UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO
PROGRAMA DE MAESTRÍA Y DOCTORADO
EN CIENCIAS QUÍMICAS

ALGORITMO OPTIMIZADO PARA UNIDADES DE PROCESAMIENTO GRÁFICO EN LA EVALUACIÓN DE ENERGÍA MP2 EN LA APROXIMACIÓN DE LA RESOLUCIÓN DE LA IDENTIDAD

TESIS
QUE PARA OPTAR POR EL GRADO DE
MAESTRO EN CIENCIAS

PRESENTA:
Q. LUIS ÁNGEL MARTÍNEZ MARTÍNEZ

TUTOR:
DR. CARLOS AMADOR BEDOLLA
FACULTAD DE QUÍMICA, UNAM

MÉXICO , D.F., Junio 2015

Agradecimientos

Aprovecho este espacio para expresar mis más sinceros agradecimientos:

- Al Dr. Carlos Amador Bedolla, por darme la invaluable oportunidad de formar parte de su grupo de trabajo, y por su apoyo y consejos para el desarrollo de este trabajo.
- A los miembros del jurado de mi examen de grado, Dr. Luis Emilio Orgaz Baqué, Dr. Alberto Marcial Vela Amieva, Dr. Francisco Miguel Castro Martínez, Dr. Jorge Martín del Campo Ramírez y Dr. Tomás Rocha Rinza, por sus valiosas sugerencias para el enriquecimiento de esta tesis.
- Al Consejo Nacional de Ciencia y Tecnología, CONACyT, por el apoyo económico brindado a través de la beca 293319, y a través del proyecto No. 129343, "Escuela de Verano en Programación Paralela y GPUs" .
- A la Dirección General de Cómputo y de Tecnologías de Información y Comunicación, DGTIC, por el acceso a los recursos de cómputo y la capacitación brindada en el manejo de éstos, la cual fue fundamental para el desarrollo del proyecto aquí expuesto.
- A mis amigos y compañeros de trabajo, de quienes tuve su apoyo y pude compartir conocimiento: Martha Flores, Gerardo Álvarez, Nancy Barrueta, Rodrigo Cortés y Xiao Ming.
- A mis amigos Uzi García y Ulrich Briones, por la amistad brindada durante todos estos años. Sus consejos y apoyo siempre han sido muy importantes para mí.
- A mi familia: mis padres, Araceli Martínez y Ángel Martínez, y mi hermano, David Martínez. Por que a ellos les debo todo lo que soy ahora.

Luis Ángel Martínez Martínez
México D.F., Junio de 2015

A mis padres.

Este trabajo fue desarrollado en el Departamento de Física y Química Teórica de la Facultad de Química de la Universidad Nacional Autónoma de México. Los resultados derivados de este proyecto fueron presentados en:

- Reunión Mexicana de Fisicoquímica Teórica. Presentación oral. Morelia, Michoacán. 5-8 de Noviembre de 2014.
- International Conference on Supercomputing in Mexico. ISUM 2015. Presentación oral. Ciudad de México. 9-13 de Marzo de 2015.
- QuimiUNAM 2015. Presentación de Póster. Ciudad de México. 22-24 de Abril de 2015.

Índice general

Algoritmo Optimizado para Unidades de Procesamiento Gráfico en la Evaluación de Energía MP2 en la Aproximación de la Resolución de la Identidad

por

Luis Ángel Martínez Martínez

Resumen

Actualmente, la habilidad para predecir con exactitud estructuras de equilibrio y diferencias de energías moleculares es uno de los principales incentivos en el uso de paquetes de química computacional.

Dicha habilidad reside en el creciente poder de cómputo disponible y la existencia de un sinnúmero de métodos de estructura electrónica. Entre los métodos *ab initio* existentes, una jerarquía basada en la exactitud de éstos ha sido desarrollada. Esta jerarquía principia con la aproximación de campo medio de Hartree-Fock, y posteriormente, sigue el tratamiento más simple para considerar correlación electrónica: la teoría de perturbaciones a segundo orden de Møller-Plesset. El esfuerzo computacional de este último escala como $O(M^5)$, en donde M hace referencia al número de funciones base empleado en el cálculo. Como consecuencia, su uso está restringido a sistemas con tamaño pequeño o modesto.

La teoría de perturbaciones a segundo orden con escalamiento de espínes opuestos (SOS-MP2, por sus siglas en inglés), constituye una variante simplificada de MP2 en la que el componente $\alpha - \beta$ de la energía MP2 es calculado y posteriormente escalado por un factor empírico, el cual produce resultados estadísticamente mejorados en comparación con la teoría MP2 convencional. Adicionalmente, la introducción de la Resolución de la Identidad, y una transformada de Laplace, se traduce en un método económico que no involucra pasos en el algoritmo de quinto orden en el tamaño del sistema. Por otra parte, el modelo de cómputo heterogéneo ha sido ampliamente usado entre la comunidad científica para acelerar sus códigos, siendo las Unidades de Procesamiento Gráfico de NVIDIA (GPUs, por sus siglas en inglés), las más conocidas. En esta tesis, se propone una serie de modificaciones en el algoritmo SOS-MP2 implementado en Q-Chem, que permiten la incorporación de GPUs en el cómputo de la evaluación de energía MP2. Es demostrado que estas modificaciones introducen una mejora en el rendimiento de casi tres veces en comparación con la versión serial, en cálculos de energía de correlación para alcanos lineales. Esto es logrado empleando una GPU Tesla M2090 y la Arquitectura de Dipositivo de Cómputo Unificado (CUDA, por sus siglas en inglés) de NVIDIA.

Optimized Algorithm for Graphical Processing Units in the Resolution-of-the-identity MP2 Energy Evaluation

by

Luis Ángel Martínez Martínez

Abstract

Nowadays, the ability to accurately predict the equilibrium structures and molecular energy differences is one of the most important incentives to use quantum chemistry computational packages. The aforementioned ability relies on the increasing computational power and the existence of numerous electronic structure levels of theory. Among the available *ab initio* methods, an accuracy-based hierarchy has been developed. This hierarchy begins with the mean-field Hartree-Fock approximation, and then follows the simplest approach to treat electronic correlation: the second-order Møller-Plesset Perturbation Theory (MP2). The computational effort of the latter scales as $O(M^5)$, where M denotes the number of basis functions employed in a calculation. As a consequence, its use is restricted to small or modest size systems.
The scaled opposite-spin second order Møller-Plesset theory (SOS-MP2), constitutes a simplified MP2 variant in which only the α-β component of MP2 energy is calculated and scaled by an empirical factor, which yields statistically improved energies and derivative properties over the conventional MP2 method. In addition, the introduction of the Resolution of the Identity (RI) approximation, and a Laplace transform results in an economical method without any fifth order computational steps. On the other hand, the heterogeneous computing model has been widely used among the scientific community to accelerate its codes, being the NVIDIA Graphics Proccessing Units (GPUs) the most known. In this work, we propose a series of modifications in the SOS-MP2 algorithm implemented in Q-Chem, which allows the incorporation of GPU computing in the energy evaluation. It is shown that those changes introduce an almost threefold improvement in performance in the correlation energy calculations for linear alkanes. This is achieved employing a NVIDIA Tesla M2090 GPU and the Compute Unified Device Architecture (CUDA) of NVIDIA.

Capítulo 1

Química, Mecánica Cuántica y Cómputo

La historia de la mecánica cuántica constituye uno de los cimientos que dieron base al desarrollo de la física y química moderna. Las contribuciones y descubrimientos de Gustav Kirchhoff[1], Ludwing Boltzmann[2], Max Planck[3], y Albert Einstein[4] [5], por citar sólo a algunos, dieron base para establecer una teoría central para la descripción del comportamiento de partículas de dimensiones ínfimas.

La introducción de un nuevo paradigma conceptual que asocia una naturaleza probabilística intrínseca en la descripción de fenómenos físicos a escala microscópica produjo diferentes interpretaciones entre la comunidad científica, algunas de ellas incluso de naturaleza filosófica.

Entre dichas interpretaciones, la más ampliamente adoptada es la de Copenhague, asociada con Bohr y seguidores, la cual establece que el proceso de medición de la posición de una partícula "forza" a la misma a "materializarse".

De ser esta interpretación cierta, esto implicaría ciertas peculiaridades al proceso de medición, lo que es motivo de debate entre la comunidad científica moderna. Éste y otros conceptos peculiares como la no-localidad y el principio de superposición, (este comúnmente ejemplificado en el experimento del gato de Schrödinger) aparentan ser de poca utilidad para la comunidad química, cuando en realidad, estos principios son comúnmente empleados para explicar fenómenos como la transferencia de energía en complejos moleculares[6] [7], o para describir la evolución

de paquetes de ondas[8] [9] [10], que juegan un papel importante en la espectroscopía con resolución temporal del orden de femtosegundos.

Dado que los sistemas moleculares se conforman de partículas bosónicas y/o fermiónicas que obedecen los postulados de la mecánica cuántica, la aplicación de estos principios a este tipo de sistemas con el objetivo de extraer información útil en la predicción de propiedades químicas, parece directa. Sin embargo, para sistemas más grandes que los monoelectrónicos, no es posible encontrar una solución exacta a las ecuaciones de la mecánica cuántica[11], y por lo tanto, para sistemas polielectrónicos, una aproximación numérica es deseable.

Desafortunadamente, los métodos numéricos disponibles en la actualidad requieren de una cantidad de recursos computacionales que escalan rápidamente con el tamaño del sistema en consideración.

Así, durante varios años, una comunidad híbrida compuesta por químicos y físicos ha lidiado con el desarrollo de aproximaciones que buscan reducir el esfuerzo computacional sin comprometer significativamente la precisión del cálculo de las principales propiedades de sistemas electrónicos complejos. Lo anterior requiere del desarrollo de algoritmos computacionales altamente eficientes que permitan el aprovechamiento máximo del *hardware* disponible en la época.

Por lo tanto, la triada química-física-cómputo constituye una intersección donde dichos campos están íntimamente relacionados.

Aunque esta tesis está mayoritariamente orientada al tercero, la teoría física subyacente es expuesta y de la misma forma su aplicación en el campo de la química.

Capítulo 2

Marco Teórico

El desarrollo de la teoría cuántica a principios del siglo XX fue incentivada en gran medida por el objetivo de comprender las propiedades de átomos y moléculas. Pronto, la ecuación de Schrödinger en conjunto con la interpretación probabilística de sus soluciones, se perfilaron como una herramienta poderosa para buscar respuestas a un sinnúmero de preguntas en física y química.

La descripción matemática de las líneas espectrales del átomo de hidrógeno es un ejemplo claro del éxito de la mecánica cuántica[12]. Sin embargo, al considerar el problema molecular, nos encontramos con una formulación complicada de muchos cuerpos, que involucra todas las coordenadas nucleares y electrónicas que constituyen la molécula. Su solución puede ser aproximada haciendo uso del hecho de que núcleos y electrones difieren en su masa en tres órdenes de magnitud, lo que permite tratar de forma independiente los grados de libertad electrónicos y nucleares. Dicha aproximación fue introducida por Born y Oppenheimer en 1927[13], bajo la cual, por ejemplo, la molécula de hidrógeno puede ser tratada.

Desde la perspectiva electrónica, la adición de un electrón adicional (en He) requiere la incorporación de interacción electrónica repulsiva. Además, dado que estamos considerando más de una partícula, debe dirigirse una especial atención a la *indistiguibilidad* de las partículas. Al considerar la naturaleza fermiónica de los electrones, imponemos una restricción en la simetría de la función de onda respecto al intercambio de las etiquetas de dos partículas, siendo en este caso, antisimétrica[1].

[1] En el caso de partículas bosónicas, la función de onda es simétrica respecto a la permutación de dos partículas.

En este capítulo, los conceptos clave introducidos en los párrafos anteriores son tratados con mayor profundidad con el propósito de constituir la base teórica sobre la que se sienta el desarrollo de esta tesis.

2.1 La Ecuación de Schrödinger Molecular

Toda la información mecano-cuántica de los estados estacionarios de un sistema molecular, está contenida en las soluciones de la ecuación de Schrödinger independiente del tiempo no relativista:

$$\hat{H}_{mol}\Psi(\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_N, \mathbf{R}_1, \mathbf{R}_2, ...\mathbf{R}_M) = E\Psi(\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_N, \mathbf{R}_1, \mathbf{R}_2, ..., \mathbf{R}_M), \tag{2-1}$$

en donde el conjunto $\{\mathbf{R}_i\}$ hace referencia al conjunto de las coordenadas espaciales de los núcleos. En el contexto de la versión no relativista de esta ecuación, la descripción completa del electrón es llevada a cabo introduciendo su espín, representando este grado de libertad adicional mediante dos funciones de espín ortonormales, $\alpha(\omega)$ y $\beta(\omega)$. De esta forma, el electrón es descrito en términos de tres coordenadas espaciales y una cuarta adicional asociada al espín. Denotaremos estas cuatro coordenadas colectivamente mediante $\mathbf{x}$.

El Hamiltoniano molecular puede separarse de la siguiente manera:

$$\hat{H}_{mol} = \hat{T}_{el} + \hat{V}_{el-nuc} + \hat{V}_{el-el} + \hat{T}_{nuc} + \hat{V}_{nuc-nuc}. \tag{2-2}$$

Aquí, la energía cinética de los electrones está dada por (m_e es la masa del electrón):

$$\hat{T}_{el} = \sum_{j=1}^{N_{el}} \frac{\hat{p}_j^2}{2m_e}, \tag{2-3}$$

y para los núcleos, tenemos:

$$\hat{T}_{nuc} = \sum_{n=1}^{N_{nuc}} \frac{\hat{P}_n^2}{2M_n} \tag{2-4}$$

siendo M_n la masa del n-ésimo núcleo. Debido a que núcleos y electrones son partículas cargadas, éstas interactúan mediante fuerzas coulómbicas. La interacción repulsiva entre un par de

electrones se define (en unidades atómicas) de la siguiente forma:

$$V_{el-el} = \frac{1}{2} \sum_{i \neq j} \frac{1}{|\mathbf{r}_i - \mathbf{r}_j|}, \tag{2-5}$$

mientras que para los núcleos es:

$$V_{nuc-nuc} = \frac{1}{2} \sum_{m \neq n} \frac{z_m z_n}{|\mathbf{R}_m - \mathbf{R}_n|}. \tag{2-6}$$

De forma similar, la interacción atractiva entre electrones y núcleos está dada por:

$$V_{el-nuc} = -\sum_{j,n} \frac{z_n}{|\mathbf{r}_j - \mathbf{R}_n|}. \tag{2-7}$$

Dado que tenemos N_{el} electrones y N_{nuc} núcleos, la molécula tiene un total de $3(N_{el} + N_{nuc})$ grados de libertad espaciales. A cada electrón se le asigna un número cuántico adicional, σ_j para considerar su espín. El concepto de este último fue introducido para explicar la estructura fina de ciertos espectros atómicos por Unlenbeck y Goudsmit en 1925[14], y poco después, el espín como un grado de libertad adicional cuántico fue introducido por Pauli. Posteriormente, su fundamento teórico surgió como una extensión relativista de la mecánica cuántica desarrollada por Dirac en 1928[15]. Cuando usamos la versión no relativista de la ecuación de Schrödinger, no existen motivos para introducir de forma rigurosa operadores de espín y derivar el potencial de interacción entre variables de coordenadas y de espín (acoplamiento espín-órbita). Por lo tanto, la existencia de operadores de espín es usualmente postulada[2] y su acción sobre funciones de espín definida.

La solución práctica de la ecuación 2-1 hace uso de la gran diferencia en la magnitud de las masas entre electrones y núcleos ($m_e/M_n < 10^{-3}$), lo que sugiere un movimiento mucho más rápido por parte de los primeros en comparación con los segundos. Por lo tanto, los grados de libertad electrónicos responden instantáneamente a cambios en la configuración nuclear. De esta forma, es razonable definir un Hamiltoniano electrónico que depende paramétricamente de

[2]Los operadores de espín fueron postulados por Pauli en 1927 como matrices de 2×2 que actúan sobre la base de los estados de espín del electrón.

las coordenadas nucleares:

$$\widehat{H}_{el}(\mathbf{R}) = \hat{T}_{el} + \hat{V}_{el-nuc} + \hat{V}_{el-el}, \tag{2-8}$$

como consecuencia, las soluciones de la ecuación de Schrödinger electrónica independiente del tiempo, correspondientes a los estados de los electrones en el campo electrostático generado por núcleos estacionarios, dependerán paramétricamente de las coordenadas nucleares:

$$\widehat{H}_{el}(\mathbf{R})\phi_a(\mathbf{x};\mathbf{R}) = E_a(\mathbf{R})\phi_a(\mathbf{x};\mathbf{R}). \tag{2-9}$$

En estas dos últimas ecuaciones, se ha hecho el uso de $\mathbf{R}$ y $\mathbf{x}$ para denotar el conjunto de coordenadas espaciales de los núcleos y las coordenadas espaciales y de espín de los electrones, respectivamente.

Las soluciones de la ecuación 2-9 son denominadas funciones de onda electrónicas adiabáticas. Éstas últimas constituyen el punto de partida para la definición formal de la aproximación de Born-Oppenheimer (para mayor detalle, referirse al Apéndice I), la cual puede resumirse en la ecuación:

$$\psi_{aM}^{(adia)} = \chi_{aM}(\mathbf{R})\phi_a(\mathbf{x};\mathbf{R}), \tag{2-10}$$

en donde $\psi_{aM}^{(adia)}$ es la solución a la Ecuación 2-1.

Dicho de otra forma, los grados de libertad vibracionales y electrónicos de la molécula están desacoplados, de manera que las funciones $\chi_{aM}(\mathbf{R})$ y $\phi_a(\mathbf{r};\mathbf{R})$ pueden calcularse resolviendo de forma separada un Hamiltoniano nuclear y un Hamiltoniano electrónico, respectivamente.

En el marco de la aproximación de Born-Oppenheimer, la solución de la ecuación 2-9 ha constituido la motivación para el desarrollo de diferentes metodologías, dada su importancia para el químico cuántico en la predicción de información termodinámica relevante en procesos químicos y biológicos[16] [17] [18], en la elucidación de mecanismos de reacción[19] [20] [21], y en la predicción de espectros de compuestos de interés[22] [23] [24] por citar sólo algunas aplicaciones. En las siguientes subsecciones, se abunda sobre los métodos de estructura electrónica más ampliamente utilizados actualmente, poniendo especial atención a los métodos basados en la función de onda, dado que en este rubro cae el método sobre el que se llevó a cabo este trabajo.

2.2 Métodos de Estructura Electrónica

Considerando la validez de la aproximación de Born-Oppenheimer, la estructura electrónica y propiedades de cualquier molécula, en cualquiera de sus estados estacionarios disponibles, puede ser determinada en principio por la solución de la ecuación de Schrödinger independiente del tiempo (ecuación 2-8). En esta sección veremos algunas de las herramientas disponibles actualmente para resolverla. Por simplicidad y al mismo tiempo para fines de esta tesis, la discusión es restringida al estado electrónico basal E_0.

2.3 Aproximación de Hartree Fock

El método de Hartree-Fock es un método variacional que constituye el punto de partida para métodos *ab initio* más avanzados que buscan incluir el efecto de la correlación electrónica[3], el cual no es considerado en el primero.

Como punto de partida, consideremos el caso del Hamiltoniano electrónico expresado en la ecuación 2-8, pero con el término de interacción electrónica igualado a 0. De esta forma, el Hamiltoniano en consideración se reduce a una suma de Hamiltonianos correspondientes a partículas individuales, $\widehat{H} = \sum_{j=1}^{N} \widehat{h}(r_j)$, los cuales contienen el término de energía cinética del j-ésimo electrón y la energía coulómbica debida a su interacción con los núcleos estáticos.

La ecuación de Schrödinger para $\widehat{h}$ es resuelta encontrando la función de onda monoelectrónica $\phi_{a_j}(\mathbf{x_j})$:

$$\widehat{h}(\mathbf{r_j})\phi_{a_j}(\mathbf{x_j}) = \left[T(\mathbf{r}_j) + V_{el-nuc}(\mathbf{r}_j)\right]\phi_{a_j}(\mathbf{x_j}) = \epsilon_{a_j}\phi_{a_j}(\mathbf{x_j}). \tag{2-11}$$

Las funciones de onda $\phi_{a_j}(\mathbf{x_j})$ son llamadas espín-orbitales. Dado que el Hamiltoniano empleado aquí es independiente del espín, la función de espín puede ser separada del orbital espacial en la función de onda de acuerdo con $\phi_{a_j}(\mathbf{x_j}) = \phi_{a_j}(r_j)\zeta_{a_j}(\omega_j)$, donde ζ_{a_j} puede ser α o β. Adicionalmente, y de acuerdo al principio de exclusión de Pauli, cada orbital espacial puede ser ocupado por dos electrones con diferente función de espín ζ. Bajo estas consideraciones, los N

[3]En este caso, el término "correlación electrónica" hace alusión a la diferencia entre la energía exacta no relativista y la energía de Hartree-Fock, ambas calculadas en una base completa, a diferencia de la *correlación de Fermi*, la cual es tratada explícitamente en la aproximación de Hartree-Fock y que se origina como consecuencia del principio de exclusión de Pauli.

electrones que componen un determinado sistema pueden ser distribuídos entre los diferentes orbitales monoelectrónicos, lo que deriva diferentes situaciones dependiendo de la paridad de N, y de que si la configuración electrónica corresponde a un estado basal o a una exitada: por un lado, si N es par y el estado electrónico es basal, la mayoría de las configuraciones son de *capa cerrada*, es decir, que todos los orbitales espaciales ocupados tienen dos electrones [4]. De otra forma, si N es impar tenemos una configuración de *capa abierta*[5].

La importancia de la discusión anterior en torno a los espín-orbitales, radica en que éstos son el punto de partida para la formulación de una forma de la función de onda que puede ser optimizada variacionalmente.

Al principio de este capítulo, se hizo referencia a la propiedad de antisimetría de la función de onda solución de la ecuación de Schrödinger. A partir de los espín-orbitales podemos generar una función de onda antisimétrica respecto a la permutación de las etiquetas de dos electrones de la siguiente manera:

$$\phi(\mathbf{x}) = \frac{1}{\sqrt{N!}} \sum_{i}^{N!} (-1)^p P_i \left[\phi^{HP}_{\{a_j\}}(\mathbf{x}) \right] \tag{2-12}$$

Donde el producto de Hartree $\phi^{HP}_{a,j}(\mathbf{x})$ es definido de la siguiente manera:

$$\phi^{HP}_{\{a_j\}}(\mathbf{x}) = \prod_{j=1}^{N} \phi_{a_j}(\mathbf{x_j}). \tag{2-13}$$

En la ecuación 2-12 el operador de permutación P_i actúa sobre la función de onda produciendo una de las $N!$ permutaciones posibles en los índices de las partículas, y el número natural p corresponde al número de permutaciones de dos partículas que componen P_i[6].

Alternativamente, la ecuación 2-12 puede ser escrita en la forma de un determinante,

[4] Existen excepciones importantes, como el oxígeno.

[5] Existe la posibilidad de capa abierta al considerar un número par de electrones y una configuración correspondiente a estado exitado.

[6] Una permutación P_i puede expresarse como el producto de transposiciones, o dicho de otra forma, permutaciones de dos elementos.

comúnmente denominado *determinante de Slater*:

$$\phi(\mathbf{x}) = \frac{1}{N!}\begin{vmatrix} \phi_1(\mathbf{x}_1) & \phi_2(\mathbf{x}_1) & ... & \phi_N(\mathbf{x}_1) \\ \phi_1(\mathbf{x}_2) & \phi_2(\mathbf{x}_2) & ... & \phi_N(\mathbf{x}_2) \\ \vdots & \vdots & ... & \vdots \\ \phi_1(\mathbf{x}_N) & \phi_2(\mathbf{x}_N) & ... & \phi_N(\mathbf{x}_N) \end{vmatrix}. \tag{2-14}$$

De esta forma, las propiedades elementales de los determinantes garantizan la propiedad de antisimetría de la función de onda.

2.3.1 Las Ecuaciones de Hartree-Fock

Hasta ahora no hemos considerado el efecto de la interacción coulómbica entre electrones en la solución de la ecuación de Schrödinger. Dentro de la teoría de Hartree-Fock, esto es llevado a cabo tomando como punto de partida la función de onda antisimetrizada discutida en la sección anterior. A partir de ésta, los espínes orbitales son optimizados variacionalmente con el propósito de aproximar la energía del estado basal electrónico. Este procedimiento conduce al cumplimiento de las ecuaciones de Hartree-Fock, que en unidades atómicas es:

$$\begin{aligned} \hat{f}\phi^j_\sigma(\mathbf{r}) &= \left[-\frac{1}{2}\nabla^2 + v_{ext}(\mathbf{r}) + v^h(\mathbf{r})\right]\phi^j_\sigma(\mathbf{r}) + \int d\mathbf{r}'\Sigma^x_\sigma(\mathbf{r},\mathbf{r}')\phi^j_\sigma(\mathbf{r}') \\ &= \left[\hat{h}(\mathbf{r}) + v^h(\mathbf{r})\right]\phi^j_\sigma(\mathbf{r}) + \int d\mathbf{r}'\Sigma^x_\sigma(\mathbf{r},\mathbf{r}')\phi^j_\sigma(\mathbf{r}') = \epsilon^j_\sigma\phi^j_\sigma(\mathbf{r}). \end{aligned} \tag{2-15}$$

Aquí, $v_{ext}(\mathbf{r}) = -\sum_j \frac{z_j}{|\mathbf{r}-\mathbf{R}_j|}$ es el potencial producido por la distribución de los núcleos atómicos, $\hat{f}$ denota el operador de Fock de una sola partícula (en donde se ha llevado a cabo la integración sobre el espín σ para eliminar la dependencia respecto a la coordenada de espín ω[7]), y $v^h(\mathbf{r})$ es el potencial de Hartree,

$$v^h(\mathbf{r}) = \int \frac{n(\mathbf{r}')}{|\mathbf{r}-\mathbf{r}'|}d\mathbf{r}', \tag{2-16}$$

con la densidad electrónica:

$$n(\mathbf{r}) = \sum_{j\sigma}^{occ} |\phi^j_\sigma(\mathbf{r})|^2 \tag{2-17}$$

[7]Esto se lleva a cabo multiplicando la ecuación por la izquierda por $\zeta(w)^*$ e integrando sobre ω.

y Σ_σ^x está determinado por:

$$\Sigma_\sigma^x(\mathbf{r},\mathbf{r}') = -\sum_j^{occ} \frac{\phi_\sigma^j(\mathbf{r})\phi_\sigma^{j*}(\mathbf{r}')}{|\mathbf{r}-\mathbf{r}'|}. \tag{2-18}$$

Las ecuaciones 2-15 - 2-18, forman un sistema de ecuaciones no lineales que tienen que ser resueltas de forma iterativa. La suma de $v^h(\mathbf{r})$ y $\int d\mathbf{r}'\Sigma_\sigma^x(\mathbf{r},\mathbf{r}')$ forman el denominado potencial de Hartree-Fock v^{HF}, que puede interpretarse como un potencial promedio producido por el resto de los electrones al solucionar la ecuación de Fock para el j-ésimo electrón.

Cuando la autoconsistencia es lograda, la energía de Hartree-Fock total, está determinada por:

$$E_{HF} = \langle\Phi_0|\hat{H}^0 + \hat{H}^{'}|\Phi_0\rangle, \tag{2-19}$$

donde $\hat{H}^0 = \sum_j \hat{f}(\mathbf{r_j})$[8], y $\hat{H}' = \sum_{i<j}\frac{1}{|\mathbf{r_i}-\mathbf{r_j}|} - \sum_j v^{HF}(\mathbf{r_j})$. Tomando en consideración las reglas de Condon-Slater[25] para encontrar expresiones de elementos de matriz de operadores mono- y bielectrónicos en una base de estados con forma determinantal, notamos que:

$$\langle\Phi_0|\hat{H}^{'}|\Phi_0\rangle = \frac{1}{2}\sum_m^N\sum_n^N\sum_{\sigma,\sigma'}\left[(mm,\sigma|nn,\sigma') - (mn,\sigma|nm,\sigma')\delta_{\sigma,\sigma'}\right] - \sum_{i=1}^N\sum_\sigma(i,\sigma|v^{HF}|i,\sigma), \tag{2-20}$$

en donde hemos hecho uso de la notación para integrales bielectrónicas comúnmente empleadas en química cuántica:

$$(mn,\sigma|nm,\sigma') = \int\int d\mathbf{r}d\mathbf{r}'\frac{\phi_\sigma^{m*}(\mathbf{r})\phi_\sigma^n(\mathbf{r})\phi_{\sigma'}^{n*}(\mathbf{r}')\phi_{\sigma'}^m(\mathbf{r}')}{|\mathbf{r}-\mathbf{r}'|} \tag{2-21}$$

$$(mm,\sigma|nn,\sigma') = \int\int d\mathbf{r}d\mathbf{r}'\frac{\phi_\sigma^{m*}(\mathbf{r})\phi_\sigma^m(\mathbf{r})\phi_{\sigma'}^{n*}(\mathbf{r}')\phi_{\sigma'}^n(\mathbf{r}')}{|\mathbf{r}-\mathbf{r}'|}. \tag{2-22}$$

Adicionalmente, podemos emplear:

$$\begin{aligned}\sum_{i,\sigma}(i,\sigma|v^{HF}|i,\sigma) &= \sum_i\sum_\sigma\int\int d\mathbf{r}d\mathbf{r}'\frac{|\phi_\sigma^i(\mathbf{r})|^2 n(\mathbf{r}')}{|\mathbf{r}-\mathbf{r}'|} - \sum_{i\sigma}\int\int d\mathbf{r}d\mathbf{r}'\phi_\sigma^{i*}(\mathbf{r})\Sigma_\sigma^x(\mathbf{r},\mathbf{r}')\phi_\sigma^i(\mathbf{r}') \\ &= \sum_i\sum_j\sum_{\sigma,\sigma'}\left[(ii,\sigma|jjn,\sigma') - (ij,\sigma|ji,\sigma')\delta_{\sigma,\sigma'}\right]\end{aligned} \tag{2-23}$$

[8] $\hat{f}(\mathbf{r_j})$ es el operador monoelectrónico para el j-ésimo electrón, definido por $\hat{f}(\mathbf{r_j}) = \hat{h}(\mathbf{r}_j) + v^{HF}(\mathbf{r_j})$

que, en conjunto con 2-20 y 2-19, nos permite obtener

$$E_{HF} = \sum_{n\sigma} \epsilon_{n\sigma} - \frac{1}{2} \sum_{i} \sum_{j} \sum_{\sigma,\sigma'} \left[(ii, \sigma | jjn, \sigma') - (ij, \sigma | ji, \sigma') \delta_{\sigma,\sigma'} \right]. \tag{2-24}$$

En general, para propósitos de implementación computacional, los espínes orbitales ϕ_σ^j, son expandidos en un conjunto de funciones base $\{\psi_i(\mathbf{r})\}$:

$$\phi_\sigma^j(\mathbf{r}) = \sum_\nu c_{j\sigma}^\nu \psi_\nu(\mathbf{r}), \tag{2-25}$$

donde $c_{j\sigma}^\nu$ son los coeficientes de la expansión. Podemos obtener una ecuación matricial para los coeficientes $c_{j\sigma}^\nu$ al sustituir la expansión anterior en la ecuación de Fock 2-15, lo que produce:

$$\sum_\nu c_{j\sigma}^\nu \hat{f} \psi_\nu(\mathbf{r}) = \epsilon_j^\sigma \sum_\nu c_{j\sigma}^\nu \psi_\nu(\mathbf{r}). \tag{2-26}$$

Multiplicando esta ecuación por la izquierda por $\psi_\mu^*(\mathbf{r})$ e integrando sobre las coordenadas espaciales, llegamos a:

$$\sum_\nu F_{\mu\nu} c_{\nu j}^\sigma = \epsilon_j^\sigma \sum_\nu S_{\mu\nu} c_{\nu j}^\sigma, \tag{2-27}$$

donde $\mathbf{S}$ es la matriz de traslape, cuyos elementos son definidos por:

$$S_{\mu\nu} = \int d\mathbf{r} \psi_\mu^*(\mathbf{r}) \psi_\nu(\mathbf{r}), \tag{2-28}$$

y $\mathbf{F}$ es la representación matricial de $\hat{f}$ en la base $\{\psi_\mu\}$. En términos de esta base, el potencial efectivo de Hartree-Fock podemos expresarlo de forma matricial de la siguiente forma:

$$\begin{aligned} V_{\nu\mu,\sigma}^{HF} &= \int\int d\mathbf{r} d\mathbf{r}' \psi_\nu(\mathbf{r}) \left[v^h(\mathbf{r}) \delta(\mathbf{r} - \mathbf{r}') + \Sigma_\sigma^x(\mathbf{r}, \mathbf{r}') \right] \psi_\mu(\mathbf{r}'), \\ &= v_{\nu\mu,\sigma}^h + \Sigma_{\nu\mu,\sigma}^x \end{aligned} \tag{2-29}$$

donde, la matriz de intercambio $\Sigma_{\nu\mu,\sigma}^x$ está dada por:

$$\Sigma_{\nu\mu,\sigma}^x = \sum_{\gamma\beta} (\nu\gamma | \beta\mu) D_{\gamma\beta,\sigma}. \tag{2-30}$$

En la ecuación 2-30, **D** es la matriz de densidad

$$D_{\gamma\beta,\sigma} = \sum_{n}^{occ} c_{n\sigma}^{\gamma} c_{n\sigma}^{\beta *}, \tag{2-31}$$

y $(\nu\gamma|\beta\mu)$ denota las integrales bielectrónicas de cuatro centros en la base $\{\psi_\mu(\mathbf{r})\}$, definida de la siguiente forma:

$$(\nu\gamma|\beta\mu) = \int\int d\mathbf{r} d\mathbf{r}' \frac{\psi_\nu^*(\mathbf{r})\psi_\gamma(\mathbf{r})\psi_\beta^*(\mathbf{r}')\psi_\mu(\mathbf{r}')}{|\mathbf{r}-\mathbf{r}'|}. \tag{2-32}$$

Así, procediendo del mismo modo con la representación matricial de v^h y h, obtenemos que

$$F_{\mu\nu} = (\mu|\hat{h}|\nu) + \sum_{\gamma\beta,\sigma'} \left[D_{\gamma\beta,\sigma'}(\mu\nu|\gamma\beta) - \delta_{\sigma',\sigma} D_{\gamma\beta,\sigma}(\mu\gamma|\beta\nu) \right]. \tag{2-33}$$

De esta expresión, y retomando las Ecuaciones 2-27 y 2-31, notamos que la matriz de Fock es dependiente de sus propios eigenvectores, y por lo tanto, se recurre al procedimiento de Campo Autoconsistente (SCF, por sus siglas en inglés) para encontrar los coeficientes de los orbitales atómicos ($\{\psi_\mu\}$) que forman los orbitales moleculares que minimizan la energía. En la figura 2.1, se ilustra un diagrama de flujo que describe el método SCF.

2.4 Teoría de Perturbaciones de Muchos Cuerpos

El método perturbativo de Rayleigh-Schrödinger es aplicable a niveles discretos de energía de un sistema físico cuyo operador Hamiltoniano puede ser dividido de la siguiente forma:

$$\widehat{H} = \widehat{H}_0 + \lambda \hat{V}. \tag{2-34}$$

De éstos, $\widehat{H}_0$ corresponde a un *Hamiltoniano no perturbado*, y $\hat{V}$ es la perturbación. En 2-34, λ es un número real entre 0 y 1, que podemos usar como un parámetro de expansión y para definir los órdenes de la perturbación.

El problema que intentamos resolver es

$$\widehat{H}\Psi_n = E_n \Psi_n, \tag{2-35}$$

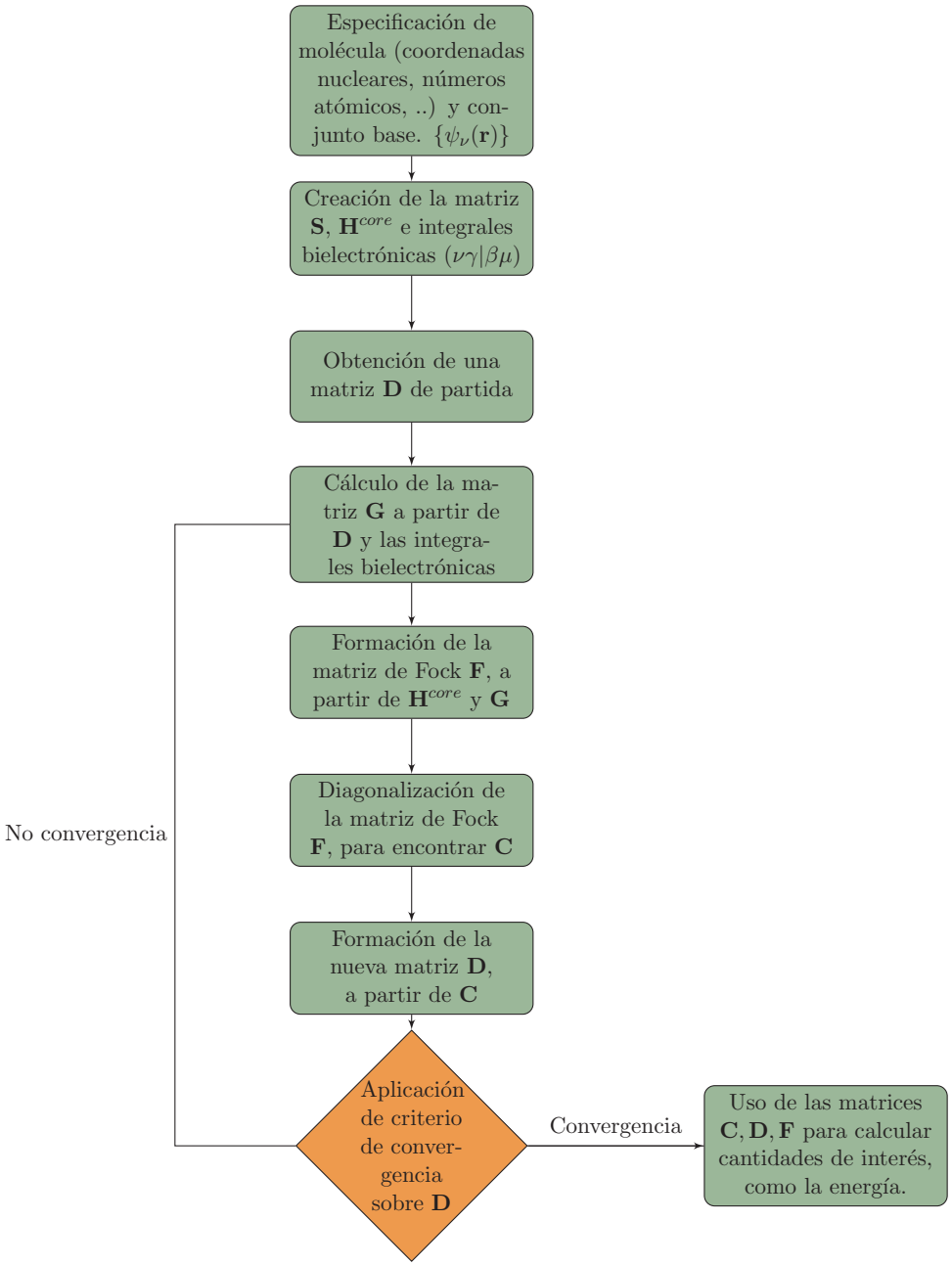

Figura 2.1: Representación esquemática del método SCF. Entre los criterios de convergencia que se pueden emplear, se encuentra, por ejemplo, la diferencia de energía entre dos iteraciones sucesivas, o bien algún criterio sobre la matriz de densidad $\mathbf{D}$, como la desviación estándar de los elementos de dos matrices de densidad sucesivas.

donde denotamos por Ψ_n al conjunto de soluciones del Hamiltoniano electrónico, y suponemos que el problema asociado:

$$\widehat{H}_0\Psi_n^{(0)} = E_n^{(0)}\Psi_n^{(0)} \tag{2-36}$$

ha sido resuelto. La idea fundamental de teoría de perturbaciones es la consideración de que tanto las eigenfunciones como eigenvalores de $\widehat{H}$ pueden ser expandidos en términos del parámetro λ:

$$\begin{aligned} E_n &= E_n^{(0)} + \lambda E_n^{(1)} + \lambda^2 E_n^{(2)} + ... \\ \Psi_n &= \Psi_n^{(0)} + \lambda\Psi_n^{(1)} + \lambda^2\Psi_n^{(2)} + ..., \end{aligned} \tag{2-37}$$

por el momento, y por simplicidad, consideramos que el nivel de energía no perturbado $E_n^{(0)}$ es no degenerado. Para encontrar un procedimiento que permita el mejoramiento sistemático de los eigenvalores y eigenfunciones de $\widehat{H}_0$, de tal manera que se aproximen a los eigenvalores y eigenfunciones del hamiltoniano total $\widehat{H}$, sustituimos las ecuaciones 2-37 en 2-34 y agrupamos términos del mismo orden en el parámetro λ, para obtener las ecuaciones mostradas abajo (considerando únicamente hasta el segundo orden):

$$\widehat{H}_0\Psi_n^{(0)} = E_n^{(0)}\Psi_n^{(0)} \tag{2-38}$$

$$\widehat{H}_0\Psi_n^{(1)} + V\Psi_n^{(0)} = E_n^{(0)}\Psi_n^{(1)} + E_n^{(1)}\Psi_n^{(0)} \tag{2-39}$$

$$\widehat{H}_0\Psi_n^{(2)} + V\Psi_n^{(1)} = E_n^{(0)}\Psi_n^{(2)} + E_n^{(1)}\Psi_n^{(1)} + E_n^{(2)}\Psi_n^{(0)}. \tag{2-40}$$

La ecuación 2-39 podemos reescribirla de la siguiente forma:

$$\left(\widehat{H}_0 - E_n^{(0)}\right)\Psi_n^{(1)} = \left(E_n^{(1)} - \hat{V}\right)\Psi_n^{(0)}. \tag{2-41}$$

El lado derecho de esta ecuación es conocido a excepción del valor de $E_n^{(1)}$, mientras que por otro lado, el vector desconocido $\Psi_n^{(1)}$ aparece a la izquierda. Así 2-41 es una ecuación inhomogénea lineal para $\Psi_n^{(1)}$. Dado que este tipo de ecuaciones son comúnmente encontradas en varios problemas de mecánica cuántica, ya sea formuladas como ecuaciones lineales, matriciales, diferenciales o integrales, un tratamiento más detallado puede encontrarse en el apéndice B.

En la ecuación 2-41, podemos hacer la identificación $A = \hat{H}_0 - E_n^{(0)}$ y notamos que la ecuación homogénea

$$A\Psi = 0, \tag{2-42}$$

tiene las soluciones notriviales $\Psi_n^{(0)}$, que asumiremos como normalizadas. Denotando mediante $\hat{P}_n$ el operador de proyección para la dirección $\Psi_n^{(0)}$, podemos encontrar la solución a 2-41 (referirse al apéndice B) si se cumple que

$$\hat{P}_n \left(E_n^{(1)} - \hat{V} \right) \Psi_n^{(0)} = 0, \tag{2-43}$$

pero

$$\hat{P}\Psi_n^{(0)} = \Psi_n^{(0)}, \tag{2-44}$$

entonces

$$E_n^{(1)}\Psi_n^{(0)} = \hat{P}_n\hat{V}\Psi_n^{(0)}, \tag{2-45}$$

y de esta manera

$$E_n^{(1)} = \langle \Psi_n^{(0)} | \hat{P}_n\hat{V}\Psi_n^{(0)} \rangle = \langle \hat{P}_n\Psi_n^{(0)} | \hat{V}\Psi_n^{(0)} \rangle = \langle \Psi_n^{(0)} | \hat{V} | \Psi_n^{(0)} \rangle = V_{nn}, \tag{2-46}$$

en donde se hizo uso de la hermiticidad de $\hat{P}_n$. De esta expresión, notamos que la corrección de primer orden a la energía, es el valor promedio de la perturbación en el estado no perturbado. Como $\Psi_n^{(0)}$ es un eigenvector no perturbado, la solución general a la ecuación 2-41 está dada por

$$\Psi_n^{(1)} = C_n^{(1)}\Psi_n^{(0)} + K_n \left(\hat{V} - E_n^{(1)} \right) \Psi_n^{(0)}, \tag{2-47}$$

donde el operador K_n es definido por las ecuaciones

$$\left(E_n^{(0)} - \hat{H}_0 \right) K_n = \hat{1} - \hat{P} \qquad y \qquad K_n\Psi_n^{(0)} = 0, \tag{2-48}$$

por lo que tenemos lo siguiente

$$\Psi_n^{(1)} = C_n^{(1)}\Psi_n^{(0)} + K_n\hat{V}\Psi_n^{(0)}. \tag{2-49}$$

Podemos simplificar 2-49 y posteriores ecuaciones estableciendo que $C_n^{(1)}$ y el resto de las constantes arbitrarias $C_n^{(k)}$ que multiplican $\Psi_n^{(0)}$, sean iguales a cero. De esta manera

$$\Psi_n^{(1)} = K_n \hat{V} \Psi_n^{(0)}. \tag{2-50}$$

El mismo procedimiento puede ser aplicado sistemáticamente para órdenes superiores. Para el caso de la corrección a la función de onda a segundo orden, debemos resolver 2-40, que reescribimos como

$$\left(E_n^{(0)} - \hat{H}_0\right) \Psi_n^{(2)} = \left(\hat{V} - E_n^{(1)}\right) \Psi_n^{(1)} - E_n^{(2)} \Psi_n^{(0)}, \tag{2-51}$$

la cual es nuevamente, del tipo inhomogéneo. La correspondiente ecuación homogénea, obtenida al reemplazar el lado derecho de 2-51 por cero, tiene soluciones no triviales $\Psi_n^{(0)}$. Adicionalmente, requerimos que la inhomogeneidad no tenga componentes en la dirección de $\Psi_n^{(0)}$, o dicho de otra forma, que su producto interno con $\Psi_n^{(0)}$ sea igual a cero:

$$\langle \Psi_n^{(0)} | \left(E_n^{(1)} - \hat{V}\right) \Psi_n^{(1)} + E_n^{(2)} \Psi_n^{(0)} \rangle = 0. \tag{2-52}$$

Dado que $\Psi_n^{(0)}$ es ortogonal a $\Psi_n^{(1)}$ (a partir de las ecuaciones 2-48 y 2-50), obtenemos la relación simple

$$E_n^{(2)} = \langle \Psi_n^{(0)} | \hat{V} | \Psi_n^{(1)} \rangle. \tag{2-53}$$

De la misma manera que el caso de la corrección a primer orden, podemos calcular la corrección $\Psi_n^{(2)}$,

$$\Psi_n^{(2)} = C_n^{(2)} \Psi_n^{(0)} - K_n \left(E_n^{(1)} - \hat{V}\right) \Psi_n^{(1)}, \tag{2-54}$$

nuevamente escogemos $C_n^{(2)} = 0$ y usamos las ecuaciones 2-50 y 2-46, para expresar $\Psi_n^{(2)}$ en términos de los eigenestados no perturbados (usando la ecuación 2-45):

$$\Psi_n^{(2)} = -E_n^{(1)} K_n^2 \hat{V} \Psi_n^{(0)} + K_n \hat{V} K_n \hat{V} \Psi_n^{(0)} = -K_n^2 \hat{V} \hat{P}_n \hat{V} \Psi_n^{(0)} + K_n \hat{V} K_n \hat{V} \Psi_n^{(0)}. \tag{2-55}$$

La teoría de perturbaciones puede desarrollarse de esta manera a cualquier orden, sin embargo, las ecuaciones rápidamente se vuelven largas y complejas. Afortunadamente, la existencia de técnicas diagramáticas[26] y programas computacionales algebraicos[27] disminuyen considera-

blemente el desarrollo de las expresiones relevantes.

El problema que resta para encontrar expresiones explícitas a las correcciones a primer y segundo órdenes, es la determinación del operador K_n, para lo cual existen diferentes metodologías. Si éste no puede ser determinado directamente, la ecuación 2-48, puede resolverse por descomposición espectral,

$$K_n = \sum_{l \neq n} \frac{1}{E_n^{(0)} - E_l^{(0)}} \hat{P}_l \tag{2-56}$$

como puede verificarse por sustitución y notando que $\hat{H}_0 \hat{P}_l = \hat{H}_0 |\Psi_l^{(0)}\rangle\langle\Psi_l^{(0)}| = E_l^{(0)} \hat{P}_l$. De

$$\hat{P}_k \hat{V} \Psi_n^{(0)} = \Psi_k^{(0)} \langle \Psi_k^{(0)} | \hat{V} | \Psi_n^{(0)} \rangle = \Psi_k^{(0)} V_{nk}, \tag{2-57}$$

y 2-56 y 2-50, tenemos que:

$$\Psi_n^{(1)} = \sum_{k \neq n} \Psi_k^{(0)} \frac{V_{kn}}{E^{(0)} - E_k^{(0)}}. \tag{2-58}$$

Por otro lado, sustituyendo 2-58 en 2-53, obtenemos

$$E_n^{(2)} = \langle \Psi_n^{(0)} | \hat{V} K_n \hat{V} \Psi_n^{(0)} \rangle = \sum_{k \neq n} \frac{V_{nk} V_{kn}}{E_n^{(0)} - E_k^{(0)}} = \sum_{k \neq n} \frac{|V_{nk}|^2}{E_n^{(0)} - E_k^{(0)}} \tag{2-59}$$

2.4.1 Teoría de Perturbaciones y Correlación Electrónica

La teoría de perturbaciones expuesta hasta ahora es de cararácter general. Un caso especial y de relevancia es aquél en el que el Hamiltoniano no perturbado corresponde al de Hartree-Fock, que se compone de la suma de Hamiltonianos de partículas independientes

$$\hat{H}_0 = \sum_j \hat{f}(\mathbf{r_j}) = \sum_j [\hat{h}(\mathbf{r_j}) + v^{HF}(\mathbf{r_j})], \tag{2-60}$$

donde, de nueva cuenta

$$\hat{h}(\mathbf{r_j}) = -\frac{1}{2} \nabla^2(\mathbf{r_j}) - \sum_n \frac{z_n}{|\mathbf{r_j} - \mathbf{R_n}|} \tag{2-61}$$

y $v^{HF}(\mathbf{r})$ se define de acuerdo a

$$v^{HF}(\mathbf{r}) \phi_\sigma^i(\mathbf{r}) = \sum_{j\sigma}^{occ} \int \frac{|\phi_\sigma^j(\mathbf{r'})|^2}{|\mathbf{r} - \mathbf{r'}|} \phi_\sigma^i(\mathbf{r}) d\mathbf{r'} - \sum_j^{occ} \int d\mathbf{r'} \frac{\phi_\sigma^j(\mathbf{r}) \phi_\sigma^{j*}(\mathbf{r'})}{|\mathbf{r} - \mathbf{r'}|} \phi_\sigma^i(\mathbf{r'}). \tag{2-62}$$

Para obtener una expansión perturbativa de la energía de correlación, particionamos el Hamiltoniano de la siguiente manera:

$$\hat{H} = \hat{H}_0 + V, \tag{2-63}$$

donde

$$V = \sum_{i<j} r_{ij}^{-1} - \sum_i v^{HF}(\mathbf{r}_j). \tag{2-64}$$

El uso del Hamiltoniano particionado mostrado arriba, en combinación con las expresiones generales de la teoría de perturbaciones de Rayleig-Schrödinger, es conocida comúnmente como teoría de perturbaciones de Møller-Plesset.

Las correcciones a la función de onda y a la energía electrónica son llevadas a cabo a partir de la función de onda Hartree-Fock $|\Psi_0\rangle$, que es una eigenfunción de $\hat{H}_0$,

$$\hat{H}_0|\Psi_0\rangle = E_0^{(0)}|\Psi_0\rangle \tag{2-65}$$

con eigenvalor

$$E_0^{(0)} = \sum_i \epsilon_i. \tag{2-66}$$

Aquí es importante señalar que usaremos la convención comúnmente encontrada en la literatura para etiquetar espín orbitales ocupados mediante $i, j, k...$; $a, b, c...$ para denotar espín orbitales virtuales, y $r, l, m...$ para denotar cualquiera. Tomando lo anterior en consideración, tenemos que la corrección a primer orden en la energía queda determinada por

$$\begin{aligned}
E_0^{(1)} &= \langle\Psi_0|V|\Psi_0\rangle \\
&= \langle\Psi_0|\sum_{i<j} r_{ij}^{-1}|\Psi_0\rangle - \langle\Psi_0|\sum_i v^{HF}(\mathbf{x_i})|\Psi_0\rangle \\
&= \frac{1}{2}\sum_i^N\sum_j^N\sum_{\sigma,\sigma'}\left[(ii,\sigma|jj,\sigma') - (ij,\sigma|ji,\sigma')\delta_{\sigma,\sigma'}\right] - \sum_{i=1}^N\sum_\sigma (i,\sigma|v^{HF}|i,\sigma) \\
&= -\frac{1}{2}\sum_i\sum_j\sum_{\sigma,\sigma'}\left[(ii,\sigma|jjn,\sigma') - (ij,\sigma|ji,\sigma')\delta_{\sigma,\sigma'}\right].
\end{aligned} \tag{2-67}$$

Retomando la expresión de la energía total de Hartree-Fock (ecuación 2-24), notamos que ésta es equivalente a la suma de las energías de orden cero y de primer orden,

$$E_0 = E_0^{(0)} + E_0^{(1)} = \sum_i \epsilon_i - \frac{1}{2}\sum_i \sum_j \sum_{\sigma,\sigma'} \left[(ii,\sigma|jjn,\sigma') - (ij,\sigma|ji,\sigma')\delta_{\sigma,\sigma'}\right], \tag{2-68}$$

por lo tanto, la primera corrección a la energía de Hartree-Fock ocurre en el segundo orden de la perturbación. El resultado general para la energía de segundo orden, derivado en la sección anterior, es

$$E_0^{(2)} = \sum_{n\neq 0} \frac{|\langle 0|V|n\rangle|^2}{E_0^{(0)} - E_n^{(0)}}, \tag{2-69}$$

donde la suma corre sobre todos los estados del sistema, a excepción del basal. En esta expresión, los estados $|n\rangle$ no pueden ser excitaciones simples, dado que

$$\langle\Psi_0|V|\Psi_i^a\rangle = \langle\Psi_0|\hat{H} - \hat{H}_0|\Psi_i^a\rangle = \langle\Psi_0|\hat{H}|\Psi_i^a\rangle - f_{ai} = 0, \tag{2-70}$$

en donde $\langle\Psi_0|\hat{H}|\Psi_i^a\rangle$ es igual a cero como consecuencia del teorema de Brillouin, y lo mismo ocurre con el segundo, dado que tanto $|\Psi_0\rangle$ como $|\Psi_i^a\rangle$ son funciones propias del operador hermitiano $\hat{H}_0$, con valores propios distintos, y por lo tanto, ortogonales. Adicionalmente, excitaciones de carácter triple o superior no se acoplan a $|\Psi_0\rangle$, dada la naturaleza del operador bielectrónico de la perturbación. Por lo tanto, en la ecuación 2-69, $|n\rangle$ corresponde a estados doblemente excitados de la forma $|\Psi_{ij}^{ab}\rangle$.

Finalmente, como

$$\hat{H}_0|\Psi_{ij}^{ab}\rangle = \left(E_0^{(0)} - (\epsilon_i + \epsilon_j - \epsilon_a - \epsilon_b)\right)|\Psi_{ij}^{ab}\rangle \tag{2-71}$$

la corrección de segundo orden a la energía es

$$E_0^{(2)} = \sum_{a<b;i<j} \frac{|\langle\Psi_0|\sum_{n<m} r_{nm}^{-1}|\Psi_{ij}^{ab}\rangle|^2}{\epsilon_i + \epsilon_j - \epsilon_a - \epsilon_b} = \sum_{a<b;i<j}\sum_{\sigma,\sigma'} \frac{|(ia,\sigma|jb,\sigma') - (ib,\sigma|ja,\sigma')|^2}{\epsilon_i + \epsilon_j - \epsilon_a - \epsilon_b}, \tag{2-72}$$

la cual podemos reescribir en la forma más comúnmente encontrada en la literatura sumando sobre todos los orbitales ocupados y virtuales de la siguiente manera

$$E_0^{(2)} = \frac{1}{4}\sum_{ab}\sum_{ij}\sum_{\sigma,\sigma'}\left[\frac{(ia,\sigma|jb,\sigma')(ai,\sigma|bj,\sigma')}{\epsilon_i+\epsilon_j-\epsilon_a-\epsilon_b} - \frac{(ib,\sigma|ja,\sigma')(ai,\sigma|bj,\sigma')\delta_{\sigma\sigma'}}{\epsilon_i+\epsilon_j-\epsilon_a-\epsilon_b}\right] + \frac{1}{4}\sum_{ab}\sum_{ij}\sum_{\sigma,\sigma'}\left[\frac{(ib,\sigma|ja,\sigma')(bi,\sigma|aj,\sigma')}{\epsilon_i+\epsilon_j-\epsilon_a-\epsilon_b} - \frac{(ia,\sigma|jb,\sigma')(bi,\sigma|aj,\sigma')\delta_{\sigma\sigma'}}{\epsilon_i+\epsilon_j-\epsilon_a-\epsilon_b}\right] \quad (2\text{-}73)$$

y por lo tanto

$$= \frac{1}{2}\sum_{ij}\sum_{ab}\sum_{\sigma\sigma'}(ia,\sigma|jb,\sigma')\left[\frac{(ai,\sigma|bj,\sigma')-(bi,\sigma|aj,\sigma')\delta_{\sigma,\sigma'}}{\epsilon_{i\sigma}+\epsilon_{j\sigma'}-\epsilon_{a\sigma}-\epsilon_{b\sigma'}}\right]. \quad (2\text{-}74)$$

En la ecuación 2-73, se hace uso del intercambio de los índices mudos a y b en la segunda suma, para dar lugar a la expresión 2-74. La delta de Kronecker en las ecuaciones 2-73 y 2-74, denota que el producto de la forma $(ib,\sigma|ja,\sigma')(ai,\sigma|bj,\sigma')$ es distinto de cero solo cuando $\sigma=\sigma'$.

2.5 Teoría SOS-MP2

La teoría Møller-Plesset constituye la alternativa más simple entre los métodos de estructura electrónica a la ampliamente utilizada teoría DFT, en el tratamiento correcto de interacciones de dispersión y puentes de hidrógeno[28]. Por lo tanto, es muy deseable la introducción de nuevos tratamientos teóricos en dicho formalismo que permitan el desarrollo de códigos más eficientes y que provean de resultados más exactos.

Entre dichos tratamientos, cabe destacar el enfoque desarrollado por Grimme[29], denominado MP2 con Componentes de Espín Escalados (MP2-SCS, por sus siglas en inglés), dado que éste introduce un marco teórico simple y sencillo que permite mejorar la exactitud del método MP2. La idea concebida por Grimme, parte de las diferencias entre las contribuciones a la energía de correlación por parte de espines paralelos (SS, por sus siglas en inglés) y espines opuestos (OS, por sus siglas en inglés) en el método MP2. Por un lado, en este método, el componente SS tiende a ser subestimado, mientras que existe una sistemática sobreestimación del componente OS.

En un intento por contrarrestar dichos defectos, Grimme propone un factor de escalamiento

para cada componente, de acuerdo a:

$$E_{SCS-MP2} = c_{SS}E^{SS}_{MP2} + c_{OS}E^{OS}_{MP2}, \tag{2-75}$$

donde, c_{SS} y c_{OS} se refieren a los factores de escalamiento para los componentes SS y OS, respectivamente, y E^{SS}_{MP2} y E^{OS}_{MP2} son las contribuciones a la energía de correlación provenientes de espines paralelos y opuestos, respectivamente. A través de cálculos de referencia y análisis estadístico, Grimme encontró los valores óptimos $c_{SS} = 1/3$ y $c_{OS} = 6/5$, para un conjunto de 51 reacciones que incluyen hidrogenaciones, adiciones, fragmentaciones, isomerizaciones, afinidades protónicas y estados de transición.

Motivados por estos resultados, Y. Jung y colaboradores[30] sugirieron una versión simplificada del método SCS-MP2. Para ello, justificaron que, dado que la amortiguación del componente SS es grande ($c_{SS} = 1/3$), es posible obtener resultados de una calidad comparable con SCS-MP2, si sólo la contribución OS es escalada, lo cual tiene ventajas algorítmicas importantes (pues de esa forma no es necesaria la evaluación de las integrales de intercambio asociadas a espines paralelos).

De esta forma, la energía de correlación de segundo orden por espines opuestos (SOS-MP2, por sus siglas en inglés), fue definida como:

$$E_{SOS-MP2} = c_{SOS}E^{OS}_{MP2}. \tag{2-76}$$

Tomando ventaja de esta simplificación, se puede demostrar que el nuevo método SOS-MP2, puede implementarse sin requerir del escalamiento de quinto orden respecto al tamaño molecular (en contraste con el método MP2 convencional), haciendo uso de una expansión de bases auxiliares[31], junto con una transformada de Laplace[32].

Retomando de nueva cuenta la teoría de perturbaciones de Møller-Plesset[33], tenemos que la corrección a la energía HF, está dada por:

$$E^{(2)}_0 = \frac{1}{2}\sum_{ij}\sum_{ab}\sum_{\sigma\sigma'}(ia,\sigma|jb,\sigma')\left[\frac{(ai,\sigma|bj,\sigma') - (bi,\sigma|aj,\sigma')\delta_{\sigma,\sigma'}}{\epsilon_{i\sigma} + \epsilon_{j\sigma'} - \epsilon_{a\sigma} - \epsilon_{b\sigma'}}\right], \tag{2-77}$$

la cual podemos particionar en las contribuciones de espines paralelos y opuestos de la siguiente manera:

$$\begin{aligned} E_{MP2} &= \frac{1}{2}\sum_{ij}\sum_{ab}\sum_{\sigma\neq\sigma'}(ia,\sigma|jb,\sigma')\left[\frac{(ai,\sigma|bj,\sigma')}{\epsilon_{i\sigma}+\epsilon_{j\sigma'}-\epsilon_{a\sigma}-\epsilon_{b\sigma'}}\right] \\ &+ \frac{1}{2}\sum_{ij}\sum_{ab}\sum_{\sigma}(ia,\sigma|jb,\sigma)\left[\frac{(ai,\sigma|bj,\sigma)-(bi,\sigma|aj,\sigma)}{\epsilon_{i\sigma}+\epsilon_{j\sigma}-\epsilon_{a\sigma}-\epsilon_{b\sigma}}\right] \\ &= E^{OS}_{MP2}+E^{SS}_{MP2}. \end{aligned} \tag{2-78}$$

Podemos reescribir la contribución OS de una forma más compacta:

$$E^{OS}_{MP2} = -\frac{1}{2}\sum_{ia}^{\alpha}\sum_{jb}^{\beta}\frac{(ia|jb)^2}{\Delta^{ia}_{jb}} - \frac{1}{2}\sum_{ia}^{\beta}\sum_{jb}^{\alpha}\frac{(ia|jb)^2}{\Delta^{ia}_{jb}} = -\sum_{ia}^{\alpha}\sum_{jb}^{\beta}\frac{(ia|jb)^2}{\Delta^{ia}_{jb}}, \tag{2-79}$$

en donde hemos usado el hecho de que tanto los índices i y j, así como a y b, son mudos, y pueden intercambiarse. Asimismo, se ha definido:

$$\Delta^{ia}_{jb} = \epsilon_a + \epsilon_b - \epsilon_i - \epsilon_j, \tag{2-80}$$

en donde se han omitido las etiquetas correspondientes a los espines, en el entendido de que a ϵ_i y ϵ_a les correponde el espín α y a ϵ_j, ϵ_b, el espín β, de acuerdo a la ecuación 2-79.

Introduciendo ahora la transformada de Laplace $\frac{1}{x} = \int_0^\infty \exp^{-xt}\,dt$, con el fin de eliminar denominadores, obtenemos:

$$E^{OS}_{MP2} = -\int_0^\infty dt \sum_{ia}^{\alpha}\sum_{jb}^{\beta}(ia|jb)^2\exp^{-\Delta^{ia}_{jb}t}. \tag{2-81}$$

Ahora, introduciendo una cuadratura discreta con Q puntos para realizar la integración numérica sobre t (una corta digresión sobre métodos de integración numérica puede encontrarse en el Apéndice C), la energía puede ser escrita como:

$$E^{OS}_{MP2} = -\sum_{q}^{Q} w_q \sum_{ia}^{\alpha}\sum_{jb}^{\beta}(ia|jb)^2\exp^{-\Delta^{ia}_{jb}t_q} = -\sum_{q}^{Q}\sum_{ia}^{\alpha}\sum_{jb}^{\beta}(i_q a_q|j_q b_q)^2, \tag{2-82}$$

donde los orbitales canónicos escalados están relacionados con cada punto de la cuadratura de acuerdo a:

$$\phi_i^q = \phi_i w_q^{1/8} \exp^{\frac{1}{2}\epsilon_i t_q}, \tag{2-83}$$

$$\phi_a^q = \phi_a w_q^{1/8} \exp^{-\frac{1}{2}\epsilon_a t_q}. \tag{2-84}$$

Introduciendo ahora una base de N orbitales auxiliares, y denotando cada orbital auxiliar mediante K,L,..., las integrales coulómbicas de cuatro centros pueden ser expresadas en términos de integrales bicéntricas y tricéntricas (aproximación de la Resolución de la Identidad, ver apéndice D):

$$(i_q a_q | j_q b_q) = \sum_K^N B_{ia}^{Kq} B_{jb}^{Kq}. \tag{2-85}$$

Donde:

$$B_{ia}^{Kq} = \sum_L^N (i_q a_q | L) \, (L|K)^{-\frac{1}{2}}. \tag{2-86}$$

Ahora, introduciendo la ecuación (2-85) en (2-82), se encuentra que la energía puede ser expresada como:

$$E_{MP2}^{OS} = -\sum_q^Q \sum_{ia}^{\alpha} \sum_{jb}^{\beta} \sum_{KL} B_{ia}^{Kq} B_{jb}^{Kq} B_{ia}^{Lq} B_{jb}^{Lq} = -\sum_q^Q \sum_{KL} X_{KL}^{\alpha q} X_{KL}^{\beta q}, \tag{2-87}$$

en la que $X_{KL}^{\alpha q}$ es definida de la siguiente forma:

$$X_{KL}^{\alpha q} = \sum_{ia}^{\alpha} B_{ia}^{Kq} B_{ia}^{Lq}, \tag{2-88}$$

con el correspondiente análogo para el espín β.

2.5.1 Implementación Computacional

En la subsección anterior, la teoría SOS-MP2 fue expuesta. En el paquete de química computacional Q-Chem [34], esta teoría está implementada y el algoritmo correspondiente se describe de forma general en los siguientes párrafos.

(1) Preparación de los coeficientes no escalados **B** de las integrales coulómbicas bi y tricéntri-

cas, para lo que se requiere evaluar:

$$B_{ia}^{K} = \sum_{L} \sum_{\mu\nu} \left\{ C_{\nu a} \left[C_{\mu i} \left(\mu\nu | L \right) \right] \right\} \left(L | K \right)^{-1/2}, \tag{2-89}$$

donde los índices griegos denotan los orbitales atómicos y los coeficientes **C** hacen referencia a los coeficentes que definen los orbitales moleculares. Para ello es necesario obtener la raíz de la matriz inversa de la matriz formada por las integrales de dos centros ($(L|K)$), que es una matriz cuadrada de $N \times N$ (N es el número de orbitales auxiliares). Esta inversión tiene un escalamiento cúbico, es decir, el costo computacional es proporcional a N^3.

Posteriormente, este paso es seguido por las transformaciones $\mu \mapsto i$ y $\nu \mapsto a$, a partir de las integrales tricéntricas $(\mu\nu|L)$, cuyo costo escala de forma proporcional a $on^2N + ovnN$, donde o, v y n corresponden al número de orbitales moleculares activos ocupados, al número de orbitales activos virtuales[9], y al número total de orbitales atómicos, respectivamente. Para hacer más evidente el escalamiento del costo, ver el algoritmo 1.

Algoritmo 1: Transformación de las integrales tricéntricas en la base de orbitales atómicos a la base de orbitales moleculares

Result: Integrales transformadas $I_{ia}^{L} = (ia|L)$

```
for i ← 0 to o do
    for L ← 0 to N do
        for ν ← 0 to n do
            for μ ← 0 to n do
                I^L_iν ← I^L_iν + C_μi (μν|L)
        for a ← 0 to v do
            for ν ← 0 to n do
                I^L_ia ← I^L_ia + C_aν I^L_iν
```

Posteriormente, requerimos de un paso adicional para la formación de los coeficientes B_{ia}^{L} a partir de las integrales tricéntricas transformadas a la base de orbitales moleculares,

[9] Cuando se hace uso de la aproximación de núcleo congelado (*frozen core*), el término de orbital molecular activo hace alusión a aquél que es considerado en las expresiones del cálculo de la energía de correlación, a diferencia de los orbitales moleculares inactivos, que no son considerados.

que consiste en la multiplicación de estas últimas por la matriz de integrales bicéntricas. Este paso escala de forma proporcional a ovN^2.

(2) Escalamiento de los coeficientes **B** para cada punto de la cuadratura, de acuerdo con:

$$B_{\overline{ia}}^{K} = B_{ia}^{K} w_q^{1/4} \exp\left(\frac{1}{2}\epsilon_i t_q\right) \exp\left(-\frac{1}{2}\epsilon_a t_q\right) \tag{2-90}$$

En este caso, el costo computacional escala de forma proporcional a $QovN$. Como las cantidades o, v y N son linealmente proporcionales al tamaño de la molécula, este paso escala con el cubo de éste, dado que el número de puntos en la malla numérica Q, es independiente del tamaño molecular.

(3) Construcción de la matriz **X**, para cada punto de la cuadratura evaluando la ecuación (2-88). Este es el paso más demandante del algoritmo y escala de forma proporcional a $QovN^2$ de tal forma que el escalamiento respecto al tamaño molecular es de cuarto orden. Aunque algunos de los primeros pasos del algoritmo descrito hasta ahora también muestran un escalamiento de cuarto orden, la construcción de la matriz $\overline{X}_{KL}^{\alpha}$, a diferencia de ellos, depende cuadráticamente de N, cantidad que es considerablemente mayor que v, o y n[35]. La aseveración anterior es corroborada en las pruebas de referencia reportadas más adelante.

Vale la pena abundar en este paso, dado que sobre éste se llevó a cabo el proceso de implementación sobre la que se sustenta esta tesis.

En Q-Chem, los coeficientes B_{ia}^{Q} son almacenados en un archivo temporal en el orden a, Q, i, siendo el índice a el más rápido e i el más lento, de tal forma que para la construcción de los coeficientes $\overline{X}_{KL}^{\alpha}$, se encuentra implementado un *loop* sobre los orbitales ocupados (i), tal que para cada i se carga a memoria una matriz B_{Ka} de tamaño N x v. Esta matriz es escalada de acuerdo a la ecuación (2-90) y es multiplicada por su transpuesta para producir una matriz de NxN que es aumentada en la matriz **X**.

Nótese que con esta información, se infiere que el tamaño de las matrices a multiplicar depende cuadráticamente del tamaño del sistema.

Capítulo 3

Programación GPGPU

La inexistencia de computadoras digitales en los primeros años de la química cuántica no limitó a los químicos computacionales en la realización de sus cálculos diarios, quienes dependían en gran medida de computadoras analógicas para llevarlos a cabo.

En 1937, Turing[36], considerado padre de las ciencias de la computación modernas, concibió la noción de algoritmo y cómputo mediante su reconocida máquina de Turing. Un año después, la primera computadora moderna fue inventada por Stibitz. En cierto modo, es sorprendente que ahora, más de 70 años después de estos acontecimientos, contemos con arquitecturas de cómputo paralelas que conforman una piedra angular de la química computacional moderna. Actualmente, las computadoras empleadas para química cuántica van desde *workstations* hasta arquitecturas que incluyen *clusters* con frecuencias de reloj del orden de petaflops.

Estas nuevas arquitecturas se traducen en grandes oportunidades para el desarrollo de la química cuántica. Así, los esfuerzos que se han realizado para la introducción de nuevas metodologías que permitan el tratamiento de sistemas más grandes, se han complementado con intentos por aprovechar al máximo la evolución del *hardware* disponible en la época. Por ejemplo, la emergencia de procesadores vectoriales en los ochentas[37], requirió en un principio de código escrito en FORTRAN para la paralelización de bucles, de tal manera que los químicos cuánticos tuvieron la disposición de reescribir sus códigos con la finalidad de obtener ganancias en el rendimiento de hasta un factor de 25, en comparación con las tecnologías convencionales.

La transición a arquitecturas paralelas ha sido mucho más compleja que el ejemplo anteriormente citado. Esto debido a que estas plataformas requiren de la escritura de una porción

significativa de código para manejar la transferencia y comunicación de tareas entre diferentes procesos y nodos. El primer paradigma para llevar a cabo esta comunicación fue la transferencia de mensajes, en el cual, el programador se vale de una rutina para transmitir datos, y de una rutina llamada desde el proceso receptor para recibirlos. Este protocolo es la base para bibliotecas tempranas como TCGMSG[38], PVM[39] y el actual MPI[40], el cual ha sido adoptado en varios códigos de química cuántica.[41] [42] [43] [44] [45] [34].

En los últimos años, los desarrollos en el *hardware*, como procesadores multi-core, han modificado la noción de paralelización de grano fino, lo que ha incentivado de nueva cuenta la modificación de códigos de química computacional para aprovechar las nuevas tecnologías disponibles. El reciente surgimiento de Unidades de Procesamiento Gráfico (GPUs por sus siglas en inglés)[46], modifica nuevamente el paradigma. En este capítulo, el modelo de programación de este *hardware*, su arquitectura, y su relevancia en el campo de la química cuántica, es tratada con mayor detalle.

3.1 La Unidad de Procesamiento Gráfico como Plataforma de Cómputo

Las GPUs constituyen una plataforma poderosa de cómputo y la razón de ello reside en su arquitectura. Tradicionalmente, el procesamiento de gráficos en videojuegos y en simulaciones 3D en tiempo real, ha forzado a las compañías de *hardware* a maximizar la velocidad a la cual los pixeles en una pantalla pueden ser procesados. El número de estos pixeles es dependiente de la resolución de la pantalla, sin embargo, éstos pueden ser procesados en paralelo mediante operaciones independientes. La forma más eficiente de llevar a cabo lo anterior es contar con un gran número de Unidades Lógico-Aritméticas (ALUs, por sus siglas en inglés), las cuales son capaces de efectuar una misma operación de manera independiente sobre diferentes datos sin la necesidad de control de flujo sofisticado. Esto último se traduce en un *hardware* altamente eficiente para la realización de cómputo paralelo, que es comúnmente clasificado como "una operación, múltiples datos" (SIMD, por sus siglas en inglés). En contraste, una Unidad de Procesamiento Central (CPU) convencional, invierte una proporción significativa de

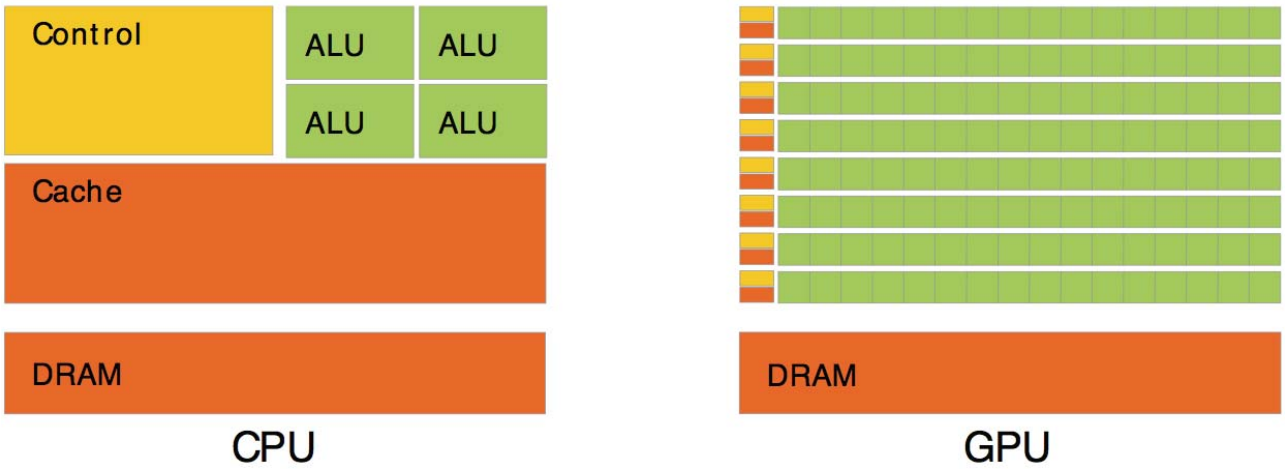

Figura 3.1: Comparación esquemática entre las arquitecturas CPU y GPU. La significativa diferencia entre el número de Unidades Lógico-aritméticas (ALUs, por sus siglas en inglés) y los recursos orientados al almacenamiento de información en memoria, hacen de la GPU una plataforma adecuada para su uso en apliaciones de cómputo de alto rendimiento. En la figura amarillo=control, verde=Unidad Lógico-artimética (ALU), naranja=memoria RAM o caché.

tiempo de procesamiento al almacenamiento de datos en memoria caché[1] y control de flujo. Las diferencias en la arquitectura entre las mencionadas unidades de procesamiento es representada esquemáticamente en la Figura 3.1.

Para cómputo general, los pixeles dejan de ser las unidades de datos procesados por la GPU, y son reemplazados por datos que van desde elementos de matriz en programas de álgebra lineal, hasta objetos que representan la posición de partículas en programas de dinámica molecular.

Sin embargo, existen también limitaciones en el empleo de GPUs para cómputo de propósito general (GPGPU por sus siglas en inglés). Con base en lo expuesto hasta el momento, es evidente que se requiere de un conjunto grande de datos que puedan ser procesados de forma independiente para obtener una computación eficiente con GPUs. Lo anterior no se satisface para todos los problemas y algoritmos.

Por otra parte, cuando un algoritmo es portado de una solución CPU a GPU, podría ser necesario modificar el algoritmo con la finalidad de hacer al problema tratable mediante SIMD.

[1]La memoria caché es empleada por la CPU para reducir los tiempos de acceso a datos almacenados en la memoria RAM, mediante el copiado de datos de localidades comúnmente utilizadas de la última. Esta estrategia reduce el tiempo de acceso dado que la memoria caché es accesada mucho más rápido por la CPU en comparación con la memoria principal (RAM), pero tiene la desventaja de ser más pequeña.

Adicionalmente, la comunicación entre la GPU y el nodo en el que se lleva a cabo el cómputo, está limitada por la latencia[2] y la velocidad del bus PCI (Componente de Conexión Periférico, por sus siglas en inglés) que comunica a ambos. Dicho de otra forma, las operaciones realizadas entre las instrucciones en el nodo y el cómputo en la GPU juegan un rol importante al momento de programar sobre esta arquitectura.

Finalmente, el empleo de GPUs requiere de considerable esfuerzo para obtener un código optimizado, aunque la existencia de bibliotecas y la inclusión de nuevas características en el *hardware* han hecho de ésta una tarea menos demandante.

En este último punto, se han logrado importantes avances: el uso temprano de GPUs entre la comunidad científica requirió de la solución de problemas en términos de *canalización de gráficos*, empleando los lenguajes de programación OpenGL o DirectX[47]. La complejidad en esta tarea motivó la búsqueda de soluciones para realizar cómputo general en GPUs. Fue así que en Noviembre de 2006, NVIDIA introdujo CUDA (Arquitectura de Dispositivo para Cómputo Unificado, por sus siglas en inglés), una plataforma de cómputo paralelo de propósito general y modelo de programación basado en las GPUs NVIDIA. De esta manera, CUDA permite a los desarrolladores la implementación de software sobre la arquitectura GPU usando C como lenguaje de alto nivel.

El modelo de programación de CUDA, considera 3 abstracciones fundamentales, que son:

- Una jerarquía de agrupaciones de hilos. La base del procesamiento paralelo en el modelo de programación reside en los procesos ligeros (comúnmente llamados " hilos" en la literatura, y que ejecutan instrucciones de forma independiente[3]), los cuales pueden ser agrupados en bloques uni-, bi- o tridimensionales. Esto último tiene la ventaja de proveer una forma natural de mapear elementos de vectores, matrices, volúmenes, etc. a los hilos constituyentes de los bloques. Adicionalmente, los bloques son agrupados en una malla.

- Tipos de memorias. Los hilos CUDA pueden accesar a diferentes espacios de memoria durante su ejecución. Cada hilo tiene una memoria local privada, y a su vez, cada bloque

[2]En términos generales, latencia es el periodo de tiempo que existe entre la inicialización de una tarea dada y el comienzo de su ejecución. En el caso de transferencia de datos, la latencia es el tiempo entre la ejecución de transferencia, y el comienzo de ésta.

[3]Los procesos ligeros reciben esta denominación debido a que requieren de poca carga computacional para su creación, en comparación con los procesos pesados, que son empleados por ejemplo, en la programación con MPI.

tiene una memoria compartida accesible a todos los hilos constituyentes del bloque. De la misma forma, todos los hilos tienen acceso a la misma memoria global. (Figura 3.2)

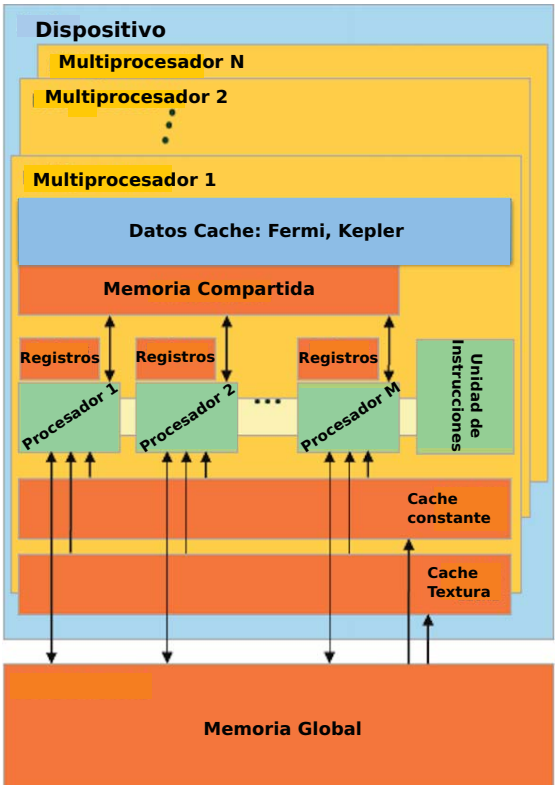

Figura 3.2: Esquema que muestra las jerarquías de memoria GPU. Las nuevas generaciones de GPUs (Kepler), incorporan una memoria cache de datos de solo-lectura.

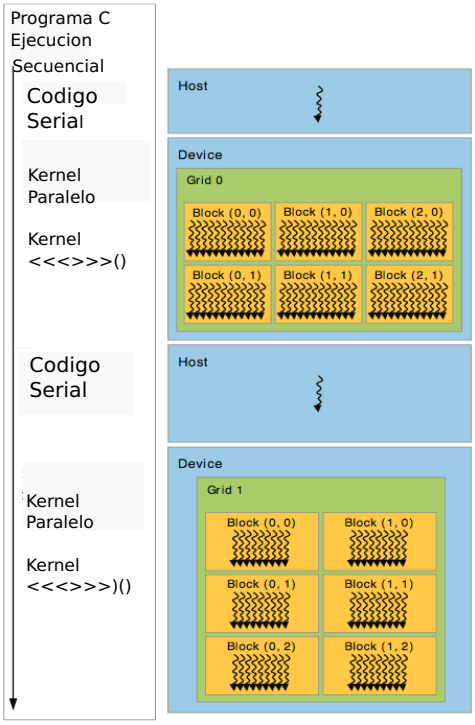

Figura 3.3: Esquema que representa esquemáticamente el modelo de programación heterogéneo CPU-GPU.

- Barreras de sincronización. Aunque los bloques de hilos son ejecutados de forma independiente, la formulación de un algoritmo puede exigir el empleo de métodos de sincronización[4] de los hilos, los cuales actúan como una barrera en la cual todos los hilos del bloque tienen que esperar, antes de que alguno proceda.

[4]Por ejemplo, si el algoritmo emplea memoria compartida en un bloque para almacenar datos intermedios para usarlos en un paso posterior por todos los hilos, es necesario establecer una barrera temporal para asegurar que todos ellos accesen a los datos correctos.

3.2 Química Cuántica en Unidades de Procesamiento Gráfico

3.2.1 Integrales de Repulsión Electrónicas

Las integrales de repulsión electrónicas (ERIS, por sus siglas en inglés):

$$(\mu\nu|\kappa\lambda) = \int\int d\mathbf{r}d\mathbf{r}' \frac{\phi_\mu(\mathbf{r})\phi_\nu(\mathbf{r})\phi_\kappa(\mathbf{r}')\phi_\lambda(\mathbf{r}')}{|\mathbf{r}-\mathbf{r}'|}, \tag{3-1}$$

juegan un papel importante en los cálculos de estructura electrónica, puesto que la formación de las ERIS involucradas en el cómputo de sistemas grandes, constituye usualmente el cuello de botella del mismo.

En 3-1, las funciones $\{\phi_\mu\}$ son funciones gausianas contraídas, es decir, son combinaciones lineales de funciones primitivas gausianas χ_p :

$$(\mu\nu|\kappa\lambda) = \sum_{pqrs} d_{\mu p} d_{\nu q} d_{\kappa r} d_{\lambda s} (pq|rs). \tag{3-2}$$

Yasuda[48] fue el primero en utilizar el potencial GPU para la aceleración de los cálculos de ERIS. En su trabajo, explica con detalle las limitaciones del uso de GPUs en su evaluación, y analiza con detalle los errores en la formación de las integrales de acuerdo a la implementación de su propuesta. Debido a las limitaciones en la memoria GPU y en la precisión de datos procesados por ésta (datos de precisión simple), Yasuda propuso un esquema híbrido GPU-Huésped para evaluar las ERIS con mayor valor absoluto (el cual estima a partir de la cota superior establecida por la desigualdad de Schwartz) usando precisión doble en el Huésped, y el resto en la GPU. Adicionalmente, tomando en cuenta la memoria disponible en el dispositivo, propuso el uso de la cuadratura Gauss-Rys[49] [50], introduciendo una nueva ecuación para el cálculo de las raíces y factores de peso, más apropiada para la arquitectura SIMD.

Posteriormente, Ufimtsev y Martínez[51] [52], desarrollaron kernels[5] CUDA para el cálculo de ERIS y la formación de la matriz de Fock que involucra orbitales s y p. En su trabajo, ellos proponen tres diferentes mapeos de la carga computacional a los hilos CUDA, concluyendo que la estrategia más apropiada consiste en el mapeo de tandas de integrales primitivas a un hilo

[5] De ahora en adelante, se hará uso del término "kernel" para referirse a las funciones CUDA que se ejecutan en el dispositivo GPU.

de acuerdo al momento angular de las funciones consideradas, para posteriormente construir la matriz de Fock a partir de ellas directamente (sin necesidad de calcular las integrales en la base de las funciones contraídas). Estos autores argumentan que esta estrategia es ideal para la implementación del procedimiento SCF en GPUs. Aunque este trabajo se limitó al cálculo de integrales de funciones primitivas s y p, DePrince y colaboradores[53] desarrollaron en el 2010, la implementación de la evaluación de integrales de momento angular superior (del tipo d hasta g), empleando la cuadratura de Rys, selección hecha con base en los bajos requerimientos de memoria del mismo.

Asimismo, Ufimtsev y Martinez han implementado el cálculo de las contribuciones coulómbicas y de intercambio a los gradientes analíticos HF con funciones base s y p.[54].

3.2.2 Cuadratura Numérica de Intercambio-Correlación

En la Aproximación del Gradiente Generalizado (GGA, por sus siglas en inglés) de DFT, el potencial de Intercambio-Correlación depende de la densidad electrónica ρ y su gradiente $\nabla\rho$, y es una función complicada en tres dimensiones.

Una cuadratura numérica es empleada para computar los elementos de la representación matricial del potencial:

$$V_{\mu\nu}^{XC(GGA)} = \int d\mathbf{r}\phi_\mu(\mathbf{r})v_{XC}^{GGA}\phi_\nu(\mathbf{r}) \approx \sum_k w_k\phi_\mu(\mathbf{r}_k)v_{XC}^{GGA}\phi_\nu(\mathbf{r}_k). \tag{3-3}$$

Esta cuadratura, es paralelizable en GPUs y Yasuda[55] fue el primero en utilizar esta plataforma para el cómputo de estos elementos. La estrategia que empleó fue que los pasos computacionales menos demandantes del algoritmo (generación de la cuadratura, evaluación de v_{XC}^{GGA} sobre ésta) son efectuados por la CPU, mientras que los pasos más demandantes del algoritmo (evaluación de ρ y $\nabla\rho$, así como el cálculo de 3-3 sobre la malla) son formuladas como productos de matrices y vectores, que pueden efectuarse sin complicaciones en la GPU.

3.2.3 Métodos de Correlación Electrónica *AB INITIO*

La forma tradicional para aproximar las soluciones a la ecuación de Schrödinger electrónica es mediante los métodos *ab initio*, basados en la función de onda. Entre las implementaciones

GPU dirigidas a la aceleración de estos métodos, se encuentran los siguientes:

3.2.3.1 Teoría de Perturbaciones Møller-Plesset en la Aproximación de la Resolución de la Identidad

La teoría de Perturbaciones Møller-Plesset a segundo orden (MP2) es el método computacional de correlación electrónica *ab initio* más popular y barato[56]. Asimismo, captura efectos de correlación de largo alcance (como la dispersión), característica de la que carecen los funcionales más populares[57], y por lo tanto, constituye una alternativa en la descripción de puentes de hidrógeno. Sin embargo, su elevado costo computacional restringe el uso de este método a sistemas de tamaño modesto.

El mayor costo computacional de cálculos MP2 es dominado por la transformación integral de la base de orbitales atómicos a la base de orbitales moleculares, el cual escala como $\mathcal{O}(N^5)$[6]. Esta transformación de cuatro índices puede ser evitada mediante la introducción de la aproximación integral de la Resolución de la Identidad (RI)[31], a través de la cual se requiere de una transformación integral de tres índices y se reduce el prefactor (es decir, se reduce la carga computacional, sin variar la complejidad del algoritmo), sin pérdida significativa de exactitud.

Aspuru-Guzik y colaboradores[58] [59], han realizado trabajos dirigidos a la aceleración de los cálculos RI-MP2. En ellos, los autores notaron que el paso computacionalmente más demandante de un cálculo RI-MP2 consiste esencialmente en multiplicaciones de matrices para aproximar las integrales moleculares de cuatro centros a partir de las integrales tricéntricas B_{ia}^{P}:

$$(ia|jb) \approx \sum_{P} B_{ia}^{P} B_{jb}^{P}. \tag{3-4}$$

En la implementación CPU sobre la que realizaron las modificaciones, el paso relativo a la ecuación 3-4, se lleva a cabo multiplicando una matriz de tamaño $N_{virt} \times N_{aux}$ (número de orbitales virtuales por número de bases auxiliares) por su transpuesta, para cada par ij de orbitales ocupados. Adicionalmente, para el mejor aprovechamiento de la GPU, las matrices

[6] En Ciencia de la Computación la notación *O grande* es usada para denotar la complejidad de un algoritmo de acuerdo a cómo responde en términos, por ejemplo, de tiempo de procesamiento o de requirimientos de *hardware*, a cambios en el tamaño del input. Esta notación no expresa explícitamente factores constantes y términos más pequeños. Así, por ejemplo, si consideramos el tiempo de procesamiento T de un algoritmo en función de una variable n, como $T(n) = 8n^3 + 2n^2 + n + 10$, entonces, en notación *O grande*, tenemos que $T(n) = \mathcal{O}(n^3)$

tienen que ser mayores a un umbral con el fin de minimizar el impacto de la latencia del bus PCI en la transferencia de datos CPU-GPU. Dependiendo del tamaño del sistema, proponen también minimizar dicho impacto al tratar varios pares ij conjuntamente.

Por otro lado, se desarrolló una biblioteca para la multiplicación de matrices con un tamaño superior al soportado por la GPU[59]. Mediante ésta, las matrices que se multiplican se descomponen en bloques, los cuales son multiplicados en la GPU usando la biblioteca CUBLAS[60], y los resultados son acumulados en la CPU. Asimismo, para mejorar la exactitud numérica del resultado, esta biblioteca implementa un modelo de precisión mixta, en el que los elementos de matriz más grandes en valor absoluto, son tratados con precisión doble en la CPU, y el resto es tratado con precisión sencilla en la GPU.

3.2.3.2 Monte Carlo Cuántico

El método de Monte Carlo Cuántico (QMC, por sus siglas en inglés) es uno de los más exactos para resolver la ecuación de Schrödinger independiente del tiempo. A diferencia de los métodos *ab initio* variacionales, QMC se basa en la evaluación estocástica de las integrales involucradas.

Anderson y sus colaboradores[61], han demostrado cómo acelerar cálculos QMC mediante la ejecución de kernels CUDA en las partes computacionalmente más intensas del algoritmo. Éstas son la evaluación de las funciones base sobre una malla numérica y multiplicaciones matriciales para la evaluación de los orbitales moleculares, sus gradientes y laplacianos.

Capítulo 4

Motivación, objetivos y metodología

4.1 Motivación

El método de estructura electrónica MP2 es restringido a sistemas de tamaño pequeño y modesto debido a su elevado costo computacional. Entre sus variantes, el método MP2 con la aproximación de Espines Opuestos Escalados (SOS–MP2, por sus siglas en inglés) representa una alternativa económica. Sin embargo, como se demuestra posteriormente en este trabajo, el tiempo de pared involucrado en la evaluación de energía de correlación en cálculos SOS-MP2, constituye el mayor porcentaje del tiempo total. Por lo tanto, es deseable la búsqueda de una estrategia que permita la reducción en la o las etapas más demandantes del algoritmo responsable del cómputo de esta energía.

El rendimiento computacional de las modernas Unidades de Procesamiento Gráfico (GPUs) y la existencia de un paradigma de programación de alto nivel que permite su empleo en aplicaciones numéricas, hacen de esta plataforma una buena alternativa para la reducción de los tiempos de ejecución en cálculos SOS-MP2.

Aunque la incorporación de MPI en el código fuente del paquete de química computacional sobre el que se llevó a a cabo este proyecto constituye una alternativa viable para la reducción del tiempo de cómputo, desafortunadamente el método aún no se encuentra paralelizado de esta forma. La correspondiente implementación MPI requiriría de mayor esfuerzo de programación, pues para maximizar las ventajas de este paradigma sería deseable el reparto de carga computacional entre varios procesadores de la evaluación de las integrales involucradas tanto

en la etapa SCF como en la evaluación de la energía SOS-MP2.

4.2 Objetivos

El objetivo de este trabajo consistió en la modificación del paso más demandante computacionalmente del algoritmo que calcula la energía de correlación SOS-MP2, para permitir la ejecución de éste en Unidades de Procesamiento Gráfico. El código sobre el que los cambios fueron realizados, está implementado en el paquete de química computacional Q-Chem [34].

Estas modificaciones fueron desarrolladas con el propósito de reducir el tiempo de ejecución de los cálculos que usan el método de estructura electrónica anteriormente mencionado, y al mismo tiempo, contribuir al desarrollo de Q-Chem como un código adaptable a los ambientes de cómputo heterogéneo modernos.

4.3 Metodología

La metodología seguida en este proyecto es resumida en los siguientes procedimientos:

- Realización de cálculos de referencia en moléculas modelo (alcanos) para respaldar la motivación de la paralelización del código. Estos cálculos indicaron el porcentaje del tiempo de pared total en cálculos de punto simple único (SP, por sus siglas en inglés) atribuídos a los pasos que componen el algoritmo, es decir, al método de campo autoconsistente y a cada uno de los pasos que conforman el cálculo de la energía de correlación SOS-MP2.

- Una vez identificado el paso más demandante del algoritmo, se procedió al desarrollo de códigos GPU modelo simples que emularan sus principales características. Estos códigos implementaron dos estrategias diferentes en el procesamiento de datos. Posteriormente se realizaron cálculos de referencia para comparar el rendimiento computacional entre ambas versiones.

- A la par de los códigos descritos en el párrafo anterior, se desarrolló un código MPI simple que realiza el mismo procesamiento numérico que los primeros. Esto fue llevado con el fin de comparar el rendimiento del procesamiento de datos en una GPU con el que se puede

obtener en una implementación que haga uso de varios procesadores CPU para efectuar el cómputo.

- Finalmente, se procedió a la implementación de la mejor estrategia encontrada en el segundo punto en el código fuente de Q-Chem. Las rutinas desarrolladas fueron probadas en moléculas de alcanos lineales de diferente longitud de cadena, y el *speedup* (aceleración) logrado fue evaluado en cada una de ellas. Se realizó esta evaluación empleando dos conjuntos base de Dunning: la cc-pVDZ y la cc-pVTZ.

Capítulo 5

Resultados

De acuerdo con la Guía de Mejores Prácticas en CUDA C, las optimizaciones de memoria constituyen el área más importante para mejorar el rendimiento de aplicaciones orientadas al cómputo numérico, en conjunto con la maximización del ancho de banda en la transferencia de datos entre la CPU y el dispositivo GPU. Esto es logrado comúnmente mediante la minimización de transferencias de datos, agrupando muchas transferencias pequeñas en una mayor o mejorando el ancho de banda entre el CPU y la GPU mediante el empleo de la llamada memoria no paginable (apartada en la CPU). Esta última opción tiene la ventaja adicional de permitir la ejecución simultánea de varios kernels en el dispositivo, y efectuar transferencias asíncronas CPU-GPU (si el dispositivo cuenta con dichas capacidades, aunque algunos dispositivos con capacidad de cómputo[1] 2.x, y superiores soportan estas características[62]).

En este trabajo, incorporamos en el código el copiado asíncrono de datos, con el objetivo de traslapar el cómputo con el copiado de memoria y dar lugar a una ejecución concurrente potencial de kernels. Aunque una solución alternativa consiste en reducir únicamente transferencias de datos (lo que se puede llevar a cabo mediante agrupaciones de varias transferencias en una sola y una serie posterior de ejecuciones secuenciales del kernel sobre segmentos del buffer[2]), esta aproximación no permite un enmascaramiento[3] de la comunicación CPU-GPU, y

[1] La capacidad de cómputo (*compute capability*) hace referencia a la versión del *hardware*.

[2] Un buffer es una región de memoria empleada para almacenar datos temporales, mientras son movidos de una dirección a otra.

[3] El término "enmascaramiento", en este contexto significa que el tiempo invertido en un proceso, puede traslaparse con otro de mayor utilidad. En el caso del cṁputo heterogéneo con GPUs, es usual la búsqueda de estrategias que traslapen el tiempo de comunicación de la CPU con el dispositivo, con procesamiento de datos.

```
for (int i = 0; i < nTandas; ++i) {
  int offset = i * streamSize;

  cudaMemcpy(&d_a[offset], &a[offset], streamBytes);

  kernel<<NumBlocks, ThreadsPerBlock >>(d_a, offset);

  cudaMemcpy(&a[offset], &d_a[offset], streamBytes);
}
```

Figura 5.1: Ejemplo que ilustra transferencia de datos no asíncrona entre el Huésped (CPU) y el dispositivo (GPU).

adicionalmente, esta opción no ofrece una mejora en el ancho de banda, tal y como lo hace un esquema de copiado asíncrono de datos.

Para obtener un esquema de comunicación asíncrona, se emplea una versión no bloqueadora de transferencia de memoria entre la CPU y GPU. Esto es llevado a cabo mediante el uso de los llamados *streams*, que son objetos creados en un contexto CUDA determinado y que permiten el procesamiento de secuencias de operaciones en un orden específico en el dispositivo GPU. La ventaja del uso de estos objetos es que *streams* diferentes pueden realizar operaciones sobre porciones diferentes de segmentos de memoria y, por lo tanto, pueden traslaparse (siempre y cuando el dispositivo GPU cuente con esta característica habilitada), lo que se traduce en un porcentaje de ocupación mayor en la memoria del dispositivo GPU.

Las diferencia entre los esquemas de comunicación asíncrona y no asíncrona pueden ilustrarse en los códigos mostrados en las figuras 5.1 y 5.2, en los que se realiza el procesamiento de varios conjuntos de datos. En la figura 5.1 (que corresponde al caso de comunicación no asíncrona), d_a y a son apuntadores en la memoria del huésped, y de la GPU, respectivamente. La estrategia ilustrada garantiza la partición de un *buffer* en segmentos de igual tamaño, que son procesados (copiados a memoria, posteriormente procesados mediante la llamada al *kernel* y finalmente copiados a la memoria del huésped), de forma secuencial. La naturaleza secuencial de este procesamiento se debe a que las funciones de copiado de memoria son *llamadas bloqueadoras*, dicho de otra forma, el flujo del algoritmo en el huésped es pausado hasta la completitud de esta función. Aunque la ejecución del kernel es una *llamada no bloqueadora*, el requerimiento del copiado GPU-CPU necesariamente introduce una barrera temporal en el flujo del algoritmo.

```
stream=malloc(nStreams*sizeof(cudaStream_t));

for (i=0;i<nStreams;i++){
cudaStreamCreate(stream+i);
}

for (int i = 0; i < nStreams; ++i) {
  int offset = i * streamSize;

  cudaMemcpyAsync(&d_a[offset], &a[offset],
  streamBytes, stream[i]);

  kernel<<NumBlocks,ThreadsPerBlock,0,stream[i]>>(d_a, offset);

  cudaMemcpyAsync(&a[offset], &d_a[offset],
  streamBytes, stream[i]);
}
```

Figura 5.2: Versión Asíncrona 1. Ejemplo de procesamiento de datos con una GPU mediante comunicación no bloqueadora.

Por otro lado, y con el fin maximizar el rendimiento computacional, es posible emplear funciones de copiado de memoria asíncronas, las cuales son no bloqueadoras. Lo anterior es ilustrado en la figura 5.2. En la Figura 5.2, se muestra una estrategia para llevar a cabo el copiado de datos al dispositivo GPU de forma asíncrona, es decir, mediante llamadas a funciones que ceden el control al huésped inmediatamente después de ser invocadas, en diferentes "canales" de operaciones. De esta manera, es posible traslapar el copiado de memoria con el cómputo en la GPU, tal y como expone en la Figura 5.3. Para lograr lo anterior, las siguientes condiciones deben ser satisfechas:

- Los apuntadores a la memoria CPU deben ser apartados mediante la función *cudaHostAlloc*, la cual aparta memoria no paginable.[4]

- Es necesaria la inicialización de objetos conocidos como *streams* en el contexto CUDA. Cuando el dispositivo soporta ejecución concurrente de kernels, las instrucciones GPU pueden traslaparse, y por lo tanto, se consigue un incremento significativo en el rendi-

[4]Este tipo de memoria es más accesible a la Memoria de Acceso Directo (DMA, por sus siglas en inglés, que es intermediaria en la comunicación CPU-GPU), que la memoria paginable, empleada por la CPU al hacer uso de la función *cudaMemcpy*.

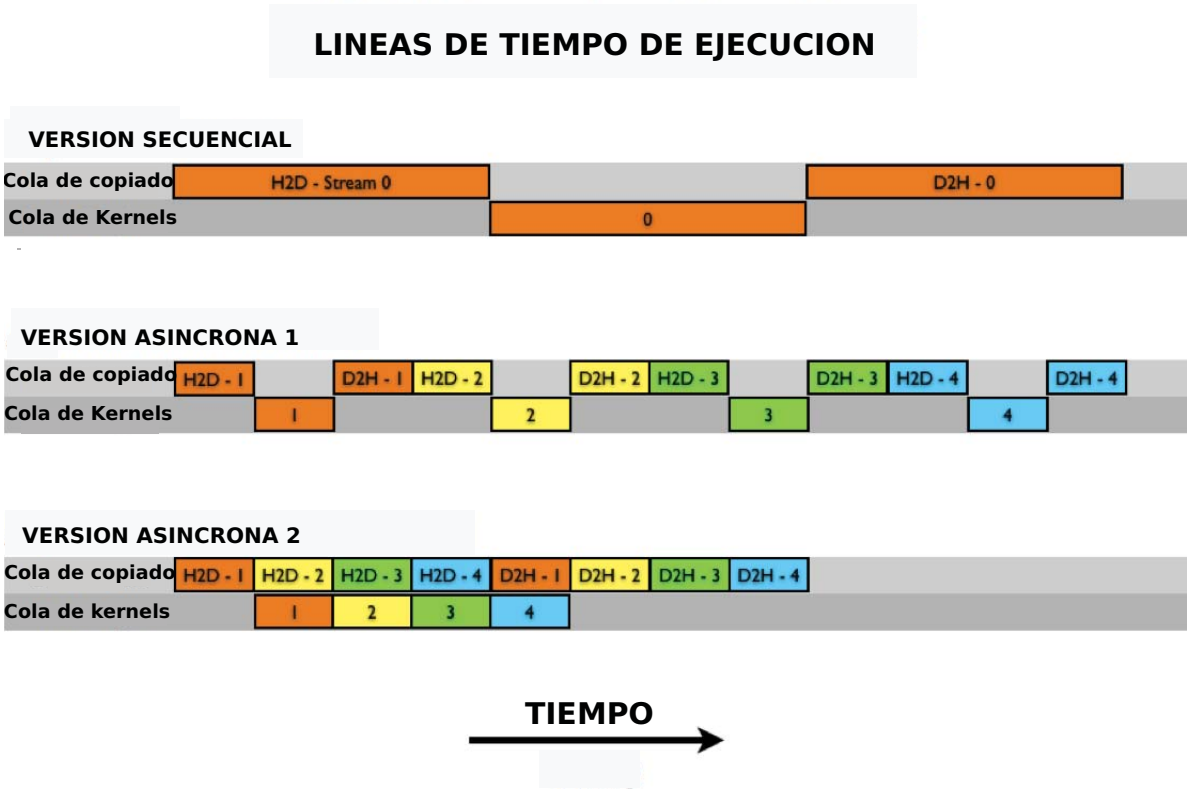

Figura 5.3: Esquema comparativo de diferentes versiones de cómputo GPU asíncrono. Las versiones presentadas son mostradas en las figuras 5.2 (Versión 1) y 5.4 (Versión 2).

miento.

- Las funciones asociadas a la comunicación GPU deben ser sustituidas por su versión asíncrona.

Aquí es importante notar que, dada la naturaleza no bloqueadora de las funciones involucradas, es posible reescribir el código expuesto en la figura 5.2 tal y como se ilustra en la figura 5.4. Ambos procedimientos producen los mismos resultados, sin embargo, se comportan de diferente forma dependiendo de la generación del dispositivo empleado[63] (ver Figura 5.3). Estas diferencias residen en la disponibilidad en el dispositivo de motores (*engines*) de copiado Huésped-Dispositivo (H-D) y Dispositivo-Huésped (D-H), así como en la existencia del soporte de ejecución concurrente de kernels. Así pues, dada la descontinuidad de las generaciones GPU que no cuentan con la capacidad de copiado de memoria asíncrono (es decir, que carecen de los *engines* para el copiado asíncrono H-D y D-H), y la amplia disponibilidad de dispositivos que

```
stream=malloc(nStreams*sizeof(cudaStream_t));

for (i=0;i<nStreams;i++){
cudaStreamCreate(stream+i);
}

int offset;
for ( i = 0; i < nStreams; ++i) {
   offset = i * streamSize;

  cudaMemcpyAsync(&d_a[offset], &a[offset],
  streamBytes, stream[i]);
  }

  for(i=0;i<nStreams;i++){
  offset = i * streamSize;
    kernel<<NumBlocks,ThreadsPerBlock,0,stream[i]>>(d_a, offset);
  }

  for{i=0;i<nStreams;i++}{
  cudaMemcpyAsync(&a[offset], &d_a[offset],
  streamBytes, stream[i]);
}
```

Figura 5.4: Versión Asíncrona 2. Procesamiento de datos alternativo a la versión 1 (figura 5.2) mostrada arriba, empleando comunicación no bloqueadora.

sí cuentan con ésta, optamos por emplear un esquema similar al ejemplo ilustrado en la figura 5.2, dado que es el más adaptado a la generación GPU con capacidad de cómputo 2.x[5]

5.1 Evaluación del Código Serial

Para justificar la paralelización del algoritmo de la evaluación de la energía de correlación SOS-MP2, se llevaron a cabo cálculos de referencia para determinar la dependencia de la fracción del tiempo de pared total invertido en su evaluación, conforme el tamaño del sistema aumenta. Estos cálculos se realizaron sobre alcanos de cadena lineal, con diferente longitud. La Figura 5.5 presenta un gráfico que muestra dicha variación. En ésta, se observa que para alcanos con una longitud de cadena mayor o igual a 30 carbonos, el tiempo de ejecución asociado a la evaluación

[5]Para la nueva generación con capacidad de cómputo 3.x, no hay modificaciones en el rendimiento al emplear cualquiera de los esquemas de comunicación presentados en las figuras 5.2 y 5.4.

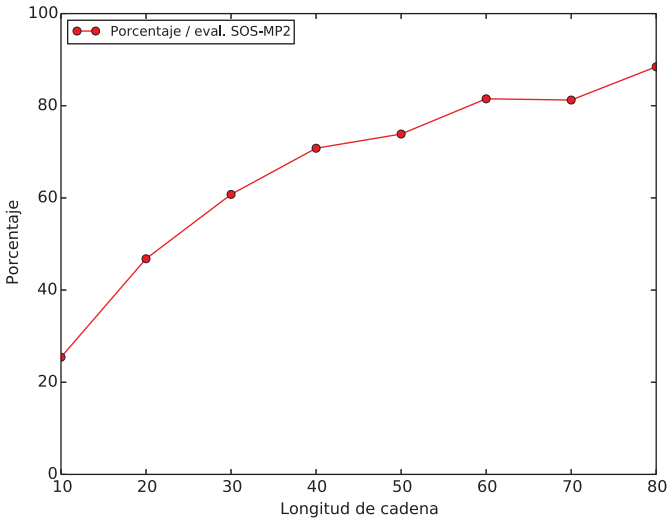

Figura 5.5: Tendencia en el porcentaje del tiempo de pared total invertido en la evaluación de la energía de correlación de cálculos de punto simple único en alcanos lineales.

de energía de correlación constituye más del 50% del tiempo de pared total (el porcentaje restante corresponde al método SCF, durante el que se construyen los orbitales moleculares que son usados posteriormente en el método SOS-MP2), y llega a constituir casi el 90% para la cadena de 80 carbonos. Esta tendencia es un incentivo para la búsqueda de una solución que permita una reducción en el tiempo de ejecución del algoritmo SOS-MP2.

Desplazando ahora el enfoque al algoritmo de evaluación de energía SOS-MP2, de los cálculos de referencia se analizaron los tiempos de ejecución de los pasos de los cálculos de referencia involucrados. Los resultados obtenidos son expuestos en la Figura 5.6.

De estos resultados, es notorio que el tiempo de pared de la evaluación de energía es dominado por la evaluación de las integrales tricéntricas $(ia|P)$ y la formación de las matrices $\bar{X}^{\alpha}_{KL}$. Dado que este último paso escala con un cuarto orden respecto el tamaño molecular y es el más costoso computacionalmente, se procedió a investigar la posibilidad de emplear GPUs para reducir el tiempo de pared asociado a dicho paso.

Primeramente, con el propósito de establecer metas en cuanto al *speedup* que puede introducirse mediante paralelización en el algoritmo, se analizó la tendencia en el porcentaje del tiempo de pared total de éste atribuído al quinto paso, cuando el tamaño del sistema aumenta. De

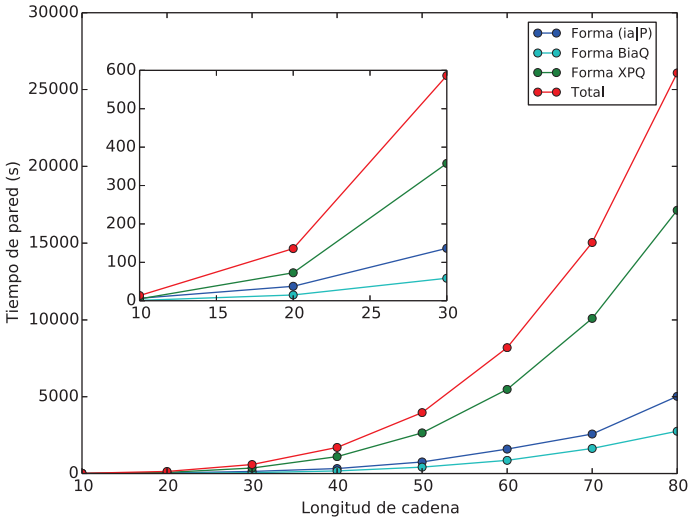

Figura 5.6: Comparación de los tiempos de pared seriales de diferentes pasos del algoritmo de evaluación de energía SOS-MP2, para alcanos lineales, empleando la base de Dunning cc-pVDZ y la base auxiliar rimp2-cc-pVDZ. Sólo las etapas que constituyen los mayores porcentajes del tiempo de pared total del algoritmo SOS-MP2 son mostradas: formación de integrales $(ia|P)$, formación de matrices B_{ia}^{Q}, y formación de matrices $\overline{X}_{KL}^{\alpha}$.

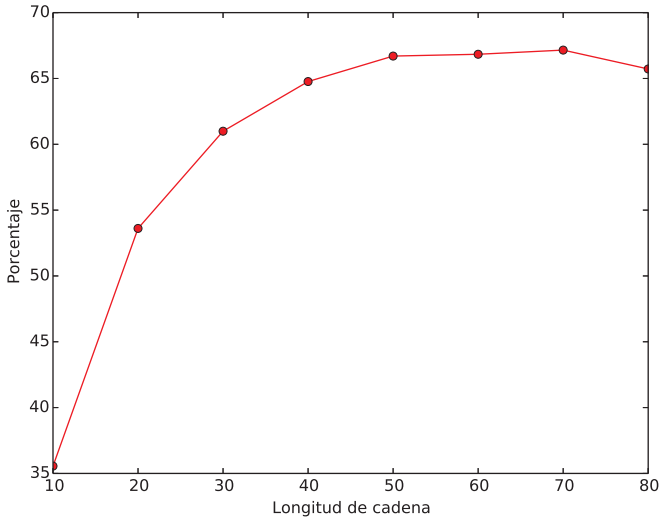

Figura 5.7: Porcentaje del tiempo de pared invertido en la formación de las matrices **X**, respecto al tiempo de pared del algoritmo SOS-MP2

esa tendencia, se encuentra que el máximo procentaje del tiempo de pared representado por la formación de las matrices **X** es del 65% (ver Figura 5.7), y la tendencia parece prevalecer para sistemas más grandes que los tratados aquí. Considerando dicha proporción como paralelizable, y asumiendo que el 35% restante es secuencial, se puede establecer una cota superior en el *speedup* que se espera observar después de la paralelización, mediante el empleo de la ley de Amdahl[62],

$$S = \frac{1}{(1-P) + \frac{P}{N}}. \tag{5-1}$$

En esta ecuación, P es la fracción del tiempo de ejecución secuencial total invertido en el segmento de código que es paralelizable, y N es el número de procesos sobre los que el código paralelo es ejecutado.

Usando la Ecuación (5-1), se encuentra que el máximo *speedup* esperado después de la implementación de paralelización, es 2.86. Si este *speedup* es logrado en el algoritmo, se espera un *speedup* de 2.0 en el cálculo SP, suponiendo que el tiempo de ejecución de la evaluación de energía constituya el 80% del tiempo de ejecución total (ver figura 5.5.)

5.2 Código Modelo y Comparación con MPI

Como ya se expuso en párrafos anteriores, el empleo de transferencias de datos CPU-GPU asíncronas para lograr concurrencia en el cómputo y copiado de memoria (y posiblemente, concurrencia de diferentes kernels) se traduce en una ocupación de memoria más eficiente y con mayor rendimiento. Tomando en consideración dicha posibilidad, se investigaron los efectos al usar una GPU como coprocesador, con y sin el uso de comunicación asíncrona en un código similar al encontrado en Q-Chem. Para ello, fue llevado a cabo un análisis de rendimiento en multiplicación de matrices usando CUBLAS[60] (la biblioteca de subrutinas de álgebra lineal de NVIDIA CUDA), y la función cblas_dgemm[64] de la biblioteca MKL de Intel. El programa escrito para efectuar este propósito fue compilado usando la biblioteca MKL, el compilador icc versión 13.0.1 y CUDA, versión 6.0. Los cálculos fueron llevados a cabo empleando un procesador CPU Intel^{TM}Xeon^{TM}a 2.60 GHz, y una GPU Tesla M2090 (capacidad de cómputo 2.0).

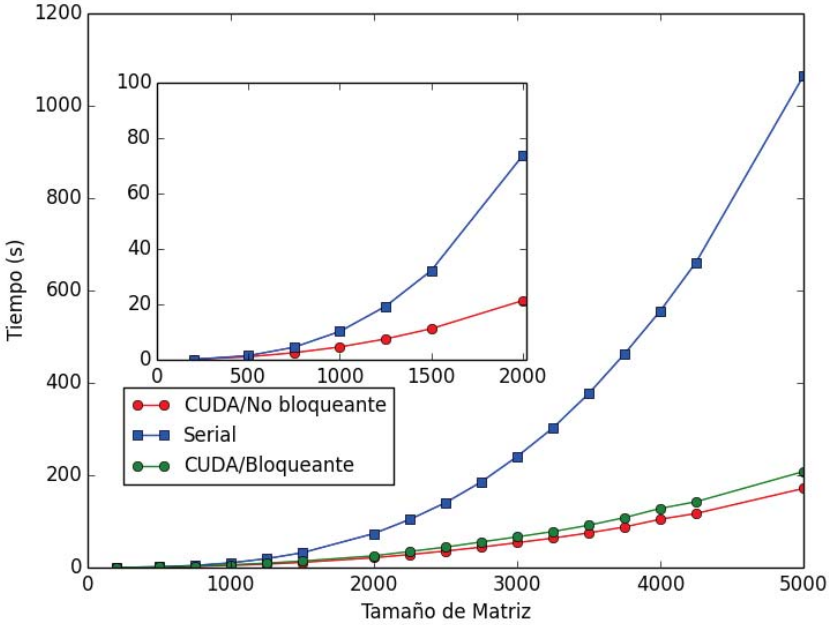

Figura 5.8: Comparación de tiempos de pared en multiplicación de matrices cuadradas, usando a) una GPU como coprocesador, sin emplear comunicación no bloqueadora, b) la función cblas_dgemm de la biblioteca MKL, y c) una GPU como coprocesador y con comunicación no bloqueadora. Los resultados mostrados consideran el procesamiento de 200 multiplicaciones, y fueron promediados sobre 10 cálculos diferentes.

Dando más detalles respecto al código empleado en este análisis (el cual puede ser consultado en el Anexo II de esta tesis), se hizo un esfuerzo por simular las características principales del paso correspondiente a la formación de las matrices **X**, de tal forma que éste inicializa un arreglo grande de datos (matriz) con números aleatorios, dentro de un bucle con un número de ciclos determinado. Dentro de este bucle, la matriz es multiplicada por su transpuesta y adicionada a una matriz de prueba. Los resultados del análisis expuesto son representados en la Figura 5.8, y la comparación de los *speedups*[6] correspondientes respecto a la versión serial, son ilustrados en la Figura 5.9.

El traslape del cómputo con el copiado de datos entre la memoria Huésped-Dispositivo, es corroborado al efectuar el perfilamiento de las llamadas a funciones CUDA (para lo cual, se

[6]El *speedup* se define como la razón entre el tiempo de pared de la versión serial y la versión paralela del mismo código.

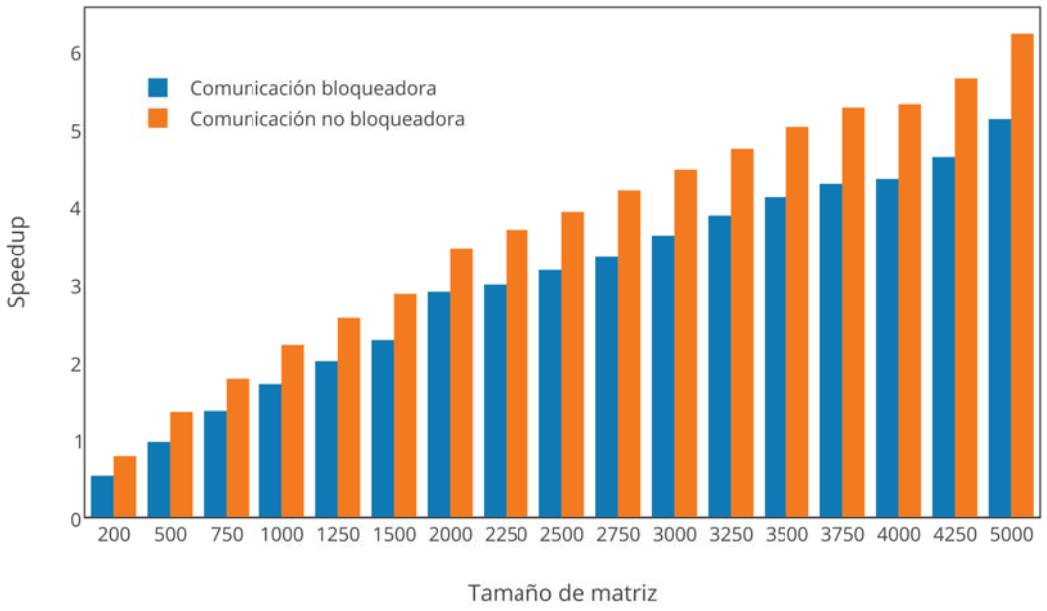

Figura 5.9: Comparación de los *speedups* logrados al emplear comunicación bloqueadora y no bloqueadora en el procesamiento de la multiplicación de matrices con una GPU.

hizo uso de la herramienta nvprof de NVIDIA), ilustrado en la Figura 5.10.

De este análisis se encontró, que cuando se emplea la GPU no hay *speedup* cuando la dimensión de la matriz cuadrada es pequeña (en este caso, igual a 200), dado que la latencia del bus PCI en esta situación representa una fracción significativa del tiempo de pared. Por otro lado, para matrices de dimensiones mayores, se observan *speedups* que se incrementan al incrementarse el tamaño de la matriz en cuestión, y éstos llegan a ser 5 y 6, para las versiones bloqueadora y no bloqueadora, respectivamente.

Para introducir concurrencia en la multiplicación de matrices y los procesos de comunicación, se propuso un esquema de procesamiento por tandas, en el cual grupos de N matrices son inicializadas y después procesadas y multiplicadas por su transpuesta dentro de una sola llamada a una función que hace uso de CUBLAS, de forma asíncrona para cada matriz de la tanda. El valor de N en este estudio fue igualado a 3 para todos los casos.

Adicionalmente, con fines comparativos, un código similar fue escrito empleando la biblioteca estándar de transferencia de mensajes entre procesos (MPI[40], por sus siglas en inglés), el cual puede ser consultado en el Anexo II de esta tesis. Este código realiza el procesamiento de

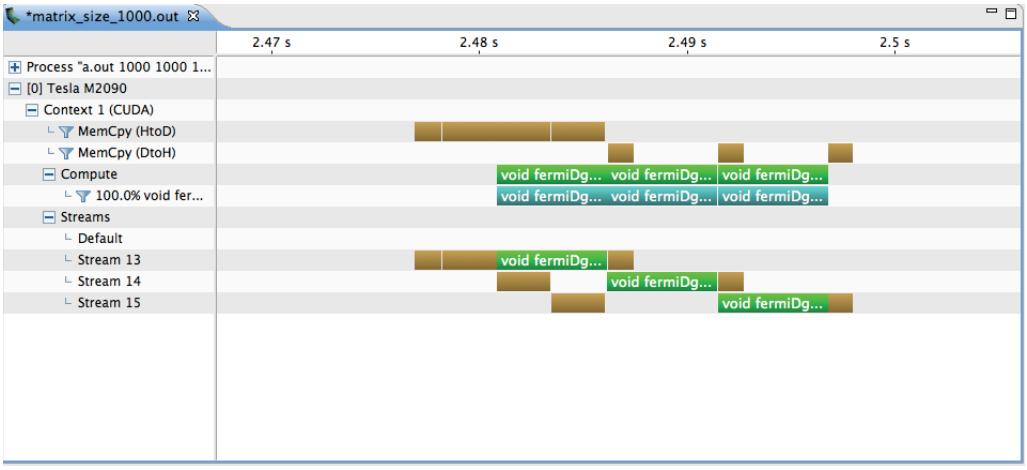

Figura 5.10: Línea de tiempo que muestra el traslape entre los procesos de copiado de memoria entre la CPU y el dispositivo, en la multiplicación de 2 matrices de 1000×1000 elementos.

un determinado número de matrices repartiendo de forma balanceada la carga computacional entre un número arbitrario de procesos. Los resultados de las pruebas de rendimiento de este código, son mostrados en la Figura 5.11. De estos resultados vale la pena hacer notar que mediante este esquema de paralelización es posible reducir el tiempo de pared en el procesamiento de datos, obteniéndose un *speedup* similar al obtenido al incorporar una GPU, cuando el cómputo se efectúa entre 6 procesadores. De esta forma, obtenemos una comparación en el rendimiento usando dos paradigmas diferentes de programación paralela, lo que permitiría realizar posteriores análisis de costo-beneficio en el empleo de éstos. Por ejemplo, Betkaoui [65], ha reportado que en aplicaciones de cómputo numérico, específicamente en operaciones de tipo 3 de la biblioteca BLAS (multiplicaciones de matrices) con elementos de punto flotante, la GPU requiere de menos cantidad de energía por operación de punto flotante en comparación con la implementación paralela sobre CPUs.

Cabe recordar que las razones por las que se llevó a cabo la programación usando CUDA, son expuestas en la sección 4.1, y además tomando en consideración que las nuevas versiones CUDA pueden usarse en un contexto MPI, es razonable realizar en primer lugar una implementación considerando CUDA, y garantizar el buen funcionamiento del código serial, para después considerar una paralelización que haga uso de MPI.

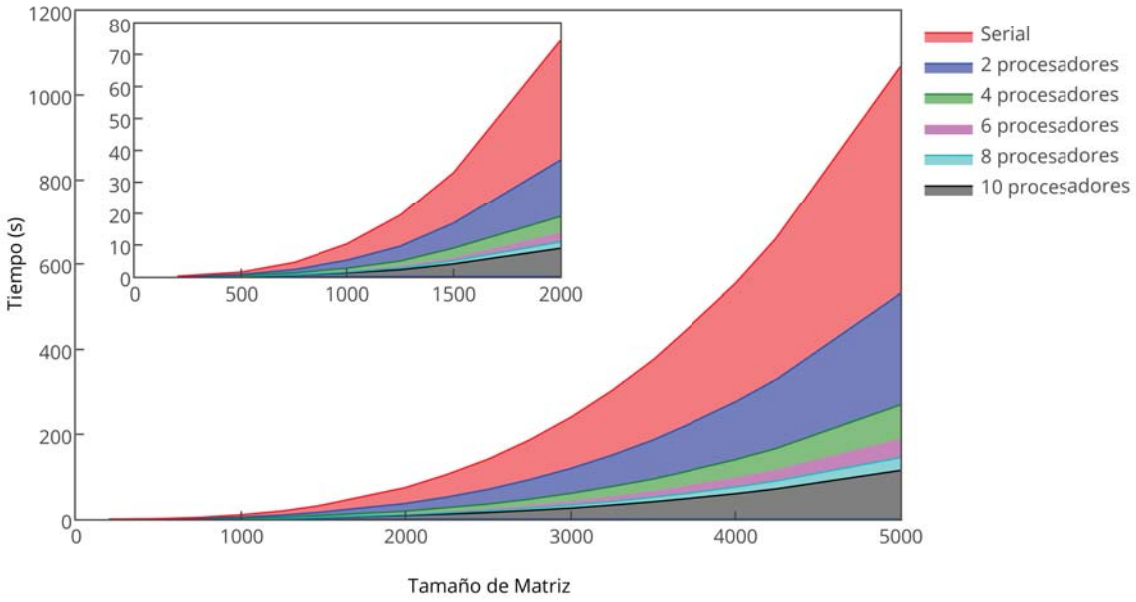

Figura 5.11: Rendimiento del código MPI en la multiplicación de 100 matrices cuadradas. Los datos mostrados son resultado de promediar sobre 10 cálculos.

5.3 Implementación Computacional

Con motivo en estos resultados, en Q-Chem se implementaron una serie de rutinas dentro del código que es responsable del cómputo de las energías SOS-MP2. Estas rutinas efectúan la inicialización de los *streams*, la multiplicación de matrices usando CUBLAS en un esquema de tandas y de forma asíncrona, y la destrucción de los *streams*. El algoritmo modificado es descrito con mayor detalle en el pseudocódigo ilustrado en la figura 5.12.

Las características principales del algoritmo descrito arriba son las siguientes:

- El valor del tamaño de la tanda, $NBatchsize$, es establecido de acuerdo a un parámetro fijado en el programa y que establece el número máximo de datos de tipo punto flotante de doble precisión a procesar en la GPU. Algunos perfilamientos que muestran cómo varía el tamaño de la tanda con el tamaño del sistema son mostrados en el Anexo I.

Algoritmo 2: Formación de la matriz **X** con CUDA.

```
Result: Formación de las matrices X con CUDA
Inicialización del contexto CUDA;
Cálculo de NBatchsize;
Creación de Streams;
for i ← 0 to 7 do
    Aparta memoria no paginable en la CPU para las matrices B y X;
    for j ← 0 to O in NBatchsize steps do
        Lee Batchsize matrices B;
        for k ← 0 to Batchsize do
            Escala los coeficientes B_ia^Q;
        for k ← 0 to Batchsize do
            if matrixsize < threshold then
                Multiplica las matrices por segmentos;
            Transferencia asíncrona CPU-GPU en el stream[i];
            Ejecución del kernel en el stream[i];
            X ← Transferencia asíncrona GPU-CPU en streams[i]
    Incrementa Energia
```

Figura 5.12: Formación de la matriz **X** empleando una unidad de procesamiento gráfico. En este algoritmo el bucle más externo corre sobre los puntos de la cuadratura numérica introducidos en la ecuación 2-82.

- El número de *streams* que se crean en el contexto es igual a $NBatchsize$, de tal forma que cada matriz de la tanda es procesada de forma independiente en cada stream, con el fin de enmascarar las comunicaciones CPU-GPU y GPU-CPU.

- Cuando las matrices a considerar son más grandes que las establecidas por el parámetro anteriormente mencionado (lo que produce que el programa establezca $NBatchsize$ igual a 1), éstas son divididas en la memoria CPU y la multiplicación de matrices se efectúa por segmentos en la memoria GPU, mediante una rutina implementada en una biblioteca en desarrollo en Q-Chem ([59])[7]. Un ejemplo de perfilamiento que muestra esta división de matrices por la rutina mencionada, es mostrado en el Anexo I.

En este trabajo se efectuó la compilación del código fuente de Q-Chem con los compiladores Intel 2013, por lo que las funciones dedicadas a operaciones de álgebra lineal (por ejemplo,

[7]Cabe resaltar que al incorporar esta rutina en el algoritmo, se encontró un error en la programación de la misma, el cual fue resuelto y la corrección fue incorporada al código.

multiplicación vector-escalar, producto punto, multiplicación de matrices.) son llevadas a cabo mediante las funciones que conforman la biblioteca MKL de Intel.
Con el objeto de evaluar la mejora que se introduce al emplear un esquema de comunicación no bloqueadora en el procesamiento de datos, se introdujo en el código una simple sustitución de la rutina que realiza la multiplicación de matrices (empleando la función serial cblas_dgemm) correspondiente a la Ecuación 2-88 por su análoga de la biblioteca CUBLAS (cublas_dgemm), que como se ha justificado en párrafos anteriores, corresponde al paso más demandante del algoritmo. Unos cálculos de referencia fueron llevados a cabo empleando esta aproximación usando alcanos lineales como referencia y la base doble-ζ cc-pVDZ, y de la misma forma, se llevaron a cabo los mismos cálculos empleando esta vez el esquema presentado en el pseudocódigo anterior. Los resultados de dichos cálculos de referencia son mostrados en la figura 5.13.
Además, para hacer más evidente la diferencia en el desempeño, los *speedups* logrados en este paso del algoritmo con ambos esquemas, son mostrados en la figura 5.14.

Como consecuencia de esta aceleración en la evaluación de las matrices $\mathbf{X}$, se obtienen los *speedups* en la evaluación de la energía de correlación SOS-MP2 que se muestran en la Figura 5.15, en donde es notorio que los cálculos de evaluación de energía SOS-MP2 se ejecutan casi tres veces más rápido que su versión serial, para los mejores casos. Por otro lado, dado que los cálculos de química computacional adquieren mayor exactitud al aumentar el tamaño de la base, los cálculos anteriores fueron repetidos utilizando una base triple-ζ y la base auxiliar rimp2-cc-pVTZ. Los resultados de estos cálculos de referencia son mostrados gráficamente en la figura 5.16, en donde se observa que la ejecución del cálculo de energía de correlación es alrededor de 2.5 veces más rápido que su versión serial, para el caso de los sistemas más grandes. De estos resultados, es notorio que las aceleraciones para la base triple-ζ son en general menores a los obtenidos en el caso de los cálculos efectuados con la base doble-ζ. Analizando con detalle los resultados, se encontró que hay una variación en el tiempo de pared asociado a la formación de las integrales $(ia|P)$ mayor en el caso de los cálculos triple-ζ (los tiempos tienden a ser mayores que sus análogos seriales), lo que se traduce en una disminución en la aceleración de los mismos. Después de realizar un seguimiento en una serie de cálculos mediante las herramientas de monitoreo de linux (top, free, ps, dstat), se encontró que los accesos de entrada/salida (debidos al almacenamiento de las integrales en archivos temporales) para los cálculos paralelizados,

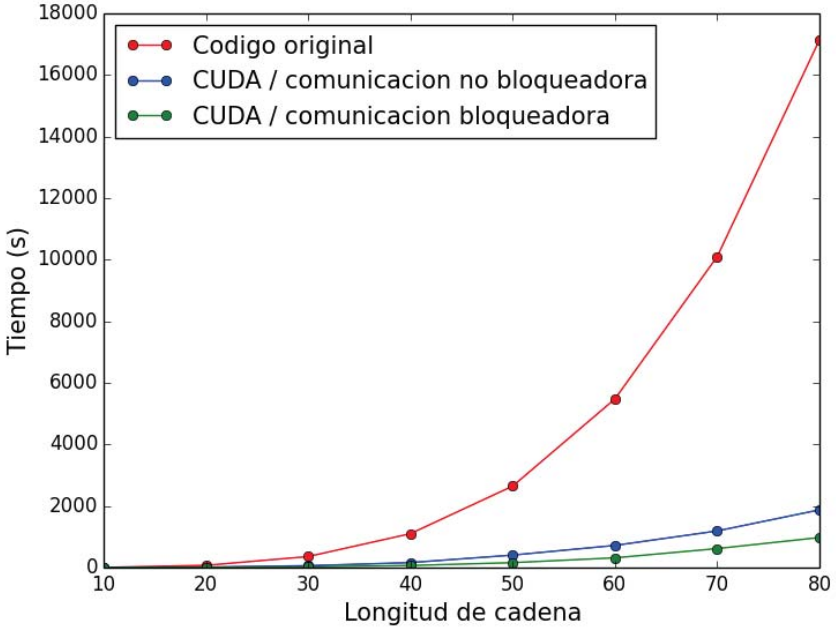

Figura 5.13: Comparación de los tiempos de ejecución en la evaluación de energía de las versiones de Q-Chem con y sin comunicación no bloqueadora, en conjunto con el tiempo de ejecución encontrado en el código original, para alcanos lineales, usando las base doble-ζ cc-pVDZ/rimp2-cc-pVDZ. El valor del tamaño de la tanda para el caso de la comunicación no bloqueadora fue fijado en cinco. Sólo se muestra el tiempo para la formación de las matrices $\mathbf{X}$.

fueron, en promedio, más lentos que los seriales. Sin embargo, no existe una razón por la cual las modificaciones hechas al código interfieran con estos accesos, dado que las modificaciones entran en juego después de la completitud de la formación de estas integrales , ver sección 3.3 Teoría SOS-MP2, por lo que concluímos que el comportamiento del *hardware* es responsable de estas variaciones, y no se debe a la implementación computacional descrita.

5.4 Calidad numérica

Dado que las rutinas desarrolladas están programadas para realizar todas las operaciones con datos de punto flotante con precisión doble, los resultados obtenidos al hacer uso de una GPU son idénticos a los obtenidos al realizar los cálculos completamente con una CPU. A excepción de unos pocos casos, los resultados variaron de forma no significativa. Por ejemplo, en el cómputo

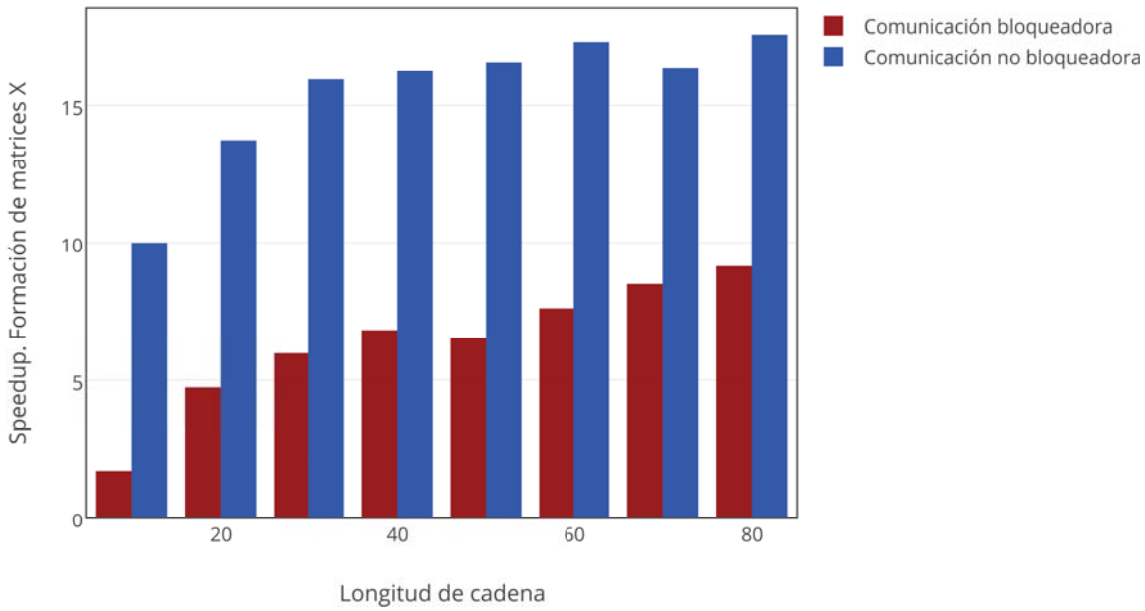

Figura 5.14: Comparación de speedups entre la versión bloqueadora y no bloqueadora, considerando solamente la formación de las matrices **X**, del algoritmo de evaluación de energía SOS-MP2, de alcanos lineales. La base que se empleó fue la doble-ζ cc-pVDZ

de la energía de correlación en el alcano de 70 carbonos empleando una base cc-pVTZ/rimp2-cc-pVTZ, la diferencia entre las energías obtenidas sin emplear y al usar una GPU fue de 4×10^{-4} Hartrees (0.26 kcal mol^{-1}).

Con el fin de comparar los valores de energía calculados con el método SOS-MP2 y la base cc-pVDZ/rimp2-cc-pVDZ, con los obtenidos mediante cálculos MP2/cc-pVDZ, se determinó el valor de la diferencia de energía de correlación por partícula[8] obtenido en cada método, para cada alcano considerado en este estudio.[9] Los resultados son mostrados en la figura 5.17.

[8] Es decir, la razón entre la energía de correlación y el número de electrones de la molécula.

[9] A excepción del alcano de 80 carbonos, dado que no fue posible calcular la energía MP2 con Q-Chem, posiblemente por un error en la programación del método.

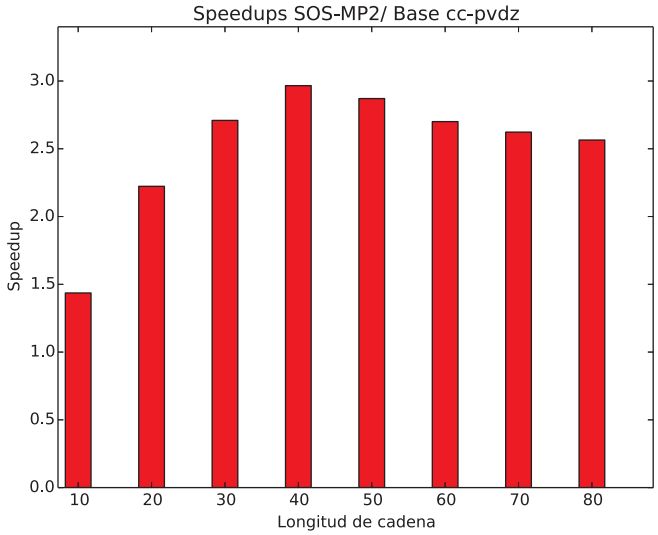

Figura 5.15: Speedups logrados como función del tamaño de la cadena de alcanos lineales, en el algoritmo responsable del cómputo de la energía SOS-MP2. La base empleada en los cálculos fue doble-ζ cc-pVDZ

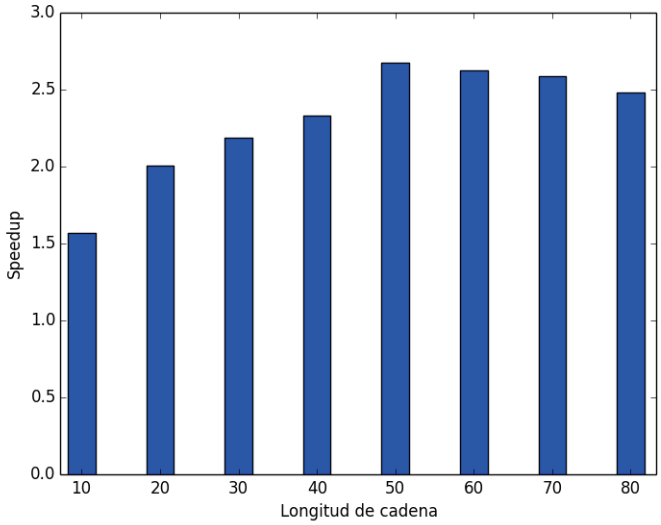

Figura 5.16: Speedups logrados como función del tamaño de la cadena de alcanos lineales, en el algoritmo responsable del cómputo de las energías SOS-MP2. La base empleada en los cálculos de referencia fue la triple-ζ

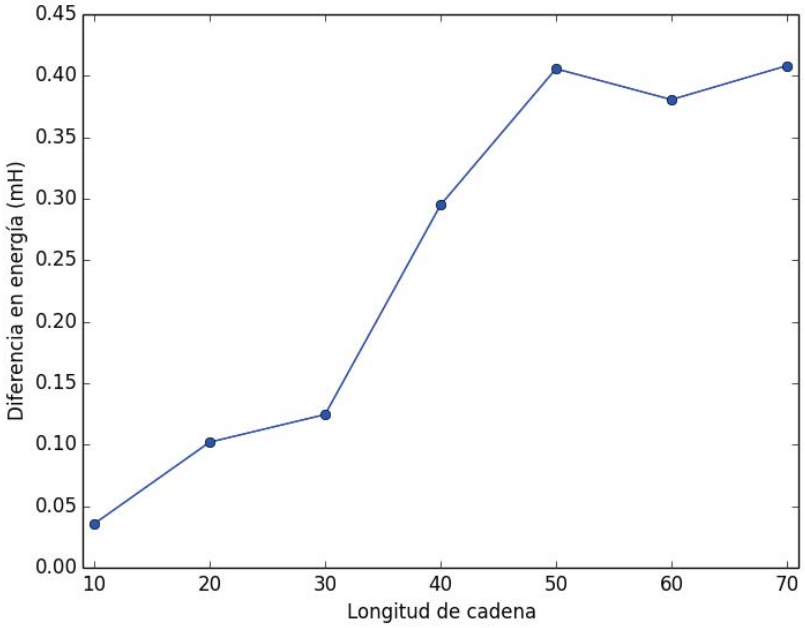

Figura 5.17: Valor de la diferencia entre la energía de correlación por partícula obtenida mediante el método SOS-MP2, base cc-pVDZ/rimp2-cc-pVDZ ($E_{part}^{SOS-MP2}$) y la obtenida mediante por el método MP2/cc-pVDZ (E_{part}^{MP2}). Los valores mostrados corresponden a $E_{part}^{SOS-MP2} - E_{part}^{MP2}$.

Capítulo 6

Conclusiones

Las conclusiones que se derivan de este trabajo son resumidas en los siguientes puntos:

- Los códigos modelo diseñados confirman que el rendimiento numérico logrado con la arquitectura de las Unidades de Procesamiento Gráfico supera el rendimiento en la mutiplicación de matrices al emplear la biblioteca MKL, caracterizada por una buena optimización para procesadores Intel. Además, este rendimiento es mejorado cuando es incorporada una comunicación no bloqueadora en el tratamiento de datos, lo que permite enmascarar el tiempo dedicado a la transferencia de información entre la GPU y CPU, y además, hay en una mejora en el ancho de banda de transferencia de los mismos. De esta forma, es posible acelerar también cálculos de sistemas pequeños, que de otra manera, serían más lentos debido a la latencia del bus PCI.

- El código modelo que incluye el procesamiento de matrices empleando MPI, predice que el rendimiento conseguido con 6 procesadores es aproximadamente el mismo que al emplear una GPU para efectuar cálculos de este tipo. Esto puede tomarse como punto de partida para llevar a cabo un estudio detallado sobre el costo-beneficio del empleo de uno o el otro paradigma de programación, puesto que ambos tienen diferentes requerimientos de energía.

- Al incorporar el modelo no bloqueador en el código fuente de Q-Chem, se obtiene una mejora significativa en el *speedup* del paso más demandante del algoritmo, en comparación

con un modelo bloqueador, y por lo tanto, se hace un uso más eficiente del *hardware* al aprovechar esta característica.

- Aunque en principio, la utilidad de la GPU se ve limitada por su capacidad de memoria relativamente pequeña (la GPU Tesla M2090 tiene una capacidad de 2 GB), incapaz de procesar matrices muy grandes, este problema ha sido solucionado al incorporar en el código una rutina implementada en una biblioteca en desarrollo de Q-Chem. Así, una vez que se realicen cálculos sobre sistemas suficientemente grandes, un parámetro umbral determina las particiones a realizar sobre las matrices en cuestión, y éstas son procesadas secuencialmente en la GPU.

- El *speedup* teórico que puede alcanzarse considerando que sólo el paso más demandante del algoritmo de evaluación de energía SOS-MP2 es el paralelizable, fue establecido de acuerdo a la ley de Amdahl. Se demuestra que se consigue un speedup que constituye aproximadamente el 90% de este límite para el caso de cálculos con la base cc-pVDZ en el alcano con cadena de 80 carbonos, y uno ligeramente menor para el mismo sistema con base cc-pVTZ.

Anexo I

En este anexo, se muestran algunos perfilamientos de las llamadas a funciones CUDA en el paso del algoritmo más demandante computacionalmente. Éstos muestran la ejecución y la duración de las funciones y fueron obtenidos empleando la herramienta gráfica nvprof de NVIDIA.

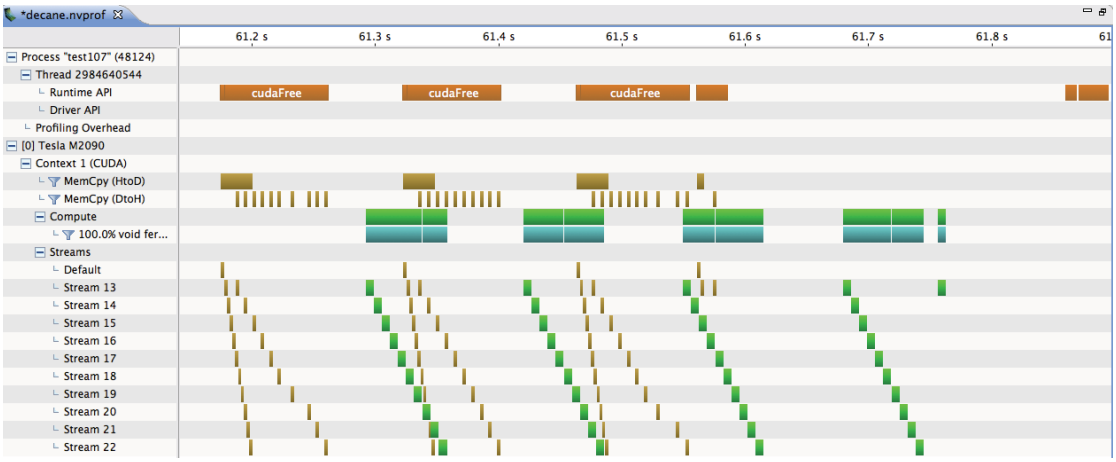

Figura 7.1: Segmento de línea del tiempo. Perfilamiento del cálculo SOS-MP2 con la herramienta NVIDIA *Visual Profiler* para decano, base triple-ζ cc-pVTZ

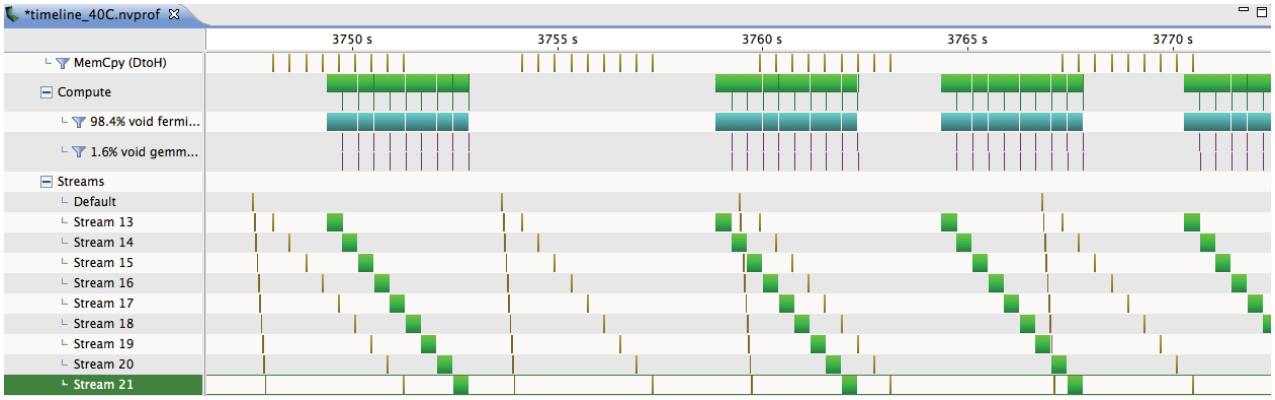

Figura 7.2: Segmento de línea del tiempo. Perfilamiento del cálculo SOS-MP2 con la herramienta NVIDIA *Visual Profiler* para el alcano lineal de 40 carbonos, base triple-ζ cc-pVTZ

Figura 7.3: Segmento de línea del tiempo. Perfilamiento del cálculo SOS-MP2 con la herramienta NVIDIA *Visual Profiler* para el alcano lineal de 80 carbonos, base triple-ζ cc-pVTZ

Manejo de matrices grandes

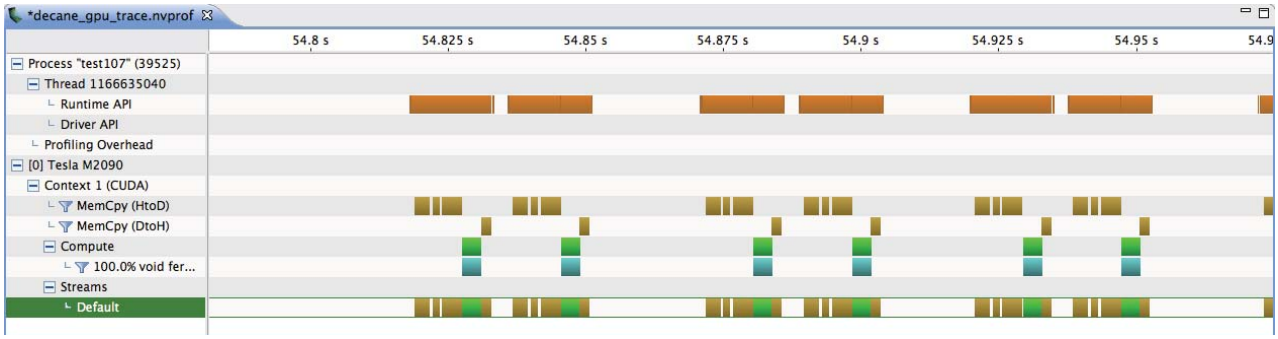

Figura 7.4: Línea del tiempo parcial. Perfilamiento del cálculo SOS-MP2 con la herramienta NVIDIA *Visual Profiler* para decano, base triple-ζ cc-pVTZ. A diferencia de la figura 7.1, en este caso el parámetro que determina el tamaño umbral para particionar las matrices fue modificado de tal forma que la multiplicación de matrices $AB = C$ es llevada a cabo dividiendo la matriz B en dos, y la operación es completada efectuando dos multiplicaciones de matrices.

Anexo II

```
/*Este programa es empleado para probar el rendimiento de GPU en la multiplicacion
 * de matrices empleando DGEMM
 * Uso:
 * ./a.out rows dummy columns ciclos
 * Donde se especifican de las dimensiones de la matriz A (rows*dummy), B(dummy*columns)
 * y ciclos es el numero de ciclos en los que el programa genera las matrices
 * aleatorias A y B y efectua la multiplicacion
 *
 * En esta version, se emplean streams para acelerar la multiplicacion de matrices
 */

/* Includir system */
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <sys/time.h>
/* Incluir cuda */
#include <cuda_runtime.h>
#include <cublas_v2.h>
#include "mkl.h"
#include "run_cublas_luis.C"
#define MIN(X,Y) ((X) < (Y) ?  (X) : (Y))
```

```
inline cudaError_t checkCuda(cudaError_t result)
{
  if (result != cudaSuccess) {
    fprintf(stderr, "CUDA Runtime Error: %sn",
            cudaGetErrorString(result));
exit(2);
}
return result;
}

double getTimeElapsed(struct timeval end, struct timeval start)
    {
        return (end.tv_sec - start.tv_sec) + (end.tv_usec - start.tv_usec) / 1000000.00;
    }

/* Main */
int main(int argc, char **argv)
{
    cublasStatus_t status;
    timeval inicio,fin;
    cudaError_t error;
    int I3, I1, dummy,j,MAX,Num1,k;
    long int NBatchsize;
    double *cpu_A, *cpu_B;
    double *h_A;
    double *h_B;
    double *h_C;
    double *h_C_ref;
    double *d_A = 0;
    double *d_B = 0;
    double *d_C = 0;
    double alpha = 1.0f;
    double beta = 1.0f;
    long int sizeA, sizeC, sizeB;
```

```
    int ciclos,verif ;
    int i, deviceCount;
    clock_t start, end;
    double error_norm, seconds;
    double ref_norm;
    double diff;
    int rows, columns, device=0;
/*Variables cuBLAS */
    cublasHandle_t handle;
    cudaStream_t *streamId;
    cudaDeviceProp deviceProp;

cudaError_t cudaResultCode = cudaGetDeviceCount(&deviceCount);
    if (cudaResultCode != cudaSuccess){
        printf("!!! Error, no hay dispositivo GPU");
        exit(1);
        }

if(argc != 5){
printf("Error, uso: ./program rows dummy columns ciclos\n");
exit(1);

}

/*Probando las capacidades del dispositivo */

cudaGetDeviceProperties(&deviceProp,device);

if(deviceProp.asyncEngineCount > 0)
        printf("El dispositivo soporta transferencia de datos y ejecucion de kernels
        de forma concurrente, %d\n",deviceProp.asyncEngineCount);

if(deviceProp.concurrentKernels == 1)
        printf("El dispositivo soporta ejecucion concurrente de kernels \n");
```

```
    rows = atoi(argv[1]);
    dummy = atoi(argv[2]);
    columns = atoi(argv[3]);
    ciclos = atoi(argv[4]);
    sizeA = rows*dummy;
    sizeC = rows*columns;
    sizeB = dummy*columns;
    /* Inicializa CUBLAS */
    printf("Test CUBLAS corriendo...\n");
    status = cublasCreate(&handle);

    if (status != CUBLAS_STATUS_SUCCESS)
    {
        fprintf(stderr, "Error en inicializacion de CUBLAS !!!\n");
        return EXIT_FAILURE;
    }

   checkCuda(cudaMallocHost(&h_C,sizeC * sizeof(h_C[0])));

    h_C_ref = (double *)malloc(sizeC*sizeof(double));
    if (h_C_ref == 0)
    {
        fprintf(stderr, "Error en apartado de memoria en el Host (C) !!!\n");
        return EXIT_FAILURE;
    }

/*Inicializacion de datos en h_C y h_C_ref*/
    for(i=0;i<sizeC; i++){
        h_C[i] = 0;
        h_C_ref[i] = 0;
   }

/*Comienza medicion de tiempo */
```

```
gettimeofday(&inicio, NULL);

/*Este programa asume que las tandas de matrices seran de 5 elementos */
NBatchsize=5;

/*Procesamiento de datos en la CPU */
#ifdef CPU_ONLY
for(i=0;i<ciclos;i+=NBatchsize){

Num1 = MIN(i+NBatchsize,ciclos) - i;

    cpu_A = (double *)malloc(sizeof(double)*sizeA*Num1);

    if (cpu_A == 0)
    {
        fprintf(stderr, "!!!! Error en el apartado de memoria en host (cpu_A)\n");
        return EXIT_FAILURE;
    }

   cpu_B = (double *)malloc(sizeof(double)*sizeB*Num1);

    if (cpu_B == 0)
    {
        fprintf(stderr, "!!!! Error en el apartado de memoria en host (cpu_B)\n");
        return EXIT_FAILURE;
    }

srand(i);
 for (k = 0; k < sizeA*Num1; k++)
    {
        cpu_A[k] = rand() / (double)RAND_MAX;
                    }

                      for(k=0; k < sizeB*Num1; k++){
```

```
                                cpu_B[k] = rand()/(double)RAND_MAX;

                                }
for(k=0;k<Num1;k++){
    cblas_dgemm(CblasColMajor,CblasNoTrans,CblasTrans,rows,columns,dummy,alpha,
    cpu_A+sizeA*k,rows,cpu_B+sizeB*k,columns,beta,h_C_ref,rows);

}

free(cpu_A);
free(cpu_B);
}
#else

/*Apartando memoria para los cudaStreams */

streamId = (cudaStream_t *)malloc(NBatchsize*sizeof(cudaStream_t));
    if (streamId == 0)
    {
        fprintf(stderr, "!!!!Error en el apartado de memoria en el host (streamId)\n");
        return EXIT_FAILURE;
    }

/*Creacion de los streams*/
for(i=0;i< NBatchsize ;i++){

        if(cudaStreamCreate(&streamId[i]) != cudaSuccess){
                printf("Error en la creacion de streams \n");
        exit(2);
        }

}

#ifdef COMPROBACION
```

```
int error_flag =0;
#endif
/*Tratando de emular las condiciones en el programa, se apartan los arreglos
 *del tamano apropiado
 *para "cargar" una tanda de datos.
 */

for(i=0;i<ciclos;i+=NBatchsize){

Num1 = MIN(i+NBatchsize,ciclos) - i;

if((Num1 != NBatchsize) || i==0){
    checkCuda(cudaMallocHost(&h_A,sizeA *Num1* sizeof(h_A[0])));

    checkCuda(cudaMallocHost(&h_B,sizeB * Num1*sizeof(h_B[0])));

   #ifdef COMPROBACION
   cpu_A = (double *)malloc(sizeof(double)*Num1*sizeA);
   cpu_B = (double *)malloc(sizeof(double)*Num1*sizeB);
   #endif

}
srand(time(NULL));
 for (k = 0; k < sizeA*Num1; k++)
    {
        h_A[k] = rand() / (double)RAND_MAX;
        #ifdef COMPROBACION
        cpu_A[k] = h_A[k];
        #endif

                            }

                        for(k=0; k < sizeB*Num1; k++){
                                h_B[k] = rand()/(double)RAND_MAX;
```

```
                                #ifdef COMPROBACION
                                cpu_B[k] = h_B[k];
                                #endif
                               }
if(i==0){
I3=1;
I1=0;

}
else if(Num1 == NBatchsize){
I3=0;
I1=1;
}
else{
I3=2;
I1=2;
}

verif=run_cublas_batched(handle,CUBLAS_OP_N,CUBLAS_OP_T,&rows,&columns,&dummy,
&alpha,h_A,&rows,h_B,&columns,&beta,h_C,&rows,&I3,&I1,Num1,streamId);
        #ifdef COMPROBACION
        printf("Entrando a cblas\n");
        for(k=0;k<Num1;k++){
        cblas_dgemm(CblasColMajor,CblasNoTrans,CblasTrans,rows,columns,dummy,alpha,
        cpu_A+sizeA*k,rows,cpu_B+sizeB*k,columns,beta,h_C_ref,rows);
        }

        diff=0.0;
        for(k=0;k<sizeC;k++){
        diff += abs(h_C[k] - h_C_ref[k]);
        }
        if(diff > 0.00001){
        error_flag += 1;;
```

```
        }
        #endif

        if(verif != 0){

        printf("Error en la funcion run_cublas_batched\n");
        exit(2);
        }

if((Num1 + NBatchsize)>ciclos ){
cudaFreeHost(h_A);
cudaFreeHost(h_B);

#ifdef COMPROBACION
free(cpu_A);
free(cpu_B);
#endif
}

}
#ifdef COMPROBACION
if(error_flag != 0){
printf("Resultado de Comprobacion de Datos: Error, los resultados CPU difieren de los GPU\n");
}else{
printf("Resultado de Comprobacion de Datos: Ok\n");

}
#endif

cudaFreeHost(h_C);
free(streamId);

#endif
```

```
gettimeofday(&fin, NULL);
double time_elapsed = getTimeElapsed(fin, inicio);
printf("Tiempo de pared : %f \n", time_elapsed);

return 0;
}
```

Notas: Este programa fue compilado empleando la biblioteca MKL de Intel. Las directivas de compilación permiten que este programa pueda ser compilado para que el cómputo se realice exclusivamente en la CPU. Cuando la variable de compilación COMPROBACION es definida (lo que se logra usando la bandera -DCOMPROBACION al momento de compilar), el programa compara los resultados obtenidos al efectuar al cómputo en la GPU con los obtenidos en la CPU, y determina si ambos son equivalentes dentro de un margen de error determinado.

Función CUBLAS para multiplicación de matrices

```
#include <stdio.h>
#include <math.h>
#include <unistd.h>
#include <stdlib.h>
#include <cublas.h>
#include <cuda_runtime.h>
/*Esta funcion ejecuta multiplicaciones de matrices almacenadas
* en arreglos (A y B), realizando una multiplicacion de forma separada
* en diferente stream */

int run_cublas_batched(cublasHandle_t handle, cublasOperation_t transA,
        cublasOperation_t transB, int *m, int *n, int *k,
        double *alpha, double *A, int *lda, double *B, int *ldb,
         double *beta, double *C, int *ldc, int *I3, int *I1,  int BatchCount,
         cudaStream_t *streams){

int sizeA= (*m) * (*k), sizeB= (*n) * (*k), sizeC= (*m) * (*n);
```

```
double *d_globalA, *d_globalB,*d_C;
int i;
cublasStatus_t status;

if(*I3 ==1){
 if (cudaMalloc((void **)&d_globalA, sizeA * sizeof(A[0])*BatchCount) != cudaSuccess)
    {
        fprintf(stderr, "!!!! Error en el apartado de memoria GPU (allocate A)\n");
        return EXIT_FAILURE;
    }

 if (cudaMalloc((void **)&d_globalB, sizeB * sizeof(B[0])*BatchCount) != cudaSuccess)
    {
        fprintf(stderr, "!!!! Error en el apartado de memoria GPU (allocate B)\n");
        return EXIT_FAILURE;
    }
}
    if (cudaMalloc((void **)&d_C, sizeC * sizeof(C[0])) != cudaSuccess)
    {
        fprintf(stderr, "!!!! Error en el apartado de memoria GPU (allocate C)\n");
        return EXIT_FAILURE;
    }

 status = cublasSetVector(sizeC, sizeof(C[0]), C, 1, d_C, 1);

    if (status != CUBLAS_STATUS_SUCCESS)
    {
        fprintf(stderr, "!!!! Error en el acceso al dispositivo (write C)\n");
        return EXIT_FAILURE;
    }

for(i=0;i<BatchCount;i++){
 status = cublasSetVectorAsync(sizeA, sizeof(A[0]), A+sizeA*i, 1, d_globalA+sizeA*i,
                1,streams[i]);
```

```
    if (status != CUBLAS_STATUS_SUCCESS)
    {
        fprintf(stderr, "!!!!  Error en el acceso al dispositivo (write A)\n");
        return EXIT_FAILURE;
    }

 status = cublasSetVectorAsync(sizeB, sizeof(B[0]), B+sizeB*i, 1, d_globalB+sizeB*i,
                1,streams[i]);

    if (status != CUBLAS_STATUS_SUCCESS)
    {
        fprintf(stderr, "!!!!  Error en el acceso al dispositivo (write B)\n");
        return EXIT_FAILURE;
    }


status=cublasSetStream(handle,streams[i]);

    if (status != CUBLAS_STATUS_SUCCESS)
    {
        fprintf(stderr, "!!!!  Error en el acceso al dispositivo (stream) \n");
        return EXIT_FAILURE;
    }
status = cublasDgemm(handle, transA, transB, *m, *n, *k, alpha, d_globalA+sizeA*i,
                      *lda, d_globalB+sizeB*i, *ldb, beta, d_C,*ldc);

    if (status != CUBLAS_STATUS_SUCCESS)
    {
        fprintf(stderr, "!!!! Error en ejecucion de cublasDgemm\n");
        return EXIT_FAILURE;
    }

 status = cublasGetVectorAsync(sizeC, sizeof(C[0]), d_C, 1, C, 1,streams[i]);
```

```
    if (status != CUBLAS_STATUS_SUCCESS)
    {
        fprintf(stderr, "!!!!  Error en el acceso al dispositivo (get C)\n");
        return EXIT_FAILURE;
    }
}
if(*I3 ==*I1){
cudaFree(d_globalA);
cudaFree(d_globalB);
}
cudaFree(d_C);


return 0;
}
```

Programa que usa MPI

```
#include <stdlib.h>
#include <math.h>
#include <stdio.h>
#include <time.h>
#include "mkl.h"
#include "mpi.h"

#define SIZE_X 1024
#define SIZE_Y 20

void aumenta_matriz(double *output_final, double *intermedio,int size){
int i;
        for(i=0;i<size;i++){
        output_final[i] = output_final[i] + intermedio[i];
        }
}
```

```
int main(int argc, char **argv){

int rank=0,size,i,j,k,l;
double *output_final,a,*intermedio, start_time, finish_time;
double *local_A,*local_B, alpha=1.0f, beta=1.0f;
int n,rows,dummy,columns,ciclos,sizeA,sizeB,sizeC;

MPI_Init(&argc, &argv);
MPI_Comm_rank(MPI_COMM_WORLD,&rank);
MPI_Comm_size(MPI_COMM_WORLD,&size);
MPI_Status status;

if(argc != 5){
printf("Error, uso: ./program rows dummy columns ciclos\n");
exit(1);

}

    rows = atoi(argv[1]);
    dummy = atoi(argv[2]);
    columns = atoi(argv[3]);
    ciclos = atoi(argv[4]);
    sizeA = rows*dummy;
    sizeC = rows*columns;
    sizeB = dummy*columns;

//Tiempo de inicio
start_time = MPI_Wtime();

//Establecemos el numero de matrices que procesara cada proceso
n = ciclos/size;

if(rank < ciclos%size){
```

```
        n = n + 1;
}

        local_A = (double *)malloc(sizeA*sizeof(double));

        local_B = (double *)malloc(sizeB*sizeof(double));

        intermedio = (double *)malloc(sizeC*sizeof(double));

        for(i=0;i<sizeC;i++){
        intermedio[i] = 0.0;
        }

for(i=0;i<n;i++){

        srand(time(NULL));

         for (k = 0; k < sizeA; k++)
            {
        local_A[k] = rand() / (double)RAND_MAX;
                    }

                       for(k=0; k < sizeB; k++){
                               local_B[k] = rand()/(double)RAND_MAX;

                               }

        cblas_dgemm(CblasColMajor,CblasNoTrans,CblasTrans,rows,columns,dummy,alpha,
                        local_A,rows,local_B,columns,beta,intermedio,rows);

}
//Los procesos diferentes del rango 0, envian sus datos al rango 0
if(rank != 0){
```

```
        MPI_Send(intermedio,sizeC,MPI_DOUBLE,0,10,MPI_COMM_WORLD);

}else{
        output_final = (double *)malloc(sizeC*sizeof(double));

//Inicializacion de la matriz output final/
        for(i=0;i<sizeC;i++){

        output_final[i]=0.0 ;
        }

        aumenta_matriz(output_final,intermedio,sizeC);

        for(i=1;i<size;i++){
        MPI_Recv(intermedio,sizeC,MPI_DOUBLE,MPI_ANY_SOURCE,10,MPI_COMM_WORLD,&status);
        aumenta_matriz(output_final,intermedio,sizeC);
        }

/*
printf ("\n Top left corner of matrix : \n");
    for (i=0; i<6; i++) {
      for (j=0; j<6; j++) {
        printf ("%12.0f", output_final[j+i*rows]);
      }
      printf ("\n");
    }
*/
        free(output_final);
}

finish_time=MPI_Wtime();
if(rank ==0){
printf("Tiempo de multiplicacion: %f segundos\n",finish_time - start_time);
}
```

```
        free(local_A);
        free(local_B);
        free(intermedio);

MPI_Finalize();
return 0;
}
```

Apéndice A

La aproximación de Born-Oppenheimer

Los estados electrónicos adiabáticos definidos en la ecuación 2-9, definen una base completa, dada la hermiticidad de H_{el}. De esta forma, las soluciones a la ecuación 2-1 pueden ser expresadas en esta base de la siguiente forma:

$$\psi(\mathbf{r},\mathbf{R}) = \sum_a \chi_a(\mathbf{R})\phi_a(\mathbf{r};\mathbf{R}) \tag{A-1}$$

Para encontrar los coeficientes $\chi_a(\mathbf{R})$ de esta expansión, procedemos de la siguiente manera: Después de la inserción de la ecuación A-1 en 2-1, obtenemos:

$$\begin{aligned}
\widehat{H}_{mol}\psi(\mathbf{r},\mathbf{R}) &= \left(\widehat{H}_{el}(\mathbf{R}) + \widehat{T}_{nuc} + \widehat{V}_{nuc-nuc}\right)\sum_a \chi_a(\mathbf{R})\phi_a(\mathbf{r};\mathbf{R}) \\
&= \sum_a \left[E_a(\mathbf{R}) + V_{nuc-nuc}\right]\chi_a(\mathbf{R})\phi_a(\mathbf{r};\mathbf{R}) + \sum_a \widehat{T}_{nuc}\chi_a(\mathbf{R})\phi_a(\mathbf{r};\mathbf{R}) \quad \text{(A-2)} \\
&= E\sum_a \chi_a(\mathbf{R})\phi_a(\mathbf{r};\mathbf{R})
\end{aligned}$$

Multiplicando la ecuación anterior por $\phi_b^*(\mathbf{r};\mathbf{R})$ e integrando sobre todas las coordenadas electrónicas, obtenemos la siguiente expresión (aprovechando la ortogonalidad de la base adiabáti-

ca):

$$\begin{aligned}\int d\mathbf{r}\phi_b^*(\mathbf{r};\mathbf{R})\widehat{H}_{mol}\psi(\mathbf{r};\mathbf{R}) &= [E_b(\mathbf{R})+V_{nuc-nuc}]\,\chi_b(\mathbf{R})+\sum_a\int d\mathbf{r}\phi_b^*(\mathbf{r};\mathbf{R})\widehat{T}_{nuc}\phi_a(\mathbf{r};\mathbf{R})\chi_a(\mathbf{R}) \\ &= E\chi_b(\mathbf{R}) \qquad \text{(A-3)}\end{aligned}$$

El efecto de $\widehat{T}_{nuc}$ sobre $\phi_a(\mathbf{r};\mathbf{R})\chi_a(\mathbf{R})$ es:

$$\begin{aligned}\widehat{T}_{nuc}\phi_a(\mathbf{r};\mathbf{R})\chi_a(\mathbf{R}) &= \sum_{n=1}^{N_{nuc}}\frac{\widehat{P}_n}{2M_n}\left[\widehat{P}_n\phi_a(\mathbf{r};\mathbf{R})\chi_a(\mathbf{R})\right] \qquad \text{(A-4)} \\ &= \sum_{n=1}^{N_{nuc}}\frac{1}{2M_n}\left[\left(\widehat{P}_n^2\phi_a(\mathbf{r};\mathbf{R})\right)\chi_a(\mathbf{R})+2\widehat{P}_n\phi_a(\mathbf{r};\mathbf{R})\widehat{P}_n\chi_a(\mathbf{R})+\phi_a(\mathbf{r};\mathbf{R})\widehat{P}_n^2\chi_a(\mathbf{R})\right]\end{aligned}$$

En donde hemos considerado $\widehat{P}_n=-i\hbar\nabla_n$ y la regla de diferenciación del producto de funciones. Insertando A-4 en A-3, y definiendo el operador de acoplamiento no adiabático:

$$\widehat{\Theta}_{ba}=\int dr\phi_b^*(\mathbf{r};\mathbf{R})\left[\widehat{T}_{nuc}\phi_a(\mathbf{r};\mathbf{R})\right]+\sum_n\frac{1}{M_n}\int d\mathbf{r}\phi_b^*(\mathbf{r};\mathbf{R})\widehat{P}_n\phi_a(\mathbf{r};\mathbf{R})\widehat{P}_n, \qquad \text{(A-5)}$$

llegamos a la expresión:

$$[E_b(\mathbf{R})+V_{nuc-nuc}]\,\chi_b(\mathbf{R})+\sum_a\left[\widehat{\Theta}_{ba}+\widehat{T}_{nuc}\right]\chi_a(\mathbf{R})=E\chi_b, \qquad \text{(A-6)}$$

que podemos reescribir de la siguiente forma:

$$\left[E_b(\mathbf{R})+\widehat{V}_{nuc-nuc}-E+\widehat{\Theta}_{bb}+\hat{T}_{nuc}\right]\chi_b(\mathbf{R})=-\sum_{a\neq b}\widehat{\Theta}_{ba}\chi_a(\mathbf{R}) \qquad \text{(A-7)}$$

La solución a este conjunto de ecuaciones puede ser simplificada considerablemente cuando el acoplamiento no adiabático entre un estado electrónico determinado b y el resto del espectro de soluciones, es 0. Cuando esto ocurre, tenemos la siguiente expresión:

$$\widehat{H}_a(\mathbf{R})\chi_a(\mathbf{R})=\left(\widehat{T}_{nuc}+U_a(\mathbf{R})\right)\chi_a(\mathbf{R})=E\chi_a(\mathbf{R}), \qquad \text{(A-8)}$$

donde hemos definido $U_a(\mathbf{R}) = E_a(\mathbf{R}) + \widehat{V}_{nuc-nuc}(\mathbf{R}) + \widehat{\Theta}_{aa}$ y $\widehat{H}_a(\mathbf{R})$ es el Hamiltoniano nuclear del estado $|\phi_a\rangle$. Al resolver la ecuación A-8, obtenemos un conjunto de funciones de onda χ_{aM} donde M es un número cuántico que identificamos con los estados vibracionales asociados a un estado electrónico adiabático determinado. De esta forma, la función de onda total adiabática toma la forma:

$$\psi_{aM}^{(adia)}(\mathbf{r};\mathbf{M}) = \chi_{aM}(\mathbf{R})\phi_a(\mathbf{r};\mathbf{R}) \tag{A-9}$$

Cuando los acoplamientos no adiabáticos son despreciados, y de esta forma llegamos a la expresión A-9, se dice que hemos hecho uso de la Aproximación de Born-Oppenheimer.

La justificación del despreciamiento de los acoplamientos no adiabáticos puede ser basada en la expansión perturbativa a segundo orden de la energía con respecto al operador no adiabático, de acuerdo a:

$$\xi_{aM}^{(2)} = \xi_{aM}^{(adia)} + \sum_{b,N} \frac{|\langle \chi_{aM}|\Theta_{ab}|\chi_{bN}\rangle|^2}{\xi_{aM}^{(adia)} - \xi_{bN}^{(adia)}} \tag{A-10}$$

De esta expresión, es notorio que cuando Θ_{ab} representa una pequeña perturbación al carácter de las funciones de onda, esto es, cuando las funciones de onda no se modifican apreciablemente con cambios en R, y adicionalmente la diferencia de energías entre los estados adiabáticos a y b es grande en comparación con el elemento de matriz $\langle \chi_{aM}|\Theta_{ab}|\chi_{bN}\rangle$, el efecto del acomplamiento no adiabático puede ser aproximado a 0.

Apéndice B

Ecuaciones Lineales Inhomogéneas

Suponiendo por el momento que el lado derecho de la ecuación 2-41 es conocido, obtenemos una expresión general del tipo:

$$\mathbf{A}\boldsymbol{\mu} = \boldsymbol{\nu}, \tag{B-1}$$

donde A es un operador hermitiano dado, $\boldsymbol{\nu}$ es un vector, y $\boldsymbol{\mu}$ es la solución buscada. Esta ecuación puede ser tratada en un espacio vectorial abstracto, de tal forma que $\mathbf{A}$ puede ser representada de igual forma por una matriz cuadrada que por un operador diferencial.
Es importante tomar en consideración dos casos:

- Ya sea que la ecuación homogénea

  $$\mathbf{A}\boldsymbol{\mu}' = 0 \tag{B-2}$$

  tiene soluciones no triviales, o dicho de otra forma $det\ \mathbf{A} = 0$, si $\mathbf{A}$ es una matriz cuadrada.

- O la ecuación B-2 no tiene soluciones no triviales. En este caso, el operador A tiene un operador inverso único $\mathbf{A}^{-1}$, de tal manera que, para determinado $\boldsymbol{\nu}$, la solución a B-1 está dada por

  $$\boldsymbol{\mu} = \mathbf{A}^{-1}\boldsymbol{\nu} \tag{B-3}$$

En el primer caso, no existe un operador lineal inverso de $\mathbf{A}$, dado que B-2 no es invertible. Por otro lado, si B-1, tiene una solución particular $\mathbf{f}$,

$$\mathbf{A}\mathbf{f} = \boldsymbol{\nu} \tag{B-4}$$

Entonces cualquier vector

$$\boldsymbol{\mu} = \boldsymbol{\mu}' + \mathbf{f} \tag{B-5}$$

donde $\boldsymbol{\mu}'$ es solución de B-2, también es una solución. De esta forma, el conjunto de soluciones expresadas mediante la ecuación B-5 constituyen la *solución general* de B-1.
En este caso, sin embargo, la existencia de la solución depende de una condición necesaria y suficiente: $\boldsymbol{\nu}$ no debe tener una componente en el subespacio producido por los vectores $\boldsymbol{\mu}'_i$, como podemos ver en las siguientes ecuaciones:

$$\langle \boldsymbol{\mu}' | \boldsymbol{\nu} \rangle = \langle \boldsymbol{\mu}' | \mathbf{A}\boldsymbol{\mu} \rangle = \langle \mathbf{A}\boldsymbol{\mu}' | \boldsymbol{\mu} \rangle = 0 \tag{B-6}$$

Si denotamos por $\hat{P}$ el operador de proyección que proyecta cualquier vector en el subespacio producido por los vectores μ', entonces la condición B-6 podemos expresarla de la siguiente manera

$$\hat{P}|\boldsymbol{\nu}\rangle = 0 \tag{B-7}$$

y podemos construir la solución $\mathbf{f}$ sistemáticamente. Usando la condición B-7, podemos reescribir la ecuación B-4 de la siguiente forma:

$$\mathbf{A}\mathbf{f} = \left(\hat{1} - \hat{P}\right)\boldsymbol{\nu} \tag{B-8}$$

Aunque A no es invertible, existen operadores hermitianos $\mathbf{K}$, tal que

$$\mathbf{A}\mathbf{K} = \hat{1} - \hat{P}. \tag{B-9}$$

Dado que existe un número infinito de operadores K, tal y como se muestra abajo:

$$\mathbf{A}(\mathbf{K} + \hat{P}\mathbf{B}) = \mathbf{A}\mathbf{K} + \sum_{\mu'} A|\boldsymbol{\mu}'\rangle\langle\boldsymbol{\mu}'|\mathbf{B} = \mathbf{A}\mathbf{K} = \hat{1} - \hat{P}; \qquad \mathbf{B} \quad arbitrario \tag{B-10}$$

en donde se ha empleado la ecuación B-2, es posible establecer la restricción

$$\mathbf{K}|\mu'\rangle = 0 \tag{B-11}$$

En el contexto de ecuaciones diferenciales lineales, el operador $\mathbf{K}$ puede ser representado por una función de Green $G(\mathbf{r}, \mathbf{r}') = \langle \mathbf{r}|\mathbf{K}|\mathbf{r}'\rangle$. La cadena de igualdades

$$\mathbf{A}(\mathbf{K}\boldsymbol{\nu}) = (\mathbf{A}\mathbf{K})\boldsymbol{\nu} = (\hat{1} - \hat{P})\boldsymbol{\nu} = \boldsymbol{\nu} \tag{B-12}$$

demuestra que

$$\mathbf{f} = \mathbf{K}\boldsymbol{\nu} \tag{B-13}$$

es una solución particular de B-1. De manera que concluímos que si $\mathbf{A}\boldsymbol{\mu}' = 0$ tiene soluciones no triviales, y $\hat{P}\boldsymbol{\nu} = 0$, entonces $\mathbf{A}\boldsymbol{\mu} = \boldsymbol{\nu}$ tiene como solución general:

$$\boldsymbol{\mu} = \boldsymbol{\mu}' + \mathbf{K}\boldsymbol{\nu} \tag{B-14}$$

Apéndice C

Integración Numérica

En este apéndice se introducen de forma breve los métodos más comunes para efectuar integraciones numéricas, para un tratamiento más detallado, el lector interesado puede consultar a Davis[66], o Hamming[67], entre otros.

El objetivo de los métodos mencionados, es aproximar la integral definida

$$\int_a^b f(x)dx, \tag{C-1}$$

por la suma

$$I = \sum_{i=0}^{n} A_i f(x_i), \tag{C-2}$$

donde las *abscisas nodales* x_i y los factores de peso A_i, dependen de la regla seleccionada para la cuadratura numérica.

Los métodos numéricos de integración pueden ser divididos en dos grupos: de Newton-Cotes, y los de cuadratura Gaussiana. Los primeros se caracterizan por abscisas uniformemente espaciadas y resultan útiles si la función $f(x)$ ha sido evaluada a intervalos iguales o puede ser evaluada a bajo costo computacional. En el caso de los métodos de cuadratura Gaussiana, la localización de las abscisas es determinada para obtener la mayor exactitud posible. Dado que este tipo de métodos requieren de menos evaluaciones para determinado nivel de precisión, éstos son más populares cuando el integrando es difícil de evaluar.

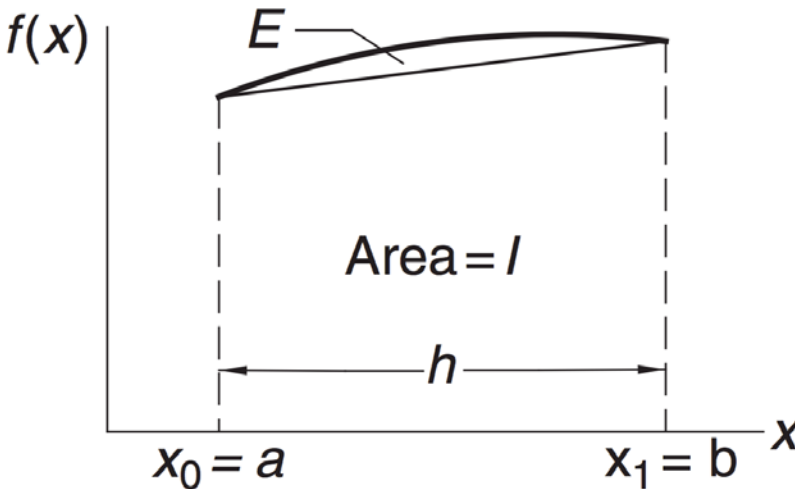

Figura C.1: Regla del trapezoide

C.1 Newton-Cotes

C.1.1 Regla del Trapezoide

La regla del trapecio consiste en aproximar una integral definida para una curva $f(x)$, como la mostrada en la Figura C.1, al área del trapezoide definido por $f(x_0)$, $f(x_1)$ y h $(x_1 - x_0)$, la cual está dada por

$$I = [f(a) + f(b)]\frac{h}{2} \tag{C-3}$$

El error en la regla del trapecio

$$E = \int_a^b f(x)dx - I \tag{C-4}$$

es el área entre la curva $f(x)$ y la línea interpoladora, tal y como se indica en la Figura C.1. Este error puede ser obtenido al integrar el error de interpolación, dado por

$$f(x) - P_n(x) = \frac{(x - x_0)(x - x_1)\ldots(x - x_n)}{(n+1)!} f^{(n+1)}(\xi), \tag{C-5}$$

en donde $P_n(x)$ es el polinomio interpolador obtenido por la fórmula de Lagrange[1]

$$P_n(x) = \sum_{i=0}^{n} y_i l_i(x) \tag{C-6}$$

$$l_i(x) = \frac{x-x_0}{x_i-x_0}\cdot\frac{x-x_1}{x_i-x_1}\cdots\frac{x-x_{i-1}}{x_i-x_{i-1}}\cdot\frac{x-x_{i+1}}{x_i-x_{i+1}}\cdots\frac{x-x_n}{x_i-x_n}$$

$$= \prod_{j=0;j\neq i}^{n}\frac{x-x_j}{x_i-x_j}. \tag{C-7}$$

En la ecuación C-5, $\{x_i\}$ son los puntos en las abscisas consideradas en la integración y $f^{(n+1)}(\xi)$ es la $(n+1)$-ésima derivada de $f(x)$, evaluada en ξ, el cual es un valor que se halla en el intervalo (x_0, x_n).

En la regla del trapecio, el polinomio interpolador es construído a partir de 2 ordenadas y 2 abscisas (n=1), por lo tanto, de acuerdo a la ecuación C-5 e integrando sobre el intervalo de la interpolación, se tiene

$$E = \frac{1}{2!}\int_a^b (x-x_0)(x-x_1)f''(\xi)dx = \frac{1}{2}f''(\xi)\int_a^b (x-a)(x-b)dx = -\frac{1}{12}(b-a)^3 f''(\xi) = -\frac{h^3}{12}f''(\xi) \tag{C-8}$$

O dicho de otra forma, el error disminuye significativamente al disminuir el espaciamiento entre las abscisas, y éste será menor en funciones que no varíen bruscamente en el intervalo considerado.

En la práctica, la regla del trapecio es aplicada por piezas, es decir, la integración sobre un intervalo (a, b) puede ser dividida en n páneles (ver Figura C.2), cada uno con longitud h. De esta manera, el área aproximada del i-ésimo panel, está dada por

$$I_i = [f(x_i) + f(x_{i+1})]\frac{h}{2} \tag{C-9}$$

Por lo tanto, el área total, que aproxima la integral, es

$$I = \sum_{i=0}^{n-1} I_i = [f(x_0) + 2f(x_1) + 2f(x_2) + \cdots + 2f(x_{n-1}) + f(x_n)]\frac{h}{2} \tag{C-10}$$

[1]La forma más simple de un interpolador es un polinomio. Se puede demostrar que siempre es posible construir un único polinomio interpolador de grado n, que pase por $n+1$ puntos.

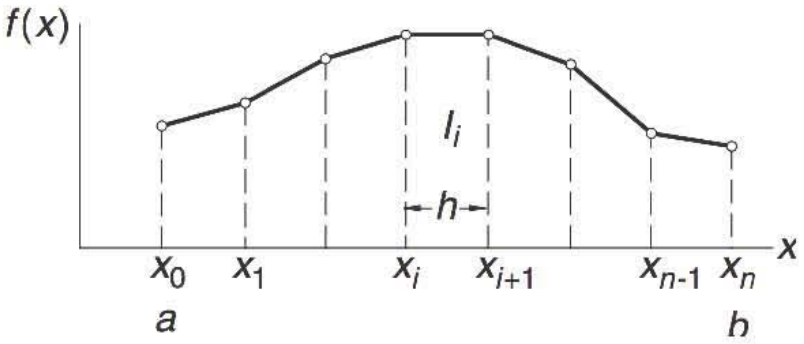

Figura C.2: Regla del trapecio, aplicado por segmentos a un intervalo de integración.

C.2 Integración Gaussiana

Las fórmulas para el cómputo numérico de integrales en cuadratura gaussiana (o de forma abreviada, fórmulas gaussianas), son empleadas en la evaluación de integrales bien comportadas, al igual que los métodos Newton-Cotes, pero además, son buenas en la aproximación de integrales de la forma

$$\int_a^b w(x)f(x)dx, \tag{C-11}$$

donde $w(x)$ es una función de peso integrable. La integración Gaussiana, tiene la misma forma que los métodos Newton-Cotes

$$I = \sum_{i=0}^{n} A_i f(x_i) \tag{C-12}$$

Las diferencias yacen en la manera en la que los factores de peso A_i y las abscisas x_i son determinadas. En una cuadratura Gaussiana, las abscisas y los factores son elegidos de tal forma que la ecuación C-12 produce la integral exacta, si $f(x)$ es un polinomio de grado $2n+1$ o menor:

$$\int_a^b w(x)\lambda_m(x)dx = \sum_{i=0}^{n} A_i\lambda_m(x_i), \qquad m \leq 2n+1, \qquad \lambda_m(x) \text{ polinomio de grado } m \tag{C-13}$$

Los métodos para la determinación de los coeficientes A_i y las abscisas x_i, se basan en conjuntos de polinomios ortogonales, los cuales son definidos en base a la propiedad

$$\int_a^b w(x)\phi_m(x)\phi_n(x)dx = C\delta_{mn}, \tag{C-14}$$

Nombre	Símbolo	a	b	w(x)	C
Legendre	$p_n(x)$	-1	1	1	$2/(2n+1)$
Chebyshev	$T_n(x)$	-1	1	$(1-x^2)^{-1/2}$	$\pi/2$ $(n>0)$
Laguerre	$L_n(x)$	0	∞	e^{-x}	1
Hermite	$H_n(x)$	$-\infty$	∞	e^{-x^2}	$\sqrt{\pi}2^n n!$

Cuadro 2.1: Parámetros que definen los conjuntos de polinomios ortogonales empleados en cuadraturas gaussianas.

donde los límites de integración a,b, la función de peso $w(x)$ y la constante C definen un conjunto. La importancia de estos polinomios, radica en los siguientes teoremas

Teorema 1 *Las abscisas $x_0, x_1, \ldots, x_n$ para aproximar una integral de acuerdo a la ecuación C-12 y que satisfacen la expresión C-13, son las raíces del polinomio $\phi_{n+1}(x)$, que pertenece a un conjunto ortogonal.*

Teorema 2

$$A_i = \int_a^b w(x) l_i(x) dx, \qquad i = 0, 1, \ldots, n \tag{C-15}$$

Donde $l_i(x)$ es una función cardinal de Legendre que considera las abscisas $x_0, x_1, \ldots, x_n$, y es definida de acuerdo a la ecuación C-7.

Una demostración de estos teoremas, para el lector interesado, es propuesta por Kiusalaas[68]. Así, dependiendo de la forma del integrando, y de los límites de integración, es posible seleccionar la cuadratura más apropiada para la integración numérica. Por fortuna, los valores de las abscisas y sus correspondientes factores de peso se encuentran disponibles en la literatura, lo que disminuye el esfuerzo de programación al solucionar problemas.

Apéndice D

Aproximación de la Resolución de la Identidad

Entre los métodos para aproximar integrales bielectrónicas de cuatro centros, definidas por

$$(ij|kl) = \int\int \frac{\phi_1(\mathbf{r})\phi_j(\mathbf{r})\phi_k(\mathbf{r}')\phi_l(\mathbf{r}')}{|\mathbf{r}-\mathbf{r}'|} d\mathbf{r}d\mathbf{r}', \tag{D-1}$$

la aproximación de la resolución de la identidad ha sido ampliamente adoptada en códigos de estructura electrónica, dado que ésta constituye la estrategia que reduce el esfuerzo computacional de forma más exitosa[69].

En la siguiente sección, se profundiza en la derivación de las ecuaciones que conforman dicha aproximación.

D.1 Fundamento teórico

La aproximación de la resolución de la identidad (RI por sus siglas en inglés), también conocida como la técnica de ajuste de la densidad, parte de la representación del producto de dos funciones atómicas en términos de una combinación lineal de funciones auxiliares, de acuerdo a

$$p_{ij}(\mathbf{r}) = \phi_i(\mathbf{r})\phi_j(\mathbf{r}) \approx \tilde{p}_{ij}(\mathbf{r}) = \sum_L c_{ij}^L P_L(\mathbf{r}), \tag{D-2}$$

en donde $\{P_L\}$ denota el conjunto de las funciones auxiliares. Usando esta expresión en la evaluación de las integrales bielectrónicas de cuatro centros (ecuación D-1), obtenemos

$$(ij|kl) = \sum_{LM} c_{ij}^{L} c_{kl}^{M} (L|M), \tag{D-3}$$

en donde

$$(L|M) = V_{LM} = \int\int \frac{P_L(\mathbf{r}) P_M(\mathbf{r}')}{|\mathbf{r} - \mathbf{r}'|} d\mathbf{r} d\mathbf{r}'. \tag{D-4}$$

La razón del éxito de la aproximación RI, reside en que el conjunto base producido por todos los productos posibles $\{\phi_i(\mathbf{r})\phi_j(\mathbf{r})\}$, constituye un conjunto redundante, dado que éste escala cuadráticamente con el número de orbitales $\phi_i(\mathbf{r})$[1], mas sin embargo el sistema puede ser descrito por un conjunto cuyo escalamiento sea lineal respecto al tamaño del sistema. De este razonamiento, se tiene entonces que existe una dependencia lineal entre la base $\{\phi_i(\mathbf{r})\phi_j(\mathbf{r})\}$ y un conjunto $\{P_L(\mathbf{r})\}$ [69].

El siguiente paso en la aproximación RI, consiste en la determinación de los coeficientes c_{ij}^{L}. Desafortunadamente, en la literatura, el procedimiento para determinarlos no es único, y cada uno conduce a diferentes expresiones para esta aproximación.

Una de dichas versiones, parte del error de la expansión en un conjunto base auxiliar del producto de pares de funciones, el cual está dado por

$$\delta p_{ij}(\mathbf{r}) = p_{ij}(\mathbf{r}) - \tilde{p}_{ij}(\mathbf{r}) = \phi_i(\mathbf{r})\phi_j(\mathbf{r}) - \sum_{L} c_{ij}^{L} P_L(\mathbf{r}). \tag{D-5}$$

Para encontrar el conjunto de coeficientes $\{c_{ij}^{L}\}$, podemos minimizar la norma del residual $\delta p_{ij}(\mathbf{r})$, el cual puede expresarse de la siguiente manera:

$$\begin{aligned} \int |\delta p_{ij}(\mathbf{r})|^2 d\mathbf{r} &= \int \left[\phi_i^2(\mathbf{r})\phi_j^2(\mathbf{r}) - 2\sum_{L} c_{ij}^{L} P_L(\mathbf{r})\phi_i(\mathbf{r})\phi_j(\mathbf{r}) + \sum_{LM} c_{ij}^{L} c_{ij}^{M} P_L(\mathbf{r}) P_M(\mathbf{r}) \right] d\mathbf{r} \\ &= \mathbf{F}[\{c_{ij}^{L}\}]. \end{aligned} \tag{D-6}$$

[1] El número de pares posibles para n orbitales atómicos, es $\frac{n(n-1)}{2}$

Podemos proceder ahora a derivar la función $\mathbf{F}[\{c_{ij}^L\}]$ respecto al coeficiente c_{ij}^L e igualar a cero, de acuerdo a

$$\frac{\partial \mathbf{F}[\{c_{ij}^L\}]}{\partial c_{ij}^L} = \int \left[-2P_L(\mathbf{r})\phi_i(\mathbf{r})\phi_j(\mathbf{r}) + \sum_M c_{ij}^M P_L(\mathbf{r})P_M(\mathbf{r}) \right] d\mathbf{r} = 0, \tag{D-7}$$

de esta manera obtenemos

$$\langle ij|L\rangle = \sum_M c_{ij}^M \langle M|L\rangle = \sum_M c_{ij}^M S_{ML}, \tag{D-8}$$

en donde el factor -2 ha sido absorbido en los coeficientes constantes, y

$$\langle ij|L\rangle = \int \phi_i(\mathbf{r})\phi_j(\mathbf{r})P_L(\mathbf{r})d\mathbf{r} \tag{D-9}$$

$$S_{ML} = \langle M|L\rangle = \int P_M(\mathbf{r})P_L(\mathbf{r})d\mathbf{r}. \tag{D-10}$$

La ecuación D-8, corresponde a una ecuación de la forma

$$A_{ij}^L = \sum_M c_{ij}^M B_{ML}, \tag{D-11}$$

que puede solucionarse encontrando la matriz inversa de $\mathbf{B}$, de tal forma que

$$c_{ij}^M = \sum_L \langle ij|L\rangle \left(\langle L|M\rangle\right)^{-1} = \sum_L \langle ij|L\rangle S_{LM}^{-1}. \tag{D-12}$$

Sustituyendo la ecuación D-12 en la ecuación D-3, llegamos a

$$\begin{aligned} (ij|kl) \approx \sum_{LM} c_{ij}^L c_{kl}^M (L|M) &= \sum_{LM}\sum_{OP} (ij|O)S_{OL}^{-1}(kl|P)S_{PM}^{-1}(L|M) \\ &= \sum_{LM}\sum_{OP} (ij|O)S_{OL}^{-1}V_{LM}S_{MP}^{-1}(kl|P), \end{aligned} \tag{D-13}$$

La ecuación D-13 es una de las formas de la aproximación RI y es comúnmente denominada aproximación "RI-SVS", por la forma que adopta la ecuación que aproxima las integrales bielectrónicas.

Un mejor criterio para obtener el conjunto $\{c_{ij}^L\}$, consiste en minimizar el error asociado a la

evaluación de las integrales bielectrónicas de cuatro centros, el cual está dado por

$$\delta I_{ij,kl} = (\tilde{p}_{ij}|\tilde{p}_{kl}) - (p_{ij}|p_{kl}). \tag{D-14}$$

Whitten[70], ha demostrado que la minimización de este error puede llevarse a cabo minimizando $\delta\epsilon_{ij} = (\delta p_{ij}|\delta p_{ij})$ y $\delta\epsilon_{kl} = (\delta p_{kl}|\delta p_{kl})$, de forma independiente.
La minimización $\delta\epsilon_{ij}$ respecto a c_{ij}^K, como se ilustra enseguida

$$\begin{aligned} \frac{\partial \delta\epsilon_{ij}}{\partial c_{ij}^K} &= \frac{\partial}{\partial c_{ij}^K}\left[\sum_{LM} c_{ij}^L c_{ij}^M (L|M) - 2\sum_L c_{ij}^L (L|ij)\right] \\ &= \sum_M c_{ij}^M (K|M) - 2(K|ij) = 0, \end{aligned} \tag{D-15}$$

da lugar a la siguiente expresión para los coeficientes c_{ij}^K:

$$c_{ij}^K = \sum_L (ij|L) V_{LK}^{-1}, \tag{D-16}$$

donde

$$(ij|K) = \int\int \frac{\phi_i(\mathbf{r})\phi_j(\mathbf{r}) P_K(\mathbf{r}')}{|\mathbf{r}-\mathbf{r}'|} d\mathbf{r} d\mathbf{r}', \tag{D-17}$$

y la matriz coulómbica $\mathbf{V}$ es definida de acuerdo a la ecuación D-4. Insertando este resultado en la ecuación D-3, llegamos a

$$\begin{aligned} (ij|kl) &\approx \sum_{LM}\sum_{OP} (ij|O) V_{OL}^{-1} V_{LM} V_{MP}^{-1} (P|kl) \\ &= \sum_{LO} (ij|O) V_{OL}^{-1} (L|kl) \\ &= \sum_{LMO} (ij|O)(O|M)^{-1/2}(M|L)^{-1/2}(L|kl) \\ &= \sum_M B_{ij}^M B_{kl}^M \end{aligned} \tag{D-18}$$

en donde se ha hecho uso de $\sum_M V_{LM} V_{MP}^{-1} = \delta_{LP}$ y de la definición de la raíz de una matriz. Este último método es comúnmente conocido como el método "RI-V", el cual es sabido que es superior al método "RI-SVS" mencionado anteriormente. Para comprobarlo, esto se puede

verificar de la siguiente manera, considerando el caso del error en la evaluación de la auto-repulsión coulómbica

$$\delta I_{ij,ij} = (\tilde{p}_{ij}|\tilde{p}_{ij}) - (p_{ij}|p_{ij}) = 2(\delta p_{ij}|\tilde{p}_{ij}) - (\delta p_{ij}|\delta p_{ij}). \tag{D-19}$$

En la aproximación RI-V, el primer término de la ecuación de arriba es igual a cero, como se verifica de acuerdo a

$$\begin{aligned}(\delta p_{ij}|\tilde{p}_{ij}) = (\tilde{p}_{ij}|\tilde{p}_{ij}) - (p_{ij}|\tilde{p}_{ij}) &= \sum_{OP} c_{ij}^{O} V_{OP} c_{ij}^{P} - \sum_{P} (ij|P) c_{ij}^{P} \\ &= \sum_{P} (ij|P) c_{ij}^{P} - \sum_{P} (ij|P) c_{ij}^{P} = 0. \end{aligned} \tag{D-20}$$

En contraste, en el método RI-SVS, el primer término es diferente de cero y representa la contribución dominante al error total. Por esta razón, el método RI-V es el preferido en las implementaciones computacionales de estructura electrónica.

Bibliografía

[1] Kirchhoff, G. "Über das Verhältniss zwischen dem Emissionsvermögen und dem Absorptionsvermögen der Körper für Wärme und Licht". *Ann. Phys.* **1860**, *185*, 275–301.

[2] Boltzmann, L. "Weitere Studien ber das Wrmegleichgewicht unter Gasmoleklen". *Sitzungsberichte der Akademie der Wissenschaften, Mathematische-Naturwissenschaftliche Klasse* **1872**, *2*, 275–370.

[3] Planck, M. "Ueber das Gesetz der Energieverteilung im Normalspectrum". *Ann. Phys.* **1901**, *4*, 553–563.

[4] Einstein, A. "Über einen die Erzeugung und Verwandlung des Lichtes betreffenden heuristischen Gesichtspunkt". *Ann. Phys.* **1905**, *322*, 132–148.

[5] Einstein, A.; Podolsky, B.; Rosen, N. "Can Quantum-Mechanical Description of Physical Reality Be Considered Complete?". *Phys. Rev.* **1935**, *47*, 777–780.

[6] Bengtson, C.; Stenrup, M.; Sjoqvist, E. "Nonlocality in the excitation energy transfer in the Fenna-Matthews-Olson complex". 2015; `arXiv:1502.07842v1`.

[7] Liu, S. G.; Carlson, R. M. K.; Tucciarone, A. "Coherence Dynamics in Photosynthesis: Protein Protection of Excitonic Coherence". *Science.* **2007**, *316*, 1462–1465.

[8] Hildner, R.; Brinks, D.; Stefani, F. D.; van Hulst, N. F. "Electronic coherences and vibrational wave-packets in single molecules studied with femtosecond phase-controlled spectroscopy.". *Phys. Chem. Chem. Phys.* **2011**, *13*, 1888–1894.

[9] Garraway, B. M.; Suominen, K.-A. "Wave packet dynamics in molecules". *Contemp. Phys.* **2002**, *43*, 97–114.

[10] Brinks, D.; Stefani, F. D.; Kulzer, F.; Hildner, R.; Taminiau, T. H.; Avlasevich, Y.; Müllen, K.; van Hulst, N. F. "Visualizing and controlling vibrational wave packets of single molecules.". *Nature* **2010**, *465*, 905–908.

[11] McQuarrie, D. A. *"Quantum Chemistry"*; University of Science Books: Mill Valley, CA, 1983; p 291.

[12] Bohr, N. "On the Constitution of Atoms and Molecules". *Philos. Mag.* **1913**, *26*, 1–13.

[13] Born, M. and Oppenheimer, R. "Zur Quantentheorie der Molekeln". *Ann. Phys.* **1927**, *389*, 457–484.

[14] Uhlenbeck, G. E. and Goudsmit, S. "Spinning Electrons and the Structure of Spectra". *Nature* **1926**, *117*, 264–265.

[15] Dirac, P. A. M. "The Quantum Theory of the Electron". *Proc. R. Soc. London A Math. Phys. Eng. Sci.* **1928**, *117*, 610–624.

[16] Jinich, A.; Rappoport, D.; Dunn, I.; Sanchez-Lengeling, B.; Olivares-Amaya, R.; Noor, E.; Even, A. B.; Aspuru-Guzik, A. "Quantum Chemical Approach to Estimating the Thermodynamics of Metabolic Reactions". *Sci. Rep.* **2014**, *4*, 7022.

[17] Fomina, E. S.; Vysotsky, Y. B.; Vollhardt, D.; Fainerman, V. B.; Miller, R. "Quantum chemical analysis of the thermodynamics of 2D cluster formation of 2-hydroxycarboxylic acids at the air/water interface". *Soft Matter* **2013**, *9*, 7601.

[18] Vysotsky, Y. B.; Belyaeva, E. a.; Fomina, E. S.; Vollhardt, D.; Fainerman, V. B.; Miller, R. "The quantum-chemical approach to calculations of thermodynamic and structural parameters of formation of fatty acid monolayers with hexagonal packing at the air/water interface.". *Phys. Chem. Chem. Phys.* **2014**, *16*, 3187–99.

[19] Fukui, K. "The Path of Chemical Reactions - The IRC Approach". *Acc. Chem. Res.* **1981**, *14*, 363–368.

[20] Himo, F. "Quantum chemical modeling of enzyme active sites and reaction mechanisms". *Theor. Chem. Acc.* **2005**, *116*, 232–240.

[21] Blomberg, M. R. A.; Siegbahn, P. E. M. "A Quantum Chemical Approach to the Study of Reaction Mechanisms of Redox-Active". *J. Phys. Chem. B* **2001**, *105*, 9375–9386.

[22] Bartlett, R. J. "Coupled-Cluster Approach to Molecular Structure and Spectra : A Step toward Predictive Quantum Chemistry". *J. Phys. Chem.* **1989**, *2*, 1697–1708.

[23] Diedrich, C.; Grimme, S. "Systematic Investigation of Modern Quantum Chemical Methods to Predict Electronic Circular Dichroism Spectra". *J. Phys. Chem. A* **2003**, *107*, 2524–2539.

[24] Alcolea, M.; Rastogi, V. K. "Quantum chemical predictions of the vibrational spectra of polyatomic molecules . The uracil molecule and two derivatives". *Spectrochim. Acta Part A.* **2002**, *58*, 411–440.

[25] Szabo, A.; Ostlund, N. S. *"Modern Quantum Chemistry. Introduction to Advanced Electronic Structure Theory"*, 1st ed.; Dover Publications: New York, 1989; pp 68–74.

[26] Goldstone, J. "Derivation of the Brueckner Many-Body Theory". *Proc. R. Soc. London* **1957**, *239*, 267.

[27] Herbert, J. M.; Ermier, W. C. "Symbolic implementation of arbitrary-order perturbation theory using computer algebra: application to vibrational-rotational analysis of diatomic molecules". *Comput. Chem.* **1998**, *22*, 169–184.

[28] Pople, J. A. and Binkley, J. S. and Seeger, R. "Theoretical models incorporating electron correlation". *Int. J. Quantum Chem.* **1976**, *S10*, 1–19.

[29] Grimme, S. "Improved second-order MøllerPlesset perturbation theory by separate scaling of parallel- and antiparallel-spin pair correlation energies". *J. Chem. Phys.* **2003**, *118*, 9095.

[30] Jung, Y.; Lochan, R. C.; Dutoi, A. D.; Head-gordon, M. "Scaled opposite-spin second order Møller Plesset correlation energy : An economical electronic structure method Scaled opposite-spin second order Møller Plesset correlation energy : An economical electronic structure method"". *J. Chem. Phys.* **2004**, *121*, 9793.

[31] Feyereisen, M. and Fitzgerald, G. and Komornicki, A. "Use of approximate integrals in ab initio theory. An application in MP2 energy calculations". *Chem. Phys. Lett.* **1993**, *208*, 359–363.

[32] Almlof, J. "Elimination of energy denominators in MøllerPlesset perturbation theory by a Laplace transform approach". *Chem. Phys. Lett.* **1991**, *181*, 319.

[33] Møller, C.; Plesset, M. S. "Note on an approximation treatment for many-electron systems". *Phys. Rev.* **1934**, *46*, 618–622.

[34] Shao, Y.; Gan, Z.; Epifanovsky, E.; Gilbert, a. T. B.; Wormit, M.; Kussmann, J.; Lange, a. W.; Behn, A.; Deng, J.; Feng, X.; Ghosh, D.; Goldey, M.; Horn, P. R.; Jacobson, L. D.; Kaliman, I. et al. "Advances in Molecular Quantum Chemistry Contained in the Q-Chem 4 Program Package". *Mol. Phys.* **2014**, *113*, 184–215.

[35] Jung, Y.; Sodt, A.; Gill, P. M. W.; Head-Gordon, M. "Auxiliary basis expansions for large-scale electronic structure calculations.". *Proc. Natl. Acad. Sci. U. S. A.* **2005**, *102*, 6692–6697.

[36] Turing, A. M. "On computable numbers, with an application to the Entscheidungsproblem". *Proc. London Math. Soc.* **1937**, *42*, 230–265.

[37] Bair, R. A. and Dunning, T. H. J. "Quantum chemistry with an attached processor". *J. Comput. Chem.* **1984**, *5*, 44–55.

[38] Harrison, R. J. "Portable tools and applications for parallel computers". *Int. J. Quantum Chem* **1991**, *40*, 847–863.

[39] Sunderam, V. S. "PVM: a framework for parallel distributed computing". *Concurrency:. Pract. Exper.* **1990**, *2*, 315.

[40] "The Message Passing Interface (MPI) standard". `http://www.mcs.anl.gov/research/projects/mpi/`.

[41] te Velde, G.; Bickelhaupt, F. M.; Baerends, E. J.; Fonseca Guerra, C.; van Gisbergen, S. J. A.; Snijders, J. G.; Ziegler, T. "Chemistry with ADF". *J. Comput. Chem.* **2001**, *22*, 931–967.

[42] Gonze, X.; Amadon, B.; Anglade, P.-M.; Beuken, J.-M.; Bottin, F.; Boulanger, P.; Bruneval, F.; Caliste, D.; Caracas, R.; Côté, M.; Deutsch, T.; Genovese, L.; Ghosez, P.; Giantomassi, M.; Goedecker, S. et al. "ABINIT: First-principles approach to material and nanosystem properties". *Comput. Phys. Commun.* **2009**, *180*, 2582–2615.

[43] Soler, J. M.; Artacho, E.; Gale, J. D.; García, A.; Junquera, J.; Ordejón, P.; Sánchez-Portal, D. "The SIESTA method for ab initio order- N materials simulation". *J. Phys. Condens. Matter* **2002**, *14*, 2745–2779.

[44] Giannozzi, P.; Baroni, S.; Bonini, N.; Calandra, M.; Car, R.; Cavazzoni, C.; Ceresoli, D.; Chiarotti, G. L.; Cococcioni, M.; Dabo, I.; Corso, A. D.; de Gironcoli, S.; Fabris, S.; Fratesi, G.; Gebauer, R. et al. "QUANTUM ESPRESSO: a modular and open-source software project for quantum simulations of materials". *J. Phys. Condens. Matter* **2009**, *21*, 395502.

[45] Vandevondele, J.; Krack, M.; Mohamed, F.; Parrinello, M.; Chassaing, T.; Hutter, J. "Quickstep: Fast and accurate density functional calculations using a mixed Gaussian and plane waves approach". *Comput. Phys. Commun.* **2005**, *167*, 103–128.

[46] de Jong, W. A.; Bylaska, E.; Govind, N.; Janssen, C. L.; Kowalski, K.; Müller, T.; Nielsen, I. M. B.; van Dam, H. J. J.; Veryazov, V.; Lindh, R. "Utilizing high performance computing for chemistry: parallel computational chemistry.". *Phys. Chem. Chem. Phys.* **2010**, *12*, 6896–920.

[47] Götz, Andreas W. and Wölfle,Thorsten and Walker, R. C. "Quantum Chemistry on Graphics Processing Units". *Annu. Rep. Comput. Chem.* **2010**, *6*, 21–35.

[48] Yasuda, K. "Two-Electron Integral Evaluation on the Graphics". *J. Comput. Chem.* **2007**, *29*, 334–342.

[49] Dupuis, M.; Rys, J.; King, H. F. "Evaluation of molecular integrals over Gaussian basis functions". *J Chem Phys* **1976**, *65*, 111.

[50] King, H. F.; Dupuis, M. "Numerical integration using rys polynomials". *J Comput Phys* **1976**, *21*, 144.

[51] Ufimtsev, I. S.; Martínez, T. J. "Strategies for Two-Electron Integral Evaluation". *J. Chem. Theory Comput.* **2008**, *4*, 222–231.

[52] Ufimtsev, I. S.; Martínez, T. J. "Graphical processing units for quantum chemistry". *Comput. Sci. Eng.* **2008**, *10*, 26–34.

[53] DePrince III, a. E.; Hammond, J. R. "Coupled Cluster Theory on Graphics Processing Units I. The Coupled Cluster Doubles Method". *J. Chem. Theory Comput.* **2011**, *7*, 1287–1295.

[54] Ufimtsev, I. S.; Martinez, T. J. "Quantum chemistry on graphical processing units. 3. Analytical energy gradients, geometry optimization, and first principles molecular dynamics". *J. Chem. Theory Comput.* **2009**, *5*, 2619–2628.

[55] Yasuda, K. "Accelerating Density Functional Calculations with Graphics Processing Unit". *J. Chem. Theory Comput.* **2008**, *4*, 1230–1236.

[56] Jensen, F. " An Introduction to the State of the Art in Quantum Chemistry". *Annu. Rep. Comput. Chem.* **2005**, *1*, 3–17.

[57] Kristyán, S.; Pulay, P. "Can (semi)local density functional theory account for the London dispersion forces?". *Chem. Phys. Lett.* **1994**, *229*, 175–180.

[58] Vogt, L.; Olivares-Amaya, R.; Kermes, S.; Shao, Y.; Amador-Bedolla, C.; Aspuru-Guzik, A. "Accelerating resolution-of-the-identity second-order Møller-Plesset quantum chemistry calculations with graphical processing units.". *J. Phys. Chem. A* **2008**, *112*, 2049–57.

[59] Olivares-Amaya, R.; Watson, M. A.; Edgar, R. G.; Vogt, L. "Matrix Multiplication Library". *J. Chem. Theory Comput.* **2010**, *6*, 135–144.

[60] Guide, U. *"Cublas library"*; 2014.

[61] Anderson, A.; Goddard-III, W.; Schroder, P. "Quantum Monte Carlo on graphical processing units". *Comput. Phys. Commun.* **2007**, *177*, 298–306.

[62] NVIDIA CORPORATION, *"CUDA C Programming Guide*.

[63] Harris, M. "How to Overlap Data Transfers in CUDA C/C++". `http://devblogs.nvidia.com/parallelforall/how-overlap-data-transfers-cuda-cc/`.

[64] Manual, R. *"Intel Math Kernel Library"*; 2008.

[65] Betkaoui, B.; Thomas, D. B.; Luk, W. "Comparing performance and energy efficiency of FPGAs and GPUs for high productivity computing". "Proc. - 2010 Int. Conf. Field-Programmable Technol. FPT'10". 2010; pp 94–101.

[66] Davis, P. J.; Rabinowitz, P. *"Methods of Numerical Integration"*, 2nd ed.; Academic Press, Inc.: London, 1984.

[67] Hamming, R. W. *"Numerical Methods for Scientists and Engineers"*; Mc Graw Hill, 1962.

[68] Kiusalaas, J. *"Numerical Methods in Engineering with Python"*, 1st ed.; Cambridge University Press: New York, 2005.

[69] Ren, Xinguo and Rinke, Patrick and Blum, Volker and Wieferink, Jurgen and Tkatchenko, Alexandre and Sanfilipo, Andrea and Reuter, Karsten and Scheffler, M. "Resolution-of-identity approach to HartreeFock, hybrid density functionals, RPA, MP2 and GW with numeric atom-centered orbital basis functions". *New J. Phys.* **2012**, *14*, 053020.

[70] Whitten, J. L. "Coulombic potential energy integrals and approximations". *J. Chem. Phys.* **1973**, *58*, 4496.