delta\tau B}e^{-\delta\tau A/2} + \mathcal{O}\left[(\delta\tau)^3\right] \tag{5.37}$$

Esta aproximación se conoce como la **aproximación del tiempo corto**, ATC (*Short Time Approximation, STA*) y es parte fundamental del algoritmo de MCD. De hecho, la ecuación anterior se implementa aplicando primero la función de Green de difusión, $G_{D,f}$ y luego la función de Green de reproducción G_{Rep}:

$$e^{-(\hat{T}+\hat{V}-E_R)\tau} \approx e^{-\hat{T}\tau}e^{-(\hat{V}-E_R)\tau} \simeq G_{D,f}G_{Rep} \tag{5.38}$$

Esta aproximación es exacta cuando $\tau \to 0$. Sin embargo, en estos momentos estamos en una paradoja, pues la solución exacta se encuentra cuando la simulación ha avanzado mucho en el tiempo $\tau \to \infty$. La manera de sobrellevar este problema es iterar un gran número de veces aplicando el operador con un $\delta\tau$ pequeño.

5.4.1. La función de Green de reproducción

La ecuación (5.36) es el propagador del proceso cinético de primer orden. Podemos imaginarnos una caja de Petri en la cual existiera comida para bacterias cerca del centro y un antibiótico en las orillas. Formando un gradiente en el que existieran condiciones favorables al crecimiento en el centro y condiciones desfavorables a éste en las orillas. Las bacterias que se encontraran cerca del centro se reproducirían mientras que las que se encontraran en las orillas morirían a causa del antibiótico.

Podemos pensar en una solución en *estado estacionario* donde las bacterias ya no crecieran ni murieran, estabilizando su población. (Ver la figura 5.1)

La simulación de la función de Green de reproducción se realiza de manera similar a este experimento simulado. Después de mover a un caminante por difusión, calculamos una probabilidad de supervivencia P basada en el propagador de la ecuación (5.36):

$$P = \exp\{-\delta\tau(\hat{V}(\mathrm{R}) + \hat{V}(\mathrm{R}') - 2E_R)/2\}. \tag{5.39}$$

Esta probabilidad de supervivencia puede incorporarse al proceso estocástico de varias maneras [74].

Monte Carlo cuántico de difusión puro

En este caso, se le asigna un peso a cada caminante a lo largo de la propagación. Este procedimiento debe de ser refinado, por que si no, a lo largo de la evolución algunos de los pesos crecerán muy rápido, mientras otros decaerán exponencialmente. Al final, la

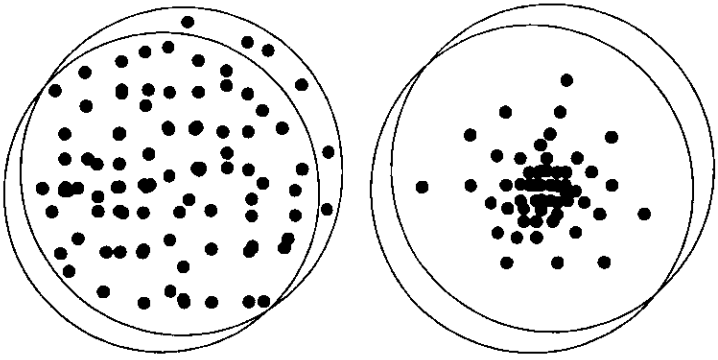


Figura 5.1.: Dos cajas de Petri llenas de bacterias: a) Caja de Petri con una distribución uniforme de bacterias; b) Caja de Petri con una distribución estacionaria de bacterias después de equlibrarse en el gradiente competencia antibiótico/nutriente.

simulación termina con un caminante con un peso dominante [20]. Sin embargo, este método combinado con el método de reproducción ha sido muy bueno para calcular propiedades que no conmutan con el Hamiltoniano y estados excitados [12,68].

Monte Carlo cuántico de difusión por reproducción

En este método, P es interpretada como la probabilidad de que un caminante sobreviva en el siguiente paso de la simulación. Por tanto:

- Si P es menor a 1 el caminante continúa evolucionando con la probabilidad P. Es decir, digamos que un valor de P de 0.2 implicaría un 20% de probabilidades de supervivencia para el caminante en el siguiente paso de simulación.
- Si P es mayor a 1, el caminante continúa evolucionando, pero se crea(n) (un) nuevo(s) caminante(s) con la probabilidad $P - 1$ en las mismas coordenadas.

Ambas probabilidades se pueden codificar en un solo comando:

```
nuevos_caminantes = (int) ( P + aleatorio_uniforme() );
```

Donde `nuevos_caminantes` es el número de caminantes que continúan al siguiente paso en la posición R, `(int)` es una petición al compilador para convertir un número de precisión de punto flotante a entero, `P` es la probabilidad de supervivencia y `aleatorio_unforme` es una función que devuelve un número aleatorio uniformemente distribuido entre 0 y 1.

Este algoritmo de reproducción nos previene de la catástrofe de acumulación de pesos en un sólo lugar, cambiándola por una gran densidad de caminantes en dicho punto.

La función de la energía de referencia E_R que hemos cargado hasta ahora es clara. E_R es ajustada de manera que el número total de caminantes fluctúe alrededor de un valor inicial. El número típico de caminantes usado en las simulaciones es del orden de 100 a 10000.

El algoritmo del Monte Carlo cuántico de difusión se resume en el recuadro *difusión Simple*.

Usemos el ejemplo del átomo de hidrógeno. Los resultados numéricos de la simulación se encuentran en el capítulo 8.

Escojamos una energía de referencia E_R de -0.5 hartrees, la energía exacta del estado basal del átomo. Posteriormente seleccionemos un conjunto de caminantes $R^{(0)}$. Corramos la simulación hasta que la distribución converja a ψ_0[3]. G_{dif} propaga hacia una distribución uniforme en el espacio. G_{rep} es la que construye la estructura de la función de onda. (Ver la figura 5.2)

Algoritmo 5 Monte Carlo cuántico de difusión simple

Salida: Distribución de equilibrio y una estimación de la energía promedio (E) de un conjunto de caminantes (W) usando una función de prueba ϕ_t muestreada por el algoritmo de Metropolis.

```
{v() energía potencial}
for C = N_cini, ..., 1 do
  for i = 1, ..., dim · N_elec do
    W'[C, i] ← W[C, i] + G_(2Dr)
5: end for
  pot1 ← v(W[C])
  pot2 ← v(W'[C])
  G_B ← exp−(0.5 * (pot1 + pot2) − E_ref)
  E ← E + G_B * pot2
10: if floor(G_B + U_0,1) = 0 then
    (W'[N_cam] → W[C])
    N_cam ← N_cam − 1
  end if{Matamos al caminante}
  if N_b ← floor(G_B + U_0,1) > 1 then
15:   for j = 1, ..., N_b − 1 do
        N_cam ← N_cam + 1
        (W'[C] → W[N_cam])
      end for
  end if{Reproducimos al caminante}
20: end for
```

[3]Recuerde que estamos muestreando la función de onda ψ_0 y no la densidad electrónica ψ_0^2

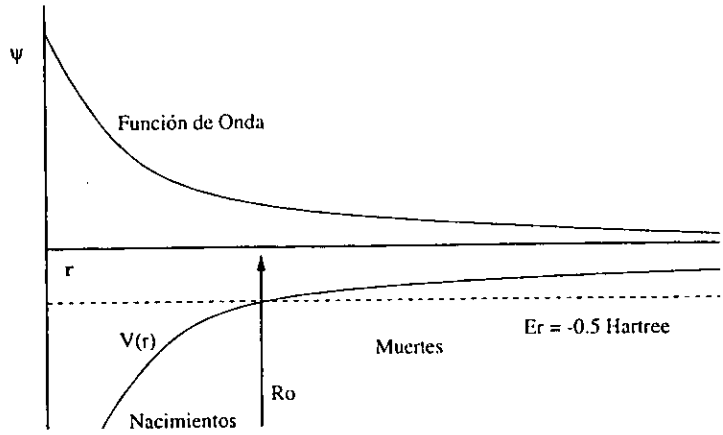


Figura 5.2.: MCD sin muestreo inducido para el átomo de hidrógeno

Problemas del método de Monte Carlo cuántico de difusión sin muestreo inducido

Al igual que en los métodos de integración por Monte Carlo (sección 3.5.1), o los métodos variacionales (sección 4.3) se puede introducir el muestreo inducido al MCD.

La primera aplicación de Muestreo Inducido al Monte Carlo cuántico fue hecha por Malvin Kalos en los 60's [69].

El potencial en la exponencial de la ecuación (5.36) puede variar significativamente de paso a paso e inclusive puede divergir cuando las partículas cochan. Ésto hace que el proceso sea muy ineficiente, y en el caso de las divergencias en el potencial, no bien definido.

5.5. Muestreo Inducido

Propongamos una función de prueba (sección 3.5.1) $\phi_(\mathrm{R})$ y usémosla como una aproximación a la función de onda $\Psi(\mathrm{R},\tau)$. Multipliquemos ambas para obtener un estimador mixto $f(\mathrm{R},\tau)$:

$$f(\mathrm{R},\tau) = \Psi(\mathrm{R},\tau)\phi(\mathrm{R}) \tag{5.40}$$

Es necesario que sepamos cómo muestrear u obtener valores de $\phi(\mathrm{R})$. Si introducimos la función (5.40) en la ecuación (5.15) y rearreglamos, obtenemos:

$$\frac{\partial f(\mathrm{R},\tau)}{\partial \tau} = D\nabla^2 f(\mathrm{R},\tau) - D\nabla\cdot(f(\mathrm{R},\tau)\vec{\mathrm{F}}_{\mathrm{Q}}) + (E_R - E_L(\mathrm{R}))f(\mathrm{R},\tau) \tag{5.41}$$

definiendo a:

- $\vec{F}_Q \equiv 2\nabla\phi/\phi\nabla ln|\phi^2|$ como la *deriva* o *fuerza cuántica*, y a
- $E_L \equiv \hat{\mathcal{H}}\phi/\phi$ como la *energía local* (ver la sección 4.2.2).

Para ver el procedimiento para obtener la ecuación (5.41) consulte la tabla 5.2.

- Partimos de:

$$f(\mathbf{R},\tau) = \Psi(\mathbf{R},\tau)\phi(\mathbf{R})$$
$$\frac{\partial\Psi}{\partial\tau} = D\nabla^2\Psi + (E_R - \hat{V})\Psi$$
$$\frac{\partial f}{\partial\tau} = \frac{\partial\Psi}{\partial\tau}\phi$$

- Sustituyendo:

$$\frac{\partial f}{\partial\tau} = D\phi\nabla^2\psi + (E_R - \hat{V})\Psi\phi$$

- Empleando la identidad:

$$\nabla^2 f = 2\nabla\Psi\cdot\nabla\phi + \Psi\nabla^2\phi + \phi\nabla^2\Psi$$

- Obtenemos:

$$\partial f\partial\tau = D\nabla^2 f - D\Psi\nabla^2\phi - D2\nabla\Psi\cdot\nabla\phi + E_R f \quad \hat{V}\Psi\phi$$

- Reagrupando:

$$\frac{\partial f}{\partial\tau} = D\nabla^2 f + \left(E_R - \frac{D\nabla^2\phi + \hat{V}\phi}{\phi}\right) - D\nabla\cdot\left(f\frac{2\nabla\phi}{\phi}\right)$$

- Obteniendo:

$$\frac{\partial f}{\partial\tau} = D\nabla^2 f - D\nabla\cdot(f\vec{\mathbf{F}}) + (E_R - E_L)f$$

Q.E.D.

Tabla 5.2.: Desarrollo del muestreo inducido para la ecuación de Schrödinger modificada.

La ecuación (5.41), al igual que la ecuación sin muestreo inducido (5.15), tiene términos que pueden ser asociados con los procesos de difusión y de replicación. Sin embargo, la difusión se modifica por un arrastre debido a la fuerza cuántica $\vec{F}_Q$. Esta fuerza modifica el proceso de difusión análogamente a lo que un campo externo (por ejemplo el arrastre de una corriente o del viento) influiría a una partícula en movimiento browniano (ver la sección 4.3).

La segunda cantidad que se obtiene es la energía local $E_L(\mathbf{R})$. Ahora la reproducción se lleva a cabo con el término de energía local en exceso, $(E_T - E_L(\mathbf{R}))$, en

lugar del potencial en exceso, $(E_T - V(\mathrm{R}))$. Debido a que la energía total tiene tanto términos de energía cinética y energía potencial, es mucho más suave que V.

Para la función de onda exacta, las singularidades en la energía cinética y la energía potencial se cancelan, dejando a $E_L(\mathrm{R}) = E_0$ como una constante. En el caso límite, haciendo que $\phi = \psi_0$ y que $E_T = E_0$, el término de reproducción desaparece, dejando una población al equilibrio de caminantes. Sin embargo, para funciones ϕ aproximadas, la cancelación no es completa, y la población cambiará para compensar las deficiencias en la función de prueba.

En particular, los caminantes:

- morirán en las regiones en las que $\phi > \psi_0$, y
- se reproducirán en las regiones en las que $\phi < \psi_0$.

$E_L(\mathrm{R})$ es llamada energía local debido a que depende del punto del espacio (caminante) que escojamos para medirla. En el límite $\phi = \psi_0$, no hay varianza en la energía, y por tanto, sólo se requiere un punto para evaluar la energía. Si las función de prueba no es idéntica a la función del estado basal, la varianza se rige por la calidad de ϕ.

Lo más importante es que el valor esperado de la energía local $E_L(\mathrm{R})$ es E_0 sin importar qué ϕ se escoja (ver la sección 3.8).

El muestreo inducido:

- reduce la varianza e incrementa la eficiencia,
- pero no tiene influencia en el valor de la energía.

La *propiedad de varianza cero* nos garantiza que la energía local converge al valor esperado cuadráticamente [69] para cada caminante. Esta propiedad es la que nos permite realizar simulaciones de sistemas grandes con demandas computacionales aceptables. También sugiere que una inversión razonable de tiempo humano en mejorar la función de prueba es importante para poder escalar el método. La mejoría en la eficiencia del método se podrá ver en la sección de resultados, pero la bibliografía afirma que la eficiencia mejora en 2 o 3 órdenes de magnitud, lo que permite realizar cálculos con cientos e inclusive miles de electrones.

Se puede notar que la ecuación de MCV es idéntica a la ecuación 5.41 sin el término de reproducción. Es fácil de verificar que sin reproducción, $\Psi(\mathrm{R}, \tau) \to \phi$, por lo tanto: $f(\mathrm{R}, \tau) \to \phi^2$. Entonces, el valor promedio de la energía local cuando se elimina la reproducción será $\frac{\langle\phi|\hat{\mathcal{H}}|\phi\rangle}{\langle\phi|\phi\rangle}$. La función de Green con muestreo inducido G^{MI} es una solución a la ecuación 5.41 con la condición de frontera:

$$G_{MI}(\mathrm{R}', \mathrm{R}; \tau) = \delta(\mathrm{R} - \mathrm{R}') \tag{5.42}$$

La parte de G^{MI} que corresponde a la reproducción,G^{MI}_{Rep} se puede obtener reemplazando a V por E_L en la ecuación (5.36):

$$G^{MI}_{Rep}(\mathrm{R}', \mathrm{R}; \delta\tau) = \exp\left\{-(\frac{1}{2}[E_L(\mathrm{R}) + E_L(\mathrm{R}')] - E_R) \cdot \tau\right\} \tag{5.43}$$

El nuevo operador de "energía cinética" en la ecuación (5.41) es:

$$\hat{T}^{MI} = -D\nabla^2 + D(\nabla \cdot \vec{\mathrm{F}}_{\mathrm{Q}}) + D\vec{\mathrm{F}}_{\mathrm{Q}} \cdot \nabla \tag{5.44}$$

Por lo que el nuevo propagador de difusión G^{MI}_{dif} es:

$$G^{MI}_{dif}(\mathrm{R}', \mathrm{R}; \delta\tau) = e^{-\delta\tau\hat{T}^{MI}} \tag{5.45}$$

Si suponemos que el vector $\vec{\mathrm{F}}_{\mathrm{Q}}$ se mantiene constante durante el movimiento, se puede resolver la ecuación diferencial para G^{MI}_{dif}:

$$\frac{\partial G^{MI}_{dif}}{\partial\tau} = -\hat{T}^{MI} G^{MI}_{dif} \tag{5.46}$$

Para obtener:

$$G^{MI}_{dif}(\mathrm{R}', \mathrm{R}; \delta\tau) = (4\pi D\delta\tau)^{-3N/2} \exp\left\{\frac{(\mathrm{R} - \mathrm{R}' - D\delta\tau\vec{\mathrm{F}}_{\mathrm{Q}}(\mathrm{R}'))^2}{4D\delta\tau}\right\} \tag{5.47}$$

5.5.1. La condición de balance detallado

La función de Green en (5.47) viola la propiedad especificada en (5.10):

$$G^{MI}_{dif}(\mathrm{R}', \mathrm{R}; \delta\tau) \neq G^{MI}_{dif}(\mathrm{R}, \mathrm{R}'; \delta\tau) \tag{5.48}$$

debido a que el operador $\hat{T}^{MI}$ no es Hermitiano. Por lo tanto es necesario imponer un balance detallado (3.86) para garantizar el equilibrio en la simulación. El balance detallado se impone aceptando el movimiento de un caminante de R a R' con la probabilidad de Metropolis (sección 3.7.3):

$$A(\mathrm{R}', \mathrm{R}; \delta\tau) \equiv \min\left(1, \frac{|\phi(\mathrm{R}')|^2 G^{MI}_{dif}(\mathrm{R}, \mathrm{R}'; \delta\tau)}{|\phi(\mathrm{R})|^2 G^{MI}_{dif}(\mathrm{R}', \mathrm{R}; \delta\tau)}\right) \tag{5.49}$$

Debido a que algunos movimientos son rechazados, hemos cambiado el paso de tiempo efectivo, por lo que debemos determinar el intervalo de tiempo apropiado a usar para G_{rep}. El intervalo de tiempo efectivo se obtiene multiplicando $\delta\tau$ por la relación de movimientos aceptados a movimientos rechazados.

El algoritmo de MCD con muestreo inducido se puede ver en el recuadro 6

Ahora regresemos al ejemplo del átomo de hidrógeno. En la figura 5.3 esquematizamos la energía local para una función de prueba aproximada [42]. Nótese el comportamiento suave[4] de E_L en comparación con V en el ejemplo sin muestreo inducido.

[4]Un comportamiento suave se traduce en menos fluctuaciones a la hora de realizar la simulación numérica.

Algoritmo 6 Monte Carlo cuántico de difusión con muestreo inducido y la aproximación de os nodos fijos

Salida: Distribución de equilibrio y una estimación de la energía promedio (E) de un conjunto de caminantes (W) usando una función de prueba ϕ_t muestreada por el algoritmo de Metropolis.

{F_Q(W) fuerza cuántica del caminante W}
{psit(W) valor de la función evaluada para el caminanteW }
{sign(a) signo de a}
for $C = N_{cini}, \ldots, 1$ do
  for $i = 1, \ldots, \dim \cdot N_{elec}$ do
    $W'[C,i] \leftarrow W[C,i] + \mathcal{G}_{(2D\tau)} + D\tau F_Q(W[C,i])$
  end for
  if sign(psit(W')) $\neq$ sign(psit(W)) then
    $G_B \leftarrow 1$ {Se rechaza el movimiento por nodos fijos}
  else
    $Q \leftarrow 0$
    for $i = 1, \ldots, \dim \cdot N_{elec}$ do
      $t1 \leftarrow \left[\frac{D\tau}{2}(F_Q W[C,i] - F_Q W'[C,i]) - (W'(C,i) - W(C,i))\right]$
      $Q \leftarrow Q + \frac{1}{2}(F_Q(W[C,i]) + F_Q W'[C,i]) \cdot t1$
    end for
    $A \leftarrow \min\left[1, (psit(W[C])/psit(W'[C]))^2 e^Q\right]$
    if $\min(A, 1.0) > \mathcal{U}_{0,1}$ then
      copiar $(W' \rightarrow W)$
      $G_B \leftarrow \exp -(0.5 * (elocal(W(C)) + elocal(W'(C))) - E_{ref})$
    else
      $G_B \leftarrow 1$
    end if
  end if
  $E \leftarrow E + G_B * elocal(W)$
  if $floor(G_B + \mathcal{U}_{0,1}) = 0$ then
    $(W'[N_{cam}] \rightarrow W[C])$
    $N_{cam} \leftarrow N_{cam} - 1$
  end if{Matamos al caminante}
  if $N_b \leftarrow floor(G_B + \mathcal{U}_{0,1}) > 1$ then
    for $j = 1, \ldots, N_b - 1$ do
      $N_{cam} \leftarrow N_{cam} + 1$
      $(W'[C] \rightarrow W[N_{cam}])$
    end for
  end if{Reproducimos al caminante}
end for

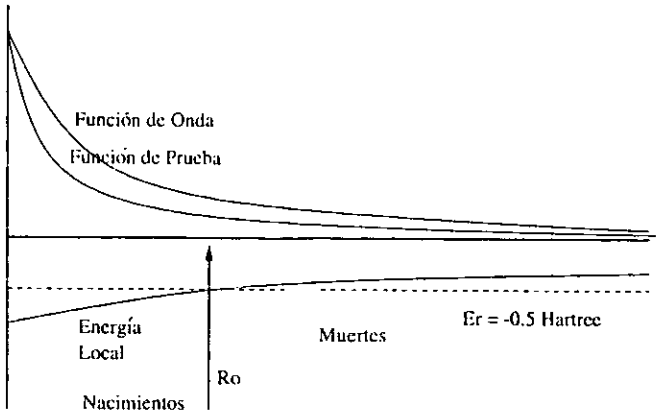


Figura 5.3.: MCD con muestreo inducido para el átomo de hidrógeno

5.6. Evaluación de la energía del estado basal

Analizaremos dos maneras de obtener la energía del sistema, una conocida como la energía de crecimiento [42] o generacional [69] y la energía local misma.

5.6.1. Evaluación de la energía de crecimiento

Sea la población total de caminantes $N(\tau)$. La normalización de la población total al tiempo τ es:

$$N(\tau) = \int f(\mathrm{R}, \tau) d\mathrm{R} \tag{5.50}$$

El cambio en el tamaño de la población al avanzar el tiempo $\delta\tau$ es:

$$N(\tau + \delta\tau) = e^{(E_0 - E_T)\delta\tau} N(\tau) \tag{5.51}$$

Una estimación de E_0 puede obtenerse si se proponen dos tiempos τ_1 y τ_2 y se rearregla la ecuación (5.51) para obtener la energía de crecimiento E_c:

$$E_c = E_T + \frac{1}{\tau_2 - \tau_1} ln \frac{N(\tau_1)}{N(\tau_2)} \tag{5.52}$$

Un problema con E_c es que la población puede variar mucho durante la simulación dando como resultado una larga desviación estándar. En la sección de resultados se puede comparar este estimador con la energía local. Además, si se analizan las razones de convergencia de ambos estimadores con respecto al límite $\tau \to 0$, la energía local converge más rápido que este estimador [87].

5.6.2. La energía local como estimador de la energía basal

El estimador de energía local también es conocido como *estimador mixto*. En una simulación de MCV, el promedio de la energía local depende de la función de prueba, $\langle E_L \rangle_{\phi^2} = E[\phi]$, mientras que en una caminata de Monte Carlo cuántico, sucede que el valor esperado de la energía local es E_0:

$$E_{MCD} = \lim_{\tau\to\infty} \frac{\langle e^{-\tau\hat{\mathcal{H}}/2}\phi|\hat{\mathcal{H}}|e^{-\tau\hat{\mathcal{H}}/2}\phi\rangle}{\langle e^{-\tau\hat{\mathcal{H}}/2}\phi|e^{-\tau\hat{\mathcal{H}}/2}\phi\rangle} = \lim_{\tau\to\infty} \frac{\langle e^{-\tau\hat{\mathcal{H}}}\phi|\hat{\mathcal{H}}|\phi\rangle}{\langle e^{-\tau\hat{\mathcal{H}}}\phi|\phi\rangle} = \frac{\langle\psi|\hat{\mathcal{H}}|\phi\rangle}{\langle\psi|\phi\rangle}$$
$$= \frac{\int f_{exacta}\left[\hat{\mathcal{H}}\phi/\phi\right] d\mathbf{R}}{\int f_{exacta} d\mathbf{R}} \cong \frac{1}{M}\sum_m \frac{\hat{\mathcal{H}}\phi(\mathbf{R}_m)}{\phi(\mathbf{R}_m)} \qquad (5.53)$$

donde $\{\mathbf{R}_m\}$ simboliza las M muestras de $f(\mathbf{R})$ resultantes de la corrida de MCD.

El error en la estimación es dependiente de $\delta\tau$. La manera de eliminarlo es usar algoritmos más complejos y/o evaluar el promedio de la energía local $\langle E_L \rangle$ como una función de $\delta\tau$ y extrapolar a $\delta\tau = 0$.

5.7. Evaluación de valores esperados de otros operadores

Los valores esperados de cantidades distintas a la energía se pueden obtener usando los estimadores mixtos (de MCD) y variacional [24]:

$$\langle\psi|\hat{O}|\psi\rangle \approx 2\frac{\langle\psi|\hat{O}|\phi\rangle}{\langle\psi|\phi\rangle} - \frac{\langle\phi|\hat{O}|\phi\rangle}{\langle\phi|\phi\rangle} + \mathcal{O}[(\psi-\phi)^2] \qquad (5.54)$$

donde $\hat{O}$ es el operador de la cantidad de interés. Para cantidades no negativas, por ejemplo la densidad electrónica, se puede usar la relación:

$$\langle\psi|\hat{O}|\psi\rangle \approx \left[\frac{\langle\psi|\hat{O}|\phi\rangle}{\langle\psi|\phi\rangle}\right]^2\left[\frac{\langle\phi|\hat{O}|\phi\rangle}{\langle\phi|\phi\rangle}\right]^{-1} + \mathcal{O}[(\psi-\phi)^2] \qquad (5.55)$$

El lector se puede referir a la literatura para encontrar varios ejemplos de propiedades que han sido medidas [2, 8, 21, 23], entre ellas se encuentran dipolos, cuadrupolos u octupolos, polarizabilidades, etc.

5.8. MCD para Fermiones

5.8.1. Antisimetría y la aproximación de los nodos fijos

Un concepto fundamental que no ha sido considerado hasta ahora es la antisimetría. Ésta implica que cualquier función de onda fermiónica para más de dos electrones tendrá tanto valores positivos como negativos.

Sin embargo, hasta ahora habíamos dicho que dentro de las restricciones para usar métodos probabilísticos en la solución de ecuaciones integrales es que las distribuciones que se muestrean deben ser no negativas en todo el dominio.

Una generalización *naïve* que extienda el algoritmo de MCD y asigne signos a los caminantes no funciona correctamente. El llamado *problema de los nodos* o *problema de los signos fermiónicos* es uno de los problemas más importantes a resolver en Monte Carlo cuántico y en los métodos de integrales de línea en general.

La aproximación más usada de 1975 a la fecha es la llamada **aproximación de los nodos fijos**. Para poder hablar de ella, debemos de definir *nodo*:

Definición 5.1. *Un nodo fermiónico de una función de onda ψ es un subconjunto del espacio 3-N dimensional para el cual $\psi(\mathrm{R}) = 0$.*

De la definición podemos darnos cuenta que, en general, un nodo es una hipersuperficie $(3N-1)$ dimensional. La aproximación de los nodos fijos consiste en igualar los nodos de nuestra solución $\Psi(\mathrm{R}, \tau)$ a los nodos de la mejor función de prueba ϕ disponible.

Esta aproximación resuelve el problema de las regiones negativas, puesto que inmediatamente se logra que el producto $f = \psi\phi$ sea no negativo en todo el espacio, y por tanto, el algoritmo de MCD continua siendo válido.

La aproximación de los nodos fijos implica, en cierta medida, un acto de fe para la gente ajena al MCC; cambia el problema de la antisimetría por un problema de condiciones a la frontera. Es difícil lograr cualquier condición de simetría en el algoritmo de MCD debido a que, en general, la simetría es una propiedad *no local*, puntos distantes en el espacio deben cumplir cierta(s) condicione(s). Esto no sucede con las condiciones de frontera, pues éstas son *locales* y fáciles de incorporar en la simulación. [69]

Consecuencias de esta aproximación

Se pueden probar las siguientes afirmaciones:

- La energía en MCD con nodos fijos es una cota superior a la energía exacta, es decir, el método es variacional.
- El error de los nodos fijos se reduce cuadráticamente con respecto al error en la superficie nodal de la función de prueba.
- Para una función de prueba exacta, la energía obtenida por el algoritmo es exacta.

La aproximación de los nodos fijos es la única **fundamental**. Todas las demás consideraciones que habíamos mencionado son de carácter numérico y son mejorables con mejores algoritmos y técnicas de programación. Sin embargo, al decidir usar

una restricción como la de los nodos fijos, estamos aceptando un error sistemático en nuestros cálculos.

No se conoce mucho de las propiedades particulares de los nodos, algunos de los artículos relevantes se incluyen en las referencias [22, 25].

La manera de implementar la restricción de los nodos fijos es la siguiente:

- En primer lugar, el muestreo inducido por sí mismo, se encargará de alejar a los caminantes de las regiones nodales, debido a que la fuerza cuántica ($\vec{F}_Q$) los aleja de dichas regiones y los acerca a las regiones donde ψ tenga valores altos.

- Sin embargo, los caminantes cruzan los nodos ocasionalmente. Esto se puede ver en que el arraste es proporcional a $\delta\tau$ mientras que las componentes del vector de difusión son proporcionales a $\sqrt{\delta\tau}$, por lo que a un $\delta\tau$ pequeño, el movimiento por difusión puede ser suficiente para hacer que un caminante cruce un nodo. Esto se manifiesta por un cambio del signo del caminante después de haberse movido.

- La manera *simple* de solucionarlo es rechazar el movimiento y dejar al caminante en la misma posición. Existen maneras más elaboradas de solucionarlo [87].

5.8.2. Error de los nodos fijos y energía de correlación

En la práctica, el error de los nodos fijos es del orden del 5% de la energía de correlación si definimos a la energía de correlación como:

Definición 5.2. *La* energía de correlación *es la diferencia entre la energía de un cálculo de tipo Hartree-Fock y la energía exacta:*

$$E_{corr} = E_{exacta} - E_{HF}$$

La energía de correlación es usada frecuentemente como una medida de la calidady precisión de los métodos de estructura electrónica que estudian los efectos de muchos cuerpos (*many body effects*). La cantidad típica de energía de correlación (95%) es generalmente mayor a la que se obtiene con otros métodos en sistemas con más de 10 o 20 electrones de valencia. La precisión obtenida con estos métodos para calcular diferencias de energías (de enlace, cohesión, excitaciones) llega a ser del 1 al 4% cuando se compara con experimentos [65][5].

[5]Se recomienda leer esta revisión si se desea ahondar en los resultados obtenidos recientemente con el método de MCD en átomos, moléculas y sólidos.

5.9. Cálculos de sistemas grandes y técnicas de pseudopotenciales en MCD

La energía de los electrones en la coraza es mucho mayor que la energía de los electrones en las capas de valencia. Para un método estocástico como el MCD, es muy difícil tratar con la varianza que implican las fluctuaciones en el movimiento de los caminantes cerca del núcleo. Éstos efectos hacen que las variaciones mínimas de energía de los electrones en la capa de valencia sean "opacadas".

Se ha derivado que el tiempo de computadora requerido para realizar cálculos de MCD para átomos crece como Z^q $(6.5 > q > 5.5)$, donde Z es el número atómico. Por lo tanto, los cálculos de sistemas atómicos pesados que incluyen explícitamente los electrones de las capas internas son muy costosos computacionalmente.

Sin embargo, si lo que se requiere son propiedades de valencia, se pueden eliminar los estados de la coraza y sólo calcular los electrones de valencia con diferencias energéticas de 0.05 eV o mejores. Los potenciales de coraza efectivos (*effective core potentials*, PCEs o pseudopotenciales) han sido desarrollados para sobrellevar este problema. En esta aproximación, los electrones de coraza son reemplazados por potenciales efectivos que imitan el efecto de éstos en los electrones de valencia.

En el método de pseudopotenciales, el Hamiltoniano de valencia para un átomo se escribe [37, 39, 69]:

$$\hat{\mathcal{H}}_{val} = \sum_i \frac{-Z_{eff}}{r_i} + \sum_{i<j} \frac{1}{r_{ij}} + \sum_i \hat{\mathcal{W}}_i, \tag{5.56}$$

donde i y j designan a los electrones de valencia, Z_{eff} es la carga nuclear efectiva, y $\hat{\mathcal{W}}$ es radialmente local y dependiente del momento angular,

$$\hat{\mathcal{W}}\phi(\mathbf{R}) = \sum_l V_l(r)\hat{\mathcal{P}}_l\phi(\mathbf{R}), \tag{5.57}$$

donde $\hat{\mathcal{P}}_l$ es un operador de proyección en los estados de momento angular definido y $\phi(\mathbf{R})$ es un pseudoorbital.

El problema con $\hat{\mathcal{W}}\phi(\mathbf{R})$, es que la presencia del operador de proyección hace que éste sea no local en su conjunto. Desafortunadamente para el algoritmo de MCD, la no localidad de este operador implica que el elemento de la matriz $\langle \mathbf{R}|e^{-\tau\hat{\mathcal{W}}}|\mathbf{R}'\rangle$ puede ser negativo para alguna combinación $\mathbf{R}, \mathbf{R}', \tau$, y de hecho, es negativo en varios potenciales efectivos. Este hecho crea otro *problema de los signos* equivalente al problema fermiónico.

Hasta ahora han habido dos maneras de tratar este problema en MCD:

- Definir un *operador local* con propiedades similares o idénticas a $\hat{\mathcal{W}}$. Esta ruta ha sido explorada por Bachelet, Ceperley y colaboradores en su método de pseudoHamiltonianos.

- Eliminar la no localidad de $\hat{W}$ proyectándolo en una función de prueba suficientemente precisa.

La idea subyacente en el segundo método consiste en suponer que las propiedades de la función de prueba serán muy parecidas a las propiedades de la función mixta f, por lo que se aplica el operador no local $\hat{W}$ a la función de prueba ϕ que conocemos localmente, convirtiendo sus propiedades en propiedades locales, y por tanto fácilmente incorporables a un algoritmo de MCD. A este procedimiento se le conoce como *aproximación de localización* y depende cuadráticamente de la calidad de la función de prueba. Esto es similar a lo que sucede con otros operadores (sección 5.7).

5.10. Comparación del método de MC de difusión con otros métodos

En esta colección de referencias, hemos mezclado algunas aplicaciones de MCV y de Monte Carlo cuántico de funciones de Green (MCFG) [5, 78], pues son métodos "hermanos" a MCD.

Las referencias al respecto son vastas [7, 12, 15, 16, 21, 27, 37–39, 41, 42, 44, 53, 56, 59, 61, 64–66, 68, 70, 76, 81] y se le recomienda al lector visitarlas pues la gama de aplicaciones es muy variada. Se han hecho cálculos en átomos ligeros [21, 41, 53, 66, 68] y pesados [27], sólidos [32, 64, 66], moléculas pequeñas [37, 59, 61, 66] cúmulos de carbón y silicio [39, 69], silanos [38], tratamiento de coordenadas de reacción [27], polarizabilidades, dipolos [44], cuadrupolos, sistemas de positrones adiabáticos y no adiabáticos [15, 16], fullerenos, cúmulos de gases nobles [7, 12, 56, 76], osciladores armónicos en cúmulos clásicos [81], sistemas discretos [70], etc.

Incluiremos una tabla de los resultados de un cálculo para un cúmulo de carbón C_{10}, realizado por Lubos Mitas 5.3 [65], complementando los rangos con sus experiencias y revisiones sobre el tema.

En la tabla 5.4 se cita el tiempo humano y computacional usado para realizar un cálculo competitivo con MCD.

Método	E_{corr}[a]	Errores en E_{coh} y E_{enl}[b]	Escalamiento num. de elec. [c]	Tiempo para C_{10}[d]
HF	0	$\approx$ 50%	N^3	14
LDA	No disp.	15-25%	N^3 [e]	1
MCV	$\approx$ 80%	2-10%	$N^3 + \epsilon N^4$ [f]	16
MCD	$\approx$ 95%	1-4%	$N^3 + \epsilon N^4$ [g]	300
CCSD(T)[h]	$\approx$ 75%	10-15%	N^7	1500

[a]Porcentaje de energía de correlación de valencia ($E_{corr} = E_{exacta} - E_{HF}$) estimada por comparaciones con cálculos de alta calidad para sistemas pequeños y datos experimentales

[b]Errores en las energías de enlace y de cohesión

[c]Escalamiento de la demanda computacional total con respecto al número de electrones

[d]Tiempo computacional requerido. La unidad usada es 90 segundos de un procesador Cray C90. El código para HF es GAMESS y para CCSD(T) GAUSSIAN94. La barra de error para los métodos de QMC fue de 0.01 eV/átomo.

[e]Sin tomar en cuenta los recientes avances en la teoría de funcionales de la densidad que escalan proporcionalmente a N.

[f]ϵ es del orden de 10^{-3} a 10^{-4}.

[g]*ídem*.

[h]con un conjunto de base 6-311G*.

Tabla 5.3.: Comparación de distintos métodos de estructura electrónica para el sistema C_{10}.

Tiempo Humano	Tarea	Tiempo de Cómputo
10%	Escoger conjuntos base, PCEs, etc.	0%
40%	Orbitales HF y/o post-HF	10%
40%	optimización MCV	20%
10%	MCD	70%

Tabla 5.4.: Costo temporal para un cálculo de MCD competitivo.

6. Funciones de prueba y optimización

Una de las ventajas de integrar por métodos Monte Carlo es que se pueden usar cualquier tipo de funciones de prueba; no estamos limitados a utilizar combinaciones lineales de orbitales atómicos.

Una buena elección de la función de prueba para el método MCV es esencial, ya que de ésta depende completamente el resultado de los valores esperados. En el caso de MCD existen al menos tres razones importantes para emplear una buena función de prueba:

1. Los errores en la posición de los nodos de la función de prueba, dan lugar al error conocido como *error de los nodos fijos (ENF)*.

2. Es usada en el muestreo inducido, sabemos que al límite en el que la función de prueba es idéntica a la función de onda, la varianza se vuelve cero.

3. Las condiciones de vértice. El error en la simulación puede disminuirse considerablemente usando funciones de prueba con condiciones de vértice adecuadas. Estas condiciones aseguran que la función de prueba tenga un buen comportamiento en las singularidades de la superficie de energía potencial.

D. M. Ceperley [24] propone una relación entre las barras de error en una simulación de MCD y la energía de una simulación de MCV:

$$\epsilon \approx \sqrt{\frac{2(E_v - E_0)}{\tau N_{iter}}} \tag{6.1}$$

Donde τ es el paso en la simulación y N_{iter} el número de iteraciones. Esta relación se deriva del análisis de operadores hecho en la sección 5.7, por lo que lo recomendable es escoger la mejor función de prueba disponible, optimizarla en MCV, y posteriormente realizar la simulación de MCD para recuperar lo más posible de la energía de correlación.

6.1. Funciones de prueba que involucran las coordenadas interelectrónicas

Los métodos de campo medio, como Hartree-Fock no toman en cuenta la correlación entre partículas. El principal motivo para realizar este tipo de aproximaciones ha sido el de poder resolver las integrales involucradas en la función de prueba analíticamente. Los métodos de interacción de configuraciones expanden a la correlación en series que convergen muy lentamente al valor exacto. Por lo que una alternativa es proponer funciones de prueba que involucren la correlación electrónica. Al escoger este tipo de función de prueba, tenemos que abandonar los métodos de autoconsistencia tradicionales y usar métodos estocásticos como MCV o MCD. La inclusión explícita de las coordenadas electrónicas ha sido explorada por Hilleraas y Coolidge en los años 30, Jastrow en los años 50, Boys y Singer en los años 60, etc. En nuestros estudios, trabajaremos con conjuntos base explícitamente correlacionados, abandonando por completo la descripción del campo medio. Nos concentramos en dos tipos de funciones de correlación: las de tipo Jastrow y las exponenciales de correlación. Sin embargo, la gran mayoría de la gente que investiga en el campo de MCC, mezcla una solución de campo medio (uno o varios determinantes de Slater) con factores de correlación del tipo Jastrow. Brevemente describiremos a estos, antes de pasar a las formas con las que nosotros trabajamos.

6.2. Funciones tipo Slater-Jastrow

Como estamos trabajando con fermiones la función de onda debe ser antisimétrica. La función antisimétrica más usada son los determinantes de Slater. Los determinantes de Slater no toman en cuenta la correlación electrónica. Por lo tanto, lo que comúnmente se hace en MCC es multiplicar a estos determinantes por una función que involucra a la correlación. Las funciones más usadas hasta la fecha se deben a Jastrow, quien las introdujo en 1955 [48].

La función de onda se puede describir de la siguiente manera:

$$\phi(\mathrm{R},\sigma) = \sum_{n+1}^{N_{det}} (d_n D_n^{\uparrow} D_n^{\downarrow}) \cdot \prod_i \exp\{J_i\} \tag{6.2}$$

$D_n^{\uparrow}$ y $D_n^{\downarrow}$ son los determinantes de Slater con espín hacia arriba y espín hacia abajo respectivamente. Los determinantes están formados de un número N_{orb} de orbitales, los cuales, a su vez, son combinaciones lineales de funciones base de Slater multiplicadas por armónicos esféricos (Y) centrados en los átomos:

$$\phi_\alpha(\mathrm{r}) = \sum_{\beta=1}^{N_{base}} C_{\alpha\beta} N_\beta r^{n_\beta - 1} e^{\zeta_\beta r} Y_{l_\beta m_\beta}(\mathrm{r}). \tag{6.3}$$

donde:

$$N_\beta = \sqrt{\frac{(2\zeta_\beta)^{2n_\beta-1}}{(2n_\beta)!}} \tag{6.4}$$

es la constante de normalización de la parte radial de las funciones base; ζ es un parametro variacional y cada orbital esta determinado por los números cuánticos n_β, l_β y m_β. Por último, los determinantes $D_n^\uparrow$ y $D_n^\downarrow$ son funcionalmente idénticos y de orden $n/2$.

Orbital	Armónico esférico	Expresión
s	$Y_{0,0}$	$\frac{1}{2}\frac{1}{\sqrt{\pi}}$
p_x	$Y_{1,-1}$	$-\frac{1}{4}\frac{\mathrm{sen}(\theta)\mathrm{sen}(\phi)\sqrt{6}}{\sqrt{\pi}}$
p_z	$Y_{1,0}$	$\frac{1}{2}\frac{\cos(\theta)\sqrt{3}}{\sqrt{\pi}}$
p_y	$Y_{1,1}$	$-\frac{1}{4}\frac{\mathrm{sen}(\theta)\cos(\phi)\sqrt{6}}{\sqrt{\pi}}$
$d_{x^2-y^2}$	$Y_{2,-2}$	$\frac{1}{8}\frac{(\mathrm{sen}(\theta))^2\,\mathrm{sen}(2\,\phi)\sqrt{30}}{\sqrt{\pi}}$
d_{xz}	$Y_{2,-1}$	$-\frac{1}{4}\frac{\mathrm{sen}(\theta)\cos(\theta)\mathrm{sen}(\phi)\sqrt{30}}{\sqrt{\pi}}$
d_{z^2}	$Y_{2,0}$	$\frac{1}{16}\frac{\left(12\,(\cos(\theta))^2-4\right)\sqrt{5}}{\sqrt{\pi}}$
d_{yz}	$Y_{2,1}$	$-\frac{1}{4}\frac{\mathrm{sen}(\theta)\cos(\theta)\mathrm{sen}(\phi)\sqrt{30}}{\sqrt{\pi}}$
d_{xy}	$Y_{2,2}$	$\frac{1}{8}\frac{(\mathrm{sen}(\theta))^2\,\mathrm{sen}(2\,\phi)\sqrt{30}}{\sqrt{\pi}}$

Tabla 6.1.: Armónicos esféricos simples

Los términos de la función de correlación pueden expandirse en términos de primer orden:

$$J_1 = \sum_i j_0(\mathbf{r}_i), \tag{6.5}$$

de segundo orden:

$$J_2 = \sum_{i<j} j_{ee}(\mathbf{r}_i, \mathbf{r}_j) + \sum_{i,I} j_{eN}(\mathbf{r}_i, \mathbf{r}_I), \tag{6.6}$$

y de tercer orden:

$$J_3 = \sum_{i<j<k} j_{eee}(\mathbf{r}_i; \mathbf{r}_j; \mathbf{r}_k) + \sum_{i<j,N} j_{eeN}(\mathbf{r}_i; \mathbf{r}_j \mathbf{r}_I) + \sum_{i,I<J} j_{eNN}(\mathbf{r}_i; \mathbf{r}_I \mathbf{r}_J) \qquad (6.7)$$

donde e representa a los electrones y N a los núcleos.

En la mayoría de las aplicaciones se usan funciones de prueba de segundo orden, es decir, que involucran a pares de partículas, sin embargo existen estudios que estudian los efectos de las interacciones de órdenes 3 [88] y 4 [45]. Estos estudios concluyen que la inclusión de términos de orden superior mejora mucho la energía en la función de onda.

En el caso general de segundo orden, la forma de J_2 es:

$$J_2 = \exp\left\{\sum_{i>j} \frac{a\mathbf{r}_{ij}}{1 + b\mathbf{r}_{ij}}\right\} \qquad (6.8)$$

donde a y b son parámetros variables.

6.2.1. Condiciones de vértice

Las funciones de onda exactas presentan una propiedad conocida como *condiciones de vértice*[1]. Éstas se deben a que la primera derivada de la interacción electrón-electrón es discontinua en el origen. Para ejemplificar, veamos el caso de el átomo de hidrógeno En él, la función radial obtenida tiene la forma $\phi = e^{-ar}$. Si graficamos esta función, encontramos el vértice a la que nos referimos (ver la figura 6.1).

Las funciones de Jastrow son excelentes candidatos a funciones de prueba ya que las exponenciales reproducen correctamente la interacción entre partículas cerca del origen.

Se han propuesto otras formas funcionales para representar la correlación, como gaussianas que dependen de las coordenadas entre partículas; el problema con éstas es que la adición de términos converge muy lentamente al valor exacto pues éstas reproducen mal las condiciones de vértice.

6.3. Abandonando el determinante de Slater

Si abandonamos el determinante de Slater, el término de Jastrow por sí solo no cumple las condiciones de antisimetría requeridas. Debemos de construir funciones de prueba que sean antisimétricas, y que tengan una forma parecida al factor de Jastrow. Además, como sabemos que nuestras funciones no son exactas[2], sería bueno pensar una manera de extender la descripción de la correlación.

[1] En inglés se denominan *cusp conditions*.

[2] ¡Si fueran exactas habríamos resuelto el problema de muchos cuerpos!

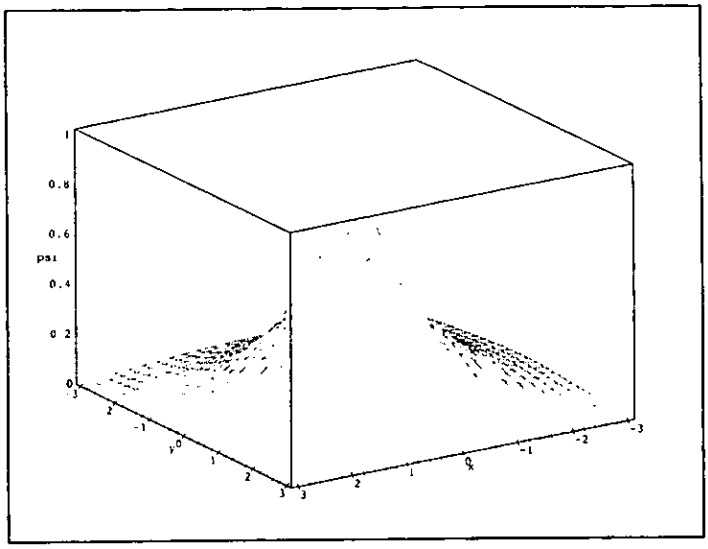


Figura 6.1.: Gráfica de la función radial del Hidrógeno (no normalizada y representada en $2D$), ejemplificando las condiciones de cúspide en la función $1s$.

La primera forma de extender la descripción de la correlación consiste en pensar en interacciones de tercer, cuarto y quinto orden como ya mencionamos en la sección anterior.

La segunda forma involucra el pensar en un conjunto de funciones base:

$$\phi = \sum_{i=1}^{L} c_i \chi \tag{6.9}$$

donde cada término χ debe cumplir con las condiciones de antisimetría. Bressanini et. al. han estudiado este tipo de sistemas con detalle [10, 14, 15]. La forma para χ que ellos proponen para una familia de funciones de prueba que involucra tanto a factores de Jastrow o exponenciales de correlación es:

$$\chi_i = \hat{A}\left\{[\hat{O}\, f_i(\mathbf{r})\, \exp(\mathbf{k}_i \cdot \mathbf{d})\, \Theta_{N,S,M_S}]\right\}. \tag{6.10}$$

En la ecuación anterior, $\hat{A}$ es el operador de antisimetría, $\hat{O}$ el operador asociado a la simetría espacial, $\mathbf{k}_i$ es el *iésimo* componente del vector de parametros, $\mathbf{d}$ es el vector de las distancias electrón-electrón y electrón-núcleo. Θ_{N,S,M_S} es una función propia de los operadores de spin $\hat{S}$ y $\hat{S}_z$, correspondiendo a la multiplicidad correcta.

La función $f_i(\mathrm{r}_{e,n})$ es necesaria cuando la simetría del sistema no es esférica:

$$f_i(\mathrm{r}_{e,n}) = \prod_{j=1}^{N}\prod_{k=1}^{M}(x_j - X_k)^{\alpha_{ijk}}(y_j - Y_k)^{\beta_{ijk}}(z_j - Z_k)^{\gamma_{ijk}} r_{jk}^{\delta_{ijk}}. \tag{6.11}$$

donde α, β, γ y δ son enteros iguales o mayores que cero y x, y y z son las coordenadas electrónicas y X, Y y Z son coordenadas nucleares, N y M son los índices electrónico y nuclear respectivamente.

Con esta construcción basta para proponer funciones de onda en las que sólo se propongan interacciones puntuales entre partículas. Las deficiencias en la función de prueba se vuelven menos graves al aumentar el número de funciones base, y por tanto, incrementar el número de parámetros variacionales.

6.4. Exponenciales de correlación

En 1960, Boys y Singer usaron funciones gaussianas explícitamente correlaciondas. El problema con este tipo de funciones es que reproducen mal las condiciones de vértice. Se requiere un número muy grande de funciones para alcanzar resultados aceptables.

Si hacemos que el término b de una exponencial de Jastrow de segundo orden, se vuelva cero, obtenemos una función de la forma:

$$C = e^{k_i r_{ab}}. \tag{6.12}$$

La función anterior cumple con la condición de ser singular en el vértice en la que dos partículas ocupan el mismo punto, y ésto se puede ver en que su derivada direccional es una constante:

$$\frac{1}{\phi}\frac{\partial \phi}{\partial \mathrm{r}_{ab}}\Big|_{r_{ij}=0} = c. \tag{6.13}$$

Por lo que la idea de eliminar un parámetro variacional de las funciones de Jastrow, conservando un comportamiento correcto con respecto a las condiciones de vértice es atractivo, ya que el costo computacional de los métodos de optimización variacional crece rápidamente al ir aumentando el número de parámetros.

Estas son las funciones de prueba que usaremos en nuestras simulaciones.

6.5. Optimización

La optimización es un problema matemático en general. Se da en distintas áreas de la ciencia, que incluyen a todos aquellos problemas sujetos a ciertas restricciones. La optimización matemática se aplica a diversos problemas como el de teoría de gráficas, el tráfico en las ciudades, redes telefónicas, diseño de circuitos impresos, etc.

En el caso de MCC, estudiaremos la optimización de la función de prueba. Ésto es, en el caso de MCV la que nos dará la mejor aproximación a la energía. En el caso de MCD, la varianza obtenida en la energía depende exclusivamente de la calidad de la función de prueba, y el error de la posición de los nodos.

Existen varias maneras de optimizar la función de prueba. Se puede optimizar la energía, $\langle E \rangle$, o la varianza de la misma. La optimización de la varianza tiene la ventaja de que conocemos el valor mínimo al que debemos llegar (cero). Además, los requisitos computacionales para minimizar la varianza son mas modestos que los requeridos para optimizar la energía [88].

Minimizamos la varianza de la energía local (E_L):

$$\sigma^2_{opt} = \frac{\sum_i^s \left[\frac{\hat{H}\psi(i)}{\psi(i)} - E_g\right]^2 w_i}{\sum_i^s w_i} \tag{6.14}$$

donde:

$$w_i = \left|\frac{\psi_i}{\psi_0}\right| \tag{6.15}$$

y E_g es el mejor estimado que se tiene de la energía del sistema, en la práctica es el resultado de un cálculo MCV anterior, se necesitan de 500 a 2000 caminantes.

Como una medida de qué tan *buena* es la función de prueba, podemos introducir un factor Q:

$$Q = -log_{10}\frac{\sigma}{E_0} \tag{6.16}$$

La función de prueba es exacta en el límite en el que $Q \to \infty$.

Basta proponer una función de prueba $\phi(\alpha, \beta, \ldots, \nu)$ que dependa de n parámetros lineales[3] o no lineales.

Posteriormente, se corre una simulación de Monte Carlo Variacional, en la que se espera que la población de caminantes llegue al equilibrio. Después, se calcula la energía local del ensamble. Se varían los parámetros para intentar minimizar la energía o la varianza.

Después, se corre una simulación con estos nuevos parámetros, y se vuelve a optimizar. Al repetir este tipo de ciclo varias veces, llegamos a un momento en el que no hay mejorías significativas en la cantidad minimizada (varianza o energía), por lo que la optimización se da por terminada.

En MOISS usamos la minimización no lineal de Powell, que puede ser consultada en la literatura [73].

[3]Muchas veces si los parámetros son lineales, se pueden construir sistemas de ecuaciones que nos ayuden a encontrar los mejores coeficientes. Este es el caso particular de los coeficientes lineales de las funciones base de las exponenciales de correlación.

7. Desarrollo del programa de simulación MOISS

Computers let you make more mistakes faster
than any other invention in human history,
with the possible exception of handguns and tequila.
-Mitch Radcliffe

7.1. El paradigma

MOISS (*Monte Carlo Integration and Sampling Software*) es el conjunto de programas y documentación que desarrollamos con el propósito de proveer a la comunidad científica y académica con un programa gratuito, de libre acceso y fácil de usar para realizar cálculos de Monte Carlo Cuántico. Los objetivos del proyecto son:

modularidad: Muchos científicos programan códigos pequeños que son apropiados para resolver un problema en particular. Cuando necesitan resolver otro problema, hacen una copia de su primer programa y la modifican para que realice una nueva tarea. El seguir este modelo de trabajo hace que terminen con varios programas pequeños muy especializados. En el caso de MOISS, hemos seguido un camino alterno. Estamos creando un programa de MCC "general" al que sólo se le tengan que agregar pedazos de código relevantes a la descripción de un nuevo sistema, conservando la funcionalidad anterior.

Esto se logra programado modularmente. La mayoría de los códigos de Monte Carlo Cuántico siguen algoritmos que pueden ser generalizados, como se puede ver en los capítulos 4 y 5 . De manera que podemos realizar un algoritmo que llame a funciones "particulares" según sea el caso pertinente a la simulación.

libertad: MOISS ha sido desarrollado usando el sistema operativo GNU/Linux [86] con las utilerías del proyecto GNU.[82]. Está liberado bajo la licencia GPL (*General Public License*) [1]. Por tanto, MOISS es considerado parte del gran acervo de

[1] contenida en el apéndice C.

programas denominados software libre (*free software*) o de código disponible (*Open Source*). En el apéndice B se ahondará en el asunto.

paralelizable: Como ya se ha mencionado en la literatura [23, 42, 50, 53] el código de Monte Carlo Cuántico es fácilmente paralelizable[2]. Esto implica que la ejecución del programa se puede realizar tanto en una computadora con varios procesadores o en un *cúmulo* (*cluster*) de procesamiento en paralelo usando alguna biblioteca de paralelización.

MOISS comprende dos programas distintos:

moiss: El programa que realiza los cálculos de Monte Carlo Cuántico. Está escrito bajo las normas de ANSI C [55] y es fácil de portar a cualquier plataforma UNIX.

gmoiss: La interfase gráfica prepara los archivos de entrada para el programa *moiss* y lo ejecuta con los parámetros adecuados. El usuario puede analizar el resultado de los cálculos dentro de ésta.

Los programas están completamente desacoplados. Es posible correr *moiss* en una computadora remota y preparar sus archivos de entrada desde una computadora con Linux y GNOME.

7.2. Capacidades del software

En esta sección hablaremos de las capacidades del programa *MOISS* a la fecha de la realización de esta tesis. Antes de comenzar la descripción del programa, definiremos los términos que usaremos a lo largo de esta sección. El formato para presentar las capacidades del software es parecido a los artículos publicados en la revista Computer Physics Communications. En particular, hemos seguido el formato de presentación de otro programa de Monte Carlo Cuántico, *Talus* [83].

7.2.1. Terminología

sistema: Entendemos como sistema al conjunto de descriptores físicos (número de partículas, número de dimensiones, potencial entre ellas, potencial externo, etc.) que definen a un problema de interés en *MOISS*.

caminante: Un caminante (*walker*, o *psip: psi-particle*) es un punto en el espacio fase. En *MOISS*, llamamos caminante también a toda la información computacional asociada con ese punto, que no necesariamente es la misma que la información matemática formal.

[2] En inglés se les conoce como métodos *embaranssingly parallel*.

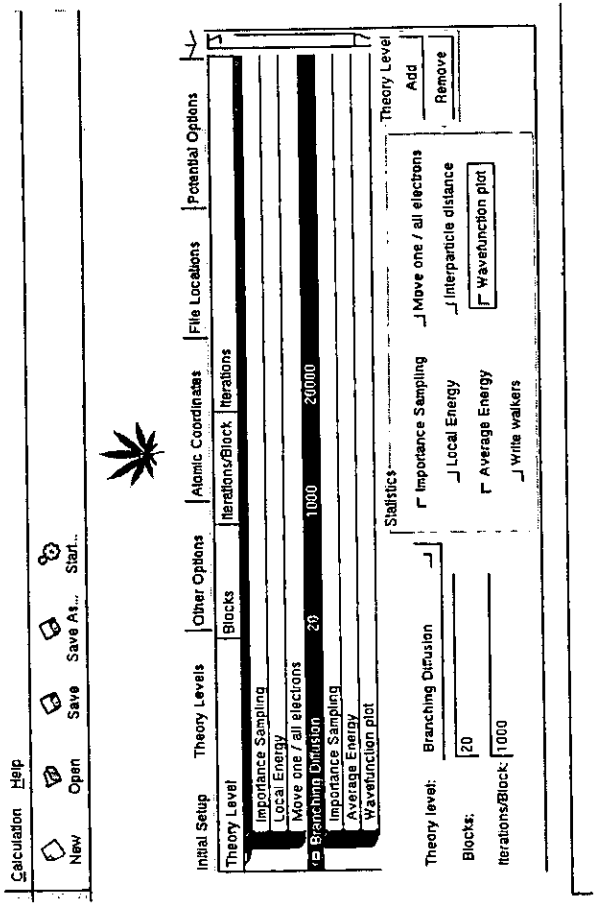


Figura 7.1.: Interfase Gráfica *GMOISS*: Seleccionando los niveles de teoría.

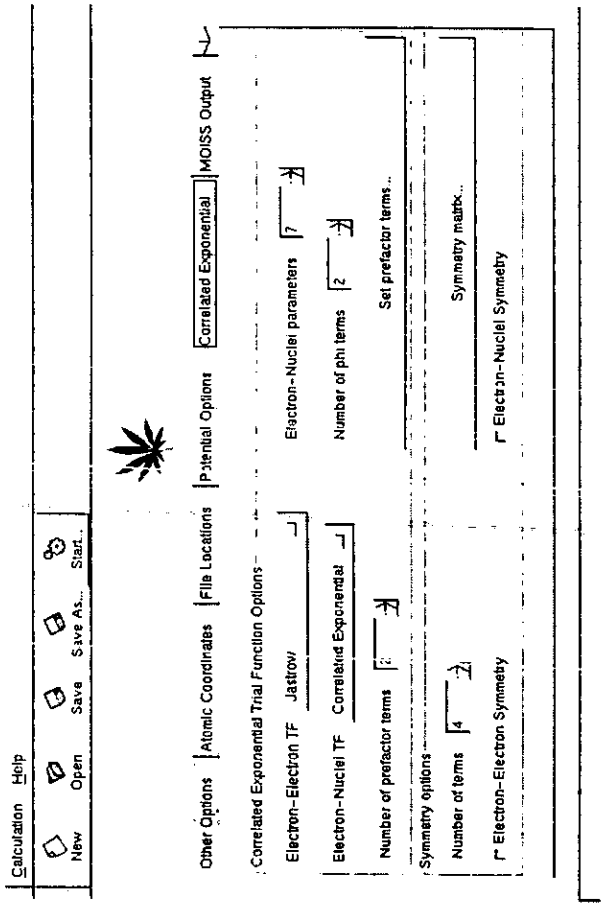


Figura 7.2.: Interase Gráfica *GMOISS*: Opciones de exponenciales de correlación.

ensamble: Llamamos ensamble al conjunto de caminantes que integran la simulación. Al igual que con los caminantes, extendemos la definición de ensamble (*ensemble* o *conjunto*) puramente matemática, a una definición que incluye los datos computacionales que puedan ser asociados a éste.

simulación: Llamamos simulación al conjunto de datos generales que conforman una representación completa de una una corrida realizada sobre un sistema en nuestro programa.

método: Un método es un algoritmo de simulación de MCC, desde los variacionales hasta los de difusión. Extendemos esta definición al considerar a la *optimización variacional* (ver el capítulo 6) como un método.

estadística: Es el grupo de propiedades cuánticas de interés que se van a promediar mientras se ejecuta un método. Éstas pueden ser evaluadores de la energía, medidas de la distancia media entre partículas, recolección de las coordenadas de los caminantes en un histograma tridimensional, etc.

corrida: Es el conjunto ordenado que describe el conjunto de métodos, condiciones particulares, y descripción de las estadísticas a aplicar a un sistema.

En la figura 7.3 se puede observar una esquematización de los distintos elementos del programa MOISS.

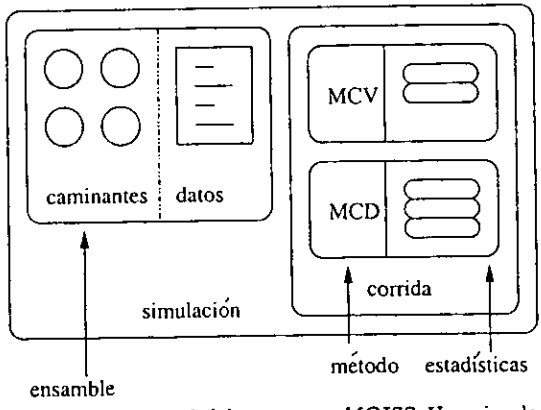


Figura 7.3.: Organización conceptual del programa *MOISS*. Una simulación comprende a un ensamble y a la corrida que se hará sobre éste. Cada corrida es un conjunto ordenado de métodos que se aplican a los caminantes, recolectando distintas estadísticas en cada uno de ellos. El ensamble es un conjunto de caminantes y los datos asociados a éstos.

7.2.2. Algoritmos (Métodos)

El programa de cálculo MOISS implementa los siguientes algoritmos de Monte Carlo cuántico:

MCV con balance detallado (Metropolis): El algoritmo implementado es el algoritmo de MCV clásico. Se propone un movimiento y es aceptado o rechazado por el cociente de aceptación del algoritmo de Metropolis. En esta implementación, no se toma en cuenta a la función de prueba para mover a los caminantes, por lo que la cantidad de rechazos es mayor que en el caso acelerado. [Código en el archivo: `vmc.c`]. Este procedimiento está descrito en la sección 4.2.2, y es el primer caso en la figura 4.2.

MCV con balance detallado y aceleración tipo Fokker-Planck: Este algoritmo implementa la *aceleración*[3] del tipo Fokker-Planck. Los movimientos son propuestos a partir de un desplazamiento Gaussiano ξ con media de cero y varianza de $2D\delta\tau$ sumado a un desplazamiento debido a una fuerza de arrastre del tipo $\vec{F} = 2\frac{1}{\phi}\nabla\phi$. Este algoritmo tiene un paso de aceptación de Metropolis para eliminar el error relacionado con el tamaño de paso finito. [Código en el archivo: `vmc.c`]. Este procedimiento está descrito en la sección 4.3, y es el tercer caso en la figura 4.2.

MCD sin muestreo inducido: Este algoritmo es poco eficiente para fines de producción de resultados con exactitud química[4]. Su incorporación al código tiene una doble funcionalidad. Primero, es la implementación del formalismo de MCD más sencilla y eso ayuda a que alguien interesado en aprender MCC pueda estudiarlo sin dificultad. Segundo, este procedimiento no requiere de una función de prueba para obtener resultados. Esta propiedad es muy importante para el estudio de sistemas modelo para los que no se conoce una función de prueba adecuada. La implementación de un potencial arbitrario n-dimensional es muy sencilla[5] y los resultados obtenidos son inmediatos. La varianza asociada a este método es elevada y las fluctuaciones en la población de caminantes son considerables debido a las singularidades en el potencial. Este algoritmo está limitado a dos partículas fermiónicas; no se pueden simular funciones de onda con nodos ya que no se usan funciones de prueba. [Código en el archivo `walk.c`, pueden verse implementaciones viejas en el archivo `method.c`]. La descripción teórica se encuentra en la sección 5.3.

[3] A este tipo de métodos de MCV se les conoce también como métodos acelerados.

[4] Nos referimos a la precisión necesaria para poder predecir diferencias de energías en reacciones químicas.

[5] En el caso de un oscilador de Morse, o un oscilador anarmónico puede ser de unas diez o veinte líneas de código.

MCD con muestreo inducido y la aproximación de los nodos fijos: El algoritmo está basado en el algoritmo de Reynolds, et al.[74], con algunas mejorías implementadas por Umrigar, et al.[87]: si un caminante intenta cruzar un nodo, el movimiento es rechazado en lugar de ser eliminado como en el algoritmo de Reynolds. Esto reduce el error $\hat{O}(\tau)$ asociado a esta restricción. Éste es el algoritmo que generalmente se usa para recuperar la energía de correlación faltante después de usar un MCV con una función de prueba de alta calidad. Debido a la creación y destrucción de caminantes, el costo computacional del MCD es mucho mayor (ver la tabla 5.4) [Código en el archivo `walk.c`]. El método se analiza con detalle en la sección 5.5.

7.2.3. Las estadísticas

El programa *MOISS* permite que las estadísticas recolectadas en cada método sean distintas. Entre las estadísticas implementadas en MOISS se encuentran:

energía local: Evaluación de la energía local como es descrito en la sección 5.6.

energía de crecimiento: Evaluación de la energía de crecimiento (sección 5.6.1).

función de onda: La función de onda se recolecta guardando la posición de los caminantes en la simulación en un histograma n-dimensional.

guardar los caminates en un archivo: La posición de los caminantes se puede ir guardando entre método y método, en el caso de que el cálculo se interrumpa por alguna falla en el sistema de cómputo. En este caso, *MOISS* puede reanudar el cálculo a partir de la última vez que se guardaron los caminantes.

En versiones subsecuentes de *MOISS* se incluirán nueva estadísticas. En este momento, la implementación permite agregar nuevos algoritmos de recolección de datos fácilmente. Éstas se pueden codificar bajo demanda; dependiendo de los parámetros que el investigador desee medir a lo largo de la simulación de un sistema en particular.

7.3. Guía del usuario

La versión de *MOISS* actual es la versión 0.5 (para información sobre como conseguirla, referirse al apéndice D).

7.3.1. Instalación

Las plataformas en las que se ha compilado el programa *MOISS* son:

GNU/Linux. Distribución Red Hat. Versión 5.2 con GNOME 1.08. Con los siguientes procesadores:

- Intel Pentium II a 233 MHz
- AMD K6 a 200 MHz.
- Digital Alpha 21164/500 MHz.

En principio:

- La interfase gráfica debe poder ser compilada en cualquier plataforma Linux con GNOME 1.08 instalado.
- El programa de cálculo puede ser instalado en cualquier plataforma UNIX sin tener que hacer cambios en el código fuente.
- Probablemente pueda ser compilado en otras plataformas con pocos cambios al código fuente.

Si desea instalar el programa en alguna otra plataforma UNIX, debe de obtener las bibliotecas del proyecto GNOME del sitio `http://www.gnome.org`, así como la biblioteca científica del proyecto GNU (*GNU Scientific Library*), GSL de
`http://sourceware.cygnus.com/gsl/`,
y seguir el procedimiento de instalación de éstas. En particular, para poder compilar *moiss*, se requieren las bibliotecas `libPropList-0.8.3`, `gsl-0.41` y `glib-1.2.1`, cuyos códigos fuente pueden ser obtenidos de `http://rufus.w3.org`.

Instalación en formato RPM

El formato RPM es un formato muy conveniente para manejar archivos binarios en UNIX. En el sitio de distribución de `moiss`, podrá encontrar distribuciones binarias del mismo en este formato. Si usted utiliza el formato RPM y tiene una instalación de RedHat 6.0, no necesita instalar `moiss`, simplemente obtenga los siguientes archivos del sitio ftp de *moiss*, `ftp://moiss.pquim.unam.mx/pub/moiss/`:

```
gmoiss-0.5-1.i386.rpm
gsl-0.4.1-1.i386.rpm
```

y ejecute el siguiente comando:

```
rpm -Uvh gmoiss-0.5-1.i386.rpm gsl-0.4.1-1.i386.rpm
```

El programa *MOISS* será instalado en su sistema automáticamente.

Descompresión

Descomprima el archivo `moiss-version.tar.gz` . Si se usa la versión GNU del comando `tar`, y se cuenta con el comando `gunzip`, esto se puede realizar de la siguiente manera:

```
tar xzvf moiss-0.5.tar.gz
```

Estructura de directorios

Usted encontrará la siguiente estructura de directorios:

```
ABOUT
AUTHORS
COPYING
INSTALL
README
TODO
acconfig.h
acinclude.m4
autogen.sh
config.h.in
configure.in
stamp.h.in
doc/
gmoiss-src/
moiss-src/
input-files/
intl/
macros/
po/
```

El contenido es:

- Los archivos `ABOUT`, `AUTHORS`, `COPYING`, `INSTALL`, `README` y `TODO` incluyen información general sobre el programa, así como la licencia GNU.

- Las entradas `acconfig.h`, `acinclude.m4`, `autogen.sh`, `config.h.in`, `configure.in` y `stamp.h.in`, son parte de los archivos que requieren las utilerías `autoconf` y `automake` de GNU para realizar su trabajo de detección de la configuración del sistema.

- Los directorios `macros/`, `po/` e `intl/` contienen respectivamente los macros de configuración requeridos por `autoconf`, y las cadenas para la internacionalización de la interfase gráfica *gmoiss*.
- El directorio `moiss-src/` contiene el código fuente del programa de simulación.
- `gmoiss-src/` contiene el código fuente de la interfase gráfica.

Configuración con autoconf/automake

Para generar los archivos de compilación `Makefile`,detectar cuáles son las funciones de la biblioteca C específicas de su sistema, y cerciorarse que las bibliotecas de desarrollo de GNOME estén instaladas, debe teclear:

```
./autogen.sh
```

Si las bibliotecas no se encuentran en su lugar canónico, probablemente tenga que revisar el comando:

```
./autogen.sh --help
```

Por ejemplo. si usted tiene una instalación de GNOME en el directorio `/opt/gnome`, y gtk instalado en */usr/local/gtk/* debe de escribir el comando:

```
./autogen.sh --prefix=/opt/gnome --with-gtk-prefix=/usr/local/gtk
```

Si después de ejecutar `autogen.sh`, aparece el siguiente mensaje:

```
Now type 'make' to compile Moiss: Molecular Integration
and Sampling Software
```

está listo para proceder con la compilación.

Compilación

Para compilar ambos programas, simplemente escriba:

```
make
```

Si usted desea compilar sólo el programa de cálculo *moiss*, debe de entrar:

```
make moiss
```

Si usted desea instalar los binarios del sistema y la documentación, basta teclear:

```
make install
```

7.3.2. Creación de un archivo de entrada

Para crear un archivo de entrada, se debe de arrancar la interfase gráfica *gmoiss*. Para hacer esto, escriba:

```
gmoiss
```

Aparecerá una ventana de la interfase gráfica en la que usted deberá escoger los parámetros adecuados para la simulación. Si usted desea correr a MOISS con el sistema por omisión, simplemente apriete el botón de `Save...` y se guardará un archivo llamado `gmoiss-input` en el directorio de trabajo. Debe de proporcionar un nombre para la simulación que desea realizar (`jobname`). El directorio en el que se grabarán los archivos de salida de la simulación también debe ser especificado (`job directory`). La opción por omisión en este caso es `/tmp/moiss/`. Puede presionar el botón `Choose directory...` para poder seleccionar otro directorio o crear uno nuevo.

7.3.3. Opciones de cálculo

Debido a que *MOISS* se encuentra en continuo desarrollo, las opciones de cálculo pueden variar considerablemente. En caso de duda, se recomienda consultar el código fuente mismo, o mandar un correo a la dirección `moiss@eros.pquim.unam.mx`.

Menú principal

El menú principal tiene las siguientes entradas:

Calculation: El submenú de cálculos contiene las opciones pertinentes a abrir, guardar y lanzar cálculos.

Help: El submenú de ayuda contiene información sobre los autores actuales y la licencia del programa. Eventualmente contendrá una traducción de este documento al idioma inglés.

Creación de un nuevo cálculo

Para crear un nuevo cálculo basta seleccionar la opción Calculation — New[Alt + N] partiendo del menú principal. Debajo del logo de MOISS aparecerá un "cuaderno virtual" con distintas opciones para especificar un cálculo.

Para navegar en el cuaderno, basta presionar sobre éste con el botón derecho del ratón para poder acceder las distintas páginas. Una manera alternativa es seleccionándolas con las flechas que se encuentran en la parte superior derecha del cuaderno.

Al ir enumerando cada parámetro, aprovecharemos para describir brevemente la forma en que aparece en el archivo de entrada *gmoiss-input*.

Páginas de opciones

Initial Setup

- **Problem**

Potential Type: El seleccionar un potencial es sólo pertinente a los problemas de carácter educativo. La lista de potenciales que se incluye sirven para realizar simulaciones de MCD sin muestreo inducido. La colección incluye potenciales generalizables a varias dimensiones y potenciales que sólo se pueden usar en una dimensión. En general, los potenciales no están diseñados para tratar más de una partícula, a menos que su descripción especifique lo contrario.

Harmonic Oscillator: Se implementa el potencial $V(x_1, x_2, x_i) = \sum_i x_i^2$, el índice i denota el número de coordenadas especificadas en la opción **Dimensions**. En este potencial se puede usar la función de prueba Gaussiana (escogiendo **Gaussian** bajo **Trial Function Type**), ya que el resultado correcto es una gaussiana con un valor de α de 1.0. Es muy buen ejemplo didáctico para analizar la varianza de la simulación con respecto a la función de prueba. En *gmoiss-input* aparece como:

```
Potential = "Harmonic Oscillator";
```

Potential Well: Una caja de barreras energéticas infinitas en n dimensiones. La fórmula para $V(x_1, x_2, x_i)$ es:

$$V(x_1, x_2, x_i) = \begin{cases} 0 \text{ si } b > x_i > a \\ \infty \text{ si } b < x_i \text{ o } x_i < a \end{cases}$$

Las dimensiones se pueden especificar en la opción **Dimensions**. El tamaño de la caja no se puede elegir en la interfase gráfica en la versión 0.5. Sin embargo, éste puede determinarse editando directamente el el archivo *gmoiss-input* de la siguiente forma:

```
Potential= "Potential Box";
"Potential Box X" = 1.000000;
"Potential Box Y" = 1.000000;
"Potential Box Z" = 1.000000;
```

Hydrogen Atom: Se resuelve el problema del átomo de hidrógeno en 3 dimensiones con la aproximación de Born-Oppenheimer. En este problema, el potencial es $V(x, y, z) = \frac{1}{\sqrt{(x^2+y^2+z^2)}}$. Se puede usar una función de prueba del tipo de Slater. Se puede variar el parámetro α para cambiar la calidad de la función de prueba. En *gmoiss-input*:

```
Potential = "Hydrogen atom";
```

Tunnel: En este ejemplo didáctico se obtienen las funciones de onda de una caja de potencial con un bloque de altura h y lados a y b sumergido en una caja de potencial. La función de onda para este problema es difícil de obtener analíticamente, por lo que el método de MCC es una buena alternativa para mostrarla a una audiencia que aprende Química Cuántica. Le llamamos *tunnel* pues este ejemplo sirve muy bien para observar el efecto de tuneleo cuántico. El potencial es h en cualquier punto dentro del área que definen a y el fin de la caja. Es 0 en cualquier punto dentro de la caja de potencial y ∞ fuera de ésta (ver figura 7.4). Este potencial sólo se puede especificar hasta un máximo de 2 dimensiones. En *gmoiss-input*:

```
Potential = "Tunnel";
"Potential Box X" = 1.000000;
"Potential Box Y" = 1.000000;
"Tunnel Potential A" = 0.500000;
"Tunnel Potential H" = 100.000000;
```

Bump: Este potencial de "salto" es muy didáctico para ver la forma que toma la función de onda al presentarse el fenómeno de "tunelaje". En este caso, a diferencia del anterior, la función de onda continúa después de la caja, pues ésta termina en el parámetro b. Para mas detalles de la forma de este potencial, refiérase a la figura 7.5. Este potencial se puede especificar hasta un máximo de dos dimensiones. Para *gmoiss-input*:

```
Potential = "Bump";
"Potential Box X" = 1.000000;
"Potential Box Y" = 1.000000;
"Tunnel Potential A" = 0.500000;
"Tunnel Potential B" = 0.700000;
"Tunnel Potential H" = 100.000000;
```

Helium Atom: El átomo de helio, esta simulación debe realizarse con un número de dos partículas, y se debe de escoger una de las dos funciones de prueba (**Trial Function Type**), ya sean **Hellium (Kellner)** o **Helium (Eckart & Hilleras)**. El átomo de helio puede ser tratado con más precisión si se utilizan las funciones de prueba de exponenciales de correlación. En *gmoiss-input:*

```
Potential="Helium";
```

Initial Positions: Esta opción determina la posición de los caminantes al inicio de la simulación. Las variantes posibles son:

Gaussian Random: Los caminantes son distribuidos al azar en una gaussiana n dimensional con una varianza σ dada por el parámetro **Gaussian Spread**, que se encuentra en la página **Other Options**. Seleccione esta entrada cuando no tenga un conocimiento previo de la posición probable de los caminantes. En *gmoiss-input*:

```
"Initial Position" = "Random";
```

Coordinates: Los caminates son colocados en las coordenadas iniciales de la página **Atomic Coordinates**. Esta opción no es útil en la versión 0.5; está planeada para simulaciones de cúmulos bosónicos, por ejemplo agrupaciones de átomos de gases nobles.

```
"Initial Position" = "Coordinates";
```

File: Si se conoce un conjunto de coordenadas de los caminantes y se desea reiniciar la simulación a partir de éste, se debe de usar **File**. En general, el nombre del archivo del que se leen las coordenadas se llama `restart`. El nombre anterior se puede modificar usando la opción de comandos `gmoiss -r file`, donde `file` es el nombre del archivo de donde se pueden leer las coordenadas. En *gmoiss-input:*

```
"Initial Position" = "File";
```

Trial Function Type. Esta entrada especifica la función de prueba de muestreo inducido, ϕ a usar a lo largo de la simulación. Los detalles sobre éstas se pueden consultar en el capítulo 6.

Gaussian: Esta función de prueba es una distribución normal (ver la sección 3.2.2) de la forma $\phi = e^{-\alpha x^2}$ donde el parámetro α se puede especificar en la opción **Gaussian TF alpha** de la sección **Other Options**.

Hydrogen (Slater): Esta función de prueba es el orbital $1s$ de tipo Slater (Slater Type Orbital). En *gmoiss-input*:

```
"Trial Function" = "Hydrogen (Slater)";
```

Helium (Kellner): Esta es una de tantas funciones de prueba propuestas para el Helio, [72]. La forma de esta función es:

$$\phi = e^{-\frac{27}{16}\frac{r_1+r_2}{a_0}}$$

En *gmoiss-input:*

```
"Trial Function" = "Helium (Kellner)";
```

Helium (Eckart & Hilleraas): Esta función tiene la forma:

$$\phi = e^{-\frac{27}{16}\frac{r_1+r_2}{a_0}} \cosh\left(\frac{r_1 - r_2}{a_0}\right)$$

En *gmoiss-input:*

```
"Trial Function" = "Helium (Eckart & Hilleraas)";
```

Corr. Exponential/Jastrow: El escoger esta entrada habilita el uso de funciones de prueba del tipo exponenciales de correlación, Jastrow, Jastrow generalizada o una combinación de éstas. Para la descripción de las funciones, refiérase a las sección 6.2; para las opciones de configuración, a la sección Correlated Exponential. En *gmoiss-input*:

```
"Trial Function" = "Corr. Exponential/Jastrow:";
```

Dimensions: En este diálogo se debe especificar el número de dimensiones para los problemas educativos: 1, 2 o 3. En *gmoiss-input*:

```
"Dimensions" = "n";
```

donde n es 1, 2, o 3.

Job name: Esta opción es de suma importancia. Aquí se especifica el nombre del subdirectorio en el cual se guardará la salida del programa *moiss*. Éste es relativo al especificado en Output base directory en la sección File Locations. Si el directorio especificado ya existe, *gmoiss* mostrará una ventana de error al momento de intentar lanzar un cálculo.

- Options

Particles: El número de partículas (electrones) en la simulación. En versiones subsecuentes de *MOISS* esta opción podrá especificar el número de bosones, positrones, etc. a simular. En ciertos casos, el programa *moiss* puede deducir automáticamente el número de partículas de las coordenadas atómicas en Atomic Coordinates y de la carga total de la molécula o átomo. En *gmoiss-input*:

```
"Particles" = "n";
```

donde n es un entero positivo.

Walkers: La cantidad de réplicas o caminantes que se simulan en el sistema. En los algoritmos de reproducción la cantidad de caminantes puede variar a lo largo de la simulación. En *gmoiss-input*:

```
"Walkers" = "n";
```

donde n es un entero positivo.

Timestep (dt): El intervalo de *tiempo imaginario*, ($\delta\tau$) que se usará a lo largo del programa. El error de la simulación, así como la relación de aceptación-rechazo de Metropolis depende de este parámetro. Usualmente es recomend-

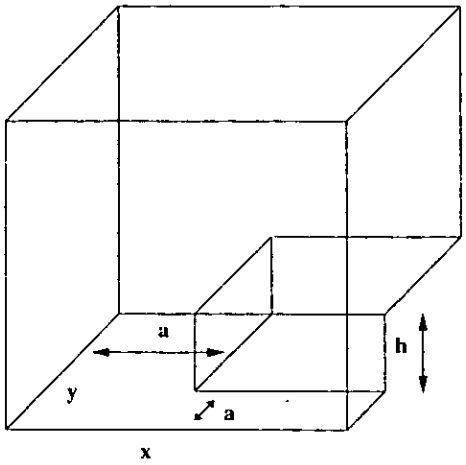


Figura 7.4.: El valor del potencial de tunelaje en dos dimensiones.

able hacer varias simulaciones y extrapolar a $\delta\tau \rightarrow 0$. En versiones futuras de *MOISS* se podrá hacer esto automáticamente. En *gmoiss-input*:

```
"Timestep" = "f";
```

donde f es un número fraccionario. Usualmente el rango de $\delta\tau$ va de 0.001 a 0.5, dependiendo del tamaño y características del sistema.

Theory Levels

En esta sección hay un árbol que representa la corrida que se va a realizar. Los botones Theory Level Add y Theory Level Remove agregan y eliminan respectivamente a los distintos niveles de teoría en cada cálculo.

La especificación los niveles de teoría se hace a través de los diálogos:

Theory level: Esta opción especifica el nivel de teoría de la rama del árbol seleccionada en azul (ver la sección 7.2.2 para mayor detalle). Las opciones son:

Variational: MCV con paso de Metropolis.

Variational FP: MCV acelerado de tipo Fokker-Planck con paso de Metropolis.

Variational Optimization: Optimización Variacional de la varianza en la energía.

Branching Diffusion: MCD con paso de Metropolis, en el caso de *prender* la estadística Importance Sampling, se implementa la aproximación de los nodos fijos.

Pure Diffussion: Opción experimental en la versión 0.5

Path Integral Monte Carlo: Opción experimental.

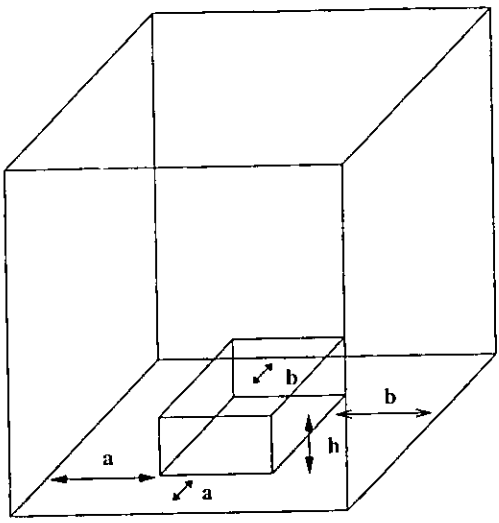


Figura 7.5.: El valor del potencial de salto en dos dimensiones.

Blocks: Especifica el número de bloques de pasos (**iterations/block**) a realizar. Los pasos se dividen en bloques para el cálculo de la varianza. La división en bloques elimina la autocorrelación [31].

Iterations/Block: El número de pasos a simular por bloque. El número total será el producto de ésta y la opción anterior, y se desplegará bajo la columna del árbol **Iterations**.

Las opciones bajo **Statistics** especifican las estadísticas a evaluar al realizar cada cálculo. Si se presiona el recuadro a la izquierda de cada una de ellas, ésta se anexará al nivel de teoría seleccionado en azul. Si se presiona de nuevo, la estadística no se calculará. Las estadísticas que se pueden calcular son:

Importance Sampling: Activa o desactiva el muestreo inducido en el nivel de teoría.

Average Energy: Calcula la energía local promedio.

Wavefunction plot: Grafica la función de onda en un histograma n dimensional al final de la corrida. El archivo en el que se graba se llama `psi`. Si se desea cambiar su nombre, se debe de correr `moiss -h opcion` donde `opcion` es el nombre a usar en vez de `psi`.

La representación del árbol de opciones en *gmoiss-input* es:

```
Calculation = (
```

```
{
  "Calculation Type" = "Variational FP";
  Blocks = 100;
  Steps = 50;
  Statistics = ("Average Energy","Wavefunction Plot");
},
{
  "Calculation Type" = "Variational Optimization";
},

...

)
```

Existe un mapeo uno a uno entre la representación gráfica del árbol en *gmoiss*, la sintaxis en el archivo de entrada *gmoiss-input*, y la manera de operar interna del programa de cálculo *moiss*.

Other Options

Esta sección contiene parámetros asociados a las opciones de las dos páginas anteriores.

- Other Options

  Max. branching: Es el número máximo de caminantes que se reproducen en un paso de MCD. Es útil cuando las poblaciones están lejos del equilibrio, y por lo tanto, se crean y destruyen muchos caminantes. En *gmoiss-input*:

  ```
  "Maximum Branches" = "n";
  ```

  donde n es un número entero positivo.

  Timestep convergence: Esta es una opción experimental. La idea es ir reduciendo el intervalo temporal $\delta\tau$ poco a poco durante la simulación. En *gmoiss-input*:

  ```
  "dt convergence factor" = "f";
  ```

  donde f es un número fraccionario entre 0 y 1.

  Gaussian Spread: Especifica el rango de distribución de la inicialización aleatoria de caminantes. En *gmoiss-input*:

  ```
  "Gaussian Spread Range"  = "f";
  ```

  donde f es un número fraccionario.

Gaussian TF alpha: Es el valor α asociado a la función de prueba $\phi = e^{-\alpha r^2}$ que se selecciona como Gaussian bajo Trial Function Type. En *gmoiss-input*:

```
"Gaussian Trial Function Alpha"  = "f";
```

donde f es un número fraccionario.

Random seed: Es la semilla del generador de números aleatorios (ver la sección 3.3.1). En *gmoiss-input*:

```
"Random Number Generator Seed"  = "n";
```

donde n es un número entero.

Diffusion Constant: Es la constante de difusión D en Monte Carlo Cuántico de Difusión. Su valor en unidades atómicas es 0.5. Es raro cambiarla y el motivo de incluirla se debe a la futura implementación de simulaciones con otro tipo de partículas. En *gmoiss-input*:

```
Diffusion Constant"  = 0.5;
```

- Graphics Options Estas opciones son pertinentes a la gráfica de la función de onda. Ésta es un histograma rectangular de n divisiones por dimensión.

  Divisions / dimension: El número n de divisiones correspondientes a cada dimensión en la gráfica. (Ver la figura 7.6). En *gmoiss-input*:

  ```
  "Graph: Divisions per Dimension"  = 200;
  ```

  Minimum histogram value: El valor en unidades atómicas del mínimo de la caja. En *gmoiss-input*:

  ```
  "Graph: Minimum"  = "-3.0";
  ```

  Maximum histogram value: El valor en unidades atómicas correspondiente al máximo de la caja. En *gmoiss-input*:

  ```
  "Graph: Maximum"  = "3.0";
  ```

Atomic Coordinates

En esta sección se especifican las coordenadas de los átomos involucrados en la simulación. La interfase gráfica es muy sencilla. Basta presionar el botón de Remove para eliminar un átomo de la lista. Para agregar un átomo a la lista, basta presionar el botón de Add.

Al seleccionar Add, aparecerá una tabla periódica en la que se puede seleccionar cualquier átomo de la tabla. Al editar la sección de Coordinates, se debe de especificar la localización del núcleo en coordenadas cartesianas.

La molécula de LiH se representa en *gmoiss-input* de la siguiente forma:

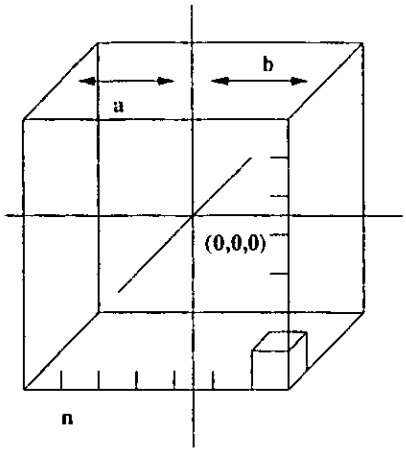


Figura 7.6.: Esquema del histograma tridimensional para la función de onda. El valor a corresponde a la opción **Graph: Minimum**, b a **Graph: Maximum**, y n a **Divisions per Dimension**.

```
 "Atomic Coordinates" = (
{x = 0.;y = 0.;z = 0.; "Atomic Number" = 3;},
{x = 0.;y = 0.;z = 3.05.; "Atomic Number" = 1;}
);
```

File Locations

En esta sección se especifican los directorios asociados a la simulación.

Parameters file path: El archivo de parámetros para las funciones de prueba de MOISS generalmente se llama `psi.inp` y se encuentra en el mismo directorio que la simulación (`./psi.inp`), si se desea cambiar esta opción, basta presionar el botón de Change... que mostrará una ventana en la que se puede elegir otro archivo. En *gmoiss-input*:

```
"Parameters File Path" = "./psi.inp";
```

'moiss' binary location: Esta opción especifica el comando a ejecutar para llamar al programa *moiss*. La opción por defecto es `moiss`. En *gmoiss-input*:

```
"Moiss Binary Path" = "moiss";
```

Output Base Directory: Cambiando este parámetro se puede especificar en donde se desea que se creen los directorios para las simulaciones de *moiss*. En *gmoiss-input*:

```
"Output Files Path" = "/tmp/moiss/";
```

Potential Options En esta página se recolectan las opciones que tienen que ver con el cálculo de la parte relacionada a la energía potencial en el Hamiltoniano.

Electronic Repulsion Factor: Este parámetro sirve para cambiar el *apantallamiento* entre electrones. En todas las simulaciones pertinentes a esta tesis, el valor que debe usarse es `1.0`. En *gmoiss-input*:

```
"Electronic Repulsion Scale" = 1.000000;
```

Change mass & charges... Esta opción permite cambiar la masa y carga de las partículas a simular. La masa es relativa a la masa del electrón m_e, y la carga a la carga del mismo, q_e. En *gmoiss-input*:

```
"Particle Charges" = ("-1.0", "-1.0", ... );
"Particle Masses" = (1.0, 1.0, 1.0, ... );
```

Correlated Exponential

En esta página se describen las opciones relativas a las funciones de prueba de exponenciales de correlación. Para la información sobre la forma de estas funciones, así como el significado de cada parámetro, ver a la sección 6.4.

- **Correlated Exponential Trial Function Options**

Electron-Electron TF: En este parámetro se selecciona la función de prueba para usar en los términos interelectrónicos. Las opciones son:

Correlated Exponential: Exponenciales de correlación. Requieren un parámetro por interacción electrón-electrón.

Jastrow: Funciones del tipo Jastrow. Requieren dos parámetros por interacción electrón-electrón.

Generalized Jastrow: Variante experimental de las funciones de tipo Jastrow. Requieren tres parámetros por interacción.

En *gmoiss-input*:

```
"Electron-Electron TF" = "Jastrow";
```

Electron-Nuclei TF: La función de prueba para usar en las interacciones electrón-núcleo. Las opciones disponibles son:

Correlated Exponential: Exponenciales de correlación. Requieren un parámetro por interacción electrón-electrón.

Jastrow: Funciones de tipo Jastrow. Requieren dos parámetros por interacción.

En *gmoiss-input*:

```
"Electron-Nuclei TF" = "Jastrow";
```

Electron-Nuclei parameters: El número de parámetros a usar en las interacciones electrón-núcleo. En un futuro, MOISS será capaz de calcular este parámetro automáticamente. En *gmoiss-input*:

```
"CE Electron-Nuclei Parameters" = 6;
```

Number of phi terms: El número de términos en la combinación lineal en las funciones de prueba de exponenciales de correlación. En *gmoiss-input*:

```
"CE Number of Terms" = 1;
```

Number of prefactor terms: El número de términos a usar en el prefactor de simetría de las funciones de exponenciales de correlación. En *gmoiss-input*:

```
"CE Prefactor Terms" = 2;
```

Set prefactor terms...: Al presionar este botón, aparecerá un diálogo de edición en el que se pueden especificar los coeficientes para los términos del prefactor. En *gmoiss-input*:

```
"CE Prefactors" = ((1.000),(1.000));
```

- Symmetry options

Number of terms: Esta opción especifica el número de términos que se usan del operador de permutación para las funciones de exponenciales de correlación. En *gmoiss-input*:

```
"CE Sym: Number of Terms" = 4;
```

Symmetry matrix: Esta es la matriz de antisimetría requerida por las funciones de prueba de exponenciales de correlación. Crece como $N!$, siendo N el número de electrones. El primer término de la matriz es el signo del renglón completo. La manera en la que se representa en *gmoiss-input* la matriz de simetría para la molécula de LiH es:

```
"CE Sym: Matrix" = (
  (1, 0, 1, 2, 3),
  ("-1", 0, 1, 3, 2),
  ("-1", 0, 2, 3, 1),
```

```
    (1, 0, 3, 2, 1),
    ("-1", 1, 0, 2, 3),
    (1, 1, 0, 3, 2),
    (1, 1, 2, 3, 0),
    ("-1", 1, 3, 2, 0),
    ("-1", 2, 0, 1, 3),
    (1, 2, 1, 0, 3),
    (1, 2, 3, 0, 1),
    ("-1", 2, 3, 1, 0),
    (1, 3, 0, 1, 2),
    ("-1", 3, 1, 0, 2),
    ("-1", 3, 2, 0, 1),
    (1, 3, 2, 1, 0)
);
```

Electron-Electron Symmetry: Especifica si existe simetría interelectrónica. En *gmoiss-input*:

```
"CE Electron-Electron Symmetry" = Yes;
```

Electron-Nuclei Symmetry: Especifica si existe simetría nuclear. En *gmoiss-input*:

```
"CE Electron-Nuclei Symmetry" = No;
```

Moiss Output

Cuando se elige la opción Start... aparece en esta pantalla una ventana en la cual se puede analizar la salida estándar (`stdout`) del programa *moiss*. En versiones futuras de MOISS se podrá grabar a un archivo o editar.

Guardado de un cálculo

Para guardar un cálculo, se debe seleccionar la opción Calculation — Save, ésto grabará un archivo llamado `gmoiss-input` en el directorio actual de trabajo, (`cwd`). Si se desea guardar en otro directorio o cambiar de nombre, se debe de usar la opción Calculation — Save as....

Corrida de cálculos

Para correr un cálculo, se debe usar Calculation — Start.... En la sección Moiss Output aparecerán los resultados de la simulación.

7.3.4. El programa de cálculo: *moiss*

El programa de cálculo tiene las siguientes opciones de línea de comando:

input: Especifica el archivo de entrada a usar, en vez de `gmoiss-input`.

restart: El archivo del cual se leerá la configuración de los caminantes después de la corrida.

histogram: El nombre del archivo donde se guarda la función de onda.

energy: El archivo en el que se guarda la función $E_{Local} = f(\tau)$.

walkers: El archivo donde se registra la posición de los caminantes. Puede ser el mismo que `restart`, de manera que si la simulación se cae, ésta pueda continuar desde el último bloque simulado.

Las anteriores se pueden consultar usando el comando `moiss -help`:

```
[aspuru@oxido moiss-src] ./moiss --help
Message: MOISS (C) 1999 R.A. Perusquia, A. Aspuru,
 C. Amador, D. Bressanini
 MOISS
 Monte Carlo Integration and Sampling Software
 http://eros.pquim.unam.mx/moiss/

 -i <a> --input <a>     <a> is the name of the input file
 -r <b> --restart <b>   <b> is the name of the restart file
 -p <c> --histogram <c> name of the histogram output file
 -e <d> --energy <d>    name of the energy vs. t output file
 -w <e> --walkers <e>   name of the walkers positions output  file
```

7.4. Datos de salida

La salida típica del programa a la terminal se presenta a continuación:

```
Trial Function: Gaussian
Branching Diffusion
Blocks  Steps     Energy       %
   1     200   0.0960119374   10.0
   2     400   0.1941269599   20.0
   3     600   0.2783498012   30.0
   4     800   0.3332381877   40.0
```

```
   5     1000    0.3523751021    50.0
   6     1200    0.3974778580    60.0
   7     1400    0.4419129614    70.0
   8     1600    0.4593114326    80.0
   9     1800    0.4853777594    90.0
  10     2000    0.5175603741    100.0
Average Energy:  0.355574237370  (+/-) 0.134565438535
Branching Diffusion
Blocks  Steps      Energy         %
  10     2000    0.4611088475    10.0
  20     4000    0.5264859371    20.0
  30     6000    0.4378637629    30.0
  40     8000    0.4863926131    40.0
  50    10000    0.5036239374    50.0
  60    12000    0.5110891010    60.0
  70    14000    0.5015158580    70.0
  80    16000    0.5058738245    80.0
  90    18000    0.4796897631    90.0
 100    20000    0.5007077211    100.0
Average Energy:  0.495253677696  (+/-) 0.023738336878
Message: CPU time: 126.20 secs. Effective time: 143.83 secs.
```

7.5. Detalles de implementación

7.5.1. La interfase gráfica (gmoiss)

La interfase gráfica utiliza las bibliotecas para el programador del proyecto GNOME [46]. Éste provee una abstracción de la interfase gráfica UNIX para el programador siguiendo la jerarquía:
Unix → X Window → glib → gdk → gtk → GNOME → aplicación.

En nuestro caso, la aplicación es nuestra interfase gráfica GMOISS.

glib Biblioteca general de UNIX, era incluida con **gtk** [85] hasta la versión 1.1.0 en la cual se separó. Algunas de las ventajas que le proporciona al programador son:

portabilidad La biblioteca glib facilita portar el código de GMOISS y MOISS a distintos ambientes de tipo UNIX e inclusive al ambiente Windows 95/98/NT usando los compiladores de GNU. Por ejemplo, una llamada del tipo:

```
matriz = g_malloc(entradas * sizeof(double));
```

es análoga a la llamada `malloc` estándar de la biblioteca C. Con la diferencia de que si el sistema al que se porta el programa no provee la función `malloc`, glib tiene macros de compilacíon que reemplazarán las llamadas a `g_malloc` por sus respectivos equivalentes en el sistema operativo donde se realiza la compilación.

programación por contrato Esta biblioteca contiene llamadas para verificar precondiciones y postcondiciones. Éstas son muy útiles para poder encontrar la fuente de un error que de otra manera sería muy difícil de detectar. Un ejemplo en pseudocódigo sería:

```
precondición( x > 0)
a = sqrt (x)
return a
```

estructuras de datos La biblioteca provee al programador de C con una amplia gama de posibilidades en lo que respecta a estructuras de datos. Facilita el uso de listas simplemente y doblemente ligadas, árboles, diccionarios, etc. Provee funciones para operar en estas listas.

gtk The Gimp Toolkit. Esta biblioteca provee una interfase gráfica moderna para el sistema X.Window [6]. Posee las siguientes ventajas:

desarrollo continuo GTK está en continuo desarrollo. La versión 1.2 fue liberada en Marzo de 1999. Continuamente se corrigen errores y se agregan nuevas características.

basado en glib El que GTK esté basado en glib ayuda a tener una coherencia para el programador.

autoconf/automake Utilerías del proyecto GNU que facilitan la portabilidad y compilación en distintos sistemas UNIX.

libPropList Biblioteca de "listas de propiedades" (*property lists*"). Esta biblioteca nos permite guardar los archivos de entrada para MOISS en una manera estándar, legible y fácilmente procesada por alguna utileria externa. Un archivo de entrada de casi cualquier programa de cuántica revisado, con excepción de MPQC [47] tiene formatos de entrada crípticos que requieren que el usuario tenga una curva de aprendizaje complicada.

7.5.2. El programa de cálculo (moiss)

El programa de cálculo está ligado con las siguientes bibliotecas:

glib Descrita en la sección 7.5.1.

gsl GNU Scientific Library. Esta biblioteca es un proyecto de la Free Software Foundation para generar un conjunto de herramientas para el programador científico. Las opciones que esta biblioteca le da al científico son:

Manejo de Errores: Contiene funciones de soporte que hacen más sencilla la programación de funciones matemáticas y la detección de errores en ellas.

Generación de números aleatorios: Contiene 29 generadores de números aleatorios, de los cuales seis tienen calidad de simulación. En futuras versiones de MOISS se permitirá al usuario escoger el generador de números aleatorios entre los 29 que contiene GSL.

Distribuciones de números aleatorios: Esta biblioteca provee 21 distintas distribuciones de números aleatorios. La que usamos en MOISS, la distribución normal, usa el algoritmo de Box-Muller, que es mucho mas lento que el algoritmo reportado por Fishman[31], que está implementado en nuestro archivo `rand.c`.

Funciones de estadísticas: Incluye funciones para obtener los momentos y propiedades de distribuciones estadísticas.

Transformadas de Fourier: Contiene varios algoritmos para realizar transformadas de Fourier.

Raíces de funciones: Provee de métodos para encontrar ceros de funciones que son útiles en los algoritmos de optimización variacional.

Funciones especiales: En futuras versiones de MOISS se requerirán varias de las funciones que provee esta biblioteca, como las funciones de Legendre, Laguerre, etc.

Vectores y Matrices: Tiene soporte para matrices y vectores que checan automáticamente su tamaño, lo cual es bueno para evitar errores de programación. En el momento que el usuario requiere velocidad, se puede desactivar el chequeo con una opción de compilación.

Histogramas: La biblioteca realiza histogramas en una y dos dimensiones. En un futuro pensamos extender su funcionalidad al tratamiento de histogramas en tres dimensiones haciéndola útil para MOISS.

Integración Numérica: Contiene algoritmos de integración numérica por métodos de Monte Carlo.

libPropList Descrita en la sección 7.5.1.

Implementación de las estructuras de datos

En MOISS usamos las siguientes estructuras de datos para representar al problema físico:

Arreglos La mayoría de los arreglos en MOISS son arreglos dinámicos. La ventaja en usar arreglos dinámicos es que no se desperdicia memoria, y éstos pueden ser cambiados de tamaño a lo largo de la simulación con llamadas a la función `realloc`. Los caminantes se guardan en un arreglo, y seguimos el algoritmo de compactación propuesto por Tobochnik *et al.* [36]:

- El arreglo se recorre del primer al último caminante.
- En cada caso, el algoritmo especifica los siguientes casos:
  - Si el caminante muere, se copia el último caminante de la lista a su sitio, y se disminuye en uno un contador del número de caminates "vivos".
  - Si no se reproduce ni muere, no se realiza ninguna operación.
  - Si se reproduce n veces, se agregan n caminantes al final del arreglo.
- El arreglo se checa periódicamente para hacerlo más grande en caso necesario.

Listas doblemente ligadas Las listas ligadas son estructuras de datos que contienen tres apuntadores: uno al siguiente elemento de la lista, uno al elemento previo y otro a los datos que contiene la lista. En el caso de *moiss*, las listas ligadas se usan para la representación interna de las corridas y las estadísticas. Esto permite tener flexibilidad en el número de elementos a evaluar y reduce el número de evaluaciones condicionales. En el caso de las estadísticas, usamos apuntadores a funciones. (Ver la figura 7.7). Para esta implementación, aplicamos el patrón de diseño de software llamado *estrategia* [35]. En el patrón de *estrategia* lo que se busca es definir una familia de algoritmos, encapsularlos y hacerlos intercambiables. Este patrón permite que el algoritmo varíe independientemente de los clientes que lo usan.

Estructuras de datos En *MOISS* usamos una estructura de datos para la simulación, una para el ensamble y otra para cada caminanate. Estas estructuras de datos encapsulan los datos y los vuelven autoconsistentes. Éstas contienen a su vez listas ligadas y arreglos, como en el caso de la estructura MonteCarlo, que contiene una lista ligada de funciones de estadísticas. Varias instancias de MonteCarlo son agrupadas en una lista que conforma la corrida. La corrida es parte de la estructura simulation.

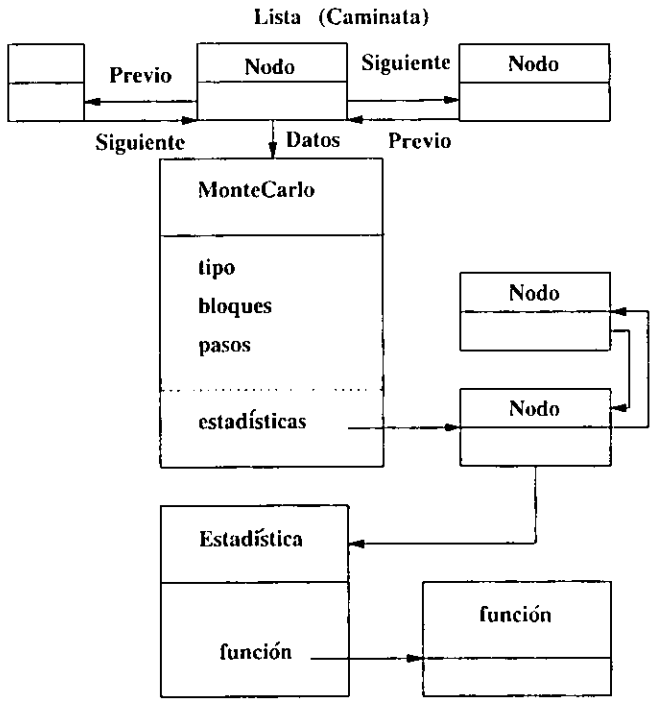


Figura 7.7.: Listas ligadas y su aplicación a las corridas y las estadísticas en *MOISS*.

Uso de apuntadores a funciones

El lenguaje C soporta el uso de apuntadores a funciones. Esta característica es muy conveniente cuando se intenta generalizar un programa usando el patrón de diseño *estrategia*. En *moiss*, cuando se lee el archivo de entrada en el formato libPropList, se inicializan los apuntadores a funciones. Actualmente usamos apuntadores a funciones para:

- Generalizar las caminatas. Existe una función llamada `walk_block` que llama a un apuntador a función, `(*Walk)`, que representa a la función de caminata que se va a usar en el programa (MCD sin muestreo inducido, MCV, etc.).

- Usamos apuntadores de funciones para generalizar las estadísticas. Todas operan sobre alguna caminata, recolectando cierta información sobre el ensamble de caminantes, por lo que se pueden abstraer y usarse sólo cuando es necesario.

- Para definir a un sistema nuevo, generalmente se requiere saber cuál será la forma de la función de prueba, su gradiente y su laplaciano, así como el potencial

$V(R)$ que actuará sobre las partículas. Ambas funciones están abstraídas en apuntadores, por lo que simplemente se debe escribir la nueva función y agregarla a los menues como se describe en la sección 7.5.3.

7.5.3. Agregando un nuevo sistema

El propósito de esta sección es documentar la manera de agregar un nuevo sistema (función de prueba/potencial) a los programas *moiss* y *gmoiss*.

- Editar el archivo `pl.c` y agregar a `trial_function_names` el nombre de la nueva función de prueba.

- En el archivo `gmoiss.h` existe una lista de enumeraciones de C (`enum`) que deben de mapear uno a uno a los nombres en `pl.c`. En este caso, se debe de editar la entrada correspondiente a `TrialFunctionType`.

- Dentro de `potential.c` se encuentran las funciones que calculan el potencial. Se debe de seguir el prototipo de las demás funciones para que se pueda acoplar con el apuntador a la función.

- Se recomienda crear un archivo nuevo `fprueba.c`. Éste puede crearse tomando el archivo `bto.c` como modelo. La función principal así como todas las funciones auxiliares deben guardarse en este archivo. Se debe seguir el prototipo de la función llamada por el apuntador (`Derivs *`).

- Por último, basta definir la lectura e inicialización de la función en el archivo `init.c`.

7.6. Comparación con otros programas

Esta es una revisión cualitativa de las distintas capacidades de los tres códigos de química cuántica por MCC disponibles libremente en la red:

- *Quantum MagiC* [43]: Desarrollado por el grupo del profesor W. A. Lester Jr. de la Universidad de California, Berkeley. Los autores del código son: B. L. Hammond, C. W. Greef, R. N. Barnett, M. M. Soto, L. Terray, P. J. Reynolds y W. A. Lester, Jr.

- *Talus* [83]: Desarrollado por el grupo del profesor W. M. C. Fowlkes del Imperial College en Londres. Los autores del código son: M. L. Stedman y W. C. Fowlkes.

Programa	Pseudopotenciales
QMagiC	⊞ Locales.
Talus	⊞ Locales. Planean extenderlo a no locales.
MOISS	⊠⊡ Aun no se implementan.

Programa	Exponenciales de Correlación
QMagiC	⊟ No implementadas
Talus	⊟ No implementadas
MOISS	⊠ Descrito en el capítulo 6

Programa	Sistemas educativos
QMagiC	⊟ No implementados
Talus	⊟ No implementados
MOISS	⊞ Descritos en este capítulo y el capítulo de resultados.

Programa	Estudio de Átomos
QMagiC	⊞
Talus	⊞
MOISS	⊞ Sólo pequeños por el tipo de funciones de prueba.

Programa	Estudio de Moléculas
QMagiC	⊞
Talus	⊠⊡ Planean implementarlo
MOISS	⊞ Sólo moléculas pequeñas (de menos de 6 electrones).

Programa	Lectura de archivos de salida de Hartree-Fock
QMagiC	⊞ Soporta el programa *HONDO*
Talus	⊞ No automatizado; formato propio
MOISS	⊠ Se planea implementar

Programa	Archivos de salida formateados.
QMagiC	⊞ Tiene el mejor formato de archivos de salida. Descriptivo y extendido
Talus	⊞ El formato de salida está planeado para poder ser analizado por alguna utilería de procesamiento posterior. La desventaja es que disminuye la legibilidad.
MOISS	⊞ Es un formato austero con la información mínima necesaria.

Programa	Interfase gráfica
QMagiC	⊠ No implementada
Talus	⊠ No implementada
MOISS	⊞ Cuenta con *gmoiss*

Programa	Formato de archivo de entrada
QMagiC	⊟ Tradicional. Orientado a líneas.
Talus	⊞ Bien formateado y flexible.
MOISS	⊞ Usa `libPropList`.

Programa	Pseudopotenciales
QMagiC	⊞ Locales.
Talus	⊞ Locales. Planean extenderlo a no locales.
MOISS	⊠⊡ Aun no se implementan.

Programa	Exponenciales de Correlación
QMagiC	⊟ No implementadas
Talus	⊟ No implementadas
MOISS	⊠ Descrito en el capítulo 6

Programa	Sistemas educativos
QMagiC	⊟ No implementados
Talus	⊟ No implementados
MOISS	⊞ Descritos en este capítulo y el capítulo de resultados.

Programa	Estudio de Átomos
QMagiC	⊞
Talus	⊞
MOISS	⊞ Sólo pequeños por el tipo de funciones de prueba.

Programa	Estudio de Moléculas
QMagiC	⊞
Talus	⊠⊡ Planean implementarlo
MOISS	⊞ Sólo moléculas pequeñas (de menos de 6 electrones).

Programa	Lectura de archivos de salida de Hartree-Fock
QMagiC	⊞ Soporta el programa *HONDO*
Talus	⊞ No automatizado; formato propio
MOISS	⊠ Se planea implementar

Programa	Archivos de salida formateados.
QMagiC	⊞ Tiene el mejor formato de archivos de salida. Descriptivo y extendido
Talus	⊞ El formato de salida está planeado para poder ser analizado por alguna utilería de procesamiento posterior. La desventaja es que disminuye la legibilidad.
MOISS	⊞ Es un formato austero con la información mínima necesaria.

Programa	Interfase gráfica
QMagiC	⊠ No implementada
Talus	⊠ No implementada
MOISS	⊞ Cuenta con *gmoiss*

Programa	Formato de archivo de entrada
QMagiC	⊟ Tradicional. Orientado a líneas.
Talus	⊞ Bien formateado y flexible.
MOISS	⊞ Usa `libPropList`.

8. Sistemás estudiados

8.1. Sistemas con potenciales simples

8.1.1. Oscilador armónico

Cuando quisimos probar nuestros metodos de MCC por primera vez, el primer sistema que estudiamos fue este, el potencial de este sistema está caracterizado por su frecuencia angular ω,

$$V_a(\mathrm{r}) = \frac{1}{2} m\omega^2 \mathrm{r}^2 \tag{8.1}$$

por lo tanto el operador Hamiltoniano de un electrón moviéndose en ese potencial de la forma:

$$\hat{H} = -\frac{\hbar^2}{2m}\frac{\partial^2}{\partial \mathrm{r}^2} + \frac{1}{2} m\omega^2 \mathrm{r}^2.$$

La ecuación de Schrödinger, $\hat{H}\psi = E\psi$ puede resolverse exactamente, dando en el caso unidimensional las siguientes soluciones:

$$E_n = \left(n + \frac{1}{2}\right)\hbar\omega$$

para el espectro de energías y con unas funciones de onda:

$$\psi_n(x) = N_n H_n(x) \exp\left\{-\frac{x^2}{2}\right\}$$

donde N_n es una constante de normalización y H_n son los polinomios de Hermite de grado n, de donde la función de onda del estado basal es:

$$\psi_0(x) = \pi^{\frac{1}{4}} \exp\left\{-\frac{x^2}{2}\right\}. \tag{8.2}$$

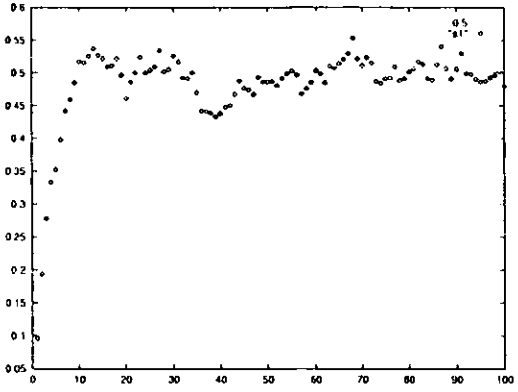


Figura 8.1.: Evolución de la energía en la simulación del oscilador harmónico.

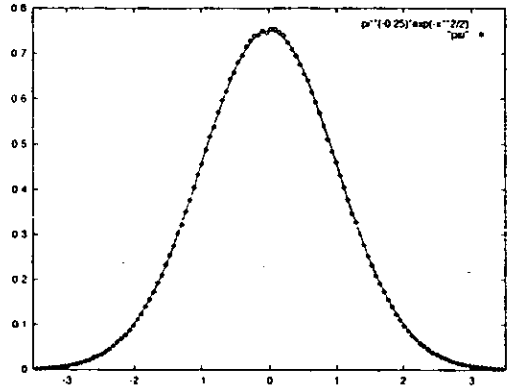


Figura 8.2.: Función de onda normalizada para el oscilador armónico, despues de una simulación con 50 bloques de 10 pasos cada uno.

8.1.2. Oscilador de Morse

El potencial de Morse es un modelo conveniente para estudiar la energía potencial de moléculas diatómicas, tiene la forma:

$$V_m(\mathbf{r}) = D(e^{-2\alpha r} - 2e^{-\alpha r}), \tag{8.3}$$

potencial para el cual la ecuación de Shrödinger también puede resolverse exactamente.

Sabemos que cuando $D = \frac{1}{2}$; la energía exacta del estado basal es: $E_0 = -\frac{1}{8}$, y la función de onda correspondiente:

$$\psi_0(x) = \sqrt{2}\exp\left\{-e^{-x} - \frac{x}{2}\right\}. \tag{8.4}$$

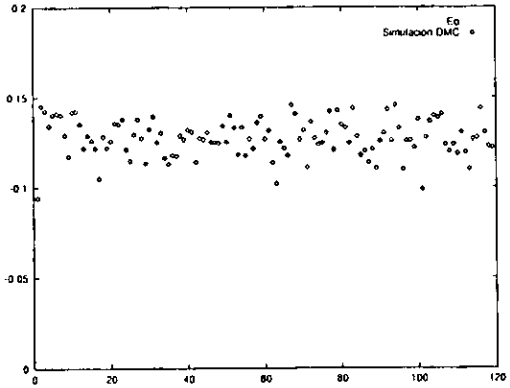


Figura 8.3.: Evolución de la energía en la simulación del oscilador de Morse.

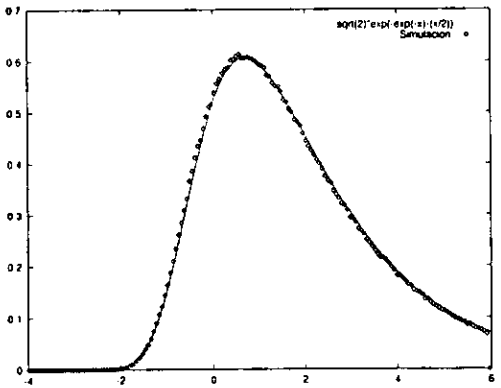


Figura 8.4.: Función de onda normalizada para el potencial de Morse, después de una simulación con 50 bloques de 10 pasos cada uno.

8.1.3. Pozo de potencial

Como ya revisamos previamente en los capítulos 2 y 3, la caja de potencial es un *clásico* en la didáctica de la cuántica.

En *MOISS* se pueden simular cajas de potencial con un potencial infinito en la frontera o con un potencial finito. En el caso del potencial finito, usamos la opción **Tunnel Potential**. Con estas condiciones, la función de onda no vale cero en el punto en el que comienza el potencial, sino que presenta un fenómeno cuántico conocido como *penetración* o *tuneleo*, en el que la probabilidad de encontrar al electrón en las regiones en las que el potencial es distinto de cero, existe.

En la figura 8.5 se puede observar el incremento en la penetración de la función de onda conforme el potencial V_0 va decreciendo.

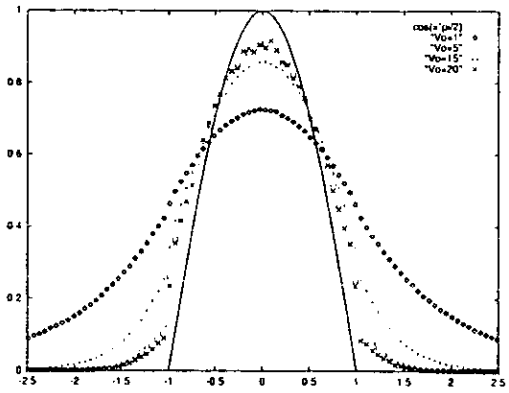


Figura 8.5.: Funciones de onda para un pozo de potencial finito obtenidas con MCD, a diferentes valores de V_0. La linea solida representa la función de onda para el pozo infinito de potencial.

8.2. Átomo de Hidrógeno

En el caso del átomo de hidrógeno, adoptaremos como unidad de longitud el radio de Bohr $a = \hbar^2/me^2 = 0.52917\mathring{A}$. Por lo tanto si escogemos $L = a$, entonces $T = \hbar^3/me^4$ y $\mathcal{E} = me^4/\hbar^2 \approx 27.21eV$. La energía del estado basal es bien conocida. $E_0 = -\frac{1}{2}$ (en unidades adimensionales), y la correspondiente función de onda es:

$$\psi_0(\mathbf{r}) = 2e^{-\mathbf{r}} \tag{8.5}$$

En este caso en particular, nuestro programa obtiene el valor exacto con una varianza de $3 \times 10^{-}7$ Hartrees, tanto en MCV como en MCD[1].

8.3. Átomo de Helio (He)

Las soluciones para varias funciones de prueba para el Helio se pueden obtener analíticamente. Sin embargo, este sistema es de interés, pues es el sistema atómico más chico en el que se pueden estudiar los efectos de la correlación.

La función de prueba que utilizamos tiene la siguiente forma:

$$\psi = \sum_j^L c_j(1 + \hat{\mathcal{P}}_{12}) \times \exp\{-\mathbf{k}_j \cdot \mathbf{d}\}(\alpha\beta - \beta\alpha), \tag{8.6}$$

donde el operador de permutación ($\hat{\mathcal{P}}$), impone la simetria correcta , y

$$(-\mathbf{k}_j \cdot \mathbf{d}) = k_{1A}r_{1A} + k_{2A}r_{2A} + k_{12}r_{12}.$$

[1]La simulación se realizó con mil caminantes y 50,000 pasos.

Los resultados obtenidos por varios grupos, junto con nuestros mejores resultados se presentan a continuación:

Método	Energía (Hartrees)	Ref.	E. Corr (%)
Hartree Fock	-2.861679993	[18]	0
IC	-2.90372435	[49]	99.998
Energía "Exacta"[a]	-2.903724377	[88][4]	100
MCFG	-2.90355(61)	[53]	100.(1)
MCD	-2.9038(10)	[53]	100.(1)
MCV	-2.903711(5)	Nosotros	99.97(1)

Tabla 8.1.: Energías reportadas para el átomo de helio.

[a]no relativista, masa infinita

Bertini *et al* [10] resolvieron analíticamente el átomo de Helio usando exponenciales correlacionadas. A continuación compararemos nuestros resultados en el átomo de Helio con distintas funciones de prueba.

En la siguiente tabla, compararemos los resultados obtenidos numéricamente con los resultados analíticos reportados en la literatura. +

Funciones base	Energía	σ	Cálculo analítico
2	-2.902859927519	0.000371	-2.903269243
3	-2.903236708232	0.000266	-2.903640610
4	-2.903506671792	0.000219	-2.903687268
5	-2.903483886272	0.000199	-2.903713090
6	-2.903542655743	0.000191	-2.903717300
7	-2.903542138698	0.000176	-2.903720078
8	-2.903536359400	0.000120	-2.903722779
9	-2.903565529807	0.000108	-2.903734087
10	-2.903584240362	0.000103	-2.903723798
11	-2.903577797520	0.000097	-2.903723965
12	-2.903609104633	0.000095	-2.903724020

Tabla 8.2.: Energía variacional del He utilizando funciones de prueba del tipo exponenciales de correlación.

Funciones base	Energía	σ
1	-2.898125494161	0.000071
2	-2.902956744654	0.000028
3	-2.903160343955	0.000024
4	-2.903639329624	0.000012
5	-2.903623971972	0.000013
6	-2.903613837176	0.000015
7	-2.903631857811	0.000013
8	-2.903532228051	0.000083
9	-2.903565747447	0.000086
10	-2.903711730177	0.000005

Tabla 8.3.: Energía variacional del He utilizando funciones de prueba del tipo exponenciales correlacionadas para la interacción electrón-núcleo y un término de Jastrow para la interacción electrón-electrón.

Como ejemplo, incluíremos el archivo de entrada *gmoiss-input*, para el átomo de Helio.

```
{
"CE Number of Terms" =9;
  "GMoiss Version" = 0.1;
   Simulation = "Molecular (BTF)";
   Molecule = Normal;
  "Molecule Charge"=0;

  Walkers = 1000;
  "Diffusion Constant" = 0.500000;
  "Time step" = 0.0100;
  "Maximum Branches" = 3;
  "dt convergence factor" = 1.000000;
  "Gaussian Spread Range" = 1.000000;
  "Random Number Generator Seed" = 13;
  "Graph: Divisions per Dimension" = 200;
  "Graph: Minimum" = "-3.000000";
  "Graph: Maximum" = 3.000000;
  "Electronic Repulsion Scale" = 1.000000;
  "Spin Parity Operator" = 0;

  "Electron-Electron TF" = "Jastrow";
  "Electron-Nuclei TF" = "Correlated Exponential";
  "CE Sym: Number of Terms" = 2;
  "CE Electron-Electron Symmetry" = No;
  "CE Electron-Nuclei Symmetry" = No;
  "CE Electron-Nuclei Parameters" = 2;
```

```
  "CE Prefactor Terms" = 0;
  "Parameters File Path" = "./psi.inp";
  "Atomic Coordinates" = ({x = 0.;y = 0.;z = 0.;"Atomic Number" = 2;});
  Potential ="Helium" ;
  "Trial Function" = "Corr. Exponential/Jastrow";
  "Initial Position" = "File";
  Dimensions = "3";
  Calculation = (
    {
      "Calculation Type" = "Variational Fokker-Planck";
      Blocks = 10;
      Steps = 50;
      Statistics = ("Local Energy", "Importance Sampling", "Average Energy");
    },
{
      "Calculation Type" = "Branching Diffusion";
      Blocks = 10;
      Steps = 50;
      Statistics = ("Local Energy", "Importance Sampling", "Average Energy");
    }
  );
  "CE Sym: Matrix" = ((1,0, 1), (1,1, 0));
  "CE Prefactors" = (("-1.000000"),
  );

"Reference Energy"="-2.903724";
"Cost Function"=1;
"Weight Averages"=0;
"Clip E "=3;
"Min Eloc"="-14.0";
"Max ELoc"=0;
"Min Weight"=.01;
"Max Weight"=10;
"Tolerance"=0;
"OPt Scale"=0.01;
"Max opt iter"=10;

}
```

8.4. Molécula de H_2: Estado basal y primer estado excitado.

La molécula de H_2, esta formada por 2 protones a una distancia de equilibrio $R_{NN} = 1.4011\mathring{A}$ y por 2 electrones.

8.4.1. Estado basal.

El estado basal de la molécula de hidrógeno tiene una multiplicidad de uno. Los dos electrones se encuentran apareados en un enlace σ.

Los resultados obtenidos por varios grupos en MCC se citan a continuación, siendo éstos comparados con nuestros mejores resultados y los resultados de nuestras simulaciones.

Método	Energía (Hartrees)	Ref.	E. Corr (%)
Energía Exacta[a]	-1.17447	[53]	100
MCD	-1.17451(10)	[53]	100.(1)
MCV	-1.17441(3)	Nosotros	99.99(4)

[a]no relativista

Tabla 8.4.: Energías reportadas para la molécula de Hidrógeno.

Simulaciones con MOISS.

Se realizo una simulación de MCV de Fokker-Planck con funciones de prueba de exponenciales de correlación de 1-16 términos en *laguna.fmedic.unam.mx* en la que se obtuvieron los siguientes resultados crudos:

Funciones Base	Energía	σ
1	-1.168702401356	0.000605
2	-1.170110511191	0.000617
3	-1.174183607958	0.000113
4	-1.174189478283	0.000087
5	-1.174211391989	0.000076
6	-1.174169594246	0.000076
7	-1.174173427371	0.000080
8	-1.174186418002	0.000079
9	-1.174179258794	0.000080
10	-1.174181856276	0.000081
11	-1.174176733897	0.000081
12	-1.174299299097	0.000275
13	-1.174402272991	0.000108
14	-1.174394185288	0.000130
15	-1.174417838072	0.000116

Tabla 8.5.: Energía variacional del H_2 utilizando funciones de prueba del tipo exponenciales correlacionadas.

Los últimos cuatro términos fueron corridos la mitad de bloques que los primeros once.

Superficie de Energía Potencial

Se realizó la superficie de energía potencial para este sistema. Los resultados se encuentran en la tabla a continuación:

Distancia	Energía	σ
0.9	-1.083205004892	0.000154
1.0	-1.124141141549	0.000127
1.1	-1.149730233835	0.000111
1.2	-1.164625897000	0.000092
1.3	-1.172003025565	0.000084
1.4011	-1.174189478283	0.000087
1.5	-1.172616932111	0.000066
1.6	-1.168375508452	0.000056
1.7	-1.162298138411	0.000053
1.8	-1.154921258292	0.000071
1.9	-1.146705615295	0.000048
2.0	-1.137959816433	0.000063
2.1	-1.128961862867	0.000070
2.2	-1.119906535434	0.000076
2.3	-1.110993877971	0.000073
2.4	-1.102268835648	0.000068
2.5	-1.093774500039	0.000057
2.6	-1.085641587156	0.000064
2.7	-1.077937738607	0.000082
2.8	-1.070635336573	0.000069
2.9	-1.063749588942	0.000065
3.0	-1.057321999683	0.000065

Tabla 8.6.: Superficie de energía potencial para la molécula de hidrógeno en su estado basal.

8.4.2. Primer estado excitado ($H_2(^3\Sigma_u^+)$)

El primer estado excitado de la molécula de Hidrógeno presenta una multiplicidad de tres. Uno de los electrones pasa al orbital de antienlace. El valor exacto para esta configuración es de -0.7831 y fue determinado por Kolos y Roothan [3].

Términos	Energía	σ
1	-0.649271079013	0.001727
2	-0.71288224439	0.000064
3	-0.712315867732	0.000073
4	-0.71292625760	0.000071
5	-0.713063775463	0.000059
6	-0.713056652118	0.000064
7	-0.713226897736	0.000069
8	-0.713510978928	0.000031
9	-0.713618795879	0.000032
10	-0.713608431436	0.000032
11	-0.713620715625	0.000031
12	-0.713619194992	0.000032
13	-0.713615536034	0.000032

Tabla 8.7.: Molécula de H_2 con multiplicidad 3. Los promedios fueron tomados de 100 bloques cada uno con 200 pasos.

La energía obtenida no puede compararse con el mejor resultado, pues el cálculo no fue realizado a la distancia de equilibrio del sistema. La distancia usada es la misma que el caso anterior.

Igual que en el caso anterior, la energía no se mejora mucho en los términos 9-13.

8.4.3. Superficie de Energía Potencial

Se realizó una simulación de MCV para distintas distancias de separación d_{HH} para encontrar la distancia asociada a la mínima energía del sistema H_2. Se usaron exponenciales de correlación con cuatro funciones base lineales.

Los resultados se encuentran en la siguiente tabla:

Distancia	Energía	σ
1.1	-0.645754925120	0.000061
1.2	-0.676113437592	0.000060
1.3	-0.697611026279	0.000060
1.4	-0.712873136659	0.000058
1.5	-0.722984455209	0.000062
1.6	-0.729964506900	0.000071
1.7	-0.734168317929	0.000060
1.8	-0.736093891901	0.000063
1.85	-0.736236286863	0.000067
1.87	-0.736242119160	0.000065
1.89	-0.736305231480	0.000066
1.9	-0.736379641600	0.000052
1.91	-0.736146835981	0.000063
2.0	-0.735254824581	0.000063
2.1	-0.733338638642	0.000056
2.2	-0.730554664196	0.000060
2.3	-0.727698814649	0.000052
2.4	-0.723954593025	0.000052
2.5	-0.719963960328	0.000054
2.6	-0.715884874696	0.000054
2.7	-0.711530511965	0.000055
2.8	-0.707283885366	0.000052
2.9	-0.702781602496	0.000055
3.0	-0.698314024797	0.000053
3.1	-0.693985150405	0.000056
3.3	-0.685013810897	0.000073
3.5	-0.676946071287	0.000078

Tabla 8.8.: Superficie de energía potencial para la molécula de hidrógeno en su primer estado excitado ($H_2(^3\Sigma_u^+)$).

La relación $E = f(r)$ puede verse en la figura 8.6

Para realizar una superficie de energía potencial, se requiere escribir un *script* en algún lenguaje como Python; un ejemplo se presenta a continuación:

```
#! /usr/bin/env python
# script to run various DMC calculations at different time steps
# it requires that the first directory to be named
# nombre-DMC-timestep, eg: Li-DMC-0.001
```

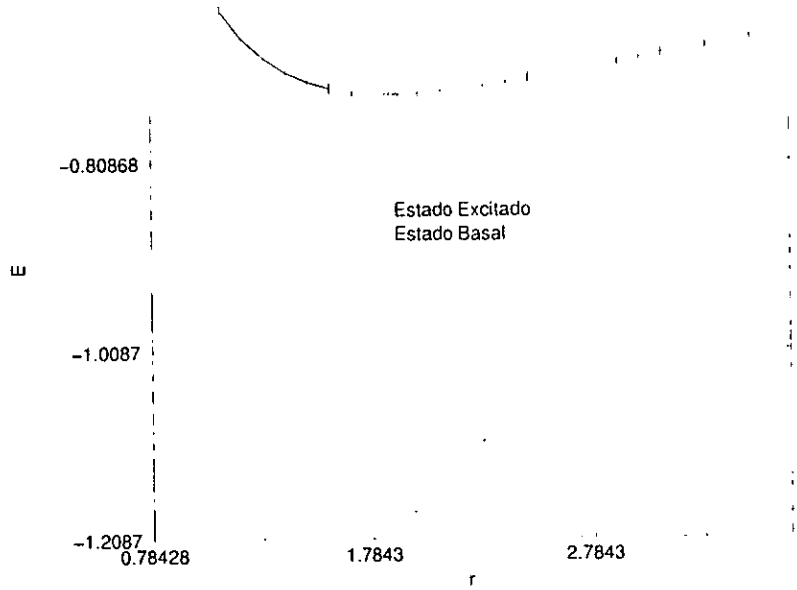


Figura 8.6.: Superficie de energía potencial para la molécula de hidrógeno en su estado basal, y en su primer estado excitado ($H_2(^3\Sigma_u^+)$).

```
# It should have:
# a gmoiss-input with "Time Step" = 0.001 in the THIRD line
# a symbolic link to the moiss binary in the directory
# optionally link restart to walk (not used here but it's nice to have it)


from math import *
from string import *
from whrandom import *
from os import system

nombre = "H2m3"
distances = [1.0, 1.1, 1.2, 1.3, 1.4, 1.5, 1.6, 1.7, 1.8]

for dt in timesteps:
    print "Corriendo la primer distancia (dist= "+str(dt) +")"
    stringy = "cd " + nombre + "-dist-" + str(dt) +"; ./moiss > output ; cd .."
    #stringy = "cd " +nombre+"-"+ str(numero)+"; ls > caca"
    system(stringy)
    string = "cp -af " + nombre + "-dist-" + str(dt) + " "
+ nombre +"-dist-" +str(timesteps[timesteps.index(dt)+1])
    system(string)
    string4 = nombre + "-dist-" + str(timesteps[timesteps.index(dt)+1])
```

```
+"/gmoiss-input"
    print string4
    archivogmoiss = open(string4,'r')
    listagmoiss = archivogmoiss.readlines();
    string5 = "{x = 0.;y = 0.;z = " + str(timesteps[timesteps.index(dt)+1])
+";\"Atomic Number\" = 1;});\n"
    #fifth line!!
listagmoiss[4] = string5
    archivogmoiss.close()
    archivogmoiss = open(string4, 'w')
    for linea in listagmoiss:
archivogmoiss.write(linea)
    archivogmoiss.close()
    print "acabe una vuelta"
stringfinal = "echo ya acabe... " + nombre + "
> mail aspuru@eros.pquim.unam.mx"
stringfinal2 = "echo ya acabe... " + nombre + "
> mail rulox@eros.pquim.unam.mx"
system(stringfinal)
system(stringfinal2)
```

8.5. Ion He_2^+

El ión dimolecular He_2^+ fue estudiado en su estado basal $^2\Sigma_u^+$. La funciónde prueba utilizada fue:

$$\psi = \sum_j^L c_j \hat{A}[(1+\hat{i}) \times \exp\{-\mathrm{k}_j \cdot \mathrm{d}\}(\alpha\alpha\beta - \alpha\beta\alpha)], \tag{8.7}$$

La distancia de mínima energía es de 2.0625 bohr.

Los resultados obtenidos fueron:

Términos	Energía	σ
1	-4.860355236787	0.000727
2	-4.963378207128	0.000511
3	-4.975146062467	0.000393
4	-4.987723686693	0.000266
5	-4.990495382097	0.000186
6	-4.991624114291	0.000168
7	-4.991758116312	0.000167
8	-4.992238092417	0.000127
9	-4.992508190083	0.000140

Tabla 8.9.: He_2^+ con exponenciales de correlación.

8.6. Hidruro de Litio (LiH)

Este sistema de cuatro electrones es el más grande que estudiamos en esta tesis. Sólo se ha realizado un cálculo más grande de exponenciales correlacionadas, el del átomo de Boro, realizado por Bertini *et. al.* [10].

Los resultados reportados, comparados con nuestra mejor energía se presentan a continuación:

Método	Energía (Hartrees)	Ref.	E. Corr (%)
Hartree Fock	-7.9873	[67]	0
GVB[a]	-8.0037	[67]	19.3
MCSCF	-8.0213	[67]	40.1
IC	-8.0606	[67]	87.9
Hilleraas-CI	-8.0630	[67]	91.6
CEPA	-8.0660	[67]	95.2
Experimental[b]	-8.0699	[67]	100
MCFG[c]	-8.07(2)	[67]	102.4
MCFG[d]	-8.07(2)	[67]	102.4
MCD	-8.0700(4)	[53]	99.5(5)
MCV	-8.0127(5)[e]	Nuestro	30.75(5)
MCV	-8.0597(2)[f]	Nuestro	84.53(7)

[a]General Valence Bond.
[b]no relativista
[c]Monte Carlo Cuántico de funciones de Green con función de prueba HF.
[d]Monte Carlo Cuántico de funciones de Green con función de prueba de tipo General Valence Bond
[e]Sólo incluyendo la interacción electrón-núcleo.
[f]2 funciones base incluyendo las interacciones electrón-electrón.

Tabla 8.10.: Energías reportadas para el hidruro de litio.

8.6.1. Nuestros Resultados

Exponenciales de correlación sin términos electrón electrón

Se corrió una simulación de MCV usando exponenciales de correlación considerando sólo las interacciones electrón-núcleo, ignorando las interacciones electrón-electrón. El número de parámetros a minimizar por función base es de 11, uno lineal, ocho distancias electrón-núcleo y dos parámetros en el prefactor de simetría.

El motivo de correr esta simulación fue el de comaparar los resultados con las funciones que contengan términos de interacción interelectrónica

Funciones Base	Energía	σ
1	-7.652713421179	0.002399
2	-7.961500377681	0.001062
3	-7.980560224483	0.000748
4	-7.984306451328	0.000752
5	-8.012773633630	0.000577

Tabla 8.11.: Cálculos variacionales para la molécula de LiH con exponenciales de correlación sin incluir las interacciones electrón-electrón.

Exponenciales de correlación con términos electrón-electrón.

Esta simulación tiene un total de 17 parámetros variacionales por término. Se incluyen seis términos no lineales de interacción electrón-electrón.

Funciones Base	Energía	σ
1	-7.977770536095	0.000671
2	-8.037807954614	0.000525
3	-8.054264394083	0.000726
4	-8.059750843488	0.000264
5	-8.059577339481	0.000241

Tabla 8.12.: Cálculos variacionales para la molécula de LiH con exponenciales de correlación incluyendo las interacciones electrón-electrón.

8.7. Computadoras usadas

Las computadoras que fueron usadas para realizar las simulaciones fueron sistemas Linux i386 ubicados en distintas instalaciones de la Universidad Nacional Autónoma de México y en casas particulares.

Los cálculos se realizaron en aproximadamente 12-48 horas de tiempo-máquina por equipo. Las computadoras usadas se listan en la tabla 8.13.

Equipo	Lugar	Procesador	Bmips[a]	Carga
eros.pquim.unam.mx	FQ[b]	Pentium II	231.01	E[c]
litio.pquim.unam.mx	FQ	AMD K6	120.04	L[d]
hunabku.pquim.unam.mx	FQ	Pentium II	231.01	L
thanatos.pquim.unam.mx	FQ	Pentium	398.95	L
metropolis.nuclecu.unam.mx	ICN[e]	Dual P-Pro	600.66	M[f]
laguna.fmedic.unam.mx	FM[g]	Pentium II	332.60	M
uxdea3.iimas.unam.mx	IIMAS[h]	Pentium Pro	333.41	M
uxdea4.iimas.unam.mx	IIMAS	Dual P-II	696.32	M
cic2.iimas.unam.mx	IIMAS	Pentium	298.19	L
agave.coseac.unam.mx	COSEAC[i]	Pentium Pro	266.24	L
mezcal.coseac.unam.mx	COSEAC	Pentium II	447.28	L
tuatara	Particular	Cyrix 686	166.30	C[j]

[a]Bogomips: Medida estimada del poder de cómputo de una computadora con el sitsema Linux; ésta no es absoluta y puede variar considerablemente de arquitectura a arquitectura.
[b]Facultad de Química, División de Estudios de Posgrado.
[c]Elevada: El sistema funciona como servidor de más de 800 cuentas de correo y páginas de WWW del departamento de Física y Química Teórica.
[d]Ligera: El servidor administra del orden de 100 cuentas de correo.
[e]Instituto de Ciencias Nucleares
[f]Mediana. Se usa para cálculos del ICN.
[g]Departamento de Bioquímica, Facultad de Medicina
[h]Instituto de Investigaciones en Matemáticas Aplicadas y Sistemas.
[i]Coordinación de Servicios Académicos
[j]Casera

Tabla 8.13.: Lista de computadoras con el sistema operativo *Linux* usadas para nuestros experimentos.

La precisión en los cálculos fue de 32 bits en todos los casos.

8.8. Análsis del programa

Con el propósito de optimizar la velocidad de ejecución de *moiss*, se realizó un análisis de una simulación de la molécula de Helio, con funciones de prueba tipo Jastrow con 9 parámetros lineales. El programa fue compilado con la opción -pg de gcc, y posteriormente analizado con el programa gprof[2].

El programa pasó el 83.56% del tiempo evaluando la función de prueba. Del resto, se tardo un 4.64% en el algoritmo mismo de caminata, y un 2.41% obteniendo números aleatorios gaussianos. Los datos anteriores revelan que si se desea mejorar la ejecución del programa, se debe de pensar en formas más eficientes de evaluación de la función de prueba.

El 31.75% del tiempo del programa se consumió en calcular cuál es el argumento del exponente del término de Jastrow. Ésto implica que encontrar nuevas formas para evaluar dicho argumento puede ser fundamental para lograr una mejoría en la velocidad.

[2]Este programa forma parte de las utilerías del compilador gcc.

9. Conclusiones

Al realizar nuestros estudios con funciones de prueba de tipo exponenciales de correlación (EC), podemos llegar a las siguientes conclusiones.

9.1. Recuperación de la energía de correlación

Las funciones de EC recuperan un buen porcentaje de la energía de correlación con pocos términos. La convergencia a mejores valores es más lenta, por lo que se sugiere optimizar unos cuantos términos y recuperar la energía restante con MCD. En el caso de la molécula de Hidrógeno, no se observa un incremento sustancial en la energía recuperada. Sólo cuando se usan 13 o 15 términos se comienza a ver una fracción del 95-100% de energía recuperada.

En sistemas más grandes, como por ejemplo el hidruro de litio, encontramos que la energía de correlación recuperada por el método de MCV es del orden del 85% para un uso de CPU razonable.

En el caso del hidruro de litio, realizamos cálculos que no incluían las interacciones electrón-electrón. Comparando los datos anteriores con las simulaciones que incluyen dicha correlación, vemos que el porcentaje de energía de correlación recuperada aumenta en un 50-60%.

La realización de cálculos con los términos electrón núcleo, puede servir como una primera base para una subsecuente optimización de los términos electrón-electrón.

9.1.1. Comparación de las distintas funciones de prueba: Jastrow vs. EC vs. analítica

En el caso del átomo de Helio, encontramos los mismos resultados que Bertini *et al.* [10] Las funciones de tipo Jastrow recuperan mayor energía con menor número de términos. Además, al comparar nuestros resultados numéricos ocn los resultados analíticos, nos damos cuenta que existen deficiencias en el motor de optimización. Estos datos podrían ser un buen punto de partida para hacer pruebas con mejores algoritmos de optimización. El algoritmo que llegue a resultados más cercanos a los analíticos optimizando las funciones de prueba para el Helio sería el mejor.

9.2. Requerimientos computacionales

Como ya habíamos visto antes, el mayor tiempo de cómputo se gasta evaluando la función de prueba. El costo de evaluar nuestras funciones de prueba crece como $N_\alpha!N_\beta!$ donde N_α y N_β son el número de electrones en distintos espines.

En el mismo tiempo en el que se obtienen resultados para doce funciones base en la molécula de hidrógeno (que contiene dos electrones y dos núcleos), se obtienen resultados para cuatro términos de la molécula de hidruro de litio sin tomar en cuenta las interacciones electrón-electrón (que hacen el cálculo por lo menos la mitad de lento).

Las funciones de prueba de exponenciales correlacionadas no son buenas candidatas para resolver sistemas grandes. El uso de pseudopotenciales y determinantes que provengan de métodos de campo medio como Hartree-Fock parece ser la opción más adecuada para realizar cálculos en sistemas grandes.

9.2.1. Optimización de la eficiencia

Existen varias optimizaciones que podríamos hacer para optimizar la eficiencia del cálculo. *MOISS* se tarda la mayor parte del tiempo calculando los términos de las permutaciones y las exponenciales asociadas a las funciones de prueba de exponenciales de correlación.

Una posible alternativa es seguir técnicas parecidas a las usadas en *atmol* [17], en las que se precalculan varias opciones en las que se gasta mucho tiempo de CPU, los resultados se guardan en arreglos en la memoria, que son subsecuentemente interpolados.

Así como precalcular lo más eficientemente posible las permutaciones electrónicas. Creemos que una optimización de este tipo nos permitiría tener una ganancia en la velocidad por lo menos de un orden de magnitud.

9.3. Acerca de la necesidad de paralelizar los cálculos.

Paralelizar un cálculo de Monte Carlo es relativamente fácil de implementar. Se puede usar el modelo de *maestro-esclavo*, en el que una computadora central envía los cálculos a los nodos, y éstos devuelven la distribución de equilibrio de los caminantes y los promedios de las propiedades que midieron. Basta con combinar los resultados de los nodos para obtener un resultado global.

El problema que se debería de abordar en el futuro es el de paralelizar la optimización de los parámetros.

Bertini *et al* [10] proponen minimizar un parámetro de las funciones de prueba a la vez:

La evaluación de la función de onda es la parte más costosa del proceso de optimización. Una ganancia en la velocidad cuando se usa un conjunto de funciones de base grande, es notar el hecho trivial de que durante la optimización de un sólo término, todos los demás no cambian, por lo que precalcular sus contribuciones a la energía local hace posible un gran ahorro de tiempo de CPU. Después de la optimización de un término dado, su contribución es guardada y el siguiente término se optimiza de la misma manera. Este proceso debe repetirse hasta que la disminución en la varianza sea despreciable. Esto usualmente toma unas cuantas "barridas" (3-10) sobre todos los términos.

Lo que tal vez sea una buena implementación paralela de la idea anterior es precalcular la contribución de cada término, y posteriormente hacer una pila de términos. Los nodos optimizarían la varianza de un término con los *mejores términos* disponibles en ese momento. Al concluir la optimización de ese término, el siguiente nodo que tome un trabajo de la pila, tendrá el nuevo valor optimizado. (ver la figura 9.1).

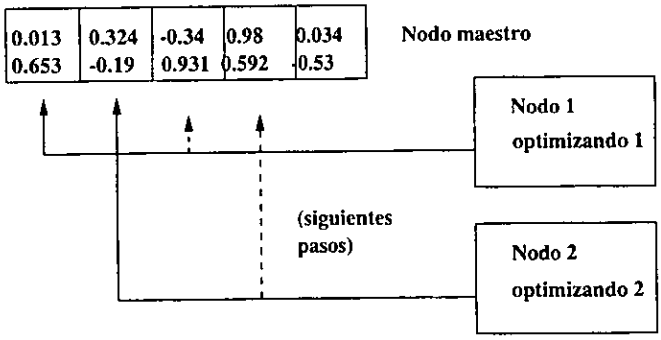


Figura 9.1.: Algoritmo de paralelización propuesto para la optimización variacional de parámetros no lineales por el método de Powell.

9.4. Las prácticas de la buena programación científica

Recientemente, P. Dubois [28] publicó un pequeño recetario de diez buenas prácticas para la programación científica. El programa MOISS fue programado pensando en algunas de ellas, pero falta trabajo para que éste cumpla con las características propuestas por el autor.

Entre ellas están:

Organización para el cambio Se deben usar técnicas de portabilidad, encapsulación a las bibliotecas de paralelización. Creemos que en este aspecto fuimos cuidadosos, y que basta continuar programando de esa manera.

Escribir un script en vez de un programa compilado El uso de programas interpretados para la *dirección*[1] del programa es una buena práctica. Sólo se deben compilar las rutinas que gastan la mayoría del tiempo. El resto puede ser implementado en algún lenguaje como Perl, Tcl o Python. En nuestra experiencia, usamos Python para generar las corridas de múltiples parámetros, que requerían:

- Cambiarse a un directorio de una simulación.
- Editar el archivo de entrada *gmoiss-input* para cambiar el número de términos a simular.
- Generar nuevos parámetros aleatorios para el siguiente término y anexarlos al archivo *psi.inp*.
- correr *moiss* registrando su salida.
- Hacer esto para un número n de parámetros.

Hacer ese tipo de tareas es unas cuantas líneas en `Python` y varias pantallas de código en `C` o `Fortran`. En un futuro queremos que todas las funciones que realizamos en el archivo `moiss.c` de MOISS sean realizadas en scripts de `Python`.

Usar un sistema moderno de manejo de código fuente Sugieren el uso de algún sistema de manejo de código fuente. En nuestro caso estamos usando `CVS` (Concurrent Versioning System), una utilería GNU muy poderosa para este propósito.

Usar *diseño por contrato* El uso de precondiciones y postcondiciones, así como aserciones es una buena manera de evitar errores en la programación científica. MOISS usa muy poco estas características, que inclusive están soportadas en `glib`. En nuestras próximas versiones insertaremos aserciones a lo largo del código para hacerlo más robusto.

Usar Fortran 95, no Fortran 77 No aplicable a nuestro código.

Dirigir el cálculo. El uso de un lenguaje interpretado para llamar código número compilado. Como anteriormente mencionamos, está en nuestros planes *dirigir* a MOISS usando `Python`.

Usar múltiples lenguajes (inteligentemente) No está en nuestros planes, a menos que sea necesario.

Usar C++ en vez de C No estamos de acuerdo con esta política. Si se desea ver una discusión al respecto, basta leer el artículo de Ian Joyner *C++? A critique of C++ and Programming Trends of the 1990's* [51].

[1] Conocido en inglés como *steering*.

9.5. Optimización de los nodos

Otra idea que podría ser desarrollada en un futuro próximo es el estudio de la optimización de los nodos. Así como la gente ha intentado optimizar la varianza de la energía local, o la energía local misma, nosotros creemos que podría ser posible intentar una optimización de la superficie nodal *n-dimensional*, si la parametrizamos por medio de *splines* o algún tipo de función de parametrización. Los errores más graves en los nodos se encuentran en las funciones más penetrantes, pues éstas son las que tienen mayor energía asociada. Tal vez intentar optimizar la superficie nodal de dichas funciones sea una buena idea.

9.6. Planes futuros para MOISS

El desarrollo del programa MOISS no terminará con la culminación de esta tesis. Esta planeada la implementación de varias características:

Paralelización En los planes se encuentra la integración de llamadas a PVM para paralelizar el código de MOISS. La paralelización se encuentra en una fase muy primitiva en los momentos de escribir esta tesis.

Integración con Python La versión 1.0 debe tener la capacidad de ser *dirigida* desde `Python` de manera que podamos agregar opciones rápidamente.

Graficación en 3D Planeamos integrar nuestro programa con `Vis5D`, una utilería GNU para graficar todo tipo de funciones.

Funciones de prueba de Slater-Jastrow y Pseudopotenciales La implementación de un motor de evaluación de funciones de prueba de tipo Slater-Jastrow, hará posible el estudio de sistemas más grandes.

Evaluación de propiedades Tenemos que implementar el código necesario para calcular fuerzas *a-la* Car y Parinello, polarizabilidades, etc.

A. Antecedentes

A.1. Nociones matemáticas

En las 2 primeras secciones de este apéndice haremos un repaso de algunos de los fundamentos matemáticos con los que se tiene que estar familiarizado para comprender el tratamiento integral de la ecuación de Schrödinger.

A.1.1. Espacios vectoriales

Primero definamos a nuestro objeto de estudio. En general trabajaremos con ciertos objetos matemáticos abstractos, repersentados por $|\rangle$. Algunos ejemplos de estos objetos son:

$$|\alpha\rangle\ , \quad |-2.3\rangle, \quad |\{3,5,0,2\}\rangle, \quad |\mathbf{R}\rangle, \cdot \textit{ etc.}$$

Ahora estudiemos las reglas del juego que son válidas para estos objetos, es decir su *composición* o álgebra. Definiremos dos operaciones básicas, a la primera le llamaremos adición. En ésta obtenemos un nuevo conjunto $|c\rangle$ a partir de otros dos: $|a\rangle$, y $|b\rangle$:

$$|c\rangle = |a\rangle + |b\rangle.$$

A la segunda le llamamos multiplicación por un escalar y nos sirve para obtener un nuevo conjunto $|c\rangle$ a partir de un número complejo y otro conjunto $|b\rangle$:

$$|c\rangle = \alpha\ |b\rangle.$$

Definición A.1. *Si los elementos $|e\rangle$ que conforman el espacio S cumplen con las siguientes condiciones se dice que forman un espacio vectorial y son denominados* vectores*:*

1. *Si $|a\rangle, |b\rangle \in S$, entonces $(|a\rangle + |b\rangle) \in S$.*

2. *Si $|a\rangle \in S$ y α es un número complejo, entonces $\alpha|a\rangle \in S$.*

3. *Existe un elemento* $|0\rangle \in S$ *tal que para cualquier* $|a\rangle \in S$ *se cumple* $|a\rangle + |0\rangle = |a\rangle$

4. *Para cualquier* $|a\rangle \in S$ *existe un elemento* $|a'\rangle \in S$ *tal que* $|a\rangle + |a'\rangle = |0\rangle$.

A continuación definiremos otra regla, en la cual asociamos a cualquier par de vectores un numero complejo, decimos que $\langle a|b\rangle$ es el *producto escalar* de $\langle a|$ con $|b\rangle$ [1].

Propiedades A.1. *Las propiedades del producto escalar son:*

1. $\langle a|b\rangle = \overline{\langle b|a\rangle}$ [2].

2. *Si* $|d\rangle = \alpha|a\rangle + \beta|b\rangle$ *entonces* $\langle c|d\rangle = \alpha\langle c|a\rangle + \beta\langle c|b\rangle$.

3. $\langle a|a\rangle \geq 0$, *el signo de igualdad sólo aparece cuando* $|a\rangle = 0$.

Basándonos en las propiedades del producto escalar podemos obtener la siguiente desigualdad:

$$|\langle a|b\rangle|^2 \leq \langle a|a\rangle\langle b|b\rangle \tag{A.1}$$

que es conocida como la desigualdad de Cauchy-Schwarz. La existencia de un producto escalar definido implica que un espacio vectorial es un *espacio métrico*, lo que nos garantiza que podemos proponer por lo menos una expresión para la distancia ente dos puntos de dicho espacio:

$$\rho(|a\rangle, |b\rangle) = \sqrt{[\langle a| - \langle b|] \cdot [|a\rangle - |b\rangle]}. \tag{A.2}$$

Se dice que un espacio es completo cuando podemos acercarnos a cualquier valor a siguiendo una secuencia arbitraria de puntos a_i:

$$\lim_{i\to\infty} \rho(|a\rangle, |a_i\rangle) = 0 \tag{A.3}$$

Dos vectores a y b son ortogonales si:

$$\langle a|b\rangle = \langle b|a\rangle = 0 \tag{A.4}$$

Un ejemplo de espacio vectorial es el conjunto de todas las funciones continuas definidas es un intervalo [a,b], considerando a todo el conjunto de valores de una función $f(x)$ como los elementos de un vector $|f)$ que pertenece a un espacio vectorial $\mathcal{F}$. El producto escalar en el espacio $\mathcal{F}$ se define como:

$$\langle f|g\rangle = \int_a^b \overline{f(x)} g(x) w(x) dx, \tag{A.5}$$

donde $w(x)$ es una función real positiva llamada *función de densidad* o *función de peso*.

[1] Los vectores $(V|$ pertenecen a un espacio vectorial isomorfo al de los vectores $|V^*)$, cuando existe una correspondencia uno a uno entre ellos, tenemos dos espacios duales. En la notación de Dirac a los vectores $(V|$ se les denomina *bra* y a los $|V^*)$ *ket*.

[2] El conjugado de un número complejo $z = \alpha + i\beta$ se representa por $\overline{z}$ que por definición es $\overline{z} \equiv \alpha - i\beta$.

A.1.2. Matrices

En esta sección recordaremos los conceptos básicos relacionados con las matrices[3].

Matriz: Una matriz (A) es un conjunto de $m \times n$ números a_{ij}, ordenados en un arreglo rectangular de m renglones y n columnas:

$$\mathrm{A} = (a_{ij}) = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{pmatrix} \tag{A.6}$$

Matriz identidad: La matriz identidad es una matriz cuadrada de tamaño $n \times n$ cuyos componentes de la diagonal principal son uno, mientras que los demás componentes son cero:

$$\mathrm{I}_n = (b_{ij}) \text{ donde } \begin{cases} 1 & \text{si } i = j \\ 0 & \text{si } i \neq j \end{cases} \tag{A.7}$$

Suma de matrices: Sean A y B dos matrices de $m \times n$. La suma de A y B es la matriz $\mathrm{C} = \mathrm{A} + \mathrm{B}$ dada por:

$$\mathrm{C} = \mathrm{A} + \mathrm{B} = (a_{ij} + b_{ij}) \tag{A.8}$$

Multiplicación de una matriz por una escalar: Si α es un escalar, la matriz $\alpha\mathrm{A}$, se obtiene multiplicando cada componente de A por α:

$$\alpha\mathrm{A} = (\alpha a_{ij}) = \begin{pmatrix} \alpha a_{11} & \alpha a_{12} & \cdots & \alpha a_{1n} \\ \alpha a_{21} & \alpha a_{22} & \cdots & \alpha a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ \alpha a_{m1} & \alpha a_{m2} & \cdots & \alpha a_{mn} \end{pmatrix} \tag{A.9}$$

Multiplicación de 2 matrices: Sea A una matriz de tamaño $m \times n$ cuyo i-ésimo renglón se denota por a_i. Sea b una matriz de tamaño $n \times p$ cuya i-ésima columna se denota por b_j. El producto AB es una matriz $\mathrm{C} = (c_{ij})$ de $m \times p$, donde cada una de las entradas se construye usando la siguiente regla:

$$c_{ij} = \mathrm{a}_i \cdot \mathrm{b}_j = a_{i1}b_{1j} + a_{i2}b_{2j} + \cdots + a_{in}b_{nj} \tag{A.10}$$

[3]Este termino lo introdujo el matemático inglés James Joseph Sylvester en 1850, refiriéndose a las "matrices" como "madres de los determinantes".

Inversa de una matriz: Sean A y B matrices de $n \times n$, si se cumple que:

$$AB = BA = I$$

entonces B se conoce como *la matriz inversa* de A, pudiéndose escribir como A^{-1}.

Mariz transpuesta: Sea $A = (a_{i,j})$ una matriz de tamaño $m \times n$, la matriz transpuesta de A, con símbolo A^t, es la matriz de tamaño $n \times n$, obtenida intercambiando cada uno de los renglones de A a columnas y cada una de las columnas a renglones:

$$A^t = (a_{j,i}) \tag{A.11}$$

Determinanete de una matriz El determinande de una matriz $n \times n$ es un número, que es obtenido de la siguiente forma:

$$\det(A) = |A| = \sum i = 1^{n!}(-1)^{p_i}\hat{\mathcal{P}}_c(a_{11}a_{22}\cdots a_{nn}) \tag{A.12}$$

donde $\hat{\mathcal{P}}_c$ es un operador que permuta [4] los indices de las columnas, el signo de la permutación es $(-1)^{p_i}$, donde p_i es el numero de *intercambios* necesarios para llegar a dicha permutación.

A.1.3. La función delta de Dirac y su familia de funciones

Además de conocer las propiedades de los espacionos vectoriales, tenemos que estar familiarizados con la función delta de Dirac y su familia de funciones, pues éstas son indispensables para la teoría de ecuaciones integrales y funciones de Green.

Ésta es una somera revisión de las propiedades de estas funciones, si el lector no comprende alguna, es conveniente que se dirija a la bibliografía correspondiente [9].

Delta de Dirac

A continuación se presenta un resumen de las propiedades de la delta de Dirac:

Definición A.2. *Si $f(x)$ es bien portada, entonces:*

$$\int_{x_1}^{x_2} \delta(x) = \begin{cases} f(0) & si\ x_1 < 0 < x_2 \\ 0 & si\ el\ origen\ se\ encuentra\ fuera\ del\ intervalo\ de\ integración \end{cases}$$

[4] Cada permitación cambia el orden de un par de elementos de una secuencia de numeros, el numero de permutaciones de una secuaencia con N elementos es $N!$.

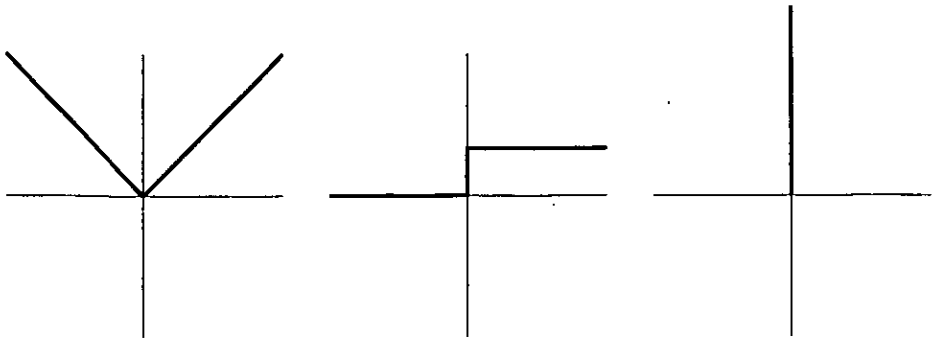


Figura A.1.: El valor absoluto, la función escalón y la función delta.

Propiedades A.2.

$$\delta(x) = 0 \ si \ x \neq 0$$

$$f(x)\delta(x) = f(0)\delta(x)$$

$$\int_{-\infty}^{\infty} dx f(x)\delta(x-a) = f(a)$$

$$\delta(ax) = \frac{1}{|a|}\delta(x)$$

en particular,

$$\delta(-x) = \delta(x)$$

$$\delta(f(x)) = \sum_{n} \frac{\delta(x-x_n)}{|f'(x_n)|}$$

donde la suma es sobre todas las raíces de $f(x) = 0$, *y* $f\prime(x) = df/dx$. *En particular,*

$$\delta(x^2 - a^2) = \frac{1}{2|a|}\{\delta(x-a) + \delta(x+a)\}$$

Función Escalón

Definición A.3. *La función escalón θ se define como:*

$$\theta = \begin{cases} 1 & si\ x > 0 \\ 0 & si\ x < 0 \end{cases}$$

La Integral y derivada de la función Delta (δ)

La integral de δ es la función escalón ya que:

$$\delta(x) = \frac{d}{dx}\theta(x) \tag{A.13}$$

La derivada de δ no se puede obtener de la definición de $\delta(x)$, sino que tiene que ser definida de acuerdo a las propiedades que deseamos que tenga *a priori*. La definición débil de δ' es:

$$\delta' = 0 \text{ si } x \neq 0 \tag{A.14}$$

$$\int_{-\infty}^{\infty} dx f(x)\delta'(x) = -f'(0) \tag{A.15}$$

también podemos decir que:

$$\delta'(-x) = \delta'(x) \tag{A.16}$$

La jerarquía de las singularidades

Existe una jerarquía entre las funciones singulares relacionadas con la delta de Dirac. En esta jerarquía:

$$\ldots\ ,\ |x|\ ,\ \theta(x)\ ,\ \delta(x)\ ,\delta\prime(x)\ ,\ \ldots$$

cada función es la derivada de la función anterior. La diferenciación hace que las funciones se vuelvan cada vez más bruscas, mientras que la integración las vuelve más suaves. Intuitivamente podemos ver que la función escalón θ es menos singular que $\delta(x)$, debido a que solamente tiene una discontinuidad, y no un brinco brusco en el origen (figura A.1.3).

A.1.4. ¿Qué son las ecuaciones diferenciales?

Al igual que una ecuación algebraica es de la forma:

$$f(x^n, x^{n-1}, \ldots, x) = 0 \tag{A.17}$$

Podemos escribir ecuaciones que dependan de las enésimas derivadas de una función $x(t)$:

$$f(x^{n\prime}, x^{n-1\prime}, \ldots, x', x, t) = 0 \tag{A.18}$$

Donde:

$$x^{n\prime} = \frac{dx^n}{dt^n} \tag{A.19}$$

A este tipo de ecuaciones las llamaremos ecuaciones diferenciales debido a que la(s) incógnita(s) es(son) operada(s) por una(s) derivada(s).

A.1.5. ¿Qué son las ecuaciones integrales?

Llamamos ecuación integral a una ecuación en la que la(s) incógnita(s) es(son) operada(s) por una(s) integral(es). Las ecuaciones integrales se clasifican dependiendo de la forma de los límites de integración dentro de ellas. Cuando los límites de integración son constantes decimos que tenemos una ecuación de Fredholm y cuando uno de los límites es una variable (x) la llamamos de Volterra.

Nosotros trabajaremos sólo con ecuaciones integrales de Fredholm, por lo que damos a continuación una definición formal de éstas:

Definición A.4. *Se llama ecuación integral lineal de Fredholm de segunda especie a la ecuación del tipo:*

$$y(x) - \lambda \int_a^b \mathcal{K}(x, x')y(x')\ dx' = f(x) \tag{A.20}$$

donde y(x) es una función incógnita, K(x,x') y f(x) son funciones conocidas, x y x' son variables reales que varían en un intervalo (a,b), y λ es un coeficiente definido.

La función K(x,x') se denomina núcleo *de la ecuación integral, y está definida en el cuadrado Ω:*

$$\Omega\{a \leq x \leq b, a \leq x' \leq b\},$$

que reposa en el plano (x,x'), y además es continuo en él, o bien, sus discontinuidades son tales que la siguiente integral tiene un valor finito:

$$\int_a^b \int_a^b |\mathcal{K}(x, x')|^2 dx dx'.$$

A.1.6. Transformación de ecuaciones diferenciales en ecuaciones integrales

Una manera de resolver ecuaciones diferenciales es transformarlas en ecuaciones integrales. Supongamos que tenemos la siguiente ecuación:

$$-i\frac{dy}{dx} = f(x) \tag{A.21}$$

Si escogemos la condición inicial $y(a) = y_0$, separamos variables, e integramos para obtener la solución de (A.21) obtenemos:

$$y(x) = y_0 + i\int_a^x f(x')dx'. \tag{A.22}$$

Ahora cambiemos las condiciones de frontera, de manera que $x \in [a, b]$, ésto transforma a la ecuación anterior en:

$$y(x) = y_0 + i\int_a^b \theta(x - x')f(x')dx' \tag{A.23}$$

donde θ es la función escalón. Esta función anula a la integral en valores de x' inferiores a x, haciendo que las ecuaciones (A.22) y (A.23) sean equivalentes.
Si definimos al operador integral $\mathcal{K}$ como:

$$\mathcal{K}f(x) = i\int_a^b \theta(x - x')f(x')dx' \tag{A.24}$$

la ecuación (A.23) se transforma en:

$$y(x) = y_0 + \mathcal{K}f(x)$$

En este caso, el núcleo (sección A.4) de este operador es $i\theta(x - x')$.

Definición A.5. *Cuando el núcleo de un operador integral proviene de la solución de una ecuación con un operador diferencial, a éste se le conoce como la* función de Green *G de dicho operador diferencial con determinadas condiciones a la frontera.*

Por tanto, en este caso en particular:

$$G(x, x') = i\theta(x - x') \tag{A.25}$$

es la función de Green para el operador $i\frac{d}{dx}$ con condiciones a la frontera $y(a) = y_0$.

A.2. Funciones de Green

La teoría de las funciones de Green, es un área de la física teórica que se ha convertido en una herramienta muy poderosa para el desarrollo de la mecánica cuántica moderna.

A.2.1. Definición formal de la función de Green

Supongamos que tenemos una ecuación diferencial de n-ésimo orden,

$$\mathcal{L}[y] \equiv p_0(x)y^{(n)} + p_1(x)y^{(n-1)} + \ldots + p_n(x)y = 0, \tag{A.26}$$

donde las funciones $p_0(x), p_1(x), \ldots, p_n(x)$ son continuas y $p_0(x) \neq 0$ en el intervalo [a,b], y tiene las siguientes condiciones a la frontera:

$$\begin{aligned} V_k(y) \equiv \alpha_k y(a) + \alpha_k^{(1)} y'(a) + \ldots + \alpha_k^{(n-1)} y^{n-1}(a) \\ \beta_k y(b) + \beta_k^{(1)} y'(b) + \ldots + \beta_k^{(n-1)} y^{n-1}(b) \end{aligned} \tag{A.27}$$

$$k = (1, 2, \ldots, n)$$

donde las formas lineales $V_1, \ldots, V_n$, $y(a), \ldots, y^{n-1}(a)$ y $y(b), \ldots, y^{n-1}(b)$ son linealmente independientes.

Debemos suponer que el problema homogéneo (A.26) con condiciones a la frontera (A.27), admite sólo la solución trivial $y(x) \equiv 0$.

Definición A.6. *Se llama función de Green del problema de frontera* (A.26) (A.27) *a la función $G(x, x')$ construida para cualquier punto x', en el que $a < x' < b$, y que cumpla con las siguientes propiedades:*

1. *$G(x, x')$ es continua y tiene derivadas continuas con respecto a x hasta de* (n-1)*-ésimo grado inclusive para $a \leq x \leq b$.*

2. *Su derivada* n*-ésima con respecto a x tiene una discontinuidad de primer orden* [5] *en el punto $x = x'$, siendo el salto igual a $\frac{1}{p_0(x)}$, es decir.*

$$\left.\frac{\partial^{n-1} G(x.x')}{\partial x^{n-1}}\right|_{x=x'+0} - \left.\frac{\partial^{n-1} G(x.x')}{\partial x^{n-1}}\right|_{x=x'-0} = \frac{1}{p_0(x)} \tag{A.28}$$

3. *En cada intervalo [a,x') y (x',b], la función $G(x, x')$, considerada como una función de x, es una solución de la ecuación* (A.26)*:*

$$\mathcal{L}[G] = 0 \tag{A.29}$$

4. *$G(x, x')$ satisface las condiciones de frontera* (A.27)

$$V_k(G) = 0 \qquad k = (1, 2, \ldots, n) \tag{A.30}$$

[5] Barton [9] la llama también discontinuidad de salto *jump condition*, porque es parecida a la discontinuidad que tiene la función valor absoluto en el origen. Es una discontinuidad que surge al derivar la función dos veces.

Teorema A.1. *Si el problema de frontera* (A.26)- (A.27) *tiene sólo una solución trivial* $y(x) \equiv 0$, *el operador* $\mathcal{L}$ *posee una y sólo una función de Green* $G(x, x')$

Las funciones de Green pueden aplicarse a problemas de frontera que contengan un parámetro:

$$\mathcal{L}[y] = \lambda y + h(x) \tag{A.31}$$

siendo transformadas a ecuaciones integrales.

Teorema A.2. *Si el problema de frontera homogéneo* (A.31)- (A.27) *tiene una función de Green* $G(x, x')$, *el problema de frontera no homogéneo* (A.31)- (A.27) *es equivalente a la ecuación integral de Fredholm*

$$y(x) = \lambda \int_a^b G(x, x') y(x')\, dx' + f(x) \tag{A.32}$$

donde:

$$f(x) = \int_a^b G(x, x') h(x')\, dx' \tag{A.33}$$

A.2.2. Funciones de Green dependientes del tiempo

En esta sección estudiaremos las técnicas de solución de ecuaciones diferenciales parciales dependientes del tiempo. Éstas tienen la forma:

$$\mathcal{D}\phi + \frac{\partial \phi}{\partial \tau} = 0 \tag{A.34}$$

donde τ es la variable temporal y $\mathcal{D}$ es un operador *independiente del tiempo*, que cuenta con un conjunto completo de funciones propias. En fisicoquímica encontramos muchos ejemplos de este tipo de ecuaciones, como lo son la ecuación de difusión:

$$\nabla^2 \rho = \frac{1}{D^2} \frac{\partial \rho}{\partial t}$$

donde ρ es la densidad del material que se esta difundiendo y D es la constante de difusión. La ecuación de Schrödinger dependiente del tiempo

$$\hat{\mathcal{H}}\phi = i\hbar \frac{\partial \phi}{\partial t}$$

Como habíamos dicho anteriormente, el operador $\mathcal{D}$ tiene asociado un conjunto completo de funciones propias

$$\mathcal{D}|\psi_i\rangle = \lambda_i |\psi_i\rangle \tag{A.35}$$

lo que nos permite representar la solución de (A.34) en términos de una expansión de dichas funciones propias[6]:

$$\phi(\mathbf{r},\tau) = \sum_i A_i(\tau)\phi_i(\mathbf{r}) \tag{A.36}$$

donde:

$$A_i = \int_{\infty}^{\infty} d(\mathbf{r})\ \psi_i^*(\mathbf{r})\phi_i(\mathbf{r},0). \tag{A.37}$$

Sustituyendo (A.36) en (A.34) obtenemos la siguiente relación:

$$\sum_i \left[\lambda_i A_i(\tau) + \frac{dA_i(\tau)}{d\tau}\right]\phi(\mathbf{r}) = 0 \tag{A.38}$$

Para que el término de la derecha de la ecuación anterior se anule, es necesario que para toda i se cumpla que:

$$\frac{dA_i(\tau)}{d\tau} + \lambda_i A_i(\tau) = 0. \tag{A.39}$$

Resolviendo le ecuación diferencial anterior para $A_i(\tau)$:

$$A_i(\tau) = A_i(0)e^{-\lambda_i\tau} \tag{A.40}$$

Por lo tanto, la ecuación (A.36) se puede reescribir como:

$$\phi(\mathbf{r},\tau) = \sum_i A_i(0)\psi_i(\mathbf{r})e^{-\lambda_i\tau} \tag{A.41}$$

Sustituyendo (A.37) en (A.41) podemos construir la ecuación integral para (A.34)

$$\phi(\mathbf{r},\tau) = \int d(\mathbf{r}')G(\mathbf{r},\mathbf{r}',\tau)\phi(\mathbf{r}',0) \tag{A.42}$$

donde la función de Green $G(\mathbf{r},\mathbf{r}';\tau)$ se puede desarrollar como:

$$G(\mathbf{r},\mathbf{r}';\tau) = \sum_i \psi_i(\mathbf{r})\psi_i^*(\mathbf{r},)e^{-\lambda_i\tau} \tag{A.43}$$

[6]debemos de recordar que el conjunto de funciones ψ_i es completo y ortonormal.

B. El software libre: ¿Una alternativa?

B.1. ¿Qué es el software libre?

La esencia del software libre [18] no es que sea gratuito (en el sentido de "free", que en inglés significa tanto gratuito como libre), sino que viene con el *código fuente*. Además viene con una licencia que afirma que la gente es libre de modificarlo para sus propio beneficio. Esta "libertad" permite a los desarrolladores hacer cambios o compilar el software en distintas máquinas.

Para aclarar el panorama, podemos dividir al software en las siguientes categorías:

- Software Comercial: Este software se distribuye en forma binaria, es decir, en una forma lista para ejecutarse en la computadora, y en una forma imposible de modificar. Generalmente los autores de este software cobran por el uso de éste, actualizaciones y a veces hasta soporte técnico. Este es el modelo que puede ser visto como el "clásico", si pensamos que el software es un producto análogo a un automóvil o a un aparato eléctrico. No hay libertad de compartir, ni de modificar y muchas veces ni de ejecutar el software en circunstancias específicas.

- Software de Dominio Público: A veces el Freeware y el software Libre se confunden con esta categoría. Si un software está en el dominio público, no pertenece a nadie. No hay restricciones en cuanto a su uso o distribución. No existe ninguna protección para el programador, de manera que alguien puede tomar su producto, modificarlo y luego revenderlo sin distribuir el código fuente. Con el Freeware y el Software Libre, los desarrolladores mantienen un nivel de control y pertenencia que es perdido si el software se libera bajo el dominio público. La licencia de dominio público es la que da más libertad al usuario, sin embargo deben existir algunas medidas para conservar el propósito inicial de la liberación del software: que quede libremente en manos de los usuarios.

- Shareware: Los programas de Shareware se distribuyen libremente, sin embargo su uso requiere de una donación al autor del programa en cuestión, además rara vez vienen acompañados del código fuente. Es decir, es software propietario

que permite un periodo de evaluación y libre distribución. Muchas veces el programa viene en forma de "demostración", por lo que sus capacidades no son activadas hasta que el usuario no pague por el programa.

- Freeware: De acuerdo a David Bunnel, el término "Freeware" fue usado por primera vez por Andrew Fluegelman, quien pensaba que era adecuado distribuir gratuitamente los binarios, mas no el código fuente. De esta manera se distribuyen actualmente algunos navegadores de Web (por ejemplo, Internet Explorer). Debido a que el código fuente de estos programas no se libera, en general se quedan en estadios de desarrollo muy primitivos.

- Software Libre: El software libre difiere del Freeware en el sentido que los programas se distribuyen tanto en forma binaria como en código fuente. Es el código fuente el que permite a los desarrolladores la posibilidad de modificar el software y adecuarlo a sus necesidades.
  La persona clave para este tipo de software es Richard Stallman, creador de la "Free Software Foundation" [35], fundación que está dedicada a promover la idea que el código fuente de todos los programas debe ser accesible libremente. Para hacerlo posible, Stallman creó la licencia pública general (General Public Licence, GPL), que define claramente los términos en los que el código fuente debe ser distribuido. En particular, la licencia GPL requiere que software que incorpore fragmentos de otro programa GPL también sea liberado bajo ésta misma licencia. Para incluir una pieza de software en esta categoría, es necesario que la licencia permita que el usuario pueda modificar el programa para adecuarlo a sus necesidades personales, o para corregir algún desperfecto.

- Open Source software: Hasta ahora, los autores no ha encontrado una traducción adecuada a este término. Debido a que existen más licencias además de la GPL, que difieren de ésta en algunos aspectos (por ejemplo la licencia Artística, la licencia BSD, etc.) surgió la necesidad de acuñar un nuevo término que englobara a todas éstas en una característica esencial, que es la **distribución del código fuente**.

Larry Wall, autor de Perl, un lenguaje de programación que está liberado bajo una licencia de tipo *Open Source*, afirma que [20]:

> Hace muchos años, todo lo que teníamos era software comercial. Claro, también escribíamos nuestros propios programas, pero no había un mecanismo adecuado para compartir el software, además ciertamente no podías lograr que tu compañía distribuyera software libre oficialmente.

> Después vino el movimiento del Software libre. Desde luego, todos pensamos en Richard Stallman en este punto. El era la punta de lanza del movimiento, pero el hecho es que el tiempo estaba maduro para ésto, y muchos de nosotros estábamos publicando software libre por esas épocas. Lo teníamos que hacer a pesar de los intereses comerciales de esa época, y teníamos que inventar nuestra propia infraestructura, por lo que naturalmente surgió un antagonismo entre la comunidad dedicada al software comercial y la comunidad del software libre. (...)
>
> Algunos de ustedes creen que el software libre tiene su lugar (...) en la creación de nuevas ideas, pero que una pieza de software realmente útil solo puede provenir de un ambiente comercializado por la industria. Otros de ustedes (...) piensan que cuando una entidad comercial tiene interés en un programa de Freeware (...) van a aplastarlo, orinarlo y estropearlo.
>
> Estas dos ideas son peligrosas (...) por que son sólo parcialmente ciertas. (...) al menos podemos pensar en un cese al fuego entre el mundo comercial y el mundo del software libre(...)
>
> Debe de existir un modelo nuevo, intermedio, en el cual la comunidad de software libre trabaje en lo que es buena, y la comunidad comercial se dedique a lo suyo (...) Hay cosas que la comunidad de software libre no hace bien (...) hay cosas que la comunidad de software comercial no hace bien. Pero podemos hacer cosas los unos para los otros.

B.2. Modelos de desarrollo de software libre

En la historia de la sopa de piedras [23], tres soldados comienzan a hervir piedras en agua, en un pueblito, proclamando que están cocinando una sopa. Convencen a los pobladores de que su sopa sería mejor si les regalaran algo de carne y vegetales para ella. Al final, logran tener una verdadera sopa a partir de todos los donativos.

Desarrollar software libre es como hacer una "sopa de piedras". El compartir en un proceso de desarrollo de software libre crea un todo mayor que la suma de sus partes.

Como Tim O' Reilly afirma [18]:

> En esta comunidad (...) uno gana prestigio de acuerdo a lo que aporta. Una buena idea debe estar respaldada por una buena implementación

> que pueda ser probada por tus colaboradores. Como en muchos otros ámbitos, Internet por sí mismo ha acelerado la velocidad de innovación. La comunicación puede ser instantánea y mundial, permitiendo un grado de colaboración sin precedentes.

La desventaja del software propietario es la misma que la de los estándares que salen después de que un comité cerrado los crea: nunca pueden obtener un buen producto. Esto se debe a que, a diferencia del software libre, su producto no se pone a prueba de manera extensa hasta que sale al mercado. El software libre siempre está a prueba por miles de gentes.

El artículo de Eric Raymond "The Cathedral and the Bazaar" [7], propone dividir a los modelos de desarrollo del software libre en dos grandes categorías:

- La Catedral: En este modelo, las decisiones sobre el desarrollo del software se centran en un pequeño comité de "monjes" , los cuales aprueban o desaprueban los cambios al código fuente con mucho cuidado y liberan versiones de software estables y muy probadas.
- El Bazar: Las decisiones que se toman en este modelo se parecen a un "bazar", en donde la gente propone una idea, ésta se discute y se implementa, arreglando los errores "bajo la marcha", en vez de cuidar demasiado el código fuente.

De hecho, este tipo de organizaciones se pueden equiparar a las estructuras verticales (catedral) y horizontales (bazar) del análisis clásico. [8].

La mayoría de los proyectos de software libre están en un intermedio entre ambos modelos, teniendo algunas características de ambos paradigmas.

B.3. Algunos ejemplos de programas de software libre

Enumeraremos una pequeñísima cantidad del software libre. Esta revisión no es exhaustiva, y simplemente trata de ejemplificar la diversidad de programas que componen la colección. Existen lugares en la red donde se pueden encontrar listados muy completos de aplicaciones de software libre. [33]

B.3.1. GNU Emacs

Emacs es un proyecto muy maduro. Es un editor que pertenece a la familia GNU. Tiene innumerables características. Un programador en UNIX puede realizar casi todas sus tareas cotidianas sin salir de este programa.

B.3.2. GCC y los programas GNU

GNU es el proyecto de la Free Software Foundation para crear un sistema operativo totalmente libre. Las herramientas GNU son los primeros bloques de construcción que se crearon para este motivo.

El compilador GCC es la herramienta de compilación estándar en el mundo del software libre. Desarrollado por el proyecto GNU, es la base en la cual se compilan todos los programas GNU. Los programas dentro de este proyecto abarcan desde juegos de Go, pasan por calculadoras, utilerias UNIX, etc. Éstos son de los hijos directos de los esfuerzos de Richard Stallman y sus innumerables seguidores.

B.3.3. TEX

Donald E. Knuth se esforzaba por crear un sistema de composición que le permitiera publicar su serie de libros The Art of Scientific Computing, en la cual necesitaba incluir gran cantidad de fórmulas, ecuaciones y símbolos matemáticos. Acabó tardándose 8 años, pero el resultado fue TEX: un lenguaje de composición robusto y extensible que funciona mediante comandos que especifican el cómo, cuándo y dónde aparecen los elementos del texto. LATEX, un derivado de TEX, es el software en el que se diseñó esta tesis.

B.3.4. Linux

Linus Thorvalds comenzó a trabajar en Linux en 1991. Al principio era un pequeño sistema operativo experimental. Ahora se ha convertido en uno de los "baluartes" más importantes del software libre. Linux, sin menospreciar a otros sistemas operativos similares (HURD, la familia BSD, etc.) es el sistema operativo de Software Libre por excelencia.

La definición de Linux de Robert F. Young, presidente de RedHat software [31] es adecuada:

> Linux es un sistema operativo avanzado, multitarea y multiusuario (...) (y) es el resultado de un desarrollo mundial a través de Internet. Se puede usar como una estación de trabajo UNIX o como servidor para tareas que abarcan desde servidor de Web altamente eficiente hasta estaciones de trabajo de bajo costo para redes cliente/servidor.
>
> La lista de características de Linux es impresionante. (...) Ofrece un ambiente estable de multitarea y multithreading en todas las plataformas

que soporta, soporte para procesamiento simétrico (SMP), y controladores para el hardware mas diverso. Sin embargo, los factores que aseguran el éxito a largo plazo de Linux tienen poco que ver con la lista de características, sino mas bien con su licencia.

B.3.5. Apache

Apache domina el mercado de los servidores de Web. Más del 50% de todos los servidores de Web trabajan con Apache. Apache fue creado por el Grupo Apache, que tomó el desarrollo del servidor que originalmente se creó en el National Center of Super Computer Applications (NCSA).

B.3.6. Netscape

Los directivos de la empresa Netscape, al darse cuenta que perdían mercado ante Microsoft Explorer en el mercado de los navegadores, ya que Microsoft regalaba su navegador (mas no el código fuente), e inclusive pretendía incluirlo por defecto en su nuevo sistema operativo Windows 98 decidieron liberar el código fuente de Netscape y cambiar la licencia a una no restrictiva del tipo Open Source.
Esta decisión se tomó después de que consultaron a Eric Raymond, el autor the "The Cathedral and the Bazaar" [7] y analizaron los comentarios de los entusiastas de software libre en diversos foros de discusión de Internet. [10], donde la comunidad de software libre sugería una apertura del código para poder lograr una mejoría en las calidades de los navegadores de esta compañía.
Netscape creó una licencia parecida a la GNU/GPL pero que se adecua más a las condiciones que deseaban conservar para su software, ésta licencia se propone inclusive para otros proyectos que no sean de Netscape. Esta licencia se conoce como licencia NPL (Netscape Public License) [11].
La decisión de Netscape dio mucha publicidad al software libre, y ha generado mucha discusión. Hay gente que la apoya y hay gente que cree que la fragmentación destruirá a los productos de la empresa.
Netscape como empresa gana la mayor parte de sus entradas en el mercado de servidores de Web (en los que compite con productos de software libre como Apache y con productos comerciales como los de Microsoft) y en el mercado de valor agregado en Internet, compitiendo con los "Web Portals" (Altavista, Yahoo, Lycos) e inclusive con Microsoft.

Al apoyar a Linux [14], Netscape ofrece servidores de Web que son diez veces mas baratos que su alternativa en Windows NT. Si el usuario utilizara Apache, el costo sería nulo.

B.3.7. Otros programas

BIND

El "Berkeley Internet Name Daemon" . Sin este programa, cualquier usuario de Internet debería escribir 207.25.97.99 en vez de www.ibm.com. Como varios programas de software libre que son la base de Internet, BIND pertenece a los primeros esfuerzos de software libre.

Perl

El lenguaje escogido por la mayoría de los programadores de CGI, y parte indispensable de la administración de casi todos los sitios de Internet, Perl ha sido descrito como "la cinta adhesiva de Internet". Originalmente creado por Larry Wall, Perl ahora es mantenido por un grupo de cientos de programadores distribuidos por la red, que se comunican por una lista de correo. Larry mantiene un "control artístico" sobre el lenguaje, pero la mayoría del desarrollo es hecho por otros. Existen mecanismos de extensiones que permiten a la gente hacer modificaciones libremente.

The Gimp

Desarrollado por un grupo de entusiastas programadores, este programa es uno de los ejemplos que hacen constar que el software libre puede llegar a construir aplicaciones para el usuario final muy complejas. Es programa con capacidades similares a las de Photoshop.

GNOME y KDE

GNOME y KDE son esfuerzos comenzados recientemente. La idea es crear "escritorios" y "conjuntos de aplicaciones " consistentes y amigables al usuario. Antes de la propuesta de ambos proyectos, las interfases gráficas del mundo del Software Libre estaban muy fragmentadas, ya que cada gente utilizaba un distinto toolkit gráfico y distintas convenciones para los programas. Ambos proyectos han recibido apoyo de varias empresas comerciales y de fundaciones de software libre. Es decir, estos proyectos intentan unificar el "look and feel" del ambiente gráfico UNIX.

B.4. ¿Y de qué vive un desarrollador de software libre? Agentes independientes y empresas.

B.4.1. Empresas

Cygnus

El ejemplo de Cygnus Solutions, Inc. se puede ver en la entrevista que le hizo Sengan de Slashdot a Michael Timman [24], autor del front-end de C++ para gcc y fundador de Cygnus. En ella, Timman afirma que :

> Comenzamos Cygnus con casi nada de dinero ... propuse que cada uno de los tres fundadores pusiera $2000 y subiera a $5000 cuando pudiera hacerlo ... no hicimos nada para interesar a los bancos, por lo que no hubo que explicarles nada. (...)
>
> Acerca de encontrar a nuestros primeros clientes ... fue tocar suficientes veces las puertas correctas. Siendo el autor de GNU C++ era muy sencillo para mí hablar con contactos técnicos de alto nivel, que me presentaban con los gerentes de mediano nivel. Vendíamos soporte de GNU primordialmente como una medida de ahorro, por lo que no necesitábamos aprobación de los vicepresidentes de las empresas. Ahora tenemos ofertas de carácter más estratégico, pero también estamos mucho mejor establecidos, por lo que podemos llamar y pedir aprobación a niveles de vicepresidencia. (...)
> Siempre le he recomendado a todo mundo que si quieren empezar un negocio de software libre, deben hacerlo sin dejar que nada los detenga.
> De los que lo hicieron, muchos lograron salir tablas o lograr pequeñas ganancias, lo que me hace creer que hay más en el éxito de Cygnus que atacar una oportunidad de mercado. Siempre hemos estado dispuestos a hacer decisiones fuertes y siempre hemos oído a nuestros clientes, hasta cuando están gritando. Los que hacen software libre por amor y no por dinero, no sobrevivirán lo que hemos sobrevivido nosotros.
> Acerca de lo que los clientes realmente quieren ... ellos quieren que alguien oiga sus necesidades, formule un plan a largo plazo, y posteriormente lo cumpla sin riesgos. A algunos clientes les gusta ver tu trabajo, a otros les gusta verte trabajar, y otros piensan que haces un buen trabajo si nunca te ven. Al final, no hay garantía de que los clientes regresen por lo que ofreces. El modelo de software libre es único debido a que da un incentivo a que los clientes trabajen contigo y no independientemente. En este mundo de alta tecnología, esa ventaja nos ha puesto en el lugar número 1 del mercado (

...)

Además, la empresa Cygnus está comprometida al desarrollo del software libre de distintas maneras:

> Invertimos como 10 mil dólares al año en el desarrollo de nuevo software libre. Corremos servidores de donde más de medio millón de personas han bajado nuestros programas. Promovemos el concepto de software libre en artículos, conferencias y anuncios. Promovemos a proyectos de software libre y le pagamos a contratistas que hacen trabajo relacionado con nuestros requerimientos de negocios. Proveemos un generoso fondo de compensación, de manera que los empleados o clientes que quieran donar algo a la FSF puedan ver multiplicada su donación.

Red Hat

Red Hat software es la empresa que desarrolla la distribución gratuita mas popular de Linux. [26]. La citamos como una empresa especial, ya que tiene a más del 10% de su gente en investigación y desarrollo. Contando con personalidades en del mundo de la programación como Alan Cox o Rasterman. Esta empresa combina la distribución de Linux en varias plataformas, la venta de aplicaciones comerciales para el sistema operativo, y la creación de software libre.
El crecimiento en las ventas de Red Hat es de hacerse notar, ya que sus ventas han subido sus ventas de 450,000 CD's en 1996 a 750,000 en 1997. Red Hat Linux ha ganado el premio de InfoWorld al mejor producto tanto en 1996 como en 1997.

Alternativas para la creación de nuevas empresas

Acerca del desarrollo del software libre como alternativa para empresas, Linus Thorvalds, el principal autor del sistema operativo Linux afirma: [27]

> (...) Lo que se ve en muchas compañías es departamentos de mercadotecnia y de desarrollo completamente separados.
> Hay algunas interfases entre ellos, y mercadotecnia es el departamento que usualmente le dice a desarrollo que hacer. El hecho es que dentro de una compañía puedes tener un grupo de desarrollo que guste de estar encima de las cosas y hacer lo correcto. Entonces, tienes un departamento de mercadotecnia con distintos objetivos y un constante conflicto de opiniones.
> (...) En un ambiente de software libre los grupos están desacoplados. (...) Netscape desacopló a Mozilla para que fuera una rama completamente

> diferente, sin embargo, siguen en el mismo edificio. Otra alternativa es el kernel de Linux. Estoy completamente desinteresado de todos los aspectos de mercadotecnia. No quiero trabajar en una empresa de Linux. (...) No quiero sentir que el éxito económico de la compañía en la que trabajo depende de las decisiones técnicas que yo haga. Yo quiero hacer las decisiones técnicas basado solamente en datos técnicos. (...) Sin embargo, eso no contradice el trabajar con un departamento de desarrollo. Por ejemplo, está Red Hat Linux. En su caso, existe un grupo separado que hace la mercadotecnia, distribución y empacado. Es similar al modelo tradicional, pero más claramente desacoplado. Esto hace más natural el tener un grupo de desarrollo, pero tal vez 15 organizaciones de mercadotecnia. (...) hace más sentido económicamente.

Muchos de los desarrolladores de software libre no viven de éste, sino de empleos alternos. (como administradores, programadores, etc).
La ventaja de Linux es que no es producto de una empresa, sino de la comunidad. Las empresas que viven de Linux sólo toman lo que ya está hecho, lo empaquetan y lo comercializan (a excepción de algunas, que también cooperan con desarrollo adicional).

B.5. El mercado del software libre

B.5.1. Análisis de la industria del software

Una manera de organizar la información acerca de una industria, de manera que nos permita analizar sus atractivos potenciales es el modelo de las cinco fuerzas, que fue desarrollado por Michael Porter [8]. El poner atención a estas cinco fuerzas es útil para ver si conviene entrar a competir a un mercado o no.

Intentaremos analizar cada una de las cinco fuerzas con respecto a la industria del software, intentando evaluar con cuidado las perspectivas que tiene el software libre dentro de esta industria.

Primera Fuerza: Los competidores

Porter afirma que la rivalidad intensa entre las compañías que participan en un mercado reduce las ganancias promedio. La *coordinación* de las industria genera mayores ganancias. En general, existen diferencias entre los intereses colectivos y los intereses individuales de un grupo de empresas competidoras, por lo que el balance de coordinación y competencia siempre es dinámico.

Oster [8] propone varios axiomas acerca de el efecto de la distribución de competidores en un mercado. Algunos de ellos son:

- Axioma 1. Un gran número de empresas hace imposible la coordinación y hace al mercado más eficiente.
- Axioma 2. En general, en industrias donde las empresas principales son del mismo tamaño, la rivalidad se intensifica.
- Axioma 3. Entre más similares sean las firmas en un mercado, será más fácil la coordinación entre ellas.
- Axioma 4. Las industrias con grandes activos específicos tendrán mayores barreras de salida, y por lo tanto tenderán a ser más competitivas.
- Axioma 5. Una demanda variada crea mayor rivalidad.

En el caso de la industria del software, existen muchos competidores, sin embargo, el principal competidor es Microsoft, con participaciones de mercado apabullantes en casi todos los segmentos. En relación a los *axiomas* de Oster, el mercado del software se encuentra en:

- Axioma 1. El número de empresas de software es grande. Por lo que se podría pensar que el precio del software debería estar en un óptimo de mercado.
- Axioma 2. Este factor es muy importante para nuestro análisis, el tamaño de las empresas es desigual. Microsoft es el principal competidor en casi todos los segmentos. Por lo que su tamaño comparativo es del orden "monopólico".
- Axioma 3. En este caso, las firmas son distintas. Microsoft al tener una impresionante sinergia y penetrar en todos los segmentos del mercado, tiene objetivos distintas a los de las demás empresas. Usualmente el objetivo de los pequeños competidores es el de ganar participación de Mercado, mientras Microsoft puede adquirir a sus competidores (como en el caso de la reciente adquisición de Hot-Mail), o hacer prácticas de "dumping" (como en el caso de Internet Explorer y Netscape)
- Axioma 4. En la industria del software, los activos específicos son muy pequeños. El principal elemento productivo es el programador, por lo que visto como activo es fácilmente reemplazable o cambiable al desarrollo de otra línea de software. Si una empresa pierde en un segmento, puede relocalizar a sus equipos de diseño y programadores para comenzar a producir software en otro completamente distinto.

- Axioma 5. La demanda de software ha crecido enormemente en los últimos años. Las empresas y personas físicas requieren de software cada vez más especializado. Este nicho de mercado es excelente para los competidores pequeños. El monopolio de Microsoft se ha concentrado en tomar los nichos de gran consumo: los sistemas operativos y las aplicaciones de oficina. El cuestionable requisito de la *compatibilidad* ha hecho que muchos clientes hayan optado por utilizar los productos de Microsoft, y constantemente actualizar sus computadoras con la versión mas reciente de dichos productos cuasi-monopólicos. Por lo que la demanda en estos segmentos está sesgada hacia comprar los productos *compatibles* y *ya conocidos*.

El software libre, como competidor en este mercado, presenta las siguientes ventajas competitivas con respecto a cada uno de los *axiomas*:

- Axioma 1: Debido a que los productos de software libre carecen de precio no se afectan por la coordinación de los competidores. Solamente cuando los competidores han querido cerrar estándares ha sido el software libre el que ha salido perjudicado. Por ejemplo, el desarrollo de Linux en la plataforma SPARC fue detenido al principio por la falta de información de las computadoras de Sun Microsystems disponible al público en general. Sin embargo, las cosas están cambiando rápidamente, por ejemplo, The Open Group, una fundación que pretende crear el estándar Unix 98, y que está colaborando con los principales actores del mercado, se acercó al grupo internacional de usuarios de Linux en la última reunión de LinuxExpo y discutió la manera de que Linux pueda adherirse a dicho estándar.

- Axioma 2: En el caso del software libre, se está alcanzando el *tamaño crítico* para lograr que la competitividad del mercado se afecte de manera considerable. En algunos segmentos, como en el de servidores de Web con Apache, la mayoría del mercado trabaja con software libre. Sin embargo, de nuevo, como Apache es gratuito, en vez de lograr cooperación entre las empresas capitalistas, tiene un efecto contrario, debilitando su capacidad de poner un precio excesivo a sus productos.

- Axioma 3: El grupo de personas que desarrolla el software libre tiene intereses diametralmente opuestos a los de las empresas de software comercial. Por lo que no son similares, así que mientras exista una mayor penetración de mercado del software libre, se romperá la cooperación.

- Axioma 4: Al no tener "activos" más que su propio intelecto, y al no dirigir a nadie más que ellos mismos, los programadores de software libre tienden a ignorar lo que presupone este axioma, y programar su aplicación hasta lograr

su objetivo. Generalmente los equipos de trabajo de software libre continúan trabajando en un proyecto aún cuando los primeros creadores del software ya no trabajen en éste.

- Axioma 5: Dentro del mercado del software, cualquier alternativa libre generará mayor rivalidad entre las empresas, pues tendrán que competir contra la aparición del nuevo producto gratuito. El problema al que se enfrentan actualmente los programadores de software libre, es el de convencer a la gente de que sus productos son igual o más aptos para su propósito que sus competidores comerciales.

Segunda Fuerza: Presencia de productos substitutos

Los productos sustitutos son aquellos que afectan a la industria del software desde otros mercados. Por ejemplo, la industria de aire acondicionado compite con la industria de los ventiladores. En el caso del software, éste compite con *hardware* que realiza actividades similares. Los procesadores de palabras compiten con las máquinas de escribir eléctricas, o la propaganda por la Web compite con la propaganda por radio o TV.
En general, el mercado del software no se encuentra amenazado fuertemente por productos sustitutos, sino que más bien amenaza a otros mercados como sustituto emergente. Existen algunas excepciones, como la competencia entre los juegos de consola y los juegos de vídeo en computadora, pero la tendencia general es la mencionada.
De esta manera, las presiones por productos sustitutos no juegan un papel importante en esta industria.

Tercera Fuerza: El poder de los compradores

Sharon M. Oster describe a esta fuerza de Porter con un buen ejemplo:

> La industria del acero vende una cantidad sustancial a la industria automotriz. Ciertamente este conjunto de clientes tiene mucho más poder que los consumidores individuales de Coca Cola. Para la Coca Cola, por ejemplo, las compañías como Burger King o McDonald's tienen mucho mayor poder.

Oster menciona cuatro factores que determinan esta fuerza de Porter:

- Factor 1: A mayor número de compradores y menos compras individuales, el poder del comprador es menor.

- Factor 2: La estandarización de los productos incrementa el poder del comprador. Debido a que le permite fácilmente cambiar de un producto a otro.

- Factor 3: Cuando los compradores pueden integrarse hacia atrás, y producir lo que están comprando, aumenta su poder de negociación.

- Factor 4: Entre más abiertas sean las transacciones comerciales, mayor es el poder de negociación de los compradores, pues pueden buscar otros productos fácilmente.

El monopolio de Microsoft se encuentra en la siguiente posición con respecto a los factores:

- Factor 1: La industria del software tiene muchos compradores. El comprador típico adquirirá una copia de un sistema operativo, algunas aplicaciones de oficina, unos juegos y tal vez alguna aplicación específica. Esto beneficia mucho a los titanes de la industria del software, debido a que la capacidad de negociación del cliente es muy limitada.

- Factor 2: Crear un sistema operativo que pueda ejecutar las aplicaciones de los sistemas operativos de Microsoft, es una labor titánica. De esta manera, Microsoft se protege contra la estandarización, y la gente debe de comprarle a ellos para poder correr muchas aplicaciones específicas. Para proteger este factor, y lograr que el comprador no pueda migrar fácilmente a otros sistemas operativos, Microsoft ha luchado tras bambalinas contra el lenguaje de programación Java de Sun Microsystems, que promete "escribir una vez, correr donde sea". Si existieran muchas aplicaciones en este lenguaje, el sistema operativo sería prescindible. [3]

- Factor 3: Casi ningún usuario tiene el tiempo de programar su propio procesador de palabras, o su propio sistema operativo por él solo. De esta manera, si quisiera integrarse hacia atrás, tendría que unirse o crear un proyecto de software libre, Freeware, etc. para satisfacer sus necesidades inmediatas. Siendo éste porcentaje de la población uno muy pequeño, el comprador no puede ejercer mucho poder de esta manera.

- Factor 4: Los compradores usualmente compran el software a través de terceras personas. La negociación con el proveedor (la compañía de software) es prácticamente inexistente, a menos que la aplicación sea hecha específicamente para el cliente.

El Software libre tiene la siguiente posición:

- Factor 1: En el software libre, los "compradores" son la gente que obtiene el programa de alguna manera y lo ejecuta. Entre más compradores existan, la interacción cliente-proveedor es mucho mejor, pues los medios electrónicos de soporte y desarrollo se enriquecen de la cantidad de "compradores".

- Factor 2: Este es un gran problema para el software libre. Competir con estándares cerrados siempre hace que sea difícil lograr estandarizarse con el software comercial. Sin embargo, muchos protocolos de Internet están estandarizados abiertamente con comités heterogéneos que determinan los pasos a seguir en el futuro.

- Factor 3: El software libre es el ejemplo perfecto de este factor, pues los desarrolladores se integraron verticalmente hacia atrás hasta lograr tener un sistema libre "autónomo", pues o depende de ninguna aplicación comercial ni para su compilación ni ejecución.

- Factor 4: Al no existir una transacción formal en el uso del software libre, sino simplemente el seguimiento implícito de la licencia GPL, éste factor no se aplica al software libre.

Cuarta Fuerza: El poder de los proveedores

La industria del software no tiene proveedores de insumos importantes. En esta industria, las compañías de software crean su producto sin tener que comprar alguna materia prima. Por lo que no existe el riesgo de perder ganancias por que algún proveedor suba los costos de los insumos.

Quinta Fuerza: Barreras de Entrada

Tratar de generalizar sobre barreras de entrada es muy difícil. Cada segmento del mercado tiene distintos comportamientos. Existen desde los mercados muy protegidos y difíciles de acceder (sistemas operativos), segmentos muy competidos (el mercado de soporte e instalación) y constantemente existen nuevos nichos que son ocupados rápidamente por ávidos competidores que desean obtener la ventaja de entrar primero.

B.5.2. Fuentes de ventajas competitivas

Sin embargo, no todo es tan fácil como suena. Creer que las fuerzas de Porter y sus factores y axiomas son los parámetros que determinan el verdadero éxito de una empresa es una utopía. Michael Pfeffer [19] afirma que la fuente de ventajas competitivas ha evolucionado a través del tiempo. Éste autor afirma que las personas son la principal fuente de ventajas competitivas, por lo que tener gente capaz, motivada

y bien entrenada, se vuelve un recurso muy importante a nivel empresa. Puede ser que dos empresas con las mismas oportunidades financieras y tecnológicas tengan dos desarrollos completamente distintos debido a la motivación o al entrenamiento de su personal.

El caso del software libre es un caso muy interesante. Debido a que mucha de la gente que colabora con ésta "empresa sin fronteras" del software libre, está motivada de manera personal. La mayoría de las veces casi no recibe motivaciones externas, fuera del reconocimiento dentro de la comunidad o el gozo de que su nuevo programa de buscaminas se vea bonito y corra correctamente.

Existen empresas como AMD (Advanced Micro Devices), que siguen modelos internos muy parecidos a los del software libre, es decir con estructuras de poder horizontales o matriciales que fomentan la interacción entre los distintos elementos de la compañía, y que además creen que el entrenamiento es parte fundamental de las actividades de los trabajadores. Inclusive, la liberación del código fuente de Navigator de Netscape sigue la misma tendencia: algunas corporaciones están intentando imitar el éxito de el modelo de desarrollo libre.

Aprender las habilidades de las empresas rivales es otra ventaja competitiva [38]. Muchas de las empresas grandes de software prefieren comprar a sus rivales en vez de desarrollar una nueva tecnología. Existen partes estratégicas que ninguna empresa *suelta* fácilmente. Microsoft no vende la tecnología que utiliza para crear sus sistemas operativos por ejemplo.

La **habilidad** de los desarrolladores de software libre es la constante innovación. Las curvas de aprendizaje, que Oster considera como factores importantes para obtener ventajas competitivas [8], son mucho menores en ambientes donde la interacción entre desarrolladores y usuarios es de primera línea.

B.5.3. Penetración de mercado del software libre

Tim O´Reilly [22] afirma que aunque Microsoft intente convencer al mundo que la capital de Internet está en Redmond, y que Netscape afirma que está en Mountain View,

> los verdaderos cuarteles están en el ciberespacio, en una distribuida comunidad mundial de desarrolladores que construyen a partir del trabajo de otros, no solo compartiendo ideas sino el código fuente que implementa dichas ideas.

Estimar el número de usuarios de Linux en el mundo es tarea difícil, RedHat software hizo una investigación [30] en la cual analiza los problemas asociados a contar el número de usuarios de Linux, y en general de cualquier programa de software libre. Entre las dificultades para contabilizar el número de usuarios se cuenta con:

- El software se distribuye libremente. Cualquier persona puede hacer una copia de él, sin tener que comprarlo con nadie
- Los datos recolectados de encuestas son de carácter parcial.

La estimación de Red Hat software es de un mínimo de 5 millones de usuarios y un máximo de 10 y medio millones de usuarios de Linux en Marzo de 1998.

El mercado de UNIX de 1998 se estima en 20 billones de dólares [37], Dataquest estima que la venta de hardware para servidores Unix llegará a ser de 22.6 billones de dólares, casi el doble que los 12.8 billones de dólares generados por ventas de NT. El mismo reporte afirma que el 49% del software de los proveedores de Internet es Windows NT, mientras que el 51% restante se reparte entre los diversos Unix: Linux 27%, Solaris 15%, System V 14%, BSDI, 12%, HPUX 6% .

B.5.4. El predominio de Microsoft

Microsoft, al ser el principal productor de software comercial, y participar en casi todos los mercados de software, es un caso especial, y en la cultura del software libre, se ve como el "enemigo" a vencer.

Caldera, una empresa que vende una distribución de Linux con valor agregado (varias aplicaciones comerciales, soporte técnico, etc.) denuncia varias prácticas monopólicas de Microsoft en contra de las empresas de software libre. [36] El CEO de Caldera, Bryan Sparks afirma:

> Microsoft ha ido y amenazado con quitarles su licencia – Y me consta que sucedió (...) Oyes que los clientes escogen a Microsoft; y después oyes (que Microsoft) está amenazando con quitarles su licencia si alguien trata de ofrecer algo diferente".

Rodger [36] afirma que al tener mas del 5% del mercado, Linux comienza a ser una amenaza a tomarse en cuenta para las estrategias de Microsoft.

Personas de la empresa Dell Computer Inc (Dell), una de las mas grandes armadoras de computadoras en Estados Unidos afirman que no pudieron aceptar una propuesta de Caldera debido a presiones de Microsoft.

B.5.5. La guerra UNIX - Windows NT

La mayoría del software libre corre bajo UNIX, un sistema operativo muy robusto y antiguo, que tiene muchas variantes, incluyendo los ya mencionados Unix es libres como Linux, FreeBSD, HURD, etc. y a los UNIXes comerciales: IRIX, AIX, HP-UX, etc.
El mercado de UNIX es el mercado de los grandes servidores corporativos, los servidores de Internet y en general, en los lugares en los que se requiere cómputo serio y altamente confiable.
Microsoft intenta entrar a ese mercado con su producto Windows NT. Ya desde 1993, se vio a la fragmentación del mundo UNIX como un problema grave, e inclusive se propuso la creación de un sistema operativo UNIX unificado y libre [20].
Microsoft promocionó a NT en Intel como un producto equivalente y superior a la máquinas UNIX/RISC. Si eso no fue un "error embarazoso", debió haberlo sido. [25].
Un ejemplo de esta campaña es lo que menciona el CEO de Microsoft, Bill Gates en 1996:

> Compara el rendimiento: Compra la Sun más cara y compara su rendimiento como servidor de Web con respecto a una máquina con Windows NT. No andemos con chistes: Los procesadores Pentium Pro tienen mayor rendimiento que los de la comunidad RISC. No estoy hablando de relación costo/efectividad, sino de efectividad absoluta.

Cualquier persona que conozca de este tipo de equipos sabe que Bill Gates se quemó al hacer esta declaración, de hecho, la revista *InfoWorld* demostró que una Dual Pentium Pro corriendo Windows NT se arrastraba en comparación con una Sun SPARC de un solo procesador.

Como la red Internet está basada en Unix, el sistema operativo tiene un retorno exitoso. El hecho de que Unix haya entrado en el mercado de las computadoras Intel con exponentes como Linux o FreeBSD ha hecho que Unix crezca de manera apabullante.

De hecho, en muchos lugares del mundo se reemplazan servidores Windows NT por Linux con SAMBA, ambos software libre, debido a razones de costo y efectividad. SAMBA es el servidor de archivos de red de software libre compatible con los protocolos de Microsoft.

De hecho Linus Thorvalds, el autor de Linux [27]afirma que:

> El gerente corporativo de una compañía de Tecnología de Internet (Internet Technology, IT) se da cuenta que la pequeña computadora en la esquina,

> que es el servidor de Web del departamento, ha estado corriendo Linux por un año y medio. Una reacción normal es escalarlo inmediatamente a NT, pero lo que sucede es que la gente regresa a Linux por que el rendimiento bajó inmediatamente. Por lo que Linux se mueve en la lista de lo que no esta aprobado oficialmente. Pero no mucha gente quiere salir del armario y decir que está usando Linux. La NASA y las Universidades abiertamente soportan a Linux. Se que Linux se utiliza en Boeing, pero no puedo decirte ahora una página de Web que lo diga.

B.5.6. Personas independientes

Larry Wall [29] afirma que:

> Estoy esperanzado que muchas empresas (e individuos) se den cuenta que es posible hacer dinero del (software libre). De hecho, yo mismo he tenido un nivel de vida respetable gracias a él.

Respondiendo a cómo se hace dinero con el Freeware, Larry contesta:

> Dotando de servicios alrededor de éste – eso puede ser desde escribir libros acerca de éste, como yo he hecho la mayoría de mi dinero – o dando soporte.

De hecho, parte de la reputación que obtienen las personas exitosas en el software libre, se debe directamente a la fama de lo que han hecho. Seguramente mucha gente reconoce a Miguel de Icaza, desarrollador de Software Libre mexicano diciendo algo como "Sí, claro Miguel de Icaza, el de Midnight Commander, Linux/SPARC y GNOME". Donde la gente reconoce el hecho de que él ha trabajado fuertemente en los tres proyectos. Larry Wall dice algo al respecto:

> (La gente) toma su valor por el estatus dentro de la comunidad. (Alguien) podría decir, "Bueno, yo ayudé a la comunidad de Perl en esto o en lo otro ", lo que es una buena cosa en tu currículum.

B.6. Empresas relacionadas con el software libre en México

En la página del grupo de usuarios de Linux en México[33], en Mayo de 98 se encontraron 4 empresas y 11 personas físicas que ofrecen sus servicios como consultores de Linux. Una empresa que ofrece bastantes servicios y que ha desarrollado varios proyectos de software comercial y libre es Matías Software Group [32].
El grupo de usuarios cuenta con listas de correo en las cuales se proporciona soporte

a los nuevos usuarios del sistema, así como a usuarios expertos. Además, se imprimen discos compactos con el sistema operativo Linux, que se venden a costo de recuperación.
El grupo de usuarios es una estructura horizontal, en la cual "todo se vale". El Grupo organiza reuniones en las cuales discute los asuntos relevantes y divulga noticias relacionadas con el software libre.

B.7. Relaciones públicas y publicidad

Debido a que el software libre no tiene apoyo promocional dirigido, es difícil pensar en ver anuncios de televisión anunciando a Linux o a Apache en algún evento deportivo. Sin embargo, la promoción de los mismos usuarios, al apropiarse de los programas de software libre es impresionante. De hecho, la gente por sí misma puede ser un factor muy importante para obtener ventajas competitivas [19].
La comunidad de usuarios de Linux ganó el premio al "Mejor Soporte Técnico de 1997" [28].

La publicidad que ha recibido la comunidad de software libre a partir de la liberación del código fuente de Netscape y el anuncio de Corel de soportar una computadora basada enteramente en Linux [6] ha hecho que el mundo corporativo se entere cada vez más de las noticias del mundo del software libre. De hecho, Corel ha anunciado que va a portar todas sus aplicaciones a este sistema operativo.

Corel creó una computadora llamada NetWinder, que está basada en la distribución Red Hat de Linux, y en el procesador StrongArm de Digital Inc.

Varias revistas de alta circulación mencionan a Linux constantemente, por lo que la comunidad de usuarios de cómputo se ha ido enterando del panorama poco a poco.

Eric S. Raymond, autor de "The Cathedral and the Bazaar" afirma que:

> Corel, una compañía corporativa de primera línea con ninguna atadura a la comunidad de "Open Source" ha aceptado completamente la lógica de ésta.

En general, creemos que el principal problema del software libre sigue siendo el de la difusión y publicidad. Gente como Richard Stallman afirma que la publicidad del software libre tiene que ser un asunto de "persona a persona", pero para nuestro gusto eso es muy puritano. Debe de existir cobertura de los medios de comunicación.

Cuando Miguel de Icaza y "La Morsa" aparecerion en Punto por Punto, mucha gente se suscribió a la lista de usuarios de Linux en México.

B.8. Algunas reflexiones para el futuro

Consideramos que estos son momentos coyunturales para el software libre. El retraso del lanzamiento de Windows 98, el alto grado de avance de los proyectos como GNOME y KDE, y el gran interés corporativo en el Software Libre en el último año, hace posible que tal vez en un futuro no muy lejano (uno o dos años), el software libre comience a penetrar de manera peligrosa en el mercado de computadoras personales. Expandiendo así su espectro de usuarios potenciales. El reto está ahí para nuestra comunidad, que debe de responder ágilmente a las nuevas demandas que estos mercados crearán.

El sistema operativo Linux y las aplicaciones de Software Libre no deben caer en los errores que cayeron las compañías comerciales. No deben de existir fragmentaciones ni escisiones importantes, pero a su vez se debe de mantener un equilibrio competitivo. La coexistencia de software comercial y software libre será más común, y eso no debe de extrañarnos, debido a que las grandes compañías seguramente portaran sus aplicaciones a Linux.

Bibliografía

[1] Norin Ludvig A. 1998. Open Source Software Development Methodology [Tesis de Maestría]] http://www.ludd.luth.se/users/no/mssc_abstract.html

[2] Yahoo. 1998. Pairing Of Red Hat Linux OS And NASA Technology Harnesses PC Power For End Users http://biz.yahoo.com/prnews/980512/nc_red_hat_1.html

[3] Zdnet. 1998. A look at life beyond Microsoft http://www.zdnet.com/zdnn/content/inwo/0508/315048.html

[4] Shock. 1998 Slashdot Editorial. Open Source's Achiles Heel. http://slashdot.org/features/9804081134201.shtml

[5] Oram Andy. 1998. Linux Ports: One body running on many legs. O' Reilly Linux News. http://linux.oreilly.com/news/oram_0498.html

[6] Statz Michael. 1998. Wired News. Corel embraces Open Source. http://www.wired.com/news/news/email/member/business/story/12187.html

[7] Raymond Eric. S. 1997. The Cathedral and the Bazaar. http://sagan.earthspace.net/ esr/writings/cathedral-bazaar/

[8] Oster, Sharon M. 1994. Modern Competitive Analysis. 2nd Edition. Oxford University Press.

[9] Forge, Kevin. 1998. The Crusher OS. Slashdot Editorial. http://slashdot.org/features/9804060950245.shtml

[10] Varias noticias. 1998 Slashdot. News for Nerds. Stuff that Matters.

[11] Información sobre el proyecto Mozilla: El Código fuente de Netscape. http://www.mozilla.org

[12] InfoBeads. 1998. ISP Hardware and Software Purchases. http://www.ci.infobeads.com/INSIDER/PAGES/TOPICS/INTERNET /ISP_HW_0505/Default.asp

[13] Lemon, Summer. 1998. Computer World Hong Kong. Linux finding it's place in Hong Kong.http://www.cw.com.hk/Analysis/a980429001.htm

[14] Elgin, Ben. 1998. Netscape Bets on Linux. ZDNet http://www.zdnet.com/sr/breaking/980504.html

[15] McVoy Larry. 1993. The Sourceware Operating System Proposal. http://www.redhat.com/redhat/mcvoy.html

[16] O Reilly News. 1998.Open Source Software–The Real Grassroots Movement! http://www.oreilly.com/news/opensource_0498.html

[17] Dougherty, Dale. 1998. What's new in Free and Open Source Software?. Web Review. http://webreview.com/wr/pub/freeware/whatsnew.html

[18] Dougherty, Dale. 1998. The Origins of Free and Open Source Software. Web Review. http://webreview.com/wr/pub/freeware/origins.html

[19] Pfeffer, Jeffrey. 1994. Competitive Advantage Through People. Harvard Business School Press.

[20] Wall, Larry. 1998. Web Review. Evolutionary Changes in Freeware http://webreview.com/wr/pub/98/03/13/feature/lwall-mozilla.html

[21] Rodger, Will. 1998. ZDNet. Caldera CEO says MS threatening PC makers. http://www.zdnet.com/zdnn/content/inwo/0427/310649.html

[22] O'Reilly, Tim. 1998. Measuring the Impact of Free Software. WebReview. http://webreview.com/wr/pub/freeware/oreilly.html

[23] Zdnet 1998. Editorial: Of Stone Soup and Mozzilla. http://www.zdnet.com/pcweek/opinion/0413/13edit.html

[24] Sengan. 1998. Free and Commercial Software: Feature. Slashdot. http://slashdot.org/articles/9845152618.shtml

[25] Petreley, Nicholas. 1998. The new Unix alters NT's Orbit. NC World. http://www.ncworldmag.com/ncworld/ncw-04-1998/ncw-04-nextten.html

[26] RedHat Software. http://www.redhat.com

[27] Vizard Michael.1998. Linus Thorvalds talks economics and operating systems. InfoWorld Electric http://www.infoworld.com/cgi-bin/displayStory.pl?/interviews/980409torvalds.htm

[28] Lash Alex. 1998. Netscape: Linux a top priority. CNET NEWS http://www.news.com/News/Item/0,4,20863,00.html

[29] Sims David, 1998. Q&A with Larry Wall, Creator of Perl TechWeb http://www.techweb.com/wire/story/TWB19980408S0020

[30] Young, Robert F. 1998. Sizing the Linux Market, Second Edition. RedHat white papers, http://www.redhat.com

[31] Young, Robert F. 1998. RedHat explains it all. RedHat white papers, http://www.redhat.com

[32] Matías Software Group. http://www.msg.com

[33] Grupo de Usuarios de Linux en México, http://www.linux.org.mx

[34] GNU´s not UNIX. http://www.gnu.org

[35] Free Software Foundation, http://www.fsf.org

[36] Rodger Will. 1998. Caldera CEO says MS threatens PC Makers. Inter@ctive Week Online. http://www.caldera.com, http://www.zdnet.net

[37] Levin Rich y Garvey Martin. 1998. Unix: The Next Generation. TechWeb News. http://www.techweb.com

[38] Prahalad C.K., y Hamel, G. 1990. The Core Competence of the Corporation. Harvard Business Review. Mayo-Junio 1998-

C. La Licencia Pública General

La licencia pública general (*General Public License*) es parte fundamental de MOISS pues es la que rige sobre todo su código fuente.

A continuación se hace una copia del documento que acompaña la distribución de MOISS bajo el nombre de "COPYING".

GNU General Public License

C.0.1. Table of Contents

GNU GENERAL PUBLIC LICENSE

C.0.2. GNU GENERAL PUBLIC LICENSE

Version 2, June 1991

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change free software — to make sure the software is free for all its

users. This General Public License applies to most of the Free Software Foundation's software and to any other program whose authors commit to using it. (Some other Free Software Foundation software is covered by the GNU Library General Public License instead.) You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the software, or if you modify it.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

We protect your rights with two steps: (1) copyright the software, and (2) offer you this license which gives you legal permission to copy, distribute and/or modify the software.

Also, for each author's protection and ours, we want to make certain that everyone understands that there is no warranty for this free software. If the software is modified by someone else and passed on, we want its recipients to know that what they have is not the original, so that any problems introduced by others will not reflect on the original authors' reputations.

Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a free program will individually obtain patent licenses, in effect making the program proprietary. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all.

The precise terms and conditions for copying, distribution and modification follow.

TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

- This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The "Program", below, refers to any such program or work, and a "work based on the Program" means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into

another language. (Hereinafter, translation is included without limitation in the term "modification".) Each licensee is addressed as "you".

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.

- You may copy and distribute verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program.

  You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

- You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

  - You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.
  - You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.
  - If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute the program under these conditions, and telling the user how to view a copy of this License. (Exception: if the Program itself is interactive but does not normally print such an announcement, your work based on the Program is not required to print an announcement.) These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to

those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.

In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

- You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:

  - Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

  - Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution, a complete machine-readable copy of the corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

  - Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b above.) The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place counts as distribution of the source code, even though third parties are not compelled to copy the source along with the object code.

- You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

- You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.

- Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.

- If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program.

  If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system, which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

- If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

- The Free Software Foundation may publish revised and/or new versions of the General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

  Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.

- If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

  NO WARRANTY

- BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE

THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

- IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

One line to give the program's name and an idea of what it does. Copyright (C) 19yy name of author

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 59 Temple Place — Suite 330, Boston, MA 02111-1307, USA.

Also add information on how to contact you by electronic and paper mail.

If the program is interactive, make it output a short notice like this when it starts in an interactive mode:

Gnomovision version 69, Copyright (C) 19yy name of author Gnomovision comes with ABSOLUTELY NO WARRANTY; for details type "show w". This is free software, and you are welcome to redistribute it under certain conditions; type "show c" for details.

The hypothetical commands "show w" and "show c" should show the appropriate parts of the General Public License. Of course, the commands you use may be called something other than "show w" and "show c"; they could even be mouse-clicks or menu items — whatever suits your program.

You should also get your employer (if you work as a programmer) or your school, if any, to sign a "copyright disclaimer" for the program, if necessary. Here is a sample; alter the names:

Yoyodyne, Inc., hereby disclaims all copyright interest in the program 'Gnomovision' (which makes passes at compilers) written by James Hacker.

Signature of Ty Coon, 1 April 1989 Ty Coon, President of Vice

This General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Library General Public License instead of this License.

FSF & GNU inquiries & questions to gnu@gnu.org Other ways to contact the FSF.

Comments on these web pages to webmasters@www.gnu.org, send other questions to gnu@gnu.org.

Copyright notice above. Free Software Foundation, Inc., 59 Temple Place — Suite 330, Boston, MA 02111, USA

Updated: 16 Feb 1998 tower

D. ¿Dónde obtener los programas?

La página de la World Wide Web (WWW) donde se encuentra el programa MOISS es:

`http://moiss.pquim.unam.mx/moiss/`

Ahi se encuentra la información general sobre el programa, así como el sitio `ftp` donde se encuentra el programa principal `moiss-version.tar.gz` (donde `version` es la versión del programa). Actualmente, la versión es 0.5, pero ésta irá evolucionando hasta llegar a la serie `1.X` estable, y continuando con la serie `2.X` de desarrollo. Este sitio es el lugar de referencia en el que se encontrarán las subsecuentes versiones de nuestro software.

Además ahí se encuentra a su vez un archivo llamado `demos-version.tar.gz` donde se incluye el conjunto de programas educativos escritos en `python`[75] y `Java` que se usaron para obtener los resultados del problema de Buffon y los círculos en n dimensiones (capítulo 3).

Bibliografía

[1] *Intel Museum Homepage.* Página de Web, 1995. http://www.intel.com/intel/museum/25anniv/Hof/moore.htm.

[2] S. A. Alexander, R. L. Coldwell, G. Aissing, and A. J. Thakkar, *Calculation of Atomic and Molecular Properties Using Variational Monte Carlo Methods*, Int. J. Q. Chem., 26 (1992), pp. 213–27.

[3] J. B. Anderson, *Quantum chemistry by random walk:* $H^2(P)$, $H_3^+(D_{3h})1_1^A$, $H_2(^3\Sigma_u^+)$, $H_4(^1\Sigma_g^+)$, $Be(1^S)$, J. Chem. Phys. , 65 (176), pp. 4121–4127.

[4] ——, *Quantum chemistry by random walk: Higher accuracy*, J. Chem. Phys. , 73 (1980), pp. 3897–3899.

[5] D. M. Arnow, M. H. Kalos, M. A. Lee, and K. E. Schmidt, *Green's function Monte Carlo for few fermion problems*, J. Chem. Phys. , 77 (1982), pp. 5562–5572.

[6] V. Autores, *XFree86*, 1999. http://www.xfree86.org.

[7] Z. Bacic, M. Kennedy-Mandziuk, J. W. Moskowitz, and K. E. Schmidt, *He_2Cl_2 and He_3Cl_3 van der Waals clusters: A quantum Monte Carlo study*, J. Chem. Phys. , 97 (1992), pp. 6472–6480.

[8] R.Ñ. Barnett, P. J. Reynolds, and W. A. L. Jr., *Monte Carlo Algorithms for Expectation Values of Coordinate Operators*, J. Chem. Phys. , 96 (1991), pp. 258–76.

[9] G. Barton, *Elements of Green's Functions and Propagation, Potentials, Diffusion and Waves*, Oxford University Press, 1991.

[10] L. Bertini, D. Bressanini, M. Mella, and G. Morosi, *Linear expansions of correlated functions: a Variational Monte Carlo case study.*, International Journal of Quantum Chemistry, (1999). In press.

[11] K. Binder and C. C., eds., *Path integral Monte Carlo methods for fermions*, vol. 49 of Monte Carlo and Molecular Dynamics of Condensed Matter Systems, SIF, Bologna, 1996.

[12] D. Blume, M. Mladenovic, M. Lewerenz, and K. B. Whaley, *Excited states of van der Waals clustres by projector Monte Carlo, with application to excitations of molecules in small* ^4He_n, J. Chem. Phys. , 110 (1999), pp. 5789–5805.

[13] D. Bohm, *Quantum theory*, Dover, 1979.

[14] D. Bressanini, M. Mella, and G. Morosi, *Many-electron correlated exponential wavefunctions. A quantum Monte Carlo aplication to* H_2 *and* He_2^+, Chemical Physics Letters, 240 (1995), pp. 566–570.

[15] ——, *Stability of four-unit-charge systems: A quantum Monte Carlo study*, Physical Review A, 55 (1997), pp. 200–205.

[16] — —, *Positron chemistry by quantum Monte Carlo II. Ground-state of positron-polar molecule complexes*, J. Chem. Phys. , 109 (1998), pp. 1716–1719.

[17] C. F. Bunge, *ATMOL*. sitio ftp. 132.248.7.41 /pub/programs/atmol.

[18] C. F. Bunge, J. A. Barrientos, and A. V. Bunge, Atomic Data and Nuclear Data Tables, 53 (1993), pp. 113–162.

[19] J. C. S. Johnson and L. G. Pedersen, *Problems and Solutions in Quantum Chemistry and Physics*, Dover, primera ed., 1986.

[20] M. Caffarel and P. Claverie, *Development of a pure diffusion quantum Monte Carlo metjhod using a full generalized Feynman-Kac formula. I. Formalism*, J. Chem. Phys. , 88 (1988), pp. 1088–1099.

[21] ——, *Development of a pure diffusion quantum Monte Carlo metjhod using a full generalized Feynman-Kac formula. II. Applications to simple systems*, J. Chem. Phys. , 88 (1988), pp. 1100–1110.

[22] M. Caffarel, X. Krodikis, and C. Mijoule, *On the Nonconservation of the Number of Nodal Cells of Eigenfunctions*, Europhysics Letters, 20 (1992), pp. 581–588.

[23] M. Caffarel, M. Rerat, and C. Pouchan, *Evaluating dynamic multipole polarizabilities and van der Waals dispersion of two-electron systems with a quantum Monte Carlo calculation: A comparison with some abinitio calculations*, Physical Review A, 47 (1993), pp. 3704–3717.

[24] D. M. Ceperley, J. Stat. Phys., 43 (1986), p. 815.

[25] ——, *Fermion Nodes*, Journal of Statistical Physics, 63 (1991), pp. 1237–1267.

[26] ——, *Path Integral Monte Carlo in the theory of Condensed Matter Helium*, Rev. of Mod. Phys., 67 (1995), pp. 279–356.

[27] D. M. Ceperley and L. Mitas, *Quantum Monte Carlo Methods in Chemistry*, in New Methods in Computational Quantum Mechanics, I. Prigogine and S. A. Rice, eds., vol. XCIII of Advances in Chemical Physics, 1996.

[28] P. F. Dubois, *Ten Good Practices in Scientific Programming*, Computers in Science and Engineering, (1999), pp. 7–11.

[29] C. Edwards and D. Penney, *Ecuaciones Diferenciales Elementales*, Pentice Hall, 1993.

[30] B. Eraker, *MCMC analysis of diffusion models with application to finance.* Norwegian School of Economics and Business Administration. Preprint en http://www.stats.bris.ac.uk/MCMC/pages/listam.html, Marzo 1998.

[31] G. S. Fishman, *Monte Carlo: Concepts, Algorithms and Applications*, Springer-Verlag, 1a ed. ed., 1996.

[32] W. Foulkes, M.Ñekovee, R. Gaudoin, M. Stedman, R.Ñeeds, R. Hood, G. R. amd M.D. Towler, P. Kent, Y. Lee, W. K. Leung., A. R. Porter, and S. J. Breuer, *Quantum Monte Carlo Simulations of Real Solids*, in High Performance Computing, R. J. Allan, M. F. Guest, A. D. Simpson, and D. S. Henty, eds., Plenum Press, 1998.

[33] R. Frazer, W. Duncan, and A. Collar, *Elementary Matrices*, Cambridge University Press, 1946.

[34] M. Galassi, J. Davies, J. Theiler, B. Gough, R. Priedhorsky, G. Jungman, and M. Booth, *GNU Scientific Library*, 1999. http://sourceware.cygnus.com/gsl/.

[35] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, *Design Patterns: Elements of Reusable Object-Oriented Software*, Addison-Wesley, primera ed., 1994.

[36] H. Gould and J. Tobochnik, *An Introduction to Computer Simulation Methods*, Addison-Wesley, segunda ed., 1996.

[37] C. W. Greef and W. A. L. Jr., *Electronic states of Al and Al_2 using quantum Monte Carlo with an effective core potential*, J. Chem. Phys. , 104 (1996), pp. 1973–1978.

[38] ——, *Quantum Monte Carlo binding energies for silicon hydrides*, J. Chem. Phys. , 106 (1997), pp. 6412–6417.

[39] ——, *A soft Hartree Fock pseudopotential for carbon with application to quantum Monte Carlo*, J. Chem. Phys. , 109 (1998), pp. 1607–1612.

[40] J. C. Grossman and L. Mitas, *Small Hidrocarbons, NCSA Condensed Matter Group Homepage*. Página de Web, 1998. http://www.ncsa.uiuc.edu/Apps/CMP/hydroc.html.

[41] E. H and W. Schattke, *Variational Quantum Monte Carlo Ground State of Lithium on a Slater orbital basis*, Phisica A, 216 (1995), p. 151.

[42] B. Hammond, W. A. Lester, and P. Reynolds, *Monte Carlo Methods in Ab Initio Quantum Chemistry*, World Scientific, 1994.

[43] B. L. Hammond, *Quantum MagiC: A Quantum Monte Carlo Program for the Electronic Structure of Atoms and Molecules. User's Guide*, v. 7.7 ed., 1996. http://www.cchem.berkeley.edu/walgrp/.

[44] H. Hollerstein, R. R. Marquardt, M. Quack, and M. A. Suhm, *Dipole moment function and equilibrium structure of methane in an analytical, anharmonic, nine-dimensional potential surface related to experimental rotational constants and transition moments by quantum Monte Carlo calculations*, J. Chem. Phys. , 101 (1994), pp. 3588–3602.

[45] C. K. Huang, C. J. Umrigar, and M. P. Nightingale, *Accuracy of Electronic Wave Functions in Quantum Monte Carlo: the Effect of High-Order Ocrrelations*. Página WWW., Mar 1997. http://xxx.lanl.gov/cond-mat/9703008/.

[46] M. D. Icaza, *GNU Object Modelling Envrinomnent*, 1999. http://www.gnome.org.

[47] C. Janseen and E. S. et al., *Massively Parallell Quantum Chemistry Program (MPQC)*, 1999. http://www.chem.limitpt.com/mpqc/.

[48] R. Jastrow, *Many-Body Problem with Strong Forces*, Physical Review, 98 (1955), pp. 1479–1484.

[49] O. Jitrik and C. F. Bunge, *Atomic configuration interaction and studies of He, Li, Be, and Ne ground states*, Physical Review A, 56 (1997), pp. 2614–2623.

[50] D. M. Jones and J. M. Goodfellow, *Parallelization Strategies for Molecular Simulation using the Monte Carlo Algorithm*, Journal of Computational Chemistry, 14 (1993), pp. 127–137.

[51] I. Joyner, *C++?? A critique of C++ and Programming and Language Trends of the 1990's.* http://www.progsoc.uts.edu.au/gĕldridg/cpp/cppcv3.html, 1996.

[52] F. B. Jr. and R. Fuller, *Mathematics of Clasical and Quantum Physics*, Dover, 1992.

[53] W. A. L. Jr. and H. B. L., *Quantum Monte Carlo for the electronic structure of atoms and molecules*, Annual Reviews in Physical Chemistry, 41 (1990), pp. 283–311.

[54] M. H. Kalos and P. A. Whitlock, *Monte Carlo Methods Volume 1: Basics*, Wiley, N.Y., 1986.

[55] B. W. Kernighan and R. D. M., *The C Programming Language*, Prentice Hall, 2nda. ed., 1998.

[56] M. V. R. Krishna and K. B. Whaley, *Wave functions of helium clusters*, J. Chem. Phys. , 93 (1990), pp. 6738–6751.

[57] I. Levine, *Quantum Chemistry*, Prentice Hall, 1991.

[58] J. P. Lowe, *Quantum Chemistry*, Academic Press, 1993.

[59] A. Luchow, J. B. Anderson, and D. Feller, *Improved estimates of the total correlation energy in the ground state of the water molecule*, J. Chem. Phys. , 106 (1997), pp. 7706–7709.

[60] N. Makri, *Path Integral Methods*, in Enciclopedia of Computational Chemistry, P. von Rague Schleyer, ed., Wiley, 1998.

[61] F. Mentch and J. B. Anderson, *Quantum chemistry by random walk: Linear H_3*, J. Chem. Phys. , 80 (1984), pp. 2675–2680.

[62] N. Metropolis, A. W. Rosenbluth, M.N̄. Rosenbluth, N. M. Teller, and E. Teller, *Equations of State Calculations by Fast Computing Machine*, J. Chem. Phys. , 21 (1953), pp. 1087–1091.

[63] N. Metropolis and S. Ulam, *The Monte Carlo Method*, J. Amer. Statist. Assoc., 44 (1949), p. 335.

[64] L. Mitas, *Quantum Monte Carlo for electronic structure of solids*, in Electronic Properties of Solids Using Cluster Methods, T. A. Kaplan and S. D. Manhanti, eds., Plenum, N.Y., 1995.

[65] L. Mitas, *Electronic structure by Quantum Monte Carlo: atoms, molecules and solids*, Computer Physics Communications, 97 (1996), p. 107.

[66] L. Mitas and R. M. Martin, *Quantum Monte Carlo of nitrogen: atom, dimer, atomic and molecular solids*, Physical Review Letters, 72 (1994), p. 2438.

[67] J. W. Moskowitz, K. E. Schmidt, M. A. Lee, and M. H. Kalos, *A new look at correaltion energy in atomic and molecular systems. II. The application of Green's function Monte Carlo method to LiH*, J. Chem. Phys. , 77 (1982), pp. 349–355.

[68] M. P. Nightingale and C. J. Umrigar, *Monte Carlo Eigenvalue Methods in Quantum Mechanics and Statistical Mechanics*, in Monte Carlo Methods in Chemistry, D. M. Ferguson, J. I. Siepmann, and D. G. Truhlar, eds., vol. 105 of Advances in Chemical Physics, Wiley, 1998, editores de la serie: i. prigogine and stuart a. 4.

[69] ——, *Quantum Monte Carlo Methods in Physics and Chemistry*, Kluwer Academic Publishers, 1999.

[70] Y.Ñonomura, *New Quantum Monte Carlo Approach to Ground-State Phase Transitions in Quantum Spin Systems*, Journal of the Physical Society of Japan, 67 (1998), pp. 5–7.

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

[72] L. Pauling and J. E. B. Wilson, *Introuduction to Quantum Mechanics with applications to Chemistry*, Dover, 1985.

[73] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes in C: The Art of Scientific Computing*, Cambridge University Press, 1993.

[74] P. J. Reynolds, D. M. Ceperley, B. J. Alder, and W. A. L. Jr., *Fixed-Node Quantum Monte Carlo fror Molecules*, Journal of Chemical Physics, 77 (1982), pp. 5593–603.

[75] G. V. Rossum, *Python*, 1999. http://www.python.org.

[76] D. L. L. S. W. Rick and J. D. Doll, *A variational Monte Carlo study of argon, neon and helium clusters*, J. Chem. Phys. , 95 (1991), pp. 3506–3519.

[77] C. D. Sherrill, *Computational Scaling of the Configuration Interaction Method with System Size.* Página de Web, Septiembre 1996. http://zopyros.ccqc.uga.edu/lec_top/ciscale/ciscale.html.

[78] D. W. Skinner, J. W. Moskowitz, K. E. Schmidt, M. A. Lee, and P. A. Whitlock, *The solution of the Schrödinger equation in imaginary time by Green's function Monte Carlo. The rigorous sampling of the attractive Coloumb singularity*, J. Chem. Phys. , 83 (1985), pp. 4668–4672.

[79] D. L. Smith, *Probability,Statistics, and Data Uncertainties in Nuclear Science and Technology*, American Nuclear Society, 1991.

[80] A. D. Sokal, *Monte Carlo Methods in Statistical Mechanics: Foundations and New Algorithms.* Notas de clase de la escual de verano de Cargese "Functional Integration: Basics and Applications".

[81] A. J. Stace, *A Monte Carlo model for simulating the behaviour of a quantum harmonic oscillator embedded in a classical cluster, liquid or solid*, Chemical Physics Letters, 232 (1995), pp. 283–288.

[82] R. Stallman, *The GNU Project and the Free Software Foundation*, 1999. http://www.gnu.org.

[83] M. L. Stedman and W. M. C. Fowlkes, *Talus: A quantum Monte Carlo Modelling Suite*, Computer Physics Communications, 113 (1998), pp. 180–198.

[84] A. Szabo and N. Ostlund, *Modern Quantum Chemistry. Introduction to Advanced Electronic Structure Theory*, McGraw-Hill, 1982.

[85] O. Taylor, *GTK: The GIMP Toolkit*, 1999. http://www.gtk.org.

[86] L. Thorvalds, *Linux*, 1999. http://www.linux.org.mx.

[87] C. Umrigar, M.Ñightingale, and K. J. Runge, *A Diffusion Monte Carlo Algorithm with Very Small Time-step Errors*, J. Chem. Phys. , 99 (1993), pp. 2865–90.

[88] C. J. Umrigar, K. G. Wilson, and J. W. Wilkins, *Optimized Trial Wave Functions for Quantum Monte Carlo Calculations*, Physical Review Letters, 60 (1988), pp. 1719–1722.

[89] R. O. Weht, D. Kohanoff, D. Estrin, and C. Chakravarty, *An ab-initio path integral Monte Carlo simulation method for molecules and clusters: Application to Li_4 and Li_5^+*, J. Chem. Phys. , 108 (1998), pp. 8848–8857.

[90] A. Williamson, *Quantum Monte Carlo Calculations of Electronic Excitations*, PhD thesis, Robinson College, Cambridge, 1996.

Índice de Materias

Esta tesis fue desarrollada por completo con herramientas de software libre.

- LaTeX
- Linux
- GNOME
- Gnuplot y Xmgr
- El estilo KOMA-SCRIPT

excepto

- Esta hoja que fue creada con el elegante programa de MICROSOFT
- Algunas ilustraciones y cálculos realizados con Maple

17 de AGOSTO de 1998 – 2 de junio de 1999