Universidad Zaragoza

Trabajo Fin de Máster

Caracterización y optimización de algoritmos para imponer ligaduras en simulaciones de Dinámica Molecular en GROMACS

Autora:

María Astón Serrano Gracia

Directores:

Pablo García Risueño

Institut für Physik und IRIS Adlershof
Humboldt Universität zu Berlin

Jesús Alastruey Benedé

Dpto. Informática e Ingeniería de Sistemas
Universidad de Zaragoza

Máster en Ingeniería de Sistemas e Informática
Escuela de Ingeniería y Arquitectura
Universidad de Zaragoza

Junio 2014

Resumen

La Dinámica Molecular permite describir la evolución en el tiempo de sistemas de partículas mediante simulaciones del movimiento de los átomos y moléculas. Herramientas como Gromacs, útiles para realizar estas simulaciones, son fundamentales en la investigación de áreas como la medicina, la química o la ingeniería de materiales.

Una opción común para realizar simulaciones más eficientes es imponer restricciones, denominadas ligaduras, en la longitud de los enlaces atómicos o en los ángulos entre enlaces. Esta técnica permite incrementar el paso temporal de las simulaciones y con ello lograr mayores tiempos simulados, con lo que se incrementa el poder predictivo de la simulación. Uno de los algoritmos más extendidos para imponer ligaduras es SHAKE, pero presenta limitaciones puesto que está basado en aproximaciones y procesos iterativos, lo cual afecta negativamente a su exactitud, eficiencia y estabilidad numérica. ILVES-S es un nuevo algoritmo, basado en la propuesta teórica del Dr. Pablo García Risueño, cuya implementación en Gromacs se realizó como proyecto fin de carrera por la autora de este trabajo.

Hemos caracterizado estos dos algoritmos para imponer ligaduras con respecto al tiempo de ejecución, las operaciones de coma flotante y la jerarquía de memoria. El objetivo es identificar posibles cuellos de botella que pudieran ser optimizados.

Como resultado de la caracterización, hemos encontrado muy buenos resultados en cuanto a la convergencia del método y el número de operaciones en coma flotante, pero ciertas simulaciones son más lentas con el nuevo algoritmo. Hemos verificado cuales son las funciones más costosas, encontrando que ILVES-S realiza muchos más accesos a memoria que SHAKE. Por último, sugerimos posibles optimizaciones que quedan planteadas como trabajo futuro.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

La Dinámica Molecular es una técnica de simulación por computador en la que átomos y moléculas interactúan durante un período de tiempo, dando lugar a una descripción visual y detallada del movimiento de las partículas [1]. Estas simulaciones se basan en aproximaciones de la física conocida y se utilizan con frecuencia en el estudio de muchas moléculas bioquímicas como proteínas, péptidos, lípidos y ácidos nucleicos. Llevar a cabo ciertos experimentos en un laboratorio conlleva un alto coste de recursos y no siempre es posible analizar las propiedades en todas las escalas temporales. Por lo tanto, las herramientas computacionales que realizan dichas simulaciones son fundamentales para los investigadores de campos tan diversos como la medicina (búsqueda de tratamientos para enfermedades como el Alzheimer, la fibrosis quística o el cáncer; diseño computacional de medicamentos [2]), la química (diseño de catalizadores) o la ingeniería de materiales [3].

Debido a la complejidad de los sistemas biológicos, sus simulaciones conllevan un alto consumo de recursos, tanto de memoria como de cálculo. La demanda de experimentos más precisos y con sistemas cada vez más complejos impulsa la búsqueda constante de mejoras en los algoritmos de Dinámica Molecular y en sus implementaciones.

Una técnica muy común que permite realizar simulaciones más eficientes es aumentar el paso temporal (*time step*) cuyos valores típicos son del orden de un femtosegundo ($10^{-15}s$) [4]. Este valor puede ser ampliado si se usan algoritmos como SHAKE [5], que permiten imponer ligaduras para fijar las vibraciones de los átomos

más rápidos. ILVES-S es un nuevo algoritmo basado en un método alternativo para resolver las ecuaciones algebraicas originadas por la imposición de ligaduras [6].

La autora de este trabajo fin de Máster realizó la implementación de una primera versión del algoritmo ILVES-S en GROMACS, paquete software de Dinámica Molecular [7], como parte de su proyecto fin de carrera. En este trabajo, caracterizamos el comportamiento de estos dos algoritmos para imponer ligaduras, el original SHAKE, de 1977, y nuestra propuesta, ILVES-S. Caracterizamos los algoritmos con respecto al tiempo de ejecución, la jerarquía de memoria o la capacidad de cálculo. El objetivo final es identificar posibles optimizaciones en la versión secuencial del algoritmo ILVES-S y obtener información útil para próximas líneas de trabajo como la paralelización del algoritmo.

1.1. Ámbito del proyecto

Este trabajo se desarrolla en el grupo de Arquitectura de Computadores de la Universidad de Zaragoza (gaZ). Además, cuenta con la colaboración del físico Dr. Pablo García Risueño (Humboldt Universität zu Berlin, Alemania), autor de la propuesta teórica para mejorar la implementación de ligaduras en simulaciones de Dinámica Molecular, y del matemático Dr. Carl Christian Kjelgaard Mikkelsen (Department of Computing Science and HPC2N, Umeå University, Sweden). Se trata de un proyecto de ámbito internacional formado por un equipo multidisciplinar.

El trabajo ha sido financiado mediante los proyectos TIN2010-21291-C02-01 (Gobierno de España y FEDER), Consolider CSD2007-00050 (Gobierno de España), gaZ: grupo de investigación T48 (Gobierno de Aragón y Fondo Social Europeo) y HiPEAC-3 NoE (European FET FP7/ICT 287759).

1.2. Objetivos

Los objetivos principales de este trabajo fin de Máster son:

- Definir la metodología usada para caracterizar los algoritmos. Esto incluye el entorno experimental, las herramientas, las características a analizar y las métricas usadas, el tipo de simulaciones y la carga de trabajo adecuada.

- Caracterizar los dos algoritmos para imponer ligaduras mediante ejecuciones clásicas o ejecuciones monitorizadas por una herramienta externa.
- Describir e implementar primeras optimizaciones, basadas en el resultado de la caracterización pero también en los conocimientos físicos y matemáticos de los colaboradores del proyecto.
- Analizar los resultados obtenidos en la caracterización y estudiar a partir de ellos las nuevas líneas de trabajo.

1.3. Organización del documento

El resto de este trabajo fin de Máster está organizado de la siguiente manera: el Capítulo 2 describe las simulaciones de Dinámica Molecular, los dos algoritmos para imponer ligaduras SHAKE e ILVES-S y la optimización previa del algoritmo ILVES-S; en el Capítulo 3 se detalla la metodología que hemos seguido; el Capítulo 4 presenta la caracterización realizada junto con el análisis de los resultados obtenidos y el Capítulo 5 detalla las conclusiones y trabajo futuro.

En forma de Apéndice encontramos la siguiente información:

A. Implementación de un nuevo algoritmo para imponer ligaduras en Dinámica Molecular. Artículo que será presentado en las XXV Jornadas de Paralelismo, Valladolid, Septiembre de 2014.

B. Implementation of a new constraint algorithm for Molecular Dynamics. Artículo que será presentado en forma de póster en la X Escuela de Verano Internacional ACACES, Fiuggi (Italia), Julio de 2014.

1.4. Contribuciones

Resumiendo, las principales contribuciones de este trabajo son:

- Hemos aplicado la metodología definida para caracterizar los algoritmos SHAKE e ILVES-S. Además, hemos definido y configurado la estructura de nuestro

proyecto en git [8], una herramienta de control de versiones, con acceso para todos los miembros del equipo.

- Hemos caracterizado los dos algoritmos para imponer ligaduras atendiendo a sus características temporales, de cálculo y de acceso a memoria.
- Hemos implementado una optimización para mejorar la convergencia del algoritmo ILVES-S y descrito una posible alternativa basada en los resultados de la caracterización.
- Los resultados obtenidos en la caracterización nos permiten definir las líneas de trabajo futuras con el objetivo de obtener un algoritmo que pueda sustituir al clásico SHAKE.
- El trabajo realizado supone la admisión de un artículo en una conferencia nacional y la publicación de un artículo y su presentación como póster en una escuela de verano internacional.

1.5. Agradecimientos

Primero de todo me gustaría agradecer al gaZ por su confianza en mí y especialmente a Chus por su enorme ayuda que ya comenzó durante el desarrollo del proyecto fin de carrera. Gracias a Pablo y Carl Christian por todo lo que he aprendido con ellos y por sus palabras de ánimo.

A los amigos que me rodean, muchas gracias por vuestra compañía. Especialmente a los que me han acompañado este último año en la universidad, en poco tiempo me habéis hecho sentir una más. Víctor gracias por todo lo que hemos vivido, todavía queda mucho por descubrir juntos.

A toda mi familia, mi padre, mi madre, mis hermanos y muy especialmente a mi abuelo, por su ánimo para que siga por este camino y por su alegría por cada uno de mis pequeños logros.

Capítulo 2

Dinámica Molecular

2.1. Simulaciones en GROMACS

El objetivo de las simulaciones de Dinámica Molecular (DM) es describir la evolución en el tiempo de un sistema de partículas. Para obtener la trayectoria del sistema se resuelve, en sucesivos instantes temporales, la ecuación clásica del movimiento según la segunda ley de Newton:

$$m_i \frac{d^2\mathbf{x}_i(t)}{dt^2} = \mathbf{F}_i(t) \tag{2.1}$$

donde $\mathbf{x}_i(t)$ es el vector posición (x_x, x_y, x_z) de la partícula i en el instante temporal t, $\mathbf{F}_i(t)$ es el vector de la fuerza que actúa sobre la partícula i-ésima en el instante temporal t debida a la interacción con el resto de partículas del sistema, y m_i es la masa de la partícula i.

La efectividad de las simulaciones de DM está fuertemente limitada por los pequeños pasos temporales (*time steps*, Δt). Una técnica muy extendida para poder aumentar significativamente el paso temporal de la simulación y, por tanto, el tiempo simulado del experimento es incluir ligaduras al sistema. Una ligadura es una restricción en la longitud de los enlaces atómicos o en el valor de los ángulos entre enlaces. Para incluir estas restricciones se añaden las denominadas fuerzas de ligadura a la ecuación clásica del movimiento de Newton (2.1). Asumiendo un sistema de N partículas que debe cumplir con N_c ligaduras, éstas se expresan como:

$$\sigma_k(\mathbf{x}_0...\mathbf{x}_{N-1}) = 0; \qquad k = 0...N_c - 1 \tag{2.2}$$

Por ejemplo, la Figura 2.1 muestra la ligadura de un enlace k entre dos átomos, i y j, definida como:

$$\sigma_{k(i,j)} := (\mathbf{x}_i - \mathbf{x}_j)^2 - (a_{i,j})^2 = 0 \tag{2.3}$$

donde $\mathbf{x}_i$ e $\mathbf{x}_j$ son los vectores posición de los átomos i y j. Esta ligadura expresa que la distancia entre estos átomos debe ser igual a $a_{i,j}$.

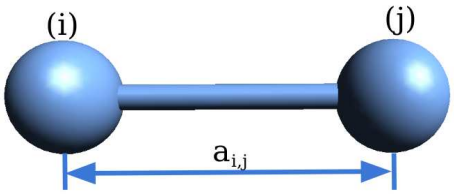

Figura 2.1: Ligadura en un enlace.

Al integrar las fuerzas de ligadura en la ecuación (2.1), el nuevo sistema de ecuaciones es el siguiente:

$$m_i \frac{d^2\mathbf{x}_i(t)}{dt^2} = \mathbf{F}_i(t) - \sum_{k=0}^{N_c-1} \lambda_k \frac{\partial \sigma_k}{\partial \mathbf{x}_i}(t) \tag{2.4}$$

donde $-\sum_{k=0}^{N_c-1} \lambda_k \frac{\partial \sigma_k}{\partial \mathbf{x}_i}(t)$ es la fuerza sobre el átomo i en el instante t que aparece debida a las ligaduras y los distintos λ_k, $k = 0...N_c - 1$ son los multiplicadores de Lagrange.

La imposición de las N_c ligaduras convierte un sistema de $3N$ ecuaciones diferenciales con $3N$ incógnitas en un sistema de $3N + N_c$ ecuaciones diferenciales con $3N + N_c$ incógnitas.

GROMACS es un versátil paquete software de Dinámica Molecular para llevar a cabo este tipo de simulaciones. Está escrito en el lenguaje de programación C y se puede ejecutar en todo tipo de plataformas desde ordenadores convencionales a supercomputadores y tanto en CPUs como en GPUs.

GROMACS resuelve el sistema de ecuaciones 2.4 en dos fases: en primer lugar se calculan las nuevas posiciones de los átomos sin tener en cuenta las ligaduras impuestas al sistema y, en segundo lugar, se calculan las fuerzas de ligadura y se corrigen

las posiciones. De la segunda fase se encargan los algoritmos para la imposición de ligaduras como SHAKE o nuestra propuesta ILVES-S.

Una vez resuelta, para cada átomo, la ecuación clásica del movimiento de Newton (2.1), es necesario resolver las ecuaciones para obtener las fuerzas de ligadura cuyas incógnitas son los multiplicadores de Lagrange.

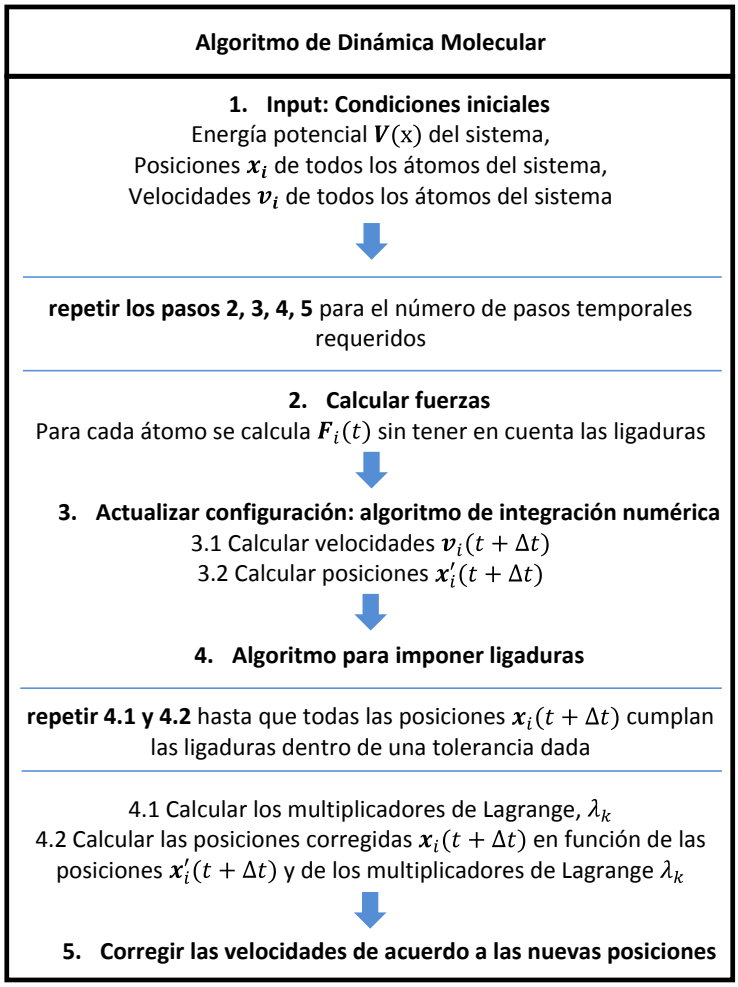

Figura 2.2: Esquema del algoritmo de Dinámica Molecular.

La Figura 2.2 muestra de forma esquemática la implementación en GROMACS del algoritmo de Dinámica Molecular. Nuestro trabajo se centra en caracterizar dos algoritmos para imponer ligaduras, es decir, nos centraremos, a partir de ahora, en el punto 4 de la Figura 2.2: el cálculo de las fuerzas de ligadura para corregir las

posiciones y velocidades de los átomos.

2.2. Algoritmos para la imposición de ligaduras

En esta Sección presentamos el algoritmo original, SHAKE, y el algoritmo alternativo, ILVES-S.

2.2.1. SHAKE

El algoritmo SHAKE calcula, en cada instante temporal $(t+\Delta t)$, un conjunto de coordenadas $x_i(t+\Delta t)$ que cumplen con la lista de ligaduras, a partir de las coordenadas $x_i(t)$ y de las coordenadas obtenidas sin considerar las fuerzas de ligadura $x'_i(t+\Delta t)$ (coordenadas a corregir).

$$\begin{aligned} &-2\Delta t^2(x'_i(t+\Delta t)-x'_j(t+\Delta t))\left(\sum_{k'=0}^{N_c-1}\gamma_{k'}(\frac{\nabla_i\sigma_{k'}}{m_i}(t)-\frac{\nabla_j\sigma_{k'}}{m_j}(t))\right)\\ &+\Delta t^4\sum_{k'=0}^{N_c-1}\sum_{k''=0}^{N_c-1}\gamma_{k'}\gamma_{k''}\left(\frac{\nabla_i\sigma_{k'}}{m_i}(t)-\frac{\nabla_j\sigma_{k'}}{m_j}(t)\right)\left(\frac{\nabla_i\sigma_{k''}}{m_i}(t)-\frac{\nabla_j\sigma_{k''}}{m_j}(t)\right)\\ &=(a_{i,j})^2-(x'_i(t+\Delta t)-x'_j(t+\Delta t))^2 \qquad \text{para } k=0...N_c-1. \end{aligned} \tag{2.5}$$

Para calcular los multiplicadores de Lagrange (y con ello, las fuerzas de ligadura (2.4)) es necesario resolver el sistema de N_c ecuaciones cuadráticas (2.5), con N_c el número de ligaduras. En realidad, SHAKE obtiene una aproximación, γ_k, de los multiplicadores de Lagrange, λ_k, resolviendo un sistema de N_c ecuaciones cuadráticas en γ_k que tiene la forma:

$$\begin{cases} a_{0,0}\gamma_0+a_{0,1}\gamma_1+...+(\gamma_0\gamma_0c_{0,0}+\gamma_0\gamma_1c_{0,1}+...)=b_0\\ a_{1,0}\gamma_0+a_{1,1}\gamma_1+...+(\gamma_0\gamma_0c_{1,0}+\gamma_0\gamma_1c_{1,1}+...)=b_1\\ a_{2,0}\gamma_0+a_{2,1}\gamma_1+...+(\gamma_0\gamma_0c_{2,0}+\gamma_0\gamma_1c_{2,1}+...)=b_2\\ ... \end{cases} \tag{2.7}$$

Excepto las γ_k, todos los términos de estas ecuaciones son conocidos: k es la

ligadura que involucra a los átomos i y j y (equivalentemente a $\nabla_j \sigma_{k'}$, $\nabla_i \sigma_{k''}$, $\nabla_j \sigma_{k''}$):

$$\nabla_i \sigma_{k'(l,m)} = 2(\mathbf{x}_l - \mathbf{x}_m)(\delta_{i,l} - \delta_{i,m}) \tag{2.8}$$

donde $\delta_{i,l}$ y $\delta_{i,m}$ representan la delta de Kronecker, función matemática de dos variables que vale 1 si son iguales y 0 si son diferentes:

$$\delta_{\alpha,\beta} = \begin{cases} 1 & \text{si } \alpha = \beta, \\ 0 & \text{si } \alpha \neq \beta. \end{cases}$$

Como hemos dicho, tenemos un sistema de N_c ecuaciones cuadráticas en γ_k (2.5) que se puede resolver de forma iterativa (método de Newton, primer proceso iterativo). La implementación en GROMACS del algoritmo SHAKE considera que los términos en γ_k^2 (es decir el término en Δt^4) son despreciables puesto que Δt, el tamaño del paso temporal, es del orden de los femtosegundos. En cada una de las iteraciones, it, del algoritmo se resuelve el sistema lineal:

$$A\gamma^{(it)} = b \tag{2.9}$$

donde, según el sistema (2.5), despreciando el término en Δt^4, y teniendo en cuenta la ecuación (2.8):

$$\begin{aligned} A_{kk'} := -2\Delta t^2 \Big(x_i'(t+\Delta t) - x_j'(t+\Delta t) \Big) 2(x_l(t) - x_m(t)) \\ \Big(\frac{\delta(i,l)}{m_i} - \frac{\delta(i,m)}{m_i} - \frac{\delta(j,l)}{m_j} + \frac{\delta(j,m)}{m_j} \Big), \end{aligned} \tag{2.10}$$

$$b_k := (a_{i,j})^2 - (x_i'(t+\Delta t) - x_j'(t+\Delta t))^2, \tag{2.11}$$

b es un vector de dimensión N_c y A es una matriz cuadrada en el número de ligaduras N_c y dispersa. Un elemento $A_{k,k'}$ es distinto de cero si las ligaduras, $k(i,j)$ y $k'(l,m)$ comparten algún átomo, esto es, $i = l$ o $i = m$ o $j = l$ o $j = m$, e igual a cero si todos los átomos involucrados en las ligaduras $k(i,j)$ y $k'(l,m)$ son distintos.

La resolución en cada iteración it del sistema $A\gamma^{(it)} = b$ se lleva a cabo de forma aproximada, de nuevo mediante un proceso iterativo (segundo proceso iterativo, N_c

iteraciones), puesto que se resuelve en primer lugar la primera ecuación, después la segunda y así sucesivamente. Se obtiene primero $\gamma_0^{(0)}$, considerando que $\gamma_1^{(0)}...\gamma_{N_c-1}^{(0)}$ son iguales a cero en la primera iteración. En las sucesivas iteraciones se usan los γ_k obtenidos en la iteración anterior. Es decir, poniendo como ejemplo el sistema (2.7), en la primera iteración (del primer proceso iterativo) se resuelve la primera ecuación de (2.7), forzando $\gamma_1 = 0, \gamma_2 = 0...$, para así obtener γ_0, después se resuelve la segunda ecuación de (2.7) usando el γ_0 obtenido y haciendo $\gamma_2 = 0, \gamma_3 = 0...$, y así sucesivamente.

De esta manera, la resolución de la ecuación k-ésima garantiza la satisfacción de la ligadura k-ésima una vez corregidas las posiciones; pero asimismo, rompe parcialmente la satisfacción de las ligaduras con índices menores a k. Por lo tanto, una vez resuelto el sistema $A\gamma^{(it)} = b$, se corrigen las posiciones y se comprueba si se cumplen las ligaduras dentro de una cierta tolerancia (en GROMACS, variable *shake_tol*). Si no es así, es necesario volver a iterar (resolver $A\gamma^{(it+1)} = b$, del primer proceso iterativo), inicializando los γ_k con los valores obtenidos en la iteración anterior.

La corrección de la posición $x_i(t+\Delta t)$ del átomo i en el instante $t+\Delta t$ se obtiene, a partir de la posición $x_i'(t+\Delta t)$ y de las aproximaciones a los multiplicadores de Lagrange γ_k, según la siguiente fórmula:

$$\begin{aligned} x_i(t+\Delta t) &= x_i'(t+\Delta t) - \frac{\Delta t^2}{m_i}\sum_{k=0}^{N_c-1}\gamma_k(t)\nabla_i\sigma_k(x_i(t)) \\ &= x_i'(t+\Delta t) - \frac{\Delta t^2}{m_i}\sum_{k=0}^{N_c-1}\gamma_k(t)2(x_l(t)-x_m(t))\big(\delta(i,l)-\delta(i,m)\big) \end{aligned} \tag{2.12}$$

El doble proceso iterativo hace que el algoritmo SHAKE sea fuente de inestabilidades numéricas y tenga problemas de convergencia. Estos problemas se agravan aún más si se imponen ligaduras en los ángulos entre enlaces. Además, por diseño SHAKE no puede ser paralelizado.

2.2.2. ILVES-S

ILVES-S propone sustituir el segundo proceso iterativo del algoritmo SHAKE por el uso de técnicas de resolución de matrices banda[1], puesto que la matriz A (2.10) es una matriz dispersa para las moléculas biológicas y puede convertirse en una matriz banda reordenando las ligaduras apropiadamente. Además, el formalismo presentado en [6] ofrece la posibilidad de imponer ligaduras en ángulos diedros, algo que no soporta el algoritmo SHAKE.

El proyecto fin de carrera de la autora de este trabajo consistió en implementar una versión preliminar del algoritmo ILVES-S para evaluar su potencial. Como hemos mencionado, el procedimiento es el siguiente: resolver el sistema $A\gamma^{(it)} = b$, ecuaciones (2.10) y (2.11), corregir las posiciones de los átomos (2.12) y comprobar si cumplen las ligaduras dentro de una cierta tolerancia. Si no es así, se vuelve a iterar (método de Newton).

De esta forma, la implementación del algoritmo consiste en dos fases bien diferenciadas. La primera fase es de inicialización y la segunda de resolución del sistema en cada paso temporal. La fase de inicialización es muy importante puesto que es necesario indexar apropiadamente las ligaduras para obtener después una matriz banda.

La fase de resolución del sistema se lleva a cabo en cada paso temporal. La Figura 2.3 muestra un esquema de esta fase. En el primer paso se calcula una aproximación de los multiplicadores de Lagrange. A diferencia de SHAKE, que los inicializa a cero, ILVES-S realiza una estimación basada en los multiplicadores obtenidos en los pasos temporales anteriores para que la convergencia sea más rápida. Después, se corrigen las posiciones de acuerdo a esta primera aproximación de los multiplicadores de Lagrange. A continuación comienza el proceso iterativo: se construye la matriz A y el vector b, se resuelve el sistema lineal y se corrigen las posiciones de los átomos. Si estas nuevas posiciones no cumplen con las ligaduras del sistema, se efectúa otra iteración.

En esta versión preliminar del algoritmo la resolución del sistema lineal se lleva a cabo mediante el uso de la librería externa LAPACK [9] que proporciona rutinas para

[1]Una matriz banda es una matriz dispersa que contiene sus valores no nulos en la diagonal principal y en cero o más diagonales a ambos lados, formando así una banda de valores no nulos.

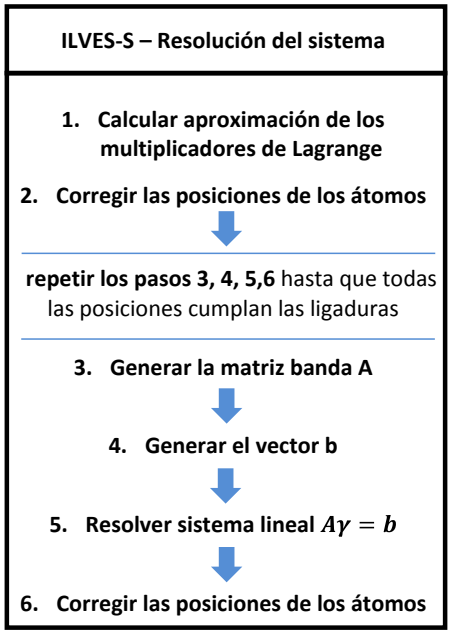

Figura 2.3: Esquema de la fase de resolución del sistema. Algoritmo ILVES-S.

resolver problemas de álgebra lineal tales como los sistemas de ecuaciones lineales. Pensamos que el uso de funciones específicas para resolver el sistema lineal mejoraría la eficiencia del algoritmo.

En el capitulo 4 caracterizamos el comportamiento de las implementaciones en GROMACS de los algoritmos SHAKE e ILVES-S. Pero antes, en el siguiente apartado, mostramos una optimización preliminar del algoritmo ILVES-S.

Optimización preliminar: Estimación inicial de los multiplicadores de Lagrange

Como hemos visto, el algoritmo ILVES-S resuelve el sistema de ecuaciones de 2.5 aplicando el método de Newton para encontrar los valores de los multiplicadores de Lagrange. La convergencia de éste método puede acelerarse si se usa un valor inicial apropiado. SHAKE no explota esta propiedad, es decir:

$$\gamma(t) = 0 \tag{2.13}$$

Con una estimación inicial sencilla y basada en los multiplicadores de Lagrange

de las dos iteraciones anteriores comprobamos que, efectivamente, la convergencia se acelera. La fórmula para obtener dicha estimación inicial es la siguiente:

$$\gamma(t) = 2 * \gamma(t-1) - \gamma(t-2) \tag{2.14}$$

La tabla de la Figura 2.4 recoge una media (en el número de pasos temporales) de los errores relativos medio (para todas las ligaduras) y máximo. El error relativo se define como:

$$error\_relativo = \frac{|(a_{i,j})^2 - (\mathbf{x}_i - \mathbf{x}_j)^2|}{(a_{i,j})^2} \tag{2.15}$$

Los errores de la tabla se muestran en función del número de iteraciones. Vemos cómo evoluciona la convergencia del algoritmo SHAKE y dos versiones distintas del algoritmo ILVES-S. La versión 0.3, al igual que SHAKE, no utiliza ninguna estimación inicial para los multiplicadores de Lagrange, es decir $\gamma(t) = 0$. Por el contrario, la versión 0.8 utiliza la estimación inicial de la ecuación 2.14. Se muestran valores para tres proteínas diferentes.

Los errores de las primeras filas (cuando Iter. vale - -) son los cometidos al comienzo del algoritmo para imponer ligaduras, es decir, los que se dan tras resolver la primera fase del algoritmo de Dinámica Molecular. Los errores de la iteración 0 son los cometidos después de aplicar la estimación inicial, antes de comenzar el proceso iterativo. Lógicamente, en el caso de SHAKE y de la versión 0.3 de ILVES-S los valores de estos errores son iguales, al usar como estimación inicial $\gamma(t) = 0$. Pero como podemos observar en la versión 0.8 de ILVES-S, el error relativo medio disminuye en una orden de magnitud. Además, si comparamos las dos versiones del algoritmo ILVES-S, tras realizar una iteración el error relativo medio es del orden de 10^{-05} sin estimación inicial y de 10^{-07} con estimación inicial.

En resumen, al utilizar una sencilla estimación inicial de los multiplicadores de Lagrange, la convergencia del algoritmo se acelera. Tenemos en cuenta que es posible realizar estimaciones iniciales más precisas, pero que a su vez, incrementarían la complejidad del algoritmo.

Molecula	SHAKE			ILVES-S v.0.3			ILVES-S v.0.8		
	Iter.	Error relativo medio	Error relativo máximo	Iter.	Error relativo medio	Error relativo máximo	Iter.	Error relativo medio	Error relativo máximo
1F2H	--	3,80E-03	6,07E-02	--	3,77E-03	5,97E-02	--	3,80E-03	6,02E-02
	0	3,80E-03	6,07E-02	0	3,77E-03	5,97E-02	0	4,57E-04	6,82E-03
	1	4,36E-04	2,87E-03	1	2,89E-05	1,63E-03	1	3,05E-07	2,54E-05
	5	1,63E-05	1,55E-04	2	7,03E-09	1,37E-06	2	6,48E-13	5,11E-10
	10	8,17E-07	1,10E-05	3	5,00E-15	1,07E-12	3	3,00E-15	1,60E-14
	15	4,46E-08	8,15E-07	4	2,00E-15	1,20E-14	4	2,00E-15	1,20E-14
	20	2,51E-09	5,93E-08	5	2,00E-15	1,10E-14	5	2,00E-15	1,10E-14
	25	1,44E-10	4,25E-09	6	2,00E-15	1,00E-14	6	2,00E-15	1,00E-14
	30	8,40E-12	3,02E-10						
1L2Y	--	3,83E-03	5,26E-02	--	3,81E-03	5,06E-02	--	3,79E-03	5,25E-02
	0	3,83E-03	5,26E-02	0	3,81E-03	5,06E-02	0	4,49E-04	4,32E-03
	1	4,46E-04	2,19E-03	1	3,16E-05	1,50E-03	1	2,99E-07	1,05E-05
	5	2,02E-05	1,33E-04	2	9,96E-09	1,22E-06	2	5,74E-13	9,37E-11
	10	1,19E-06	9,24E-06	3	6,00E-15	9,97E-13	3	2,00E-15	7,00E-15
	15	7,36E-08	6,77E-07	4	1,00E-15	5,00E-15	4	1,00E-15	5,00E-15
	20	4,64E-09	4,89E-08	5	1,00E-15	5,00E-15	5	1,00E-15	4,00E-15
	25	2,94E-10	3,44E-09	6	1,00E-15	5,00E-15	6	1,00E-15	4,00E-15
	30	1,86E-11	2,36E-10						
1LXL	--	3,64E-03	6,49E-02	--	3,63E-03	6,34E-02	--	3,63E-03	6,14E-02
	0	3,64E-03	6,49E-02	0	3,63E-03	6,34E-02	0	4,39E-04	6,98E-03
	1	4,41E-04	2,84E-03	1	2,75E-05	1,85E-03	1	2,85E-07	2,67E-05
	5	1,85E-05	1,57E-04	2	7,03E-09	1,77E-06	2	5,78E-13	5,61E-10
	10	1,02E-06	1,05E-05	3	6,00E-15	1,81E-12	3	4,00E-15	2,40E-14
	15	5,95E-08	7,38E-07	4	3,00E-15	1,70E-14	4	3,00E-15	1,60E-14
	20	3,55E-09	5,30E-08	5	3,00E-15	1,50E-14	5	3,00E-15	1,50E-14
	25	2,14E-10	3,80E-09	6	3,00E-15	1,40E-14	6	3,00E-15	1,40E-14
	30	1,30E-11	2,70E-10						

Figura 2.4: Errores relativos medio y máximo en función del número de iteraciones.

Capítulo 3

Metodología

Para analizar nuestro algoritmo y compararlo con el algoritmo original, realizamos una serie de experimentos en Gromacs consistentes en simular el movimiento de una proteína. Además de comprobar y comparar aspectos relacionados con el tiempo de ejecución o la precisión del algoritmo, utilizamos la herramienta VTune para caracterizar el comportamiento de esta aplicación.

Este capítulo describe la metodología utilizada para caracterizar los algoritmos ILVES-S y SHAKE. Primero, describimos el procesador y el entorno experimental. Después, se expone en qué consisten los análisis realizados y las métricas utilizadas y, por último, la carga de trabajo y el tipo de simulaciones que hemos considerado.

3.1. Procesador y entorno experimental

Hemos ejecutado una serie de simulaciones en un equipo con procesador Intel(R) Core(TM) i7-2600 (4 núcleos, 2 threads/núcleo, 3.40GHz, arquitectura Sandy Bridge), 8 GB de memoria RAM y sistema operativo CentOS versión 6.3.

La figura 3.1 muestra la jerarquía de memoria del procesador: 3 niveles de cache on-chip - L1I + L1D: 32 kB + 32 kB (por núcleo), L2: 256 kB (por núcleo), L3: 8 MB (compartida) - y la memoria principal de 8 GB. La cache de nivel 3, L3, se conoce también como LLC (*Last Level Cache*).

Por otra parte, la versión de Gromacs utilizada es la 4.6.5, compilada en doble precisión. La versión utilizada en el proyecto fin de carrera era una anterior (versión

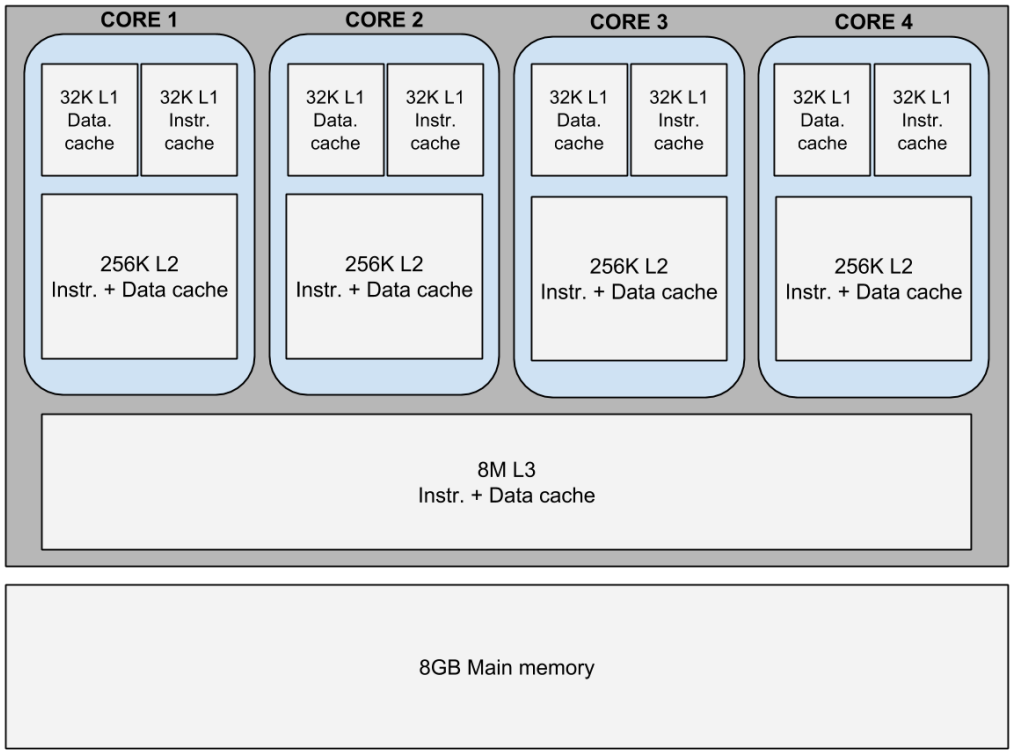

Figura 3.1: Jerarquía de Memoria.

4.5.5) por lo que fue necesario migrar la implementación del algoritmo ILVES-S. Además, decidimos usar el software de control de versiones git [8], por lo que configuramos el nuevo proyecto en este software.

3.2. Análisis de rendimiento

Para monitorizar el rendimiento de la CPU cuando se ejecutan los dos algoritmos ILVES-S y SHAKE, hemos utilizado la herramienta Intel(R) VTune(TM) Amplifier XE 2013. Podremos, de esta manera, identificar cualquier tipo de problema de rendimiento, (cuellos de botella, acceso a memoria, ...) así como las funciones más costosas.

VTune utiliza los contadores hardware del procesador y muestreo basado en eventos (*Event Based Sampling o EBS*). Esta técnica consiste en interrumpir periódicamente al procesador para obtener el contexto de ejecución. Los contadores hardware son unos registros especiales presentes en la arquitectura de los procesadores actua-

les diseñados expresamente para que el desarrollador pueda obtener información del contexto de ejecución de su aplicación. Estos contadores cuentan el número de veces que ocurre un evento durante la ejecución de la aplicación. Los eventos pueden referirse, por ejemplo, al número de instrucciones ejecutadas, a la cantidad de fallos de cache o al número de operaciones en coma flotante.

Dado que el objetivo es el análisis de una aplicación de ámbito HPC (*High Performance Computing*), es importante centrarse en la precisión del algoritmo y el tiempo de ejecución pero también comprobar el uso de las unidades funcionales de cálculo y el comportamiento de la memoria. Por ello, la caracterización se divide en cuatro fases:

- Precisión: El comportamiento general del algoritmo en cuanto a precisión y tiempo es el más interesante de cara al usuario final. Por ello, estudiamos la precisión del algoritmo propuesto y su convergencia con respecto al algoritmo original.

- Tiempo: Por la misma razón que el caso anterior, realizamos un análisis del tiempo total de ejecución de ambos algoritmos. También es importante comprobar el tiempo de ejecución de cada función, e incluso de cada instrucción de alto nivel, para identificar las partes críticas del código.

- Número de instrucciones y operaciones de cálculo: Según la propuesta teórica, el nuevo algoritmo ILVES-S realiza menos operaciones de coma flotante. Lo comprobamos de forma experimental analizando el número de éstas operaciones y el número de instrucciones ejecutadas.

- Memoria: Los requerimientos de memoria para este tipo de aplicaciones son un factor importante. Si los accesos a memoria no son fluidos esto puede retrasar la ejecución, dejando a la CPU detenida (*idle time*). Por ello, es importante verificar el comportamiento de la aplicación frete a la jerarquía de memoria, poniendo especial atención en los fallos de LLC cuya latencia es mucho mayor.

3.3. Métricas

En esta Sección analizamos las métricas usadas para cada fase de la caracterización.

3.3.1. Precisión

Para medir la precisión del algoritmo, es necesario considerar que el error cometido debe ser menor a una tolerancia dada (parámetro *shake_tol* en GROMACS). Teniendo en cuenta la ecuación que define una ligadura 2.3, se define el error relativo para toda ligadura $\sigma_{k(i,j)}$ en cualquier instante temporal como:

$$\frac{|(a_{i,j})^2 - (\mathbf{x}_i - \mathbf{x}_j)^2|}{(a_{i,j})^2} <= 2 * shake\_tol \tag{3.1}$$

Es decir, dada una ligadura de enlace entre los átomos i y j que impone una distancia $a_{i,j}$, la fórmula anterior expresa que el error relativo entre esa distancia y la real, $(\mathbf{x}_i - \mathbf{x}_j)^2$, debe ser menor o igual al doble de la tolerancia dada, *shake_tol*.

Al tratarse en ambos algoritmos de un proceso iterativo, nos interesa analizar el número de iteraciones con respecto a la tolerancia requerida, es decir, analizar la convergencia del método.

3.3.2. Tiempo

Considerando el tiempo de ejecución, nos centramos en dos métricas ambas en segundos (s) o milisegundos (ms), dependiendo de la duración del experimento:

- El tiempo de ejecución total de los algoritmos con respecto a la tolerancia requerida.
- El tiempo de ejecución de cada función para identificar cuales consumen la mayor parte del tiempo.

3.3.3. Número de instrucciones y operaciones de cálculo

Para analizar el rendimiento de la aplicación en cuanto a la capacidad de cálculo, vamos a analizar las siguientes métricas:

- CPU_CLK: Número de ciclos de reloj cuando el núcleo no está detenido.
- INST_RETIRED: Instrucciones retiradas de la fase de ejecución.
- CPI (Ciclos Por Instrucción): Es una métrica fundamental que indica aproximadamente cuantos ciclos de reloj le cuesta a cada instrucción ejecutarse. Los actuales procesadores superescalares ejecutan hasta 4 instrucciones por ciclo por lo que el CPI máximo teórico sería 0.25. Pero varios factores (latencia de memoria, operaciones en coma flotante, instrucciones vectoriales (SIMD) o instrucciones no retiradas debido a errores en la predicción de saltos) tienden a incrementar el CPI. Un CPI igual a 1 se considera generalmente aceptable para aplicaciones de alto rendimiento (HPC - High Performance Computing) pero depende mucho del dominio de la aplicación. El CPI es una excelente métrica para juzgar y ajustar el rendimiento en general de una aplicación.
- FP_SCALAR_DOUBLE Número de μops[1] en coma flotante, escalares en doble precisión (SIMD, Extensiones vectoriales SSE* o AVX-128).
- FP_OPS: Número de μops en coma flotante (FADD, FSUB, FCOM, FMULs, integer MULsand IMULs, FDIVs, FPREMs, FSQRTS, integer DIVs, y IDIV).

3.3.4. Jerarquía de memoria

Las métricas utilizadas para caracterizar el comportamiento de la aplicación frente a la jerarquía de memoria son las siguientes: LLC Miss (%), LLC Hit (%) e ICache Misses. A continuación describimos cada una de ellas.

LLC Miss (%): La LLC (Last Level Cache) tiene la latencia más alta puesto que es el último nivel de cache en la jerarquía de memoria, antes de la memoria principal. Cualquier petición de memoria que falla en este nivel tiene que ser servida por la memoria principal aumentando significativamente la latencia. Esta métrica muestra el porcentaje de ciclos debidos a los fallos de cache con respecto al total de ciclos consumidos:

$$LLCMiss(\%) = \frac{180 * LLC\_MISS}{CPU\_CLK} * 100 \tag{3.2}$$

[1] μops: micro-operaciones, también denominadas uops, son instrucciones detalladas de bajo nivel usadas para implementar instrucciones de código máquina más complejas [10]

LLC_MISS indica el número de fallos de LLC y 180 son los ciclos de penalización por fallo de LLC.

LLC Hit (%): Aunque los aciertos de LLC son servidos mucho más rápido que los accesos a memoria principal, todavía suponen una importante penalización en el rendimiento. Esta métrica mide esta penalización, incluyendo además la penalización por mantener la coherencia entre memoria compartida.

$$LLCHit(\%) = \frac{26 * LLC\_HIT + 43 * XSNP\_HIT + 60 * XSNP\_HITM}{CPU\_CLK} * 100 \tag{3.3}$$

donde:

- $26 * LLC\_HIT$: Ciclos de penalización por acierto en LLC (cuando el bloque no está compartido en otra cache).
- $43 * XSNP\_HIT$: Ciclos de penalización por acierto en LLC. En este caso el bloque está compartido en la cache de otro núcleo (el bloque lo sirve ésta cache no LLC).
- $60 * XSNP\_HITM$: Ciclos de penalización por acierto en LLC. En este caso el bloque está compartido en la cache de otro núcleo (el bloque debe ser invalidado en ésta cache y lo sirve LLC).

ICache Misses: Esta última métrica es un ratio entre el número de fallos en la cache de instrucciones y el número de instrucciones totales.

$$ICacheMisses = \frac{ICACHE.MISSES}{INST\_RETIRED} \tag{3.4}$$

3.4. Tipo de simulaciones y carga de trabajo

Consideramos los dos tipos de simulaciones más importantes e interesantes científicamente: el primero impone ligaduras a todos los enlaces atómicos y el segundo impone, además, ligaduras a los ángulos de enlaces en los que intervienen átomos de hidrógeno. Estos dos tipos de simulaciones se diferencian en Gromacs por un parámetro de entrada: *constraints*, cuyo valor es *all-bonds* para simulaciones con ligaduras en los enlaces y *h-angles* para simulaciones con ligaduras en enlaces y en

ángulos de hidrógeno. Lógicamente, las simulaciones con *constraints=h-angles* son más costosas que las simulaciones con *constraints=all-bonds* .

Las proteína usada en los experimentos es la siguiente:

1L2Y - NMR Structure of Trp-Cage Miniprotein Construct TC5b [11]

Las propiedades que nos interesan de ésta proteína son las siguientes:

- Número de átomos: 1045
- Número de ligaduras: 310 (all-bonds) y 376 (h-angles).

Los pasos temporales de las simulaciones son de 2 fs y el número de pasos temporales efectuados depende de la caracterización. Los resultados presentados en cuanto a tiempo total de ejecución corresponden a simulaciones con 8 *threads* para explotar la capacidad *multithreading* del procesador y 100,000 pasos temporales (tiempo simulado 200 ps). Cuando caracterizamos las aplicaciones en cuanto a precisión, tiempo de ejecución por función, operaciones de cálculo o jerarquía de memoria, realizamos las simulaciones con 1 thread y 1,000 pasos temporales (tiempo simulado 2 ps).

La Tabla 3.1 recoge los parámetros más relevantes de las simulaciones.

Tabla 3.1: Parámetros de simulación.

Molécula	1L2Y
Nº pasos temporales	1,000 - 100,000
Tamaño paso temporal	2 fs
Número de threads	1 - 8
Ligaduras	all-bonds - h-angles
shake_tol (tolerancia)	from 10^{-4} to 10^{-14}

Capítulo 4

Caracterización y análisis de los resultados

El objetivo de este capítulo es caracterizar los dos algoritmos para la imposición de ligaduras. La caracterización es una técnica basada en el muestreo temporal para identificar el comportamiento de código en ejecución. De esta manera podremos identificar las partes críticas de nuestro código así como posibles cuellos de botella, es decir, paradas en la unidad de ejecución, que pueden deberse a varios factores: acceso a memoria (fallos de cache, por ejemplo), errores en la predicción de saltos u operaciones en coma flotante, que tienen mayor latencia. Esta caracterización nos permite, además, comparar el algoritmo original, SHAKE, y nuestra implementación de la propuesta teórica, ILVES-S.

El resultado de esta caracterización se muestra para dos tipos de simulación de la misma proteína, 1L2Y. Las simulaciones se diferencian en la tolerancia requerida y el tipo de ligaduras que se imponen al sistema:

- Simulación 1: 1L2Y – all-bonds – shake_tol=10^{-04}
- Simulación 2: 1L2Y – h-angles – shake_tol=10^{-14}

Las simulaciones de tipo 1 son las más rápidas puesto que *constraints=all-bonds* y la tolerancia requerida es la menor, 10^{-04}. Al contrario, las simulaciones de tipo 2 son las más costosas ya que *constraints=h-angles* y la tolerancia es 10^{-14}.

4.1. Número de iteraciones y precisión

En este apartado presentamos los resultados en cuanto al número de iteraciones y la precisión del algoritmo ILVES-S, integrado en GROMACS [7], y los comparamos con los obtenidos por la implementación, en el mismo paquete software, de SHAKE.

Como hemos visto en la Sección 2.2, SHAKE e ILVES-S tienen un proceso iterativo común. La Figura 4.1 muestra el número medio[1] de iteraciones que requiere cada algoritmo para obtener una tolerancia dada (*shake_tol*). La precisión válida para llevar a cabo una simulación varía entre 10^{-4} y 10^{-14} (precisión máquina).

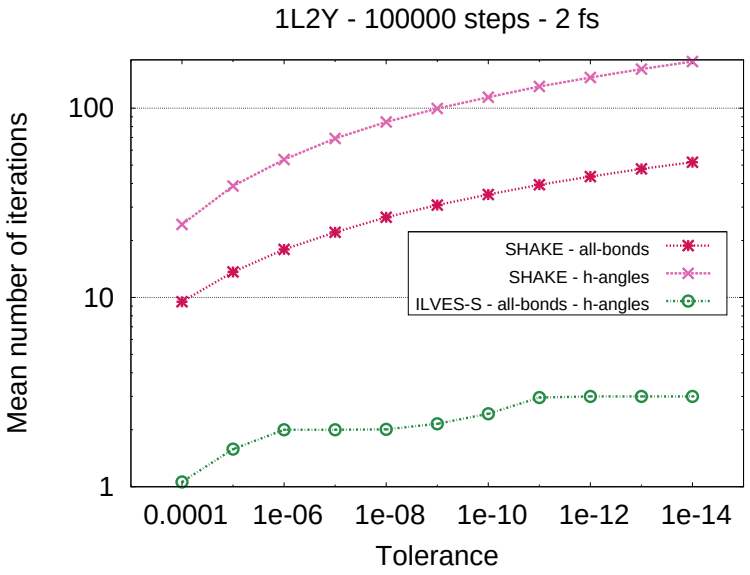

Figura 4.1: Número medio de iteraciones en función de la tolerancia

ILVES-S alcanza una tolerancia moderada de 10^{-4} con una media de poco más de una iteración. Por su parte, SHAKE alcanza la misma precisión con una media de poco menos de 10 iteraciones, cuando se imponen ligaduras en enlaces (*all-bonds*), y con casi 25 cuando se imponen ligaduras en enlaces y ángulos que involucran átomos de hidrógeno (*h-angles*).

Estos resultados ponen de manifiesto que, tal y como habíamos dicho, SHAKE converge muy lentamente, especialmente cuando se imponen ligaduras en ángulos.

[1] Promedio de iteraciones requeridas en los distintos pasos temporales.

ILVES-S sólo necesita 3 iteraciones para alcanzar precisión máquina, frente a las 51 (*all-bonds*) y 176 (*h-angles*) iteraciones que requiere SHAKE.

4.2. Tiempo

4.2.1. Tiempo total de ejecución

La ventaja de ILVES-S con respecto a SHAKE, si tenemos en cuenta el número de iteraciones necesarias por cada algoritmo, es muy significativa, pero lo realmente importante es considerar el tiempo de ejecución. En este caso observamos dos tendencias diferentes originadas por el tipo de ligaduras que se imponen al sistema.

Las Figuras 4.2 y 4.3 muestran el impacto de la tolerancia en el tiempo de ejecución debido al algoritmo de ligaduras, métrica proporcionada por uno de los ficheros de salida (*logfile*) de Gromacs. Cuando se imponen ligaduras sólo en los enlaces (Figura 4.2), el tiempo del algoritmo SHAKE es menor, a pesar de ejecutar muchas más iteraciones. Esto es debido a que una iteración de ILVES-S, con la implementación actual, tiene un coste temporal mucho mayor que una iteración de SHAKE. Sin embargo, pensamos que el tiempo de ILVES-S puede ser mucho menor usando funciones específicas para la resolución del sistema en lugar de las proporcionadas por LAPACK [9]. Además, ILVES-S puede ser paralelizado mientras que SHAKE no. Nuestro trabajo continúa en esta línea.

Cuando se imponen ligaduras en los enlaces y en los ángulos de enlaces con átomos de hidrógeno observamos en la Figura 4.3 que, a partir de una tolerancia 10^{-7}, el tiempo de ejecución de ILVES-S es menor al de SHAKE, es decir la convergencia de SHAKE es mucho más lenta que la de ILVES-S, otra vez motivada por la imposición de ligaduras en los ángulos.

Los números asociados a cada punto en las gráficas de las Figuras 4.2 y 4.3 representan el porcentaje de tiempo imputable al algoritmo de ligaduras respecto al tiempo total de ejecución de Gromacs. Cuanto mayor es este porcentaje, mayor es el impacto que tienen los algoritmos para imponer ligaduras en el tiempo total de la aplicación. Como podemos observar, este porcentaje es significativo y crece rápidamente al exigir mayor precisión.

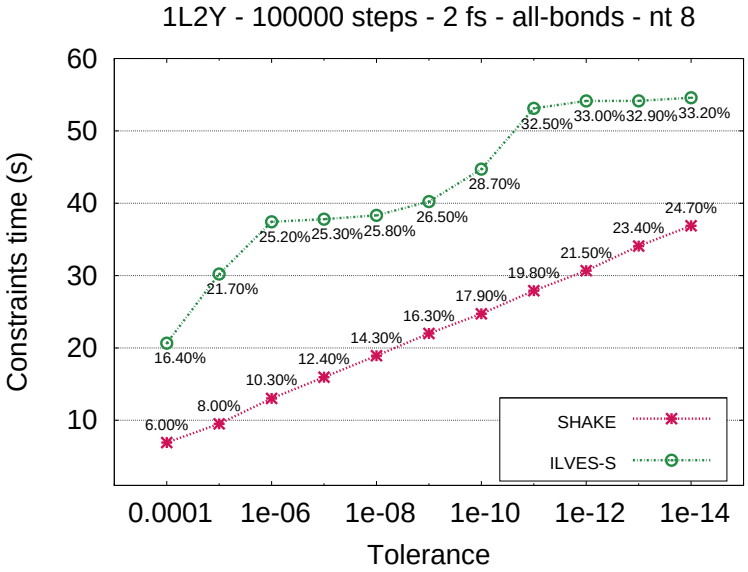

Figura 4.2: Tiempo de ejecución (*wall time*) del algoritmo para imponer ligaduras en función de la tolerancia. El porcentaje indica el tiempo sobre el total de simulación (*all-bonds*)

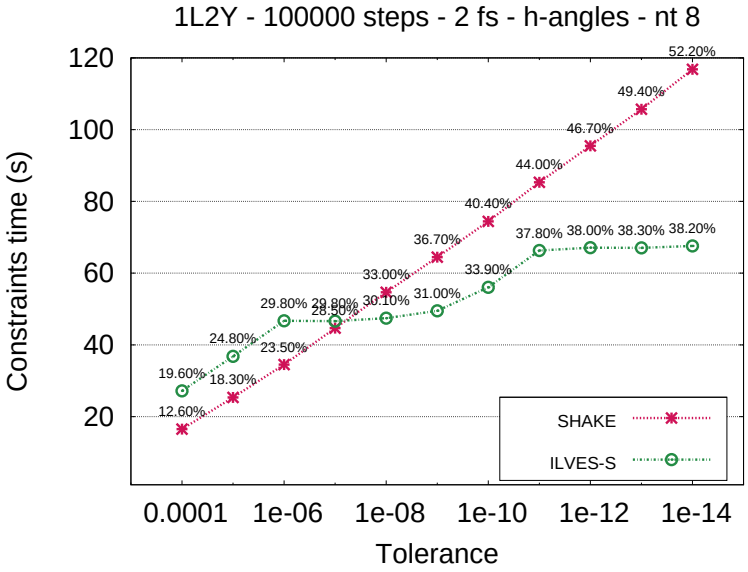

Figura 4.3: Tiempo de ejecución (*wall time*) del algoritmo para imponer ligaduras en función de la tolerancia. El porcentaje indica el tiempo sobre el total de simulación (*h-angles*)

4.2.2. Tiempo de ejecución por función

Tras conocer el comportamiento del algoritmo en cuanto a tiempo de ejecución y comprobar que sólo en algunos casos ILVES-S es más rápido que SHAKE, el objetivo ahora es identificar las partes críticas de nuestro código para tratar de optimizarlas. En las figura 4.4 y 4.5 se muestra el tiempo de ejecución de cada una de las funciones del algoritmo ILVES-S (parte izquierda) y de la función correspondiente al algoritmo SHAKE (parte derecha).

La figura 4.4 corresponde a los tiempos de ejecución de la simulación que impone ligaduras sólo en los enlaces ($shake\_tol = 10^{-04}$). Como ya sabíamos, en este caso, SHAKE es más rápido que ILVES-S. Identificamos las funciones críticas: la función que construye la matriz banda (*make_banded_matrix*) y la función que resuelve el sistema lineal (*solve_banded_system*) usando la librería externa LAPACK [9].

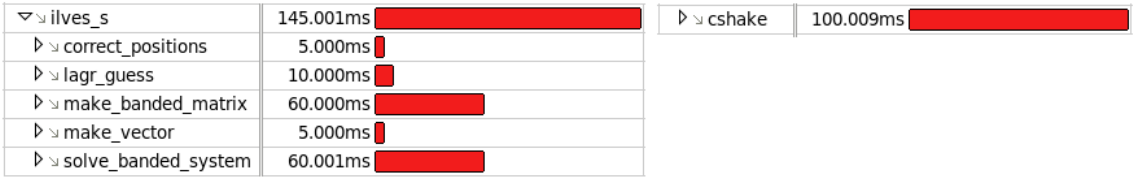

Figura 4.4: Tiempo de ejecución por función. 1L2Y – all-bonds – shake_tol=10^{-04}

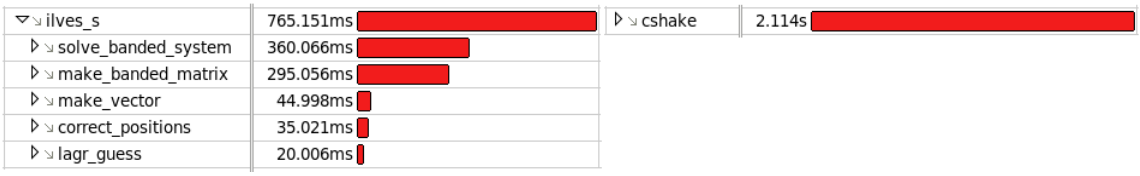

Figura 4.5: Tiempo de ejecución por función. 1L2Y – h-angles – shake_tol=10^{-14}

La figura 4.5 muestra los tiempos de ejecución para las simulaciones que impone ligaduras en los enlaces y en los ángulos de hidrógeno ($shake\_tol = 10^{-04}$). Observamos la misma tendencia que en el caso anterior en cuanto a funciones críticas y corroboramos que para este caso SHAKE es mucho más lento que ILVES-S.

4.3. Número de instrucciones y operaciones de cálculo

Las Tablas 4.1 y 4.2 muestran los datos correspondientes a las métricas CPU_CLK, INST_ RETIRED y CPI (sección 3.3.3 del Capítulo 3). El número de instrucciones ejecutadas y los ciclos de reloj consumidos están de acuerdo con los tiempos obtenidos, es decir, para la simulación con ligaduras en enlaces el número de ciclos y de instrucciones es mayor en ILVES-S y para la simulación con ligaduras en enlaces y ángulos de hidrógeno es mayor en SHAKE. Además el ratio CPI, es similar para los dos algoritmos, incluso ligeramente mejor para ILVES-S. En ambos casos es un buen ratio puesto que, como comentamos en la sección 3.3.3, el máximo teórico sería 0.25 (hasta 4 instrucciones por ciclo). Esto nos indica que estamos obteniendo un buen rendimiento para estas aplicaciones.

Tabla 4.1: Ciclos, instrucciones y CPI. 1L2Y – all-bonds – shake_tol=10^{-04}

Algorithm	**Source Function**	**CPU_CLK (x 10^3)**	**INST_ RETIRED (x 10^3)**	**CPI**
SHAKE	*cshake*	254,525	485,775	0.524
ILVES-S	*ilves-s*	5,525	9,900	0.558
	lagr_guess	31,100	78,225	0.398
	make_vector	42,175	80,675	0.523
	make_banded_matrix	201,900	373,650	0.540
	solve_banded_system	46,025	89,050	0.516
	correct_positions	28,225	78,800	0.358
	TOTAL	354,950	710,300	0.499

Teniendo en cuenta las operaciones de coma flotante, Tablas 4.3 y 4.4, es muy importante destacar que el número de operaciones en coma flotante es siempre mayor para el algoritmo SHAKE. Estos resultados son los esperados por la propuesta teórica del algoritmo ILVES-S [6]. La idea es obtener mejores resultados en un tiempo menor. Si la convergencia de ILVES-S es mucho más rápida (muchas menos iteraciones), nos planteamos por qué el tiempo de ejecución es mayor en la mayoría de simulaciones con ILVES-S si, además, el número de operaciones en coma flotante es siempre mayor

Tabla 4.2: Ciclos, instrucciones y CPI. 1L2Y – h-angles – shake_tol=10^{-14}

Algorithm	**Source Function**	**CPU_CLK** (**x** 10^3)	**INST_ RETIRED** (**x** 10^3)	**CPI**
SHAKE	*cshake*	5,648,300	11,225,500	0.503
ILVES-S	*ilves-s*	6,200	11,425	0.543
	lagr_guess	31,950	83,950	0.381
	make_vector	86,550	171,800	0.504
	make_banded_matrix	656,300	1,219,450	0.538
	solve_banded_system	140,450	286,625	0.490
	correct_positions	89,525	252,100	0.355
	TOTAL	1,010,975	2,025,350	0.499

para SHAKE. Para esclarecer esto, en el apartado siguiente continuamos con la caracterización de los algoritmos con respecto a la jerarquía de memoria.

Tabla 4.3: Operaciones FP. 1L2Y – all-bonds – shake_tol=10^{-04}

Algorithm	**Source Function**	**FP_SCALAR_DOUBLE** (**x** 10^3)	**FP_OPS** (**x** 10^3)
SHAKE	*cshake*	60,400	7,600
ILVES-S	*ilves-s*	0	0
	lagr_guess	4,400	0
	make_vector	8,800	1,200
	make_banded_matrix	18,800	0
	solve_banded_system	6,000	1,400
	correct_positions	7,000	0
	TOTAL	45,000	2,600

La Tabla 4.5 muestra, para cada simulación y para cada algoritmo, el porcentajes de operaciones en coma flotante con respecto al total de operaciones. De esta forma, resumimos el análisis realizado en este apartado: la cantidad de operaciones en coma flotante con respecto al total es menos relevante para el algoritmo ILVES-S que para SHAKE.

Tabla 4.4: Operaciones FP. 1L2Y – h-angles – shake_tol=10^{-14}

Algorithm	**Source Function**	**FP_SCALAR_DOUBLE (x 10^3)**	**FP_OPS (x 10^3)**
SHAKE	*cshake*	1,947,200	153,600
ILVES-S	*ilves-s*	0	0
	lagr_guess	7,200	0
	make_vector	23,600	4,200
	make_banded_matrix	72,600	0
	solve_banded_system	23,600	8,400
	correct_positions	22,600	0
	TOTAL	149,600	12,600

Tabla 4.5: % Operaciones FP con respecto al total.

Algorithm	**1L2Y all-bonds shake_tol=10^{-04}**	**1L2Y h-angles shake_tol=10^{-14}**
SHAKE	14.00 %	18.71 %
ILVES-S	6.70 %	8.01 %

4.4. Jerarquía de memoria

En esta sección caracterizamos los algoritmos SHAKE e ILVES-S con respecto a la jerarquía de memoria.

Las Tablas 4.6 y 4.7 muestran el porcentaje de ciclos debidos a fallo de LLC o acierto de LLC sobre el total de ciclos consumidos. También aparece el ratio de fallos en la cache de instrucciones con respecto al total de instrucciones (métricas descritas en la sección 3.3.4).

Como podemos observar, estos ratios son muy bajos por lo que consideramos que el impacto en el rendimiento de los algoritmos por penalizaciones en el acceso a memoria es mínimo. Aun así, los ratios para el algoritmo ILVES-S son ligeramente superiores que para SHAKE.

Tabla 4.6: Métricas jerarquía de memoria. 1L2Y – all-bonds – shake_tol=10^{-04}

Algorithm	Source Function	LLC Miss	LLC Hit	ICache Misses
SHAKE	*cshake*	0.4 %	0.1 %	0
ILVES-S	*ilves-s*	0	4.7 %	0.002
	lagr_guess	2.3 %	4.2 %	0
	make_vector	0.4 %	0.6 %	0
	make_banded_matrix	0.3 %	0.2 %	0
	solve_banded_system	0	0	0.267
	*dgbtrf_*	0.4 %	0	0.001
	*dgbtrs_*	0	0	0.005
	correct_positions	0	1.4 %	0.001

Tabla 4.7: Métricas jerarquía de memoria. 1L2Y – h-angles – shake_tol=10^{-14}

Algorithm	Source Function	LLC Miss	LLC Hit	ICache Misses
SHAKE	*cshake*	0.1 %	0	0
ILVES-S	*ilves-s*	0	2.1 %	0.002
	lagr_guess	3.9 %	3.3 %	0
	make_vector	0.2 %	0	0
	make_banded_matrix	0.3 %	0.2 %	0
	solve_banded_system	0	0	0
	*dgbtrf_*	0.1 %	0.1 %	0
	*dgbtrs_*	0	0	0.1
	correct_positions	0	0.4 %	0

4.5. Resultados

De los resultados obtenidos, podemos concluir que la limitación actual de nuestro algoritmo es la cantidad de operaciones de memoria. No encontramos cuellos de botella en el acceso a memoria porque esto lo veríamos reflejado en la métrica CPI de ILVES-S, que sería mucho mayor, y además, el % de ciclos de penalización por accesos a memoria con respecto al total es mínimo. El problema es que el número de accesos a memoria (lecturas y escrituras) es mucho mayor para el algoritmo ILVES-S, y así nos lo indica la métrica INST_RETIRED. Es decir, aunque el número de

operaciones en coma flotante siempre sea menor para ILVES-S, el número total de instrucciones es mayor para los casos en los que el tiempo de ejecución también es mayor.

4.6. Optimización basada en los resultados: Nueva representación matricial

Como hemos mencionado, uno de los problemas detectados al realizar la caracterización de los algoritmos es que ILVES-S realiza muchos más accesos a memoria, lecturas y escrituras, que SHAKE. Además, creemos que el uso de la librería externa LAPACK y sus funciones para resolver sistemas lineales banda no son las más eficientes para el problema que nos ocupa. Esto es así puesto que la matriz que describe estos sistemas es dispersa, casi banda pero con algunos elementos muy dispersos. El formato que utilizamos para guardar actualmente la matriz en memoria se ve perjudicado por estos elementos dispersos.

En la Figura 4.6 aparece el ejemplo de una pequeña molécula y la representación de su matriz dispersa cuando se incluyen ligaduras en los enlaces (39 enlaces, matriz cuadrada 39x39). Los elementos "x" son los elementos no nulos de la matriz, el resto son elementos nulos.

La versión actual de ILVES-S guarda la matriz en un formato especial para matrices banda usado por la librería LAPACK. Este formato reduce el número de elementos en memoria cuando se trata de matrices banda reales. Sin embargo, en nuestro caso incrementa todavía más el espacio que hay que reservar en memoria, espacio ocupado mayormente por elementos nulos. Este formato reserva memoria para, en el caso de matrices cuadradas simétricas $(3 * half\_bandwidth + 1)$ filas y N_c columnas, siendo $half\_bandwidth$ el semiancho de banda de la matriz[2] y N_c el número de ligaduras. En el ejemplo de la Figura 4.6 tenemos una matriz 39x39 cuyo semiancho de banda es 21. Si usáramos un formato que guardara todos los elementos necesitaríamos almacenar $39x39 = 1521$ elementos, pero con formato banda necesitamos $(3*21+1)*39 = 2496$ elementos en memoria. Si la matriz de la Figura 4.6 fuera

[2]Dada una matriz simétrica $A = (a_{ij})$, de dimensiones nxn, se define el semiancho de banda de A como $max|i-j|$, $a_{ij} = 0$

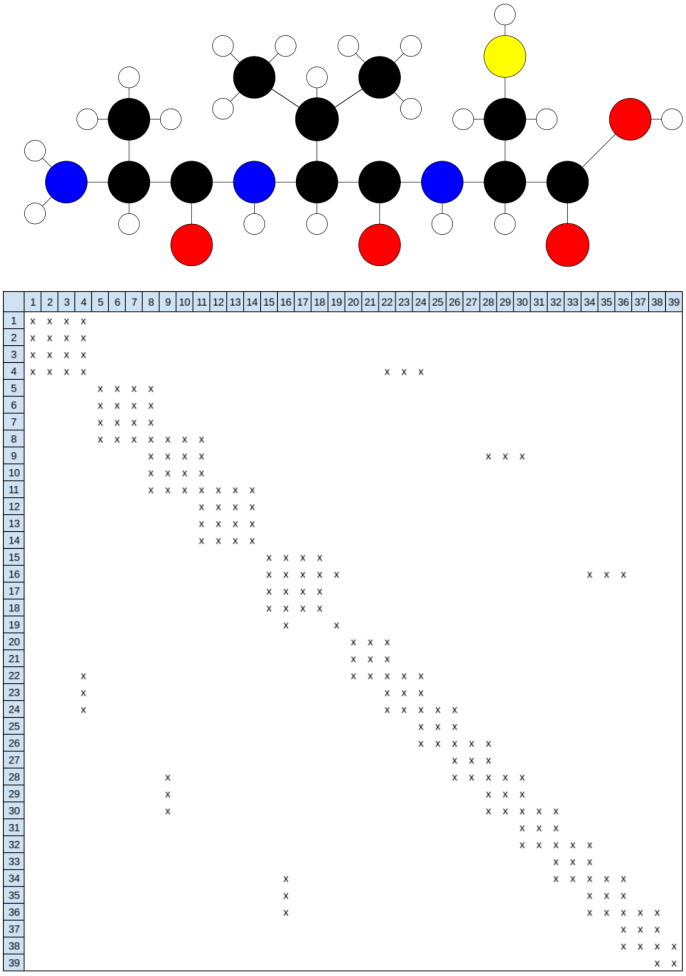

Figura 4.6: Ejemplo de molécula y la representación de su matriz dispersa.

realmente una matriz banda, es decir, si eliminamos los elementos dispersos el semi-ancho de banda sería 3 y por lo tanto, necesitaríamos almacenar $(3*3+1)*39 = 390$ elementos.

El nuevo formato utilizado para disminuir el espacio de memoria requerido y las lecturas y escrituras de elementos nulos, es el formato coordenada (Coordinate Format - COO). Este formato sólo almacena los elementos no nulos mediante el uso de tuplas (fila, columna, valor). Es decir, son necesarios tres vectores para almacenar las filas, las columnas y sus correspondientes valores no nulos.

La nueva librería que utilizaremos para comprobar los nuevos resultados con esta

optimización es HSL MA48 [12]. Son funciones de resolución de sistemas con matrices dispersas que utiliza la eliminación de Gauss-Jordan. De esta forma sustituiremos las dos funciones más críticas del algoritmo ILVES-S (sección 4.2.2): la que construye la matriz, *make_banded_matrix*, y la que resuelve el sistema, *solve_banded_system*.

Nuestra línea de trabajo más próxima se centra en esta nueva representación matricial y la implementación de las nuevas funciones usando la librería HSL MA48.

Capítulo 5

Conclusiones y trabajo futuro

La Dinámica Molecular es una técnica de uso extendido en numerosas áreas relacionadas con la física, la química y la biología donde es necesario comprender, estudiar y analizar el comportamiento de los átomos y las moléculas. Buscando mejorar la eficiencia de las simulaciones de Dinámica Molecular, se implementó una primera versión del algoritmo ILVES-S e integró en Gromacs.

En este trabajo fin de Máster hemos caracterizado este algoritmo y el original, SHAKE. Hemos analizado la precisión, el tiempo de ejecución y a más bajo nivel, las operaciones de cálculo y la jerarquía de memoria. Estas son las conclusiones que obtenemos de la caracterización realizada:

- El número medio de iteraciones que ILVES-S necesita para alcanzar la tolerancia deseada es considerablemente menor: tres para alcanzar precisión máquina frente a las decenas o incluso centenas de iteraciones que necesita SHAKE. Sin embargo, estas iteraciones todavía son muy costosas y por ello, el tiempo de ejecución de ILVES-S es mayor que el de SHAKE en cierto tipo de simulaciones.
- Hemos identificado las funciones críticas del algoritmo ILVES-S y propuesto alternativas para optimizarlas.
- Hemos comprobado que la cantidad de operaciones de coma flotante de ILVES-S, tal y como se esperaba en su propuesta teórica, es menor que la de SHAKE.
- Sin embargo, la proporción de instrucciones correspondientes a operaciones de acceso a memoria es mayor para ILVES-S, lo que identificamos como el princi-

pal problema de la implementación actual del algoritmo. La misma propuesta para sustituir las funciones críticas del algoritmo pretende disminuir también el número de operaciones y de accesos a memoria.

En conclusión, hemos caracterizado los algoritmos para imponer ligaduras en simulaciones de Dinámica Molecular para compararlos, identificar cuellos de botella y plantear optimizaciones del algoritmo ILVES-S. Hemos propuesto una primera posible optimización que disminuiría la cantidad de memoria necesaria y que reemplazaría las funciones críticas de la primera versión implementada.

5.1. Publicaciones

Las publicaciones relacionadas con este trabajo fin de Máster son las siguientes:

- Mª Astón Serrano-Gracia, Carl Christian Kjelgaard Mikkelsen, Jesús Alastruey-Benedé, Pablo Ibáñez-Marín y Pablo García-Risueño. *Implementation of a new constraint algorithm for Molecular Dynamics*. Publicación en actas y presentación de póster, X Escuela de Verano Internacional ACACES. Julio 2014.
- Mª Astón Serrano-Gracia, Carl Christian Kjelgaard Mikkelsen, Jesús Alastruey-Benedé, Pablo Ibáñez-Marín y Pablo García-Risueño. *Implementación de un nuevo algoritmo para imponer ligaduras en Dinámica Molecular*. Actas de las XXV Jornadas de Paralelismo. Septiembre 2014.

5.2. Trabajo futuro

Nuestro trabajo futuro se va a centrar en optimizar la versión actual de ILVES-S aplicando diversas técnicas.

La línea de trabajo más inmediata es la implementación de las funciones con la nueva representación matricial y la librería HSL MA48. De esta manera esperamos una importante mejora en el rendimiento de la versión actual de ILVES-S. Pero, por otra parte, pensamos que el rendimiento sería todavía mejor con el diseño de funciones de resolución de sistemas específicas para este problema. Por lo tanto, esperamos diseñar estas funciones y posteriormente, integrarlas en GROMACS.

Otra línea de trabajo es la paralelización del algoritmo ILVES-S. Puesto que el diseño del algoritmo SHAKE hace inviable su paralelización, confiamos en superar ampliamente su eficiencia.

Actualmente, tanto en SHAKE como en ILVES-S, la definición de ligaduras en ángulos de enlace es mediante dos ligaduras en enlaces imaginarios y además, no se contempla la imposición de ligaduras en ángulos dihedros, en los que se involucran 4 átomos. Imponer este tipo de ligaduras supone congelar los grados de libertad más rápidos y por tanto permitiría aumentar mucho más el tamaño del paso temporal de las simulaciones. Adaptar el algoritmo ILVES-S para incluir restricciones en estos grados de libertar es uno de los objetivos principales del proyecto y queda pendiente como trabajo futuro.

Bibliografía

[1] Benedict Leimkuhler and Sebastian Reich. Simulating hamiltonian dynamics. 2004.

[2] Christian Gossens, Ivano Tavernelli, and Ursula Rothlisberger. DNA structural distortions induced by Ruthenium- Arene anticancer compounds. *Journal of the American Chemical Society*, 130(33):10921–10928, 2008.

[3] Gregor Hlawacek, Peter Puschnig, Paul Frank, Adolf Winkler, Claudia Ambrosch-Draxl, and Christian Teichert. Characterization of step-edge barriers in organic thin-film growth. *Science*, 321(5885):108–111, 2008.

[4] Alexey K Mazur. Hierarchy of fast motions in protein dynamics. *The Journal of Physical Chemistry B*, 102(2):473–479, 1998.

[5] Jean-Paul Ryckaert, Giovanni Ciccotti, and Herman J.C Berendsen. Numerical integration of the cartesian equations of motion of a system with constraints: molecular dynamics of n-alkanes. *Journal of Computational Physics*, 23(2):327 – 341, 1977.

[6] Pablo García-Risueño, Pablo Echenique, and José Luis Alonso. Exact and efficient calculation of lagrange multipliers in biological polymers with constrained bond lengths and bond angles: Proteins and nucleic acids as example cases. *Journal of computational chemistry*, 32(14):3039–3046, 2011.

[7] David Van Der Spoel, Erik Lindahl, Berk Hess, Gerrit Groenhof, Alan E Mark, and Herman JC Berendsen. GROMACS: fast, flexible, and free. *Journal of computational chemistry*, 26(16):1701–1718, 2005.

[8] Scott Chacon and Junio C Hamano. *Pro git*, volume 288. Springer, 2009.

[9] Edward Anderson, Zhaojun Bai, Christian Bischof, Susan Blackford, James Demmel, Jack Dongarra, Jeremy Du Croz, Anne Greenbaum, S Hammerling, Alan McKenney, et al. LAPACK Users'guide. 1999.

[10] Agner Fog. The microarchitecture of Intel, AMD and VIA CPUs: An optimization gguide for assembly programmers and compiler makers. Technical report, Technical University of Denmark, 2014.

[11] Jonathan W Neidigh, R Matthew Fesinmeyer, and Niels H Andersen. Designing a 20-residue protein. *Nature Structural & Molecular Biology*, 9(6):425–430, 2002.

[12] Iain S Duff. Sparse system solution and the hsl library. *Some topics in industrial and applied mathematics*, 8:78–94, 2006.

Apéndice A

Implementación de un nuevo algoritmo para imponer ligaduras en Dinámica Molecular

Este apéndice incluye el artículo *Implementación de un nuevo algoritmo para imponer ligaduras en Dinámica Molecular* de Mª Astón Serrano-Gracia, Carl Christian Kjelgaard Mikkelsen, Jesús Alastruey-Benedé, Pablo Ibáñez-Marín y Pablo García-Risueño.

El artículo ha sido admitido en las XXV Jornadas de Paralelismo, organizadas por la Sociedad de Arquitectura y Tecnología de Computadores (SARTECO), y será presentado en Valladolid en Septiembre de 2014.

Implementación de un nuevo algoritmo para imponer ligaduras en Dinámica Molecular

Mª Astón Serrano-Gracia[1], Carl Christian Kjelgaard Mikkelsen[2], Jesús Alastruey-Benedé[1], Pablo Ibáñez-Marín[1] y Pablo García-Risueño[3] [4]

Resumen— **La Dinámica Molecular permite describir la evolución en el tiempo de sistemas de partículas mediante simulaciones del movimiento de los átomos y moléculas. Una opción común para realizar simulaciones más eficientes es imponer restricciones, denominadas ligaduras, en la longitud de los enlaces atómicos o en los ángulos entre enlaces. Esta técnica permite incrementar el paso temporal de las simulaciones y con ello lograr mayores tiempos simulados, con lo que se incrementa el poder predictivo de la simulación. Uno de los algoritmos más extendidos para imponer ligaduras es SHAKE, pero presenta limitaciones puesto que está basado en aproximaciones y procesos iterativos, lo cual afecta negativamente a su exactitud, eficiencia y estabilidad numérica. Presentamos la implementación en Gromacs de un nuevo algoritmo, ILVES-S, que, todavía en fase experimental, pretende mejorar la eficiencia y limitaciones del clásico SHAKE.**

Palabras clave— **Dinámica Molecular, GROMACS, SHAKE, ligaduras, Física Computacional, Química Computacional.**

I. Introducción

La Dinámica Molecular es una técnica de simulación por computador en la que átomos y moléculas interactúan durante un período de tiempo, dando lugar a una descripción detallada del movimiento de las partículas [1]. Estas simulaciones se basan en aproximaciones de la física conocida y se utilizan con frecuencia en el estudio de muchas moléculas biológicas como proteínas, péptidos, lípidos y ácidos nucleicos.

Llevar a cabo ciertos experimentos en un laboratorio conlleva un alto coste de recursos y no siempre es posible analizar las propiedades en todas las escalas temporales. Por lo tanto, las herramientas computacionales que realizan dichas simulaciones son fundamentales para los investigadores de campos tan diversos como la medicina (búsqueda de tratamientos para enfermedades como el Alzheimer, la fibrosis quística o el cáncer; diseño computacional de medicamentos [2]), la química (diseño de catalizadores) o la ingeniería de materiales [3].

Debido a la complejidad de los sistemas biológicos, sus simulaciones requieren una gran cantidad de recursos, tanto de memoria como de cálculo. La demanda de experimentos más precisos y con sistemas cada vez más complejos impulsa la búsqueda constante de mejoras en los algoritmos de Dinámica Molecular y en sus implementaciones.

Una técnica muy común para mejorar la eficiencia de la experimentación es aumentar el paso temporal, *time step*, de las simulaciones cuyos valores típicos son del orden de un femtosegundo (10^{-15} s) [4]. Para ello es necesario usar algoritmos, como SHAKE [5] o LINCS [6], que permiten fijar las vibraciones de los átomos más rápidos mediante la imposición de ligaduras. Una ligadura es una restricción en la longitud de los enlaces atómicos o en el valor de los ángulos entre enlaces. Estas restricciones no alteran el valor de las propiedades que se desean observar y permiten realizar simulaciones más eficientes.

Como consecuencia de la imposición de ligaduras aparecen, sobre el sistema de partículas, las denominadas fuerzas de ligadura, que en la práctica se traduce en nuevas ecuaciones. Para un sistema con N_c ligaduras, es necesario resolver un conjunto de N_c ecuaciones no lineales. SHAKE las resuelve mediante un proceso iterativo que normalmente converge. Sin embargo, esta convergencia puede llegar a ser muy lenta con ciertas topologías, especialmente, si se imponen ligaduras en ángulos entre enlaces atómicos.

ILVES-S es un nuevo algoritmo basado en un método alternativo para resolver las ecuaciones algebraicas originadas por la imposición de ligaduras [7]. Este método aprovecha que la estructura de los polímeros biológicos típicos, como los ácidos nucleicos y las proteínas, es esencialmente lineal para aplicar técnicas de resolución de matrices banda. De este modo, pretendemos mejorar la estabilidad y convergencia de SHAKE, así como su tiempo de ejecución. Además, y al contrario que SHAKE, es posible paralelizar el algoritmo ILVES-S para alcanzar un mayor rendimiento.

En este trabajo abordamos la implementación de ILVES-S en Gromacs, un versátil, rápido y extendido paquete software de Dinámica Molecular [8]. Para evaluar la implementación de este nuevo algoritmo, realizamos simulaciones de una proteína sintética, 1L2Y [9]. Los resultados muestran que ILVES-S necesita un máximo de tres iteraciones para alcanzar precisión máquina, mientras que SHAKE necesita más de 50 o 170 según el tipo de simulación que realicemos. Sin embargo, esta primera implementación de ILVES-S no siempre es más rápida que SHAKE. Esto se debe a varios factores, como por ejemplo el uso de costosas librerías externas. Nuestro trabajo continúa con el diseño de funciones es-

[1]gaZ. Dpto. de Informática e Ing. de Sistemas. Inst. de Investigación en Ing. de Aragón. Universidad de Zaragoza, e-mail: {maston,jalastru,imarin}@unizar.es.

[2]Department of Computing Science and HPC2N, Umeå University, Sweden, e-mail: spock@cs.umu.se.

[3]Instituto de Biocomputación y Física de Sistemas Complejos (BIFI), Universidad de Zaragoza

[4]Institut für Physik, Humboldt Universität zu Berlin, Germany, e-mail: risueno@physik.hu-berlin.de

pecíficas para nuestro problema.

La organización del resto del artículo es la siguiente: en la Sección II realizamos una breve introducción al algoritmo de Dinámica Molecular para después centrarnos, en la Sección III, en los algoritmos para la imposición de ligaduras, tanto en el algoritmo original SHAKE como en la nueva propuesta ILVES-S. La Sección IV describe la metodología usada en los experimentos cuyos resultados se presentan en la Sección V. Por último, en la Sección VI resumimos las conclusiones y el trabajo futuro de este proyecto.

II. Dinámica Molecular en Gromacs

El objetivo de las simulaciones de Dinámica Molecular es describir la evolución en el tiempo de un sistema de partículas. Para obtener la trayectoria del sistema se resuelve, en sucesivos instantes temporales, la ecuación clásica del movimiento según la segunda ley de Newton:

$$m_i \frac{d^2\mathbf{x}_i(t)}{dt^2} = \mathbf{F}_i(t) \qquad (1)$$

donde $\mathbf{x}_i(t)$ es el vector posición (x_x, x_y, x_z) de la partícula i en el instante temporal t, $\mathbf{F}_i(t)$ es el vector de la fuerza que actúa sobre la partícula i-ésima en el instante temporal t debida a la interacción con el resto de partículas del sistema, y m_i es la masa de la partícula i.

Incluir ligaduras al sistema es una técnica muy extendida para poder aumentar significativamente el paso temporal de la simulación y, por tanto, el tiempo simulado del experimento. Para ello, se añaden las denominadas fuerzas de ligadura a la ecuación clásica del movimiento de Newton (1). Asumiendo un sistema de N partículas que debe cumplir con N_c ligaduras, éstas se expresan como:

$$\sigma_k(\mathbf{x}_0 ... \mathbf{x}_{N-1}) = 0; \qquad k = 0...N_c - 1 \qquad (2)$$

Por ejemplo, la Figura 1 muestra la ligadura de un enlace k entre dos átomos, i y j, definida como:

$$\sigma_{k(i,j)} := (\mathbf{x}_i - \mathbf{x}_j)^2 - (a_{i,j})^2 = 0 \qquad (3)$$

donde $\mathbf{x}_i$ e $\mathbf{x}_j$ son los vectores posición de los átomos i y j. Esta ligadura expresa que la distancia entre estos átomos debe ser igual a $a_{i,j}$.

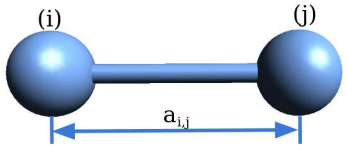

Fig. 1. Ligadura en un enlace.

Al integrar las fuerzas de ligadura en la ecuación (1), el nuevo sistema de ecuaciones es el siguiente:

$$m_i \frac{d^2\mathbf{x}_i(t)}{dt^2} = \mathbf{F}_i(t) - \sum_{k=0}^{N_c-1} \lambda_k \frac{\partial \sigma_k}{\partial \mathbf{x}_i}(t) \qquad (4)$$

donde $-\sum_{k=0}^{N_c-1} \lambda_k \frac{\partial \sigma_k}{\partial \mathbf{x}_i}(t)$ es la fuerza sobre el átomo i en el instante t que aparece debida a las ligaduras y los distintos λ_k, $k = 0...N_c-1$ son los multiplicadores de Lagrange.

La imposición de las N_c ligaduras convierte un sistema de $3N$ ecuaciones diferenciales con $3N$ incógnitas en un sistema de $3N + N_c$ ecuaciones diferenciales con $3N + N_c$ incógnitas.

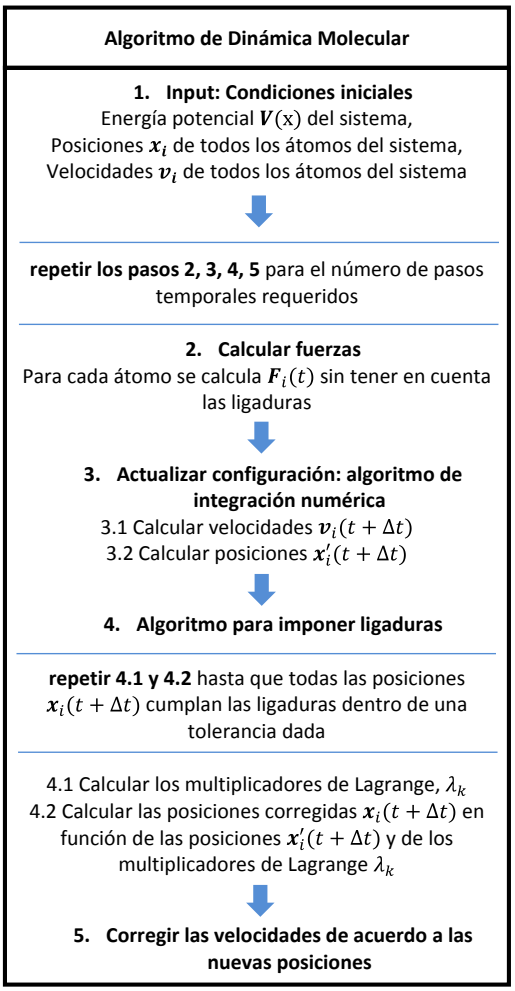

Fig. 2. Esquema del algoritmo de Dinámica Molecular.

Gromacs resuelve estas ecuaciones en dos fases: en primer lugar se calculan las nuevas posiciones de los átomos sin tener en cuenta las ligaduras impuestas al sistema y, en segundo lugar, se calculan las fuerzas de ligadura y se corrigen las posiciones. De la segunda fase se encargan los algoritmos para la imposición de ligaduras como SHAKE o nuestra propuesta ILVES-S.

Una vez resuelta, para cada átomo, la ecuación clásica del movimiento de Newton (1), es necesario resolver las ecuaciones para obtener las fuerzas de ligadura cuyas incógnitas son los multiplicadores de Lagrange.

La Figura 2 muestra de forma esquemática la implementación en Gromacs del algoritmo de Dinámica Molecular. Nuestro trabajo se centra en la implementación de un nuevo algoritmo para im-

$$-2\Delta t^2(x'_i(t+\Delta t)-x'_j(t+\Delta t))\left(\sum_{k'=0}^{N_c-1}\gamma_{k'}(\frac{\nabla_i\sigma_{k'}}{m_i}(t)-\frac{\nabla_j\sigma_{k'}}{m_j}(t))\right)$$
$$+\Delta t^4\sum_{k'=0}^{N_c-1}\sum_{k''=0}^{N_c-1}\gamma_{k'}\gamma_{k''}\left(\frac{\nabla_i\sigma_{k'}}{m_i}(t)-\frac{\nabla_j\sigma_{k'}}{m_j}(t)\right)\left(\frac{\nabla_i\sigma_{k''}}{m_i}(t)-\frac{\nabla_j\sigma_{k''}}{m_j}(t)\right)$$
$$=(a_{i,j})^2-(x'_i(t+\Delta t)-x'_j(t+\Delta t))^2 \qquad \text{para } k=0...N_c-1. \tag{5}$$

poner ligaduras, es decir, nos centraremos, a partir de ahora, en el punto 4 de la Figura 2: el cálculo de las fuerzas de ligadura para corregir las posiciones y velocidades de los átomos.

III. Algoritmos para la imposición de ligaduras

En esta Sección presentamos el algoritmo original, SHAKE, y el algoritmo alternativo, ILVES-S.

A. SHAKE

El algoritmo SHAKE calcula, en cada instante temporal $(t+\Delta t)$, un conjunto de coordenadas $x_i(t+\Delta t)$ que cumplen con la lista de ligaduras, a partir de las coordenadas $x_i(t)$ y de las coordenadas obtenidas sin considerar las fuerzas de ligadura $x'_i(t+\Delta t)$ (coordenadas a corregir).

Para calcular los multiplicadores de Lagrange (y con ello, las fuerzas de ligadura (4)) hay que resolver el sistema de N_c ecuaciones cuadráticas (5), con N_c el número de ligaduras. En realidad, SHAKE obtiene una aproximación, γ_k, de los multiplicadores de Lagrange, λ_k, resolviendo un sistema de N_c ecuaciones cuadráticas en γ_k que tiene la forma:

$$\begin{cases} a_{0,0}\gamma_0+a_{0,1}\gamma_1+...+(\gamma_0\gamma_0c_{0,0}+\gamma_0\gamma_1c_{0,1}+...)=b_0 \\ a_{1,0}\gamma_0+a_{1,1}\gamma_1+...+(\gamma_0\gamma_0c_{1,0}+\gamma_0\gamma_1c_{1,1}+...)=b_1 \\ a_{2,0}\gamma_0+a_{2,1}\gamma_1+...+(\gamma_0\gamma_0c_{2,0}+\gamma_0\gamma_1c_{2,1}+...)=b_2 \\ ... \end{cases} \tag{6}$$

Excepto las γ_k, todos los términos de estas ecuaciones son conocidos: k es la ligadura que involucra a los átomos i y j y (equivalentemente a $\nabla_j\sigma_{k'}$, $\nabla_i\sigma_{k''}$, $\nabla_j\sigma_{k''}$):

$$\nabla_i\sigma_{k'(l,m)}=2(\mathbf{x}_l-\mathbf{x}_m)(\delta_{i,l}-\delta_{i,m}) \tag{7}$$

donde $\delta_{i,l}$ y $\delta_{i,m}$ representan la delta de Kronecker, función matemática de dos variables que vale 1 si son iguales y 0 si son diferentes:

$$\delta_{\alpha,\beta}=\begin{cases}1 & \text{si } \alpha=\beta,\\ 0 & \text{si } \alpha\neq\beta.\end{cases}$$

Como hemos dicho, tenemos un sistema de N_c ecuaciones cuadráticas en γ_k (6) que se puede resolver de forma iterativa (método de Newton, primer proceso iterativo). La implementación en Gromacs del algoritmo SHAKE considera que los términos en γ_k^2 (es decir el término en Δt^4) son despreciables puesto que Δt, el tamaño del paso temporal, es del orden de los femtosegundos. En cada una de las iteraciones, it, del algoritmo se resuelve el sistema lineal:

$$A\gamma^{(it)}=b \tag{8}$$

donde, según el sistema (5), despreciando el término en Δt^4, y teniendo en cuenta la ecuación (7):

$$A_{kk'}:=-2\Delta t^2\Big(x'_i(t+\Delta t)-x'_j(t+\Delta t)\Big)$$
$$2(x_l(t)-x_m(t))$$
$$\Big(\frac{\delta(i,l)}{m_i}-\frac{\delta(i,m)}{m_i}-\frac{\delta(j,l)}{m_j}+\frac{\delta(j,m)}{m_j}\Big), \tag{9}$$
$$b_k:=(a_{i,j})^2-(x'_i(t+\Delta t)-x'_j(t+\Delta t))^2, \tag{10}$$

b es un vector de dimensión N_c y A es una matriz cuadrada en el número de ligaduras N_c. Un elemento $A_{k,k'}$ es distinto de cero si las ligaduras, $k(i,j)$ y $k'(l,m)$ comparten algún átomo, esto es, $i=l$ o $i=m$ o $j=l$ o $j=m$, e igual a cero si todos los átomos involucrados en las ligaduras $k(i,j)$ y $k'(l,m)$ son distintos.

La resolución en cada iteración it del sistema $A\gamma^{(it)}=b$ se lleva a cabo de forma aproximada, de nuevo mediante un proceso iterativo (segundo proceso iterativo, N_c iteraciones), puesto que se resuelve en primer lugar la primera ecuación, después la segunda y así sucesivamente. Se obtiene primero $\gamma_0^{(0)}$, considerando que $\gamma_1^{(0)}...\gamma_{N_c-1}^{(0)}$ son iguales a cero en la primera iteración. En las sucesivas iteraciones se usan los γ_k obtenidos en la iteración anterior. Es decir, poniendo como ejemplo el sistema (6), en la primera iteración (del primer proceso iterativo) se resuelve la primera ecuación de (6), forzando $\gamma_1=0,\gamma_2=0...$, para así obtener γ_0, después se resuelve la segunda ecuación de (6) usando el γ_0 obtenido y haciendo $\gamma_2=0,\gamma_3=0...$, y así sucesivamente.

De esta manera, la resolución de la ecuación k-ésima garantiza la satisfacción de la ligadura k-ésima una vez corregidas las posiciones; pero asimismo, rompe parcialmente la satisfacción de las ligaduras con índices menores a k. Por lo tanto, una vez resuelto el sistema $A\gamma^{(it)}=b$, se corrigen las posiciones y se comprueba si se cumplen las ligaduras dentro de una cierta tolerancia (en Gromacs, variable *shake_tol*). Si no es así, es necesario volver a iterar (resolver $A\gamma^{(it+1)}=b$, del primer proceso iterativo), inicializando los γ_k con los valores obtenidos en la iteración anterior.

La corrección de la posición $x_i(t+\Delta t)$ del átomo i en el instante $t+\Delta t$ se obtiene, a partir de la posición $x'_i(t+\Delta t)$ y de las aproximaciones a los multiplicadores de Lagrange γ_k, según la siguiente fórmula:

$$\begin{aligned} x_i(t+\Delta t) &= x'_i(t+\Delta t) - \frac{\Delta t^2}{m_i} \sum_{k=0}^{N_c-1} \gamma_k(t) \nabla_i \sigma_k(x_i(t)) \\ &= x'_i(t+\Delta t) \\ &\quad -\frac{\Delta t^2}{m_i} \sum_{k=0}^{N_c-1} \gamma_k(t) 2(x_l(t) - x_m(t)) \\ &\quad \big(\delta(i,l) - \delta(i,m)\big) \end{aligned} \tag{11}$$

El doble proceso iterativo hace que el algoritmo SHAKE sea fuente de inestabilidades numéricas y tenga problemas de convergencia. Estos problemas se agravan aún más si se imponen ligaduras en los ángulos entre enlaces. Además, por diseño SHAKE no puede ser paralelizado.

B. ILVES-S

ILVES-S propone sustituir el segundo proceso iterativo del algoritmo SHAKE por el uso de técnicas de resolución de matrices banda[1], puesto que la matriz A (9) es una matriz dispersa para las moléculas biológicas y puede convertirse en una matriz banda reordenando las ligaduras apropiadamente. Además, el formalismo presentado en [7] ofrece la posibilidad de imponer ligaduras en ángulos diedros, algo que no soporta el algoritmo SHAKE.

Hemos implementado una versión preliminar del algoritmo ILVES-S para evaluar su potencial. Como hemos mencionado, el procedimiento es el siguiente: resolver el sistema $A\gamma^{(it)} = b$, ecuaciones (9) y (10), corregir las posiciones de los átomos (11) y comprobar si cumplen las ligaduras dentro de una cierta tolerancia. Si no es así, se vuelve a iterar (método de Newton).

De esta forma, la implementación del algoritmo consiste en dos fases bien diferenciadas. La primera fase es de inicialización y la segunda de resolución del sistema en cada paso temporal. La fase de inicialización es muy importante puesto que es necesario indexar apropiadamente las ligaduras para obtener después una matriz banda.

La fase de resolución del sistema se lleva a cabo en cada paso temporal. La Figura 3 muestra un esquema de esta fase. En el primer paso se calcula una aproximación de los multiplicadores de Lagrange. A diferencia de SHAKE, que los inicializa a cero, ILVES-S realiza una estimación basada en los multiplicadores obtenidos en los pasos temporales anteriores para que la convergencia sea más rápida. Después, se corrigen las posiciones de acuerdo a esta primera aproximación de los multiplicadores de Lagrange. A continuación comienza el proceso iterativo: se construye la matriz A y el vector b, se resuelve el sistema lineal y se corrigen las posiciones de los átomos. Si estas nuevas posiciones no cumplen con las ligaduras del sistema, se efectúa otra iteración.

[1]Una matriz banda es una matriz dispersa que contiene sus valores no nulos en la diagonal principal y en cero o más diagonales a ambos lados, formando así una banda de valores no nulos.

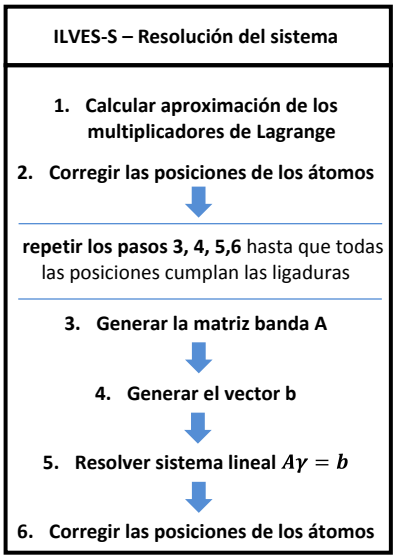

Fig. 3. Esquema de la fase de resolución del sistema. Algoritmo ILVES-S.

En esta versión preliminar del algoritmo la resolución del sistema lineal se lleva a cabo mediante el uso de la librería externa LAPACK [10] que proporciona rutinas para resolver problemas de álgebra lineal tales como los sistemas de ecuaciones lineales. Pensamos que el uso de funciones específicas para resolver el sistema lineal mejoraría la eficiencia del algoritmo. Por ello, estamos trabajando en el diseño e implementación de estas funciones.

La sección de resultados compara el rendimiento de las implementaciones en GROMACS de los algoritmos SHAKE e ILVES-S.

IV. METODOLOGÍA

A. Procesador y entorno experimental

Hemos ejecutado una serie de simulaciones en un equipo con procesador Intel(R) Core(TM) i7-2600 (4 núcleos, 2 threads/núcleo, 3.40GHz), 8 GB de memoria RAM y sistema operativo CentOS versión 6.3. La versión de GROMACS utilizada es la 4.6.5, compilada en doble precisión.

B. Tipo de simulaciones y carga de trabajo

Consideramos dos tipos de simulaciones: la primera impone ligaduras a todos los enlaces atómicos y la segunda impone, además, ligaduras a los ángulos de enlaces en los que intervienen átomos de hidrógeno. Estos dos tipos de simulaciones se diferencian en GROMACS por un parámetro de entrada: *constraints*, cuyo valor es *all-bonds* para simulaciones con ligaduras en los enlaces y *h-angles* para simulaciones con ligaduras en enlaces y en ángulos de hidrógeno.

La proteína usada en los experimentos que presentamos, cuya entrada PDB es 1L2Y [9], tiene 20 residuos[2], 304 átomos, se disuelve en agua. El número de ligaduras es de 310 con *constraints=all-bonds* y de 376 con *constraints=h-angles*. En todas las simulaciones se efectúan 100000 pasos temporales de 2 fs, por lo que la simulación es de 200 ps.

Para medir la precisión del algoritmo, es necesario considerar que el error cometido debe ser menor a una tolerancia dada (parámetro *shake_tol* en GROMACS). Se define el error para toda ligadura $\sigma_{k(i,j)}$ en cualquier instante temporal como:

$$\frac{|(a_{i,j})^2 - (\mathbf{x}_i - \mathbf{x}_j)^2|}{(a_{i,j})^2} <= 2 * shake\_tol \qquad (12)$$

Es decir, dada una ligadura de enlace entre los átomos i y j que impone una distancia $a_{i,j}$, la fórmula anterior expresa que el error relativo entre esa distancia y la real, $(\mathbf{x}_i - \mathbf{x}_j)^2$, debe ser menor al doble de la tolerancia dada, *shake_tol*.

La Tabla I recoge los parámetros más relevantes de las simulaciones.

TABLA I
PARÁMETROS DE SIMULACIÓN.

Molécula	1L2Y (304 átomos)
Nº pasos temporales	100.000
Tamaño paso temporal	2 fs
Número de threads	8
Ligaduras	all-bonds (310 lig.) h-angles (376 lig.)
***shake_tol* (tolerancia)**	from 10^{-4} to 10^{-14}

V. RESULTADOS

En este apartado presentamos los resultados de la primera implementación del algoritmo ILVES-S, integrada en GROMACS [8], y los comparamos con los obtenidos por la implementación en el mismo paquete software de SHAKE.

A. Número de iteraciones

Como hemos visto en la Sección III, SHAKE e ILVES-S tienen un proceso iterativo común. La Figura 4 muestra el número medio[3] de iteraciones que requiere cada algoritmo para obtener una tolerancia dada (*shake_tol*). La precisión válida para llevar a cabo una simulación varía entre 10^{-4} y 10^{-14} (precisión máquina).

ILVES-S alcanza una tolerancia moderada de 10^{-4} con una media de poco más de una iteración. Por su parte, SHAKE alcanza la misma precisión con una media de poco menos de 10 iteraciones, cuando se imponen ligaduras en enlaces (*all-bonds*), y con casi 25 cuando se imponen ligaduras en enlaces y ángulos que involucran átomos de hidrógeno (*h-angles*).

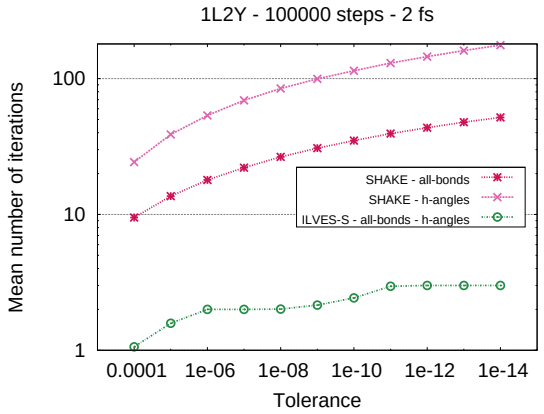

Fig. 4. Número medio de iteraciones en función de la tolerancia

Estos resultados ponen de manifiesto que, tal y como habíamos dicho, SHAKE converge muy lentamente, especialmente cuando se imponen ligaduras en ángulos. ILVES-S sólo necesita 3 iteraciones para alcanzar precisión máquina, frente a las 51 (*all-bonds*) y 176 (*h-angles*) iteraciones que requiere SHAKE.

B. Tiempo

La ventaja de ILVES-S con respecto a SHAKE, si tenemos en cuenta el número de iteraciones necesarias por cada algoritmo, es muy significativa, pero lo realmente importante es considerar el tiempo de ejecución. En este caso observamos dos tendencias diferentes originadas por el tipo de ligaduras que se imponen al sistema.

Las Figuras 5 y 6 muestran el impacto de la tolerancia en el tiempo de ejecución debido al algoritmo de ligaduras, métrica proporcionada por uno de los ficheros de salida (*logfile*) de GROMACS. Cuando se imponen ligaduras sólo en los enlaces (Figura 5), el tiempo del algoritmo SHAKE es menor, a pesar de ejecutar muchas más iteraciones. Esto es debido a que una iteración de ILVES-S, con la implementación actual, tiene un coste temporal mucho mayor que una iteración de SHAKE. Sin embargo, pensamos que el tiempo de ILVES-S puede ser mucho menor usando funciones específicas para la resolución del sistema en lugar de las proporcionadas por LAPACK [10]. Además, ILVES-S puede ser paralelizado mientras que SHAKE no. Nuestro trabajo continúa en esta línea.

Cuando se imponen ligaduras en los enlaces y en los ángulos de enlaces con átomos de hidrógeno observamos en la Figura 6 que, a partir de una tolerancia 10^{-7}, el tiempo de ejecución de ILVES-S es menor al de SHAKE, es decir la convergencia de SHAKE es mucho más lenta que la de ILVES-S, otra vez motivada por la imposición de ligaduras en los ángulos.

Los números asociados a cada punto en las gráficas de las Figuras 5 y 6 representan el porcentaje de

[2]Se denomina residuo a cada unidad sencilla dentro de un polímero, por ejemplo, cada uno de los aminoácidos integrados en la cadena polipeptídica de una proteína.

[3]Promedio de iteraciones requeridas en los distintos pasos temporales.

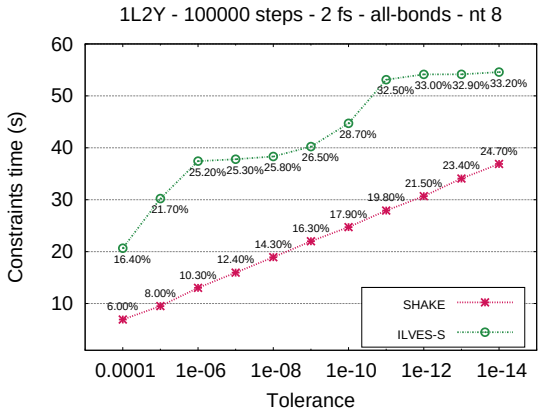

Fig. 5. Tiempo de ejecución (*wall time*) del algoritmo para imponer ligaduras en función de la tolerancia. El porcentaje indica el tiempo sobre el total de simulación (*all-bonds*)

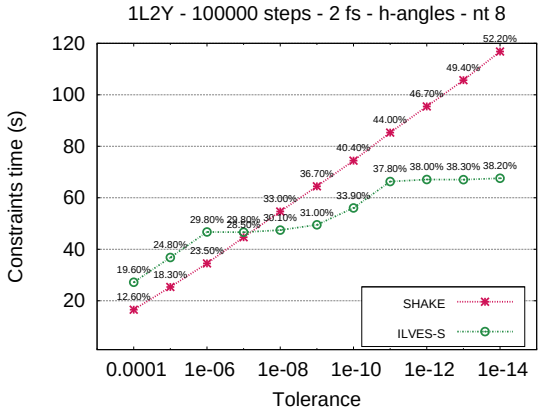

Fig. 6. Tiempo de ejecución (*wall time*) del algoritmo para imponer ligaduras en función de la tolerancia. El porcentaje indica el tiempo sobre el total de simulación (*h-angles*)

tiempo imputable al algoritmo de ligaduras respecto al tiempo total de ejecución de Gromacs. Cuanto mayor es este porcentaje, mayor es el impacto que tienen los algoritmos para imponer ligaduras en el tiempo total de la aplicación. Como podemos observar, este porcentaje es significativo y crece rápidamente al exigir mayor precisión.

La Figura 7 muestra el tiempo de ejecución de la simulación completa de Dinámica Molecular en Gromacs usando los dos algoritmos para imponer ligaduras (SHAKE e ILVES-S). Para ambos tipos de simulaciones (*all-bonds* y *h-angles*) la tendencia del tiempo total de ejecución cuando varía la tolerancia es similar a la tendencia del tiempo de SHAKE o ILVES-S.

VI. Conclusiones y trabajo futuro

La Dinámica Molecular es una técnica muy extendida en numerosas áreas relacionadas con la física, la química y la biología, donde es necesario comprender, estudiar y analizar el comportamiento de los átomos y las moléculas. Buscando mejorar la eficiencia de las simulaciones de Dinámica Molecular, hemos implementado una primera versión del algoritmo ILVES-S y la hemos integrado en Gromacs. Hemos observado que el número medio de iteraciones que ILVES-S necesita para alcanzar la tolerancia deseada es considerablemente menor: tres para alcanzar precisión máquina frente a las decenas o incluso centenas de iteraciones que necesita SHAKE. Sin embargo, estas iteraciones todavía son muy costosas.

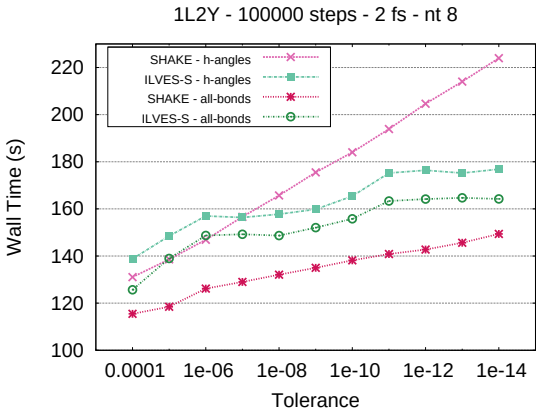

Fig. 7. Tiempo de ejecución del algoritmo de Dinámica Molecular

Nuestro trabajo futuro se va a centrar en optimizar la versión actual de ILVES-S aplicando diversas técnicas. Una de ellas es el diseño de funciones de resolución de sistemas banda específicas para este problema, en lugar de usar costosas funciones externas. Otra línea de trabajo es la paralelización del algoritmo ILVES-S. Además, puesto que el diseño del algoritmo SHAKE hace inviable su paralelización, confiamos en superar ampliamente su eficiencia.

Agradecimientos

El presente trabajo ha sido financiado mediante los proyectos TIN2010-21291-C02-01 (Gobierno de España y FEDER), Consolider CSD2007-00050 (Gobierno de España), gaZ: grupo de investigación T48 (Gobierno de Aragón y Fondo Social Europeo), HiPEAC-3 NoE (European FET FP7/ICT 287759), VR A0581501 (Swedish Research Council) y eSSENCE, a strategic collaborative eScience programme.

Referencias

[1] B. Leimkuhler y S. Reich, *Simulating hamiltonian dynamics*, Cambridge University Press, 2004.

[2] C. Gossens; I. Tavernelli y U.Rothlisberger, *DNA structural distortions induced by Ruthenium?Arene anticancer compounds*, Journal of the American Chemical Society, 2008.

[3] G. Hlawacek; P. Puschnig; P. Frank; A. Winkler; C. Ambrosch-Draxl y C. Teichert, *Characterization of step-edge barriers in organic thin-film growth*, Science, 2008.

[4] A. K. Mazur, *Hierarchy of fast motions in protein dynamics*, The Journal of Physical Chemistry B, 1998.

[5] J. P. Ryckaert; G. Ciccotti y H. J. C. Berendsen, *Numerical integration of the cartesian equations of motion of a system with constraints: molecular dynamics of n-alkanes*, Journal of Computational Physics, 1977.

[6] B. Hess; H. Bekker; H. J. Berendsen y J. G. Fraaije, *LINCS: a linear constraint solver for molecular simulations*, Journal of computational chemistry, 1997.

[7] P. García-Risueño; P. Echenique y J. L. Alonso, *Exact and efficient calculation of lagrange multipliers in biological polymers with constrained bond lengths and bond angles: Proteins and nucleic acids as example cases.*, Journal of computational chemistry, 2011.

[8] D. Van Der Spoel; E. Lindahl; B. Hess; G. Groenhof; A. E. Mark; y H. J. Berendsen, *GROMACS: fast, flexible, and free*, Journal of Computational Chemistry, 2005.

[9] J. W. Neidigh; R. M. Fesinmeyer y N. H. Andersen, *Designing a 20-residue protein*, Nature Structural Biology, 2002.

[10] E. Anderson; Z. Bai; C. Bischof; S. Blackford; J. Demmel; J. Dongarra; J. Du Croz; A. Greenbaum; S. Hammerling; A. McKenney y D. Sorensen, *LAPACK Users' guide*, Siam, 1999.

Apéndice B

Implementation of a new constraint algorithm for Molecular Dynamics

Este apéndice incluye el artículo *Implementation of a new constraint algorithm for Molecular Dynamics* de Mª Astón Serrano-Gracia, Carl Christian Kjelgaard Mikkelsen, Jesús Alastruey-Benedé, Pablo Ibáñez-Marín y Pablo García-Risueño.

El artículo ha sido admitido en la X Escuela de Verano Internacional ACACES (*Tenth International Summer School on Advanced Computer Architecture and Compilation for High-Performance and Embedded Systems*), organizada por HiPEAC (*European Network of Excellence on High Performance and Embedded Architecture and Compilation*). La escuela de verano consiste en una semana de cursos impartidos por importantes científicos de empresas y universidades de todo el mundo. Además, el artículo se publica en las actas de la escuela de verano y se presenta en forma de póster. La escuela de verano tendrá lugar en Fiuggi (Italia) en Julio de 2014 y la autora de este trabajo fin de máster es beneficiaria de una beca que cubre la cuota de inscripción.

Implementation of a new constraint algorithm for Molecular Dynamics

Mª Astón Serrano-Gracia[*,1],
Carl Christian Kjelgaard Mikkelsen[†,2],
Jesús Alastruey-Benedé[*,1],
Pablo Ibáñez-Marín[*,1],
Pablo García-Risueño[‡,§,3]

[*] *gaZ–DIIS–I3A, Universidad de Zaragoza, 50018 Zaragoza, Spain*
[†] *Dept. of Computing Science–HPC2N, Umeå University, SE-90187 Umeå,Sweden*
[‡] *BIFI, Universidad de Zaragoza 50018 Zaragoza, Spain*
[§] *Institut für Physik, Humboldt Universität zu Berlin, 12489 Berlin, Germany*

ABSTRACT

A widely used method in Molecular Dynamics is to constrain bonds, bond angles and dihedral angles. This allows to increase the time step in order to enable more efficient simulations. SHAKE is a very popular constraint algorithm but it is based on approximations and iterative procedures which negatively affects its precision, efficiency and numerical stability. We present a preliminary implementation in GROMACS of a new algorithm: ILVES-S. Our aim is to improve the efficiency and limitations of SHAKE.

KEYWORDS: Molecular Dynamics, SHAKE, GROMACS, constraints.

1 Introduction

Molecular Dynamics (MD) is a very popular computational tool for describing the trajectory of molecules. It is essential for researchers in different fields, such as materials development or drug design. The time step used to integrate the equations of motion in the simulations can be increased using constraints algorithms such as SHAKE [RCB77]. These algorithms impose constraints on the fastest internal degrees of freedom, i.e. bonds, bond angles and dihedral angles, allowing faster simulations without compromising the results.

The imposition of constraints makes new equations to appear, the *equations of constraint*. For a system with N_c constraints, a set of N_c non-linear equations has to be solved. SHAKE solves these equations using an iterative procedure which usually converges. However, this procedure converges slowly specially if bond angles are constrained.

ILVES-S is a new algorithm based on an alternative method to solve the *equations of constraint* [GREA11]. This method leverages the lineal structure of the typical biological polymers, such as nucleic acids and proteins, to use banded techniques. Thus, we hope to accel-

[1]E-mail: {maston,jalastru,imarin}@unizar.es
[2]E-mail: spock@cs.umu.se
[3]E-mail: risueno@physik.hu-berlin.de

erate the convergence and improve the numerical stability of SHAKE as well as its efficiency. Besides, contrary to SHAKE, ILVES-S can be parallelized to increase performance.

In this work we present the preliminary implementation of ILVES-S in GROMACS, a versatile, fast and widespread software package to perform Molecular Dynamics.

2 MD simulations with constraints

A MD simulation solves, in each time step, the Newton's second law of motion to get the atom positions and velocities. The existence of N_c constraints makes new forces to appear (the *constraint forces*) in the set of Newton equations, that is:

$$m_i \frac{d^2\mathbf{x}_i(t)}{dt^2} = \mathbf{F}_i(t) - \sum_{k=0}^{N_c-1} \lambda_k \frac{\partial \sigma_k}{\partial \mathbf{x}_i}(t) \tag{1}$$

where m_i is the mass of the atom i, $\mathbf{x}_i(t)$ is the position vector (x_x, x_y, x_z) of the atom i at the time step t, $\mathbf{F}_i(t)$ represents the force acting on the atom i at t due to the interaction with the rest of the atoms and $-\sum_{k=0}^{N_c-1} \lambda_k \frac{\partial \sigma_k}{\partial \mathbf{x}_i}(t)$ is the constraint force acting on the atom i at t. λ_k, $k = 0...N_c - 1$ are the Lagrange multipliers associated with the constraints. For example, a bond length constraint k is defined as:

$$\sigma_{k(i,j)} := (\mathbf{x}_i - \mathbf{x}_j)^2 - (a_{i,j})^2 = 0 \tag{2}$$

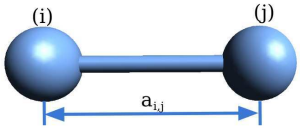

Figure 1: Bond constraint.

where $\mathbf{x}_i$ and $\mathbf{x}_j$ are the position vector of atoms i and j. As Figure 1 shows, this constraint sets the distance between i and j to $a_{i,j}$.

GROMACS solves equation 1 in two phases: first of all, new positions are calculated without considering the constraint forces, and secondly, constraint forces are calculated and positions are corrected. Constraints algorithms are in charge of solving the second phase, i.e. they solve the constraints equations where the Lagrange multipliers are the unknowns.

3 SHAKE

SHAKE calculates an approximation, γ_k, $k = 0...N_c - 1$, of the Lagrange multipliers by solving equations:

$$\begin{aligned} &- 2\Delta t^2 (x'_i(t+\Delta t) - x'_j(t+\Delta t)) \left(\sum_{k'=0}^{N_c-1} \gamma_{k'} \left(\frac{\nabla_i \sigma_{k'}}{m_i}(t) - \frac{\nabla_j \sigma_{k'}}{m_j}(t) \right) \right) \\ &+ \Delta t^4 \sum_{k'=0}^{N_c-1} \sum_{k''=0}^{N_c-1} \gamma_{k'} \gamma_{k''} \left(\frac{\nabla_i \sigma_{k'}}{m_i}(t) - \frac{\nabla_j \sigma_{k'}}{m_j}(t) \right) \left(\frac{\nabla_i \sigma_{k''}}{m_i}(t) - \frac{\nabla_j \sigma_{k''}}{m_j}(t) \right) \\ &= (a_{i,j})^2 - (x'_i(t+\Delta t) - x'_j(t+\Delta t))^2 \qquad \text{for } k = 0...N_c - 1. \end{aligned} \tag{3}$$

This is a system of N_c equations quadratic in γ_k. All terms, except γ_k, are known: k is the constraint which involves atoms i and j and $\nabla_i \sigma_{k'(l,m)} = 2(\mathbf{x}_l - \mathbf{x}_m)(\delta_{i,l} - \delta_{i,m})$, where $\delta_{i,l}$ and $\delta_{i,m}$ represent the Kronecker delta [1].

[1] Function of two variables that returns 1 if the variables are equal, and 0 otherwise.

GROMACS implementation of SHAKE considers γ_k^2 terms insignificant. Thus, system (3) can be solved in an iterative way (Newton's method - first iterative procedure). The linearized system $A\gamma^{(it)} = b$ is solved in each iteration, it, where:

$$A_{kk'} := -4\Delta t^2\Big(x_i'(t+\Delta t) - x_j'(t+\Delta t)\Big)(x_l(t) - x_m(t))\Big(\frac{\delta(i,l)}{m_i} - \frac{\delta(i,m)}{m_i} - \frac{\delta(j,l)}{m_j} + \frac{\delta(j,m)}{m_j}\Big), \quad (4)$$

$$b_k := (a_{i,j})^2 - (x_i'(t+\Delta t) - x_j'(t+\Delta t))^2 \quad (5)$$

The resolution of every linearized system is carried out by a further iterative process (second iterative procedure), first solving the first equation of the system, $A_{0k'}\gamma_k = b_0$, then the second one, $A_{1k'}\gamma_k = b_1$, and so on. This way to proceed guarantees the constraint k is satisfied but then, previous constraints ($k-1$, $k-2$, etc.) can be violated.

Once the Lagrange multipliers are calculated, the atom positions are corrected as follows:

$$x_i(t+\Delta t) = x_i'(t+\Delta t) - \frac{\Delta t^2}{m_i}\sum_{k=0}^{N_c-1}\gamma_k(t)\nabla_i\sigma_k(x_i(t)) \quad (6)$$

The second iterative procedure could be source of instabilities and may have convergence problems, especially with bond angles. Moreover, SHAKE can not be parallelized.

4 ILVES-S

The aim of ILVES-S is to replace the second iterative procedure of SHAKE by analytical banded techniques that solve the system $A\gamma^{(it)} = b$. This can be done because matrix A is sparse for biological molecules and can be defined as a banded matrix if the constraints are re-indexed properly.

We have implemented a preliminary version of ILVES-S in GROMACS, which consists on an initialization phase and a resolution phase. In the first phase, the re-indexation of the constraints is carried out. In the second phase, which is executed at each time step, the banded system $A\gamma^{(it)} = b$ is solved and the atom positions corrected, until the constraints are satisfied. Before first iteration, we calculate a guess of the Lagrange multipliers, which is based on the ones obtained in the previous time steps, in order to speed up the convergence.

In the current ILVES-S version, we solve the linearized system using an external library, LAPACK [ABB+99], but we think that using specific functions for our problem would improve the algorithm efficiency.

5 Results

In this section, we compare the performance of the SHAKE and ILVES-S implementations in GROMACS. We simulate a synthetic protein, 1L2Y, which has 304 atoms. The simulation impose 310 constraints if bonds are constrained (all-bonds), or 376 constraints if bonds and angles that involve hydrogen atom are constrained (h-angles).

SHAKE and ILVES-S have to iterate, solving the system $A\gamma_k = b$, while the constraints are not satisfied under a given tolerance. Figure 2 shows the number of iterations of this iterative procedure as a function of the tolerance. ILVES-S needs only 3 iterations to reach

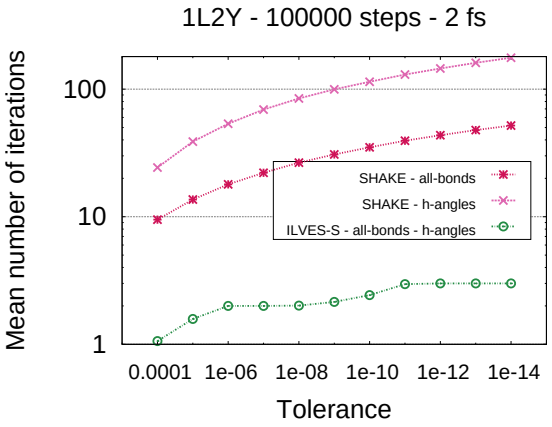

Figure 2: Mean number of iterations as a function of the tolerance

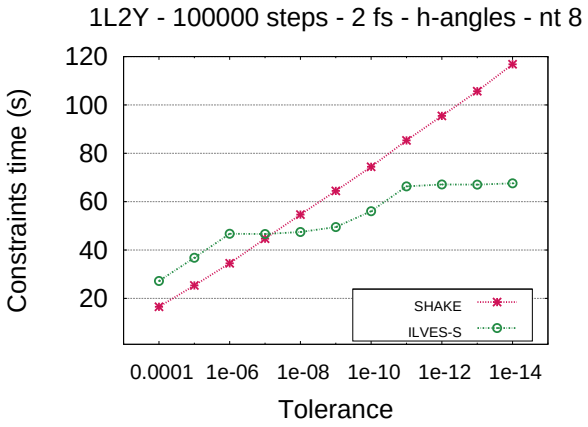

Figure 3: Constraints wall time as a function of the tolerance.

machine precision while SHAKE needs 51 (all-bonds) and 176 iterations (for h-angles). This highlights what we had said: SHAKE converges slowly, especially when bond angles are constrained. But more important than the number of iterations is the execution time. Figure 3 shows the constraint algorithms execution time as a function of the tolerance. From a required tolerance of 10^{-7}, ILVES-S is faster than SHAKE for a h-angles simulation. For an all-bonds simulation, SHAKE is still faster than ILVES-S.

6 Conclusions and future work

Molecular Dynamics is a widely used tool in many areas related to physics, chemistry and biology where the molecules and atoms behavior need to be analyzed. Our aim is to improve the simulations performance by implementing a new constraint algorithm, which represents up to 50% of the total simulation time. The first implemented version, which has been integrated in GROMACS, needs a considerably smaller number of iterations than the original algorithm, SHAKE. However, this iterations are time-consuming. Our future work focuses on optimizing the current ILVES-S version by (*i*) designing a specific solver which would replace LAPACK functions and (*ii*) parallelizing the algorithm.

Acknowledgments. This work was supported in part by grants TIN2010-21291-C02-01 (Spanish Government, FEDER), Consolider CSD2007-00050 (Spanish Government), gaZ: T48 research group (Aragón Government, European ESF), HiPEAC-3 NoE, VR A0581501 (Swedish Research Council) and eSSENCE, a strategic collaborative eScience programme.

References

[ABB+99] Edward Anderson, Zhaojun Bai, Christian Bischof, Susan Blackford, James Demmel, Jack Dongarra, Jeremy Du Croz, Anne Greenbaum, S Hammerling, Alan McKenney, et al. Lapack users' guide. 1999.

[GREA11] Pablo García-Risueño, Pablo Echenique, and José Luis Alonso. Exact and efficient calculation of lagrange multipliers in biological polymers with constrained bond lengths and bond angles: Proteins and nucleic acids as example cases. *Journal of computational chemistry*, 32(14):3039–3046, 2011.

[RCB77] Jean-Paul Ryckaert, Giovanni Ciccotti, and Herman J.C Berendsen. Numerical integration of the cartesian equations of motion of a system with constraints: molecular dynamics of n-alkanes. *Journal of Computational Physics*, 23(2):327 – 341, 1977.