UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

Facultad de Química

Implementación semi-automatizada del método modificado de duplicación de J en experimentos de 2D de RMN

QUE PARA OBTENER EL TÍTULO DE

Q U Í M I C O

P R E S E N T A

Amed Muñoz Ramos

México, D. F. 2010

Jurado asignado:

Presidente:	Liliana Virginia Raquel Saldivar y Osorio
Vocal:	José Federico del Río Portilla
Secretario:	Araceli Patricia Peña Alvarez
1^{er} Suplente:	Rosa Isela del Villar Morales
2^{do} Suplente:	Martín Daniel Trejo Valdez

Lab. 1 de Bioquímica del Instituto de Química

Amed Muñoz Ramos

Dr. Federico del Río Portilla
Tutor

Las matemáticas son el alfabeto con el cual Dios ha escrito el Universo.

Galileo Galilei (1564-1642)

Físico y astrónomo italiano

Toda la historia del progreso humano se puede reducir a la lucha de la ciencia contra la superstición.

Gregorio Marañón (1887-1960)

Médico endocrino, científico, historiador, escritor y pensador español

Sería posible describir todo científicamente, pero no tendría ningún sentido; carecería de significado el que usted describiera a la sinfonía de Beethoven como una variación de la presión de la onda auditiva.

Albert Einstein (1879–1955)

Físico de origen alemán, nacionalizado posteriormente suizo y estadounidense

Implementación semi-automatizada del método modificado de duplicación de J en experimentos de 2D de RMN

por **Amed Muñoz Ramos**

tutoría del **Dr. José Federico del Río Portilla**

Versión 0.1 al 7 de octubre de 2010

Resumen

La información que proporcionan las constantes de acoplamiento acerca de la estructura y conformación de las moléculas, así como de la distribución y geometría de los enlaces químicos [4], es una de las cinco piezas de un espectro de RMN que debería ser considerada. En la práctica, sin embargo, el cálculo de su valor puede no resultar una tarea fácil. Uno de los diversos métodos para facilitar dicha tarea es el método modificado de duplicación de J, útil cuando hay translape en el multiplete de investigación.

A pesar de que puede extraerse mucha información de espectros de RMN de 1-D haciendo un análisis detallado, experimentos COSY adecuadamente ajustados pueden ser el único método para conocer el valor de constantes de acoplamiento del orden de 0.1 a 0.5 Hz, en los que los picos podrían no estar resueltos debido al ancho de línea [8].

El método de duplicación de J modificado se implementó en el dominio de las frecuencias para su uso en espectros de una y dos dimensiones. Una vez seleccionado el multiplete de estudio, $f(x)$, la señal es convolucionada con una función denominada δ, $g(x)$, que tiene n valores distintos de cero a intervalos iguales y del tamaño de una constante de acoplamiento de ensayo J^*. Cuando $J^* = J_{IS}$, es decir, cuando la constante de ensayo es igual al desdoblamiento real, la convolución $(f * g)(x)$ (denominada *de*convolución dependiendo de la forma de la función δ[7]) resulta un multiplete más simplificado, reducido en el valor de la constante de acoplamiento [1].

La implementación del método en un programa de computadora pretende ser una herramienta útil que permita realizar el cálculo de constantes de acoplamiento de manera rápida y sencilla. Al tener su interfaz

un esquema basado en web, permite manipular de manera gráfica los espectros de resonancia magnética nuclear (RMN) de una o dos dimensiones, sin importar la plataforma del usuario (sistema operativo, navegador de páginas web).

Se empleó el programa *nuup* para calcular las constantes de acoplamiento de la toxina denominada *Tttx1* en un experimento TOCSY.

Agradecimientos

El decir gracias a las personas que han hecho posible este trabajo es una tarea difícil de realizar: simplemente no existen palabras o hechos que puedan retribuírles todo su apoyo, que además ha sido incondicional. La primera persona que me viene a la mente es, por supuesto, mia madre. Gracias Blanca Imelda por estar allí siempre. Siento que la tesis es más tuya que mia, ¿sabes? Sin ti este proyecto, y no me refiero sólo a la tesis, sino a toda la carrera no habría sido posible (como muchos otros proyectos).

Fede... tampoco encuentro palabras suficientes para decirte gracias. Ya no sé cuánto tiempo ha pasado desde aquella clase de espectroscopía en la que tu forma de ser y tu pasión por el conocimiento me hicieron involucrarme más con tus proyectos de investigación. Durante todo este tiempo y gracias a ti, principalmente, he tenido la oportunidad de aprender mil cosas y disfrutado de muchas otras. También me parece que el que esta tesis salga finalmente a la luz es por causa tuya.

En todo este tiempo en el laboratorio de Bioquímica 1, en el Instituto de Química, la vida me ha regalado la compañía de muchas personas valiosas. Sin que haya un orden en particular, gracias Belén, David, Alma, Enrique, Héctor, Adriana, Ernesto Sánchez, Ernesto Ladrón, Angélica... gracias por las charlas, las comidas, cada momento compartido con ustedes.

Y aunque al final no fuiste parte directa de este trabajo, gracias Luis Brieba por tus conocimientos en la producción de la *Tttx1* y todo lo que eso representa.

Muy especiales gracias Alma, Angélica, por los espectros proporcionados.

Quiero agradecerte a ti en especial, Paulina, porque gran parte de la tesis fue escrita en esos meses de

aislamiento en tu depa. Gracias también a la gente de la Torre de Ingeniería por los minutos otorgados para la afinación de detalles y por los recursos.

Gracias a la vida por todo lo aprendido durante el tiempo que duró este trabajo, por las experiencias vividas, y por la compañía de la gente que estuvo conmigo en el proceso. Incluso en el final final, cuando esto fue escrito, gracias por alegrarse conmigo, Julio, Guadalupe, Lucía, Vicente (¿que querías estar en los agradecimientos? ¡Pa'que no te quejes, ahí estás!), Pilar, ¡y por supuesto tú, Ana! (que no por estar al final final, eres la menos importante.)

Finalmente, ¡gracias a ti que estás leyendo este trabajo! ¿Será que te será útil? ¿Faltó tu nombre en la lista? Lo siento, mi memoria no da para tantos nombres, pero siéntete libre de ponerte aquí :D

Gracias a las becas 59297 de CONACYT y PAPIIT IN205110 por el apoyo invaluable otorgado.

Índice general

Índice de figuras

Capítulo 1

Introducción

En la interpretación de espectros de resonancia magnética nuclear (RMN), la información sobre la molécula se obtiene de los desplazamientos químicos, las constantes de acoplamiento, los tiempos de relajación (T_1), los NOE's y finalmente, información acerca de la dinámica e intercambio atómico. Toda esta información debería ser considerada en la elucidación de la estructura y conformación de una molécula, en conjunto con el conocimiento químico de ella. Es, sin embargo, la creencia de muchos químicos que los NOE's son "el todo por el todo" en la resolución de moléculas pequeñas, así como la aparente disminución de su habilidad en interpretar espectros relativamente simples, entre otros factores, lo que ha llevado la ignorar información estructural y conformacional valiosa que ofrecen los desplazamientos químicos y las constantes de acoplamiento. [8]

Cuando un desdoblamiento está bien resuelto, el medir directamente la constante de acoplamiento entre los picos es una tarea relativamente sencilla. Las dificultades se presentan cuando el multiplete en investigación está pobremente resuelto. Para un doblete en fase, el desdoblamiento aparente es más pequeño que la constante de acoplamiento debido al translape. Para un doblete en anti fase, el desdoblamiento aparente se vuelve más grande que la constante de acoplamiento, debido a la cancelación mutua de las señales en la región de traslape.

La implementación del método modificado de duplicación de J pretende ser una herramienta útil que permita realizar el cálculo de constantes de acoplamiento de manera rápida y sencilla, en especial cuando se tienen multipletes pobremente resueltos. Se implementó un programa que pudiera ser empleado por un gran número de personas, el cual tiene un esquema de programa basado en web debido a las ventajas que este tipo de programas como la posibilidad de tener una interfaz gráfica que permita la manipulación los espectros de RMN de una o dos dimensiones de manera sencilla, y la independencia de la plataforma del operador (sistema operativo, navegador web).

Capítulo 2

Antecedentes

2.1. Resonancia Magnética Nuclear

La espectroscopia de resonancia magnética nuclear (RMN) está fundada en las propiedades magnéticas de los núcleos atómicos. La interacción del momento angular del espín nuclear con un campo magnético externo, lleva a los núcleos a experimentar una transición de un estado de menor a otro de mayor energía. Al medir la energía de dicha transición, se obtienen valores discretos debido a que la energía de los núcleos está cuantizada, es decir que éstos se encuentran en niveles energéticos específicos. Estas transiciones pueden estimularse empleando un transmisor de frecuencias electromagnéticas, y la absorción de esta energía puede ser detectada y representada como una línea espectral (ver figura 2.1.)

2.2. Espectroscopía de RMN

Como en todas las formas de espectroscopía, un espectro de RMN es una gráfica de intensidad de absorción (o emisión) en el eje vertical, contra la frecuencia en el eje horizontal. El intervalo de frecuencias en los espectros de RMN está entre los 10 y los 800 MHz, que corresponde a longitudes de onda de los 30 m hasta los 40 cm. Sin embargo, es usual que en el eje horizontal se utilice 'ppm' (partes por millón) en lugar de

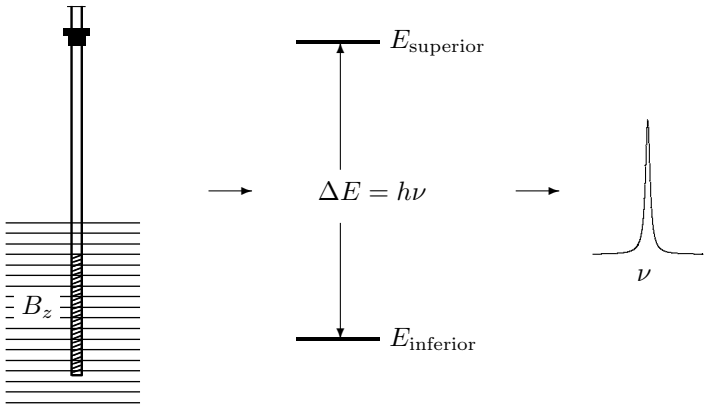


Figura 2.1: Descripción básica de una señal de RMN. La energía, dada por $h\nu$, coincide con la separación entre los niveles $E_{\text{superior}} - E_{\text{inferior}}$. El resultado es la línea de absorción del espectro, a la frecuencia ν.

frecuencia, que es la escala del desplazamiento químico. La razón de usar esta escala es que la frecuencia de las líneas espectrales es proporcional al campo magnético aplicado, de modo que al aumentar al doble la densidad del flujo magnético, aquella también aumenta al doble. Al usar una escala independiente del campo, es posible comparar las frecuencias de absorción entre espectrómetros de diferente densidad de flujo magnético.

2.2.1. Escala de desplazamientos químicos

Para construir esta escala es necesario contar con un *compuesto de referencia* cuya frecuencia de absorción se define como el cero de la escala. Esto es debido a la gran variabilidad del desplazamiento químico. El compuesto de elección es arbitrario, pero es común seleccionar uno que sea inerte y estándar para fines de comparación. Expresado matemáticamente, la escala se define como:

$$\delta = \frac{\nu - \nu_{\text{ref}}}{\nu_{\text{ref}}} \times 10^6 \text{ [ppm]} \tag{2.1}$$

2.2.2. Líneas espectrales

En un espectro de RMN se pueden observar líneas a diferentes valores de frecuencia debido a que los núcleos en las moléculas se encuentran en un ambiente químico diferente. Estas señales de resonancia están separadas en lo que se conoce como *desplazamiento químico*. Cuando estas líneas están a suficiente distancia unas de otras como para distinguirlas como un solo pico, se dice que el espectro está bien resuelto.

Dos características importantes en la interpretación de espectros de RMN son el ancho y la forma de las líneas espectrales. Es común que estas líneas sean delgadas en moléculas pequeñas, de unos cuantos Hz. Si dos núcleos tienen ambientes químicos muy similares, las líneas espectrales estarán tan cerca que se traslaparán hasta el grado en que será imposible distinguir dos líneas definidas. En el caso límite en que dos núcleos tengan el mismo desplazamiento químico, el área bajo la curva será del doble que cuando se trata de uno solo. Es decir que esta *integración* del espectro proporciona información acerca de la cantidad de núcleos involucrados en la señal.

2.2.3. Acoplamiento escalar

No todas las líneas que se observan en un espectro son simples, es decir, lo que se conoce como *singuletes*. En algunos casos se observan patrones de desdoblamiento característicos que forman *tripletes* o *cuartetos*. Estos desdoblamientos de las señales se deben a una interacción magnética entre núcleos diferentes, está mediado por los enlaces químicos, y se denomina *acoplamiento escalar* o acoplamiento. A la magnitud de esta interacción se le conoce como *constante de acoplamiento*, J, y depende del tipo y número de enlaces que separa a los núcleos involucrados, por lo que es una herramienta en la elucidación del arreglo espacial que guardan los átomos entre sí en una molécula.

2.2.4. Sistemas de espín

Se le llama "sistema de espín" a un grupo de n núcleos de número cuántico $I = 1/2$ que se caracteriza por no más de n frecuencias de resonancia (ν_i) y $n(n-1)/n$ constantes de acoplamiento (J_{ij}). Este grupo

no interactúa magnéticamente con ningún otro grupo de núcleos. Los núcleos con igual desplazamiento químico se etiquetan con la misma letra mayúscula y el número de dicho núcleo se indica con un subíndice. De este modo, los protones de un grupo metilo forman un sistema de espín A_3, mientras que los de un grupo etilo constituyen un sistema A_3B_2. La posición relativa de núcleos diferentes en un sistema de espín se indican con la posición en el alfabeto de las letras utilizadas para etiquetarlos. Para un grupo CH_3CF_2—, se usa A_3X_2 para señalar la gran diferencia en los desplazamientos químicos de los núcleos de hidrógeno y de los de flúor. Núcleos magnéticamente no equivalentes se etiquetan con apóstrofes, de este modo, el 1,1-difluoroetileno se etiqueta como AA'XX', y el 1,2-diclorobenceno como AA'BB'.

2.3. Mecánica cuántica

La teoría de la mecánica cuántica es, hasta nuestros días, una poderosa herramienta que nos proporciona, en esencia, una descripción completa del mundo microscópico. El hecho de que podamos observar distintas líneas en un espectro de RMN demuestra que la energía de un sistema de espín en un campo magnético está *cuantizado*. Uno de los postulados fundamentales de la mecánica cuántica establece que, cuando se mide la energía de un sistema, el valor que obtendremos corresponderá *siempre* a uno de los niveles de energía. Sin embargo, *no dice* enfáticamente que el sistema debe estar en uno de esos niveles energéticos.

La espectroscopía es un modo de medir la *diferencia* entre dos niveles energéticos, y no la energía de tales niveles.

2.3.1. Funciones de onda

Una *función de onda* es una función matemática que contiene la descripción completa de un sistema: si se conoce la función de onda, se puede extraer de ella aquello que queremos saber, tal como la posición de una partícula o su energía.

2.3.2. Operadores

Un operador es un artefacto matemático que actúa sobre una función para producir a una nueva función. Por ejemplo, $\hat{H}\psi$ significa que la operación definida por el operador $\hat{H}$ será ejecutada sobre la función de onda ψ. Los operadores son importantes en la mecánica cuántica debido a que representan "observables", es decir, cosas que pueden ser medidas, tales como la energía.

Se acostumbra utilizar el acento circunflejo ($\hat{}$) sobre las letras que representan un operador.

2.3.3. Funciones y valores propios

Al aplicar un operador sobre cierto tipo de funciones, el resultado es la misma función multiplicada por una constante. Se dice que estas funciones son *propias* de ese operador; el valor de la constante es denominado *valor propio*. (También suele hacerse referencia a estos dos términos como *eigenfunciones* y *eigenvalores* debido a su origen en alemán *Eigenfunktionen* o *Eigenzustände* y *Eigenwerte*.)

La relación entre funciones y valores propios de un operador, denominada *ecuación propia*, es:

$$\hat{O}\psi = A\psi$$

donde $\hat{O}$ es un operador cualquiera actuando sobre la función ψ.

> *Cuando se mide el valor de un observable, el resultado será* siempre *uno de los* valores propios *del operador que representa dicho observable.*

En el ejemplo anterior, la constante A es el valor medido del observable representado por el operador $\hat{O}$ al ser aplicado a la función ψ, la cual a su vez, contiene toda la información del sistema. El operador $\hat{O}$ y la función ψ pueden ser números complejos, pero la constante A debe ser un número real.

> *En la mecánica cuántica, los operadores tienen únicamente valores propios reales.*

Una consecuencia de que los valores propios deban ser reales es que las funciones propias satisfacen la

condición:

$$\int_{-\infty}^{\infty} \psi_m^*(x)\psi_n(x)dx = 0 \qquad m \neq n \tag{2.2}$$

y se dice que las funciones son *ortogonales* entre sí.

Cuando una función propia satisface la condición:

$$\int_{-\infty}^{\infty} \psi_m^*(x)\psi_n(x)dx = 1 \qquad m = n \tag{2.3}$$

se dice que está *normalizada*.

Cuando un conjunto de funciones son ortogonales entre sí y que están normalizadas se dice que es un conjunto *ortonormal*. La condición de ortonormalidad se puede escribir como:

$$\int_{-\infty}^{\infty} \psi_m^*(x)\psi_n(x)dx = \delta_{mn} \tag{2.4}$$

donde

$$\delta_{mn} = \begin{cases} 1 & m = n \\ 0 & m \neq n \end{cases} \tag{2.5}$$

En la mecánica cuántica, operadores tales como $\hat{O}$ en los que sus valores propios son reales, deben satisfacer la condición:

$$\int_{\text{todo el espacio}} f^*(x)\hat{O}g(x)dx = \int_{\text{todo el espacio}} g(x)[\hat{O}f]^*(x)dx \tag{2.6}$$

donde $f(x)$ y $g(x)$ son dos funciones de estado. En general, cuando un operador satisface la ecuación 2.6 se dice que es *Hermitiano*.

> *A cada observable en la mecánica clásica le corresponde un operador lineal y Hermitiano en la mecánica cuántica.*

Los valores propios de operadores Hermitianos son reales y sus funciones propias son ortonormales.

2.3.4. Operadores Hamiltonianos

El *operador Hamiltoniano* representa la energía observable de un sistema. Para este operador se acostumbra utilizar las letras H y $\mathscr{H}$ y agregar el acento circunflejo para indicar su naturaleza de operador.

2.3.5. La ecuación de Schrödinger

La ecuación de Schrödinger debe verse como un postulado fundamental o axioma de la mecánica cuántica.

La relación entre la energía (E) de una partícula y su función de onda ψ.

Esta ecuación puede escribirse en su forma más simple (independiente del tiempo) como una ecuación de valor propio:

$$\hat{H}\psi = E\psi \tag{2.7}$$

2.3.6. Momento angular de espín nuclear

Al igual que los electrones, los núcleos atómicos tienen un momento angular de espín intrínseco y un dipolo magnético asociado a él. A diferencia de los electrones, los espines nucleares no están restringidos a 1/2 (ver tabla 2.1).

Operador Hamiltoniano

Para un espín nuclear en un campo magnético B_z aplicado a lo largo del eje z, el Hamiltoniano que representa la energía de interacción entre el momento angular de espín, I_z y el campo magnético es:

$$\hat{H}_{núcleo\ aislado} = -\gamma B_z \hat{I}_z \tag{2.8}$$

donde γ es la constante giromagnética, una propiedad fundamental del núcleo en cuestión.

Núcleo	Espín	Constante giromagnética γ $[10^7 \text{rad} \cdot \text{T}^{-1} \cdot \text{s}^{-1}]$	Abundancia natural [%]
^{1}H	1/2	26.7522	99.98
^{2}H	1	4.1066	0.0156
^{10}B	3	2.88	18.83
^{11}B	3/2	8.58	81.17
^{13}C	1/2	6.7283	1.108
^{14}N	1	1.9338	99.635
^{15}N	1/2	-2.71	0.365
^{17}O	5/2	-3.63	0.037
^{19}F	1/2	25.17	100.0
^{23}Na	3/2	7.080	100.0
^{29}Si	1/2	-5.31	4.70
^{31}P	1/2	10.841	100.0
^{113}Cd	1/2	5.934	100.0

Tabla 2.1: Isótopos importantes en la espectroscopía de RMN

La ecuación de Schrödinger

La correspondiente ecuación de Schrödinger para el espín nuclear es:

$$\hat{H}\psi = -\gamma B_z \hat{I}_z \psi = E\psi \tag{2.9}$$

Las funciones de onda en este caso son las funciones propias del espín, tales que $\hat{I}\psi_I = \hbar m_I \psi_I$, donde $m_I = I, I-1, \ldots, -I$. La ecuación 2.9 nos da entonces:

$$E = -\hbar\gamma m_I B_z \tag{2.10}$$

La ecuación 2.10 puede utilizarse para calcular la diferencia en energía entre un protón alineado a favor del campo magnético y uno alineado en contra de él:

$$\Delta E = E(m_I = -1/2) - E(m_I = 1/2) = \hbar\psi B_z \tag{2.11}$$

Observe que ΔE tiene una dependencia lineal con la magnitud del campo magnético. La figura 2.1 muestra ΔE como función de B_z para un núcleo con espín 1/2. Si a un protón alineado con un campo magnético se le aplica una radiación electromagnética de una frecuencia dada por $\Delta E = \hbar\psi B_z = h\nu = \hbar\omega$, experimentará una transición del estado de menor energía ($m_I = 1/2$) al de mayor energía ($m_I = -1/2$). Con esta relación, la frecuencia asociada con la transición está dada por:

$$\nu = \frac{\gamma B_z}{2\pi} \qquad [\text{Hz}] \tag{2.12}$$

O bien,

$$\omega = \gamma B_z \qquad [\text{rad} \cdot \text{s}^{-1}] \tag{2.13}$$

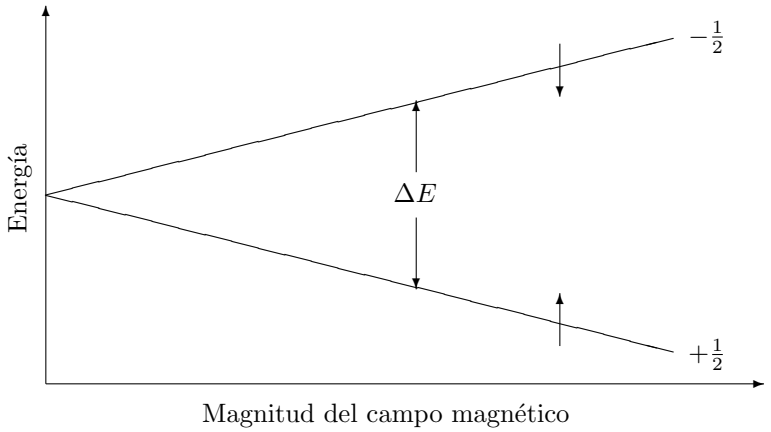


Figura 2.2: Energías relativas de un núcleo con espín 1/2 en un campo magnético. En el estado de menor energía, el núcleo está alineado a favor del campo ($m_I = +1/2$), y en el estado de mayor energía, está alineado en contra del campo ($m_I = -1/2$). La diferencia de energía depende de la magnitud del campo magnético.

Frecuencia de Larmor

La *frecuencia de Larmor* se define como:

$$\nu_0 = -\frac{\gamma B_z}{2\pi} \qquad [\text{Hz}] \tag{2.14}$$

O bien:

$$\omega_0 = -\gamma B_z \qquad [\text{ppm} \cdot \text{s}^{-1}] \tag{2.15}$$

Niveles energéticos

Los dos niveles de energía o estados disponibles para núcleos con espín 1/2 son:

$$E_m = -m\hbar\gamma B_z \qquad m = \pm\frac{1}{2} \tag{2.16}$$

Para núcleos con espín 1/2 se acostumbra etiquetar como α al estado $m = +1/2$ y describirse como "espín hacia arriba". Del mismo modo se etiqueta como β al nivel de energía con $m = -1/2$, y referirse a él como "espín hacia abajo":

$$E_\alpha = -\frac{1}{2}\hbar\gamma B_z \qquad E_\beta = +\frac{1}{2}\hbar\gamma B_z \tag{2.17}$$

Una de las reglas de la mecánica cuántica dice que las transiciones permitidas son aquellas en las que m cambia en ± 1. Así, la transición del estado α al estado β es:

$$\Delta m_{\alpha\to\beta} = -\frac{1}{2} - \left(\frac{1}{2}\right) = -1 \tag{2.18}$$

Que es una transición permitida. Se puede calcular con facilidad que para la transición de β a α, $\Delta m = +1$,

la cual también es permitida. Las funciones propias α y β son ortonormales:

$$\begin{aligned} \int \alpha^*(x)\alpha(x)dx &= \int \beta^*(x)\beta(x)dx = 1 \\ \int \alpha^*(x)\beta(x)dx &= \int \alpha(x)\beta^*(x)dx = 0 \end{aligned} \tag{2.19}$$

Ecuaciones de valor propio

Las ecuaciones de valor propio del espín nuclear son:

$$\hat{I}^2\alpha = \frac{1}{2}\left(\frac{1}{2}+1\right)\hbar^2\alpha \qquad \hat{I}^2\beta = \frac{1}{2}\left(\frac{1}{2}+1\right)\hbar^2\beta \tag{2.20a}$$

$$\hat{I}_z\alpha = \frac{1}{2}\hbar\alpha \qquad \hat{I}_z\beta = -\frac{1}{2}\hbar\beta \tag{2.20b}$$

Desplazamiento químico

De acuerdo a la ecuación 2.12, todos los núcleos activos en una molécula absorberían a la misma frecuencia, es decir, que la diferencia de energía, por ejemplo, entre los dos niveles en núcleos con espín 1/2 (ver figura 2.1), sería la misma. El valor de B_z es el campo que el núcleo experimenta, mientra que para un núcleo aislado se trata del campo magnético externo. En una molécula los núcleos están rodeados de electrones, los cuales, al ser partículas con carga eléctrica, generan un pequeño campo magnético adicional; el campo externo promueve además que dichos electrones experimenten un movimiento circular. El efecto neto, para muchas sustancias, es un campo magnético debido a los electrones, B_{elec}, que se opone al campo magnético externo. La magnitud del campo magnético B_{elec} es proporcional al campo aplicado:

$$B_{\text{elec}} = -\sigma B_0 \tag{2.21}$$

donde B_0 es el campo magnético aplicado (y que se asume que se encuentra en el eje z), y σ es la constante de proporcionalidad. El signo negativo significa que B_{elec} se opone a B_0. El valor de la constante de

proporcionalidad, denominada *constante de apantallamiento* debido a que los electrones están rodeando al núcleo como una pantalla, depende del entorno químico del núcleo.

El campo magnético total que cualquier núcleo experimenta es la suma del campo externo, B_0, y el campo debido a los electrones, B_{elec}:

$$B_z = (1 - \sigma)B_0 \tag{2.22}$$

Si se sustituye la ecuación 2.22 en las ecuaciones 2.12 y 2.13, la frecuencia de la transición entre niveles energéticos en un campo magnético fijo es:

$$\nu = \frac{\gamma(1 - \sigma)B_0}{2\pi} \qquad [\text{Hz}] \tag{2.23}$$

O bien,

$$\omega = \gamma(1 - \sigma)B_0 \qquad [\text{ppm} \cdot \text{s}^{-1}] \tag{2.24}$$

De la ecuación 2.8, el operador Hamiltoniano para un núcleo aislado es entonces:

$$\hat{H}_{núcleo\ aislado} = -\gamma(1 - \sigma)B_0\hat{I}_z \tag{2.25}$$

2.3.7. Teoría de perturbaciones

Si la ecuación de Schrödinger (ecuación 2.7) no se puede resolver para un sistema, pero se conoce la solución de un sistema similar, podemos escribir el Hamiltoniano de esta forma:

$$\hat{H} = \hat{H}^{(0)} + H^{(1)} \tag{2.26}$$

donde

$$\hat{H}^{(0)}\psi^{(0)} = E^{(0)}\psi^{(0)} \tag{2.27}$$

es la ecuación de Schrödinger que puede resolverse exactamente. Al primer término de la ecuación 2.26 se le denomina *operador Hamiltoniano sin perturbar*, mientras al término adicional se le conoce como *perturbación*.

Para aplicar la teoría de perturbaciones a la solución de la ecuación 2.7 con el Hamiltoniano de la ecuación 2.26, ψ y E pueden escribirse en esta forma:

$$\psi = \psi^{(0)} + \psi^{(1)} + \psi^{(2)} + \cdots \tag{2.28}$$

$$E = E^{(0)} + E^{(1)} + E^{(2)} + \cdots \tag{2.29}$$

donde $\psi^{(0)}$ y $E^{(0)}$ son la solución del problema sin perturbar (ecuación 2.27), $\psi^{(1)}$,$\psi^{(2)}$, ... son correcciones sucesivas a $\psi^{(0)}$ y $E^{(1)}$,$E^{(2)}$, ... son correcciones a $E^{(0)}$. Se asume que cada corrección sucesiva es cada vez menos significativa.

Si el efecto de $H^{(1)}$ es muy pequeño:

$$\begin{aligned} \psi &= \psi^{(0)} + \Delta\psi \\ E &= E^{(0)} + \Delta E \end{aligned} \tag{2.30}$$

donde $\Delta\psi$ y ΔE son pequeños. Sustituímos las ecuaciones 2.30 en la ecuación de Schrödinger, donde el hamiltoniano está dado por la ecuación 2.26:

$$\begin{aligned} &\hat{H}^{(0)}\psi^{(0)} + \hat{H}^{(1)}\psi^{(0)} + \hat{H}^{(0)}\Delta\psi + \hat{H}^{(1)}\Delta\psi \\ &= E^{(0)}\psi^{(0)} + \Delta E\psi^{(0)} + E^{(0)}\Delta\psi + \Delta E\Delta\psi \end{aligned} \tag{2.31}$$

Los primeros términos de cada lado de la ecuación 2.31 se cancelan debido a la ecuación 2.27. Además, podemos despreciar los últimos términos debido a que son el producto de dos números muy pequeños. Si acomodamos los términos, la ecuación 2.31 queda:

$$\hat{H}^{(0)}\Delta\psi + \hat{H}^{(1)}\psi^{(0)} = E^{(0)}\Delta\psi + \Delta E\psi^{(0)} \tag{2.32}$$

donde $\Delta\psi$ y ΔE son las cantidades desconocidas.

Los términos en la ecuación 2.32 son del mismo orden en el sentido de que cada uno es el producto de un término sin perturbar y un término pequeño. Se dice que esta ecuación es de primer orden y que estamos utilizando la *teoría de perturbaciones de primer orden.* Los términos despreciados en la ecuación 2.31 son términos de orden superior y llevan a correcciones de orden superior.

La ecuación 2.32 puede simplificarse. Si multiplicamos ambos lados por $\psi^{(0)*}$ e integramos en todo el espacio:

$$\int \psi^{(0)*}[\hat{H}^{(0)} - E^{(0)}]\Delta\psi d\tau + \int \psi^{(0)*}\hat{H}^{(1)}\psi^{(0)}d\tau = \Delta E \int \psi^{(0)*}\psi^{(0)}d\tau \tag{2.33}$$

La integral en el último término de la ecuación 2.33 es uno porque $\psi^{(0)}$ está normalizada. $\hat{H}^{(0)} - E^{(0)}$ es un operador Hermitiano (ver sección 2.3.3), es decir que:

$$\int \psi^{(0)*}[\hat{H}^{(0)} - E^{(0)}]\Delta\psi d\tau = \int \{[\hat{H}^{(0)} - E^{(0)}]\psi^{(0)}\}^*\Delta\psi d\tau \tag{2.34}$$

y de acuerdo a la ecuación 2.27, el integrando se hace cero. La ecuación 2.33 se convierte en:

$$\Delta E = \int \psi^{(0)*}\hat{H}^{(1)}\psi^{(0)}d\tau \tag{2.35}$$

A la ecuación 2.35 se le llama *corrección de primer orden* de $E^{(0)}$. La energía es, entonces:

$$E = E^{(0)} + \int \psi^{(0)*}\hat{H}^{(1)}\psi^{(0)}d\tau + \text{términos de orden superior} \tag{2.36}$$

o bien

$$E = E^{(0)} + E^{(1)} + \text{términos de orden superior} \tag{2.37}$$

2.3.8. Acoplamiento espín-espín de un sistema AX

La *multiplicidad* de las señales en los espectros de RMN se debe a la interacción magnética entre núcleos activos y es transmitida a través de los electrones de enlace por medio de los cuales dichos núcleos están conectados (ver sección 2.2.3).

Consideremos un sistema de espín AX, es decir, uno en el que el ambiente químico de un núcleo con respecto al otro es muy diferente. Las funciones de onda son los productos de las funciones de espín del sistema:

$$\begin{aligned} \psi_1 &= \alpha(1)\alpha(2) \qquad \psi_2 = \beta(1)\alpha(2) \\ \psi_3 &= \alpha(1)\beta(2) \qquad \psi_4 = \beta(1)\beta(2) \end{aligned} \tag{2.38}$$

Utilizando la ecuación 2.25, el operador Hamiltoniano en ausencia de interacción espín-espín es:

$$\hat{H} = -\gamma B_0(1-\sigma_1)\hat{I}_{z1} - \gamma B_0(1-\sigma_2)\hat{I}_{z2} \tag{2.39}$$

donde σ_j es el desplazamiento químico del j-ésimo núcleo.

La expresión clásica para la interacción entre los momentos de dos dipolos magnéticos involucra un factor $\vec{\mu}_1 \cdot \vec{\mu}_2$, donde $\vec{\mu}_j$ son los vectores que representan los momentos de los dipolos magnéticos (una forma de representar cantidades vectoriales consiste en utilizar una flecha o una barra sobre las letras). En la mecánica cuántica, $\vec{\mu}$ es proporcional al momento angular de espín nuclear, $\vec{I}$:

$$\vec{\mu} = g_{\mathrm{N}} \frac{q}{2m_{\mathrm{N}}} \vec{I} = g_{\mathrm{N}} \beta_{\mathrm{N}} \vec{I} = \gamma \vec{I} \tag{2.40}$$

donde g_{N} es el *factor nuclear g*, β_{N} es el *magnetón nuclear* $(q/2m_{\mathrm{N}})$, m_{N} es la masa del núcleo, y $\gamma = g_{\mathrm{N}}\beta_{\mathrm{N}}$ es la constante giromagnética. La magnitud del vector $\vec{I}$ está dada por:

$$|\vec{I}^2| = \vec{I} \cdot \ \vec{I} = \hbar\,[I((I+1)] \tag{2.41}$$

Para tomar en cuenta el efecto del acoplamiento espín-espín en el Hamiltoniano, se incluye un término proporcional a $\widehat{\vec{I}\cdot\vec{I}}$. Siendo J_{12} la constante de proporcionalidad, el Hamiltoniano del sistema de dos espines es:

$$\hat{H} = \underbrace{-\gamma B_0(1-\sigma_1)\hat{I}_{z1} - \gamma B_0(1-\sigma_2)\hat{I}_{z2}}_{\hat{H}^{(0)}} + \underbrace{\frac{h}{\hbar^2}J_{12}\widehat{\vec{I}_1\cdot\vec{I}_2}}_{\hat{H}^{(1)}} \tag{2.42}$$

El factor $\frac{h}{\hbar^2}$ se incluye para definir en Hz las unidades de la constante J_{12}, la cual se denomina *constante de acoplamiento espín-espín*. Asumimos que la interacción espín-espín puede ser tratada como una pertubación de primer orden, por lo que el término $\hat{H}^{(0)}$ es el operador Hamiltoniano no perturbado y $\hat{H}^{(1)}$ es la perturbación (ver sección 2.3.7).

La energía utilizando la corrección de primer orden está dada por (ecuaciones 2.36 y 2.37):

$$E_j = E_j^{(0)} + \int d\tau_1\tau_2\psi_j^*\hat{H}^{(1)}\psi_j \tag{2.43}$$

Las $E_j^{(0)}$ son:

$$E^{(0)}\psi_j = E_j^{(0)}\psi_j \tag{2.44}$$

donde ψ_j están dadas por las ecuaciones 2.38.

Resolvamos primero las ecuaciones del sistema sin perturar, tomando $\psi_1 = \alpha(1)\alpha(2)$ y usando las

ecuaciones 2.20b y 2.39:

$$\begin{aligned}
\hat{H}^{(0)}\psi_1 &= \hat{H}^{(0)}\alpha(1)\alpha(2) \\
&= -\gamma B_0(1-\sigma_1)\hat{I}_{z1}\alpha(1)\alpha(2) - \gamma B_0(1-\sigma_2)\hat{I}_{z2}\alpha(1)\alpha(2) \\
&= -\gamma B_0\left[(1-\sigma_1)\left(\frac{\hbar}{2}\right)\alpha(1)\alpha(2) + (1-\sigma_2)\alpha(1)\left(\frac{\hbar}{2}\right)\alpha(2)\right] \\
&= -\frac{\gamma B_0\hbar}{2}\left[(1-\sigma_1)+(1-\sigma_2)\right]\alpha(1)\alpha(2) \\
&= -\frac{\gamma B_0\hbar}{2}(2-\sigma_1-\sigma_2)\alpha(1)\alpha(2) \\
&= E_1^{(0)}\alpha(1)\alpha(2) \\
&= E_1^{(0)}\psi_1
\end{aligned} \tag{2.45}$$

es decir que:

$$E_1^{(0)} = -\frac{\gamma B_0\hbar}{2}(2-\sigma_1-\sigma_2) \tag{2.46}$$

Del mismo modo, con ψ_2:

$$\begin{aligned}
\hat{H}^{(0)}\psi_1 &= \hat{H}^{(0)}\beta(1)\alpha(2) \\
&= -\gamma B_0(1-\sigma_1)\hat{I}_{z1}\beta(1)\alpha(2) - \gamma B_0(1-\sigma_2)\hat{I}_{z2}\beta(1)\alpha(2) \\
&= -\gamma B_0\left[(1-\sigma_1)\left(-\frac{\hbar}{2}\right)\beta(1)\alpha(2) + (1-\sigma_2)\beta(1)\left(\frac{\hbar}{2}\right)\alpha(2)\right] \\
&= -\frac{\gamma B_0\hbar}{2}\left[(\sigma_1-1)+(1-\sigma_2)\right]\beta(1)\alpha(2) \\
&= -\frac{\gamma B_0\hbar}{2}(\sigma_1-\sigma_2)\beta(1)\alpha(2) \\
&= E_2^{(0)}\beta(1)\alpha(2) \\
&= E_2^{(0)}\psi_2
\end{aligned} \tag{2.47}$$

por lo que:

$$E_2^{(0)} = -\frac{\gamma B_0\hbar}{2}(\sigma_1-\sigma_2) \tag{2.48}$$

Siguiendo el mismo procedimiento:

$$E_3^{(0)} = \frac{\gamma B_0 \hbar}{2}(\sigma_1 - \sigma_2) \tag{2.49}$$

$$E_4^{(0)} = \frac{\gamma B_0 \hbar}{2}(2 - \sigma_1 - \sigma_2) \tag{2.50}$$

La regla para las transiciones entre estados del espín nuclear dicen que sólo un tipo de núcleo a la vez puede cambiar de estado en $\Delta m = \pm 1$. Por ello, las transiciones permitidas para la absorción (ver figura 2.3) son:

$$\begin{aligned}
\alpha(1)\alpha(2) &\longrightarrow \beta(1)\alpha(2) \qquad (1 \to 2)\\
\alpha(1)\alpha(2) &\longrightarrow \alpha(1)\beta(2) \qquad (1 \to 3)\\
\beta(1)\alpha(2) &\longrightarrow \beta(1)\beta(2) \qquad (2 \to 4)\\
\alpha(1)\beta(2) &\longrightarrow \beta(1)\beta(2) \qquad (3 \to 4)
\end{aligned}$$

Para calcular la frecuencia asociada a las transiciones permitidas en el sistema sin perturbar tomamos:

$$\begin{aligned}
\Delta E_{2\to 4}^{(0)} &= E_4^{(0)} - E_2^{(0)}\\
&= \frac{\gamma B_0 \hbar}{2}(2 - \sigma_1 - \sigma_2) - \left[-\frac{\sigma B_0 \hbar}{2}(\sigma_1 - \sigma_2)\right]\\
&= \frac{\gamma B_0 \hbar}{2}(2 - \cancel{\sigma_1} - \sigma_2 + \cancel{\sigma_1} - \sigma_2)\\
&= \frac{\gamma B_0 h}{2\pi}(1 - \sigma_2)\\
&= h\nu_0(1 - \sigma_2)
\end{aligned} \tag{2.51}$$

donde

$$\nu_0 = \frac{\gamma B_0}{2\pi} \tag{2.52}$$

Como $\Delta E = h\nu$

$$\begin{aligned} \Delta E^{(0)}_{2\to 4} &= h\nu_{2\to 4} = h\nu_0(1-\sigma_2) \\ \nu_{2\to 4} &= \nu_0(1-\sigma_2) \end{aligned} \tag{2.53}$$

Con el mismo procedimiento

$$\nu_{1\to 3} = \nu_0(1-\sigma_2) \tag{2.54}$$

$$\nu_{1\to 2} = \nu_0(1-\sigma_1) \tag{2.55}$$

$$\nu_{3\to 4} = \nu_0(1-\sigma_1) \tag{2.56}$$

Las ecuaciones anteriores pueden expresarse como:

$$\nu_1 = \nu_0(1-\sigma_1) \tag{2.57}$$

$$\nu_2 = \nu_0(1-\sigma_2) \tag{2.58}$$

La frecuencia de separación entre los picos es

$$\begin{aligned} \Delta\nu &= |\nu_2 - \nu_1| \\ &= |\nu_0(\cancel{1} - \sigma_1) - \nu_0(\cancel{1} - \sigma_2)| \\ &= \nu_0|\sigma_2 - \sigma_1| \end{aligned} \tag{2.59}$$

Para calcular las correcciones de primer orden se deben evaluar las integrales:

$$H_{ii} = \frac{h}{\hbar} J_{12} \int d\tau_1 d\tau_2 \psi_i^* \widehat{\vec{I}_1 \cdot \vec{I}_2} \psi_i \tag{2.60}$$

El producto punto de $\widehat{\vec{I}_1 \cdot \vec{I}_2}$ es:

$$\widehat{\vec{I}_1 \cdot \vec{I}_2} = \hat{I}_{x1}\hat{I}_{x2} + \hat{I}_{y1}\hat{I}_{y2} + \hat{I}_{z1}\hat{I}_{z2} \tag{2.61}$$

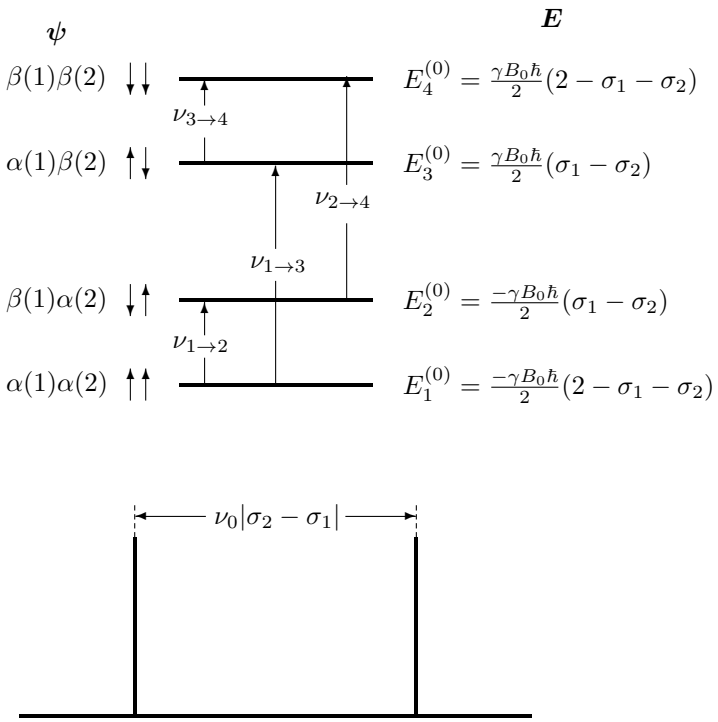


Figura 2.3: Niveles de energía de un sistema de espín AX en ausencia de acoplamiento.

por lo que la ecuación a resolver es:

$$H_{ii} = \frac{h}{\hbar^2} J_{12} \left[\int d\tau_1 d\tau_2 \psi_i^* \left(\hat{I}_{x1}\hat{I}_{x2} + \hat{I}_{y1}\hat{I}_{y2} + \hat{I}_{z1}\hat{I}_{z2} \right) \psi_i \right] \tag{2.62}$$

Para evaluar los términos de la integral que involucran $\hat{I}_{x1}\hat{I}_{x2}$ y $\hat{I}_{y1}\hat{I}_{y2}$ es necesario conocer la relación que guardan las componentes en x y y del momento angular de espín nuclear con α y β (ver el apéndice A):

$$\begin{aligned} \hat{I}_x\alpha &= \frac{\hbar}{2}\beta \qquad \hat{I}_y\alpha = \frac{i\hbar}{2}\beta \\ \hat{I}_x\beta &= \frac{\hbar}{2}\alpha \qquad \hat{I}_y\beta = -\frac{i\hbar}{2}\alpha \end{aligned} \tag{2.63}$$

Si tomamos ψ_1 como está definida en la ecuación 2.38, y usando las ecuaciones 2.63:

$$\begin{aligned}\hat{I}_{x1}\hat{I}_{x2}\alpha(1)\alpha(2) &= [\hat{I}_{x1}\alpha(1)][\hat{I}_{x2}\alpha(2)] \\ &= \frac{\hbar}{2}\beta(1)\frac{\hbar}{2}\beta(2) \\ &= \frac{\hbar^2}{4}\beta(1)\beta(2)\end{aligned}$$

por lo que

$$\begin{aligned}H_{x,11} &= \frac{h}{\hbar^2}J_{12}\int\int d\tau_1 d\tau_2 \alpha^*(1)\alpha^*(2)\hat{I}_{x1}\hat{I}_{x2}\alpha(1)\alpha(2) \\ &= \frac{h}{\hbar^2}J_{12}\int\int d\tau_1 d\tau_2 \alpha^*(1)\alpha^*(2)\frac{\hbar^2}{4}\beta(1)\beta(2)\alpha(1)\alpha(2) \\ &= \frac{h}{4}J_{12}\int d\tau_1 \alpha^*(1)\beta(1)\int d\tau_2 \alpha^*(2)\beta(2) = 0\end{aligned}$$

donde utilizamos la ortogonalidad de las funciones α y β. Se puede probar de modo similar que los términos x y y de $\widehat{\vec{I}_1 \cdot \vec{I}_2}$ no contribuyen en el valor de la energía para este caso.

Para $\psi_1 = \alpha(1)\alpha(2)$ y la componente en z de $\widehat{\vec{I}_1 \cdot \vec{I}_2}$:

$$\begin{aligned}\hat{I}_{z1}\hat{I}_{z2}\alpha(1)\alpha(2) &= [\hat{I}_{z1}\alpha(1)][\hat{I}_{z2}\alpha(2)] \\ &= \frac{\hbar}{2}\alpha(1)\frac{\hbar}{2}\alpha(2) \\ &= \frac{\hbar^2}{4}\alpha(1)\alpha(2)\end{aligned}$$

y por ello

$$\begin{aligned}H_{z,11} = &= \frac{h}{\hbar^2}J_{12}\int\int d\tau_1 d\tau_2 \alpha^*(1)\alpha^*(2)\hat{I}_{z1}\hat{I}_{z2}\alpha(1)\alpha(2) \\ &= \frac{h}{\hbar^2}J_{12}\frac{\hbar^2}{4}\int d\tau_1 \alpha^*(1)\alpha(1)\int d\tau_2 \alpha^*(2)\alpha(2) \\ &= \frac{h}{4}J_{12}\end{aligned}$$

Del mismo modo, con $\psi_2 = \beta(1)\alpha(2)$:

$$\begin{aligned}\hat{I}_{z1}\hat{I}_{z2}\beta(1)\alpha(2) &= [\hat{I}_{z1}\beta(1)][\hat{I}_{z2}\alpha(2)] \\ &= -\frac{\hbar}{2}\beta(1)\frac{\hbar}{2}\alpha(2) \\ &= -\frac{\hbar^2}{4}\beta(1)\alpha(2)\end{aligned}$$

y

$$\begin{aligned}H_{z,22} = &= \frac{h}{\hbar^2}J_{12}\int\int d\tau_1 d\tau_2 \beta^*(1)\alpha^*(2)\hat{I}_{z1}\hat{I}_{z2}\beta(1)\alpha(2) \\ &= \frac{h}{\hbar^2}J_{12}\left(-\frac{\hbar^2}{4}\right)\int d\tau_1 \beta^*(1)\alpha(1)\int d\tau_2 \beta^*(2)\alpha(2) \\ &= -\frac{h}{4}J_{12}\end{aligned}$$

Con ψ_3:

$$\begin{aligned}H_{z,33} = &= \frac{h}{\hbar^2}J_{12}\int\int d\tau_1 d\tau_2 \alpha^*(1)\beta^*(2)\hat{I}_{z1}\hat{I}_{z2}\alpha(1)\beta(2) \\ &= \frac{h}{\hbar^2}J_{12}\int\int d\tau_1 d\tau_2 \alpha^*(1)\beta^*(2)\hat{I}_{z1}\alpha(1)\hat{I}_{z2}\beta(2) \\ &= \frac{h}{\hbar^2}J_{12}\left(-\frac{\hbar^2}{4}\right)\int d\tau_1 \alpha^*(1)\beta(1)\int d\tau_2 \alpha^*(2)\beta(2) \\ &= -\frac{h}{4}J_{12}\end{aligned}$$

Finalmente, con ψ_4:

$$\begin{aligned}H_{z,44} = &= \frac{h}{\hbar^2}J_{12}\int\int d\tau_1 d\tau_2 \beta^*(1)\beta^*(2)\hat{I}_{z1}\hat{I}_{z2}\beta(1)\beta(2) \\ &= \frac{h}{\hbar^2}J_{12}\int\int d\tau_1 d\tau_2 \beta^*(1)\beta^*(2)\hat{I}_{z1}\beta(1)\hat{I}_{z2}\beta(2) \\ &= \frac{h}{\hbar^2}J_{12}\frac{\hbar^2}{4}\int d\tau_1 \beta^*(1)\beta(1)\int d\tau_2 \beta^*(2)\beta(2) \\ &= \frac{h}{4}J_{12}\end{aligned}$$

La energía para cada nivel, con correcciones de primer orden es:

$$\begin{aligned} E_1 &= -\frac{h\nu_0}{2}(2-\sigma_1-\sigma_2)+\frac{h}{4}J_{12} \\ E_2 &= -\frac{h\nu_0}{2}(\sigma_1-\sigma_2)-\frac{h}{4}J_{12} \\ E_3 &= \frac{h\nu_0}{2}(\sigma_1-\sigma_2)-\frac{h}{4}J_{12} \\ E_4 &= \frac{h\nu_0}{2}(2-\sigma_1-\sigma_2)+\frac{h}{4}J_{12} \end{aligned} \tag{2.64}$$

Las frecuencias asociadas con las transiciones permitidas:

$$\begin{aligned} \nu_{1\to 2} &= \nu_0(1-\sigma_1)-\frac{1}{2}J_{12} \\ \nu_{1\to 3} &= \nu_0(1-\sigma_2)-\frac{1}{2}J_{12} \\ \nu_{2\to 4} &= \nu_0(1-\sigma_2)+\frac{1}{2}J_{12} \\ \nu_{3\to 4} &= \nu_0(1-\sigma_1)+\frac{1}{2}J_{12} \end{aligned} \tag{2.65}$$

Las ecuaciones anteriores pueden expresarse de manera simplificada como:

$$\begin{aligned} \nu_1^{\pm} &= \nu_0(1-\sigma_1)\pm\frac{1}{2}J_{12} \\ \nu_2^{\pm} &= \nu_0(1-\sigma_2)\pm\frac{1}{2}J_{12} \end{aligned} \tag{2.66}$$

La condición para utilizar la teoría de perturbaciones de primer orden es que $J_{12} \ll \nu_0|\sigma_1-\sigma_2|$. Cuando en un espectro se cumple esta condición, se observan dos dobletes separados y al espectro se le llama *espectro de primer orden*. Valores típicos de constantes de acoplamiento son de alrededor de 5 Hz, por lo que un espectro de primer orden resultará si la distancia entre multipletes es de 100 Hz o más.

Si bien la separación entre picos es dependiente de la densidad del flujo magnético (ver ecuaciones 2.52 y 2.65), el valor de las constantes de acoplamiento es independiente de él.

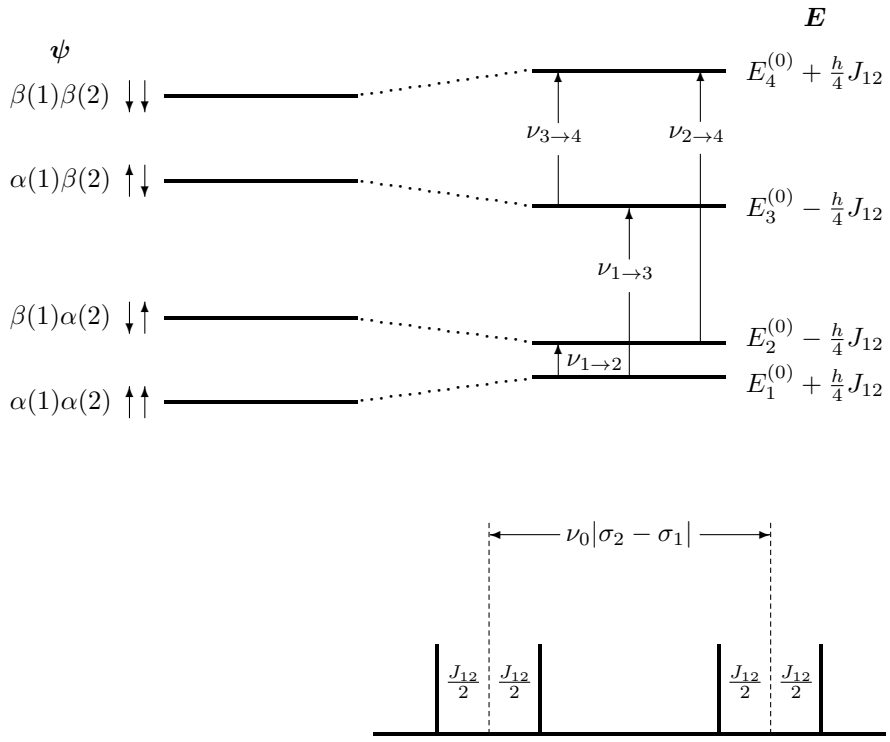


Figura 2.4: Niveles de energía de un sistema de espín AX calculado con la teoría de perturbaciones de primer orden.

2.3.9. Acoplamiento espín-espín de un sistema A_2

El espectro de primer orden de un sistema AX consiste en un par de dobletes. En un sistema de espín A_2, es decir, cuando los dos núcleos son químicamente equivalentes, el operador Hamiltoniano es (ver ecuación 2.42):

$$\hat{H} = -\gamma B_0(1-\sigma_A)\hat{I}_{z1} - \gamma B_0(1-\sigma_A)\hat{I}_{z2} + \frac{h}{\hbar^2}J_{AA}\widehat{\vec{I_1}\cdot\vec{I_2}} \tag{2.67}$$

el cual es similar al del sistema AX con excepción de que las constantes de apantallamiento son iguales. Del mismo modo que en la sección anterior

$$\hat{H}^{(0)} = -\gamma B_0(1-\sigma_A)(\hat{I}_{z1} + \hat{I}_{z2}) \tag{2.68}$$

es el operador Hamiltoniano sin pertubar y

$$\hat{H}^{(1)} = \frac{h}{\hbar^2} J_{\mathrm{AA}} \widehat{\vec{I_1} \cdot \vec{I_2}} \tag{2.69}$$

es el término de la perturbación. La diferencia principal entre la teoría de perturbación de primer orden en el tratamiento del sistema AX y la del sistema A_2 es la forma de las funciones de onda sin perturbar del espín. Debido a que los dos núcleos son equivalentes, y por tanto indistinguibles en el caso de A_2, se deben usar combinaciones de α y β que sean tanto simétricas como antisimétricas. Las cuatro combinaciones aceptables son:

$$\begin{aligned} \phi_1 &= \alpha(1)\alpha(2) & \phi_2 &= \frac{1}{\sqrt{2}}[\alpha(1)\beta(2) - \beta(1)\alpha(2)] \\ \phi_3 &= \frac{1}{\sqrt{2}}[\alpha(1)\beta(2) + \beta(1)\alpha(2)] & \phi_4 &= \beta(1)\beta(2) \end{aligned} \tag{2.70}$$

Si utilizamos la ecuación 2.43 para calcular las cuatro energías de primer orden tendremos:

$$\begin{aligned} E_1 &= -\hbar\gamma B_0(1-\gamma_{\mathrm{A}}) + \frac{h}{4} J_{\mathrm{AA}} \\ E_2 &= -\frac{3h}{4} J_{\mathrm{AA}} \\ E_3 &= \frac{h}{4} J_{\mathrm{AA}} \\ E_4 &= \hbar\gamma B_0(1-\gamma_{\mathrm{A}}) + \frac{h}{4} J_{\mathrm{AA}} \end{aligned} \tag{2.71}$$

Las reglas de selección establecen que sólo un espín a la vez puede experimentar una transición, pero sólo están permitidas las transiciones entre estados de la misma simetría. Así, las transiciones permitidas son $1 \leftarrow 3$ y $3 \leftarrow 4$. Las frecuencias que corresponden a estas transicones son:

$$\nu_{1\leftarrow 3} = \nu_{3\leftarrow 4} = \frac{E_3 - E_1}{h} = \frac{\gamma B_0(1-\gamma_{\mathrm{A}})}{2\pi} = \nu_0(1-\gamma\mathrm{A}) \tag{2.72}$$

Por lo tanto, aunque el acoplamiento espín-espín entre núcleos equivalentes altera los niveles energéticos, las reglas de selección son tales que el efecto de la constante de acoplamiento espín-espín se cancela en las frecuencias de transición, de modo que sólo se observa la resonancia de un núcleo único.

2.3.10. Acoplamiento espín-espín de otros sistemas

La relativa simplicidad de los espectros de primer orden se debe a que las constantes de acoplamiento son pequeñas en relación con la separación de los multipletes. Cuando éste no es el caso, para predecir un espectro correctamente es necesario utilizar un cálculo variacional. Sin embargo, no es objetivo del presente trabajo el describir matemáticamente toda la teoría detrás de la RMN y, específicamente, de las constantes de acoplamiento, sino únicamente mostrar la forma en que se puede explicar el fenómeno.

2.4. Constantes de acoplamiento

Las constantes de acoplamiento son una fuente rica de información acerca de la estructura y conformación molecular. Los acoplamientos escalares indican que hay unión entre núcleos a través de los enlaces químicos, y proporcionan información de la distribución electrónica y geometría de estos enlaces. [4]

La tabla 2.2 muestra valores de constantes de acoplamiento H–H de diferentes moléculas. Puede observarse que los valores están en un intervalo de 5 a 7 Hz y que son muy sensibles a la estereoquímica de las moléculas. Un ejemplo lo ofrece la interacción espín-espín en protones olefínicos, donde la J_{trans} siempre es más grande que la J_{cis}. Del mismo modo puede observarse que en el ciclohexano, $J_{\text{aa}} > J_{\text{ee}}$. [3]
Existen valores de J entre 9 y 17 hz que también son muy sensibles a la estereoquímica de las moléculas.

El acoplamiento ^{13}C–^{1}H es muy sensible a efectos locales tales como la distribución electrónica, los ángulos de los enlaces y efectos debido a los sustituyentes. Sin embargo, es hasta estudios más recientes con derivados de hidratos de cabono que se han observado efectos conformacionales en acoplamientos anoméricos carbón-protón. Por ejemplo, en los anómeros α (1) y β (2) los valores de J_{CH} para el carbono 1 son 170.1 y 161.3 Hz, respectivamente (figura 2.5).

Un descubrimiento importante por parte de Barfield y Grant fue que la magnitud de $^{2}J_{\text{HH}}$ dependía del $\cos^2\phi$, donde ϕ es el ángulo de rotación formado por los ángulos H–C alrededor de los enlaces H_2C–X, siendo X un átomo con hibridación sp^2. Una aplicación importante de esta relación se puede observar en los residuos de glicina en cadenas peptídicas y proteínas. [8]

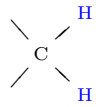
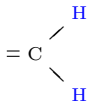
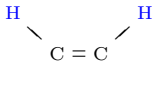
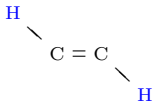
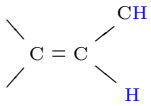
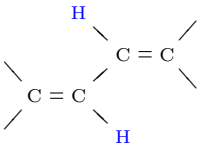
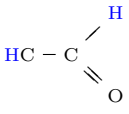
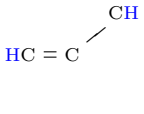
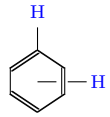
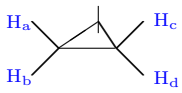
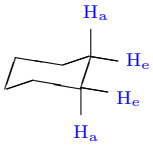
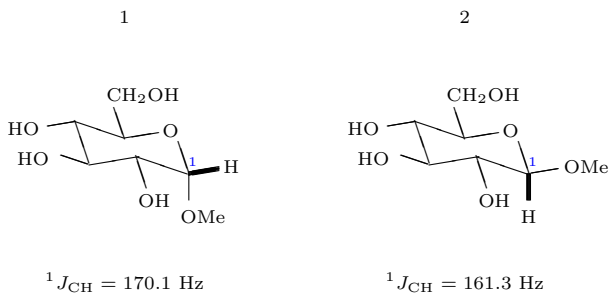
			
$12 - 29$	$0 - 3.5$		
$HC - CH$			
$2 - 9$	$6 - 14$	$11 - 18$	$4 - 10$
			$HC - C \equiv C - H$
$10 - 13$	$3 - 7$	$1 - 3$	$2 - 3$
			
$J_o = 7 - 10$	$J_{bd} = 5 - 12$	$J_{aa} = 10 - 13$	
$J_m = 2 - 3$	$J_{cd} = 4 - 10$	$J_{ae} = 2 - 5$	
$J_p = 0.1 - 1$	$J_{bc} = 3 - 9$	$J_{ee} = 2 - 5$	

Tabla 2.2: Valores típicos de constantes de acoplamiento H-H en Hz.



$^1J_{CH} = 170.1$ Hz $^1J_{CH} = 161.3$ Hz

Figura 2.5: Anómeros α y β de los metil-D-glucósidos.

Karplus describió la relación de la constante de acoplamiento a tres enlaces de distancia con el ángulo dihedro, ϕ, entre dos vectores C–H, esto es, en fragmentos HCCH:

$$^{3}J_{\mathrm{HH}} = A\cos\phi + B\cos^{2}\phi + C \tag{2.73}$$

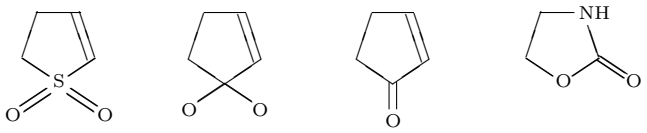
A pesar de que usualmente es necesario emplear valores empíricos para A, B y C, esta relación ha probado ser muy útil en la práctica. Esta ecuación ha sido modificada para mejorar su exactitud y para comprender sus límites y excepciones. Un ejemplo es la ecuación extendida de Imai y Osawa que incluye 11 términos estructurales independientes y 22 parámetros ajustables, seleccionados con bases empíricas como una combinación lineal. Esta nueva ecuación reproduce 198 constantes de acoplamiento vecinales protón-protón con una desviación estándar de 0.33 Hz:

$$\begin{aligned} J = {} & A\cos\theta + B\cos 2\theta + C\cos 3\theta + D\cos^{2}2\theta \\ & + W\left(E\cos\theta\sum\Delta\chi_i\cos\phi_i + F\sum\Delta\chi_i\cos 2\phi_i + G\sum\Delta\chi_i\right) \\ & + H\left[\frac{\omega_1+\omega_2}{2} - 110\right] + I(r_{\text{C-C}} - 1.5) \\ & + K\sum\Delta\chi_j^{\beta}\cos 2\psi_j + Lr^{-4} + M \end{aligned} \tag{2.74}$$

Ahora se sabe que además de la dependencia del ángulo dihedro, las constantes de acoplamiento vecinales dependen de la electronegatividad y orientación de los sustituyentes del fragmento H–C–C–H, de los ángulos del enlace H–C–C, el traslape de orbitales de los núcleos adjacentes, posiblemente de los pares de electrones libres y de efectos de hiperconjugación. Por ello debe tenerse cuidado en la interpretación de la magnitud de constantes de acoplamiento H–H a tres enlaces de distancia en términos de conformación.

La figura 2.4 muestra las constantes de acoplamiento *cis* y *trans* en el fragmento –CH_2CH_2–, donde el ángulo dihedro entre protones es aproximadamente de 120°.

Los espectros de RMN proporcionan cinco piezas de información acerca de las moléculas: desplazamientos químicos, constantes de acoplamiento, tiempos de relajación, los NOE's y la dinámica molecular. Todas estos parámetros espectroscópicos deberían ser tomados en cuenta en la determinación de la estruc-



$^3J_{cis}$	9.00	8.60	7.20	9.21
$^3J_{trans}$	4.20	4.20	2.24	6.81

Figura 2.6: Constantes de acoplamiento en anillos de heteroátomos

Constantes de acoplamiento vecinales *cis* y *trans* en fragmentos –CH_2CH_2– en anillos de heteroátomos de cinco miembros.

tura y conformación de moléculas. Los ejemplos mostrados pretenden resaltar el hecho de que calcular la magnitud de las constantes de acoplamiento puede ser de gran ayuda.

2.4.1. Medición de J

Cuando el multiplete de interés está bien resuelto, la medición de la constante de acoplamiento es algo relativamente sencillo que consiste en localizar los picos, ya sea manualmente o mediante la ayuda de un programa de computadora. Sin embargo, los problemas surgen cuando el multiplete está pobremente resuelto o cuando hay traslape de señales.

La espectroscopía de correlación bidimensional heteronuclear ofrece algunas veces un modo de evitar las dificultades debidas al traslape. Las líneas espectrales separadas en un multiplete que normalmente no está resuelto, en un experimento heteronuclear se verán más separadas, permitiendo la medición directa de la separación homonuclear. [1]

Existen varios métodos, tanto experimentales como de manejo de datos para realizar el cálculo de J. En el primer caso, el recurso obvio es incrementar la resolución de los espectros. En el lado del manejo de datos, se pueden aplicar procedimientos tanto en el dominio del tiempo, como en el de las frecuencias. Entre estos métodos están el ajuste por cuadrados mínimos, diversos métodos de comparación, el método de extensión de J, la deconvolución de J y el método de duplicación de J. Los métodos en el dominio de

la frecuencia han mostrado ser más rápidos debido a que evitan el uso iterativo de transformaciones de Fourier. En el presente trabajo se utilizó una modificación al método de duplicación de J en el dominio de las frecuencias,

2.4.2. Método modificado de duplicación de J

El método modificado de duplicación de J en el dominio de las frecuencias está basado en la convolución. La convolución de dos funciones $F(\omega)$ y $G(\omega)$ está definida como:

$$H(\omega) = F(\omega) \otimes G(\omega) = \int F(\omega')G(\omega - \omega')d\omega' \tag{2.75}$$

En este método, $F(\omega)$ corresponde a un multiplete ideal sin ruido, $G(\omega)$ un multiplete reducido, y $H(\omega)$ el multiplete de estudio, el cual necesita ser aislado, digitalizado y procesado para corregir la línea base. El objetivo del método consiste en obtener a partir del multiplete de estudio, un multiplete simplificado, extrayendo en el proceso el valor de la constante de acoplamiento. Este procedimiento es conocido como *deconvolución*, pues consiste en resolver la ecuación 2.75 para determinar $G(\omega)$, conociendo $H(\omega)$ y $F(\omega)$. La convolución es conmutativa, y puede escribirse con una notación más simple:

$$H(\omega) = F(\omega) \otimes G(\omega) = G(\omega) \otimes F(\omega) \tag{2.76}$$

Puede probarse que, para obtener $G(\omega)$ a partir de la ecuación 2.76 basta que $F(\omega)$ posea una función *recíproca con respecto a la convolución*. [7] Si denotamos a esta función como $F(\omega)^{\otimes -1}$, tendremos que:

$$F(\omega)^{\otimes -1} \otimes F(\omega) = F(\omega) \otimes F(\omega)^{\otimes -1} = N(\omega) \tag{2.77}$$

donde $N(\omega)$ es el elemento neutro con respecto a la convolución. Es decir que, $F(\omega) \otimes N(\omega) = N(\omega) \otimes$

$F(\omega) = F(\omega)$. Si dicha función existe, puede obtenerse $G(\omega)$ de la ecuación 2.76:

$$\begin{aligned} F(\omega)^{\otimes -1} \otimes H(\omega) &= F(\omega)^{\otimes -1} \otimes F(\omega) \otimes G(\omega) \\ &= N(\omega) \otimes G(\omega) \\ &= G(\omega) \end{aligned} \tag{2.78}$$

Denominaremos a $F(\omega)^{\otimes -1}$ como *función* δ con lo que definimos la deconvolución como:

$$G(\omega) = \delta(\omega) \otimes H(\omega) \tag{2.79}$$

La modificación al método de duplicación de J original estriba en la definición de la función δ [2]. La forma de esta función depende a su vez de la forma del multiplete de estudio. La figura 2.7 muestra las funciones δ adecuadas a la forma del multiplete. El resultado mostrado, es decir el multiplete reducido, se obtiene cuando el valor de la J real, J_{IS}, es igual a la separación entre los valores distintos de cero de la función δ, J^* (llamada J de prueba.)

Para resolver la ecuación 2.79, es decir

$$G(\omega) = \int_0^{\omega} \delta(\omega') H(\omega - \omega') d\omega' \tag{2.80}$$

escrita de forma explícita, es necesario definir las funciones $\delta(\omega)$ y $H(\omega)$, y los límites del dominio de la función resultante, $G(\omega)$. A modo de ejemplo, tomemos el multiplete $H(\omega)$ en antifase como se muestra en la figura 2.8. La función $H(\omega)$ representada se trata en realidad de datos experimentales obtenidos por un experimento de RMN, específicamente, la región del espectro correspondiente al multiplete. Por ello, no se trata de una función contínua sino discreta, con una separación entre valores de intensidad de la resolución del espectro. El dominio de $H(\omega)$ es $\omega = [\text{resolución}, \text{ventana espectral}]$. Para reducir un

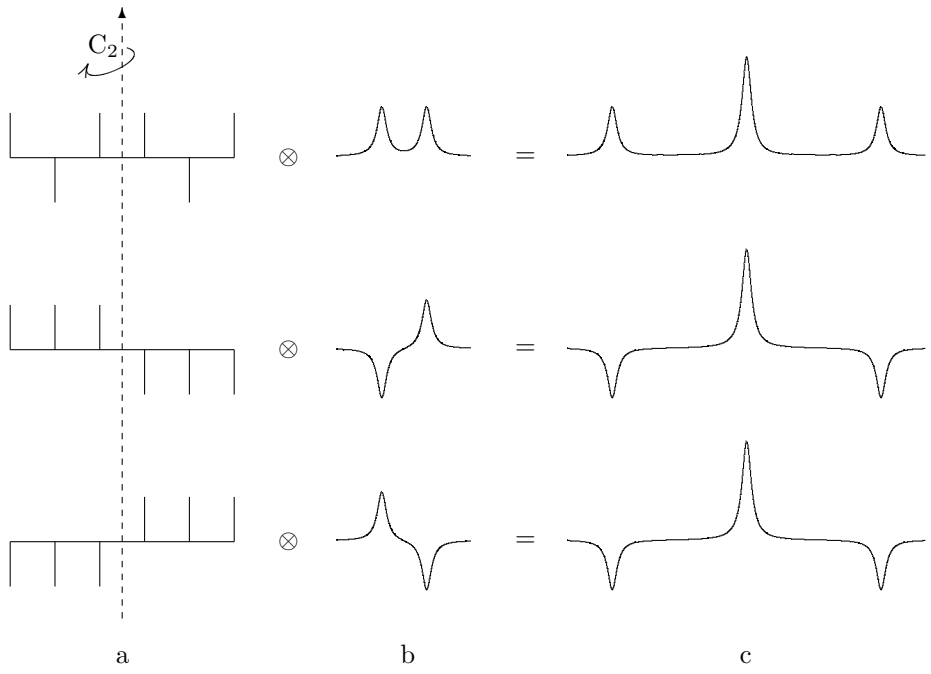


Figura 2.7: Deconvolución de multipletes en fase y en antifase con el uso de funciones δ simétricas. Una función δ específica (a) para cada multiplete (b), resulta en dos submultipletes en los extremos y el multiplete deconvolucionado en el centro (c).

multiplete en antifase como éste, la función δ adecuada se define como:

$$\delta(\omega) = \begin{cases} +1 & & & \omega & = & 0 \\ 0 & 0 & < & \omega & < & J^* \\ +1 & & & \omega & = & J^* \\ 0 & J^* & < & \omega & < & 2J^* \\ +1 & & & \omega & = & 2J^* \\ \vdots & & & & & \\ +1 & & & \omega & = & \left(\frac{n}{2}-1\right) J^* \\ 0 & \left(\frac{n}{2}-1\right) J^* & < & \omega & < & \left(\frac{n}{2}\right) J^* \\ -1 & & & \omega & = & \left(\frac{n}{2}\right) J^* \\ 0 & \left(\frac{n}{2}\right) J^* & < & \omega & < & \left(\frac{n}{2}+1\right) J^* \\ -1 & & & \omega & = & \left(\frac{n}{2}+1\right) J^* \\ 0 & \left(\frac{n}{2}+1\right) J^* & < & \omega & < & \left(\frac{n}{2}+2\right) J^* \\ -1 & & & \omega & = & \left(\frac{n}{2}+2\right) J^* \\ \vdots & & & & & \\ -1 & & & \omega & = & (n-1)J^* \end{cases} \tag{2.81}$$

donde $n = [4, 6, 8, 10, \cdots]$. En la figura 2.8 se muestra una función δ con $n = 6$, es decir:

$$\delta(\omega) = \begin{cases} 1 & & & \omega & = & 0 \\ 0 & 0 & < & \omega & < & J^* \\ 1 & & & \omega & = & J^* \\ 0 & J^* & < & \omega & < & 2J^* \\ 1 & & & \omega & = & 2J^* \\ -1 & & & \omega & = & 3J^* \\ 0 & 3J^* & < & \omega & < & 4J^* \\ -1 & & & \omega & = & 4J^* \\ 0 & 4J^* & < & \omega & < & 5J^* \\ -1 & & & \omega & = & 5J^* \end{cases} \tag{2.82}$$

El dominio de la ecuación 2.82 es $\omega = [0, 5J^*]$. El dominio de $G(\omega) = (\delta * H)(\omega)$ es, por tanto, $\omega =$ [resolución, $(n-1)J^*$ + ventana espectral].

Las funciones δ son simétricas con respecto al centro, es decir que se tiene un número par de picos (n). Debido a ello, el multiplete resultante de la convolución se encontrará en el centro de su dominio, independientemente de si $H(\omega)$ se encuentra centrado. La cantidad de picos es variable, pero entre más sean, la separación entre el multiplete reducido en el centro y los submultipletes en los extremos será mayor.

La determinación del valor de J^* que resultará en la reducción del multiplete, se determina con ayuda de la gráfica de la función que denominaremos *función de integral*, y que está definida como:

$$f(J^*) = \sum_{J^*=J^*_{\min}}^{J^*_{\max}} |\delta(\omega) \otimes H(\omega)|$$

donde $J^*_{\min}$ y $J^*_{\max}$ son el valor de la resolución y la ventana espectral de $H(\omega)$, respectivamente. $f(J^*) > 0$ en todo su intervalo, debido a que es la suma del valor absoluto de $\delta(\omega) \otimes H(\omega)$. Sin embargo, cuando $J^* = J_{IS}$, se obtienen valores que son mínimos locales fáciles de identificar. Para ese caso, el valor de $G(\omega)$ entre el multiplete reducido y los submultipletes en los extremos, tenderá a cero. La profundidad de los valores mínimos de $f(J^*)$ dependen principalmente de la cantidad de picos de la función δ.

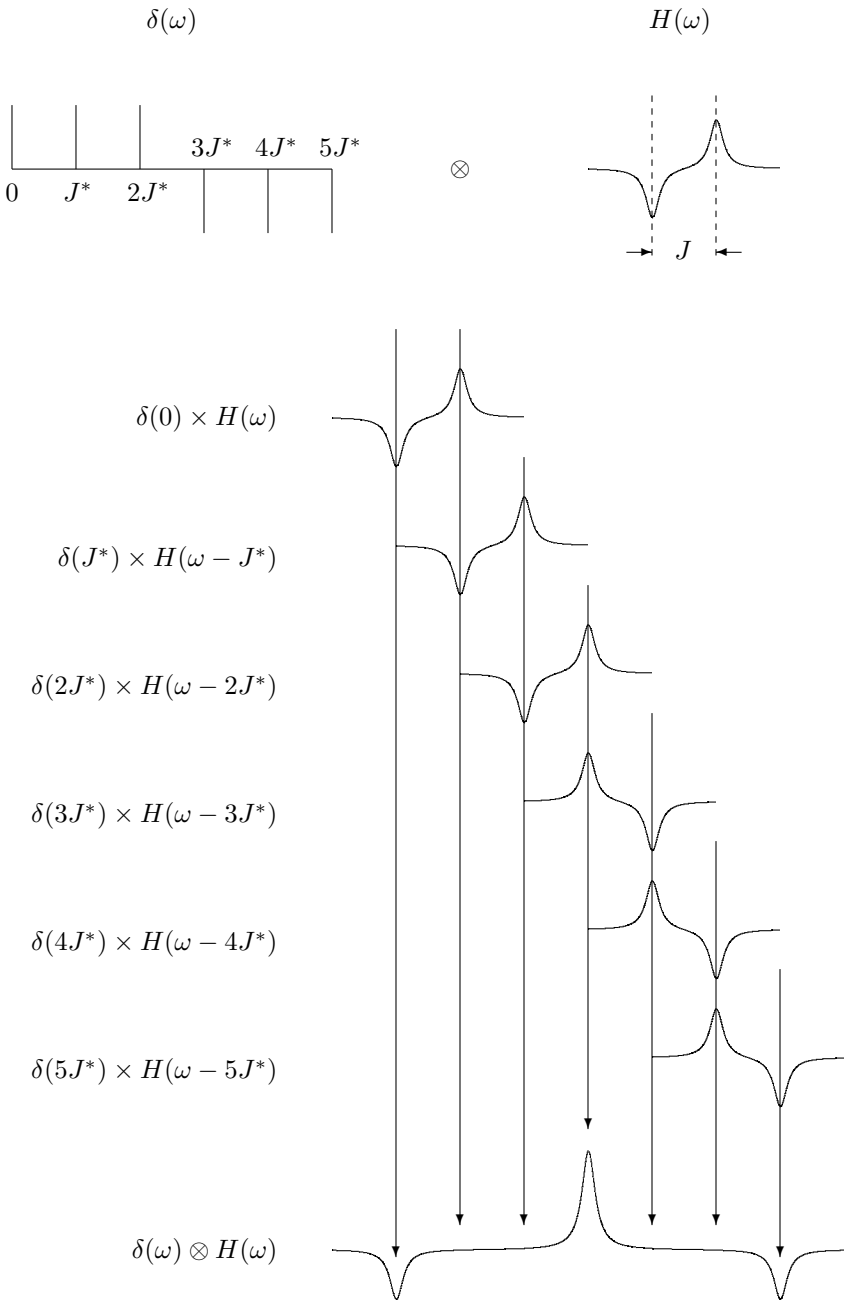


Figura 2.8: Proceso de deconvolución paso a paso de una señal en antifase cuando $J^* = J$, utilizando una función δ de seis picos.

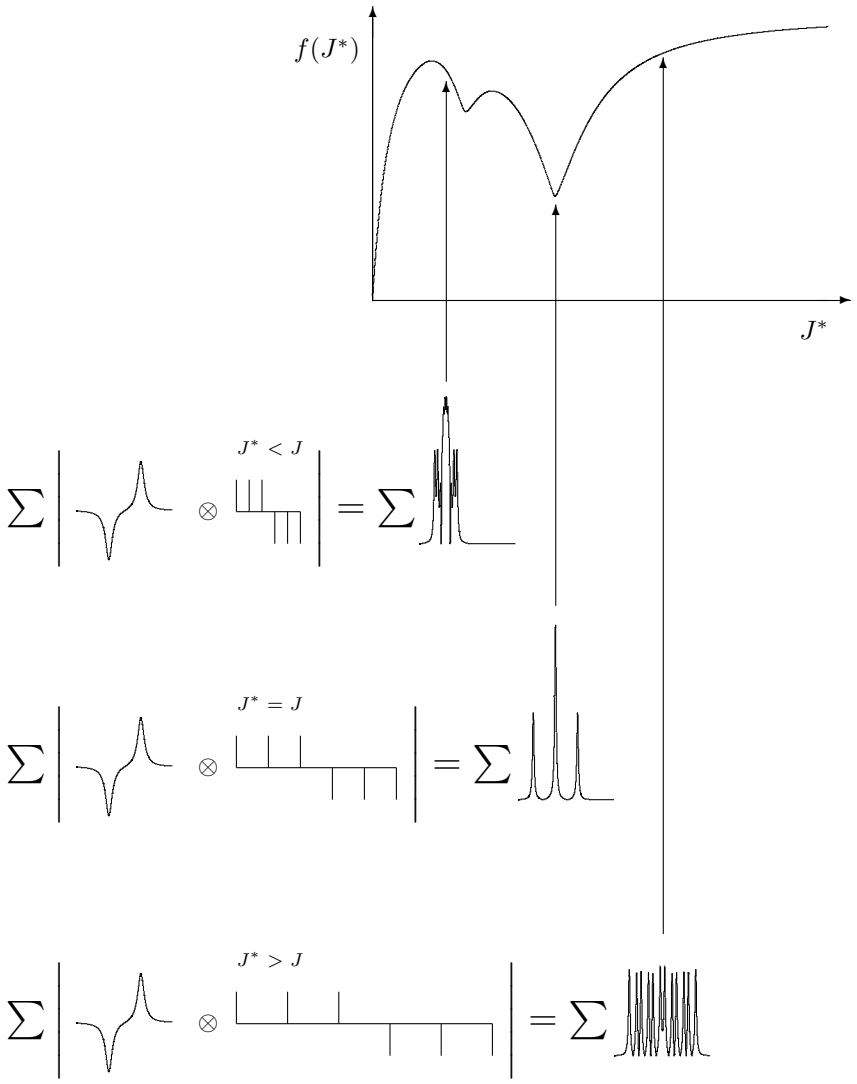


Figura 2.9: Gráfica de integral de un multiplete en antifase utilizando una función δ de seis picos. $f(J^*) = \sum |\delta(\omega) \otimes H(\omega)|$, para $J^* = [J^*_{min}, J^*_{max}]$.

2.5. Programas de computadora

Cuando se desea desarrollar una aplicación o programa de computadora, sin importar cuál sea su función, es necesario decidir sobre qué plataforma será ejecutado, pues ello determinará el lenguaje de programación a emplear y su estructura. La parte medular de esta plataforma es el sistema operativo, estando *Windows*, *linux* y *MacOs* entre los más populares. Cuando el programa a desarrollar no necesita una interfaz gráfica,

es decir que la interacción con el usuario será través de la línea de comandos (llamados *programas de consola*), la compatibilidad entre sistemas operativos es muy alta. Es decir que resulta relativamente sencillo compilar e instalar un programa de este tipo. Sin embargo, cuando se desea hacer una aplicación gráfica, con una interfaz más amigable para el usuario, la compatibilidad entre los diferentes sistemas operativos disminuye considerablemente.

Si una aplicación a desarrollar quiere instalarse en diferentes plataformas, se tienen diversas alternativas:

1. Compilar la aplicación en cada plataforma
2. Utilizar un motor de ejecución que pueda instalarse en cada plataforma de modo que la interacción de la aplicación sea con el motor y no con el sistema operativo
3. Crear una aplicación cliente-servidor, siendo el cliente un programa local muy compatible en las distintas plataformas, y el servidor un programa instalado en una plataforma única

2.5.1. Ejecutables en cada plataforma

Compilar un programa, es decir, traducirlo de un lenguaje de programación al lenguaje binario legible para el sistema operativo (llamado "código máquina"), en general es una tarea sencilla: se selecciona el lenguaje de programación de nuestra preferencia, se instala su compilador y las bibliotecas requeridas. Conforme se requieren más recursos, como una base de datos, una interfaz gráfica, o tiempo de procesamiento para cálculos matemáticos complejos, se va haciendo necesario instalar más bibliotecas de funciones. Cuando una aplicación requiere de muchos recursos, el problema que se presenta cuando se desea exportarla a otra plataforma, es la disponibilidad de dichos recursos.

Supongamos que deseamos crear un programa con una interfaz gráfica, es decir, una aplicación que utilice ventanas, botones, cuadros de texto, menúes, etc. Existen algunas bibliotecas que proporcionan estos elementos, como *qt* o *gtk*, y que pueden utilizarse con diversas plataformas de desarrollo. Del mismo modo, si la aplicación requiriera de una base de datos como *MySQL* o *PostgreSQL*, sería necesario instalar

el servidor de bases de datos deseada, y una biblioteca de funciones para comunicarse con ella desde la aplicación.

Si se seleccionan recursos compatibles entre las diferentes plataformas en las que se desea utilizar una aplicación, es posible compilar e instalar ésta de una forma relativamente sencilla. Sin embargo, existen ciertas desventajas de este esquema:

- Es necesario instalar los recursos que requiere la aplicación (bibliotecas de funciones, servidor de ventanas, motor de base de datos, etc.) en cada plataforma
- Se debe contar con los sistemas operativos en los que se quiere instalar, ya sea en diferentes computadoras, en particiones especiales, o en máquinas virtuales.
- La aplicación debe compilarse en cada plataforma cuyo sistema operativo es distinto. Comúnmente esto implica a su vez el tener que crear diferentes programas de instalación.
- En caso de que las bibliotecas de funciones utilizadas sean diferentes en diferentes plataformas (debido a su actualización), es necesario hacer también distintas versiones de la aplicación de acuerdo a esas variaciones.

En resumen, si bien el desarrollar una aplicación para diferentes plataformas bajo este esquema permite tener una interfaz más flexible, los procesos de creación e instalación podrían requerir de mucho tiempo, tanto para el desarrollador como para el usuario encargado de su instalación.

2.5.2. Motores de ejecución

Con la intención de resolver el problema de la compatibilidad entre las distintas plataformas, siendo el sistema operativo el más importante, se han creado motores de ejecución. Los motores de ejecución son programas que se instalan fácilmente cuya función es servir de intermediario entre la aplicación y la plataforma sobre la que está instalado. Existen dos tipos principales de motores de ejecución, los que interpretan directamente algún lenguaje de programación, o aquellos que hacen uso de un lenguaje

intermedio compilado. En el caso de programas interpretados, el proceso de traducción puede ser muy lento, lo cual a su vez se ve reflejado en el desempeño de la aplicación. Aunque este problema se resuelve en gran medida conforme se utilizan computadoras más poderosas, también la complejidad de los programas tiende a aumentar. En el caso de que se utilice un lenguaje precompilado, la rapidez es, de cualquier modo, menor que en un programa compilado específicamente para el sistema operativo debido a que no se tiene que hacer uso de dicho motor de ejecución.

Además de la rapidez de ejecución, otra desventaja de esta modalidad es la necesidad de instalar el motor de ejecución para poder utilizar la aplicación. Los motores más extendidos son *Java*, desarrollado por *Sun Microsystems* y la plataforma *.NET* de *Microsoft*, siendo el *Proyecto Mono* un grupo de herramientas basadas en *GNU/linux* compatibles con *.NET*.

2.5.3. Aplicaciones cliente-servidor

La arquitectura de este tipo de aplicaciones consiste en un programa *cliente* que hace peticiones a otro programa denominado *servidor*. Este esquema se aplica tanto en una sola computadora como en una red de computadoras. Por ejemplo, en unix el sistema de ventanas X es un servidor que proporciona los recursos gráficos a programas cliente, los cuales pueden estar ejecutándose en la misma computadora o en computadoras remotas.

Con el crecimiento de la internet, los servidores de páginas web están entre los programas más utilizados. Su función es proporcionar hipertexto a los clientes (llamados exploradores o navegadores web), el cual es un formato de archivo especial que tiene referencias cruzadas a otros archivos (denominados *hipervínculos*). Es importante mencionar que el hipertexto hace referencia no sólo a texto plano, sino también a imágenes, formularios, botones y a objetos incrustados de sonido e incluso vídeo. El protocolo de comunicación entre el servidor de páginas web y los navegadores se llama *protocolo de transferencia de hipertexto* o HTTP (por sus siglas en inglés, Hypertext Transfer Protocol). El tipo de hipertexto más comunmente transferido a los navegadores es *lenguaje de marcado de hipertexto* o HTML (Hypertext Markup Language), con

el cual están diseñados la mayor parte de los sitios web. Sin embargo, los archivos en HTML contienen información "estática", por lo que una vez interpretado por los navegadores, la interacción con el usuario es prácticamente nula. Para crear contenido activo, se ha implementado en los navegadores la posibilidad de interpretar otro lenguaje llamado *JavaScript*. Este lenguaje orientado objetos permite modificar y leer las propiedades y eventos asociados con cada uno de los objetos del documento desplegado en la ventana del navegador, que es conocido como Modelo de Objetos del Documento (DOM, Document Object Model), así como ciertos objetos asociados directamente con el navegador. Es importante mencionar que, tanto el texto en código HTML como el texto en JavaScript son leídos del servidor una vez y conservados en el cliente durante el tiempo de vida de la página web.

Los dos programas más usados como servidores web son *Apache*, de la *Apache Software Foundation* y *IIS* de *Microsoft*, mientras que los exploradores más populares son *Firefox*, *Mozilla*, *Microsoft Internet Explorer*, *Opera*, *Google Chrome* y *Safari*, aunque existen otros.

Además de la transferencia de hipertexto, los servidores web pueden ejecutar un tipo de programas conocidos como *aplicaciones web*. Este tipo de programas son distintos a los programas con interacción contínua entre la interfaz del usuario (ver figura 2.10), en los cuales es el sistema operativo el encargado de crear y mantener activo un proceso para cada instancia del programa ejecutado por el usuario. Mientras el proceso no sea destruido, el programa envía y recibe datos del usuario a través de su interfaz de manera casi contínua (ello depende esencialmente del tiempo asignado a todos y cada uno de los procesos que ejecuta la computadora). En una aplicación web, por el contrario, la interfaz (el navegador web) no tiene una conexión directa con el programa en sí. Más aún, la aplicación ni siquiera es un proceso vivo en el lado del servidor, sino que es ejecutado por el servidor web cada vez que se requiere, esto es, cada que el usuario desde el navegador le pide información. Esto significa que una aplicación web debe recibir parámetros definidos del lado del cliente, para enviar de regreso información distinta cada vez que es ejecutado.

Los clientes solicitan información al servidor de páginas web por medio de lo que se conoce como

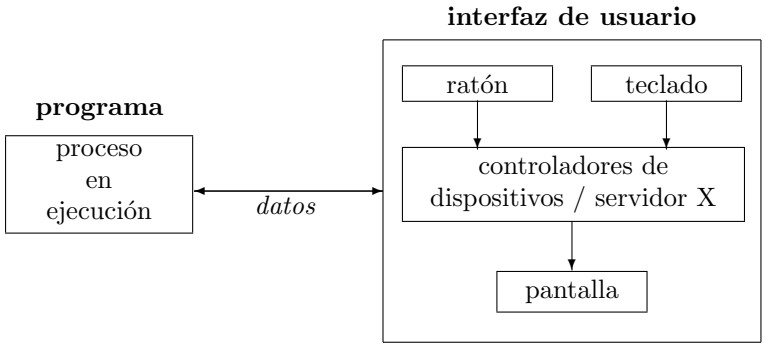


Figura 2.10: Arquitectura de una aplicación con interacción contínua con la interfaz del usuario. El sistema operativo crea un proceso para una instancia del programa el cual, mientras no sea destruído, enviará datos a los dispositivos de salida como la pantalla, y recibirá información de los dispositivos de entrada como el ratón y el teclado.

interfaz de entrada común o CGI (Common Gateway Interface). Las aplicaciones en el lado del servidor se conocen como CGI's y generan un objeto MIME, o extensión mutipropósito de correo de internet (Multipurpose Internet Mail Extension), enviado a la salida estándar. El servidor captura estos datos, le agrega información propia y la envía al cliente. Las extensiones MIME son la forma de transferir todo tipo de achivos de modo transparente para el usuario: archivos de texto plano, archivos en codificación distinta de ASCII, todos los archivos binarios como son los ejecutables, vídeos, archivos de sonido, presentaciones, etc. Es responsabilidad de las CGI's el definir el tipo MIME que se enviará al cliente mediante el servidor de páginas web. Los CGI's son programas binarios que deben compilarse e instalarse, y pueden estar escritos en muy diversos lenguajes de programación, siendo los más habituales C, C++, Perl, Java, Visual Basic, e incluso PHP (como CGI).

Además de los CGI's, existen otro tipo de aplicaciones web que también son ejecutados en el lado del servidor. La diferencia básica entre los CGI's y estas otras aplicaciones es que no son programas compilados y ejecutados por el sistema operativo a petición del servidor de páginas web, sino que son interpretados por módulos instalados directamente en el servidor de páginas web. Esto es así porque el código en estos lenguajes se incrusta en los archivos HTML, de modo que la interpretación ocurre antes de ser enviados al cliente.

Debido a que tanto en los CGI's como en los programas interpretados, el proceso ocurre del lado del servidor, dichos programas utilizan los recursos de la computadora en la que se ejecutan (memoria, espacio en disco para archivos, bases de datos, etc.), por lo que el rendimiento de la aplicación no depende del cliente (ver Figura 2.11).

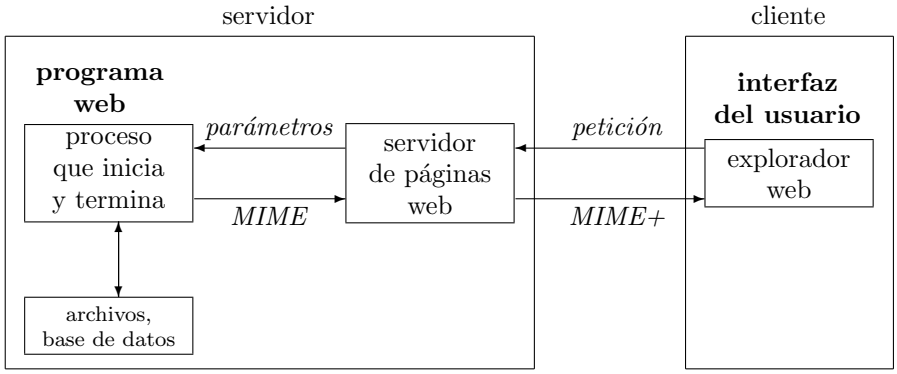


Figura 2.11: Arquitectura de una aplicación web. Los navegadores web cliente hacen peticiones al servidor de páginas web el cual, a su vez, ejecuta la aplicación web. El objeto MIME creado por la aplicación web es enviado de regreso al cliente con información extra (MIME+).

Ventajas de las aplicaciones web:

- El usuario final no tiene que instalar nada si ya cuenta con un explorador web y, en el caso de estar en una computadora remota, una conexión a la internet.
- Cualquier actualización es transparente para el usuario.
- No es necesario compilarlas e instalarlas en diferentes plataformas
- Los recursos físicos (memoria, espacio en disco) de los clientes no son un factor importante en su desempeño, el cual depende de los recuros del servidor y de la rapidez en la conexión al servidor web.

Sus principales limitantes:

- La interfaz con el usuario está restringida a las capacidades de los exploradores web.

- El intercambio de datos entre el servidor y el cliente depende de la disponibilidad y rapidez de la conexión entre ellos.

Capítulo 3

nuup

Debido a la disponibilidad de una computadora como servidor de recursos, tanto de cálculo como de datos, es que se seleccionó una aplicación web como arquitectura para implementar el método modificado de duplicación de J. El nombre de la aplicación web es *nuup*, que signfica "acoplar" o "unión" en lenguaje maya, y lo utilizaré de aquí en adelante para referirme a ella.

3.1. Estructura de archivos

Una práctica común en el desarrollo de aplicaciones consiste en dividirlas en archivos de funciones más pequeños, denominados *bibliotecas de funciones*, de modo que la depuración de errores sea mucho más sencilla. Estos archivos de funciones podrían ser genéricos o venir de otros programas; incluso archivos creados para una aplicación específica podrían ser empleados posteriormente en otros programas. La programación de *nuup* sigue este esquema, y por tanto, hace uso de bibliotecas de funciones tanto externas como propias. Entre las bibliotecas de funciones *externas* están aquellas encargadas del manejo de parámetros HTML (`htmlUtils`, `htmlParametros`, `htmlGalletas`), de la interacción con la base de datos PostgreSQL (`pqOperaciones` a través de `libpq`), y la biblioteca encargada del manejo de datos de espectros de RMN (`quimDatos`). Las bibliotecas de funciones *internas* son las que están encargadas de la creación de la inter-

faz de *nuup* con el usuario. La figura 3.3 muestra los principales archivos utilizados por *nuup*, siendo las palabras dentro de cuadros la representación de dichas bibliotecas (las bibliotecas externas se encuentran fuera del recuadro más grande de línea gruesa y las bibliotecas internas dentro de éste.) Es importante mencionar que, a diferencia de las bibliotecas internas, para hacer uso de las bibliotecas externas *no* es necesario contar con el código a partir del cual fueron creadas, basta con conocer el *prototipo* de las funciones que contiene y el código binario de la biblioteca en sí. En sistemas Windows, estas bibliotecas suelen tener extensión `.dll`, mientras en sistemas unix es común encontralas con las extensiones `.so` o `.a`. El prototipo de las funciones está formado por el nombre de la función y los parámetros que recibe y devuelve. Dependiendo del lenguaje de elección, los archivos que contienen los prototipos pueden ser distintos.

Se seleccionó *Lenguaje C* para desarrollar *nuup* porque al tener una plataforma *linux*, también se tenía disponible *gcc 4*, un compilador de C libre y gratuito. La opción de sistema operativo ofrecía, además, todas las herramientas necesarias para el desarrollo una aplicación web: un servidor de páginas web (Apache 2) y una de base de datos (PostgreSQL 8.4). Las bibliotecas representadas en la figura 3.3 implican la existencia tanto de la biblioteca de funciones en sí, como de los archivos de encabezado. Muchas de las bibliotecas de funciones externas también han sido programadas por mi en lenguage C, por lo que la extensión que las caracteriza es `.o` y no `.so` como es común en sistemas unix. De este modo, el "quimDatos" representa a la biblioteca de funciones `quimDatos.o` y el archivo de encabezado `quimDatos.h`, y que es la encargada de la manipulación de archivos de datos de espectros de RMN. Del mismo modo, "htmlUtils", "htmlParametros", "htmlGalletas", son la representación de `htmlUtils.o` y `htmlUtils.h`, `htmlParametros.o` y `htmlParametros.h` y `htmlGalletas.o` y `htmlGalletas.h`; estas bibliotecas contienen funciones dedicadas a la manipulación de parámetros HTML provenientes del servidor de páginas web, así como funciones encargadas de la creación de objetos MIME, por mencionar algunas. La biblioteca "pqOperaciones", es decir, `pqOperaciones.o` y `pqOperaciones.h`, realiza transacciones con la base de datos PostgreSQL a través de `libpq.so` con `libpq.h` como archivo de encabezado.

El objetivo de los archivos de encabezado (los archivos con extensión `.h`) es declarar los prototipos de las funciones que se utilizarán en el código escrito en los archivos `.c`. También es en estos archivos de encabezado que se declaran los prototipos de variables globales y tipos de datos nuevos, así como valores constantes que serán reemplazados por el compilador (declarados con `#define`). En esencia, es a través de los archivos de encabezado que se establece la estructura lógica de *nuup*.

Es en los archivos de código (archivos con extensión `.c`) que se define explícitamente el comportamiento de las funciones. Cualquier función que se desee usar en un archivo `.c` distinto a aquel en que se define deberá tener una referencia a su prototipo escrito en algún archivo `.h`. Si bien es cierto que todo el código de *nuup* podría haberse escribirse en un sólo archivo (por ejemplo `nuup.c`), la utilidad de dividirlo en múltiples archivos radica en la facilidad de depuración de errores, como se había mencionado antes.

En lenguaje C, un programa está estructurado de principio a fin en funciones, siendo `main` la función inicial. Es en esta función que se ejecuta la cascada de funciones definidas en todos los archivos de código que conforman el programa. Para transferir información de función en función, cuando no es posible utilizar una lista de parámetros por ser demasiado larga, se han creado estructuras de datos. Estas estructuras de datos pueden denominarse "objetos" debido a que a través del nombre de variable del tipo nuevo que define, es posible acceder a toda la serie de datos que contiene y que pueden verse como "propiedades del objeto". Uno de estos objetos, declarado en `osUtils.h`, el archivo de encabezados de la bibilioteca `osUtils.o`, es `ou_datos` y está definido como:

```
typedef struct {
  // Los parámetros
  HP_2 hp2ID;
  HP_2 hp2ACCION;
  HP_2 hp2IDIOMA;
  HP_2M hp2mFUNCIONES_G;
  HP_2M hp2mFUNCIONES;
  // La lista de objetos...
```

```
  nlista lstOs;
} ou_datos;
```

hp2ID, hp2ACCION, hp2IDIOMA, hp2mFUNCIONES_G y hp2FUNCIONES son variables de tipo hp2_2 y HP_2M, tipos de datos declarados a su vez en el archivo htmlParametros.h de la biblioteca htmlParametros.o y cuya definición es:

```
typedef struct {
  char achrNombre[HC_MAX_NOMBRE];
  char achrValor[HC_MAX_VALOR];
} HP_2;


typedef struct {
  char achrNombre[HC_MAX_NOMBRE];
  char achrValor[HC_MAX_MSJ];
} HP_2M;
```

Del mismo modo, la variable lstOs está declarada en UtilsLista.h:

```
typedef struct {
  lst_nodo *Nodo;
  int intNElementos;
  int intElementoActual;
} nlista;
```

Podemos observar en la declaración de todos estos tipos de datos que se trata de estructuras de datos anidadas y que albergan, en el nivel más profundo, tipos de datos básicos del lenguaje C como son cadenas de caracteres (char), números enteros (int), etc. Otra cosa que es importante notar es que, a medida que los programas usan mayor cantidad de tipos distintos definidos en las diversas bibliotecas de funciones (en sus archivos de encabezado), así como de las funciones mismas, se hace necesario tener una forma de

identificar en dónde está su declaración. Esto se hace utilizando prefijos tanto en los tipos de datos y como en las funciones en las que se utilizan. En la declaración de `ou_datos`, el prefijo "ou" indica que se trata de un tipo utilizado en las funciones de la biblioteca `osUtils.o`, "HP" un tipo de datos constante de la biblioteca `htmlParametros.o`, "nlista" una estructura para almacenar datos en forma de lista doblemente ligada, etc.

La variable declarada en `nuup.c` (ver el apéndice B.2) de tipo `ou_datos` y que se llama `ou` contiene datos para crear un objeto HTML denominado *formulario* (`Form`) y que a su vez sirve de contenedor para otros objetos especiales llamados *controles* (botones, cuadros de texto, casillas de verificación, etc.) Los diferentes formularios en un documento HTML tienen un nombre que los identifica (`ou.hp2ID`), la acción a realizar (`ou.hp2ACCION`) y que en el caso de *nuup* es el programa ejecutable `nuup.cgi`, un parámetro que indicará qué archivos serán enviados al servidor de páginas web de acuerdo al idioma seleccionado (`ou.hp2IDIOMA`) y la referencia a archivos en lenguaje Javascript que contienen funciones que se utilizarán en los diferentes archivos HTML para interactuar de manera dinámica con el usuario por medio del navegador web (`ou.hp2mFUNCIONES_G` son archivos que son utilizados por todos los documentos HTML, mientras `ou.hp2mFUNCIONES` contendrá la referencia a archivos Javascript específicos para el documento HTML enviado de acuerdo a los requerimientos del usuario.) Es en la lista `lstOs` que se guarda un objeto genérico que contendrá toda la información necesaria para crear los elementos gráficos desplegados en el navegador web (el objeto MIME).

La creación de los objetos gráficos mostrados en el navegador son los diferentes formularios creados para un propósito específico y, por tanto, contendrán controles también específicos. El programa *nuup* está diseñado como un programa tipo asistente, en los cuales el proceso que realizan está dividido en una serie de pasos hasta su culminación. Cada formulario representa un paso distinto en el proceso de deconvolución de un multiplete y está diseñado en un archivo HTML. Sin embargo, estos archivos HTML necesitan ser modificados antes de ser enviados al cliente para reflejar la información obtenida de un espectro de resonancia. Para lograrlo, estos archivos HTML contienen etiquetas que serán reemplazadas por

nuup. Son estas etiquetas y el texto que será reemplazado lo que se guarda en el objeto `ou`. Si observa la figura 3.1, los archivos HTML en donde están diseñados los formularios se encuentran encerrados en cajas con bordes circulares. Por ejemplo, el archivo `onuupPaso2-es.html` contiene el diseño del formulario mostrado en la figura 4.2, en español. El paso previo a la lectura del archivo es el de definir los valores de todas las etiquetas que contiene, como es el nombre del archivo de trabajo, los nombres de los archivos de sesión, los botones, etc. Todo el código que realiza ésto para el paso dos, está contenido en los archivos `onuupPaso2.c` y `onuupPaso2.h`. La biblioteca de funciones que tiene la decisión de cuál paso se desplegará es `osAcoplamiento`. Debido a que el paso 1 de *nuup* no requiere muchas líneas de código, la función que define los valores de las etiquetas que requieren ser cambiadas en los archivos del formulario del paso 1 se encuentra también en el archivo `osAcoplamiento.c`.

El programa *nuup* organiza los archivos de datos por usuario, para lo cual es necesario un mecanismo de autenticación de usuarios. La información de los usuarios es almacenada en una base de datos de PostgreSQL y se tiene acceso a ella a través de un formulario creado por la biblioteca `osInicioSesion`. La decisión de si el formulario a mostrar corresponde a la ventana de inicio de sesión o a alguno de los pasos del asistente es tomada en la biblioteca de funciones `interfaz`.

Una vez que se ha definido si crear un objeto de inicio de sesión, o un objeto definido por la estructura `onuup_datos` en el archivo `osAcoplamiento.h` que representa uno y sólo uno de los pasos antes mencionados, la función `ou_hazlo(&ou)` en la función `main` (archivo `nuup.c`, sección B.2) lee el archivo HTML adecuado y reemplaza todas las etiquetas por los valores almacenados en el objeto `ou` en memoria (específicamente en la variable a la que apuntan cada uno de los objetos en la lista `lstOs`, que para el caso de *nuup* es sólo una). El paso final de la función `main` es leer el archivo `index.html` (sección B.5) y reemplazar las etiquetas `%OU_FUNCIONES_G%`, `%OU_FUNCIONES%` por la referencia a las funciones declaradas en archivos Javascript (ver figuras 3.2 y 3.3), así como la etiqueta `%ONUUP1%` por todo el código HTML en memoria y que corresponde al formulario mostrado en el explorador web.

En este punto, a la función `main` lo único que le queda por hacer es liberar el espacio en memoria

utilizado, y una vez hecho (funciones `ou_libera` y `hp_libera`), el proceso disparado al ejecutar el archivo `nuup.cgi` es eliminado por el sistema operativo. Esto significa que la interacción de *nuup* con el usuario pasa a ser responsabilidad del navegador web desde el momento en que el servidor de páginas web le envía el código HTML (habiendo hecho las modificaciones pertinentes en el proceso.) Es ésta la razón de incluir en el código HTML una serie de instrucciones en lenguaje Javascript: es la forma de interactuar dinámicamente con el usuario a través del navegador web. Existe una forma alterna de incluir código Javascript en archivos HTML y consiste en escribirlo en archivos separados y leerlos de este modo (el código de uno de los archivos que hacen este enlace está en la sección B.6):

```
<script Language="JavaScript" src="/aplicaciones/nuup/os/js/onuup.js"></script>
```

donde "onuup.js" es el nombre del archivo que contiene código Javascript. Las figuras 3.2 y 3.3 muestran la relación de los archivos Javascript con los archivos HTML de cada paso. Puede observarse que distintos archivos HTML usan un mismo archivo de código Javascript, esto es porque las funciones contenidas en estos archivos son genéricas y realizan una función específica. Así, `dom-drag.js` contiene las funciones que provocan que se pueda arrastrar la ventana del espectro en miniatura, mientras `onuupF.js` contiene las funciones para mostrarla u ocultarla (ver figuras 4.3 y 4.4), `onuupI.js` alberga funciones para la colocación inicial de las imágenes de la interfaz y `onuup.js` funciones muy generales como la de detectar el tipo de navegador web del usuario.

3.1.1. La biblioteca de funciones quimDatos

El objetivo principal de *nuup* fue la implementación del método modificado de duplicación de J. No obstante, para poder extraer la información necesaria a partir de los archivos de datos de espectros de RMN, eran necesarias una serie de funciones que permitieran obtener el multiplete de estudio, es decir, la función $H(\omega)$ de la ecuación 2.79. La biblioteca `quimDatos` es una compilación de todas esas funciones y de otras encargadas de la deconvolución empleada en el método modificado de duplicación de J.

Si bien los archivos de datos están guardados en diferentes formatos dependiendo su origen o el tra-

tamiento que hayan recibido los espectros, todos contienen los datos de intensidad de los n trazos que contienen. Eso permite el que, independientemente del tipo de archivo, puedan extraerse dichos datos en una estructura genérica definida como sigue:

```
typedef struct {
  double         *dd[DRMN_DIMS];
  drmn_info       info;
} drmn_datos;
```

donde la variable `*dd` es un apuntador a valores dobles. La macro `DRMN_DIMS` es reemplazada por el compilador por su valor tal como está definido:

```
#define DRMN_DIMS               3
```

y representa la dimensión del arreglo de números dobles que almacenará dicho apuntador. Por ejemplo, los archivos con formato MestRe-C contienen dos columnas de datos: el desplazamiento químico y la intensidad. En este caso, `dd[1]` contendrá la referencia al espacio en memoria en que se almacenarán los n números de la primera columna, mientras `dd[2]` se referirá a la intensidad.

El tipo de datos `drmn_info` definido como:

```
typedef struct {
  int             intDim;
  long int        lNP;
  long int        lNT;
  long int       *llMinimos[DRMN_DIMS];
  long int       *llMaximos[DRMN_DIMS];
  double          ddLI[DRMN_DIMS];
  double          ddLS[DRMN_DIMS];
  double          ddLI_PPM[DRMN_DIMS];
  double          ddLS_PPM[DRMN_DIMS];
```

```
  double          ddLI_HZ[DRMN_DIMS];
  double          ddLS_HZ[DRMN_DIMS];
} drmn_info;
```

contendrá toda la información relacionada con esos datos, como es la cantidad de columnas o dimensión (`intDim`), la cantidad de puntos (`lNP`) y de trazos (`lNT`), la referencia en puntos a los valores de intensidad máximo y mínimo (`llMinimos` y `llMaximos`), así como los valores máximo y mínimo para cada dimensión en diferentes unidades. Algunos de estos valores se determinan en el momento de leer el archivo de datos, lo cual no siempre es necesario. Esto es, es posible conocer gran parte de la información que contiene un archivo de datos leyendo sólo una parte. Estos datos se guardan en el siguiente tipo:

```
typedef struct
{
  char            strRuta[UA_TAM_MAX];
  FILE           *aFOrigen;
  DRMN_TIPO_ARCH  tipo;
  long int        lInicioDatosB;
  void           *encabezado;
} drmn_arch;
```

haciendo uso de las siguientes funciones:

```
drmn_arch *
drmn_inicializaArch   (drmn_arch      *,
                       const char     *ruta_archivo,
                       DRMN_TIPO_ARCH  tipo);


nbool
drmn_abreArch         (drmn_arch *);
```

donde `DRMN_TIPO_ARCH` es un valor entero y corresponde a los tipos de archivos soportados por *nuup*:

```
#define DRMN_TIPO_ARCH    int

#define DRMN_XYZ          1

#define DRMN_MESTRE       2

#define DRMN_XEASY        4

#define DRMN_VARIAN       8

#define DRMN_DESCONOCIDO  DRMN_XYZ+DRMN_MESTRE+DRMN_XEASY+DRMN_VARIAN
```

Nótese que la macro `DRMN_DESCONOCIDO` es la suma de todos los tipos de archivos. De ser utilizado este valor para el miembro `tipo` del objeto `drmn_arch`, la función `drmn_abreArch` intentará determinar el tipo de datos del archivo. Aunque puede ser de utilidad, cuando se conoce el tipo de datos del archivo, el proceso de apertura del archivo es más rápido y es recomendable. La función `drmn_abreArch` recibe como parámetro un apuntador a una variable de tipo `drmn_arch` que, compo puede verse arriba, contiene cinco miembros: `strRuta` es la ruta de acceso al archivo de datos, `aFOrigen` es un apuntador que contendrá la referencia al archivo una vez abierto, `tipo` es el formato del archivo, `lInicioDatosB` es el desplazamiento en bytes hasta el inicio de los valores numéricos y que es a partir de donde será leído el archivo para llenar una variable de tipo `drmn_datos` y, finalmente, `encabezado`, la cual es otra variable de tipo apuntador que contendrá datos específicos del archivo de acuerdo a su tipo. Esta última variable es un apuntador de un tipo de datos nulo, `void`, debido a que los datos contenidos en los archivos de datos varían entre los diferentes formatos. Una vez abierto el archivo, la variable `encabezado` apuntará a uno de los tipos de datos siguientes:

```
typedef struct
{
  // Encabezado de archivo
  drmn_archVarian  av;       // Tipo de archivo de varian
  long             lNB;      // BLOQUES
  long             lNT;      // TRAZOS por BLOQUE
  long             lNP;      // PUNTOS por TRAZO
```

```
  long             lNE;      // Encabezados por bloque
  long             lBP;      // BYTES por PUNTO
  // Encabezado de bloque
  short    sEstado;
  short    sIndiceIni;
  short    sModo;
} drmn_Varian;
```

donde `drmn_arcVarian` es:

```
typedef enum
  {
    DRMN_VARIAN_DATA,
    DRMN_VARIAN_PHASE
  } drmn_archVarian;
```

Y son los datos extraídos por achivos de tipo Varian;

```
typedef struct
{
  char            strT[1024];    // Título o descripción
  drmn_verMestre  v;             // Versión del MestRe-C
  float           fFE;           // Frecuencia del espectrómetro
  float           fFB;           // Frecuencia más baja
  float           fVE;           // Ventana espectral
  long            lNP;           // Número de puntos
  float           fFD;           // Filtro digital
  long int        lInicioDatosL; // Línea en que inician los datos
  DRMN_ESC        escX;          // Escala en X
} drmn_MestRe;
```

con

```
typedef enum
  {
    DRMN_MESTRE_VIEJO,
    DRMN_MESTRE_NUEVO
  } drmn_verMestre;
```

contiene la información de archivos con formato MestRe-C;

```
typedef struct
{
  long       lNP;              // Número de puntos
  double     dLPP;             // Límite izquierdo (ppm)
  double     dRPP;             // Límite derecho (ppm)
  double     dLFF;             // Límite izquierdo (hz)
  double     dRFF;             // Límite derecho (hz)
  long int   lInicioDatosL;    // Línea en que inician los datos
} drmn_XEASY;
```

almacena la información proporcionada por archivos de XEASY; y

```
typedef struct
{
  long           lNP;              // Número de puntos
  long           lNT;              // Número de puntos
  char           strT[1024];
  long int       lInicioDatosL;    // Línea en que inician los datos
} drmn_XYZ;
```

se utiliza cuando se trata de archivos de texto plano que únicamente almacena valores numéricos.

Una vez que se lee un determinado archivo y se conoce la información y tipo de datos que contiene, es posible leer los datos en una variable de tipo `drmn_datos` con la función `drmn_leeDatos` cuyo prototipo es:

```
nbool
drmn_leeDatos     (drmn_datos *,
                   drmn_arch  *);
```

Cuando se tienen los datos de un multiplete en memoria, puede hacerse con ellos cualquier tratamiento, lo que incluye modificaciones directas en los valores, extracción de trazos, determinación de puntos de inflexión y, naturalmente, la convolución con una función δ. Cada familia de funciones δ, ya sea pares o impares, tienen regiones en los que su rango es cero. Para cuestiones de la programación de la deconvolución, las funciones δ (la ecuación 2.81 define una de las familias de funciones δ impares) pueden verse como funciones discretas y que pueden traducirse en arreglos de números enteros:

$$\begin{aligned} \delta_{\text{par}}[n] &= [\cdots, +1, -1, +1, -1, +1, +1, -1, +1, -1, +1, \cdots] \\ \delta_{\text{impar}}[n] &= [\cdots, +1, +1, +1, +1, +1, -1, -1, -1, -1, -1, \cdots] \\ \delta_{\text{impar}}[n] &= [\cdots, -1, -1, -1, -1, -1, +1, +1, +1, +1, +1, \cdots] \end{aligned} \tag{3.1}$$

donde n es la cantidad total de valores o picos. Las funciones que crean el espacio en memoria para almacenar dicho arreglo y luego liberarlo son:

```
nbool
drmn_creaFuncDelta (drmn_func_delta *,
                    int              n,
                    drmn_tipoDelta   tipo);

void
drmn_liberaFuncDelta (drmn_func_delta *);
```

donde `drmn_tipoDelta` y `drmn_func_delta` son:

```
typedef enum
  {
    DRMN_DELTA_SIM,
    DRMN_DELTA_ANTI_IZQ,
    DRMN_DELTA_ANTI_DER
  } drmn_tipoDelta;


typedef struct
{
  int  n;
  int *aiPicos;
} drmn_func_delta;
```

Una vez que se tiene definida función δ, es decir su tipo y cantidad de picos, la convolución se obtiene con la siguiente función:

```
nbool
drmn_dameConvolucion (drmn_datos      *convolucion,
                      drmn_datos      *datos_espectro,
                      drmn_func_delta *func_delta,
                      double           J);
```

donde `datos_espectro` contiene únicamente los datos del multiplete, y `J` es el valor de J^* o J de prueba y que sera también el valor de la posible J real, J_{IS}. Esta función calcula la convolución con dicho valor y devuelve el resultado en `convolucion`.

Para determinar el valor de J^* es de mucha utilidad crear la gráfica de integral definida por la ecuación 2.4.2. Los datos de esta gráfica los crea la función `drmn_creaSumaJPs`:

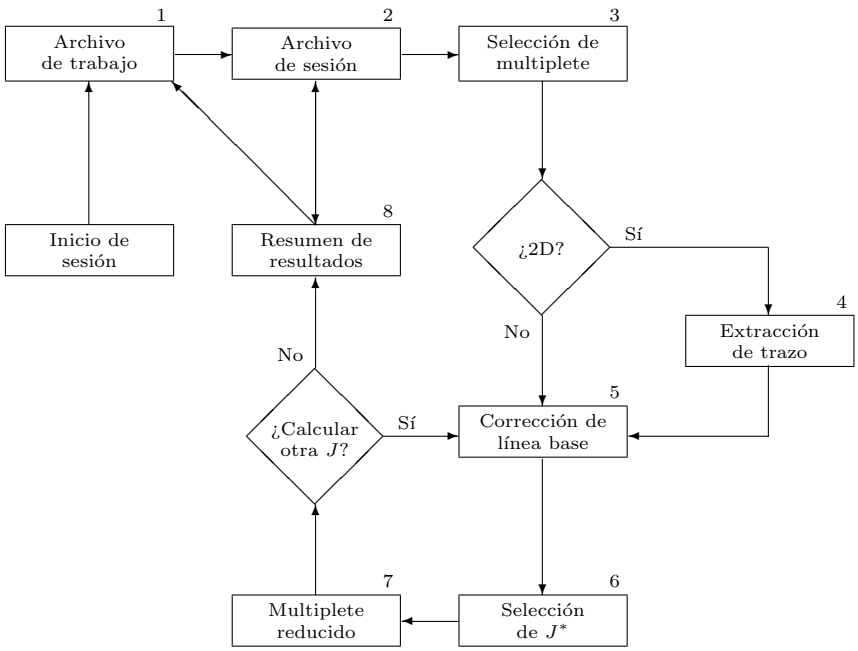
```
nbool

drmn_creaSumaJPs (drmn_datos      *suma,

                  drmn_datos      *datos_espectro,

                  drmn_func_delta *func_delta,

                  double           J_maxima);
```

en donde el valor mínimo de J^* está dado por la resolución de los datos del espectro.

Nota. El apéndice B muestra únicamente el código en el archivo inicial, `nuup.c`, y los archivos de encabezado más importantes. Esto se debe a que *nuup* es una aplicación con más de 28,000 líneas de código (lenguaje C, html, Javascript) que representarían unas $1,000$ páginas en papel.

3.2. Guía rápida del cálculo de J

La siguiente figura resume el procedimiento a seguir para calcular las constantes de acoplamiento en un espectro de RMN:



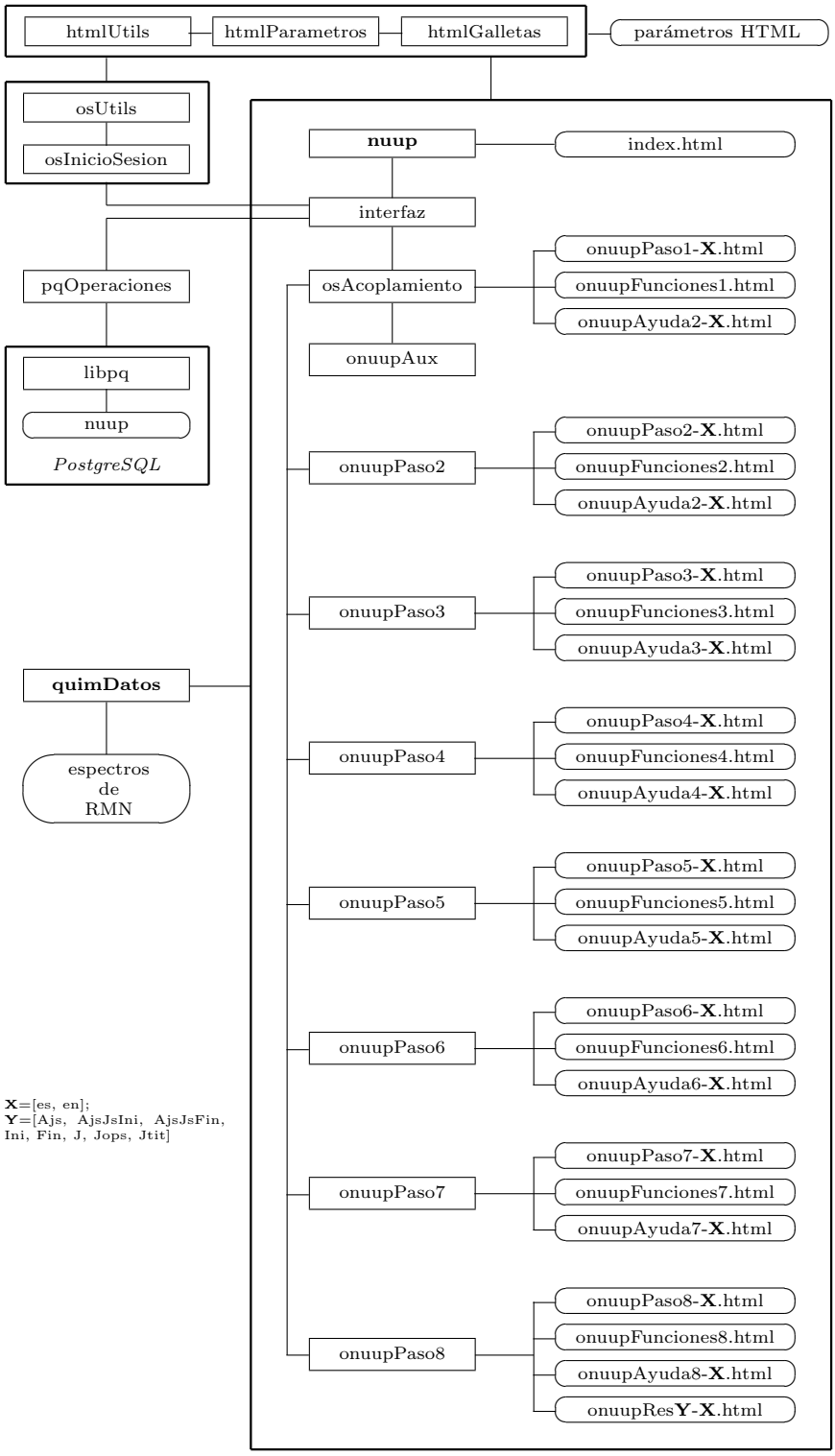


Figura 3.1: Estructura de archivos de *nuup*.

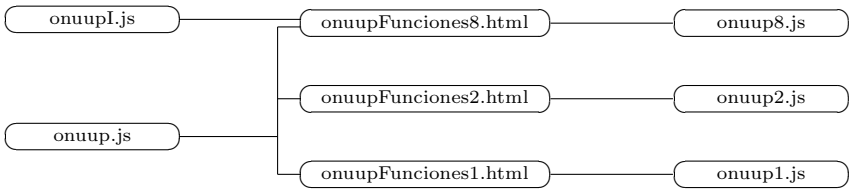


Figura 3.2: Archivos HTML que no requieren operaciones gráficas.

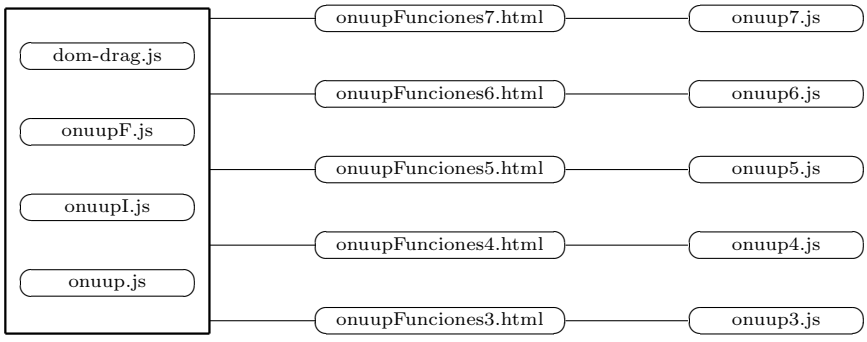


Figura 3.3: Archivos HTML que requieren de operaciones gráficas.

Capítulo 4

Resultados

4.1. Interfaz

La aplicación *nuup* tiene una interfaz de usuario tipo asistente de ocho pasos, más un formulario de inicio de sesión. Una vez que el usuario a ingresado su nombre de usuario y contraseña, se le muestran una serie de formularios numerados, cada uno de los cuales tiene un propósito específico.

4.1.1. Paso 1: Selección del archivo de trabajo

El nombre de usuario se utiliza para la creación de un directorio que contendrá únicamente los archivos utilizados por ese usuario. Esto significa que para poder calcular las diversas constantes de acoplamiento que pudiera tener un espectro, primero debe copiarse el archivo de datos que contiene dicho espectro en ese directorio, y para ello hay dos mecanismos. El primero consiste en enviar directamente el archivo que contiene los datos del espectro por medio del navegador web, lo que se hace por medio del cuadro de selección "Cargar archivo desde su computadora" (ver figura 4.1).

Cuando los archivos son pequeños ésta es una opción muy cómoda, sin embargo, podría no ser la mejor elección cuando se trata de archivos de algunos cientos de megabytes. Para estos casos es mejor utilizar

Figura 4.1: Interfaz de usuario en el paso 1, mostrando que se desea trabajar con el espectro del compuesto α-pineno.

la opción "Transferir archivo a mi directorio", en la cual es posible utilizar la dirección URL del archivo. *nuup* utiliza `wget -c` para descargar dicho archivo, y la ventaja de esto es que, en caso de interrupción, en un intento posterior el programa `wget` adicionará la parte faltante a la fracción ya descargada. La desventaja es la disponibilidad de un servidor público en la internet que soporte los protocolos HTTP, HTTPS o FTP para alojar el archivo. Un ejemplo es:

```
http://labna.iquimica.unam.mx/experimentos_rmn/b-pineno
```

Existe una tercera opción para subir archivos al directorio de trabajo del usuario que depende de la disponibilidad de una cuenta de usuario en la computadora en que está alojado *nuup*, `labna.iquimica.unam.mx`. Si se tiene una cuenta en esta computadora puede utilizarse alguno de los programas de transferencia de archivos seguros como `scp` (el cual usa `ssh` para la transferencia de archivos), o incluso `wget`. En labna existe el usuario *nuup* para este propósito así como para la actualización de contraseñas e información de los usuarios de la aplicación *nuup*, debido a que sus usuarios son virtuales, es decir, están únicamente en

una base de datos. Ya sea que se cuente con una cuenta de usuario en labna, o se utilice el usuario nuup, estos son unos ejemplos de cómo puede subirse un archivo desde una computadora remota:

```
scp b-pineno usuario@labna.iquimica.unam.mx:~/
scp b-pineno usuario@labna.iquimica.unam.mx:/home/usuario/
scp b-pineno nuup@labna.iquimica.unam.mx:~/
scp b-pineno nuup@labna.iquimica.unam.mx:/home/nuup/
```

Una vez que los archivos están en el servidor labna, basta con introducir la ruta en el campo "URL" (el uso de "file://" es opcional). Los tres ejemplos siguientes transfieren el mismo archivo al directorio de trabajo del usuario:

```
file:///home/usuario/b-pineno
/home/usuario/b-pineno
b-pineno
```

En el momento de ser transferidos los archivos al directorio de trabajo de *nuup*, al nombre se le agrega una extensión o se cambia la que ya tiene de modo que refleje el tipo de datos que contiene. Esto lleva a que un archivo diferente podría tener el mismo nombre de un archivo existente en el directorio de trabajo. Por ejemplo, si se tiene un archivo de nombre "datos.txt" de tipo MestRe, el nuevo nombre en *nuup* será "datos.mestre". Si en el directorio de trabajo ya existe un archivo con el mismo nombre, es decir "datos.mestre", el archivo no será reemplazado a menos que se haya activado la casilla "Reemplazar si ya existe".

Los archivos en el directorio de trabajo del usuario aparecen enlistados en la sección "En el directorio de trabajo". Una vez seleccionado el archivo de trabajo, se hace click en el botón "Siguiente>>" para pasar al paso 2.

4.1.2. Paso 2: Sesiones y resultados

El primer paso en la medición de las constantes de acoplamiento consiste en crear un "archivo de sesión". Toda la información relacionada con el multiplete de trabajo se guarda en estos archivos, los cuales son creados en este paso con la opción "Crear un archivo de sesión nuevo". Se pueden crear tantos archivos de sesión como se quieran, pudiendo incluso almacenar la información de una misma región. Una vez que se ha realizado todo el proceso de cálculo de la constante de acoplamiento, cada multiplete reducido se guarda en archivos diferentes listados en la sección "constantes de acoplamiento calculadas".

La figura 4.2 muestra todos los archivos de sesión creados para un espectro de ejemplo y que conservarán el estado actual en el que se encuentra el proceso de cálculo de J para el multiplete al que hacen referencia. Al hacer click en el nombre de algún archivo de la lista, aparecerán los nombres de los archivos relacionados con las constantes de acoplamiento calculadas, de existir. Es importante mencionar que, una vez que se ha calculado una constante de acoplamiento, el archivo de sesión se referirá a la última constante de acoplamiento calculada. Por lo tanto, si se ha calculado más de una J en el multiplete de trabajo, los multipletes previos a la reducción no podrán ser modificados a menos que sean eliminados los cálculos realizados posteriormente. La opción "Mostrar resultados" estará activa una vez que se haya calculado al menos una constante de acoplamiento en cualquier lugar del espectro.

4.1.3. Paso 3: Selección de la región a procesar

Una vez creado el archivo de sesión, el siguiente paso consiste en seleccionar el multiplete de trabajo. Si el archivo de sesión es de nueva creación, se mostrará el espectro completo, pero si ya se ha seleccionado una región en una sesión previa con *nuup*, dicha región será la mostrada. En la parte inferior de la ventana principal se muestran varias secciones dependiendo de si se trata de un espectro de un experimento en una dimensión o en dos dimensiones, así como del tipo de archivo que se trate. La figura 4.3 muestra un espectro de 1D de un archivo en formato MestRe, en tanto que en la figura 4.4 se puede observar un espectro de 2D de un archivo de Varian.

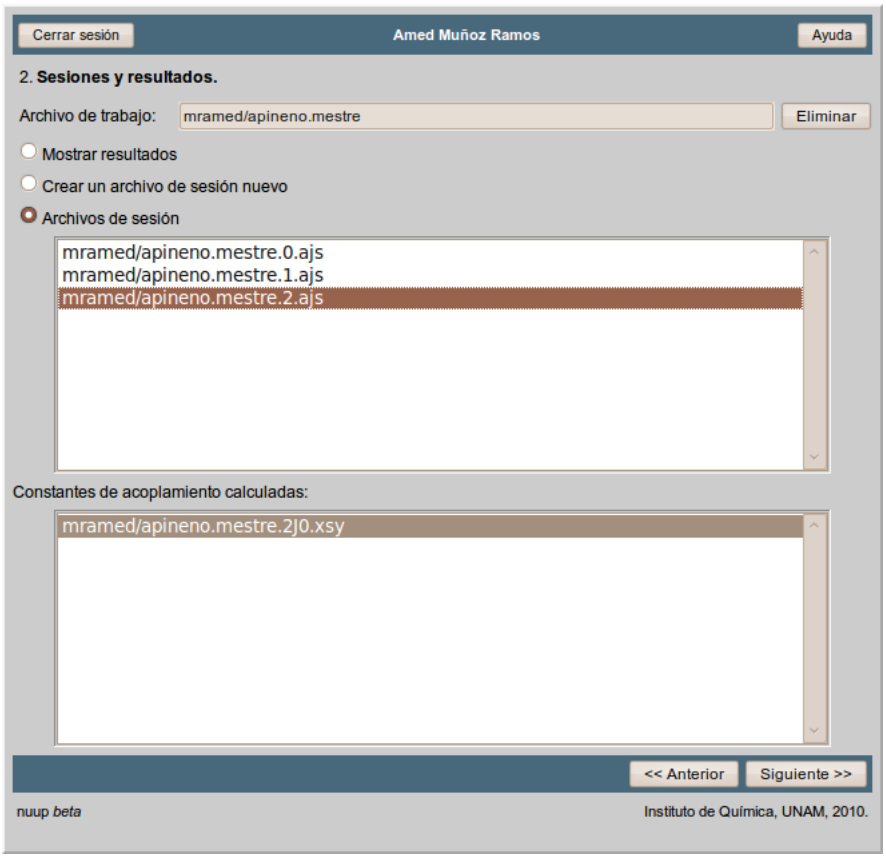


Figura 4.2: Interfaz de usuario en el paso 2, mostrando los archivos de sesión del espectro de α-pineno y el archivo relacionado a la constante de acoplamiento calculada.

Cuando los archivos contienen toda la información referente a las unidades de los datos espectrales, la sección izquierda no podrá modificarse, como es el caso de los archivos MestRe. En este caso, seleccionar el multiplete de trabajo es directo y puede hacerse de dos maneras: haciendo click con el ratón en el espectro, o modificando los valores en los cuadros de texto de la sección "Intervalo de trabajo". El cuadro de texto etiquetado como $\Delta X_{i,f}I$ es útil cuando se desea desplazar la sección definida por los cuadros de texto X_iI y X_fI, y que representan la ventana espectral de dicha sección del espectro.

Si el archivo de datos espectrales no contiene toda la información del espectro, como ocurre con los archivos de texto plano, o los archivos Varian, es necesario proporcionarla antes de poder hacer click en el botón "Graficar" que es el responsable de mostrar únicamente la región deseada. En el caso de

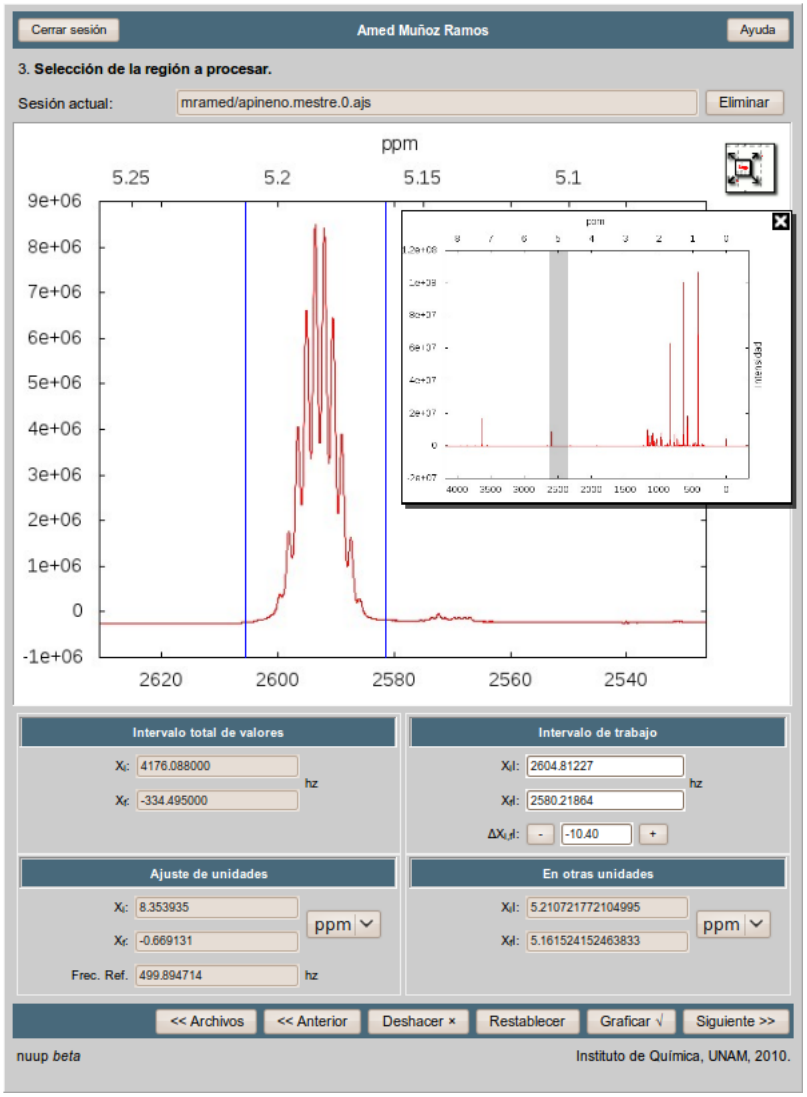


Figura 4.3: Selección de un multiplete en un experimento de RMN de una dimensión.

los archivos Varian, es posible "subir" un archivo `procpar`, que es donde el software de Varian guarda dicha información. La información del espectro extraída de un archivo procpar puede verse en la sección "Ajuste de unidades", como aparece en la figura 4.4. Sin embargo, aún estos datos pueden modificarse si se selecciona el cuadro de opción "Directamente". Toda la parte derecha de la ventana, es decir las secciones "Intervalo de trabajo" y "En otras unidades", muestran los datos de la región del espectro seleccionada, y es posible verla tanto en Hertz como en partes por millón. En esta misma figura puede verse la sección

"Propiedades del gráfico", únicamente visible en experimentos en 2D, y que permite seleccionar de una lista desplegable el tipo de gráfico a mostrar.

> **Nota importante.** La creación del gráfico mostrado es responsabilidad del programa `gnuplot` y, cuando se trata de gráficas de superficie, la rapidez con que es creado depende de la cantidad de puntos de la región a mostrar. Es debido a esto que una gráfica de curvas de nivel es preferible en regiones muy grandes o con muchos puntos, mientras que una de superficie podría llevar muchísimo tiempo en ser creada.

El botón "Deshacer" aparece activo cuando se han modificado los valores en la sección "Intervalo de trabajo" para graficar una región, o se ha hecho click con el ratón en el espectro, y restablecerá los valores iniciales que se muestran al llegar a este paso. El botón "Restablecer" se mostrará únicamente cuando ya se ha graficado una región del espectro y permitirá graficar nuevamente toda la ventana espectral.

> **Nota.** Sólo cuando se ha definido toda la información necesaria para graficar una región es que se puede hacer click en el botón "Graficar".

A un lado del cuadro de texto donde aparece el nombre del archivo de trabajo actual, se encuentra el botón "Eliminar" que, como su nombre indica, borrará el archivo de datos del espectro del directorio de trabajo.

> **Nota importante.** Al eliminar el archivo de trabajo, también serán borrados todos y cada uno de los archivos relacionados con éste, lo que incluye archivos de sesión, constantes de acoplamiento calculadas, así como todos los archivos de apoyo durante el cálculo de éstas.

En el momento en que se ha graficado el multiplete de estudio con clicks sucesivos en el botón "Graficar", hacer click en "Siguiente >>" nos llevará al paso 4 en el caso de experimentos de 2D, y directamente al paso 5 para espectros de 1D.

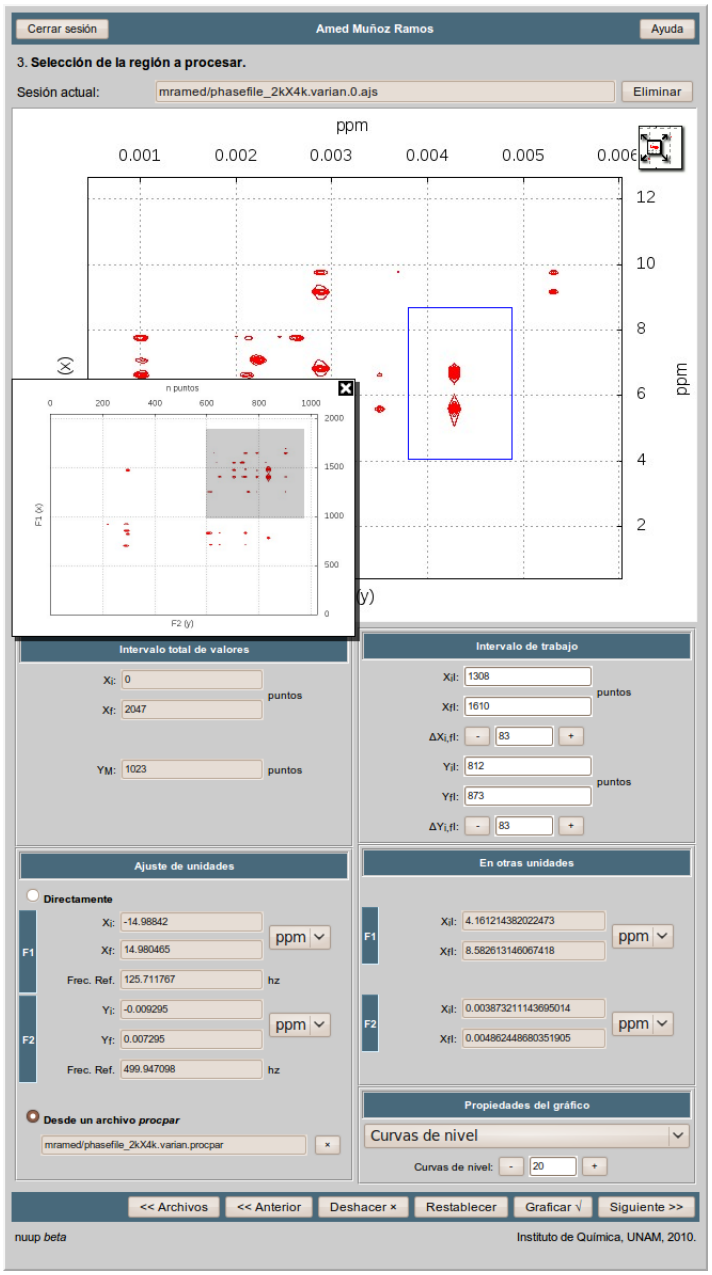


Figura 4.4: Selección de un multiplete en un experimento de RMN de dos dimensiones.

4.1.4. Paso 4: Extracción de un trazo

Para calcular la J en espectros de 2D es necesario primero extraer un solo trazo en cualquiera de las dos dimensiones o ejes de frecuencias, ya sea la del núcleo principal de estudio (primera dimensión) o

bien el resultado de la modulación de la señal como función de la dependencia debida al acoplamiento y desplazamiento químico (segunda dimensión). Para seleccionar una de las dos dimensiones se hace click en el cuadro de opción "Rotar los ejes" en la sección "Operaciones" de la ventana principal. (ver figura 4.5) Cuando se ha elegido la dimensión, existen tres formas de seleccionar un trazo específico con la opción "Extraer trazo": "Máximo central" calcula el pico de intensidad máxima más próximo al centro de la región mostrada; "Específico" no tiene ninguna restricción y permite al usuario seleccionar cualquier trazo; y "Encontrado" muestra una lista con los puntos de inflexión encontrados en toda la región. En las dos últimas opciones se mostrará en el espectro la posición aproximada del trazo en cuestión.

> **Nota.** Tanto en la selección del multiplete de estudio en los pasos anteriores como en el de la selección de trazo, el área seleccionada se distingue ya sea por un rectángulo, o bien por una línea. Sin embargo, dicha región podría no reflejar exactamente los valores mostrados en los cuadros de texto, *los cuales son los valores calculados y por tanto, los más confiables.*

Al hacer click en "Graficar" se mostrará el trazo elegido y podrá tenerse acceso a una imagen miniatura del espectro original haciendo click en el icono en la parte superior derecha de la imagen. En la sección "Intervalo de trabajo" se mostrará toda la información relacionada con el trazo mostrado y, en un recuadro con fondo azul, la dimensión seleccionada.

En este momento, el botón "Siguiente >>" llevará a *nuup* a moverse al paso 5.

4.1.5. Paso 5: Ajustes en los valores

La deconvolución tiene el potencial de hacer medibles constantes de acoplamiento que son normalmente difíciles de estimar, debido a efectos de cancelación parciales. Es por ello que es importante utilizar la opción "Corregir la línea base" para disminuír al mínimo la aparición de artefactos matemáticos debido a la relación señal-ruido. El resto de las herramientas en la sección "Operaciones" siguen el mismo objetivo: aislar al máximo el multiplete de estudio, con los extremos de intensidad lo más cerca del cero.

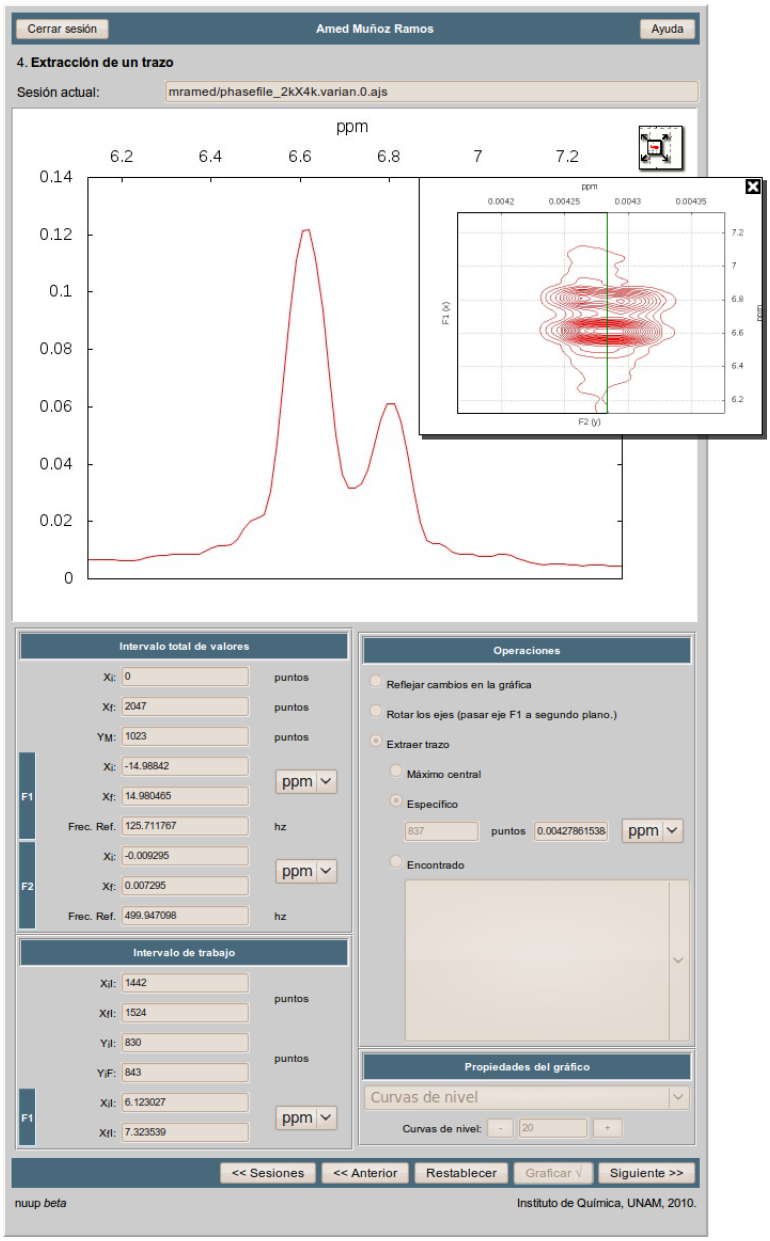


Figura 4.5: Extracción de un trazo (sólo experimentos 2D).

4.1.6. Paso 6: Selección de J^*

Antes de poder elegir un valor de la J de prueba, J^*, con la cual se llevará a cabo la deconvolución para obtener el multiplete reducido, es necesario estimar su valor. Para ello se crea la gráfica de integral (ecuación 2.4.2), cuyos parámetros se definen en este paso. El valor mínimo válido para J^* lo define

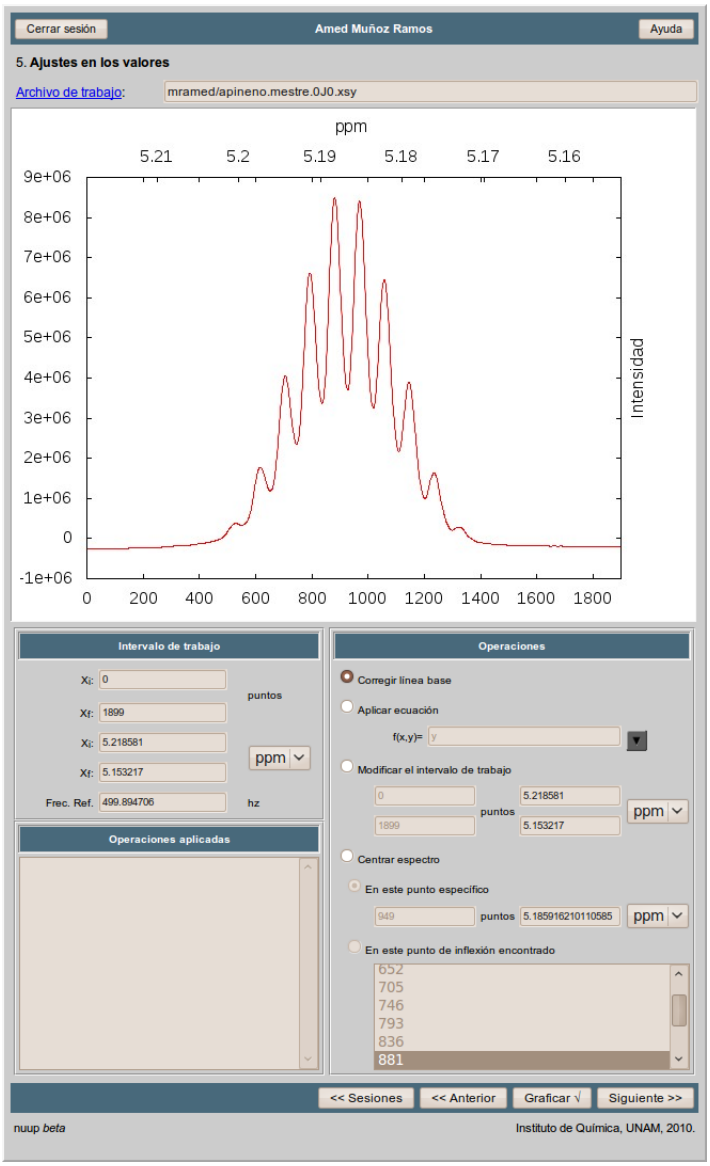


Figura 4.6: Ajustes en los valores.

la resolución del espectro, "Res. Espectro" y el valor máximo $J^*_{\text{máximo}}$. Una vez que se define el tipo de función δ y la cantidad de picos (valores distintos de cero), el botón "Graficar" creará y mostrará la gráfica de integral.

Con ayuda de la gráfica de integral es posible seleccionar un valor de J^*, el cual será un valor mínimo. Estos valores se mostrarán en la lista "Mínimos encontrados". Hacer click en "Siguiente >>" realizará la

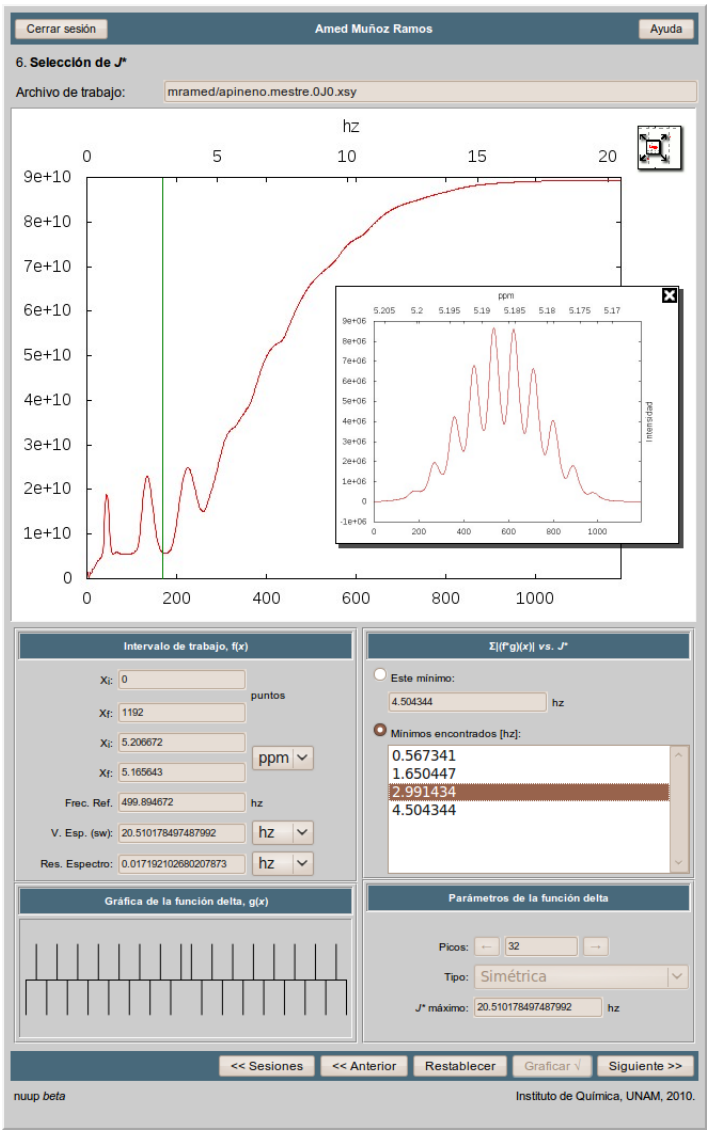


Figura 4.7: Selección de J^*.

deconvolución con el valor seleccionado, y el multiplete reducido será mostrado en el paso 7.

4.1.7. Paso 7: Espectro reducido

Se muestra sólo el espectro reducido por el proceso de deconvolución con la J^*, es decir, únicamente la región central y que corresponde a la misma ventana espectral del multiplete de trabajo. En la ventana

en miniatura es posible observar todo el dominio de la deconvolución, lo cual es útil para observar si la cancelación de señales alrededor del multiplete reducido, ha sido efectiva. De ser el caso, el valor de J^* será el valor de la J_{IS}. En caso contrario será necesario hacer click en el botón "<< Anterior" para seleccionar un valor distinto.

Todas las constantes de acoplamiento calculdas al momento aparecerán listadas en la parte inferior izquierda y, al hacer click en cada elemento de la lista se mostrará el espectro correspondiente.

Si la reducción del multiplete original fue la más adecuada, la ventana espectral del multiplete reducido será menor en el valor de la J utilizada en la deconvolución. Es posible recortar el multiplete reducido en esa cantidad utilizando la opción "Recortar el espectro". La opción "Utilizar espectro" llevará al programa *nuup* de regreso al paso 5 para poder utilizar el multiplete reducido en el cálculo de otra constante de acoplamiento.

> **Nota importante.** Cuando *nuup* ha desplegado el multiplete reducido en el paso 5, el botón "<< Anterior" NO desplazará a *nuup* hacia el paso 7, sino al paso 3. Eso significa que todas las constantes calculadas al momento se perderán para poder rededinir la región del espectro en que se encuentra el multiplete de trabajo. Para descartar el multiplete reducido y regresar al paso 7, es necesario "Eliminar" el archivo de trabajo actual que será el relacionado con la constante de acoplamiento actual y que tendrá la extensión ".xsy", en la parte superior de la ventana.

Al utilizar la opción "Cerrar archivo de sesión" se está indicando a *nuup* que se han calculado todas las constantes de acoplamiento en el multiplete de trabajo. Una vez que se ha cerrado el archivo de sesión, sólo podra ser visualizado en el paso 8.

4.1.8. Paso 8: Resultados

Se puede llegar al paso 8 por dos vías, desde el paso 2 con la opción "Mostrar resultados", o desde el paso 7 cuando se cierra el archivo de sesión. En este paso se muestran todos los archivos de sesión relacionados

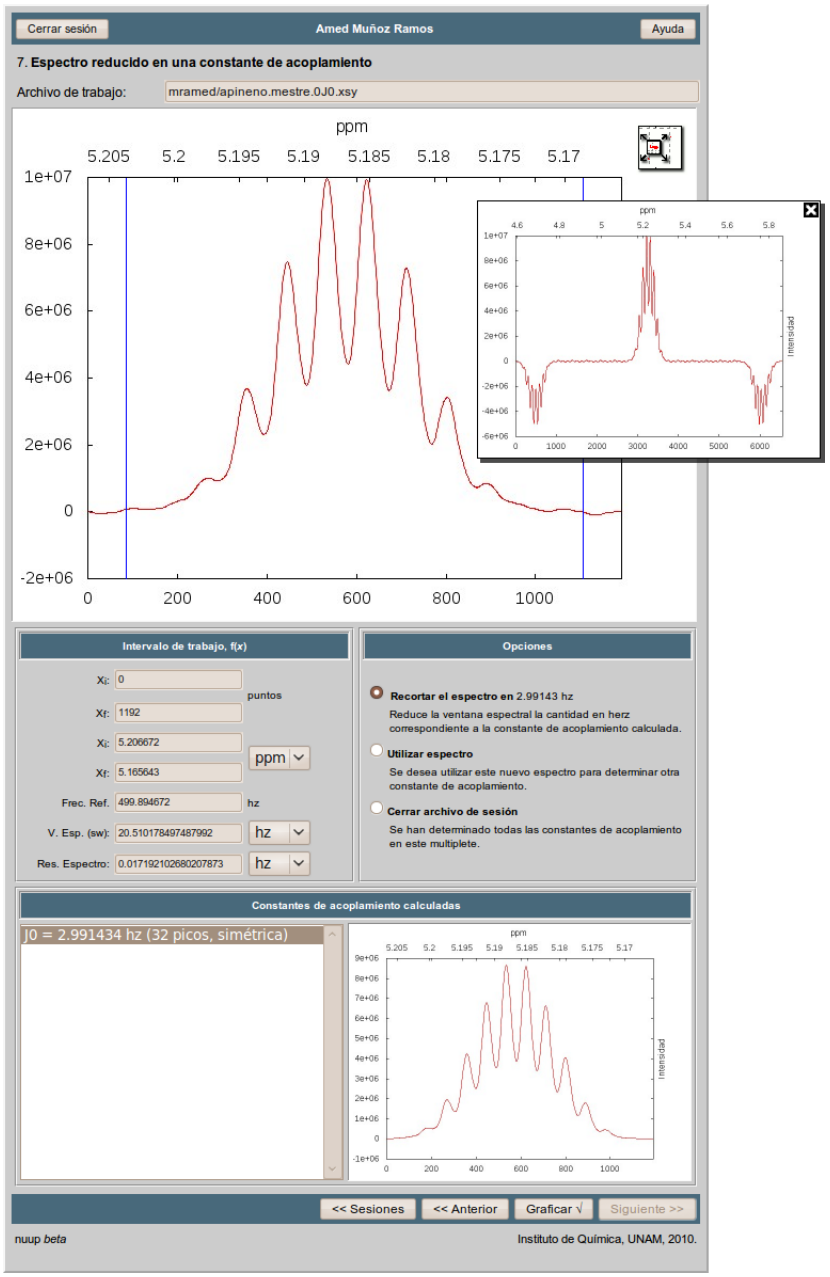


Figura 4.8: Espectro reducido.

con el espectro de RMN y las constantes de acoplamiento calculadas. Haciendo click en el botón “Detalles” se obtiene una versión extendida de la información desplegada en la ventana principal de *nuup* en este paso.

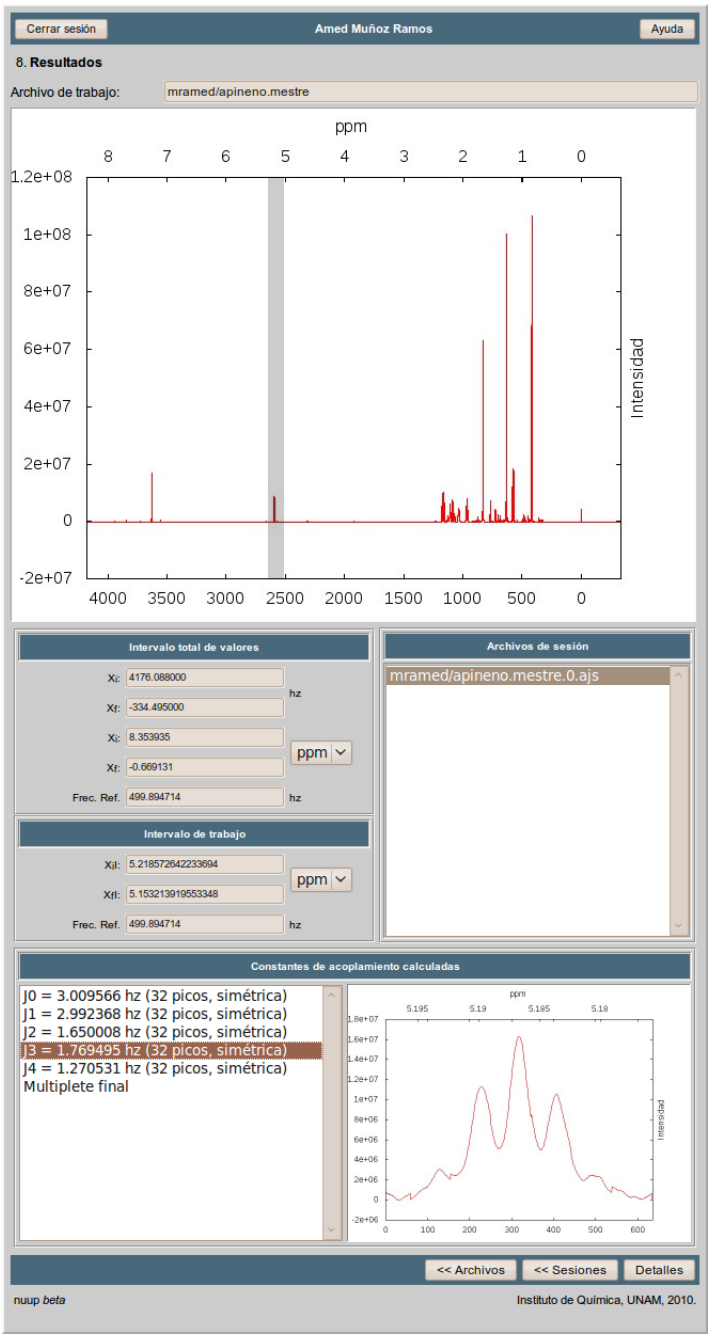


Figura 4.9: Resultados.

4.2. Cálculo de J en espectro de 2D

Se utilizó el programa *nuup* con el espectro TOCSY (ver figura 4.11) de la toxina de alacrán denominada *Tttx1* cuya estructura primaria es la siguiente:

Gly-Ser-Gly-Cys-Met-Pro-Glu-Tyr-Cys-Ala-Gly-Gln-Cys-Arg-Gly-Lys-Val-Ser-Gln-Asp-Tyr-Cys-Leu-Lys-Asn-Cys-Arg-Cys-Ile-Arg-Cys

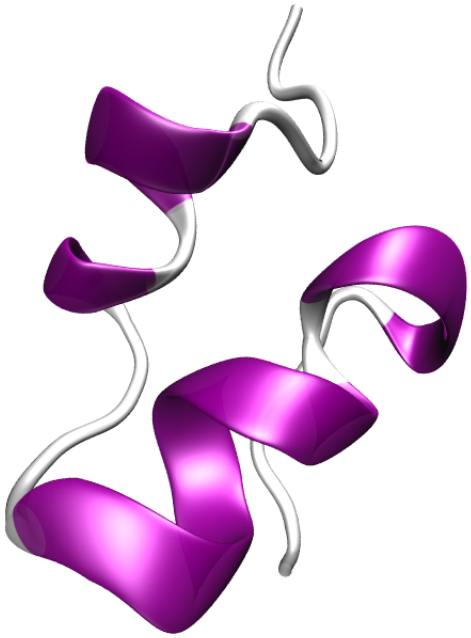


Figura 4.10: Estructura tridimensional de la *Tttx1* (Cortesía de Alma Saucedo, estructura calculada como parte de su tesis doctoral.)

4.2.1. Paso 1: El archivo de trabajo

Seleccionamos el archivo que contiene los datos del espectro TOCSY de la *Tttx1*, nombrado `tx1.txt`. La figura 4.13 muestra el paso 1 del programa *nuup* con el archivo seleccionado, procesado con el programa `nmrPipe`.

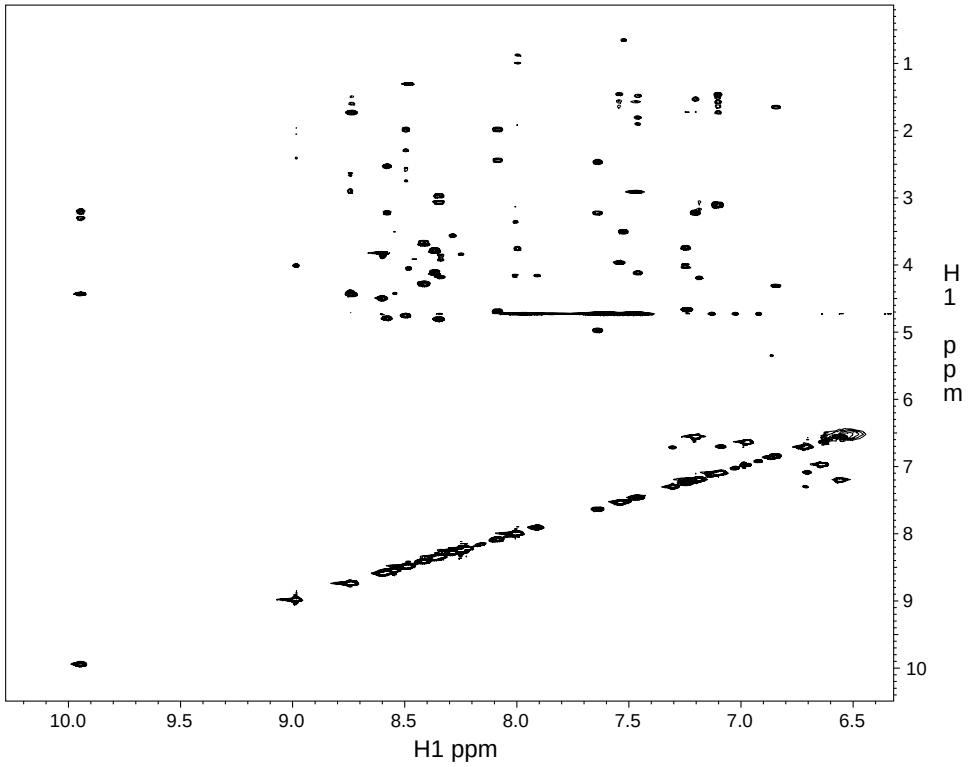


Figura 4.11: Región 7.8 a 9 ppm en H1 y 5 a 3.5 ppm en H2 del espectro TOCSY de la *Tttx1* visto con el programa CARA/NEASY. (Cortesía de Alma Saucedo, experimento de RMN parte de su tesis doctoral.)

4.2.2. Paso 2: Archivo de sesión

Para calcular una constante de acoplamiento creamos un archivo de sesión nuevo (ver figura 4.14). Este archivo contendrá toda la información relacionada con un pico del espectro.

4.2.3. Paso 3: Selección de multiplete

Debido a que el archivo generado por `nmrPipe` no contiene la información del espectro, la escala en ambas dimensiones se tuvo que introducir directamente en la sección "Ajuste de unidades" (los valores se extrajeron del programa `nmrPipe`.) La figura 4.15 muestra la región del espectro TOCSY de la *Tttx1* correspondiente al residuo HA 2S/HN 2 S, esto es, de 4.654 a 4.341 ppm en la dimensión F1 y de 8.661 a 8.554 ppm en F2.

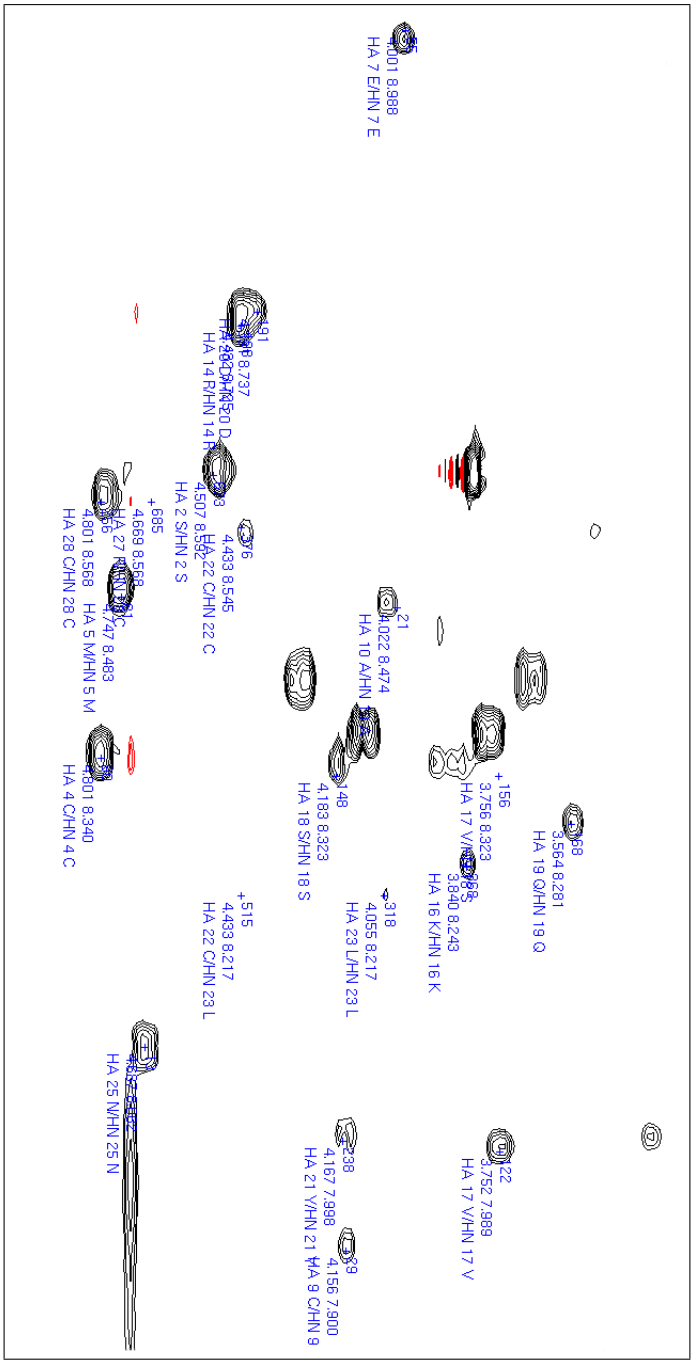


Figura 4.12: Espectro TOCSY de la *Tttx1*. (Datos del espectro y asignación cortesía de Alma Saucedo como parte de su tesis doctoral.)

Figura 4.13: Interfaz de usuario en el paso 1 con el archivo procesado con el programa nmrPipe de la Tttx1.

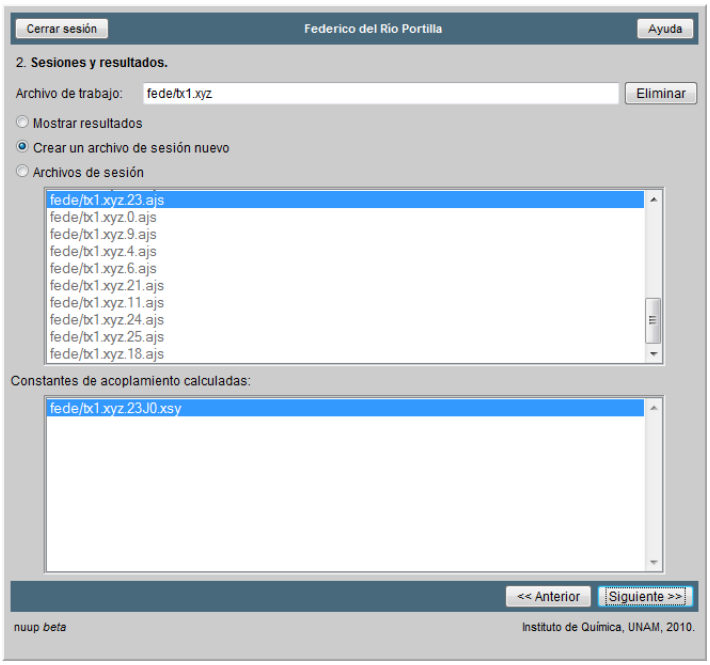


Figura 4.14: Interfaz de usuario en el paso 2 para la creación de un archivo de sesión nuevo para el espectro TOCSY de la *Tttx1*.

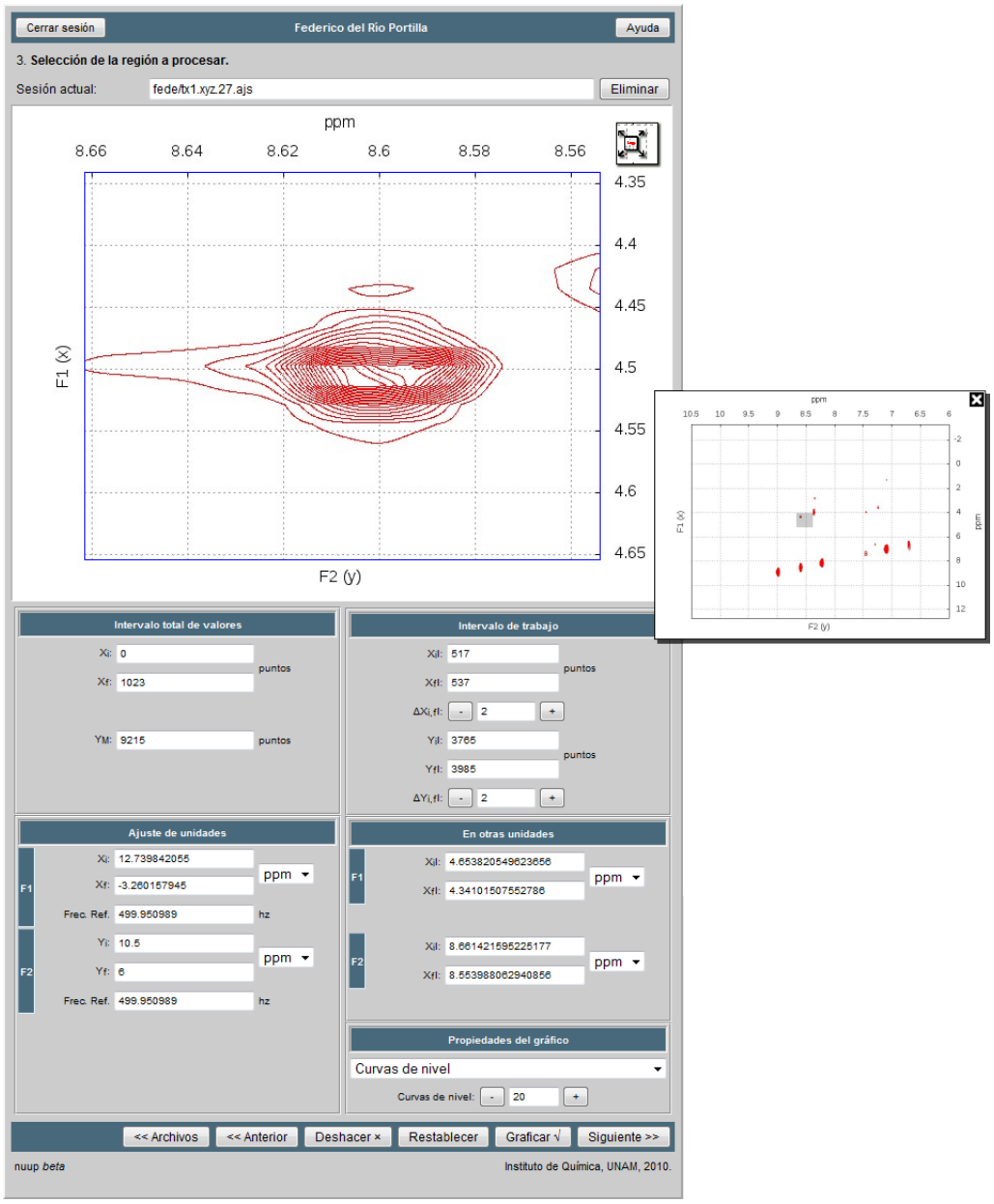


Figura 4.15: Paso 3 de *nuup* que muestra el pico a 4.507 ppm en F1 y 8.592 ppm en F2 correspondiente al residuo HA 2S/HN 2 S de la *Tttx1*.

4.2.4. Paso 4: Extracción del trazo en F1

Recortamos el espectro a 4.513 ppm de F1 (en el intervalo de 8.661 a 8.55 ppm en F2) utilizando la opción

"Extraer trazo" y "Máximo central" (ver figura 4.16)

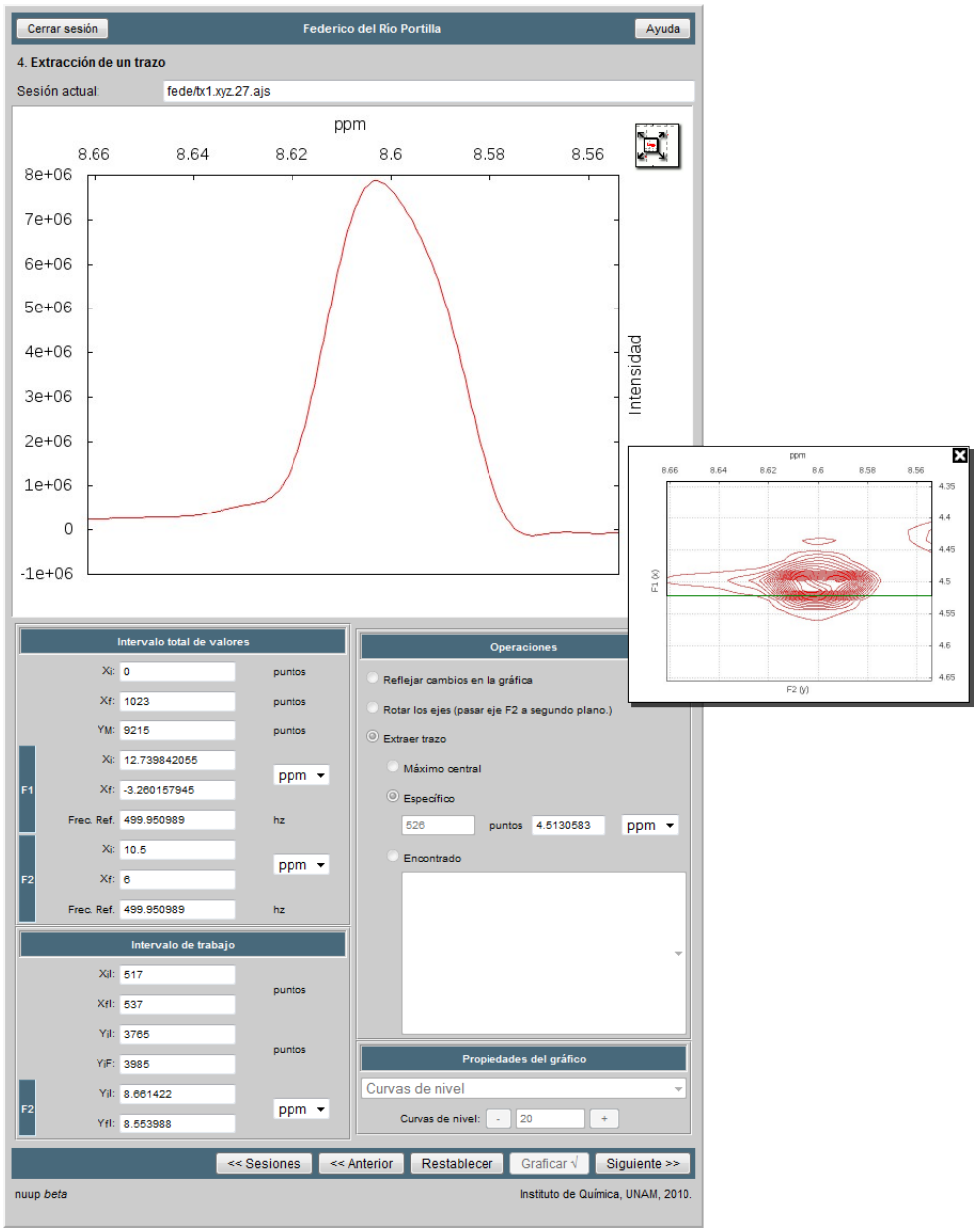


Figura 4.16: Paso 4 de *nuup* muestra el trazo a 4.513 ppm en F1.

4.2.5. Paso 5: Corrección de línea base

La corrección de la línea base es un paso importante para el método modificado de duplicación de J. La figura 4.17

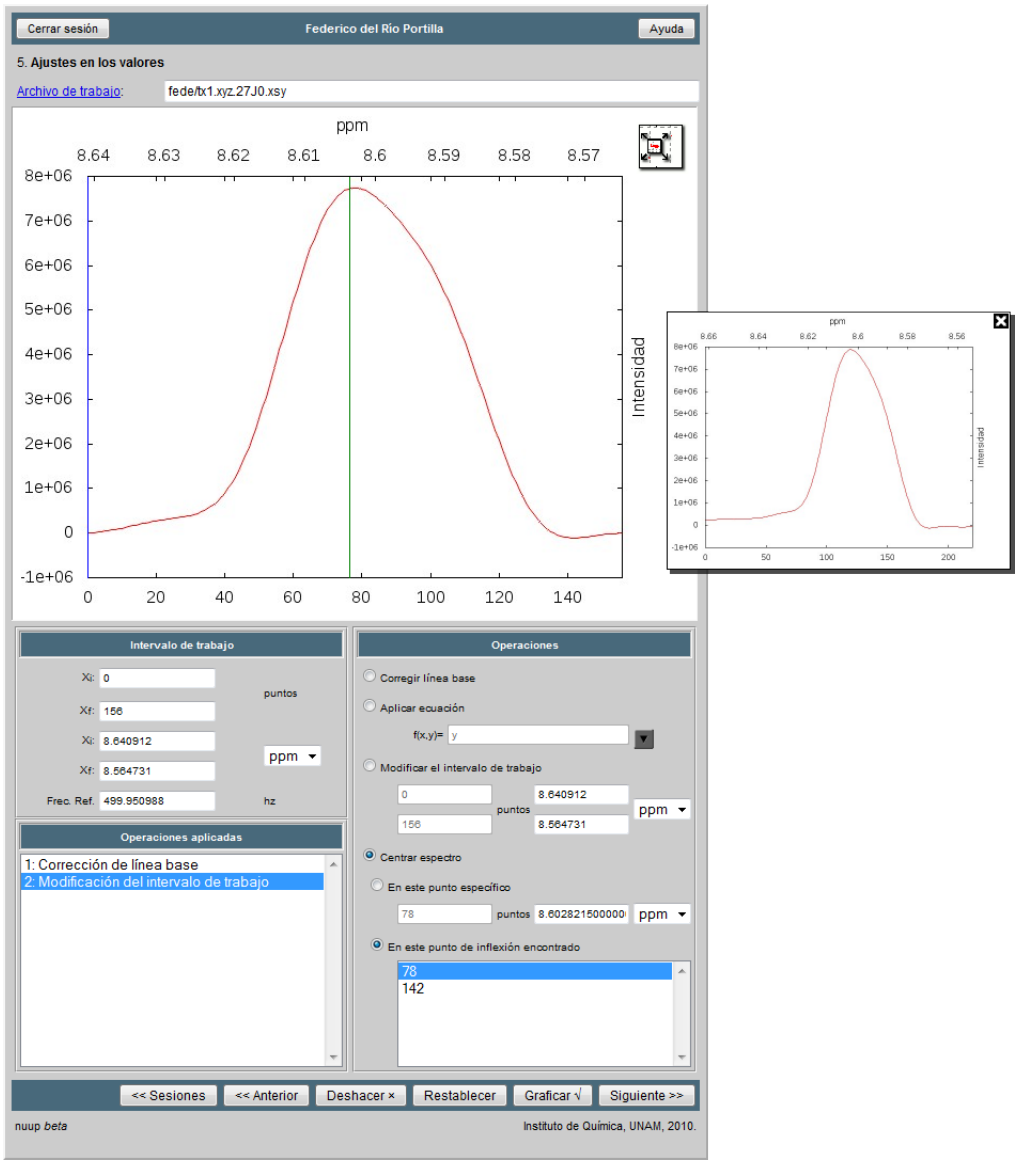


Figura 4.17: Paso 5 de *nuup* con la corrección de la línea base, después de haber recortado el intervalo.

4.2.6. Paso 6: Selección de J^*

En el paso 6 del programa *nuup* seleccionamos la cantidad de picos y el tipo de función δ a utilizar para generar la gráfica de integral, de donde seleccionamos el valor mínimo más profundo como el valor de la J de prueba (ver figura 4.18.)

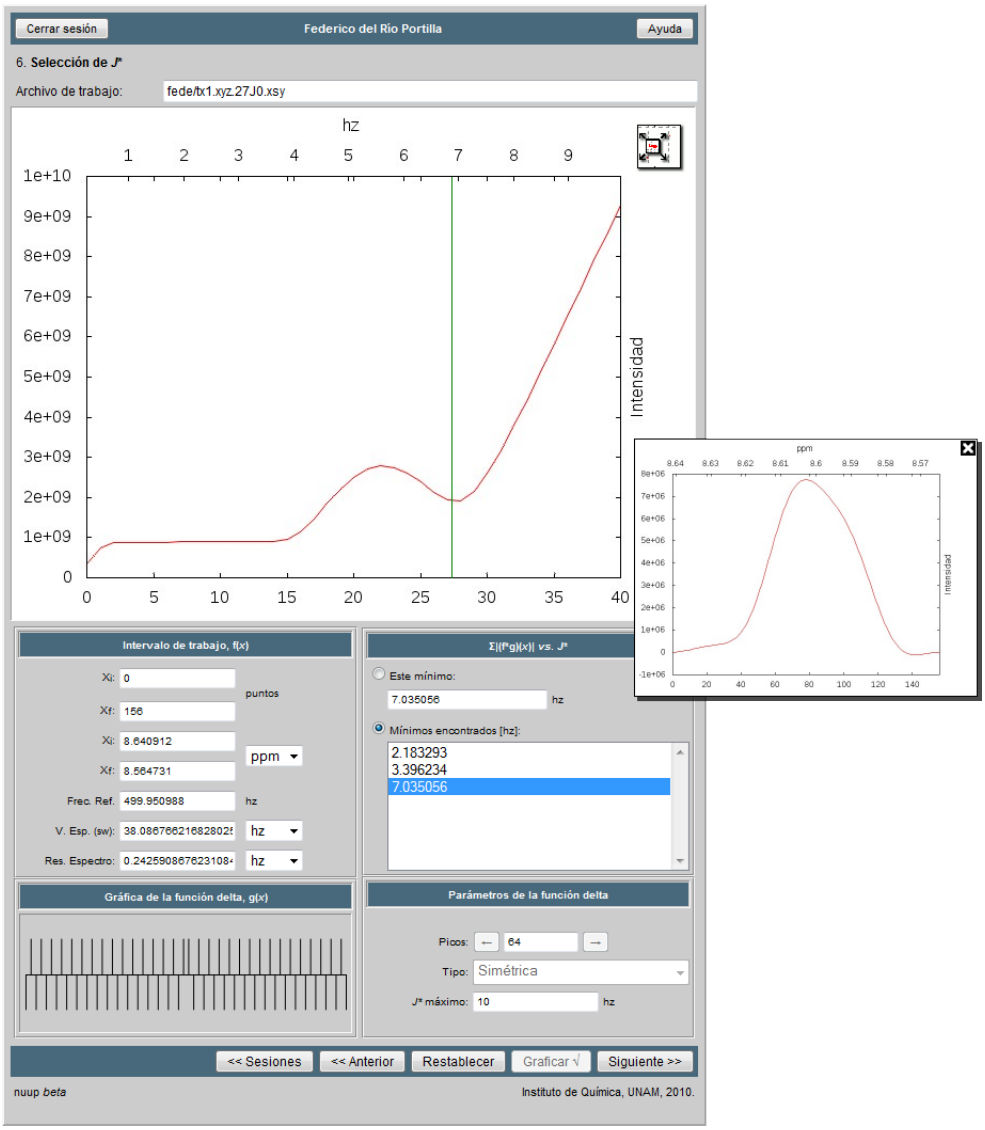


Figura 4.18: Paso 6 de *nuup* que muestra la función de integral utilizando 64 picos en una familia de funciones δ simétricas. El valor mínimo más profundo está a 7.03 hz y es el valor de la J^* a utilizar.

4.2.7. Paso 7: Reducción del multiplete

La deconvolución se realizó con $J^* = 7.035$ hz, siendo el pico mostrado en la figura 4.19 la reducción del multiplete.

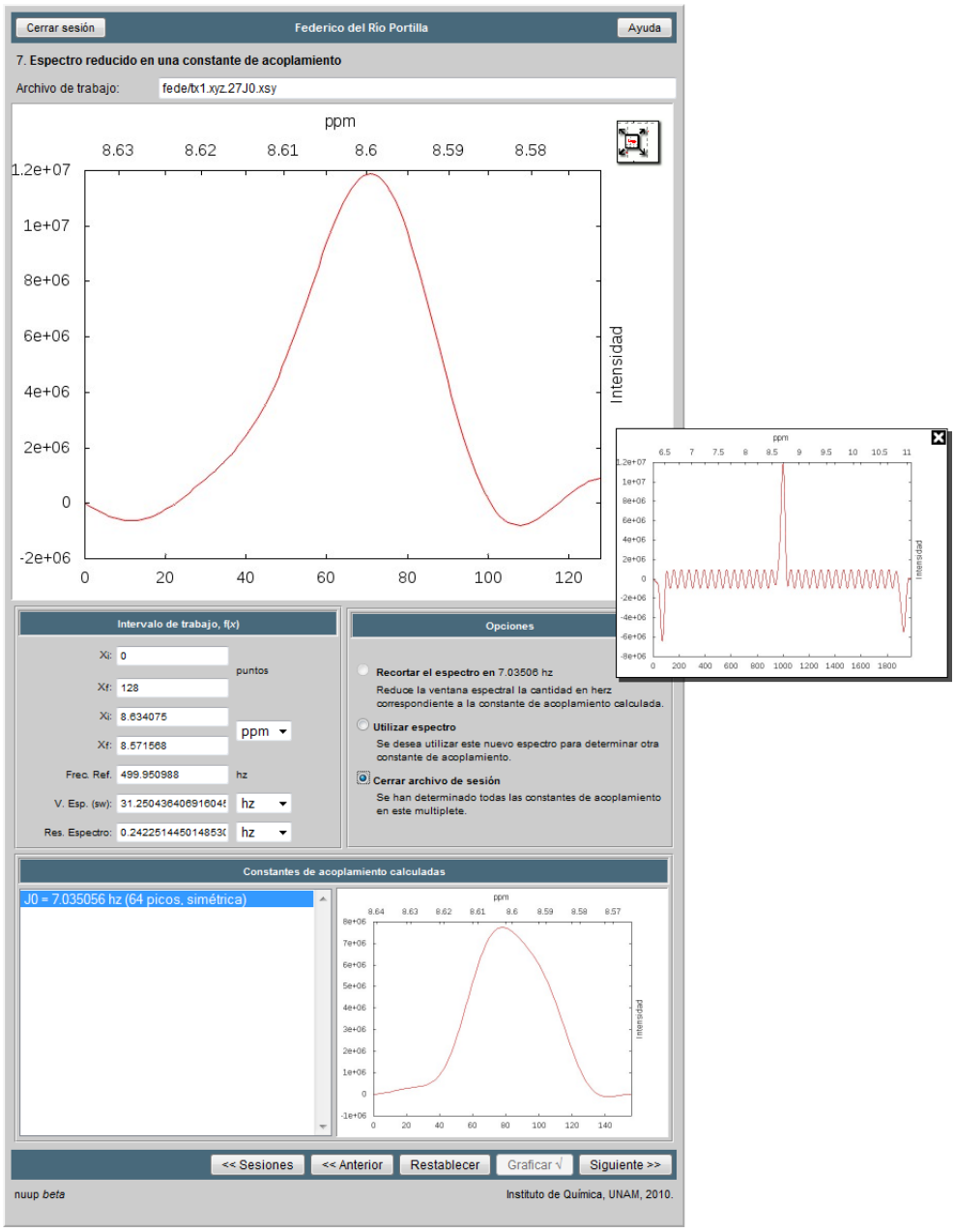


Figura 4.19: Paso 7 de *nuup* con el multiplete reducido.

4.2.8. Paso 8: Resultados

El paso 8 de *nuup* muestra un resumen de todas las constantes de acoplamiento calculadas utilizando el espectro TOCSY de la *Tttx1* (y cuyos valores se resumen en la tabla 4.1). Para los valores de los residuos

21 y 23 la gráfica de integral no mostró valores mínimos tan profundos y, debido a ello, la reducción del multiplete presentó armónicos demasiado pronunciados.

4.2.9. Perspectivas

Haciendo uso de los valores de J calculados podría llevarse a cabo un cálculo dinámico de la estructura de la Tttx1.

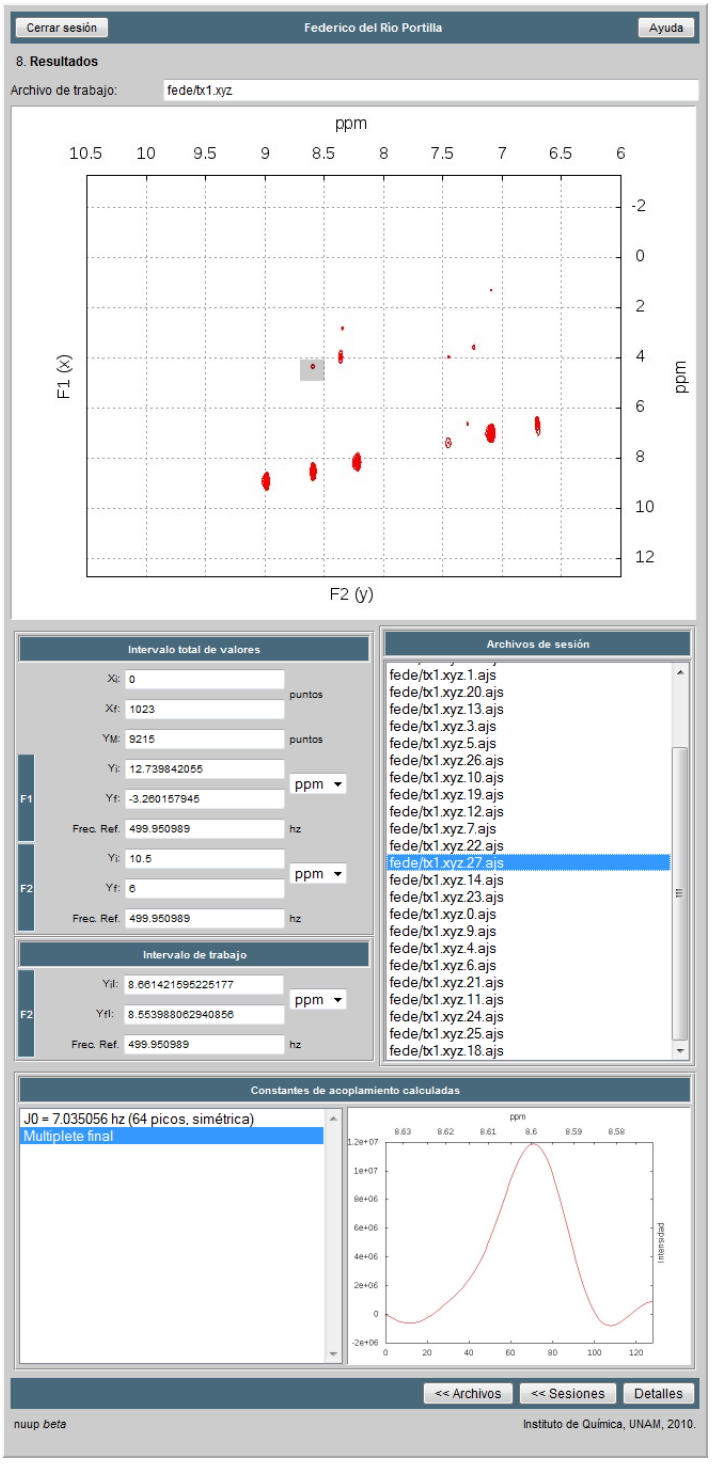


Figura 4.20: Paso 8 de *nuup*; se muestra la lista de todos los valores de J calculados para los residuos de la *Tttx1*.

		F1	F2	
pico	**archivo de sesión** [.ajs]	[ppm]	[ppm]	**J** [hz]
HA 2 S/HN 2 S	mramed/tx1.xyz.1	4.507	8.592	7.03
HA 4 C/HN 4 C	fede/tx1.xyz.7	4.801	8.340	7.51
HA 5 M/HN 5 M	mramed/tx1.xyz.0	4.747	8.483	6.06
HA 7 E/HN 7 E	fede/tx1.xyz.6	4.001	8.988	3.86
HA 8 Y/HN 8 Y	fede/tx1.xyz.10	4.198	7.174	3.88
HA 9 C/HN 9 C	fede/tx1.xyz.9	4.156	7.900	3.64
HA 10 A/HN 10 A	fede/tx1.xyz.5	4.055	8.479	3.88
HA 12 Q/HN 12 Q	fede/tx1.xyz.11	4.119	7.457	7.28
HA 13 C/HN 13 C	fede/tx1.xyz.16	4.951	7.630	8.98
HA 14 R/HN 14 R	fede/tx1.xyz.17	4.432	8.725	7.52
HA 16 K/HN 16 K	fede/tx1.xyz.15	3.840	8.243	3.63
HA 17 V/HN 17 V	fede/tx1.xyz.18	3.752	7.989	3.63
HA 18 S/HN 18 S	fede/tx1.xyz.1	4.183	8.323	4.61
HA 19 Q/HN 19 Q	fede/tx1.xyz.14	3.564	8.281	4.85
HA 20 D/HN 20 D	fede/tx1.xyz.13	4.391	8.737	7.54
HA 21 Y/HN 21 Y	fede/tx1.xyz.20	4.167	7.998	7.53
HA 21 Y/QD 21 Y	fede/tx1.xyz.21	4.173	7.203	4.13
HA 22 C/HN 22 C	fede/tx1.xyz.3	4.431	8.545	4.60
HA 23 L/HN 23 L	fede/tx1.xyz.22	4.055	8.217	4.60
HA 24 K/HB 24 K	fede/tx1.xyz.23	4.320	6.849	8.01
HA 25 N/HN 25 N	fede/tx1.xyz.24	4.687	8.082	8.50
HA 26 C/HN 26 C	fede/tx1.xyz.4	4.442	9.940	6.79
HA 27 R/HN 27 R	fede/tx1.xyz.25	4.670	7.238	6.79
HA 28 C/HN 28 C	fede/tx1.xyz.12	4.801	8.561	7.28
HA 29 I/HN 29 I	fede/tx1.xyz.2	3.525	7.528	4.85
HA 30 R/HN 30 R	fede/tx1.xyz.26	3.951	7.525	6.31

Tabla 4.1: Algunas constantes calculadas con el espectro TOCSY de la *Tttx1*. Para los resíduos 21 y 21 la gráfica de integral no mostró valores mínimos tan profundos y el multiplete reducido presentó armónicos demasiado pronunciados.

Apéndice A

Mecánica cuántica

Esta sección surgió con la intención de colocar herramientas matemáticas que sirven para la demostración de algún tema concreto en la RMN, pero que no son específicas de esta última y que podrían omitirse. Las he escrito porque muestran la belleza de las matemáticas en sí.

A.1. Relación de I_x e I_y con α y β

Los operadores $\hat{I}_x$, $\hat{I}_y$ y $\hat{I}_z$, como todos los operadores de momento angular, obedecen las relaciones de conmutación

$$[\hat{I}_x, \hat{I}_y] = i\hbar\hat{I}_z, \quad [\hat{I}_y, \hat{I}_z] = i\hbar\hat{I}_x, \quad [\hat{I}_z, \hat{I}_x] = i\hbar\hat{I}_y$$

Si definimos los operadores (no Hermitianos)

$$\hat{I}_+ = \hat{I}_x + i\hat{I}_y \tag{A.1}$$

$$\text{y} \quad \hat{I}_- = \hat{I}_x - i\hat{I}_y \tag{A.2}$$

y sabiendo que

$$\hat{I}_z\hat{I}_+ = \hat{I}_+\hat{I}_z + \hbar\hat{I}_+ \tag{A.3}$$

$$\hat{I}_z\hat{I}_- = \hat{I}_-\hat{I}_z - \hbar\hat{I}_- \tag{A.4}$$

y

$$\begin{aligned}\hat{I}_+\hat{I}_- &= \hat{I}^2 - \hat{I}_z^2 + \hbar\hat{I}_z \\ \hat{I}_-\hat{I}_+ &= \hat{I}^2 - \hat{I}_z^2 - \hbar\hat{I}_z\end{aligned} \tag{A.5}$$

donde

$$\hat{I}^2 = \hat{I}_x^2 + \hat{I}_y^2 + \hat{I}_z^2 \tag{A.6}$$

y el hecho de que $\hat{I}_z\beta = -\frac{\hbar}{2}\beta$, se puede demostrar que

$$\hat{I}_z\hat{I}_+\beta = \hat{I}_+\left(-\frac{\hbar}{2}\beta + \hbar\beta\right) = \frac{\hbar}{2}\hat{I}_+\beta \tag{A.7}$$

Debido a que $\hat{I}_z\alpha = \frac{\hbar}{2}\alpha$, la ecuación anterior muestra que

$$\hat{I}_+\beta \propto \alpha = c\alpha \tag{A.8}$$

donde c es una constante de proporcionalidad. Del mismo modo,

$$\hat{I}_z\hat{I}_-\alpha = \hat{I}_-\left(\frac{\hbar}{2}\alpha - \hbar\alpha\right) = -\frac{\hbar}{2}\hat{I}_-\alpha \tag{A.9}$$

muestra que

$$\hat{I}_-\alpha \propto \beta = c\beta \tag{A.10}$$

Para calcular el valor de la constante de proporcionalidad, resolvemos la integral

$$\frac{\int \alpha^* \alpha d\tau}{\int (\hat{I}_+\beta)^*(\hat{I}_+\beta)d\tau} = \frac{1}{c^2}$$

Utilicemos $\hat{I}_+ = \hat{I}_x + i\hat{I}_y$ (ecuación A.1) y el hecho de que $\hat{I}_x$ e $\hat{I}_y$ son Hermitianos, con el denominador

$$\int (\hat{I}_x\hat{I}_+\beta)^*\beta d\tau + i\int (\hat{I}_y\hat{I}_+\beta)^*\beta d\tau = c^2$$

Ahora tomemos el complejo conjugado de ambos lados para tener que

$$\begin{aligned}
\int \beta^*\hat{I}_x\hat{I}_+\beta d\tau - i\int \beta^*\hat{I}_y\hat{I}_+\beta d\tau &= c^2 \\
\int \beta^*(\hat{I}_x\hat{I}_+ - i\hat{I}_y\hat{I}_+)\beta d\tau &= c^2 \\
\int \beta^*(\hat{I}_x - i\hat{I}_y)\hat{I}_+\beta d\tau &= c^2 \\
\int \beta^*\hat{I}_-\hat{I}_+\beta d\tau &= c^2
\end{aligned}$$

Sabiendo que $\hat{I}^2\beta = \frac{1}{2}\left(\frac{1}{2}+1\right)\hbar^2\beta$ y utilizando la ecuación A.5,

$$\begin{aligned}
c^2 &= \int \beta^*\hat{I}_-\hat{I}_+\beta d\tau \\
&= \int \beta^*(\hat{I}^2 - \hat{I}_z^2 - \hbar\hat{I}_z)\beta d\tau \\
&= \int \beta^*\left(\frac{3}{4}\hbar^2 - \frac{1}{2}\hbar\frac{1}{2}\hbar + \hbar\frac{\hbar}{2}\right)\beta d\tau \\
&= \left(\frac{3}{4} - \frac{1}{4} + \frac{2}{4}\right)\hbar^2\int \beta^*\beta d\tau \\
&= \hbar^2 \\
\therefore c &= \hbar
\end{aligned}$$

Las ecuaciones A.8 y A.10 se convierten en

$$\hat{I}_+\beta = \hbar\alpha \tag{A.11}$$

$$\hat{I}_-\alpha = \hbar\beta \tag{A.12}$$

Debido a que el operador $\hat{I}_+$ "asciende" la función de espín α a β y que el operador $\hat{I}_-$ "desciende" la función de espín β a α se les denomina, respectivamente, operadores *de ascenso* y *de descenso*. Una consecuencia de las propiedades de estos operadores es que

$$\hat{I}_+\alpha = 0 \quad \text{y} \quad \hat{I}_-\beta = 0 \tag{A.13}$$

Utilizando las ecuaciones A.11, A.12 y A.13 con la ecuación A.1:

$$\begin{aligned} [\hat{I}_+ - i\hat{I}_y]\beta &= [\hat{I}_x + \cancel{i\hat{I}_y} - \cancel{i\hat{I}_y}]\beta \\ \hat{I}_+\beta - i\hat{I}_y\beta &= \hat{I}_x\beta \\ \hbar\alpha - i\hat{I}_y\beta &= \hat{I}_x\beta \end{aligned} \tag{A.14}$$

$$\begin{aligned} [\hat{I}_+ - i\hat{I}_y]\alpha &= \hat{I}_x\alpha \\ \cancelto{0}{\hat{I}_+\alpha} - i\hat{I}_y\alpha &= \hat{I}_x\alpha \\ -i\hat{I}_y\alpha &= \hat{I}_x\alpha \end{aligned} \tag{A.15}$$

Con la ecuación A.2:

$$\begin{aligned} [\hat{I}_- + i\hat{I}_y]\beta &= [\hat{I}_x - \cancel{i\hat{I}_y} + \cancel{i\hat{I}_y}]\beta \\ \cancelto{0}{\hat{I}_-\beta} + i\hat{I}_y\beta &= \hat{I}_x\beta \\ i\hat{I}_y\beta &= \hat{I}_x\beta \end{aligned} \tag{A.16}$$

$$\begin{aligned} [\hat{I}_- + i\hat{I}_y]\alpha &= \hat{I}_x\alpha \\ \hat{I}_-\alpha + i\hat{I}_y\alpha &= \hat{I}_x\alpha \\ \hbar\beta + i\hat{I}_y\alpha &= \hat{I}_x\alpha \end{aligned} \tag{A.17}$$

Si sustituímos A.17 en A.15 tenemos

$$\begin{aligned} -i\hat{I}_y\alpha &= \hbar\beta + i\hat{I}_y\alpha \\ -i\hat{I}_y\alpha - i\hat{I}_y\alpha &= \hbar\beta + \cancel{i\hat{I}_y\alpha} - \cancel{i\hat{I}_y\alpha} \\ \left(\frac{i}{\cancel{2}}\right)(-\cancel{2}i\hat{I}_y\alpha) &= \left(\frac{i}{2}\right)\hbar\beta \\ -(-1)\hat{I}_y\alpha &= \frac{i\hbar}{2}\beta \\ \hat{I}_y\alpha &= \frac{i\hbar}{2}\beta \end{aligned} \tag{A.18}$$

Con A.18 en A.17

$$\begin{aligned} \hat{I}_x\alpha &= \hbar\beta + i\left(\frac{i\hbar}{2}\beta\right) \\ &= \left(\frac{2\hbar}{2} - \frac{\hbar}{2}\right)\beta \\ \therefore \hat{I}_x\alpha &= \frac{\hbar}{2}\beta \end{aligned} \tag{A.19}$$

Ahora, sustituyendo A.16 en A.14,

$$\begin{aligned} \hbar\alpha - i\hat{I}_y\beta &= i\hat{I}_y\beta \\ \hbar\alpha - \cancel{i\hat{I}_y\beta} + \cancel{i\hat{I}_y\beta} &= 2i\hat{I}_y\beta \\ \left(-\frac{i}{2}\right)\hbar\alpha &= \cancel{-i^2}\hat{I}_y\beta \\ \therefore \hat{I}_y\beta &= -\frac{i\hbar}{2}\alpha \end{aligned} \tag{A.20}$$

Finalmente, utilizando A.20 en A.16

$$\begin{aligned} \hat{I}_x\beta &= i\left(-\frac{i\hbar}{2}\alpha\right) \\ \hat{I}_x\beta &= \frac{\hbar}{2}\alpha \end{aligned} \tag{A.21}$$

Apéndice B

Código de *nuup*

B.1. nuup.h

```
\begin{verbatim}
#ifndef ACOPLAMIENTO_J
#define ACOPLAMIENTO_J


#include "interfaz.h"


#define NUUP_ARCH_HTML              "index.html"
#define NUUP_DIR_HTML_VIRTUAL       "aplicaciones/nuup/"
#define NUUP_DIR_CGI_VIRTUAL        "aplicaciones/nuup/"


// Se crea en el directorio del usuario:
#define NUUP_DIR_DATA_VIRTUAL       "data/"



#endif
\end{verbatim}
```

B.2. nuup.c

```
#include "nuup.h"


/**/
int main (int iArgC, char *asArgC[])
{
  /* VARs */
  nbool blnCorrecto = VERDADERO;
  ou_datos ou;


  /*
   * La salida (NULL -> stdout) y el directorio de las páginas Web
   * ¡DEBE SER LA PRIMERA FUNCIÓN, SI NÓ, TODO SE MANDA A "NULL" = segmentation fault!
   *
   *  <Salida>,  <dir HTML>, <dir CGI>, <dir Virtual>, <arch CGI>
   *
   */
  hu_inicializa (NULL,    // salida (stdout)
 NULL, NUUP_DIR_HTML_VIRTUAL,      // dir html, virtual
 NULL, NUUP_DIR_CGI_VIRTUAL, asArgC[0] ); // dir cgi, virtual, nombre cgi







  // INICIALIZAMOS LOS OBJETOS...


  // ...propiedades generales de los objetos.
  // Desde allí se llaman a estas funciones (se trae los parámetros):
  //  - hp_inicializa (NULL, hp_nombreArchivoSimple, directorio_datos_virtual);
  //  - hp_obten ();
  //
  // EL SEGUNDO PARÁMETRO es para definir dónde se descargan archivos
  // cuando se hace desde un formulario
  //
  blnCorrecto = ou_inicializa (&ou, NUUP_DIR_DATA_VIRTUAL);
```

```
  // CREAMOS LOS OBJETOS...
  if (blnCorrecto)
    blnCorrecto = crea_ventanaAJ (&ou);

  // armamos en memoria todos los objetos
  if (blnCorrecto)
    ou_hazlo (&ou);



  // MOSTRAMOS LA PÁGINA



  //// Contenido cambiante
  // ... el tipo de datos a mandar (texto)
  hu_preparaCont (NULL);

  if (!blnCorrecto)
    {
      printf ("<html>\n<head>\n"\
      "<meta http-equiv=\"Content-Type\"content=\"text/html; charset=UTF-8\">\n");
      printf ("<meta http-equiv=\"Pragma\" content=\"no-cache\"></head>\n");
      printf ("<body>\n"\
      "¡Error desconocido!<br><br>Favor de notificar al "\
      "<a href='mailto:sssnegro@servidor.unam.mx'>administrador</a> de este sitio.\n"\
      "</body>\n</html>");
    }
    else
      hu_sPresenta (NULL);

  // LIBERAMOS MEMORIA
  // ...los errores
  ou_libera (&ou, FALSO);

  // ...los parámetros
  hp_libera (FALSO);
}
```

B.3. interfaz.h

```
#ifndef NUUP_INTERFAZ

#define NUUP_INTERFAZ


#include <negroBibs/UtilsHTML/os/osInicioSesion.h>

#include <negroBibs/UtilsPGSQL/pqOperaciones.h>

#include <negroBibs/UtilsEncrip8.h>

#include "nuup.h"

#include "os/osAcoplamiento.h"



#define NUUP_NOMBRE                "nuup"

#define NUUP_VERSION               "<i>beta</i>"

#define NUUP_MENSAJE               "Si no cuenta con un nombre de usuario u olvidó"\

  " su contraseña, envíe un correo al <a class='os_mensaje'"\

  " href='mailto:sssnegro@servidor.unam.mx'"\

  " title='Administrador de %s'>administrador</a>"

#define NUUP_MENSAJE_EN            "If you don't have a user name or forgot"\

  " your password send an e-mail to the <a class='os_mensaje'"\

  " href='mailto:sssnegro@servidor.unam.mx'"\

  " title='%s administrator'>administrator</a>."

#define NUUP_BOTON                 "Entrar"

#define NUUP_BOTON_EN              "Login"


// Base de datos

#define NUUP_BD_NOMBRE             "nuup"

#define NUUP_BD_USUARIO            "www-data"



extern char  nuup_achrNombre[128];

extern char  nuup_achrTitulo[128];

extern nbool nuup_blnDesarrollo;


/**/

nbool

crea_ventanaAJ (ou_datos *ou);
```

```
#endif
```

B.4. osAcoplamiento.h

```
#ifndef OS_ACOPLAMIENTOJ_H

#define OS_ACOPLAMIENTOJ_H



#include <negroBibs/UtilsHTML/os/osUtils.h>

#include <negroBibs/UtilsQuim/quimDatosRMN.h>

#include <math.h>

#include "onuupAux.h"



// ¿Debería ser OC_DIR_HTMLs ? (Donde se encuentran los os)

#define ONUUP_DIR_HTML            "os/html/"

#define ONUUP_ARCH_HTML_P         "onuupPaso%d-%s.html"

#define ONUUP_ARCH_HTML_A         "onuupAyuda%d-%s.html"


#define ONUUP_OPT                 "<OPTION>%s</OPTION>"

#define ONUUP_OPT_SEL             "<OPTION SELECTED>%s</OPTION>"


#define ONUUP_OPT_VAL             "<OPTION VALUE=\"%s\">%s</OPTION>"

#define ONUUP_OPT_VAL_SEL         "<OPTION VALUE=\"%s\" SELECTED>%s</OPTION>"


#define ONUUP_ARCH_FUNCIONES      "<!--"\

  "#include virtual=\"%s%sos/html/onuupFunciones%d.html\"-->"

#define ONUUP_NOMBRE              "ONUUP"


// Generales


#define ONUUP_USUARIO_GEN         "nuup"

#define ONUUP_PASO                "ONUUP_PASO"

#define ONUUP_OPASO               "ONUUP_OPASO"


// Textos
```

```
#define ONUUP_USUARIO             "ONUUP_USUARIO"
#define ONUUP_TITULO              "ONUUP_TITULO"
#define ONUUP_TEXTO               "ONUUP_TEXTO"
#define ONUUP_BOTONES             "ONUUP_BOTONES"
#define ONUUP_BOTON_ELIMINAR      "ONUUP_BOTON_ELIMINAR"
#define ONUUP_BOTON_E_PROCPAR     "ONUUP_BOTON_E_PROCPAR"
#define ONUUP_BOTON_AYUDA         "ONUUP_BOTON_AYUDA"
#define ONUUP_BOTON_CERRAR        "ONUUP_BOTON_CERRAR"

// Paso 1: pedir archivos

#define ONUUP_ORIGEN              "ONUUP_ORIGEN"
#define ONUUP_LOCAL               "ONUUP_LOCAL"
#define ONUUP_URL                 "ONUUP_URL"
#define ONUUP_ARCHIVO             "ONUUP_ARCHIVO"
#define ONUUP_REEMP_ARCH          "ONUUP_REEMP_ARCH"

// Paso 2: acción

#define ONUUP_ACCION              "ONUUP_ACCION"
#define ONUUP_SESION              "ONUUP_SESION"
#define ONUUP_TIPO_ARCH           "ONUUP_TIPO_ARCH"
#define ONUUP_DELTAX              "ONUUP_DELTAX"
#define ONUUP_DELTAX_O            "ONUUP_DELTAX_O"
#define ONUUP_INACTIVA            "ONUUP_INACTIVA"


// Paso 2, 3, 4, 5
#define ONUUP_ELIMINAR            "ONUUP_ELIMINAR"
#define ONUUP_E_PROCPAR           "ONUUP_E_PROCPAR"
#define ONUUP_JS                  "ONUUP_JS"
#define ONUUP_J                   "ONUUP_J"

// Paso 3: graficar datos
#define ONUUP_X_IZQ               "ONUUP_X_IZQ"
#define ONUUP_X_DER               "ONUUP_X_DER"
#define ONUUP_X_ESC               "ONUUP_X_ESC"
#define ONUUP_DELTAY              "ONUUP_DELTAY"
```

```
#define ONUUP_DELTAY_O            "ONUUP_DELTAY_O"

// ...partes por millón

#define ONUUP_X_IZQ_PPM1          "ONUUP_X_IZQ_PPM1"

#define ONUUP_X_DER_PPM1          "ONUUP_X_DER_PPM1"

#define ONUUP_X_FR1               "ONUUP_X_FR1"

#define ONUUP_X_IZQ_PPM2          "ONUUP_X_IZQ_PPM2"

#define ONUUP_X_DER_PPM2          "ONUUP_X_DER_PPM2"

#define ONUUP_X_FR2               "ONUUP_X_FR2"


#define ONUUP_X_FE1               "ONUUP_X_FE1"

#define ONUUP_X_FE2               "ONUUP_X_FE2"


// ...subintervalo

#define ONUUP_X_IZQ_I             "ONUUP_X_IZQ_I"

#define ONUUP_X_DER_I             "ONUUP_X_DER_I"

#define ONUUP_X_ESC_I             "ONUUP_X_ESC_I"

// ...leído

#define ONUUP_X_IZQ_IL            "ONUUP_X_IZQ_IL"

#define ONUUP_X_DER_IL            "ONUUP_X_DER_IL"

#define ONUUP_Y_INF_IL            "ONUUP_Y_INF_IL"

#define ONUUP_Y_SUP_IL            "ONUUP_Y_SUP_IL"

#define ONUUP_NP_I                "ONUUP_NP_I"

#define ONUUP_NT_I                "ONUUP_NT_I"

#define ONUUP_Y                   "ONUUP_Y"

#define ONUUP_Y_INF_I             "ONUUP_Y_INF_I"

#define ONUUP_Y_SUP_I             "ONUUP_Y_SUP_I"

#define ONUUP_INICIO_DATOS        "ONUUP_INICIO_DATOS"

#define ONUUP_JPG                 "ONUUP_JPG"

#define ONUUP_JPG_TODO            "ONUUP_JPG_TODO"

#define ONUUP_JPG_INT             "ONUUP_JPG_INT"

#define ONUUP_I_SESION            "ONUUP_I_SESION"

#define ONUUP_I_PASO_SESION       "ONUUP_I_PASO_SESION"

#define ONUUP_TIPOS_GRAF          "ONUUP_TIPOS_GRAF"

#define ONUUP_PROP_GRAF           "ONUUP_PROP_GRAF"

#define ONUUP_3D                  "ONUUP_3D"

#define ONUUP_CN                  "ONUUP_CN"


// Paso 5
```

```
#define ONUUP_SESION_J            "ONUUP_SESION_J"
#define ONUUP_ENLACE_J            "ONUUP_ENLACE_J"
#define ONUUP_OPS                 "ONUUP_OPS"
#define ONUUP_ECS                 "ONUUP_ECS"
#define ONUUP_EC                  "ONUUP_EC"

// Paso 6
#define ONUUP_PICOS               "ONUUP_PICOS"
#define ONUUP_JMAX                "ONUUP_JMAX"
#define ONUUP_TIPO_FDELTA         "ONUUP_TIPO_FDELTA"
#define ONUUP_JS                  "ONUUP_JS"



#define ONUUP_FORMATO_SESION      "Xmin=%s\nXmax=%s\nXesc=%s\nYmax=%s\n"\
  "Xizq=%s\nXder=%s\nYinf=%s\nYsup=%s\nXizqL=%s\nXderL=%s\nYinfL=%s\nYsupL=%s\n"\
  "NP=%s\nNT=%s\nDYsup=%lf\nDYinf=%lf\nCN=%s\ntipoArch=%d\n" \
  "tipoGraf=%s\npropGraf=%s\niPaso=%d\nInicioDatos=%ld\nDeltaX=%s\nDeltaY=%s\n"\
  "3D=%s\nDim2D=%s\nXizqPPM1=%s\nXderPPM1=%s\nFE1=%s\nXizqPPM2=%s\nXderPPM2=%s\nFE2=%s\n"\
  "TE=%s\nnPicos=%s\ntipoFDelta=%s\nJmax=%s\n"



// Paso 3
#define ONUUP_OPCION_A            "ONUUP_OPCION_A"
#define ONUUP_OPCION_A_DIRECTO    "DIRECTO"
#define ONUUP_OPCION_A_PROCPAR    "PROCPAR"



// Paso 4: ajustes 3D
#define ONUUP_DIM_2D              "ONUUP_DIM_2D"
#define ONUUP_OPCION_A_ACTUALIZAR "ACTUALIZAR"
#define ONUUP_OPCION_A_ROTAR      "ROTAR"
#define ONUUP_OPCION_A_RECORTAR   "RECORTAR"
#define ONUUP_OPCION_I            "ONUUP_OPCION_I"
#define ONUUP_OPCION_I_CENTRAL    "CENTRAL"
#define ONUUP_OPCION_I_ESPECIFICO "ESPECIFICO"
#define ONUUP_OPCION_I_ENCONTRADO "ENCONTRADO"
#define ONUUP_GRAF_SEL            "ONUUP_GRAF_SEL"
```

```
#define ONUUP_TE                  "ONUUP_TE"
#define ONUUP_TS                  "ONUUP_TS"
#define ONUUP_X_IZQ_PPM1i         "ONUUP_X_IZQ_PPM1i"
#define ONUUP_X_DER_PPM1i         "ONUUP_X_DER_PPM1i"
#define ONUUP_X_FR1i              "ONUUP_X_FR1i"
#define ONUUP_X_IZQ_PPM2i         "ONUUP_X_IZQ_PPM2i"
#define ONUUP_X_DER_PPM2i         "ONUUP_X_DER_PPM2i"
#define ONUUP_X_FR2i              "ONUUP_X_FR2i"
#define ONUUP_PROCPAR             "ONUUP_PROCPAR"

// Paso 4 o 5
#define ONUUP_AJUSTE              "ONUUP_AJUSTE"



// Paso 5: ajustes 2D
#define ONUUP_OPCION_A_BASE       "BASE"
#define ONUUP_OPCION_A_MODIFICAR  "MODIFICAR"
#define ONUUP_OPCION_A_ECUACION   "ECUACION"
#define ONUUP_OPCION_A_CENTRAR    "CENTRAR"



#define ONUUP_ENC_XY_TIT          "#X\t\tY\n------------\n"
#define ONUUP_ENC_Z_TIT           "#Z\n------------\n"
#define ONUUP_M                   "ONUUP_M"
#define ONUUP_M_O                 "ONUUP_M_O"



#define ONUUP_EC_BASE             "y-(x*((yf-yi)/xf)+yi)"

/*
#define ONUUP_DELTAY_CERO         "(x-x1)((y2-y1)/(x2-x1))+y1"
#define ONUUP_DELTAY_INCLINAR     "(1/2)*(((y2-y1)/(x2-x1))-m)*(2*x-x1-x2)" */



// Paso 6
#define ONUUP_OPCION_A_ESPECIFICO "ESPECIFICO"
#define ONUUP_OPCION_A_ENCONTRADO "ENCONTRADO"
```

```
// Errores

#define ONUUP_ERROR               "ONUUP_ERROR"
#define ONUUP_ADVERTENCIA         "ONUUP_ADVERTENCIA"
#define ONUUP_AVISO               "ONUUP_AVISO"



typedef enum {
  ONUUP_ORIGEN_LOCAL=0,
  ONUUP_ORIGEN_URL,
  ONUUP_ORIGEN_ARCHIVO
} onuup_origen;
typedef enum {
  ONUUP_ACCION_RESULTADOS=0,
  ONUUP_ACCION_NUEVA,
  ONUUP_ACCION_SESION,
  ONUUP_ACCION_ELIMINAR
} onuup_accion;
typedef enum {
  ONUUP_AJUSTE_NINGUNO=0,
  ONUUP_AJUSTE_GRAFICAR,
  ONUUP_AJUSTE_DESHACER,
  ONUUP_AJUSTE_RESTABLECER
} onuup_ajuste;




typedef struct {
  char             achrURL[2*UA_TAM_MAX];
  // intervalo máximo:
  HP_2             hp2XIzq;
  HP_2             hp2XDer;
  HP_2             hp2XEsc;
  HP_2             hp2Y;
  // intervalo graficado:
  HP_2             hp2XIzqI;
  HP_2             hp2XDerI;
```

```
  HP_2             hp2YInfI;
  HP_2             hp2YSupI;
  // intervalo leído:
  HP_2             hp2XIzqIL;
  HP_2             hp2XDerIL;
  HP_2             hp2YInfIL;
  HP_2             hp2YSupIL;
  HP_2             hp2NPI;
  HP_2             hp2NTI;
  // gráfico:
  double           dDYsup, dDYinf;
  HP_2             hp2CN;
  DRMN_TIPO_ARCH   tipoArch;
  HP_2             hp2GrafSel;    // el archivo a leer
  HP_2             hp2PropGraf;
  // datos de la sesión:
  int              iPasoSesion;
  long int         lInicioDatos;
  HP_2             hp2DeltaX;
  HP_2             hp2DeltaY;
  // datos del espectro:
  HP_2             hp23D;
  HP_2             hp2Dim2D;
  HP_2             hp2XIzqPPM1;
  HP_2             hp2XDerPPM1;
  HP_2             hp2XFE1;
  HP_2             hp2XIzqPPM2;
  HP_2             hp2XDerPPM2;
  HP_2             hp2XFE2;
  HP_2             hp2TE;
  // gráfica de suma
  HP_2             hp2Jmax;
  HP_2             hp2Picos;
  HP_2             hp2TipoFDelta;
} onuup_sesion;
```

```
typedef struct {
  // El archivo de sesión
  onuup_sesion       ajs;

  // Textos
  char             achrNombreUsuario[34+1]; // psql
  HP_2             hp2Usuario;
  HP_2             hp2Titulo;
  HP_2M            hp2MTexto;
  HP_2M            hp2MBotones;

  // Datos generales
  int              iPaso;
  HP_2             hp2URL;
  char             achrURLjpg[2*UA_TAM_MAX];
  char             achrURLjpgT[2*UA_TAM_MAX];
  char             achrURLplot[2*UA_TAM_MAX];
  char             achrURLdat[2*UA_TAM_MAX];
  char             achrURLtra[2*UA_TAM_MAX];
  char             achrURLtmp[2*UA_TAM_MAX];
  char             achrURLpis[2*UA_TAM_MAX];
  char             achrURLops[2*UA_TAM_MAX];
  char             achrURLprocpar[2*UA_TAM_MAX];
  HP_2C            hp2CSesion;

  // Paso 1: pedir archivo
  onuup_origen       origen;
  char             achrDirTrabajo[2*UA_TAM_MAX];
  char             achrDirUsuario[2*UA_TAM_MAX];
  HP_2C            hp2CLocal;

  // Paso 2: acción a realizar
  onuup_accion       accion;
  HP_2             hp2BotonEliminar;
  HP_2             hp2BotonEProcpar;
  HP_2             hp2BotonAyuda;
  HP_2             hp2BotonCerrar;
  HP_2             hp2Inactiva;
```

```
  HP_2C            hp2CJS;
  // Paso 3: graficar datos
  int              iSesion;
  int              iJ;
  HP_2             hp2XEscI;
  HP_2             hp2JPG;
  HP_2             hp2JPGT;
  HP_2             hp2TiposGraf;
  HP_2             hp2Procpar;

  // Paso 4: ajustes 2D
  onuup_ajuste     ajuste;
  HP_2C            hp2CTS;
  HP_2             hp2XIzqPPM1i;
  HP_2             hp2XDerPPM1i;
  HP_2             hp2XFR1i;
  HP_2             hp2XIzqPPM2i;
  HP_2             hp2XDerPPM2i;
  HP_2             hp2XFR2i;
  HP_2             hp2OpcionI;
  HP_2             hp2OpcionA;
  // Paso 5: ajustes 1D
  HP_2C            hp2OPS;
  // ...
  HP_2C            hp2CSesionJ;
  HP_2C            hp2CEnlaceJ;
  HP_2C            hp2EC;
  HP_2C            hp2ECS;
  // Paso 6: suma de integral
  // Errores
  HP_2             hp2Error;
  HP_2             hp2Advertencia;
  HP_2             hp2Aviso;
} onuup_datos;



/**/
void
```

```
onuup_hazlo     (char *asSalida,
       ou_o *,
       char *idioma);









/*************************************
 * Las funciones que se usan.
 */



/**/
void
onuup_inicializa     (onuup_datos  *,
    char       *nombre_a_mostrar,
    char       *titulo,
    int         iPaso,
    char       *asIdioma);
/**/
nbool
onuup_setDirArchivos (onuup_datos  *aonuup,
    char       *nombre_usuario,
    char       *directorio);
/**/
nbool
onuup_procesa        (onuup_datos   *,
    char        *idioma);



/**/
nbool
onuup_agrega         (ou_datos    *,
    onuup_datos   *,
    char        *ID,
    nbool        pide_memoria);
```

```
/*****************************************
 * Funciones auxiliares
 */

/**/
nbool
onuup_leeSesion (onuup_sesion *,
       HP_2       *mensaje,
       char       *idioma);

/**/
nbool
onuup_escribeSesion (onuup_sesion *,
   HP_2       *mensaje,
   char       *idioma);




// Las Js
typedef struct
{
  int            iPicos;
  drmn_tipoDelta tipoFDelta;
  int            iAjs;
  double         dValor;
} onuup_j;

typedef struct
{
  int    n;
  onuup_j *js;
} onuup_js;

typedef struct
{
  int    iAjs;
```

```
  char  *as;
} onuup_arch;



//////////////////////
//
// OPERACIONES
//

typedef enum {
  ONUUP_OP_MODIFICAR=1,
  ONUUP_OP_BASE=2,
  ONUUP_OP_ECUACION=4,
  ONUUP_OP_CENTRAR=8,
  ONUUP_OP_DESHACER,
  ONUUP_OP_NINGUNA,
} ONUUP_OP;



typedef struct
{
  ONUUP_OP  id;
  char    desc[256];
} onuup_op;

#endif
```

B.5. index.html

```
<html>


  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <meta http-equiv="Pragma" content="no-cache" >


    <link rel="stylesheet" type="text/css" href="/css/os.css">
    <link rel="stylesheet" type="text/css" href="/css/osActual.css">
```

```
    <title> Objetos </title>

    %OU_FUNCIONES_G%
    %OU_FUNCIONES%

  </head>

  <body>

    <table class="os_cont">
      <tr>
<td>
  %ONUUP1%
</td>
      </tr>
    </table>

  </body>

</html>
```

B.6. onuupFunciones1.html

```
<script Language="JavaScript" src="/aplicaciones/nuup-des/os/js/onuup.js"></script>
<script language="JavaScript" src="/aplicaciones/nuup-des/os/js/onuup1.js"></script>
```

B.7. quimDatos.h

```
////////////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////////////
////
////  MANIPULACIÓN DE ARCHIVOS EN FORMATO X-Y (MestRe),
////  XEASY y Varian.
////
```

```
////////////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////////////



#ifndef QUIM_DATOS_RMN
#define QUIM_DATOS_RMN

#include "../Tipos.h"
#include "../UtilsArchs.h"
#include "../UtilsNums.h"
#include "../UtilsLista.h"



#ifdef  X86_64
#define DRMN_LONG int
#else
#define DRMN_LONG long
#endif



////////////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////////////
////
////  ENCABEZADOS DE LOS ARCHIVOS
////
////////////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////////////



#define DRMN_ESC     int
#define DRMN_ESC_PPM 1
#define DRMN_ESC_HZ  2
#define DRMN_ESC_P   4
```

```
//////////////////////
//
// XYZ
//


typedef struct
{
  long           lNP;              // Número de puntos
  long           lNT;              // Número de puntos
  char           strT[1024];
  long int       lInicioDatosL;    // Línea en que inician los datos
} drmn_XYZ;


//
drmn_XYZ *
drmn_inicializaEncXYZ (drmn_XYZ *,
       const char *titulo);

//////////////////////
//
// MestRe
//

typedef enum
  {
    DRMN_MESTRE_VIEJO,
    DRMN_MESTRE_NUEVO
  } drmn_verMestre;
typedef struct
{
  char           strT[1024];       // Título o descripción
  drmn_verMestre v;                // Versión del MestRe
  float          fFE;              // Frecuencia del espectrómetro
  float          fFB;              // Frecuencia más baja
```

```
  float          fVE;              // Ventana espectral
  long           lNP;              // Número de puntos
  float          fFD;              // Filtro digital
  long int       lInicioDatosL;    // Línea en que inician los datos
  DRMN_ESC       escX;             // Escala en X
} drmn_MestRe;

//
drmn_MestRe *
drmn_inicializaEncMestRe (drmn_MestRe *);



//////////////////////////
//
// XEASY
//

#define DRMN_ENC_XEASY "Number of data points      : %ld\n"\
  "Chemical shift range (ppm) : %lf\t%lf\nChemical shift range (Hz)  : %lf\t%lf\n"
typedef struct
{
  long       lNP;              // Número de puntos
  double     dLPP;             // Límite izquierdo (ppm)
  double     dRPP;             // Límite derecho (ppm)
  double     dLFF;             // Límite izquierdo (hz)
  double     dRFF;             // Límite derecho (hz)
  long int   lInicioDatosL;    // Línea en que inician los datos
} drmn_XEASY;

//
drmn_XEASY *
drmn_inicializaEncXEASY (drmn_XEASY *);



//////////////////////////
//
// Varian
```

```
//


typedef enum
  {
    DRMN_VARIAN_DATA,
    DRMN_VARIAN_PHASE
  } drmn_archVarian;
typedef struct
{
  // Encabezado de archivo
  drmn_archVarian  av;       // Tipo de archivo de varian
  long             lNB;      // BLOQUES
  long             lNT;      // TRAZOS por BLOQUE
  long             lNP;      // PUNTOS por TRAZO
  long             lNE;      // Encabezados por bloque
  long             lBP;      // BYTES por PUNTO
  // Encabezado de bloque
  short    sEstado;
  short    sIndiceIni;
  short    sModo;
} drmn_Varian;



//
drmn_Varian *
drmn_inicializaEncVarian (drmn_Varian *);



////////////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////////////
////
////  ARCHIVOS
////
////////////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////////////



#define DRMN_TIPO_ARCH    int
```

```
#define DRMN_XYZ          1
#define DRMN_MESTRE       2
#define DRMN_XEASY        4
#define DRMN_VARIAN       8
#define DRMN_DESCONOCIDO  DRMN_XYZ+DRMN_MESTRE+DRMN_XEASY+DRMN_VARIAN



typedef struct
{
  char             strRuta[UA_TAM_MAX];
  FILE            *aFOrigen;
  DRMN_TIPO_ARCH  tipo;
  long int         lInicioDatosB;   // posición en bytes
  void            *encabezado;
} drmn_arch;



//
drmn_arch *
drmn_inicializaArch   (drmn_arch      *,
       const char     *ruta_archivo,
       DRMN_TIPO_ARCH  tipo);

////////////////////////////////////
//
// Abre el archivo y lee su encabezado
// Regresa FALSO si no se trata de un
// tipo de archivo conocido.
//
nbool
drmn_abreArch         (drmn_arch *);

//
void
drmn_cierraArch       (drmn_arch *);

//
void
```

```
drmn_rebobinaArch     (drmn_arch *);



////////////////////////////////////////////////////////
//
// Devuelve el encabezado del archivo
//
////

//
drmn_XYZ *
drmn_encXYZ    (drmn_arch *);

//
drmn_MestRe *
drmn_encMestRe (drmn_arch *);

//
drmn_XEASY *
drmn_encXEASY  (drmn_arch *);

//
drmn_Varian *
drmn_encVarian (drmn_arch *);




extern char *drmn_encs[];   // el tipo de datos (en texto)



//////////////////////////////////////////
// Devuelve el tipo de datos como cadena
//
char *
drmn_sTipo (drmn_arch *);

//
DRMN_TIPO_ARCH
```

```
drmn_tipo (drmn_arch *);


////////////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////////////
////
////  Los datos
////
////////////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////////////



// Si no se define la cantidad de puntos al pedir memoria,
// la memoria se amplía cada DRMN_PUNTOS_BLOQUE_MEM puntos.
// (con realloc).
#define DRMN_PUNTOS_BLOQUE_MEM   1000000
#define DRMN_DIMS                3


typedef struct {
  int             intDim;
  long int        lNP;                   // PUNTOS x TRAZO
  long int        lNT;                   // TRAZOS (totales)
  long int       *llMinimos[DRMN_DIMS];  // Un mínimo por cada trazo
  long int       *llMaximos[DRMN_DIMS];  // Un máximo por cada trazo
  double          ddLI[DRMN_DIMS];       // Límites inferiores en cada coordenada con datos
  double          ddLS[DRMN_DIMS];       // Límites superiores en cada coordenada con datos
  double          ddLI_PPM[DRMN_DIMS];   // Límites inferiores en cada coordenada (ppm)
  double          ddLS_PPM[DRMN_DIMS];   // Límites superiores en cada coordenada (ppm)
  double          ddLI_HZ[DRMN_DIMS];    // Límites inferiores en cada coordenada (Herz)
  double          ddLS_HZ[DRMN_DIMS];    // Límites superiores en cada coordenada (Herz)
} drmn_info;



typedef struct {
  double         *dd[DRMN_DIMS];
  drmn_info       info;
```

```
} drmn_datos;



//
drmn_datos *
drmn_inicializaDatos (drmn_datos *);




//
void
drmn_liberaDatos (drmn_datos *);

#define DRMN_LINEA     1024



////////////////////////////////////////////////////////////////////////////////
//
// Devuelve VERDADERO en caso de éxito o FALSO si el archivo no tiene
// exactamente la cantidad de puntos especificado.
//



//
nbool
drmn_leeDatos     (drmn_datos *,
   drmn_arch  *);

#define     DRMN_REF      int
#define     DRMN_REF_X    1
#define     DRMN_REF_Y    2
#define     DRMN_REF_Z    4
#define     DRMN_REF_XY   DRMN_REF_X+DRMN_REF_Y
#define     DRMN_REF_XZ   DRMN_REF_X+DRMN_REF_Z
#define     DRMN_REF_YZ   DRMN_REF_Y+DRMN_REF_Z
#define     DRMN_REF_XYZ  DRMN_REF_X+DRMN_REF_Y+DRMN_REF_Z
```

```
//
typedef struct
{
  double sfrq;     // 6
  double dfrq;     // 1
  double reffrq[2];
  double sw[2];
  double rfl[2];
  double rfp[2];
} drmn_procpar;


////////////////////////////////
//
// Cambia los límites en F1 y F2
// leyendo los datos del archivo
// procpar.

//
nbool
drmn_fLeeProcpar (drmn_procpar *,
  FILE         *archivo);
//
nbool
drmn_sLeeProcpar (drmn_procpar *,
  const char   *nombre_archivo);
//
void
drmn_agregaProcpar  (drmn_datos    *,
     drmn_procpar  *);

//
nbool
drmn_sAgregaProcpar (drmn_datos    *,
     const char    *nombre_archivo);
```

```
//
nbool
drmn_fAgregaProcpar (drmn_datos    *,
     FILE          *archivo);




//
void
drmn_cambiaLimites_PPM (drmn_datos *,
double     inferior,
double     superior,
DRMN_REF   referencia);
//
void
drmn_cambiaLimites_HZ (drmn_datos *,
       double     inferior,
       double     superior,
       DRMN_REF   referencia);




//
nbool
drmn_normalizaDatos (drmn_datos *,
     DRMN_REF  columna);




////////////////////////////////////////////////////////////////////////////////
//
// En las funciones de escritura DRMN_TIPO_ARCH no puede
// ser DRMN_DESCONOCIDO.
//
///
```

```
typedef struct {
  double       dInicio;
  double       dFin;
  DRMN_REF     ref;
  DRMN_ESC     esc;
  long         lTini;
  long         lTfin;
} drmn_intervalo;



// Información de un intervalo
typedef struct {
  long int        lNP;
  long int        lNT;
  long int        jIni,  jFin;
  long int        jTIni, jTFin;
  int             iDir;                   // En datos 3D (espectros 2D),
                                          // -1 indica una rotación en el eje
  long int       *llMinimos[DRMN_DIMS];   // Un mínimo por cada trazo
  long int       *llMaximos[DRMN_DIMS];
} drmn_info_i;




typedef enum {
  DRMN_MINIMO,
  DRMN_MAXIMO,
  DRMN_NMNM         // Ni mínimo ni máximo
} drmn_info_pi;



typedef struct {
  long int      lP;
  long int      lT;   // necesario para saber en qué bloque de datos está
  drmn_info_pi  info;
} drmn_pi;
```

```
typedef struct {
  drmn_pi   *ps;
  long int   lNP;   // cantidad de puntos de inflexión
} drmn_pis;


typedef struct {
  drmn_pis   pis;
  DRMN_REF   y;
  double     dy;
} drmn_pInflexion;


////////////////////////////////////////////////////////////////
//
// Obtiene los puntos de inflexión en un espectros de un solo un trazo,
// en todo el intervalo. Es equivalente a utilizar drmn_damePsInflexion
// con intervalo_a_leer == info_intervalo == NULL.
//
// dy es el valor mínimo entre puntos de inflexión
// en porcentaje (DRMN_PI_DY) con respecto al valor máximo en el eje Y.
//
// Pide memoria para cada punto de inflexión que es liberada con
// drmn_liberaPInflexion.
//
// Devuelve el número de puntos de inflexión encontrados.
//

#define DRMN_PI_DY 0.01        // 1%

long int
drmn_damePInflexion (drmn_pInflexion  *,
     drmn_datos       *,
     double            dy);
//
void
drmn_liberaPInflexion (drmn_pInflexion *);
```

```
////////////////////////////////////////////////////////////
//
// Obtiene todos los puntos de inflexión en el espectro, en el
// intervalo especificado por intervalo_a_leer, o todo el
// espectro si es NULL. Si info_intevalo no es NULL y
// info_intevalo.lNT> 0, se asumirá que ya se tiene
// información del intervalo. Si info_intevalo.lNT <= 0,
// se buscarán los máximos por trazo en dicho intervalo.
// (Si info_intevalo es NULL, también se buscarán, aunque no
// habrá forma de regresar los datos.)
//
// dy es el valor mínimo entre puntos de inflexión
// en porcentaje (DRMN_PI_DY) con respecto al valor máximo en el eje Y
//
// Devuelve la cantidad total de puntos de inflexión encontrados.
// lstPs es una lista de "drmn_pis", que contiene los puntos de
// inflexión en cada trazo.
//


typedef struct {
  nlista     lstPs;
  long int   lNP;    // cantidad de puntos de inflexión
  DRMN_REF   y;      //
  double     dy;
} drmn_psInflexion;


//
long int
drmn_damePsInflexion  (drmn_psInflexion *,
       drmn_datos       *,
       drmn_intervalo   *intervalo_a_leer,
       double            dy,
       drmn_info_i      *info_intervalo);  // obtiene o da información
```

```
                                        //del intervalo (máximos, mínimos...)


void
drmn_liberaPsInflexion (drmn_psInflexion *);









/*************************************************
 * Busca el valor más grande y lo toma como
 * centro para crear una nueva ventana espectral.
 * En las orillas, busca el valor mínimo y recorta
 * el espectro. Si se tienen marcados los límites
 * en otras unidades, las ajusta también a que queden
 * simétricas tomando el cero como centro.
 *
 * Utiliza dy para calcular el tamaño mínimo entre picos.
 * Si dy = -1, se utiliza DRMN_PI_DY del valor máximo.
 *
 * Funciona cuando el número de trazos es uno. Si no es
 * el caso, devuelve FALSO.
 *
 * (¿Es útil para más de un trazo?)
 */
/*
nbool
drmn_centraPico (drmn_datos *, double dy);
*/




////////////////////////////////////////////////////////////////////////////////
//
// Envía los datos en el intervalo (o todos si es NULL)
```

```
// a la función afValor, así como el número de punto leído.
// Si en dicha función se regresa FALSO, se dejan de
// enviar datos y la función devuelve
// FALSO. En caso de mostrarse los datos requeridos, la
// función devuelve VERDADERO.
//



typedef struct {
  long int     lN;
  long int     lT;
} drmn_punto;

//
nbool
drmn_damePuntos (drmn_datos      *,
 drmn_intervalo  *intervalo,
 nbool          (* afPunto) (drmn_punto *, drmn_datos *, void *extra),
 void            *extra );




////////////////////////////////////////////////
//
// Obtiene los puntos inicial y final del intervalo
//
void
drmn_dameLimites (drmn_datos      *,
  drmn_intervalo  *intervalo,
  long int        *punto_inferior,
  long int        *punto_superior);

////////////////////////////////////////////////////////////////////////////////

//
drmn_intervalo *
```

```
drmn_inicializaIntervalo (drmn_intervalo *,
  double         inicio,
  double         fin,
  DRMN_REF       referencia,
  DRMN_ESC       escala,
  long int       trazo_inicial,  // Pred (-1): 0 (si NT=1) o inicio
  long int       trazo_final);   // Pred (-1): 0 (si NT=1)
/**/
drmn_info_i *
drmn_inicializaInfoIntervalo (drmn_info_i *);



/*********************************************************
 * Extrae de la información del intervalo especificado.
 * Si el intervalo es NULL, se obtiene la información
 * simplificada de todo el intervalo (ver la estructura
 * drmn_info_i).
 *
 * Regresa la información del intervalo y pide memoria
 * para los máximos y mínimos que tiene que ser liberada
 * con
 */
drmn_info_i *
drmn_dameInfoI (drmn_info_i    *,
drmn_datos     *,
drmn_intervalo *);



/**/
void
drmn_liberaInfoIntervalo (drmn_info_i   *);





/* Sólo copia de la información de todo el espectro,
 * como si fuera un sólo intervalo. No rotación de
 * valores, no pide memoria.
```

```
 */
drmn_info_i *
drmn_infoAInfoI (drmn_info_i    *,
 drmn_datos     *);




////////////////////////////////////////////////////////////////////////////////



typedef enum {
  DRMN_CENTRAL,
  DRMN_MAYOR,
  DRMN_ESPECIFICO
} drmn_pos;



typedef struct {
  char      achrTitulo[1024];
  int       iDir;               // (+/-)1:  2D: rotar ejes
  DRMN_REF  x;                  // Primera columnas a escribir
  DRMN_ESC  esc;                // En qué escala se escribe.
  // escribe datos:
  long int  lPMax;              // máximo de puntos a escribir (-1 == todos).
                                // Si lPMax > lNP, lPMax==lNP
  // extrae trazo:
  drmn_pos  pos;                // Pred: -1 = DRMN_CENTRAL
} drmn_escribe;



//
drmn_escribe *
drmn_inicializaEscribe (drmn_escribe *,
const char   *titulo,
DRMN_REF      x,                // -1: Sólo Y o Z
                                // (aunque se tengan dos columnas de datos)
DRMN_ESC      escala,           // -1: DRMN_ESC_P (puntos)
```

```
int           dir,              // -1: 1D, invierte datos; Espectros 2D, rotar.
long int      puntos_maximo);   // -1: todos los datos

// con las opciones predeterminadas: x=-1, dir=1, puntos_máximo=-1
drmn_escribe *
drmn_inicializaEscribeP (drmn_escribe *);



////////////////////////////////////////////////////////////////////////////////
//
// Si "detalles" de la escritura == NULL, no se pone título, y se escriben TODOS los
// puntos, pero SÓLO los contenidos en la estructura de datos.
//
//

//
nbool
drmn_fEscribeDatosXYZ  (FILE            *,
drmn_datos      *,
drmn_escribe    *detalles,
drmn_intervalo  *intervalo_a_escribir,
drmn_info_i     *info_intervalo);  // Si != NULL, obtiene máximos
                                                        // y mínimos locales al intervalo
// Si intervalo_a_escribir = NULL, se escriben TODOS los
// datos en el
nbool
drmn_sEscribeDatosXYZ  (const char      *nombre_arch,
drmn_datos      *,
drmn_escribe    *detalles,
drmn_intervalo  *intervalo_a_escribir,
drmn_info_i     *info_intervalo);



////////////////////////////////////////////////////////////////////////////////
//
// Inicializan la estructura para la extracción de trazos
// (detalles de escritura)
//
```

```
// Opciones predeterminadas;
//
//  titulo     = NULL
//  pos_trazo  = -1    =  DRMN_CENTRAL
//  x          = -1    =  Ninguna columna a escribir
//  escala     = -1    =  DRMN_ESC_PPM si x=(DRMN_REF_X, DRMN_REF_Y)
//  dir        = 1     =  1D: invierte los datos;  2D: rota los datos (1=f1, -1=f2)
//



//
drmn_escribe *
drmn_inicializaEscribeTrazo (drmn_escribe  *,
     const char    *titulo,
     drmn_pos       pos_trazo,
     DRMN_REF       x,
     DRMN_ESC       escala,
     int            dir);

// Inicializa la estructura con las opciones predeterminadas
//
drmn_escribe *
drmn_inicializaEscribeTrazoP (drmn_escribe  *);




////////////////////////////////////////////////////////////////
//
// Extrae el trazo máximo especificado por el intervalo_a_escribir
//
// Si detalles_escritura = NULL, se toman los parámetros
// predeterminados: trazo mayor central (pos = DRMN_CENTRAL) y
// primer dimensión espectral (iDir = 1 = f1).
//
// Si se trata de un espectro de un sólo trazo, las funciones son
// equivalentes a drmn_xEscribeDatosXYZ, donde x=(f,s)
//
nbool
```

```
drmn_fExtraeTrazoMaxXY  (FILE           *,
 drmn_datos      *,
 drmn_escribe    *detalles_escritura,
 drmn_intervalo  *intervalo_a_escribir,
 drmn_info_i     *info_intervalo);
//
nbool
drmn_sExtraeTrazoMaxXY  (const char      *nombre_arch,
 drmn_datos      *,
 drmn_escribe    *detalles_escritura,
 drmn_intervalo  *intervalo_a_escribir,
 drmn_info_i     *info_intervalo);



////////////////////////////////////////////////////////////////
//
// Escribe un archivo XEASY con los datos
//
// Si los datos tienen más de un trazo, se extrae el
// trazo de acuerdo a los detalles_escritura o bien,
// si es NULL, a los parámetros predeterminados, esto es,
// el trazo mayor más cerca del centro (pos = DRMN_CENTRAL)
// y la primera dimensión espectral (iDir = 1 = f1).
//
//
nbool
drmn_fEscribeDatosXEASY  (FILE            *,
  drmn_datos      *,
  drmn_escribe    *detalles_escritura,  // normalmente NULL para
                                        // espectros con un solo trazo
  drmn_intervalo  *intervalo_a_escribir,
  drmn_info_i     *info_intervalo);

//
nbool
drmn_sEscribeDatosXEASY  (const char      *nombre_arch,
  drmn_datos      *,
```

```
  drmn_escribe    *detalles_escritura,  // normalmente NULL para
                                        //espectros con un solo trazo
  drmn_intervalo  *intervalo_a_escribir,
  drmn_info_i     *info_intervalo);




typedef enum
  {
    DRMN_DELTA_SIM,
    DRMN_DELTA_ANTI_IZQ,
    DRMN_DELTA_ANTI_DER
  } drmn_tipoDelta;



typedef struct
{
  int  n;
  int *aiPicos;
} drmn_func_delta;

////////////////
// FUNCIÓN DELTA
//
nbool
drmn_creaFuncDelta (drmn_func_delta *,
    int              n_picos,   // valores distintos de cero
    drmn_tipoDelta   tipo);
/**/
void
drmn_liberaFuncDelta (drmn_func_delta *);



/**/
nbool
drmn_creaSumaJPs (drmn_datos      *suma,
  drmn_datos      *datos_espectro,
  drmn_func_delta *func_delta,
```

```
  double           J_maxima);




/* Crea la convolución con el valor de J
 */
nbool
drmn_dameConvolucion (drmn_datos      *convolucion,
      drmn_datos      *datos_espectro,
      drmn_func_delta *func_delta,
      double           J);



/* Recorta la convolución al tamaño de la J
 */
nbool
drmn_recortaConvolucion (drmn_datos    *convolucion,
 drmn_datos    *datos_espectro,
 double         J);



#define DRMN_TESIS
#ifdef DRMN_TESIS
extern int DRMN_iSuma;
extern int DRMN_iConvol;
#endif


#endif
```

Bibliografía

[1] Ray Freeman. *Spin Choreography. Basic Steps in High Resolution NMR*, page 408. Oxford University Press, Estados Unidos de Norteamérica, 1999.

[2] Acely Garza-García, Giaconda Ponzanelli-Velázquez, and Federico del Río-Portilla. Deconvolution and measurement of spin-spin splittings by modified j doubling in the frequency domain. 148:214–219, 2001.

[3] Harald Günther. Die bedeutung des vorzeichens der allylischen kopplungskonstanten (^4j_t) für die konformationsanalyse. *Z. Naturforschg.*, 24b:680–685, 1969.

[4] M. Martín-Pastor J. C. Cobas, V. Constantino-Castillo and F. del Río-Portilla. A two-stage approach to automatic determination of ^1h nmr coupling constants. *Magn. Reson. Chem.*, 43:843–848, 2005.

[5] James Keeler. *Understanding NMR Spectroscopy.* John Wiley and Sons, 2007.

[6] Donald A. McQuarrie and John D. Simon. *Physical Chemistry.* University Science Books, Estados Unidos de Norteamérica, 1997.

[7] Geoffrey Bodenhausen Peter Huber. Simplification of multiplets by deconvolution in one- and two-dimensional nmr spectra. *J. Magn. Reson.*, Serie A, 102:81–89, 1993.

[8] W. A. Thomas. Unravelling molecular structure and conformation—the modern role of coupling constants. *Prog. NMR Spectrosc.*, 30:183–207, 1997.

Índice alfabético