Facultad de Ciencias

Desarrollo de un Código Kinetic Monte Carlo para su Aplicación en Ciencia de Materiales

Development of a Kinetic Monte Carlo Program for its Application to Materials Science

Trabajo de Fin de Máster
para acceder al

MÁSTER EN COMPUTACIÓN

Autor: Pablo Martín García

Director\es: Hegoi Manzano Moro
Julio Luis Medina Pasaje

12 – 2015

A mi familia.

Índice

1. Introducción **4**

1.1. Planteamiento y objetivos 4

2. Método de Monte Carlo Cinético **5**

2.1. Historia 5

2.2. Aspectos teóricos 5

2.3. Aspectos prácticos de un código KMC 7

3. Diseño del programa KMC **10**

3.1. Metodología 10

3.2. Escritura del programa KMC 10

3.2.1. Creación del sistema de simulación 14

3.2.2. Búsqueda de vecinos 15

3.2.3. KMC 18

3.2.4. Agregación de efectos adicionales 21

3.2.5. ramdon number 24

3.2.6. Modificaciones para el litio catión 25

4. Aplicación al estudio de difusión de átomos intersticiales en cristales **26**

4.1. Introducción 26

4.2. Cálculo de la barrera de energía de difusión de átomos intersticiales en óxido de calcio 26

4.2.1. Barreras de energía del protón 27

4.2.2. Barrera de energía del litio catión 28

4.3. Estudio de la difusión del protón en el óxido de calcio 30

4.4. Estudio de la difusión del catión del litio en el óxido de calcio 33

5. Conclusiones **34**

Resumen

La creación de nuevos materiales con propiedades mejoradas es un tema de actualidad, y la simulación a escala atómica o molecular es una herramienta útil que permite comprender el origen de ciertas propiedades de dichos materiales y sugerir rutas y métodos para mejorarlas. Sin embargo, surge el problema de trasladar los resultados de la simulación atomística a una escala macroscópica. Métodos como Density Functional Theory (DFT) o Molecular Dynamics (MD) permiten el estudio de los materiales en una escala de tiempo muy pequeña, del orden de femtosegundos y nanosegundos respectivamente, mientras que propiedades como la difusión de iones dentro del material, disolución, o degradación necesitan una escala temporal mucho mayor para ser perceptible. El método de simulación Monte Carlo cinético, Kinetic Monte Carlo (KMC) permite simulaciones en una escala temporal del orden de segundos a partir de las barreras de energía de los procesos involucrados. Por tanto, el objetivo principal de este trabajo es desarrollar un código KMC flexible para su aplicación en el campo de ciencia de materiales.

Abstract

The creation of new materials with improved propierties is a present aim, and the simulation in an atomistic or molecular escale is an useful tool which allows to understand the origin of some propierties of these materials and to suggest routes and methods to improve them. However, there is a problem in transfer the simulation results to a macroscopic scale. Methods like Density Functional Theory (DFT) or Molecular Dynamics (MD) enable the study of the materials in a very short time scale, in the order of femtoseconds and nanoseconds respectively, while propierties like ion diffusion in a material, disolution, or degradation need a much higher time scale in order to be perceptible. The simulation method of Kinetic Monte Carlo (KMC) enables simulations in a time scale of seconds using energy barriers of the involeved processes. So, the main target of this essay is to develop a flexible KMC program for its aplication in the material science field.

1. Introducción

1.1. Planteamiento y objetivos

La simulación se ha convertido en una parte importante de la investigación en las últimas décadas gracias al cada vez mejor desarrollo de los computadores de gran capacidad o super ordenadores. Su uso se extiende diversos campos como la física, la química, la biología, la economía y las ciencias sociales [1]. Es habitualmente empleada para dar soluciones a sistema cuya solución analítica de forma cerrada simple no es posible. Es una herramienta que, si bien es cierto no puede sustituir los resultados de un experimento real, si puede ser una herramienta complementaria que permita por un lado explicar el origen de los resultados experimentales, y por otro lado realizar predicciones sobre el comportamiento de un sistema, de manera que ayuden a diseñar dispositivos experimentales que faciliten medidas de sus propiedades.

Para que un programa de simulación sea bueno debe utilizar un modelo simplificado de la realidad que reproduzca su comportamiento de la manera más precisa posible, también en las condiciones del sistema. Y que además, sea eficiente; de nada sirve predecir el tiempo a dos días vista si tenemos un programa de simulación que tarda tres días en darnos la respuesta. La característica común de todos los modelos de simulación es generar una muestra de escenarios representativos, en la que una enumeración completa de todos los posibles estados sería compleja o imposible.

En el *grupo de Espectroscopía Molecular* del Departamento de *Química-Física* de la universidad del País Vasco UPV/EHU, se investiga, entre otras cosas, procesos físico-químicos que tienen lugar en materiales para aplicaciones energéticas, y estructurales. La creación de nuevos materiales con propiedades mejoradas es un tema de actualidad y la simulación a escala atómica o molecular es una herramienta útil que permite entender el origen de ciertas propiedades de dichos materiales y sugerir rutas y métodos para mejorarlas. Sin embargo surge el problema de trasladar los resultados de la simulación atomística una escala macroscópica. Los métodos de relajación molecular como *Density Functional Theory* (DFT) o *Molecular Dinamycs* (MD) permiten el estudio de los materiales en una escala de tiempo muy pequeña, del orden de femtosegundos y nanosegundos respectivamente. Propiedades como la difusión de iones dentro del material, disolución, o degradación necesitan una escala temporal mucho mayor para ser percibible. El método de simulación Monte Carlo cinético, *Kinetic Monte Carlo* en inglés (KMC) permite simulaciones en una escala temporal del orden de segundos gracias al uso de la estadística, lo que permite obtener su evolución general promedio.

Debido a ello, el *grupo de Espectroscopía Molecular* identificó los métodos KMC como interesantes para su investigación. Tras hacer un estudio del arte de las diferentes herramientas computacionales, se comprobó la falta de un programa general KMC open-source como ocurre en otros campos. Inicialmente se pensó en utilizar el programa *SPPARKS* desarrollado por los laboratorios nacionales de *Sandia* en Estados Unidos [2]. No obstante, esta herramienta disponía de módulos demasiado específicos como para emplearse de manera general. Se decide entonces intentar modificar *SPPARKS* para generalizarlo, pero viendo las complicaciones que suponía, se opta por realizar un nuevo programa basado en este pero con los requisitos que se deben cumplir. Por tanto, el objetivo principal de este trabajo es **desarrollar un código KMC flexible que sea capaz de simular diversos procesos que tienen lugar en materiales de interés**, como pueden ser la difusión de iones en baterías [3, 4] y la degradación de materiales y disolución [5, 6].

En la realización del programa es necesario enfocarse en un sistema lo más simple posible, pero teniendo en cuenta que deberá ser ampliado para que pueda ser igual de útil para materiales de diversas estructuras. Como problema modelo para probar la primera versión del programa, se ha elegido estudiar la difusión intersticial de protones en el óxido de calcio (CaO). A partir de la implementación de este caso se podrá extender a sistemas más generales con relativa sencillez. Como método de comprobación del funcionamiento del programa también se trata la difusión del litio en el óxido de calcio.

2. Método de Monte Carlo Cinético

2.1. Historia

El método comúnmente denominado Monte Carlo Cinético tiene ya varios años de historia. Su antepasado, el método de Monte Carlo se remonta a 1946 de mano de Stanislaw Ulam y John von Neumann [7]. Ulam, durante una enfermedad estaba jugando al solitario y se percató de que era mucho mas fácil tener una idea del resultado general del solitario haciendo numerosas pruebas con las cartas y contando las proporciones de los resultados que computar todas las posibilidades de combinación formalmente. Se le ocurrió que el mismo procedimiento podía ser utilizado en el estudio en el que trabajaba de dispersión de electrones en *Los Álamos*, en el que debía resolver ecuaciones integro-diferenciales con una difícil solución analítica. Ulam le mencionó el método a von Neumann, quien después de un escepticismo inicial, acogió con interés el método y comenzó a aplicarlo para rastrear la generación isotrópica desde una composición variable de material activo a lo largo del radio de una esfera. Una de las primeras aplicaciones de este método a un problema determinista fue llevada a cabo en 1948 por Enrico Fermi, Ulam y von Neumann cuando consideraron los valores singulares de la ecuación de Schrödinger [8].

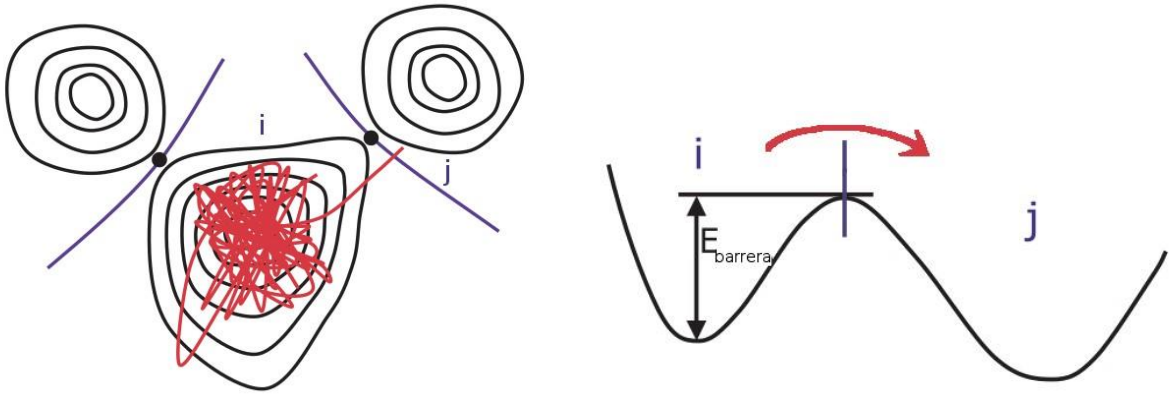
El método KMC como lo conocemos en la actualidad se desarrolló en 1966 por Young y Elcock [9] para el estudio de difusión de huecos en una aleación binaria. De forma independiente al trabajo de Young y Elcock, Bortz, Kalos y Lebowitz [10] desarrollaron un algoritmo KMC para simular el modelo de *Ising* (crecimiento de un material), el cual llamaron *n-fold way*. Las bases de su algoritmo son las mismas que en el caso de (*Young 1966*), aunque ellos proporcionaron mucho más detalles sobre el método. En la actualidad tiene diversas aplicaciones como son el estudio de reacciones químicas [11], cinética de agregación coloidal [12], separación de fases [13, 14], procesos de absorción y emisión [13], transporte difusivo [3, 4], catálisis heterogéneas [15] y crecimiento y disolución de minerales y materiales [5, 6], entre otras.

2.2. Aspectos teóricos

El KMC es la aplicación a sistemas que evolucionan con el tiempo del método de Monte Carlo. En analogía con el juego de cartas de Ulam, los átomos de una molécula o un sólido pueden vibrar en su estado de equilibrio, o desplazarse a una nueva posición muchas veces en pocos femtosegundos. Sin embargo tras un cierto tiempo, en promedio el átomo habrá cambiado su posición, o seguirá en la misma (figura 1(a)).

En un sistema en el que un estado solo puede evolucionar a estados vecinos, como es el caso que nos ocupa, se puede demostrar matemáticamente que la probabilidad de tener una transición entre estados es independiente del tiempo anterior. Es lo que se conoce con el nombre de propiedad de Markov [17]. Físicamente hablando, se puede suponer que el estado o las transiciones anteriores no interfiere en la probabilidad de la transición desde el estado actual, ya que el tiempo de residencia en un estado es mucho mayor que el periodo de vibración del átomo, y por tanto "se olvida" del proceso que le llevo al estado actual. Así pues, se puede definir una tasa constante k_{ij} que caracteriza la probabilidad, por unidad de tiempo, de que el sistema i evolucione a un estado vecino j, y esa probabilidad es independiente de los pasos que precedieron. Dichas constantes de transición vienen determinadas únicamente por la barrera de energía potencial entre estados (figura 1(b)). Más adelante se describirá como se pueden obtener estas barreras de energía empleado simulaciones atomísticas (apartado 4.2).

Por lo tanto, durante un corto incremento de tiempo, un átomo tiene la misma probabilidad de escapar del potencial que la que tenía en el incremento de tiempo anterior, lo que da una probabilidad de decaimiento exponencial. La probabilidad de que el átomo no haya escapado es:



(a) (b)

Figura 1: Esquema del movimiento de un átomo en el potencial de un cristal. Un átomo vibra en su posición de equilibrio hasta que cambia su posición de del estado i al j al superar la barrera de energía.
[16]

$$p_{residencia}(t) = e^{-k_{tot}} \tag{1}$$

donde k_{tot} es la tasa total de escape de la partícula. Se desea obtener la función de probabilidad $p(t)$ para el primer escape en un tiempo t^i ya que a partir de cierto instante se reinicia el proceso. Sabiendo que la integral de dicha función en dicho instante debe ser igual a $1 - p_{residencia}(t^i)$ y teniendo en cuenta la derivada de la ecuación 1, se obtiene la expresión:

$$p(t) = k_{tot}.e^{-k_{tot}t} \tag{2}$$

El tiempo promedio de escape τ es el primer momento o media de esta distribución:

$$\tau = \int_0^\infty t.p(t)dt = \frac{1}{k_{tot}} \tag{3}$$

Es necesario tener en cuanta que el tiempo que se va incrementar en cada paso Δt no depende de la transición que se va a llevar a cabo, sino de la probabilidad de escape total. Si el escape del átomo puede ocurrir a más de una nueva posición, cada una de estos escapes tiene su propia tasa:

$$k_{tot} = \sum_j k_{ij} \tag{4}$$

y por lo tanto, su propia función de probabilidad

$$p_{ij}(t) = k_{ij}.e^{-k_{ij}t} \tag{5}$$

El escape de un átomo a un estado vecino sigue una distribución de Boltzman en lo que se conoce como ecuación de Arrhenius [18, 19]:

$$k_{ij} = A.e^{-E_a/K_b T} \tag{6}$$

donde E_a es la energía de activación, es decir, la energía necesaria para realizar un salto entre estados (eV) (ver

figura 1(b)), K_b la constante de Boltzman ($8{,}61733x10^{-5}\ eV.K^{-1}$), T la temperatura ($K$)y A es la denominada frecuencia fundamental, una constante que tiene en cuenta la contribución a la probabilidad de transición en el equilibrio debido a la vibración de los átomos (energía cinética). La frecuencia fundamental se puede aproximar a su primer término determinado por la ecuación 7 [20, 21].

$$A = \frac{K_b.T}{h} = 2{,}084x10^{10}\ T = 6{,}25x10^{12}\ s^{-1} \tag{7}$$

Donde h es la constante de Planck ($4{,}13566733x10^{-15}eV.s$). A temperatura ambiente, $A = 6{,}25x10^{12}\ s^{-1}$

El método KMC hace uso de probabilidades aleatorias tanto como para determinar la evolución temporal como para determinar la transición que se va a llevar a cabo. En el primer caso, necesitamos un número aleatorio que siga una distribución exponencial, ya que la función de probabilidad así lo exige. Por lo tanto, la forma de obtener el incremento de tiempo, a partir de la ecuación 3 es:

$$\Delta t = \frac{1}{k_{tot}}.ln\ \ \frac{1}{u^i} \tag{8}$$

donde u^i es un número aleatorio de distribución uniforme $u^i \in (0, 1]$.

Así pues, recopilando, la aplicación práctica de este método sería la siguiente (ver figura 2):

1. Se define el sistema y se fija en un tiempo inicial $t = 0$.
2. Se crea una lista de todas las transiciones posibles de cada una de los átomos k_{ij}.
3. Se calcula la función cumulativa $K_i = \sum_{j=1}^{i} k_{ij}$ para todo $i = 1, \cdots, N$ donde N es el número de transiciones.
4. Se genera un número aleatorio u según una distribución uniforme $u \in (0, 1]$.
5. Se busca la transición i que se llevará a cabo determinada por el número aleatorio, debiendose cumplir que $R_{i-1} < uR_N \leq R_i$. Generalmente se usa el método '*bynary search*' o búsqueda en arbol.
6. Se lleva a cabo la transición encontrada.
7. Se genera un nuevo número aleatorio u^i según una distribución uniforme $u^i \in (0, 1]$.
8. Se actualiza el tiempo $t = t + \Delta t$. Se obteniene Δt de la ecuación 8.
9. El estado de nuestro sistema ha cambiado y es necesario crear una nueva lista con todas las posibles transiciones, es decir, volver al punto 2.

2.3. Aspectos prácticos de un código KMC

Terminada la presentación del método KMC, se pueden indicar algunos aspectos importantes:

- El tiempo de ejecución depende directamente del número de transiciones del sistema N. Esto implica atendiendo a la ecuación 8, por un lado, que si el número de transiciones que se pueden llevar a cabo es muy pequeño, la evolución temporal daría tiempos muy altos, y sería entonces necesario simular tiempos aún más altos para que el proceso se pudiera considerar estadístico. De la misma forma, si el número de transiciones es muy elevado, desencadenaría en una evolución temporal quizás más pequeña de la deseada en tiempos de simulación largos.

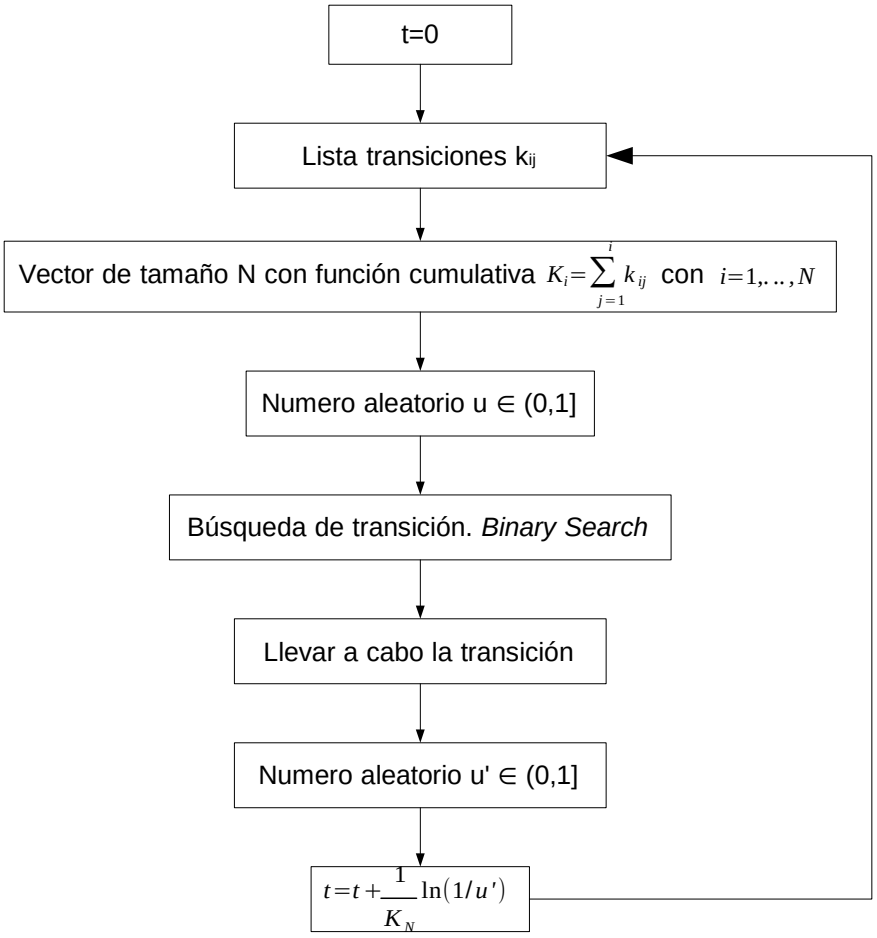


Figura 2: Diagrama de flujo del método KMC

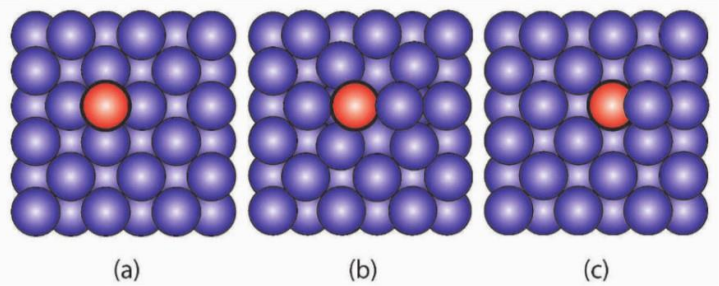


Figura 3: Cambio de posición del átomo resaltado en un metal con estructura fcc. a) estado inicial, b) paso intermedio. c) estado final. En lugar del intercambio con un átomo vecino, el átomo resaltado obliga al venido ha desplazarse a una tercera posición para ocupar su lugar.
[16]

- Fenómeno similar ocurriría en caso de que, aun disponiendo de un elevado número de transiciones, muchas de estas fueran muy probables (barrera de energía muy pequeñas) como se da por ejemplo en el caso de difusión de átomos intersticiales en metales ($\sim$ 0.1 ev) [16, 19]. En este caso, serían necesarias simulaciones muy largas (muchas transiciones) para que el sistema evolucionase a través de otra transición de mayor barrera de energía. Una forma de solucionar este problema sería aumentar dichos valores de barrera de energía artificialmente. Otra forma más elegante pero al fin y al cabo análoga es llevar constancia de las últimas transiciones de estado del sistema realizadas y en caso de ser un gran número de ellas la misma, obligar al sistema a que escoja cualquier otra quitándola de la lista de transiciones del siguiente paso. Se obtendrían buenas predicciones si los procesos rápidos no implican un cambio significativo de las propiedades de estudio, o que su promedio neto sea nulo. Evidentemente, en este caso es necesario evaluar el efecto de la manipulación de las barreras de energía en el comportamiento físico de la simulación.

- El método KMC podría en condiciones ideales dar una evolución exacta del sistema. Para ello, todas las posibles transiciones de cada uno de los átomos deberían estar perfectamente catalogadas. Esto es, que se conociesen todas las posibles transiciones con su respectiva tasa de transición k_{ij}, también con los átomos vecinos alejados con quienes, aunque muy improbablemente, pudieran intercambiarse. Además habría que tener en cuenta que los materiales no son perfectos, tienen defectos estructurales y/o químicos que interfieren en las barreras de energía. Imaginando el hipotético caso de que tengamos todo este catálogo de transiciones perfectamente definido, hay que tener en cuenta que hay que hacer una lista de transiciones por cada átomo, y en un sistema con un número de átomos muy elevado, sería computacionalmente costoso. Aunque bien es cierto que estos inconvenientes no son muy relevantes para KMC [22]. Lo realmente importante es que el catálogo de efectos construido pueda tener en cuenta todas las posibles transiciones relevantes para el estudio de la propiedad deseada. En la figura 3 se muestra un fenómeno que es el principal método de difusión de átomos en metales de estructura fcc. No es un proceso intuitivo, ya que la difusión tiene lugar mediante un intercambio atómico en lugar de un desplazamiento, y no fue descubierto hasta 1990, por lo que simulaciones por KMC que no tuviesen en cuanta este mecanismo no reproducirían correctamente la evolución temporal de la difusión [23].

- La eficiencia en una simulación KMC depende fundamentalmente de la forma en la que se busca la transición que se va a llevar a cabo, punto 5 anteriormente descrito.

  El método más extendido es el llamado *'binary search'* [24] y es el elegido para este trabajo. En este método, en cada paso de simulación se deshecha el catálogo de transiciones posibles construido anteriormente y se hace uno

nuevo. Seguidamente se busca en el catálogo nuevo la transición que se va a realizar en ese paso. La forma en que se hace esta búsqueda se detallada más adelante (sección 3.2.3). Únicamente indicar que tiene una eficiencia de $O(lnN)$ donde N es el número de transiciones.

Existe otro método de búsqueda que consiste en hacer un catálogo que ordene y agrupe las tasas de transición en una lista inversa (de más a menos tasa) lo que asegura una búsqueda inmediata[25]. Este método es independiente del número de transiciones N, pero su eficiencia desciende con el número de diferentes tasas de transición en el sistema, es decir, de grupos en los que se agrupan. Como se ha mencionado antes, en un sistema que se quiera que represente fielmente la realidad se esperan encontrar muchos valores distintos para k_{ij}.

En nuestro programa nos hemos decantado por el método *'binary search'* por dos motivos: su sencillez de implementación y porque se busca realizar un programa general que pueda albergar numerosos valores para las tasas k_{ij}.

3. Diseño del programa KMC

3.1. Metodología

Este trabajo se ha elaborado enteramente en un ordenador personal dotado con el sistema operativo *'linux Mint'* de 64-bits. El Programa esta desarrollado en el lenguaje *c++*. Para las posibles dudas en el lenguaje de programación, se ha acudido al manual [26, 27] y múltiples foros.

Se ha utilizado para su escritura el editor de texto *'Sublime text'* que ofrece ciertas utilidades que facilitan la tarea del programador, tales como el resaltado de color de sintaxis en numerosos lenguajes de programación, una visión global del código, y multientrada de texto [28].

Se ha comprobado su funcionamiento al compilarse con la version 4.1.2 y 4.8.4 de g++, aunque las librerías empleadas son tan básicas que es esperable que funcione con cualquiera.

Se han realizado pruebas de su funcionamiento tanto en él, como en el servicio de computación de alto rendimiento del País vasco *i2basque* [29]. En este último, además se realizaron numerosas simulaciones con el programa *SPPARKS*.

Para las simulaciones realizadas con *SIESTA* para el cálculo de las barreras de energía descrito más adelante (sección 4.2) se utilizó el servicio general de informática aplicada a la investigación de la upv [30].

Para la visualización de los resultados se han utilizado herramientas como *OVITO* [31], *VMD* [32] y *VESTA* [33]. La primera y segunda son herramientas más especializadas y complejas y que permiten además la visualización de películas de los diferentes estados. Mientras que la última es una herramienta sencilla e intuitiva que permite un visualizado rápido.

3.2. Escritura del programa KMC

Se busca elaborar un programa KMC eficiente para el estudio de diversos procesos tales como difusión iónica y disolución en materiales. Para ello se parte del caso particular de estudiar la difusión de protones en el óxido de calcio y a partir de ahí, generalizarlo. Se han introducido también algunas fenomenologías en el modelo como es la existencia de un campo eléctrico o la incorporación en el sistema de nuevos átomos para completar el programa. Como comprobación del funcionamiento del programa, y además como objeto de estudio, se ha tratado el caso de la difusión del litio como catión Li^+ en el mismo material.

El programa desarrollado consta de un archivo (`CaOH.cpp`) y su cabecera (`CaOH.h`). La cabecera, además de definir los métodos de (`CaOH.cpp`), contiene una serie de variables de control de simulación, tales como el tiempo

de simulación, numero de pasos, tamaño de la celda de simulación, etc. El contenido de la cabecera se describe resumidamente a continuación, para luego dar paso a una explicación más detallada de cada uno de los métodos y su papel en el conjunto del programa.

CaOH.h

```
#define Kboltz  8.6173325e-5 //eV K-1
#define numeroposiciones 56

int dimension_x=5;
int dimension_y=5;
int dimension_z=5;

double parametro_a=4.85;
double parametro_b=4.85;
double parametro_c=4.85;

static double epsilon= 0.00000001;
double distancia_ca_o=0.5*4.85;
double distancia_h_ca= sqrt((0.5*sqrt(2)/2)*(0.5*sqrt(2)/2)+((0.5*sqrt(2)/2-1/4.85)
    *(0.5*sqrt(2)/2-1/4.85)))*4.85;
double distancia_ca_ca= sqrt(2.0)*0.5*4.85;         //=distancia oxigeno - oxigeno
double distancia_h_o=1;
double distancia_hc_hc= sqrt(pow(0.0-1/(sqrt(2)*4.85),2.0)+pow((0.5-1/(sqrt(2)*4.85)
    -0.5),2.0))*4.85;
double distancia_hl_hl=(0.5*sqrt(2.0)-2/4.85)*4.85;
double distancia_maxima=distancia_ca_ca;

static int numeropasos=20;      //estados que vamos a tener (numeropasos + inicial)
static double tiemposimulacion=1.0; //segundos
static double temperatura=300;  //K
double intensidadE = 0.0025; // eV/A

int maximos_vecinos=0;

double tiempo=0.0;
int step=0;

//-------------------------------------------------------------//
//-------------------------Metodos-----------------------------//
//-------------------------------------------------------------//

int crea_celda(double **&celda);

int crea_box(double **&box, double **&celda,int dimension_x,int dimension_y,int
    dimension_z );

int obten_max_vecinos(double **&celda);

int obten_vecinos(int **&boxvec, double **&box, int maximos_vecinos);

int mi_kmc(int **&boxvec, double **&box);

int imprimeestado(double **&box);
```

Parámetros

1 `Kboltz` define el valor de la constante de Boltzman.

2 `numeroposiciones` define el número de posiciones por celda. Es importante notar que el numero de posiciones por celda va a ser siempre constante, y que estas posiciones van a permanecer inmóviles en el espacio. Es decir, el sistema esta discretizado en posiciones prefijadas y la identidad de dichas posiciones va a evolucionar con el tiempo. A cada tipo de posición le corresponde un mismo tipo de átomo, o un hueco, pudiendo un átomo desplazarse entre posiciones vecinas mediante un intercambio de identidades entre posiciones.

4-6 `dimension_x`, `dimension_y` y `dimension_z` define la dimensión del sistema de simulación. La periodicidad que posteriormente implementaremos (sección 3.2.2) permite simular *bulks* (porción de material infinitamente mayor) mediante representaciones más pequeñas de su estructura. Es necesario optimizar el tamaño del sistema a estudiar para alcanzar un compromiso entre la fiabilidad de la simulación y el tiempo de simulación. Por un lado, se debe definir un sistema con un tamaño mínimo que pueda reproducir las transiciones a estudiar. Por otro lado, definir un sistema mayor va a repercutir en la carga de trabajo computacional.

8-10 `parametro_a`, `parametro_b` y `parametro_c` es el tamaño de la celda unidad. Dependerá del material estudiado.

12 `epsilon` es un parámetro necesario para definir distancias en los métodos computacionales, fruto de la falta de precisión. En métodos como en la búsqueda de vecinos (ver 3.2.2) en lugar de definir una distancia exacta a la que buscar determinado vecino, se define un pequeño rango en torno a esta distancia en la que buscar, con lo que se asegura encontrarle, pero sin dejar que `epsilon` sea tan grande que introduzca vecinos indeseados. Este parámetro se utiliza también para excluir el valor 0.0 al generar números aleatorios.

13-19 indica las distancias existente entre un tipo de átomo y sus átomos vecinos. Son utilizadas por el método de búsqueda de vecinos. Evidentemente cuanto más irregular sea el sistema más cantidad de distancias se van a definir en esta parte. En cambio, cuanto mas regular sea, hay más posibilidades de que haya más de un tipo de átomo con un vecino a una determinada distancia. Lo que puede llevar a errores en el algoritmo de búsqueda de vecinos. Por ejemplo en el hipotético caso de que se quiera hacer una lista de vecinos oxígenos que tiene un protón y que estén situados a una distancia de 1 Å el algoritmo de búsqueda de vecinos los detectará correctamente, pero si resulta que a esa misma distancia hay un protón que no se ha tenido en cuenta, se incluirá como vecino y se tendrá una lista errónea.

21 `numeropasos` determina el número de estados deseados que se quieren ver de la simulación, esto es, las 'fotos' del sistema. Pueden ser tantos como se quieran.

22 `tiemposimulacion` determina el tiempo durante el cuál se quiere simular el sistema, en segundos.

23 `temperatura`, en kelvin, es la temperatura a la que está el sistema, necesaria para determinar la tasa de cada transición (ver ecuación 6).

24 `intensidadE`, en eV/Å es la intensidad del campo eléctrico en la dirección z a la que está sometido el sistema (ver sección 3.2.4).

26 `maximos_vecinos` es el número máximo de vecinos que puede tener cualquier tipo de átomo. Tiene como principal función la comprobación del correcto funcionamiento del método de búsqueda de vecinos.

28 `tiempo` representa el tiempo actual en una simulación que se está llevando a cabo. Es una variable de control para hacer fotos de los estados que se van generando, y como comprobante del tiempo que se ha simulado antes de tener un posible error.

29 `step` es el número de transiciones que se van llevando a cabo. Si se desea, se puede hacer una foto del estado cada vez que haya una única transición poniéndola como variable de control al hacer la película. También se puede fijar como parámetro limitante de la simulación en lugar del tiempo.

En futuras versiones se prevé implementar un método de lectura de las variables desde un archivo externo o input que el usuario pueda especificar ciertos parámetros de la simulación sin necesidad de modificar la cabecera y recompilar.

Métodos

En la elaboración del programa se han utilizado tanto métodos computacionales ya conocidos con una eficiencia optimizada (‘binary search’) como métodos implementados a *motu propio* (condiciones de contorno periódicas del material (PBC) y cálculo de vecinos).

35 En el método `crea_celda` se especifican las posiciones de la celda unidad donde van a estar definidos los átomos y huecos.

37 El método `crea_box` utiliza la celda unidad que se pasa como parámetro, la (definida en `crea_celda`) y la multiplica en las direcciones del espacio deseadas especificadas por los parámetros `dimension_x` , `dimension_y` y `dimension_z`(4-6). El resultado será el sistema final de simulación.

39 El método `obten_max_vecinos` es un método que se le pasa como parámetro un sistema de simulación y devuelve el máximo número de vecinos que puede tener cualquier átomo definido en él en un sistema de simulación dado. Más adelante (ver 3.2.2) se especifica más detalladamente el algoritmo empleado.

41 El método `obten_vecinos` calcula los vecinos para cada átomo definido en el sistema de simulación `box` y los guarda en una matriz `boxvec`. El algoritmo empleado es en esencia el mismo que el anterior `obten_max_vecinos`, pero con un grado de complejidad mayor al añadirle la periodicidad de la caja de simulación (ver 3.2.2).

43 El método `mi_kmc` contiene todo el algoritmo KMC (ver 3.2.3). En el se define una matriz con los posibles eventos en base a la matriz de vecinos anteriormente definida `boxvec`, y se modifica una única posición en el sistema de simulación `box`. El proceso se repetirá mientras el tiempo que transcurre en cada paso sea menor que el tiempo de simulación que se haya definido `tiempo < tiemposimulacion` (22,28).

45 El método `imprimeestado` es el encargado de crear un fichero con formato legible para *OVITO* de los estados del sistema (ver apéndice). Son las fotos de la simulación. Este formato es el utilizado por *SPPARKS* en sus ficheros de salida.

Librerías

Dentro del fichero `CaOH.cpp` se incluyen una serie de librerías necesarias para los siguiente fines:

```
1 #include <math.h>
2 #include <stdio.h>
3 #include <iostream>
4 #include <fstream>
5 #include <stdlib.h>
6 #include <string.h>
7 #include "CaOH.h"
```

1 `math.h` permite el uso de funciones matemáticas tales como el logaritmo, el cuadrado, o la raíz cuadrada.

2-4 `stdio.h`, `iostream` y `fstream` contiene instrucciones para la entrada y salida, tanto a ficheros como a consola. Permite el reporte de errores a consola durante la simulación y de la creación de ficheros con los resultados.

5 `stdlib.h` es la librería estándar. Contiene métodos básicos tales como la generación de números aleatorios.

6 `string.h` contiene a parte de operaciones con strings, métodos para la manipulación de la memoria. Útil para la creación dinámica de matrices.

7 `CaOH.h` es la cabecera donde se encuentran definidos los parámetros y métodos propios.

3.2.1. Creación del sistema de simulación

```
int crea_celda(double **&celda){

        double posiciones[numeroposiciones][4] = {
                //sitios del calcio
                  0.0,0.0,0.0,1,
                  0.0,....
                //sitios del oxigeno
                  0.5,0.0,0.0,2,
                  0.0,0.5....
                //sitios del vacio o proton
                  //Cara X
                  0.0,1/(sqrt(2)*4.85),0.5-1/(sqrt(2)*4.85),3,
                  0.0,0.5-1/(sqrt(2)*4.85),1/(sqrt(2)*4.85),3,
                  0.0,0.5+1/(sqrt(2)*4.85),....
                        ..............................
```

Las posiciones son creadas dentro del método `crea_celda`. Para ello se define una matriz, `posiciones` de 4 columnas y tantas filas como `numeroposiciones` haya en la celda unidad. Las tres primeras filas son ocupadas por valores de doble precisión (*doubles*) con el valor de las coordenadas relativas en x, y, y z; el máximo valor será 1. La cuarta columna contiene un número que corresponde a la identidad de la posición, por ejemplo el tipo de átomo o un sitio vacío. En el código anterior $1 \rightarrow Ca$, $2 \rightarrow O$, $3 \rightarrow H$, $4 \rightarrow hueco$. Una vez se tienen las posiciones relativas, falta multiplicar por el tamaño de celda en cada una de las direcciones del espacio; `parametro_a`, `parametro_b` y `parametro_c`.

```
for (int i=0; i<numeroposiciones; i++){
        celda[i][0]=posiciones[i][0]*parametro_a;
        celda[i][1]=posiciones[i][1]*parametro_b;
        celda[i][2]=posiciones[i][2]*parametro_c;
        celda[i][3]=posiciones[i][3];
}
```

Aunque en principio se ha pensado en que estas coordenadas sean valores positivos, introducir números negativos no producirá ningún inconveniente, algo importante en el caso de sistemas periódicos. Con este método, se consigue una celda unitaria, pero el sistema de simulación será generalmente más grande. Para ello, en el método `crea_box` se multiplica la `celda` creada en las tres dimensiones del espacio según las `dimension_x`, `dimension_y` y `dimension_z` deseadas.

```
1  for (int i=0;i<numeroposiciones;i++){
2          for (int x= 0;x < dimension_x;x++){
3                  for (int y= 0;y < ...
4                          box[j][0]= celda[i][0]+x*parametro_a;
5                          box[j][1]= celda[i][1]+y*parametro_b;
6                          box[j][2]= celda[i][2]+z*parametro_c;
7                          if ((int)celda[i][3]==3){
8                                  box[j][3]= (int)(celda[i][3]+1.100-(0.101*drand48()));
9                          }else{
10                                 box[j][3]= (int)(celda[i][3]);
11                         }
12                         box[j][4]= (double) j ;     //para el seguimiento del atomo
13                 ...
14         ...
```

El sistema de simulación se guarda en `box`, una matriz de 5 columnas y tantas filas como posiciones totales tenga la caja de simulación, esto es, `numerposiciones` x `dimension_x` x `dimension_y` x `dimension_z`. El hecho de implementar una columna adicional tiene como finalidad hacer un seguimiento de los átomos. Cada átomo tiene un *id* asociado.

Se puede ver en las lineas 7-11 que en el caso de que el tipo de átomo sea 3, es decir un protón, puede ser transformado en un valor 4, un hueco con bastante alta probabilidad; 100 de cada 101 serán redefinidos como huecos. En un sistema físico real se espera una densidad muy baja de protones en el sistema. Pero ¿por qué esta forma enrevesada de definirlo? Simplemente porque se busca que los protones estén ubicados desordenadamente en un instante inicial, como se espera encontrarlos. En caso de haberse definido un número de protones en `celda` y después haberse multiplicado, se tendría que los protones ocupan la misma posición periódicamente. Lo que es realmente improbable.

En un futuro se implementará un método de lectura de posiciones y átomos de tal forma que un usuario pueda utilizar el programa con su propio sistema predefinido. Y se mantendrá este método de creación del sistema en un módulo aparte, por si se quisiera crear el sistema con esta utilidad.

3.2.2. Búsqueda de vecinos

Como se ha indicado anteriormente, este programa esta diseñado para poderse orientar a sistemas más complejos. Para dar cabida a numerosos sistemas, se ha desarrollado un método de búsqueda eficiente de vecinos.

Una vez creado el sistema de simulación, `obten_max_vecinos` se encarga de ver cuantas posiciones vecinas tiene un sitio. Este número de vecinos dependerá de las distancias posibles en las que pueda estar un vecino, esto es, de los valores `distancia_ca_ca`, `distancia_ca_o`, etc. El método consiste en crear un sistema de simulación `maxvec` de dimensiones 3x3x3 a partir de la `celda`. Con estas dimensiones se tiene el mínimo sistema en el que una celda unidad esta rodeada en las tres direcciones del espacio por otra; la celda central. Mientras el resto de celdas devolverán un valor de vecinos inferior al que tendrían si se considerase la periodicidad, la celda central no. Evidentemente esta suposición es válida para sistemas en el que los vecinos están ubicados a una distancia menor que el tamaño de una celda. En caso contrario habría que considerar sistemas mayores; 5x5x5 en caso de tener una distancia máxima de 2 celdas, 7x7x7 de 3, y sucesivamente. Buscando siempre que cualquier punto a la máxima distancia de vecino desde la celda central este dentro del sistema definido.

El método considera cada átomo uno por uno y calcula la distancia que hay a todo el resto de átomos. Si esta distancia coincide con la distancia a la que se espera encontrar un vecino +/- `epsilon`, se incrementa la cuenta de vecinos para ese átomo, hasta que se ha comparado con todos y se pasa al siguiente átomo, poniendo la cuenta en cero y repitiendo la operación. Debido a que las operaciones matemáticas tienen cierta precisión determinada por la

precisión del *double* que a su vez depende del sistema operativo, es necesario definir un rango muy pequeño en torno al valor que se quiera buscar, sin ser tan grande que añada valores no deseados. Este rango viene determinado por el parámetro `epsilon`.

Ir recorriendo una por una las distancias posibles e ir comparándolas no es muy eficiente, por eso la importancia de que el sistema sea lo menor posible.

```
for (int i=0; i < j ;i++){
                for (int l = 0; l < j; l++){
                        r=sqrt((maxvec[l][0]-maxvec[i][0])*(maxvec[l][0]-maxvec[i][0])
                            +(maxvec[l][1]-maxvec[i][1])*(maxvec[l][1]-maxvec[i][1]) +
                            (maxvec[l][2]-maxvec[i][2])*(maxvec[l][2]-maxvec[i][2]));

                if (r<distancia_ca_o+epsilon && r>distancia_ca_o-epsilon || r<
                    distancia_h_ca+epsilon && r>distancia_h_ca-epsilon || r<
                    distancia_ca_ca+epsilon && r>distancia_ca_ca-epsilon || r<
                    distancia_hc_hc+epsilon && r>distancia_hc_hc-epsilon || r<
                    distancia_hl_hl+epsilon && r>distancia_hl_hl-epsilon || r<
                    distancia_h_o+epsilon && r>distancia_h_o-epsilon ){
                        numerovecinos++;
                        if (numerovecinos>max){
                                max=numerovecinos;
        ........
```

Mismo mecanismo de búsqueda de vecinos utiliza el método `obten_vecinos`, que además implementa la periodicidad y guarda los vecinos de la posición en una matriz `boxvec`, que después será necesaria para definir transiciones. La periodicidad es necesaria para identificar los vecinos de las celdas que están situadas en el perímetro del sistema de simulación `box`. Sin ella no se podría dar idea de la evolución de un sistema mucho mayor. Para su implementación, todas aquellas posiciones que pueden tener vecinos fuera del sistema definido, esto es, que están en los límites, son desplazados según el número de veces que estén como esquinas. Un átomo puede estar en la esquina x, en la esquina y, en la esquina z, en dos a la vez, o en tres de ellas. (ver figura 4). Se definen unas variables de control, `esquinax` , `esquinay`, `esquinaz`, `esquinaxy`, etc según este situado el átomo considerado.

```
if (box[i][0]<distancia_maxima-epsilon){
        esquinax= 1;
}
if (box[i][0]>=parametro_a*dimension_x-distancia_maxima-epsilon){
        esquinax= -1;
}
if (box[i][1]<distancia_maxima-epsilon){
        esquinay= 1;;
}
...
```

En la figura 4 el átomo 1 señalado está situado en la parte superior del sistema, por lo tanto tendrá vecinos por encima de él que no se están considerando. La solución es, una vez localizados los vecinos en su posición original, se desplaza momentáneamente a la parte inferior del sistema, en el que detectará el resto de vecinos. Para volver posteriormente a ubicarle en su posición original.

Hay otros átomos como el 2, que está situado en la parte inferior izquierda de la caja. Desde su posición original le faltarían localizar vecinos por debajo de él, por la izquierda, y además por debajo a la izquierda. Es necesario desplazar el átomo 3 veces para localizar todos sus vecinos; Desplazándole horizontalmente, verticalmente, y vertical y horizontalmente simultáneamente. Procedimiento similar ocurre con el átomo 3, situado en la esquina del cubo. Este

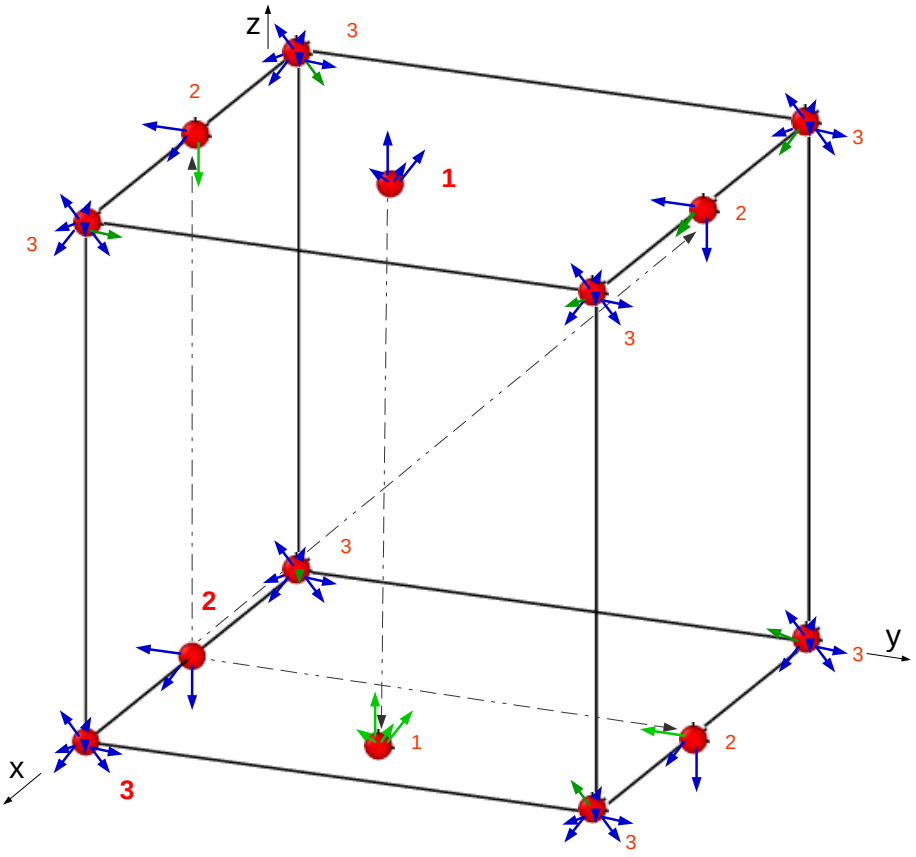


Figura 4: Esquema de las distintas posiciones a adoptar por un átomo situado en el perímetro del sistema de simulación. En azul, las direcciones en las que el átomo considerado tiene vecinos que no se encuentran dentro del sistema. En verde, dirección en la que sí se encuentran vecinos.

átomo tendrá vecinos en todas las direcciones, por eso es necesario desplazarle al resto de esquinas del sistema, en total 7 veces, para completar su catálogo de vecinos.

La forma de desplazar los átomos consiste en, una vez se ha echo su lista pertinente de vecinos (`l` ha llegado al final del bucle `l==j-1`) se va viendo si el átomo está en alguna esquina comprobando el valor de las variables de control. Si no lo está en ninguna se pasa al siguiente átomo. Pero en caso contrario, se posiciona el átomo en la nueva posición (lineas 4-5 del código inferior), y se reinicializa el bucle (`l=0`). Añadiendo entonces al catálogo los nuevos vecinos detectados. Se cambia el valor de las variables de control (lineas 6-7) para no repetir el proceso para el mismo átomo.

```
1  if (l==j-1 && esquinaxz==1){  //si ya se ha terminado de hacer los vecinos y esta en
       la esquinaxz
2          esquinax=esquinax/3;
3          esquinaz=esquinaz/3;
4          box[i][0]=box[i][0]+parametro_a*dimension_x*esquinax;
5          box[i][2]=box[i][2]+parametro_c*dimension_z*esquinaz;
6          l=0;
7          esquinax=3*esquinax;
8          esquinaz=3*esquinaz;
9          esquinaxz=2;
10 }
```

Nada nos impide obviar el método `obten_max_vecinos` y utilizar este también para obtener el mismo resultado con una pequeña modificación. Durante la escritura del código se elaboró primero, y dado su simplicidad y su función de comprobación se ha mantenido. Además permite definir el tamaño de la matriz `boxvec` a priori, sin necesidad de una creación dinámica. `obten_vecinos` es aun más pesado y requiere una alta carga computacional. Afortunadamente este algoritmo solo interviene una vez; los vecinos de cada posición no cambia con el desplazamiento de los átomos. Por tanto, en simulaciones con alto número de transiciones esta carga queda en una anécdota.

3.2.3. KMC

Una vez se tiene por un lado definidas todas las posiciones del sistema y los átomos que las ocupan en `box` y por otro el catálogo de vecinos que pertenece a cada átomo en `boxvec` es el momento de definir una lista de transiciones o eventos que pueden tener lugar en nuestro sistema. La identidad de cada posición tiene cierta probabilidad de intercambiarse con la identidad de una posición vecina. Este evento será empleado para simular el movimiento de un átomo mediante el intercambio de la identidad ‘átomo’ de una posición con la identidad ‘hueco’ de otra posición (ver punto 2 de la sección 2.3).

La lista de transiciones o eventos la definimos en la matriz `eventos`. Dicha matriz se define con tres columnas. En la primera, una vez calculado el valor k_{ij} correspondiente a la transición (ver ecuación 6), se guarda el valor acumulado (ver punto 3). En la segunda, el id del átomo que va a sufrir la transición. Y en la tercera, el id del átomo con el que se va a intercambiar, en caso de que haya; no todos los eventos son intercambio de posición. En caso de que no sea un intercambio, se guardará en la tercera posición un número identificativo del tipo de evento (ver sección 3.2.4).

El número de filas no se puede predecir a priori. En cada paso de KMC se redefine la lista de eventos completamente y no tiene porque haber el mismo número de eventos posibles. Por ejemplo, la aparición de un protón en el sistema implica un aumento del numero neto de ellos. Es por lo tanto necesario crear la matriz de forma dinámica, aumentando su tamaño al registrar un posible evento. La implementación de los eventos será mas compleja cuanto más variables del entorno del átomo haya que tener en cuenta, pero se pueden definir eventos tan complicados como

uno crea oportuno.

A continuación se muestra como se crea un evento para los protones del sistema. Este código esta ubicado dentro de un bucle que recorre todos los átomos del sistema, y en caso de cumplirse las condiciones especificadas, se guarda en la matriz `eventos`. Describe el intercambio de un protón con un hueco cercano.

```
if ((int)box[i][3]==3){          //si el atomo es un proton
        if ((int)box[boxvec[i][j]][3]==4){      // si tiene un vecino que es un hueco
                rate = ffunda * exp(-0.08/(Kboltz*temperatura));  //se calcula la tasa
                     k_ij
                propensity+=rate;        //funcion cumulativa
                //Ampliacion de la matriz dinamicamente
                pointer=(double**)realloc(eventos,(numerotransiciones+1)*sizeof(double
                    *));
                pointer[numerotransiciones]=(double*)malloc(3*sizeof(double));
                .....
                //Se guarda la funcion cumulativa
                eventos[numerotransiciones][0]=propensity;
                eventos[numerotransiciones][1]=i;    //el atomo que sufre el evento
                //el atomo (hueco) por el que se intercambia
                eventos[numerotransiciones][2]=boxvec[i][j];
                .....
```

Una vez definido `eventos` en donde la primera columna da cuenta de la función cumulativa (ecuación 4), se genera el número aleatorio (ver punto 4) y se busca el evento que va a tener lugar (ver punto 5). El método empleado para esta búsqueda es el *binary search*. En la figura 5 se muestra el funcionamiento de este algoritmo. Básicamente, consiste en dividir en dos el rango de búsqueda y descartar uno de ellos si el valor a buscar no está en él. Se va acotando el rango de búsqueda hasta dar con el valor. Si se hiciera una búsqueda lineal, en el peor de los casos se tardaría N pasos en encontrarse, con N el número de transacciones. Con este algoritmo se reduce en el peor caso a lnN. El rango de búsqueda esta restringido por dos variables de control, el suelo `s` y el techo `t`. La posición elegida para la siguiente comparación, `c` vendrá determinada por ambos valores; por su valor medio.

```
while (encontrado){
        if (u*ktot>eventos[c-1][0]){
                if ( u*ktot<=eventos[c][0]){
                        encontrado=!encontrado;
                }else{
                        s=c;
                        c=c+ceil((t-s)/2.0);
                }
        }
        if (u*ktot<eventos[c-1][0] ){
                t=c;
                c=c-ceil((t-s)/2.0);
        }
}
```

En el momento que se detecte que el valor de numero aleatorio u por k_{tot} es mayor que el valor de la función cumulativa del anterior K_{c-1} (ver punto 3) pero menor que el de la propia K_c, la posición se habrá encontrado, c. En dicha posición se había guardado, en la tercera fila de la matriz el tipo de evento asociado. Así pues, según ese valor, se cambia el sistema en consecuencia. A continuación se muestra el caso más común y sencillo en el que dos posiciones intercambian sus identidades, un protón por un hueco, de manera que el evento desde el punto de vista físico es el movimiento del protón. Se usa una variable auxiliar `memoria` para guardar el tipo de átomo y el id

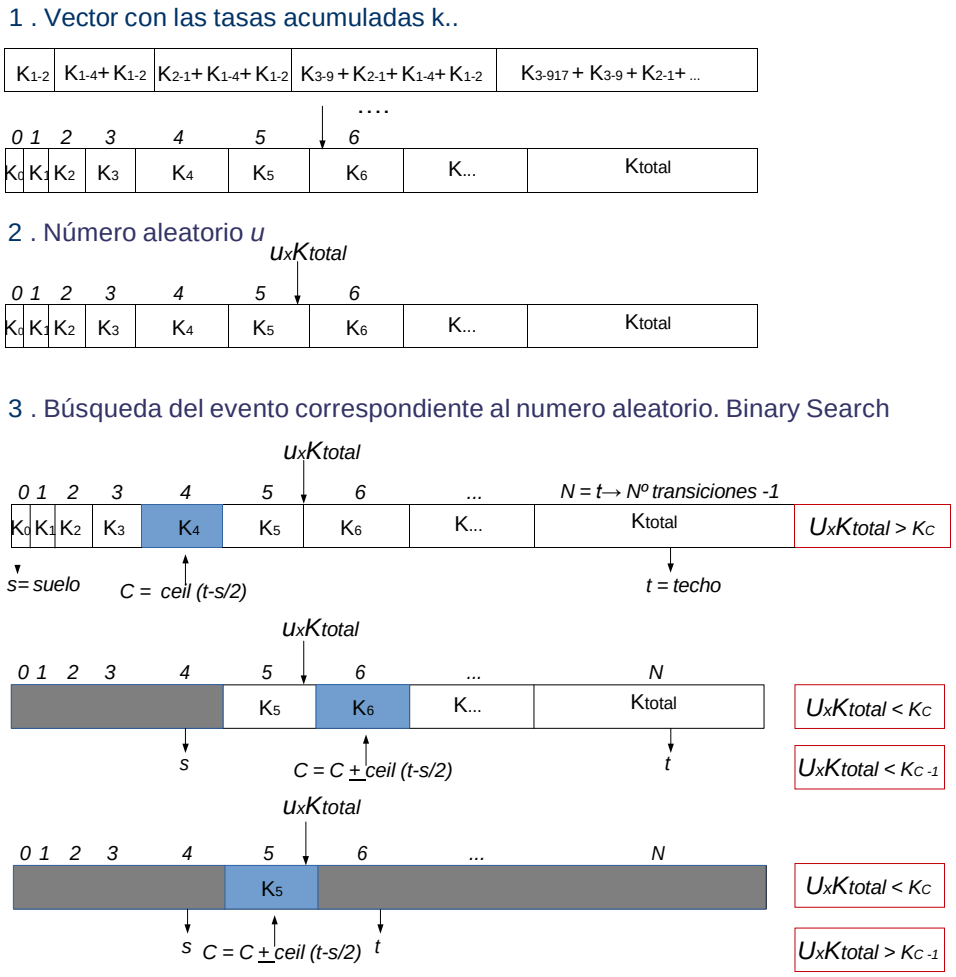


Figura 5: Esquema del funcionamiento de *'binary search'*. Después de haberse definido la función cumulativa, se genera el número aletorio y se busca el evento correspondiente

momentáneamente.

```
int memoria[2];
memoria[0]=box[(int)eventos[l][1]][3];    //se guarda el tipo de atomo
memoria[1]=box[(int)eventos[l][1]][4];    //se guarda el id del atomo

//intercambiamos
box[(int)eventos[l][1]][3]=box[(int)eventos[l][2]][3];
box[(int)eventos[l][1]][4]=box[(int)eventos[l][2]][4];

box[(int)eventos[l][2]][3]=memoria[0];
box[(int)eventos[l][2]][4]=memoria[1];
```

Ahora solo falta avanzar el tiempo del sistema. Para ello se genera un nuevo número aleatorio `uprima`, y se avanza según la ecuación 8. Tener en cuenta que el valor 0.0 debe ser excluido de los posibles valores. `drand48()` no excluye el 0 por lo que es necesario hacerlo manualmente. Para ello se adiciona el parámetro `epsilon` definido anteriormente en caso de encontrarle.

```
double uprima = drand48();
if (uprima==0) uprima+=epsilon;
tiempo+=1/ktot*log(1.0/uprima);
```

Con esto tendríamos el programa completo. Únicamente sería necesario imprimir en un fichero en un formato legible para un programa de visualización externo, como pueden ser *OVITO* o *VMD* (ver apéndice). De esto se encarga el método `imprimeestado`, el cual correrá en bucle imprimiendo en el fichero de salida datos hasta el `tiemposimulacion` deseado. Mientras el método `mi_kmc` esta corriendo constantemente, `imprimeestado` va a sacar datos solo cada cierto tiempo supeditado por el numero de pasos indicado.

```
double control=0.0;
while (tiempo<tiemposimulacion){
        mi_kmc(b,q);
        if (tiempo-control>tiemposimulacion/(double)numeropasos){
                imprimeestado(q);
                control+=tiemposimulacion/(double)numeropasos;
        }
}
```

3.2.4. Agregación de efectos adicionales

En futuras versiones se persigue que el programa sea lo más general posible tanto en el tipo de sistema a estudiar como en la propiedad a investigar. Para ello se han introducido una serie de movimientos adicionales o modificadores de la función de probabilidad que pueden describir diferentes procesos fisico-químicos.

Nuevos átomos

Según la teoría *ligand-exhange*, un proceso de disolución puede entenderse de la manera siguiente. Primeramente, las moléculas del disolvente reaccionan en la superficie del material. De esta manera, los enlaces de la red cristalina se rompen, siendo sustituidos por átomos del disolvente. Poco a poco aumenta el número de átomos en el sistema provenientes de moléculas de disolvente disociadas. La tasa de entrada al sistema dependerá por tanto de la barrera

de energía de disociación del disolvente en la superficie, y conociendo el valor de esta barrera se pueden incluir en el sistema eventos de este tipo de la misma forma descrita en la sección anterior.

A este tipo de evento se le ha otorgado un número identificador -3, que se guarda en la tercera columna de la matriz `eventos`.

```
if ((int)box[i][3]==4){    //Si es un hueco
        //Ampliacion de la matriz dinamicamente
        pointer=(double**)realloc(eventos,(numerotransiciones+1)*sizeof(double*));
        pointer[numerotransiciones]=(double*)malloc(3*sizeof(double));
        ...
        rate = ffunda * exp(-0.75/(Kboltz*temperatura));
        propensity+=rate;
        eventos[numerotransiciones][0]=propensity; //Se guarda la funcion cumulativa
        eventos[numerotransiciones][1]=i; //el atomo que sufre el evento
        eventos[numerotransiciones][2]= -3.0; //numero identificativo tipo de evento
}
```

Un aspecto a destacar es el hecho de que a la hora de la verdad, el disolvente entrará al sistema por una, varias, o todas las superficies del material. Esto conlleva la necesidad de una ruptura de las condiciones periódicas. Para un caso especifico, suponiendo que el material capta protones por la cara superior del cubo $z = 5.parametro\_c$, Se puede seguir afirmando que se cumple la simetría en las paredes del cubo. Los átomos en las paredes $x = 0$ e $y = 0$ siguen teniendo vecinos en $x = 5.parametro\_a$ y $y = 5.parametro\_b$, e inversamente. Pero no así lo átomos de $z = 0$ con $z = 5.parametro\_c$ (ver figura 4).

Disolución

Siguiendo el razonamiento del apartado anterior, aquellas unidades de la red cristalina que tenga un mayor número de enlaces rotos abandonarán la superficie del cristal pasando a la disolución. Conociendo el valor de la barrera de energía que un átomo necesita para disolverse según el entorno en el que se encuentra (número y tipo de vecinos), se pueden definir eventos de disolución del sistema. A este tipo de evento se le otorga un número identificador -2.

```
if ((int)box[boxvec[i][j]][3]==3) cuentaox++; //numero de protones que le rodean
        switch (cuentaox){ //diferente tasa segun el numero
                case 3:
                        rate = ffunda * exp(-0.12/(Kboltz*temperatura));
                case 4:
                        rate = ffunda * exp(-0.09/(Kboltz*temperatura));
                ...
                }
        //Ampliacion de la matriz dinamicamente
        pointer=(double**)realloc(eventos,(numerotransiciones+1)*sizeof(double*));
        pointer[numerotransiciones]=(double*)malloc(3*sizeof(double));
        ...
        propensity+=rate;
        eventos[numerotransiciones][0]=propensity; //funcion cumulativa
        eventos[numerotransiciones][1]=i; //atomo que sufre el evento
        eventos[numerotransiciones][2]=-2.0; //numero identificativo tipo de evento
...
```

Al crear matrices de forma dinámica, se hace un uso de la memoria virtual que no se libera hasta una vez se termina el programa. La creación y ampliación de la matriz `eventos` se produce cada paso de simulación de KMC, esto es, infinidad de veces. Es por lo tanto obligado liberar la memoria ocupada entre paso y paso.

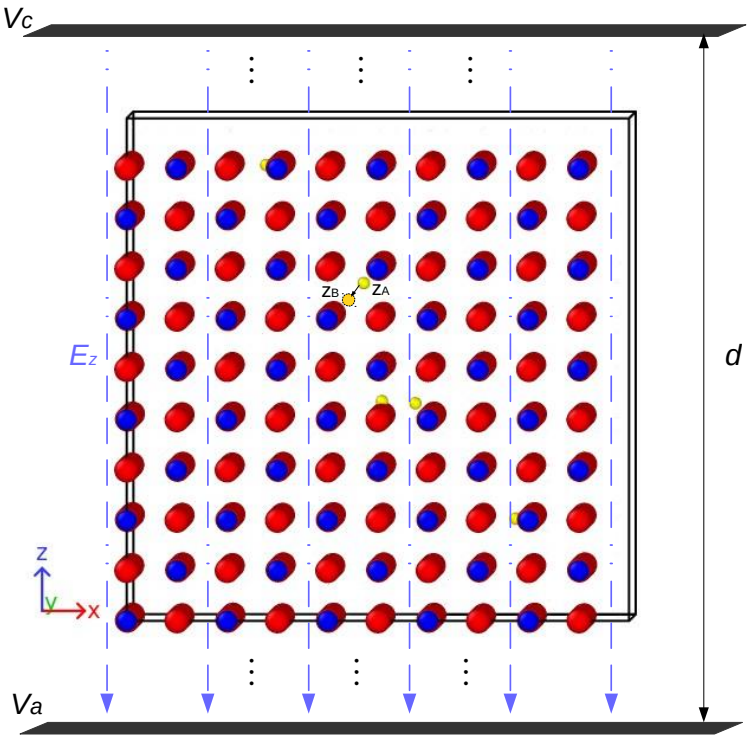


Figura 6: Sistema implementado para el estudio de la difusión con el campo eléctrico E. Un cátodo en la parte superior y un ánodo en la inferior crea un campo electrico homogeneo en el sistema de simulación, favoreciendo la transición de cargas positivas como el protón a posiciones de z menor

```
// Elimina cada vector de la matriz
for (int i = 0; i < numerotransiciones+1; i++) delete[] eventos[i];
// Elimino el vector principal
delete[] eventos;
```

Campo eléctrico

Un fenómeno que puede ser importante estudiar es la difusión en presencia de un potencial, por ejemplo eléctrico el caso de baterías, o químico en fenómenos de difusión y ósmosis.

Se ha supuesto el caso sencillo que campo eléctrico está en una única dirección, la z. Aunque implementarlo en otras direcciones del espacio sería análogo. Se ha considerado un campo constante, producido por el típico problema de dos placas infinitas y paralelas situadas por encima y debajo del sistema, suficientemente alejadas del sistema de tal forma que no interfieran en él. Las placas tienen la diferencia de potencial necesaria para producir en el material el campo eléctrico deseado(ver figura 6).

Introducir en el sistema un campo eléctrico no rompe las condiciones de simetría en caso de que este sea constante en todo el espacio; cuando un átomo se fuga por la parte de abajo del sistema se introduce uno ‘nuevo’ por la parte de arriba proveniente del resto del material. No ocurre lo mismo, por ejemplo, si se quisiese considerar una ubicación de las placas más cercana al sistema de simulación e imponiendo la condición de que los átomos en difusión no pudieran atravesarlas.

En un sistema de placas plano-paralelas la intensidad del campo eléctrico viene determinada por la diferencia de

potencial Vc del cátodo y Va del ánodo y la distancia d entre ellos según la ecuación 9 [34].

$$E = (Vc - Va)/d \tag{9}$$

El campo eléctrico del sistema realiza un trabajo en la dirección del campo, que puede ser tanto positivo como negativo dependiendo de la carga de la partícula considerada. En el caso del protón por su carga positiva preferirá ir a potenciales negativos, esto es, hacia $z = 0$. Dicho con otras palabras, se producirá una aumento de la barrera de energía si el protón intenta ir hacia z mayores que su posición, una disminución hacia z menores, y permanecerá igual si va hacia cualquier posición en la horizontal.

Este aumento o disminución de la barrera de energía viene determinado por el trabajo que realiza el campo al desplazar el átomo a su posición final B desde su posición inicial A según la ecuación 10

$$W_{AB} = q.E.(z_A - z_B) \tag{10}$$

En la implementación es necesario modificar la sección del código ya descrito en 3.2.3.

```
... //si el atomo es un proton
if ((int)box[boxvec[i][j]][3]==4){      // si tiene un vecino que es un hueco
    trabajocampo=intensidadE*(box[i][2]-box[boxvec[i][j]][2]); //trabajo del campo E
    rate = ffunda * exp((-0.03+trabajocampo)/(Kboltz*temperatura));

    propensity+=rate;    //funcion cumulativa
    //Ampliacion de la matriz dinamicamente
    pointer=(double**)realloc(eventos,(numerotransiciones+1)*sizeof(double*));
    pointer[numerotransiciones]=(double*)malloc(3*sizeof(double));
    .....
    //Se guarda la funcion cumulativa
    eventos[numerotransiciones][0]=propensity;
    eventos[numerotransiciones][1]=i;    //el atomo que sufre el evento
    //el atomo (hueco) por el que se intercambia
    eventos[numerotransiciones][2]=boxvec[i][j];
....
```

Otro tipo de potenciales, como puede ser el potencial químico, pueden ser simulados empleando la misma ecuación de trabajo variando la función lineal de campo eléctrico por otro tipo de funciones como pueden ser exponenciales o polinómicas.

3.2.5. ramdon number

El número aleatorio cobra especial importancia en el método KMC a la hora de escoger el evento que se llevará acabo, y a la hora de avanzar el tiempo del sistema. Hay que tener claro las características que poseen los números aleatorios que se generan. Habitualmente, los números creados por ordenadores no son realmente aleatorios; son pseudo-aleatorios. Esto quiere decir que la forma en la que se definen no da un comportamiento exactamente idéntico al que se espera de un conjunto de números que realmente si lo son, como pueden ser los obtenidos en algunos procesos físicos como el ruido atmosférico [35, 36], el ruido térmico [35] o fenómenos cuánticos[35, 37].

Los número generados por un ordenador responden a un algoritmo. Dicho algoritmo (ver ecuación 11) presenta dos características principales. Por un lado el número creado va a depender de una semilla *'seed'* x_0; para un determinado valor *'seed'* siempre se va obtener el mismo. Y por otro, va a tener un cierto periodo de repetición determinado por m; periodo m en el mejor de los casos. Esto se traduce en una disminución en la entropía de los datos.

$$x_{n+1} = (x_n.a + c)/mod\ m \tag{11}$$

Recientemente se ha descubierto una forma de hacer números aleatorios reales [38] a partir del jitter (variación del tiempo de comunicación de señales digitales entre sistemas) y ya hay disponibles herramientas para su implementación [39]. No obstante, para este trabajo interesa simulaciones aleatorias pero reproducibles. Es por ello que se ha hecho uso de la librería estándar de C `stdlib.h`, la cual tiene valores $c = 125$, $m = 32768$ y $a = 1103515245$. Y en la cual se puede fijar el valor de '*seed*' con `srand48(123)`.

Aunque de momento con la librería estándar `stdlib.h` se pueden tener buenos resultados en los sistemas simples que se han estudiado, se implementarán en un futuro números aleatorios reales.La falta de precisión por la periodicidad puede conllevar a la exclusión de una transición muy poco probable.

3.2.6. Modificaciones para el litio catión

Las modificaciones para la implementación del caso de la difusión del Li^+ han sido inmediatas ya que supone una reducción en la complejidad respecto al caso de la difusión de protones debido a las posiciones que estos ocupan. Esencialmente se ha realizado un cambio en la cabecera, en este caso llamada `CaOLi.h`, donde se han definido nuevas distancias.

```
....
double distancia_ca_o=0.5*4.85;
double distancia_li_ca= sqrt((sqrt(2.0)*0.25)*(sqrt(2.0)*0.25)+0.25*0.25)*4.85;
double distancia_ca_ca= sqrt(2.0)*0.5*4.85;   //=distancia oxigeno - oxigeno
double distancia_li_o= distancia_li_ca;
double distancia_li_li=distancia_ca_o;
double distancia_maxima=distancia_ca_ca;
....
```

y dentro del método `crea_celda` ya que la posición relajada del Li^+ en el óxido de calcio es la central del hueco intersticial [40].

```
int crea_celda(double **&celda){

        double posiciones[numeroposiciones][4] = {
                //sitios del calcio
                  0.0,0.0,0.0,1,
                  0.0,....
                //sitios del oxigeno
                  0.5,0.0,0.0,2,
                  0.0,0.5....
                //sitios del Li y de los vacios
                  0.25,0.25,0.25,3,
                  0.75,0.25,0.25,3,
                  0.75,0.75,....
                  ..............................
```

En un trabajo futuro se tratará de flexibilizar el programa para que los eventos y las tasas que tienen lugar en el sistema puedan ser definidos externamente en un fichero de configuración, flexibilizando el programa de simulación KMC en la medida de lo posible.

4. Aplicación al estudio de difusión de átomos intersticiales en cristales

4.1. Introducción

El primer proceso físico que se va a estudiar para testear el programa es la difusión de átomos intersticiales en cristales. Este proceso esta implicado en diversas propiedades de materiales como la conductividad iónica y térmica [41], ductilidad y otras propiedades mecánicas [42][43], la resistencia a oxidación en metales [44], o el desarrollo de baterías de estado sólido [45].

En nuestro caso escogemos estudiar la difusión de átomos intersticiales en el óxido de calcio debido a su simple estructura, y a su interés tecnológico. Desde la antigüedad se ha usado de conglomerante en materiales de construcción, por ejemplo para encalar fachadas de los edificios [46], y en la fabricación de fuego griego, el cual fue usado por los bizantinos para ganar numerosas batallas navales al tener la peculiaridad de seguir ardiendo en el agua [47]. En la actualidad, el óxido de calcio continua aplicándose en construcción, siempre ligado a un proceso de hidratación del material. Es empleado como estabilizante de suelo [48], para restaurar monumentos históricos y en la elaboración de pinturas y adobes [46]. Además, el óxido de calcio es empleado en nuevos campos, expecialmente en el sector energético. Por ejemplo se emplea como catalizador en la producción de biodiesel [49, 50, 51, 52] donde la capacidad catalítica del material depende en gran medida de la densidad de grupos OH en la superficie. Otra de sus utilidades es como almacenamiento de energía en un proceso denominado ‘thermochemical energy storage’ [53] que consiste en el almacenamiento de energía en procesos termodinámicos de alta reversibilidad. En el caso del óxido de calcio este proceso puede darse por adsorpción de dióxido de carbono y conversión en calcita [54].

$$CaCO_3 = + CaO + CO_2$$

o por adsorción de agua y conversión en portlandita [55]:

$$CaO + H_2O = + Ca(OH)_2$$

lo que es posible gracias a la alta reversibilidad de ambas reacciones.

En todas las aplicaciones mencionadas la presencia de protones provenientes de agua es fundamental. Por tanto, un estudio detallado de la difusión de protones en la red cristalina es de gran interés tanto científico como tecnológico.

4.2. Cálculo de la barrera de energía de difusión de átomos intersticiales en óxido de calcio

Como hemos visto anteriormente, para el estudio de una determinada propiedad (difusión, disolución, crecimiento, etc) mediante KMC es necesario conocer las barreras de energía de todos los posibles procesos que tienen lugar. La difusión de los iones intersticiales tiene lugar siguiendo un solo proceso en el caso de Li (el salto de un hueco a otro) y dos en el caso del protón (el salto de un oxígeno a otro y la rotación alrededor del oxígeno)

En este trabajo las barreras de energía para la difusión de átomos intersticiales de H y Li en óxido de calcio se han calculado utilizando simulaciones *ab-initio* basadas en el método de *density functional theory* (DFT). Los métodos DFT resuelven el problema de muchos cuerpos en un sistema polielectrónicos a partir de funcionales de la densidad electrónica del sistema [56]. Las barreras de energía se han calculado empleando el método denominado *Constrained Energy Minimization* [57]. Consiste en minimizar la energía del sistema modificando las posiciones de los átomos, imponiéndole una o más restricciones, como puede ser fijar una determinada posición de un átomo o imponer una distancia o ángulo entre átomos. De esta manera se hace evolucionar al sistema a través de la superficie de energía

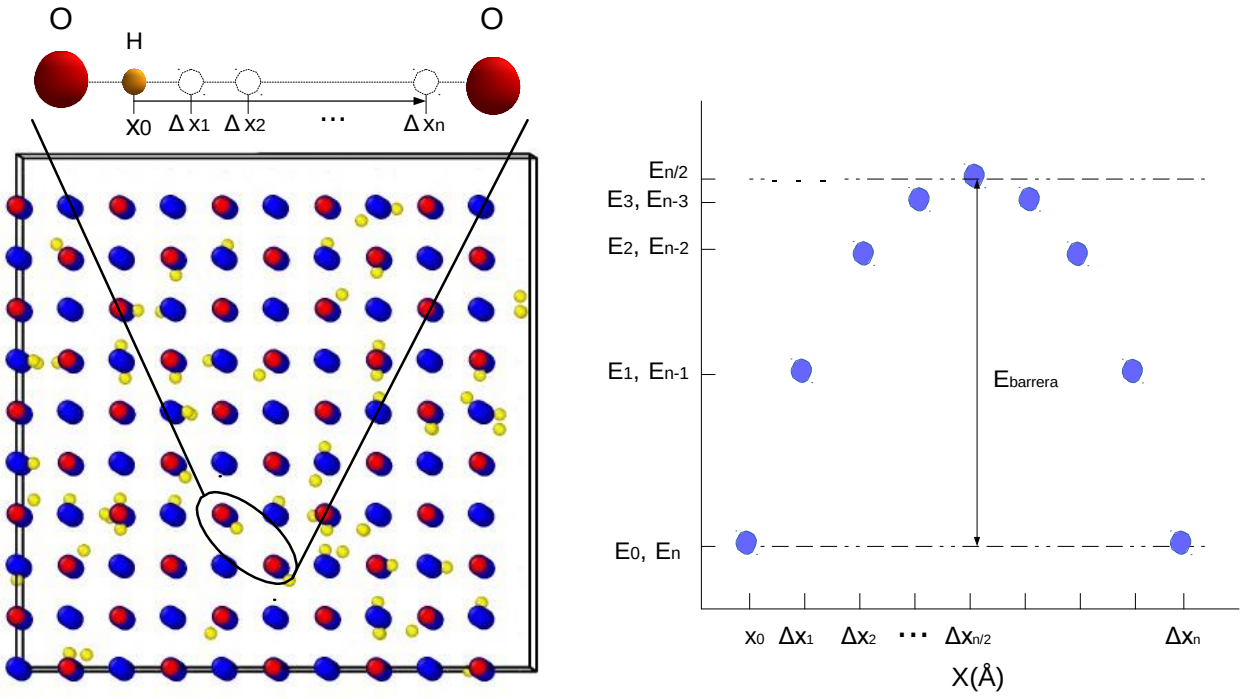


Figura 7: Posiciones intermedias para el cálculo de la Energia de gap en la transferencia de un protón entre dos oxígenos en el óxido de calcio utilizando la metodología *Constrain Energy Minimization*. Para cada posición del protón se obtiene una energía total del sistema que describe una función de cuyo máximo corresponde con el valor $E_{barrera}$,

potencial desde el punto de equilibrio inicial al final, siguiendo una trayectoria determinada por la variable prefijada. Monitorizando la diferencia de energía en diversos puntos de esa trayectoria se obtiene la barrera de energía del proceso, como muestra la figura 7. Como ejemplos que hacen uso de esta metodología se puede mencionar el cálculo de las barreras de energía de disociación de una molécula de agua en la superficie del óxido de magnesio [58] y el de la barrera de energía de la difusión de protones en cristales de $CaZrO_3$ [59].

Los cálculos de las barreras de energías se ha realizado empleando el código *SIESTA* [60]. Este emplea pseudopotenciales para describir electrones internos y combinación lineal de orbitales atómicos para los electrones de valencia. Los pseudopotenciales y bases (tipo Doble Zeta Polarizada) se han obtenido de trabajos previos en la literatura que han demostrado su precision para el estudio de óxidos [61]. Los cálculos se han realizado empleando el funcional de intercambio-correlación *PBE* [62], dentro de la *Generalized Gradient Approximations*. La carga neta al introducir iones positivos de H^+ o Li^+ fue compensanda con un background electrónico de carga contraria. Todas las simulaciones fueron realizadas en una supercelda de tamaño 2x2x2, condiciones periódicas, y las relajaciones se llevaron a cabo hasta reducir las fuerzas en cada átomo por debajo de 0.004 $eV \cdot Å^{-1}$.

4.2.1. Barreras de energía del protón

En primer lugar se ha realizado el cálculo de los valores de las barreras de energía para el protón. En el equilibrio, el próton ocupa una posición en torno al oxígeno, formando un enlace predominantemente covalente, a una distancia aproximada de 1 Å y orientado en la dirección de un oxígeno vecino (ver figura 8(a)). En total hay 12 posibles posiciones en la que un protón puede estar en equilibrio alrededor de un oxígeno.

El proceso de difusión del protón consta de dos posibles transiciones. La primera de ellas implica un salto o transferencia desde la posición de equilibrio en la que está enlazado a un oxígeno hacia una nueva posición de equilibrio

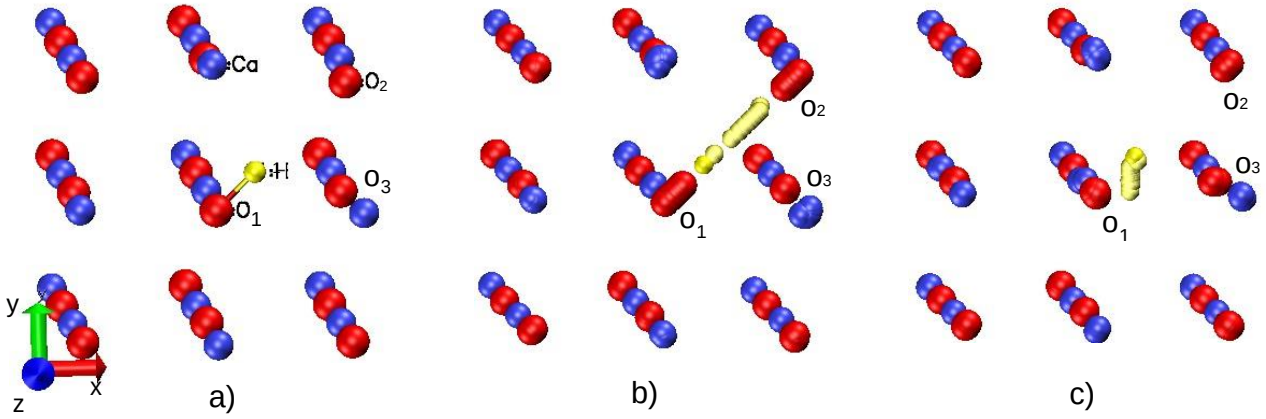


Figura 8: Estados de simulación de la supercerlda 2x2x2 de óxigo de calcio en el programa *SIESTA* para el cálculo de las barreras de energia del protón en difusión. a) Estado relajado. El protón, en amarillo, se encuentra en una posición cercana al oxígeno, en dirección a O_2. La distancia del enlace se modifica para obtener la barrera de energía para su paso a O_2. b) Movimiento del protón al oxigeno cercano O_2. c) Movimiento del protón a la zona de equilibrio más próxima en torno al mismo oxígeno. El protón se situa a la misma distancia, pero cambia su angulo para ponerse en dirección a O_3. Mientras los oxígenos procuran acercarse al protón, los calcios se alejan.

alrededor del oxígeno vecino. Es el caso descrito en la figura 7 y 8(b). Para obtener la barrera de energía, comenzando desde la posición de equilibrio a una distancia de 0.98 Å respecto el oxígeno, se ha incrementado esta distancia en 10 pasos de 0.15 Å cada uno. La segunda transición es el movimiento de rotación del protón alrededor del propio oxigeno. Para cada posición de equilibrio, 4 de las 11 posiciones restantes (1 la ocupa él mismo) son accesibles en un único paso de simulación (figura 8(c). La barrera de energía se ha obtenido, a partir de la misma posición de equilibrio anterior, modificando el valor del ángulo que el protón formaba con la horizontal, de 45º a 0º, en pasos entre 3º y 5º. Siendo en intervalo más pequeño en la zona central para una mayor precisión.

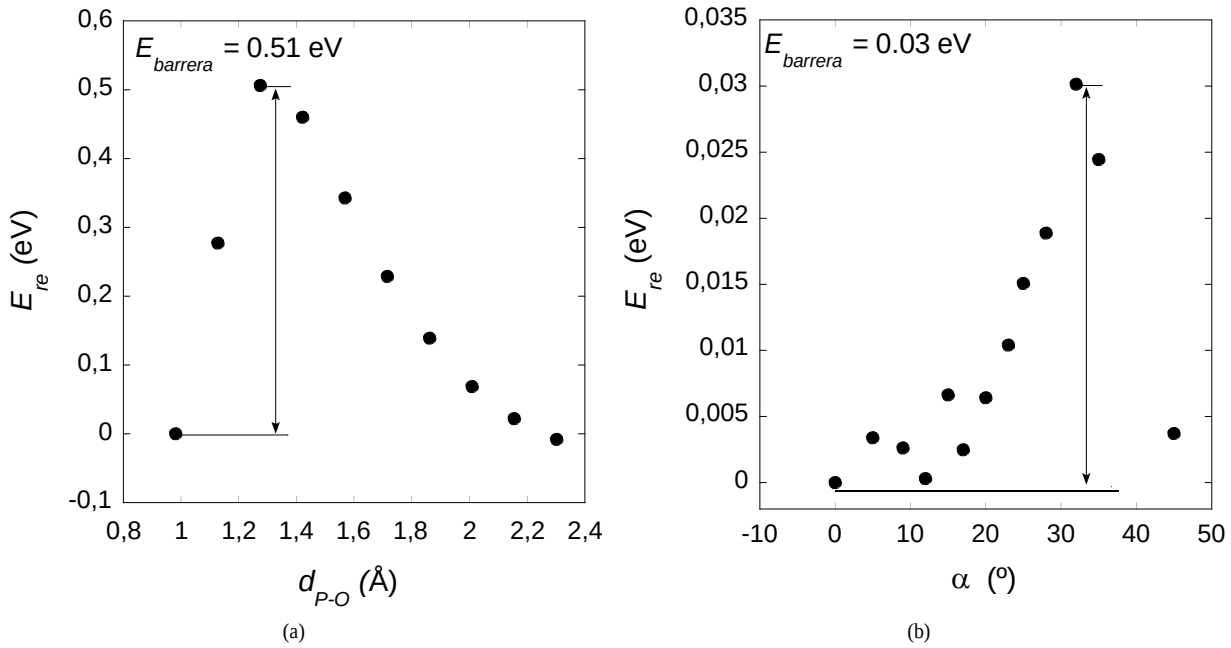
Los resultados para ambas barreras de energía pueden verse en la figura 9. Para el protón se obtienen una barrera de energía de transiciones a posiciones a oxígenos vecinos $E_{barrera} = 0{,}51\ eV$. Pese a no disponer de datos experimentales para validar la simulación, nuestros resultados son del orden de los reportados para otros óxidos iónicos. Por ejemplo, las barreras de energía de la transición del protón entre oxígenos en $CaZrO_3$ [59] varía entre 0.58 eV y 0.74 eV, siendo la distancia entre en protón y el oxígeno aceptor de $\sim$ 2.6 Å ($\sim$ 2.45 Å en nuestro caso)

Para la rotación del protón en una misma posición la barrera de energía calculada es de $E_{barrera} = 0{,}03\ eV$. Pese a ser baja, esta barrera de energía es esperada, ya que la rotación del H en un grupo OH puede tener lugar a temperatura ambiente (300 K $\sim$ 0.026 eV). Además barreras de energía de transferencia de protones del orden de 0.03 eV se han encontrado para ciertos óxidos. En el $BaZrO_3$ [63] se reportan valores del orden de los obtenidos, pero curiosamente contrarias. Mientras que en la trasferencia la barrera de energía es de 0.03 eV, en la rotación, es 10 veces mayor, de 0.3 eV, lo que pone de manifiesto la importancia de la estructura del material en la difusión.

4.2.2. Barrera de energía del litio catión

El caso del litio es considerablemente más sencillo, ya que en equilibrio el Li^+ se sitúa en el centro del hueco intersticial formado por 4 oxígenos y 4 calcios [40] (ver figuras 10 y 15). Por lo tanto, presenta una única transición, el paso de un hueco intersticial al inmediato siguiente. Hay que tener en cuenta que hay 6 posibles transiciones ya que la estructura cristalina del óxido de calcio es simétrica en las 3 direcciones del espacio.

Para el cálculo de la barrera se obtiene en primer lugar la distancia entre su posición de equilibrio y un átomo de oxígeno de la celda contigua, 4.026 Å. Posteriormente se va reduciendo esta distancia en 10 pasos de 0.2 Å,



(a) (b)

Figura 9: Energía relativa respecto a la posición de equilibrio E_{re} en cada una de las posiciones representadas en la figura 8(b) y 8(c). a) Para la transferencia de un protón de un oxígeno a otro se obtiene un valor para la barrera de energía $E_{barrera} = 0{,}51\ eV$. b) Para la rotación del protón, se obtiene $E_{barrera} = 0{,}03\ eV$

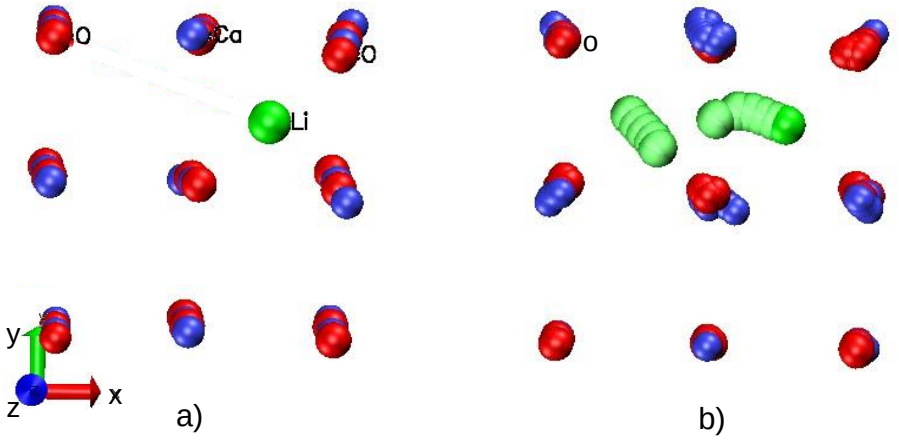


Figura 10: Estados de simulación de la supercerlda 2x2x2 de óxigo de calcio en el programa *SIESTA* para el cálculo de las barreras de energia del Li^+ en difusión. a) Estado relajado. El litio, en verde, se encuentra en una posición central. La distancia del enlace tomado respecto a un oxigeno en la celda contigua se modifica para obtener la barrera de energía para su paso a esta. b) Movimiento del Li^+ a la celda contigua. A su paso, el calcio se aleja y el oxigeno se acerca ligeramente.

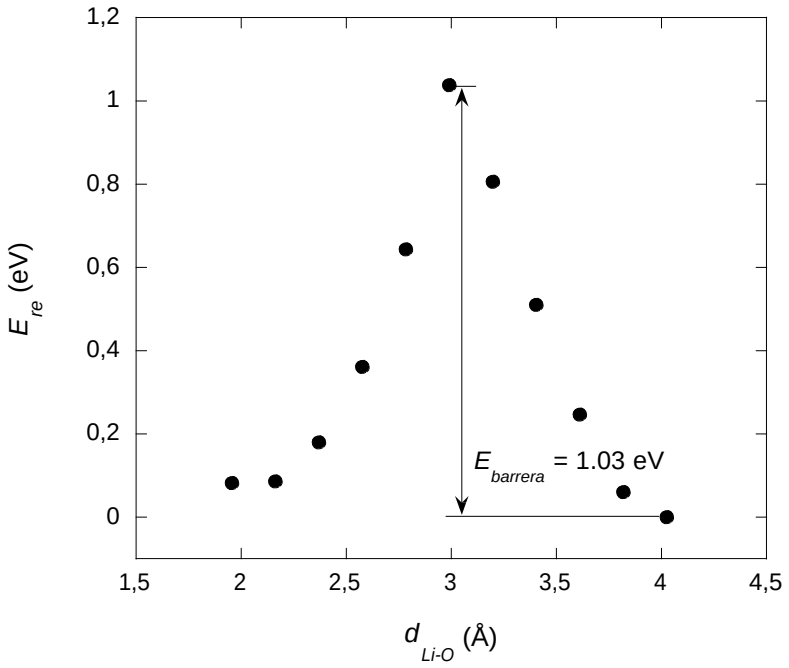


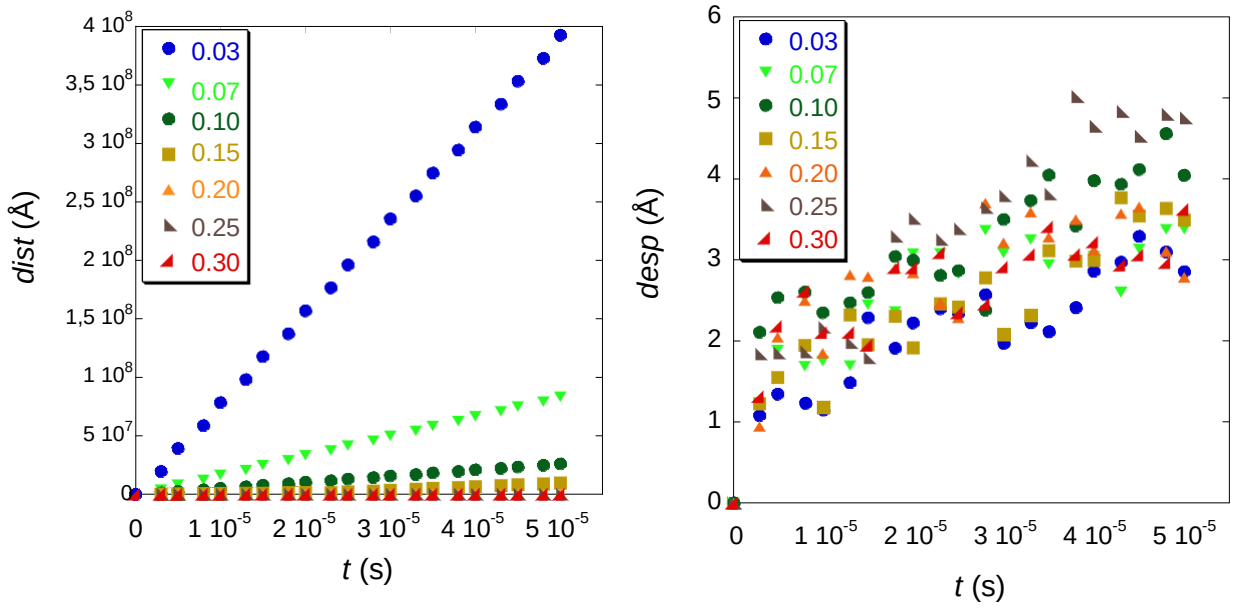
Figura 11: Energía relativa respecto a la posición de equilibrio E_{re} en cada una de las posiciones representadas en la figura 10(b). Se obtiene un valor para la barrera de energía $E_{barrera} = 1{,}03\ eV$.

obligando al Li^+ a ir al hueco intersticial vecino . Se obtiene un valor entre las transiciones intersticiales (figura 11) de $E_{barrera} = 1{,}03\ eV$. Este valor es considerablemente alto, pero del mismo orden de magnitud que en materiales usados normalmente con para el desarrollo de baterías de litio como son Li_3N, Li_2O, Li_2S, Li_2Se entre otras y el valor calculado está en torno 0.24 - 0.80 eV [64]. Por lo tanto el valor obtenido parece razonable.

4.3. Estudio de la difusión del protón en el óxido de calcio

En la sección 2.3 se habló de la pérdida de eficiencia que suponía la existencia de barreras de energía muy pequeñas en simulaciones KMC. Este es el caso que nos ocupa. La difusión del protón en el CaO debido a que su movimiento de rotación de un oxígeno es muy probable (baja barrera de energía) se repite infinidad de veces antes de que una transferencia a un oxígeno vecino, de mucha menor tasa, tenga lugar. Mientras la segunda transición da la oportunidad al protón de moverse por el material, en la primera se queda prácticamente inmóvil. El cambio significativo en el material vendrá por esta segunda transición. Como se ha descrito anteriormente, una posible solución es aumentar el valor de la barrera de energía para esta transición artificialmente. Es lógico pensar que un aumento de la barrera de energía en la primera transición no cambiará en esencia la simulación; el tiempo de salto a oxígenos vecinos seguirá siendo el mismo, y se ganará en eficiencia. No obstante, es necesario evaluar el efecto de manipular la barrera de energía en la distancia y desplazamiento, definidos a continuación.

Suponiendo una posición inicial (x_0, y_0, z_0) y transcurrido un cierto tiempo, un átomo puede cambiar de posición a (x_t, y_t, z_t). Se define la cantidad desplazamiento $desp$ como se indica en la ecuación 12. Sin embargo, para llegar a esa posición puede haber descrito una trayectoria más directa o más irregular. Se define distancia recorrida en la ecuación 13 $dist$ como la cantidad de movimiento neta invertida para llegar a ese punto. Será necesariamente igual o mayor que el desplazamiento.



(a) (b)

Figura 12: a) Distancia y b) desplazamiento medio de los cinco protones en el óxido de calcio. Una mayor barrera de energía en la transición conlleva una disminución de la distancia recorrida. Pero no así de su desplazamiento.

$$desp = \sum_{j} \sqrt{(x_t - x_0)^2 + (y_t - y_0)^2 + (z_t - z_0)^2} \tag{12}$$

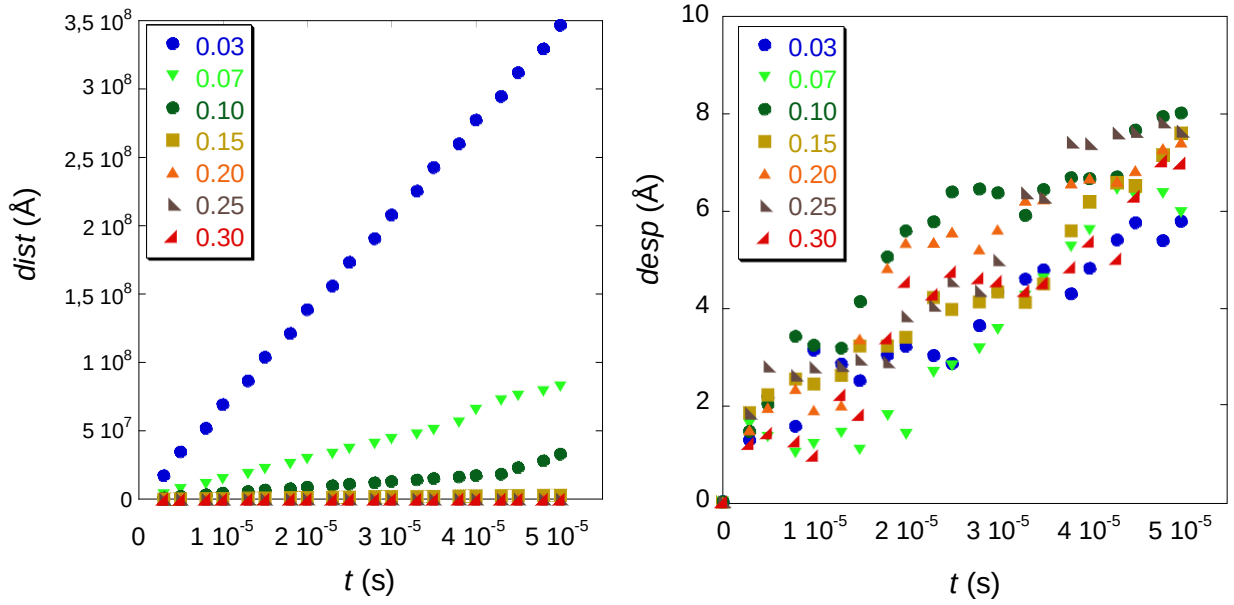
$$dist = \sum_{1}^{n} s_j \tag{13}$$

donde s_j es en desplazamiento recorrido en cada transición y n el número de pasos.

Se han realizado simulaciones con diferentes valores de barrera de energía para la rotación, observándose el comportamiento del desplazamiento de los protones dentro del cristal, tanto libres, como condicionados por un campo eléctrico en z, descrito anteriormente (sección 3.2.4). Se ha escogido un sistema de simulación de 5x5x5 celdas unitarias con los parámetros de configuración tal como se muestran anteriormente en la descripción de la cabecera `CaOH.h` (ver sección 3.2). En él, se han colocado 5 protones, siendo todos los resultados obtenidos la media de los 5 átomos. Para determinar el tiempo de simulación `tiemposimulacion`$= 0{,}00005\ s$ se ha buscado un compromiso entre que el cálculo fuera rápido, pero que a su vez diese tiempo al protón a tener varias transiciones entre oxígenos. Se obtiene tanto el valor medio de la distancia $\overline{dist}$ de los protones, como de su desplazamiento $\overline{desp}$ (figuras 12(a) y 12(b)).

Con el aumento de la barrera de energía se observa un fuerte decaimiento de la distancia neta recorrida por los protones, como era previsible. En la figura 12(b) no hay correlación entre el valor de la barrera de energía y el desplazamiento medio recorrido. Resaltar que a pesar de que se reduce significativamente el movimiento de los protones, no afecta al desplazamiento que recorren dentro del material. Por lo tanto se puede afirmar que para este sistema en concreto, que restringir el movimiento de rotación del protón alrededor del oxígeno al aumentar su barrera de energía, no interfiere en su difusión. Sin embargo, es necesario que el número de movimientos de transferencia sea bastante mayor que los producidos entre oxígenos para asegurarse un movimiento aleatorio. Un máximo en la barrera de energía de $0{,}3\ eV$ frente al $0{,}51\ eV$ lo asegura.

El mismo estudio se hace en presencia de un campo eléctrico en dirección z. Se fija un valor `IntensidadE`



(a) (b)

Figura 13: a) Distancia y b) desplazamiento medio de los cinco protones en el óxido de calcio bajo un campo eléctrico de intensidad $0{,}025\ eV/$Å en dirección z. La existencia de campo no conlleva un aumento de movimiento neto, pero al tener este movimiento preferencia en una dirección, si se observa un aumento en el desplazamiento.

$= 0{,}025\ eV/$Å. Se ha escogido este valor de intensidad de campo con vistas a que no fuera tan alto que se aumentase indeseablemente el tiempo de calculo al hacer ciertas transiciones muy probables, y sin embargo se apreciase su existencia. En las gráficas 13(a) y 13(b) se obtienen sus valores de distancia media $\overline{dist}$ y desplazamiento medio $\overline{desp}$.

Al comparar las figuras 12(a) y 13(a) se observa que la distancia neta es la misma, no hay un cambio significativo. Pese a que el protón tiene preferencia por moverse en dirección z, en promedio se mueve tanto como con ausencia de campo eléctrico. Se observa sin embargo un aumento significativo en el desplazamiento recorrido. De tener un desplazamiento medio en el último paso de simulación entre 3 y 5 Å en la figura 12(b), se obtienen con la imposición del campo en la figura 13(b) desplazamientos en un rango entre 6 y 8 Å. Este aumento se debe necesariamente a la tendencia a moverse en dirección del campo en lugar de al azar. Para estudiar este efecto, estudiamos el carácter aleatorio del movimiento de los átomos, o *random walk*, mediante la fórmula:

$$desp = D_m . \sqrt{f_s . t} \tag{14}$$

donde f_s es la frecuencia de salto del átomo y D_m es la distancia media de salto. En un cristal tridimensional donde los átomos intersticiales difunden de manera aleatoria, esta fórmula prevé la distancia recorrida respecto al tiempo [65]. En el sistema estudiado únicamente se han considerado 5 protones de los 6000 posiciones disponibles que podían ocupar. Se puede considerar entonces que estos átomos no van a interferir entre ellos y que individualmente, están aislados. En cambio, en presencia de un campo eléctrico esta relación no debería cumplirse, ya que el campo favorece el desplazamiento en una dirección.

En la figura 12(b) se intuye una dependencia de raíz cuadrada de la distancia media $\overline{dist}$ con respecto al tiempo t. Gracias al un aumento de la barrera de energía podemos simular tiempos más altos `tiemposimulacion`$= 0{,}001\ s$, y por lo tanto obtener resultados más claros, reflejado en la figura 14. Se confirma por un lado que el cumplimento del *random walk* de la ecuación 14, y por otro, la dependencia lineal que involucra la existencia de un campo eléctrico.

Considerando que la difusión viene gobernada principalmente por la transferencia entre oxígenos, el valor de la

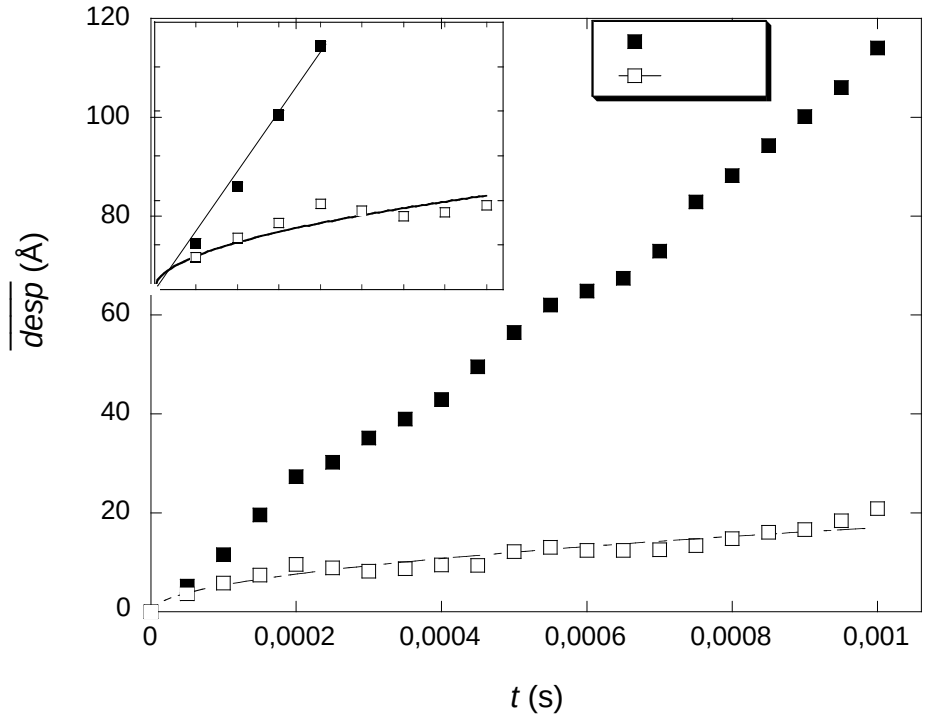


Figura 14: Desplazamiento medio de los cinco protones en el óxido de calcio libres, y bajo un campo eléctrico intensidad $0{,}025\ eV/$Å en dirección z. En el primer caso se observa el cumplimiento del *random walk*. En el segundo se favorece una dependencia lineal. La gráfica insertada es una ampliación de los primeros puntos que resalta esta dependencia. La recta únicamente tiene la función de guiar el ojo, no es un ajuste lineal.

frecuencia de salto f_s puede ser determinada a partir de la tasa de esta transición k_{ij} en la ecuación 6 con $E_{barrera} = 0{,}5\ eV$ y $T = 300\ K$.

$$f_s = 2{,}084x10^{10}.T.e^{-E_{barrera}/Kb.T} = 24902{,}85\ s^{-1} \tag{15}$$

A partir de este valor, En la figura 14 se obtiene del ajuste un valor para $D_m = 3{,}41 \pm 0{,}09$ Å el cual corresponde con la distancia entre oxígenos en el óxido de calcio (3,429 Å). Es importante señalar que esta distancia media de salto D_m también tiene en cuenta el movimiento de rotación; Un protón que acaba de saltar a otro oxígeno se desplaza alrededor de este reorientándose en otra dirección aleatoria. En promedio el protón se habrá movido la distancia de transferencia más la distancia de rotación, es decir, la distancia entre oxígenos.

4.4. Estudio de la difusión del catión del litio en el óxido de calcio

En el caso del catión del litio, un valor de 1 eV en la barrera de energía las transiciones a temperatura ambiente son poco probables, y cada evento de difusión tardaría del orden de minutos. En este caso puede ser interesante estudiar entonces la dependencia de la difusión del Li^+ en el óxido de calcio con la temperatura.

Se escoge un `tiemposimulacion`$= 2\ s$, necesariamente más alto que en el caso anterior por la alta barrera de energía. Los parámetros de simulación son los ya citados en la sección 3.2.6. El sistema de simulación, de dimensión 5x5x5, tiene la estructura representada en la figura 15. Al igual que antes se introducen en el sistema únicamente 5 cationes de litio y se comprueba el comportamiento de la difusión con un movimiento libre, y restringido por un campo eléctrico, con la misma disposición e intensidad de campo `intensidadE`$= 0{,}025\ eV/$Å.

En las figuras 16(a) y 16(b) se observa que tanto la distancia media $\overline{dist}$ como el desplazamiento medio $\overline{desp}$

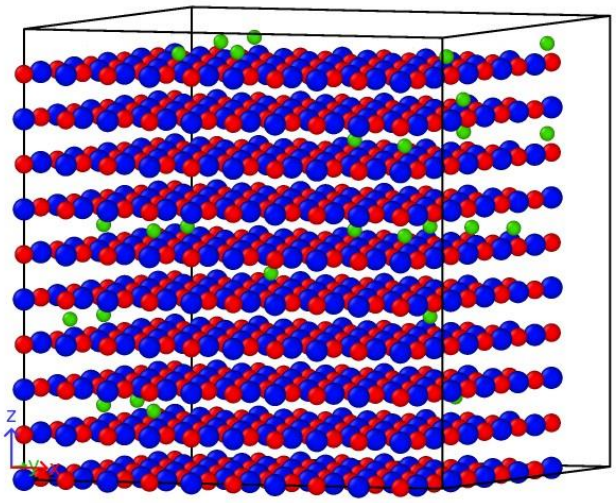


Figura 15: Sistema de simulación del litio, en verde, en el óxido de calcio.

aumentan con la temperatura. Se comprueba por un lado que la difusión es mucho menor que en caso del protón, como era de esperar, y por otro lado que se cumple la ecuación 14 en el desplazamiento medio $\overline{desp}$ respecto al tiempo t. Homólogamente al caso anterior del protón, se puede considerar que los cationes de litio están aislados en el material ya que la proporción respecto a posiciones que puede ocupar, 2000 en este caso, es muy pequeña. Por lo cual se hace un ajuste a la recta de mayor temperatura; hay mayor número de transiciones y por lo tanto se espera un menor error. Sabiendo que el valor de la frecuencia de salto f_s para esta transición con $E_{barrera} = 1{,}03\ eV$ y temperatura $T = 480\ K$:

$$f_s = 2{,}084x10^{10}.T.e^{-E_{barrera}/Kb.T} = 153{,}325\ s^{-1} \tag{16}$$

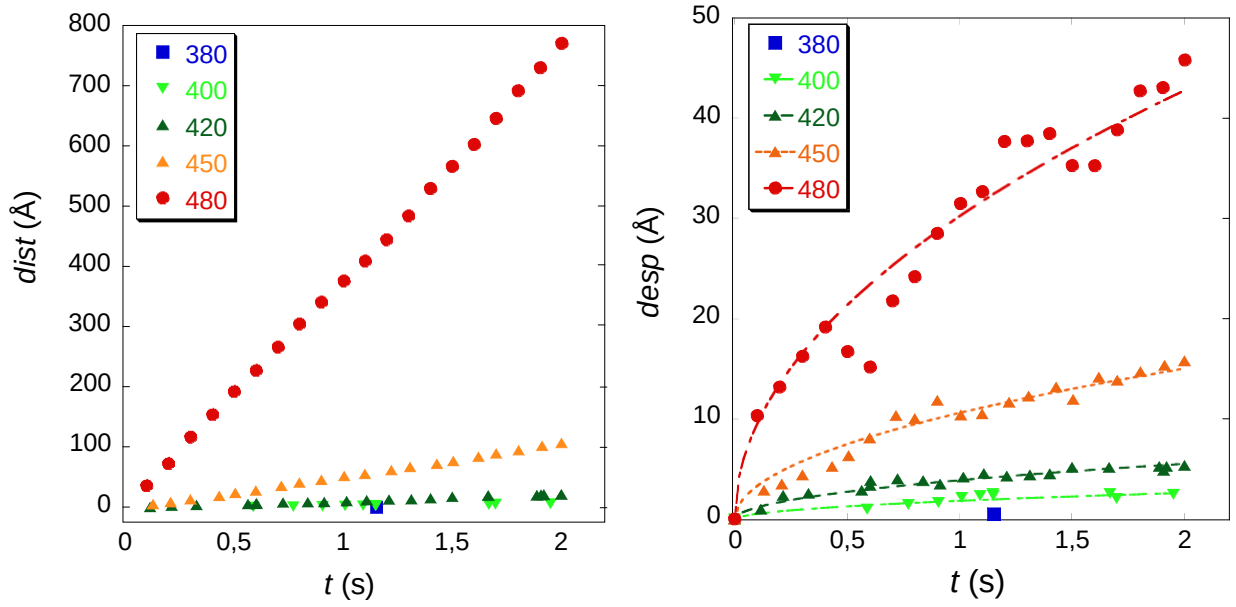
Se obtiene un valor para la distancia media de salto $D_m = 2{,}44 \pm 0{,}05$ Å. Este valor corresponde a la distancia entre cavidades intersticiales que pueden ser ocupadas por el Li^+ (2,425 Å)

En las figuras 17(a) y 17(b) se puede observar que el comportamiento del sistema al introducir un campo eléctrico es análogo al ya obtenido en el caso del protón. Al igual que antes, no se obtiene un cambio significativo en la distancia al introducir un campo. No así en el desplazamiento, que experimenta un gran aumento a temperaturas más altas. Notar que al ser la relación entre la intensidad del campo eléctrico y la barrera de energía menor, se obtiene una menor dependencia lineal en el desplazamiento.

El tiempo de cálculo depende directamente del número de transiciones que se llevan a cabo. En las simulaciones para el protón se tienen tiempos de simulación mucho mayores que para el Li^+. En el caso del protón, las simulaciones llevadas a cabo con la barrera baja de energía $E_{barrera} = 0{,}03\ eV$ suponen un tiempo de cálculo de más de un día. Este tiempo se ve reducido drásticamente al aumentar la barrera a $E_{barrera} = 0{,}3\ eV$, donde el cálculo ronda el minuto. Las simulaciones para el Li^+ rondaban los segundos.

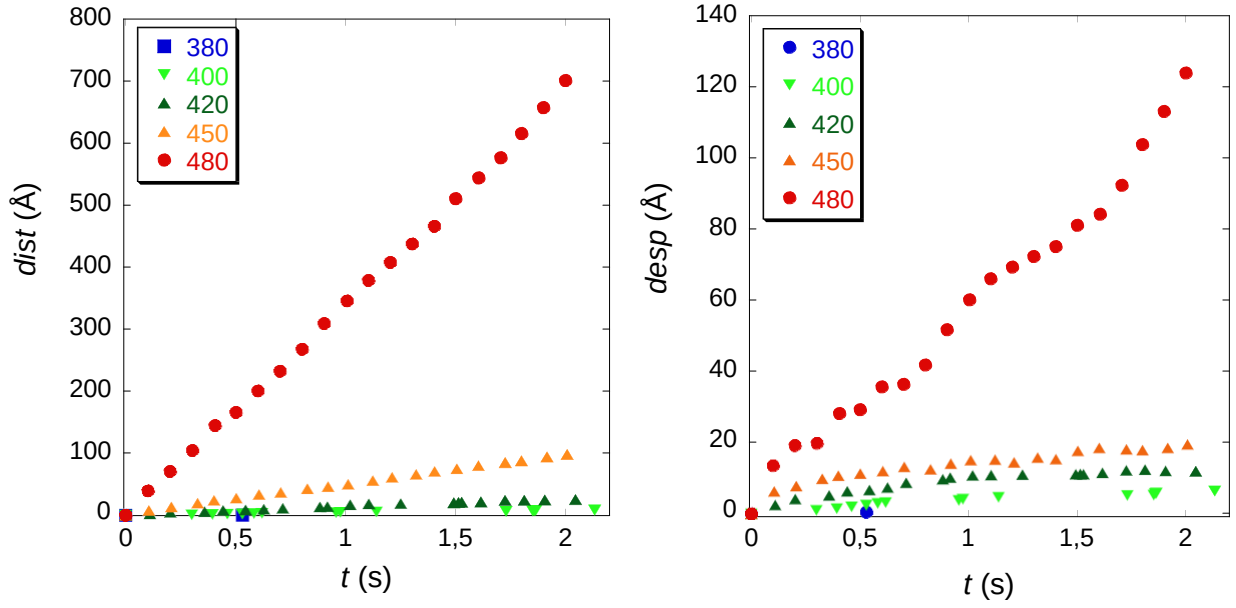
5. Conclusiones

El método KMC es una potente herramienta que permite estudiar el comportamiento de ciertas propiedades en rangos de tiempo mucho mayores que otras herramientas de simulación. Su facilidad de implementación y versatilidad permiten su aplicación en varios ámbitos de estudio. En este trabajo se ha desarrollado un código Kinetic Monte Carlo



(a) (b)

Figura 16: a) Distancia y b) desplazamiento medio de los cinco cationes de litio el óxido de calcio. El aumento de temperatura conlleva un aumento de su movimiento y también de su desplazamiento. Se aprecia una forma raíz cuadrada en la dependencia de desplazamiento medio $\overline{desp}$ frente al tiempo t, típica del *random walk*



(a) (b)

Figura 17: a) Distancia y b) desplazamiento medio de los cinco cationes de litio en el óxido de calcio bajo un campo eléctrico de intensidad $0{,}025\ eV/Å$ en dirección z. Al igual que en caso del protón la existencia de campo no conlleva un aumento de movimiento neto, pero al tener este movimiento preferencia en una dirección, si se observa un aumento en el desplazamiento.

para el estudio de diversos procesos que tienen lugar en materiales. En concreto, la difusión de átomos intersticiales tales como protones y Li^+ en el óxido de calcio. El objetivo final no obstante es ampliar el programa dotándole de flexibilidad para el estuido de un rango más amplio de problemas y propiedades. Las características más reseñables del código son:

- Pese a ser una primera versión del programa, funciona, y su eficiencia es más que razonable, debido en gran parte a la eficiencia en el método que más gasto computacional supone, la búsqueda de transciones del sistema, al emplear el algoritmo *binary search*
- El código tiene una gran portabilidad al estar escrito en el extendido lenguaje de programación C++, reproducibilidad de resultados gracias al *seed*, y es fácilmente ampliable.
- El algoritmo desarrollado de búsqueda de vecinos tiene utilidad más allá de este trabajo. Puede ser igualmente utilizado para la creación de sistemas de estudio en otras herramientas computacionales tales como *SPPARKS*.

Para probar el funcionamiento del código se ha estudiado la difusión de protones y cationes de litio en el óxido de calcio. Como conclusiones del estudio podemos resaltar:

- Las barreras de energía de las transiciones del protón y el litio catión han sido calculadas mediante simulaciones DFT usando el método *Contrain Energy Minimization*. Las barreras de energía son considerables más altas para el Li^+ que para los protones, como era de esperar. Además, son del mismo orden de las barreras de energía de esas transiciones en materiales similares.
- Se ha comprobado que, en caso de tener transiciones muy rápidas que 'bloquean' la evolución del sistema, se pueden incrementar las barreras de energía de dichas transiciones sin que se altere la física del sistema.
- Los resultados en la difusión del protón y el catión de litio en el óxido de calcio determinados en este trabajo son coherentes; responden a la ley *random walk* propia de sistemas sólidos tridimensionales y se obtienen las esperables desviaciones al aplicar un campo eléctrico.
- La difusión del Li^+ en el óxido de calcio es mucho menor que la obtenida para el protón.

Trabajo futuro

En el futuro se pretende mejorar y ampliar muchos aspectos del programa. Por ejemplo, se buscará que el sistema de simulación pueda ser leído desde un fichero externo en caso de que el usuario así lo desee. Y algo aún mas complejo, que la descripción de las transiciones que pueden tener lugar en el sistema se puedan definir externamente en un fichero, estableciendo el usuario tantas condiciones como quiera para ganar en flexibilidad. El programa se ordenará en módulos o clases diferenciadas y se aprovechará todo el potencial de la programación orientada a objetos que ofrece c++.

Se han realizado además ciertas pruebas empleando otra de la fenomenología descrita en el trabajo. En la figura 18 se observa una posible evolución compleja del óxido de calcio disolviéndose. Pese a que los valores de las barrera de energía no son los reales en los fenómenos de disolución y disociación del disolvente, se observa que el código es capaz de describir un proceso de disolución. En un futuro se estudiará el valor de estas barreras de energía para determinar una evolución realista.

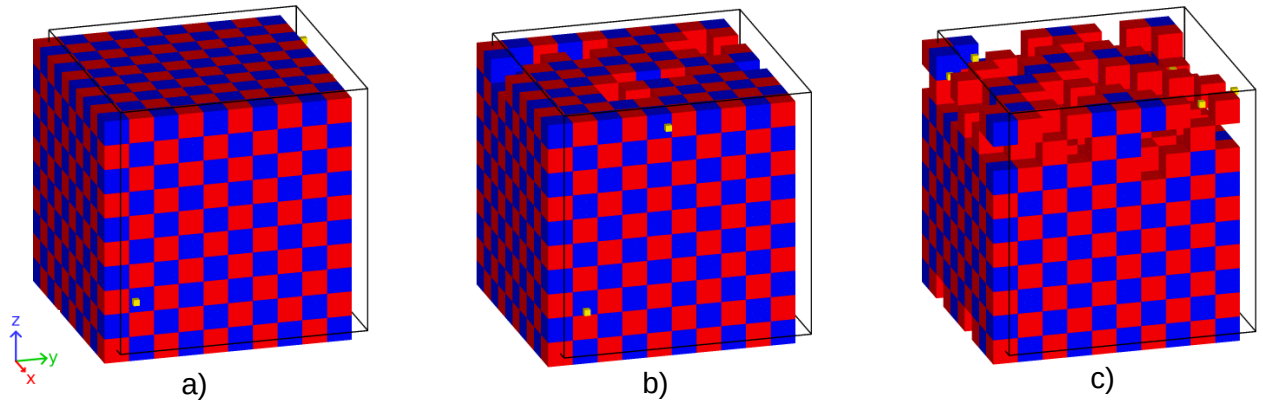


Figura 18: Prueba de simulación de la difusión del protón en el óxido de calcio, con las distintas fenomenología descritas. El sistema, que capta protones en su superficie superior, se disuelve con el tiempo a) Instante inicial b) Instante intermedio. Algunos oxigenos rodeados empiezan a disolverse en la superficie, lo que conlleva la disolución del calcio. c) Instante final. El aumento de los protones acentúa la disolución en la superficie.

Agradecimientos

Me gustaría expresar mi agradecimiento a Hegoi Manzano por la oportunidad de realizar un trabajo en el campo de la simulación, que tanto me gusta, y por su compromiso en que saliera adelante. A Eduardo Duque por resolverme tantas dudas. Y al servicio técnico proporcionado por i2basque y IZO-SGI (SGIker) de la UPV/EHU

Referencias

[1] Luis R Izquierdo, José Manuel Galán Ordax, José I Santos, and Ricardo Del Olmo Martínez. Modelado de sistemas complejos mediante simulación basada en agentes y mediante dinámica de sistemas. *Empiria. Revista de metodología de ciencias sociales*, (16):85–112, 2008.

[2] Varios Autores. Sitio web spparks. `http://spparks.sandia.gov/`, 2012.

[3] Scott M. Auerbach and Horia I. Metiu. Diffusion in zeolites via cage-to-cage kinetics: Modeling benzene diffusion in Na-Y. *The Journal of Chemical Physics*, 105(9):3753, September 1996.

[4] Maria Jose Caturla. Toward a predictive atomistic model of ion implantation and dopant diffusion in silicon. *Computational Materials Science*, 12(4):319–332, November 1998.

[5] A. Saul, G. Treglia, and B. Legrand. Kinetics of segregation and dissolution in Cu-Ag and surface phase transition: comparison between mean field and Monte Carlo calculations. *Surface Science*, 307-309:804–809, April 1994.

[6] E. van Veenendaal, P. van Beurden, W. J. P. van Enckevort, E. Vlieg, J. van Suchtelen, and M. Elwenspoek. Monte Carlo study of kinetic smoothing during dissolution and etching of the Kossel (100) and silicon (111) surfaces. *Journal of Applied Physics*, 88(8):4595, October 2000.

[7] Nicholas Metropolis and Stanislaw Ulam. The monte carlo method. *Journal of the American statistical association*, 44(247):335–341, 1949.

[8] Nicholas Metropolis. The beginning of the monte carlo method. *Los Alamos Science*, 15(584):125–130, 1987.

[9] WM Young and EW Elcock. Monte carlo studies of vacancy migration in binary ordered alloys: I. *Proceedings of the Physical Society*, 89(3):735, 1966.

[10] Alfred B Bortz, Malvin H Kalos, and Joel L Lebowitz. A new algorithm for monte carlo simulation of ising spin systems. *Journal of Computational Physics*, 17(1):10–18, 1975.

[11] Shannon D. Piersall and James B. Anderson. Direct Monte Carlo simulation of chemical reaction systems: Simple bimolecular reactions. *The Journal of Chemical Physics*, 95(2):971, 1991.

[12] Tamás Vicsek and Fereydoon Family. Dynamic Scaling for Aggregation of Clusters. *Physical Review Letters*, 52(19):1669–1672, May 1984.

[13] Corbett C Battaile and David J Srolovitz. Kinetic monte carlo simulation of chemical vapor deposition. *Annual Review of Materials Research*, 32(1):297–319, 2002.

[14] Richard Weinkamer, Peter Fratzl, Himadri S. Gupta, Oliver Penrose, and Joel L. Lebowitz. Using Kinetic Monte Carlo simulations to study phase separation in Alloys. *Phase Transitions*, 77(5-7):433–456, May 2004.

[15] RM Ziff, E Gulari, and Y Barshad. Kinetic phase transitions in an irreversible surface-reaction model. *Physical review letters*, 56(24):2553–2556, June 1986.

[16] Arthur F Voter. Introduction to the kinetic monte carlo method. In *Radiation Effects in Solids*, pages 1–23. Springer, 2007.

[17] Luis Rincón. Introducción a los procesos estocásticos. *Departamento de de Matemáticas, Facultad de Ciencias UNAM*, pages 27–39, 2008.

[18] Paul Meakin and Kevin M Rosso. Simple kinetic Monte Carlo models for dissolution pitting induced by crystal defects. *The Journal of chemical physics*, 129(20):204106, November 2008.

[19] Fedwa El-Mellouhi, Normand Mousseau, and Laurent J. Lewis. Kinetic activation-relaxation technique: An off-lattice self-learning kinetic Monte Carlo algorithm. *Physical Review B*, 78(15):153202, October 2008.

[20] Susan Louise Brantley, James David Kubicki, and Art F White. *Kinetics of water-rock interaction*, volume 168. Springer, 2008.

[21] A.C. Lasaga. Transition state theory. *Rev. Mineral.; (United States)*, 8, January 1981.

[22] Shikha Nangia and Barbara J Garrison. Advanced monte carlo approach to study evolution of quartz surface during the dissolution process. *Journal of the American Chemical Society*, 131(27):9538–9546, 2009.

[23] Peter J Feibelman. Diffusion path for an al adatom on al (001). *Physical review letters*, 65(6):729, 1990.

[24] James L Blue, Isabel Beichl, and Francis Sullivan. Faster monte carlo simulations. *Physical Review E*, 51(2):R867, 1995.

[25] Tim P Schulze. Efficient kinetic monte carlo simulation. *Journal of Computational Physics*, 227(4):2455–2462, 2008.

[26] Varios Autores. Tutorial c++. `http://www.cplusplus.com/doc/tutorial/`, 2015.

[27] Michael Kerrisk. Tutorial linux. `http://man7.org/linux/man-pages/`, 2015.

[28] Varios Autores. Sublime text. www.sublimetext.com, 2015.

[29] Política Lingüistica y Cultura del País Vasco Departamento de Educación. i2basque. http://www-es.i2basque.es/index.php/Portada, 2015.

[30] Universidad del País Vasco. Servicio general de informática de la upv. http://www.ehu.eus/sgi/, 2015.

[31] Alexander Stukowski. Visualization and analysis of atomistic simulation data with OVITO–the Open Visualization Tool. *Modelling and Simulation in Materials Science and Engineering*, 18(1):15012, 2010.

[32] William Humphrey, Andrew Dalke, and Klaus Schulten. VMD: visual molecular dynamics. *Journal of molecular graphics*, 14(1):33–38, 1996.

[33] Koichi Momma and Fujio Izumi. *VESTA*: a three-dimensional visualization system for electronic and structural analysis. *Journal of Applied Crystallography*, 41(3):653–658, Jun 2008.

[34] Paul Allen Tipler and Gene Mosca. *Física para la ciencia y la tecnología*, volume 2. Reverté, 2005.

[35] Benjamin Jun and Paul Kocher. The intel random number generator. *Cryptography Research Inc. white paper*, 1999.

[36] Dr Svend Haahr Dr Mads Haahr. Random number generator. https://www.random.org/, 2015.

[37] André Stefanov, Nicolas Gisin, Olivier Guinnard, Laurent Guinnard, and Hugo Zbinden. Optical quantum random number generator. *Journal of Modern Optics*, 47(4):595–598, 2000.

[38] Stephan Müller. Cpu time jitter based non-physical true random number generator. pages 23–47, 2014.

[39] Varios Autores. Ncomputers. http://ncomputers.org/projects/random, 2015.

[40] Alexander Urban, Jinhyuk Lee, and Gerbrand Ceder. The configurational space of rocksalt-type oxides for high-capacity lithium battery electrodes. *Advanced Energy Materials*, 4(13), 2014.

[41] Frederick C Brown. *Física de los Sólidos*. Reverté, 1970.

[42] William F Smith. Fundamento de la ciencia e ingeniería de materiales. In *Fundamento de la ciencia e ingeniería de materiales*. McGraw-Hill, 1996.

[43] Harry D Moore and Donald R Kibbey. Materiales y procesos de fabricación. *Industria metal mecánica y de plásticos*, 1996.

[44] Harvey J Goldschmid. *Interstitial alloys*. Springer, 2013.

[45] Nancy J Dudney, William C West, and Jagjit Nanda. *Handbook of Solid State Batteries*, volume 6. World Scientific, 2015.

[46] Asociación Nacional de Fabricantes de Cales y Derivados de España. *La Cal y sus aplicaciones*. ANCADE, 2012.

[47] Jaime Alvar Ezquerra. Theron, rex hispaniae citerioris (macr., sat. i, 20, 12). *Gerión*, (4):161–176, 1986.

[48] Jesús Díaz Minguela and Ángel Sampedro Rodríguez. La adecuada elección del conglomerante en la estabilización de suelos.

[49] Xuejun Liu, Huayang He, Yujun Wang, Shenlin Zhu, and Xianglan Piao. Transesterification of soybean oil to biodiesel using CaO as a solid base catalyst. *Fuel*, 87(2):216–221, February 2008.

[50] J. Calero, Diego Luna, Enrique Sancho Puebla, Carlos Luna, G. Cumplido, Alejandro Posadillo, F.M. Bautista, A.A. Romero, and Cristóbal Verdugo. Síntesis de un nuevo biocombustible que integra la glicerina empleando óxido de calcio como catalizador heterogéneo, 2013.

[51] YJ Mendoza-Chávez, NA Medellín-Castillo, R Ocampo-Pérez, SE Escoto-Chávez, MC Maza-Moheno, J Morales-Rueda, and JF Toro-Vázquez. Síntesis de biodiésel mediante catálisis heterogénea. *Facultad de Ingeniería*, page 39.

[52] Hegoi Manzano, Roland JM Pellenq, Franz-Josef Ulm, Markus J Buehler, and Adri CT van Duin. Hydration of calcium oxide surface predicted by reactive force field molecular dynamics. *Langmuir*, 28(9):4187–4197, 2012.

[53] Gunnar Wettermark. Thermochemical energy storage. In *Energy Storage Systems*, pages 673–681. Springer, 1989.

[54] Ronald Barker. The reactivity of calcium oxide towards carbon dioxide and its use for energy storage. *Journal of Applied Chemistry and Biotechnology*, 24(4-5):221–227, April 2007.

[55] F Schaube, L Koch, A Wörner, and H Müller-Steinhagen. A thermodynamic and kinetic study of the de-and rehydration of ca (oh) 2 at high h 2 o partial pressures for thermo-chemical heat storage. *Thermochimica acta*, 538:9–20, 2012.

[56] Robert G Parr and Weitao Yang. *Density-functional theory of atoms and molecules*, volume 16. Oxford university press, 1989.

[57] W. F. Van Gunsteren and M. Karplus. A method for constrained energy minimization of macromolecules. *Journal of Computational Chemistry*, 1(3):266–274, 1980.

[58] Céline Chizallet, Guylène Costentin, Michel Che, Françoise Delbecq, and Philippe Sautet. Revisiting acido-basicity of the mgo surface by periodic density functional theory calculations: Role of surface topology and ion coordination on water dissociation. *The Journal of Physical Chemistry B*, 110(32):15878–15886, 2006.

[59] M. Saiful Islam, R. Andrew Davies, and Julian D. Gale. Proton Migration and Defect Interactions in the CaZrO 3 Orthorhombic Perovskite: A Quantum Mechanical Study. *Chemistry of Materials*, 13(6):2049–2055, June 2001.

[60] José M Soler, Emilio Artacho, Julian D Gale, Alberto García, Javier Junquera, Pablo Ordejón, and Daniel Sánchez-Portal. The siesta method for ab initio order-n materials simulation. *Journal of Physics: Condensed Matter*, 14(11):2745, 2002.

[61] Hegoi Manzano, Jorge S Dolado, and Andres Ayuela. Structural, mechanical, and reactivity properties of tri-calcium aluminate using first-principles calculations. *Journal of the American Ceramic Society*, 92(4):897–902, 2009.

[62] JP Perdew, K Burke, and M Ernzerhof. Phys rev lett 77: 3865. *Errata:(1997) Phys Rev Lett*, 78:1396, 1996.

[63] Paolo Raiteri, Giovanni Bussi, and Julian D Gale. Reactive force field for proton diffusion in bazro3. Technical report, 2011.

[64] Yukinori Koyama, Yasuhiro Yamada, Isao Tanaka, Shigeto R Nishitani, Hirohiko Adachi, Masahiro Murayama, and Ryoji Kanno. Evaluation of migration energy of lithium ions in chalcogenides and halides by first principles calculation. *Materials Transactions*, 43(7):1460–1463, 2002.

[65] Paul Heitjans and Jörg Kärger. *Diffusion in Condensed Matter: Methods, Materials, Models.* 2006.

Apendice.Formato de salida del programa

Formato visualizable por *OVITO*. Se encarga de su impresión el método `imprimeestado`.

```
ITEM: TIMESTEP
<paso>  <tiempo>

ITEM: NUMBER OF ATOMS
<numero de atomos>

ITEM: BOX BOUNDS\n");
<limite inferior celda>         <limite superior celda>
<limite inferior celda>         <limite superior celda>
<limite inferior celda>         <limite superior celda>

ITEM: ATOMS id i2 x y z <otras variables>
<id>    <tipo>  <X>     <Y> <Z>
<id>    <tipo>  <X>     <Y> <Z>
<id>    <tipo>  <X>     <Y> <Z>
...
```

Por ejemplo

```
ITEM: TIMESTEP
0          0
ITEM: NUMBER OF ATOMS
7000
ITEM: BOX BOUNDS
0 24.25
0 24.25
0 24.25
ITEM: ATOMS id i2 x y z
1 1 0 0 0
2 1 0 0 4.85
3 1 0 0 9.7
4 1 0 0 14.55
5 1 0 0 19.4
6 1 0 4.85 0
7 1 0 4.85 4.85
...
ITEM: TIMESTEP
1          0.400706
ITEM: NUMBER OF ATOMS
7000
ITEM: BOX BOUNDS
0 24.25
0 24.25
0 24.25
ITEM: ATOMS id i2 x y z
1 1 0 0 0
2 1 0 0 4.85
3 1 0 0 9.7
4 1 0 0 14.55
5 1 0 0 19.4
6 1 0 4.85 0
7 1 0 4.85 4.85
...
```