



UNIDAD QUERETARO

Desactivación eléctrica en cristales de silicio altamente dopado con arsénico

Tesis que presenta

Joel Andrés Calderón Guillén ✓

**CINVESTAV
IPN
ADQUISICION
DE LIBROS**

para obtener el Grado de

Maestro en Ciencias

en la Especialidad de

Materiales

Director de tesis: Dr. Alberto Herrera Gómez

Santiago de Querétaro, Qro. noviembre de 2002

CINVESTAV I. P. N.
SECCION DE INFORMACION
Y DOCUMENTACION

CLAS: _____
ADQUIS: TESIS-03, 017
FECHA: 20 MAYO-2003
PROCE: Serv. Bibliotecas
S _____

Agradecimientos

Agradezco y dedico muy especialmente esta tesis a:

 Mi padre: Eduardo Calderón Amador †

Pues donde quiera que se encuentre estará satisfecho, ya que su esfuerzo de ofrecernos una educación no fue en vano.

 Mi madre: Trinidad Guillén Armenta

Quien con su bondad y sentimientos siempre esta conmigo.

 Mi compañera y esposa: Maxi Gpe. Luque Mascareño

Ya que su apoyo y comprensión ha sido determinante en la realización de esta meta.

 A mi hijo: Joel Andrés

Pues con sus sonrisas y juegos me enseña que la vida es graciosa, hermosa y llevadera.

 A mis familiares y amigos

A todos ellos, suegros, hermanos y compañeros de generación gracias por apoyarnos mutuamente y por tolerarnos durante este tiempo.

 Escuela de Ingeniería de Los Mochis de la U.A.S.

Quienes con su apoyo económico ha sido posible llegar a concluir estos estudios. Especialmente quiero agradecer al Ing. Eleazar Luna Barraza por su desinteresado compañerismo que ha mostrado.

CINVESTAV-QRO

Pues la eficiencia de cada uno de sus trabajadores se ve reflejado en los logros obtenidos y este en especial, es parte ellos. En especial quisiera resaltar la enorme labor que realiza el personal a cargo de la Coordinación Académica que acertadamente dirige el Dr. Arturo Mendoza Galván.

 Investigadores del CINVESTAV-QRO

Ya que las enseñanzas obtenidas de todos ellos han sido de gran ayuda en la culminación del presente trabajo, además de la confianza y amistad que me han brindado. El gran esfuerzo que han realizado cada uno de ellos en sacar adelante este proyecto de maestría es digno de alabarse felicidades a todos.

Existen dos personas de la que no puedo pasar por alto:

 A mi compañero: M.C. Servando Aguirre Tostado

Quien de forma muy entusiasta y desinteresada estuvo apoyándome con sus conocimientos durante la realización del presente trabajo

 A mi director de tesis: Dr. Alberto Herrera Gómez

Porque su apoyo no solo se limitó a dirigir acertadamente esta tesis, sino que además de sus brillantes ideas, me ofreció una confianza y amistad, mismas que favorecieron el ambiente de trabajo.

Por último expreso mi agradecimiento al Consejo Nacional de Ciencia y Tecnología por la beca otorgada, al Centro de Investigación de Estudios Avanzados del I.P.N por darme la oportunidad de ingresar y permanecer en esta Unidad Querétaro y a la Universidad Autónoma de Sinaloa que con el Programa de Mejoramiento del Profesorado a hecho posible que un servidor esté en estos momentos por obtener el grado de Maestro en Ciencias.

Contenido

Índice de Tablas	iii
Índice de Figuras	iv
Resumen	ix
Abstract	x
I. Introducción y Antecedentes	1
I.A. Descripción de la Tesis	1
I.B. Antecedentes	1
I.B.1. Dopado de silicio	1
I.B.2. Ruta crítica de desarrollo tecnológico (Roadmap)	2
I.C. Descripción de EXAFS	4
I.C.1. Fuente de luz	4
I.C.2. Espectro de absorción EXAFS	9
I.D. Cálculos teóricos de los parámetros EXAFS	16
I.E. Estudios de SIMS	16
I.F. Descripción del proyecto	17
I.F.1. Elaboración y caracterización de las muestras	17
I.F.2. Descripción general del trabajo a desarrollar	18
Bibliografía	19
II. Objetivos	21
II.A. Objetivo General	21
II.B. Objetivos específicos	21
III. Resultados Experimentales	22
III.A. Resultados de EXAFS	22
III.B. Tratamiento de los datos experimentales de XAS	23
III.B.2. El espacio k	24
III.B.3. El espacio R	25
III.C. Estudios de SIMS	30
IV. Descripción del Modelo Físico	33
IV.A. Proceso de desactivación eléctrica	33
IV.B. Modelos Teóricos	33
IV.C. Formulación Matemática	34
IV.C.1. Sistema de Referencia	34
IV.C.2. Trayectorias Modelo 1 (V-As ₄)	36
IV.C.3. Trayectorias Modelo 2 (As ₃ -V-Si)	41
IV.C.4. Trayectorias Modelo 3 (As ₂ -V-Si ₂)	46
IV.C.5. Trayectorias Modelo 4 (Si ₄ -As ₁)	51
Bibliografía	54
V. Descripción del Software	55
V.A. Esquema de Funcionamiento del Programa	56
V.A.2. Módulo "Data"	57
V.A.3. Módulo "Models"	57
V.A.4. Módulo "Fitting Parameters"	58
V.A.5. Módulo "Data and Fit Plot K-space"	59
V.A.6. Módulo "Data and Fit Plot R-space"	59

V.A.7. Módulos “Result Plot” y “Result Table”	60
V.A.8. Módulo “File Editor”	60
V.A.9. Módulo “Miscellaneous”	61
V.B. Diagrama de flujo de la rutina de ajuste	61
V.B.2. Subrutina ObtainBestParametersLevenberg()	65
Bibliografía	69
VI. Análisis y Discusión	71
VI.A. Consideraciones generales	71
VI.B. Consideraciones especiales durante el ajuste	72
VI.C. Discusión de los resultados del análisis	74
VII. Conclusiones y Perspectivas	76
VII.A. Conclusiones	76
VII.B. Perspectivas y trabajo futuro	76
Apéndice A: Fases y Amplitudes.....	77
A. Modelo 1 (V-As ₄)	77
B. Modelo 2 (Si-V-As ₃)	80
C. Modelo 3 (Si ₂ V-As ₂)	81
D. Modelo 4 (Si ₄ -As)	82
Apéndice B: Código fuente de la rutina de ajuste.....	83
A. Programa principal	83
B. FitAllFiles()	83
C. CreateFreeVariablesVectorWithModel()	85
D. GetXExpChainandYExpChainandTotNumP()	87
E. GetAllFits()	88
F. ObtainBestParametersLevenberg()	89
G. mrqmin()	91
H. GetCalculationsAndIAs()	93
I. mrqcof()	94
J. CalculateYCalChainAndGradientCalChain()	95
K. CalculateYCalChainAndGradientCalChainAbs()	95
L. CalculateYCalChainAndGradientCalChainReAndIm()	96
M. :CopyNewValuesToFreeParameters()	97
N. GetAllGradientsKspace()	99
O. GetAllGradientsRspace()	102
P. GetChiSqModels()	106
Q. ChiSqModels()	108
R. GradChiSqModels()	109
S. GradChiSqModelsAbs()	109
T. GradChiSqModelsReAndIm()	118
U. GetFitAndChisqModels()	131
V. GetOneFitAndChiSqrModels()	132
W. GetOneFitAndChiSqrModelsAbs()	132
X. GetOneFitAndChiSqrModelsReAndIm	133
Bibliografía	136

Índice de Tablas

Tabla I.1	Nomenclatura de las muestras a estudiar	18
Tabla I.2	Nomenclatura adicional de acuerdo al espacio de ajuste de los datos experimentales	18
Tabla VI.1	Porcentajes de cada uno de los modelos en el ajuste de los datos experimentales para las muestras de 500-900 °C de tratamiento térmico, comparadas con la de sin tratamiento (ST).	74
Tabla .1	Concentrado de las diferentes trayectorias que tiene el fotoelectrón para una estructura correspondiente al modelo 1	77
Tabla .2	Fases y amplitudes calculadas a través del programa feff.exe de la Universidad de Washington	80
Tabla .3	Listado de Feffxxx.dat para el modelo, mostrando únicamente las primeras 30 trayectorias	81
Tabla .4	Listado de feffxxx.dat, obsérvese la variación en la degeneración (deg) y la relación de amplitud con respecto al modelo anterior	82
Tabla .5	Listado de feffxxx.dat para el modelo Si ₄ As	82

Índice de Figuras

Figura I.1	Resultados de los cálculos para la energía total de tres diferentes configuraciones de arsénico en silicio. Las energías por átomo de As están dadas para una configuración ideal (líneas punteadas) y para una configuración relajada (líneas sólidas), haciendo igual a cero la energía idealizada de un átomo de As aislado	3
Figura I.2	Distribución de sincrotrones en todo el mundo el cuadro de abajo podría incorporarse en la figura	5
Figura I.3	Brookhaven National Laboratories, en New York	6
Figura I.4	Vista superior del laboratorio, muestra el anillo de VUV que opera con energías del electrón hasta de 800 MeV y el anillo de rayos X opera con energías del electrón hasta de 2.5 GeV.	6
Figura I.5	Imagen que muestra una serie de anillos magnéticos tanto para evitar que choquen los electrones con las paredes del anillo, como para restituir la energía que se pierde en cada cambio de dirección.	7
Figura I.6	Cambio en la dirección de un paquete de electrones, produce la radiación sincrotrón.	7
Figura I.7	Esquema de funcionamiento de una línea de Luz Sincrotrón	8
Figura I.8	Esquema que muestra el experimento realizado para obtener los datos de XAS (Espectroscopia de Absorción de Rayos X, por sus siglas en inglés)	8
Figura I.9	La fuente de radiación Sincrotrón puede cubrir un ancho amplio de ondas electromagnéticas, que va desde las micro ondas hasta rayos gamma.	9
Figura I.10	Esquema que muestra la interacción de la luz con la materia, así como los diferentes fenómenos que pueden ser observados y medidos.	10
Figura I.11	Espectro de absorción característico de un estudio XAS.	11
Figura I.12	Absorción y emisión de un fotoelectrón proveniente del nivel 1s que viaja como ondas esféricas	12
Figura I.13	Trayectorias posibles que puede seguir un fotoelectrón, tenemos dispersión simple (a) y dispersión múltiple (b).	13

Figura I.14	Espectro de XAS donde se muestra la diferencia en la absorción de luz de átomos aislados y dentro de una estructura. Las oscilaciones son resultado de la presencia de vecinos. Por esta razón, EXAFS provee información sobre la estructura local, desglosado donde se muestra la contribución del átomo que absorbe.	14
Figura I.15	Gráfico de EXAFS normalizado, correspondiente a la muestra sin tratamiento térmico.	15
Figura I.16	Obsérvese como un ion de oxígeno de alta energía, incide sobre la superficie y desprende iones de las especies presentes, que son detectadas por un analizador de masas.	16
Figura I.17	Diagrama de flujo general para el desarrollo del proyecto	17
Figura I.18	Diagrama de flujo general del trabajo a desarrollar durante la ejecución del proyecto de tesis	19
Figura III.1	Datos de XAS obtenidos directamente en el sincrotrón de Brookhaven National Laboratories en New York.	22
Figura III.2	Datos obtenidos de los estudios de XAS que corresponden a la región extendida oscilatoria (EXAFS), la primera sin tratamiento térmico (S/T) y el resto con tratamientos de 500-1000 °C.	23
Figura III.3	Espectro completo de un estudio XAS del As K-edge	24
Figura III.4	Datos de EXAFS en el espacio k , pesados con k^1	25
Figura III.5	Datos de EXAFS en el espacio R, muestra Sin Tratamiento térmico y muestra tratada a 400°C	26
Figura III.6	Datos de EXAFS en el espacio R, imágenes sobrepuestas de las muestras a ST, 500 y 600°C de tratamiento térmico	26
Figura III.7	Datos de EXAFS en el espacio R de muestras tratadas térmicamente de 600-1000°C.	27
Figura III.8	Datos de EXAFS, parte real de la transformada de Fourier	28
Figura III.9	Datos de EXAFS, parte imaginaria de la transformada de Fourier	29
Figura III.10	Figura que muestra los datos obtenidos por SIMS y como se observa a 4000Å de la superficie tenemos una cantidad significativa de dopantes por arriba de 7.00E+020 As/cm ³	31

Figura IV.1	Modelos físicos propuestos fundamentados por Pandey et al.	34
Figura IV.2	Localización de los primeros y segundos vecinos a una Vacancia (o arsénico) respecto al sistema X-Y-Z.	35
Figura IV.3	Traslación de ejes de referencia de la vacancia al átomo de arsénico que absorbe.	35
Figura IV.4	Trayectoria 1. Nombre: Feff0001. Contribución: 100%. Número de ramales: 2.	37
Figura IV.5	Trayectoria 2. Nombre Feff0002. Contribución 71.91%. Número de ramales: 2.	37
Figura IV.6	Trayectoria 3. Nombre Feff0003. Contribución 29.78%. Número de ramales: 2.	37
Figura IV.7	Trayectoria 4. Nombre Feff0004. Contribución 3.875%. Número de ramales: 3. d_r es la distancia total promedio del recorrido de la trayectoria	38
Figura IV.8	Trayectoria 5. Nombre Feff0005. Contribución 27.9%. Número de ramales: 3	38
Figura IV.9	Trayectoria 6. Nombre Feff0006. Contribución 58.41%. Número de ramales: 2	39
Figura IV.10	Trayectoria 8. Nombre Feff0008. Contribución 4.48%. Número de ramales: 4	39
Figura IV.11	Trayectoria 9. Nombre Feff0009. Contribución 3.80%. Número de ramales: 3	40
Figura IV.12	Trayectoria 10. Nombre Feff0010. Contribución 12.20%. Número de ramales: 3.	40
Figura IV.13	Trayectoria 1. Nombre Feff0001. Contribución 100%. Número de ramales: 2	41
Figura IV.14	Trayectoria 2. Nombre Feff0002. Contribución 79.91%. Número de ramales: 2	42
Figura IV.15	Trayectoria 3. Nombre Feff0003. Contribución 19.856%. Número de ramales: 2	42

Figura IV.16	Trayectoria 4. Nombre Feff0004. Contribución 3.872%. Número de ramales: 3	43
Figura IV.17	Trayectoria 5. Nombre: Feff005. Contribución: 27.877%. Número de ramales: 3.	43
Figura IV.18	Trayectoria 6. Nombre: Feff006. Contribución: 58.41%. Número de ramales: 2.	44
Figura IV.19	Trayectoria 8. Nombre: Feff008. Contribución: 4.472%. Número de ramales: 4.	44
Figura IV.20	Trayectoria 9. Nombre: Feff009. Contribución: 4.428%. Número de ramales: 3.	45
Figura IV.21	Trayectoria 10. Nombre: Feff010. Contribución: 12.19%. Número de ramales: 3. En la figura se colocó únicamente una trayectoria genérica para los tres tipos existentes.	46
Figura IV.22	Trayectoria 1. Nombre Feff0001. Contribución 100%. Número de ramales: 2	47
Figura IV.23	Trayectoria 2. Nombre Feff0002. Contribución 87.905%. Número de ramales: 2	47
Figura IV.24	Trayectoria 3. Nombre Feff0003. Contribución 9.929%. Número de ramales: 2	48
Figura IV.25	Trayectoria 4. Nombre Feff0004. Contribución 3.869%. Número de ramales: 3	48
Figura IV.26	Trayectoria 5. Nombre: Feff005. Contribución: 27.848%. Número de ramales: 3.	49
Figura IV.27	Trayectoria 6. Nombre: Feff006. Contribución: 58.41%. Número de ramales: 2.	49
Figura IV.28	Trayectoria 8. Nombre: Feff008. Contribución: 4.472%. Número de ramales: 4.	50
Figura IV.29	Trayectoria 9. Nombre: Feff009. Contribución: 5.058%. Número de ramales: 3.	50
Figura IV.30	Trayectoria 10. Nombre: Feff010. Contribución: 12.18%. Número de ramales: 3.	51

Figura IV.31	Trayectoria 1. Nombre Feff0001. Contribución 100%. Número de ramales: 2	52
Figura IV.32	Trayectoria 2. Nombre Feff0002. Contribución 71.9%. Número de ramales: 2	52
Figura IV.33	Trayectoria 3. Nombre Feff0003. Contribución 5.8%. Número de ramales: 3	53
Figura IV.34	Trayectoria 4. Nombre: Feff004. Contribución: 27.8%. Número de ramales: 3.	53
Figura IV.35	Trayectoria 5. Nombre: Feff005. Contribución: 43.81%. Número de ramales: 2.	54
Figura V.1	diagrama de funcionamiento global del software desarrollado	56
Figura V.2	Esquema que muestra las virtudes del modulo para la captura de los datos experimentales, tambien se pueden tratar los datos para realizar presentaciones en otro software como por ejemplo "Origen"	57
Figura V.3	Esquema que muestra el manejo de las variables que intervienen en la ecuación de EXAFS. Para cada modelo se realizó este mismo modulo.	58
Figura V.4	Diagrama de flujo del programa principal de ajuste de datos	62
Figura V.5	Diagrama de flujo de FitAllFiles ()	63
Figura V.6	Continuación del diagrama de flujo de FitAllFiles().	64
Figura V.7	Diagrama de flujo de la subrutina ObtainBestParametersLevenberg()	65
Figura V.8	Continuación del diagrama de flujo de la subrutina ObtainBestParametersLevenberg().	66
Figura V.9	Continuación del diagrama de flujo de la subrutina ObtainBestParametersLevenberg().	67
Figura V.10	Diagrama de flujo de subrutina mrqmin()	68
Figura VI.1	Ajuste de cuatro archivos de datos experimentales	73
Figura VI.2	En esta figura se muestra la influencia de cada uno de los modelos y su variación con respecto a la temperatura de tratamiento	74
Figura VI.3	Estructura del defecto producido por la vacancia	75

Resumen

El silicio puede ser dopado con arsénico a niveles por arriba de su solubilidad, resultando en un sistema en cuasiequilibrio. La desactivación eléctrica de los átomos de arsénico es lograda a temperatura ambiente en periodos mayores a los diez años, y en segundos a 1000°C. En este trabajo se reporta la estructura de los defectos responsables de la desactivación eléctrica de impurezas de arsénico en cristales de silicio cuando estas muestras son sometidas a tratamientos térmicos entre 500 y 1000°C. En un trabajo anterior, sustratos de silicio(100) fueron implantados con arsénico y licuados por láser, logrando así una densidad activa y uniforme de $\sim 1 \times 10^{21}$ átomos por cm^3 en los primeros 4000Å de la oblea. Esto corresponde a una concentración activa cinco veces por encima de su nivel de equilibrio a temperatura ambiente. El perfil de dopaje fue determinado a través de estudios de SIMS (Secondary Ion Mass Spectroscopy). En la literatura existe un cierto consenso de que la desactivación eléctrica es causada por agregados de arsénico alrededor de vacancias ($\text{As}_4\text{-V}$, $\text{As}_3\text{-V-Si}$ y $\text{As}_2\text{-V-Si}_2$). La técnica de Extended X-Ray Absorption Fine Structure (EXAFS), aplicada al nivel As 1s, fue usada para determinar los detalles de la estructura local de estos agregados y su concentración en función de la temperatura. Para incorporar las simetrías de estos defectos en el análisis de los datos EXAFS, se creó un software de más de 22 mil líneas. La deformación local se describió con parámetros adimensionales correspondiente a los desplazamientos axiales de los primeros y segundos vecinos centrados en la vacancia. Los parámetros que se contemplan en el software son, además de aquellos incluidos en la teoría de EXAFS, los parámetros adimensionales arriba mencionados. Los resultados obtenidos en el ajuste muestran una concordancia con la investigación realizadas por Berding et al., la cual reporta que la presencia del agregado del tipo $\text{As}_2\text{-V-Si}_2$ se incrementa con la temperatura de tratamiento. Chadi, Pandey y otros calcularon, por primeros principios, que la longitud del arsénico con su primer vecino silicio es del orden de 2.38 y 2.40Å, respectivamente, mientras que nuestros ajustes mostraron una longitud de 2.32-2.38Å dependiendo del modelo.

Abstract

Silicon can be doped with arsenic at levels above its solubility limit, yielding to systems in quasiequilibrium. At room temperature, the lifetime of electrically-active arsenic impurities is longer than ten years, although it is of the order of seconds at 1000 °C. From previous reports (experimental and first principle calculations), it was proposed that the electrical deactivation is produced by arsenic cluster around the vacancies (As_4-V , As_3-V-Si , As_2-V-Si_2 , $As-Si_4$). In this work it is reported the detail of the structure of these defects, and its dependence with temperature (500 to 1000 °C). In a previous work, silicon (001) wafers were implanted with arsenic impurities through laser melting, obtaining an active and uniform density of $\sim 1 \times 10^{21}/\text{cm}^3$ in the first 4000 Å of the substrate. This corresponds an active concentration five times above the saturation levels at room temperature. The doping profile was determined by Secondary Ion Mass Spectroscopy (SIMS). The details of the local structure was obtained through Extended X-ray Absorption Fine Structure (EXAFS) applied to the 1s level of arsenic. To incorporate the symmetry of this defects in the data analysis of EXAFS, it was developed an ad hoc software of more than 22,000 lines. The local deformation of the first and second neighbors was described with non-dimensional parameters corresponding to the axial displacements from the vacancy. The parameters allowed to vary in the software are, besides those included in the EXAFS theory, non-dimensional parameters mentioned above. The results shows a concordance with the theoretical predictions by Berding et al. The concentration of the As_2-V-Si_2 cluster increases with the annealing temperature. Chadi, Pandey et al. calculated from first principles a first-neighbor As-Si distance of 2.38 at 2.40 Å, and our results shown a distance in the range of 2.32 to 2.38 e of 2.32 to 2.38 Å, depending of the model.

I. Introducción y Antecedentes

I.A. Descripción de la Tesis

El presente proyecto consistió en determinar los procesos de desactivación eléctrica en silicio altamente dopado con arsénico, para lo cual se tuvo que implementar un software, que auxiliara en la tarea de ajuste de los datos experimentales obtenidos por EXAFS¹ (Extended X-ray Absorption Fine Structure) en el sincrotrón del Brookhaven National Laboratories, en Nueva York. Para el análisis de los datos se propuso un modelo físico basado en las investigaciones ya realizadas,² el cual se muestra de manera amplia en el Capítulo III. La caracterización experimental de las muestras de silicio altamente dopado con arsénico consistente en los estudios de EXAFS y SIMS (Secondary Ion Mass Spectroscopy) fue obtenida anteriormente, tal como se describe en el Capítulo II. La metodología empleada en la caracterización realizada se muestra en el Capítulo IV, conteniendo también un análisis completo así como las conclusiones que se lograron.

I.B. Antecedentes

I.B.1. Dopado de silicio

El dopaje por difusión ha sido por más de 40 años la tecnología usada por la industria semiconductora para la introducción de impurezas en los circuitos integrados. Los átomos dopantes del silicio más comunes son los elementos del grupo III y V de la tabla periódica. Los dopantes han sido clasificados como donadores o aceptores, dependiendo de si da o acepta electrones al incorporarse al semiconductor. Para el silicio los dopantes donadores más comunes son: Fósforo (P), Antimonio (Sb) y Arsénico (As). Los dopantes aceptores más comunes son: Boro (B), Galio (Ga), Indio (In) y Aluminio (Al).

La difusión en silicio es considerablemente más complicado que una predicción simple de la ley de Fick. Un concepto básico de la teoría de difusión es que los átomos dopantes (para el caso del arsénico) difunden en forma sustitucional en la red de silicio y se fijan con la presencia de ciertos defectos, perdiendo además su contribución de portadores eléctricos. La formación de agregados de arsénico influencia fuertemente la difusión de impurezas a altas concentraciones y por años a sido de gran interés el estudio de la formación de agregados de arsénico impurificado

en silicio en altos niveles de dopado, pues se ha comprobado su afectación en la desactivación eléctrica con los tratamientos térmicos.

I.B.2. Ruta crítica de desarrollo tecnológico (Roadmap)

Organismos internacionales tales como SIA³ (Semiconductor Industry Association), SRC³ (Semiconductor Research Corporation), SEMATECH,⁴ HP, Intel, y Motorola, entre otros, han coincidido en la necesidad de lograr dispositivos electrónicos del orden de 0.1 micras, lo cual conlleva a diseñar semiconductores con altos niveles de conductividad.⁴ Una forma de obtener estos materiales es con impurezas eléctricamente activas con una concentración del orden de $10^{21}/\text{cm}^3$ Aunque existen métodos de lograr estas concentraciones,⁵ cuando se dan tratamientos térmicos la resistividad aumenta.⁶

Los fenómenos de desactivación eléctrica han sido ampliamente estudiados.⁷ Algunos autores han propuesto que el defecto responsable de la desactivación eléctrica consiste de agregados de arsénico alrededor de una vacancia,^{2,9} mientras que otros han propuesto otras estructuras completamente diferentes.⁸ A pesar del gran número de trabajos sobre el tema, hasta el momento no se ha logrado establecer con precisión la estructura del defecto, ni mucho menos la dependencia que estos tienen con la temperatura. Los organismos arriba mencionados establecieron en 1995 la necesidad de realizar estudios de EXAFS para determinar de una manera más precisa la estructura de los defectos, proponiendo a la vez la urgencia de establecer modelos apropiados para sus procesos de producción.

Muchas de las investigaciones realizadas en silicio altamente dopado con arsénico han sido encaminadas a proponer la formación de agregados de arsénico a través de modelos teóricos²

- En 1988 K. C. Pandey,⁹ y G.S. Cargill III,⁹ calcularon la energía total para la formación de tres diferentes configuraciones de arsénicos en silicio SiAs_4 , Si_4As , Vac-As_4

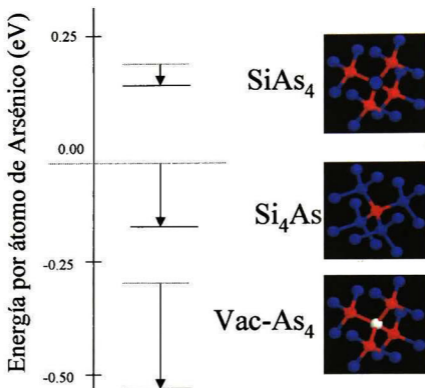


Figura I.1 Resultados de los cálculos para la energía total de tres diferentes configuraciones de arsénico en silicio. Las energías por átomo de As están dadas para una configuración ideal (líneas punteadas) y para una configuración relajada (líneas sólidas), haciendo igual a cero la energía idealizada de un átomo de As aislado

La relajación tiene su más grande efecto en la configuración Vac-As₄. Se obtuvo una energía libre de 0.35 eV por átomo de As más bajo que la configuración Si₄As mientras que el agregado SiAs₄ tiene la más alta energía de 0.27 eV por átomo de arsénico, más alta que la configuración de un átomo aislado de As. como está mostrado en la Figura I.1.

- Posteriormente M. Berding et al. (1998),² muestra que los agregados con energía favorable para su formación durante los tratamientos térmico a altas temperaturas son: Si₄As , Si-Vac-As₃ ,Si₂-Vac-As₂, Vac-As₄, lo cual no contradice los resultados obtenidos por Pandey.et al.
- Ese mismo año (1988) D.J. Chadi et al¹⁰, proponen un modelo también referido a arsénicos alrededor de la vacancia considerando distorsiones angulares producidas por la atracción mutua entre los arsénicos, estas distorsiones no serán por lo pronto consideradas en el presente proyecto.
- Hay quienes se enfocaron a investigar la formación de agregados del tipo As_m,^{11,12} (donde m = 5,6,7,... átomos) con alguna evidencia de precipitación observada a través de estudios de

TEM, argumentando que la precipitación es un importante mecanismo de desactivación eléctrica en el semiconductor,¹³

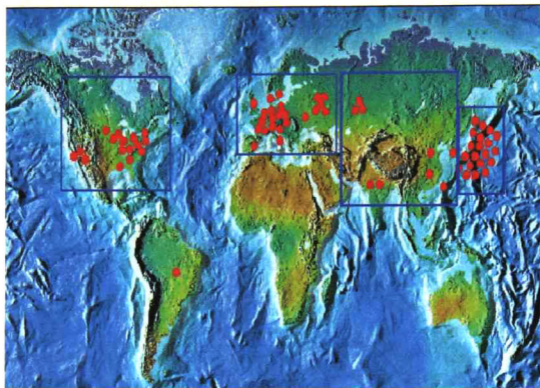
- A través de experimentación con XSW A. Herrera-Gómez et al,¹⁴ mostraron que el arsénico desactivado continúa ocupando sitios sustitucionales dentro de la red de silicio.

Haciendo un análisis de los resultados teóricos y experimentales obtenidos por Pandey et al.,⁹ y Berding et al.² y adoptando sus modelos, el presente estudio se enfoca a medir experimentalmente la dependencia de la concentración de los defectos propuestos con la temperatura, así como la estructura detallada de estos. Esto ayudará a predecir en el futuro los niveles de desactivación eléctrica para un semiconductor de silicio altamente dopado con arsénico, cuando es sometido a tratamientos térmicos.

I.C. Descripción de EXAFS

I.C.1. Fuente de luz

EXAFS implica la incidencia de rayos X de energía cercana a algún borde de absorción del átomo que se desee excitar. Un aspecto muy importante es la fuente de luz, la cual consiste de un sincrotrón. Estos laboratorios se encuentran dispersos en todo el mundo de la manera mostrada en la Figura I.2.



Asia y Sud América: ~10, Norte América:~20
 Europa:~20, Japón: ~20

Figura I.2 Distribución de sincrotrones en todo el mundo el cuadro de abajo podría incorporarse en la figura

Lo costoso del equipo y su mantenimiento los hacen poco factibles para su construcción en países con poca capacidad económica. Sin embargo, la visión de los japoneses y su potencial en este rubro es grande pues sus sincrotrones son aplicados totalmente a la industria. En los casos de Europa y Asia son utilizados primordialmente para la investigación científica, mientras que en Norte América combina ambos aspectos (industria e investigación). Aunque su acceso está muy restringido por obvias razones, siempre hay disposición de los responsables de estos centros de investigación para facilitar las instalaciones a grupos de trabajo en estas áreas. Un sincrotrón es un acelerador de partículas ya sea circular o recto. Un ejemplo de ello es la Figura I.3.



Figura I.3 Brookhaven National Laboratories, en New York

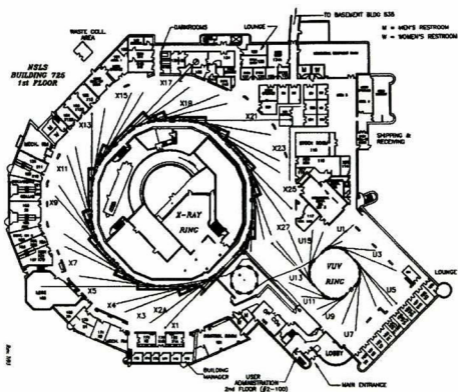


Figura I.4 Vista superior del laboratorio, muestra el anillo de VUV que opera con energías del electrón hasta de 800 MeV y el anillo de rayos X opera con energías del electrón hasta de 2.5 GeV.

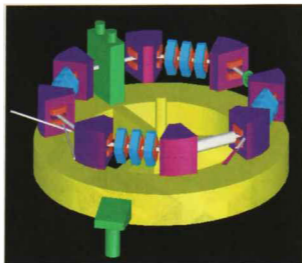


Figura I.5 Imagen que muestra una serie de anillos magnéticos tanto para evitar que choquen los electrones con las paredes del anillo, como para restituir la energía que se pierde en cada cambio de dirección.

Obsérvese que la radiación sincrotrón se emite en cada cambio de dirección del haz y va dirigido a cabinas de trabajo para su uso en experimentación, un detalle de la emisión de la radiación sincrotrón se muestra en la Figura I.6.

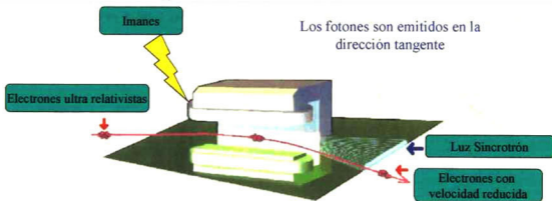


Figura I.6 Cambio en la dirección de un paquete de electrones, produce la radiación sincrotrón.

Cuando la radiación sincrotrón es emitida, ésta es confinada, colimada y dirigida hacia un monocromador para después incidir en la muestra, tal como se muestra en la Figura I.7

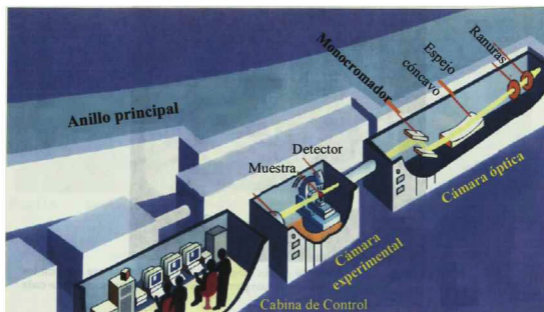


Figura I.7 Esquema de funcionamiento de una línea de Luz Sincrotrón

Cuando el haz sale de la fuente llega a un monocromador y a unas aberturas que regulan el ancho y dirección del haz (ver Figura I.8). De ahí pasa a una cámara donde se mide su intensidad (I_0) para después incidir en la muestra y medir en los detectores que se tengan ya sea la transmitancia (I_T) o bien la fluorescencia. En nuestro caso se usó fluorescencia.

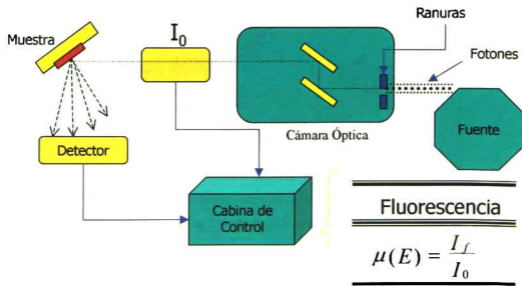


Figura I.8 Esquema que muestra el experimento realizado para obtener los datos de XAS (Espectroscopia de Absorción de Rayos X, por sus siglas en inglés)

La longitud de onda del fotón incidente es variada y de acuerdo a su magnitud podemos utilizarla en diferente tipo de muestras, tal como se describe en la Figura I.7.

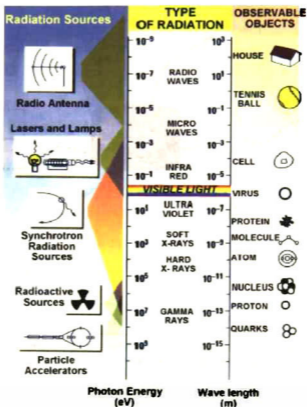


Figura I.9 La fuente de radiación Sincrotrón puede cubrir un ancho amplio de ondas electromagnéticas, que va desde las micro ondas hasta rayos gamma.

I.C.2. Espectro de absorción EXAFS

EXAFS es una técnica usada para determinar de manera muy precisa la estructura local en un material. Sabemos que la luz interactúa con la materia pudiendo observar y medir diferentes fenómenos como los que se describen en la Figura I.10. Algunas formas de determinar el espectro de absorción es observando la fluorescencia, la transmitancia o bien los electrones Auger.

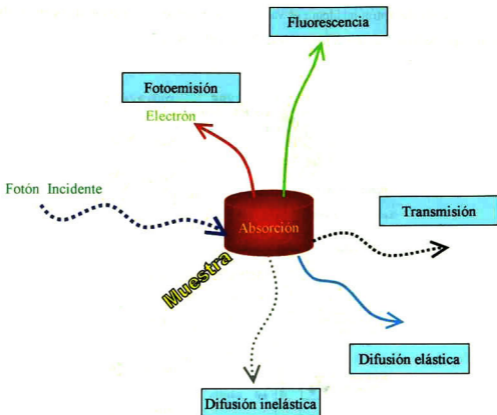


Figura I.10 Esquema que muestra la interacción de la luz con la materia, así como los diferentes fenómenos que pueden ser observados y medidos.

Cada punto obtenido en la curva de absorción de XAS es un promedio muy amplio de lecturas tomadas por el detector y procesadas por el computador, alrededor de 10,000. Por esta razón, los datos obtenidos por este medio son estadísticamente confiables. Las curvas características de XAS se obtienen al graficar la absorción con la energía del haz incidente sobre la muestra.

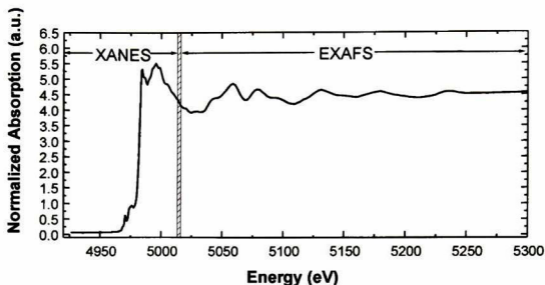


Figura 1.11 Espectro de absorción característico de un estudio XAS.

El espectro de absorción es dividido en dos partes fundamentales para su análisis. La primera es la que está cerca del borde de absorción y se conoce como XANES (X-ray Absorption Near Edge Spectroscopy) y es usada para conocer la distribución electrónica del átomo que absorbe. La parte oscilatoria al final del espectro es conocida como EXAFS y proporciona información muy relevante de la estructura local en la vecindad al átomo que absorbe. Es precisamente el interés del presente proyecto de analizar esta parte del espectro. Para entender y poder realizar el análisis de los datos obtenidos es necesario entender los orígenes de la parte oscilatoria en la curva.

Pensemos que tenemos un átomo con sus niveles discretos bien establecidos tal y como se muestra en la siguiente figura.

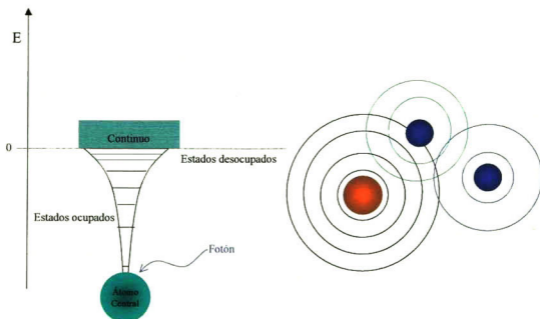


Figura I.12 Absorción y emisión de un fotoelectrón proveniente del nivel 1s que viaja como ondas esféricas

Al llegar un fotón con energía igual o mayor a la energía del electrón en el orbital 1s, el átomo absorbe esa energía y la utiliza para sacar al electrón fuera del átomo dispersándose en forma de ondas esféricas a través del material y retrodispersándose con los átomos vecinos. Las interferencias entre las diferentes trayectorias dan como resultado la parte oscilatoria del espectro de EXAFS. Las trayectorias posibles que sigue el fotoelectrón durante la dispersión depende precisamente de la geometría local en la vecindad del átomo que absorbe.

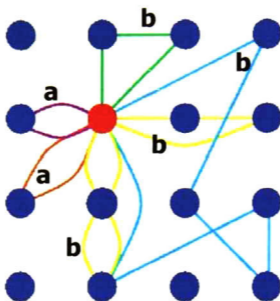


Figura I.13 Trayectorias posibles que puede seguir un fotoelectrón, tenemos dispersión simple (a) y dispersión múltiple (b).

Todas las trayectorias posibles que puede seguir un fotoelectrón durante su dispersión ocurren al mismo tiempo pero con distinta probabilidad. Como podrán observar las oscilaciones presentes en el espectro de EXAFS están ligadas intrínsecamente con la estructura local en la vecindad del átomo que absorbe. Esta técnica es sensible a la estructura local con un alcance aproximadamente de 5Å.

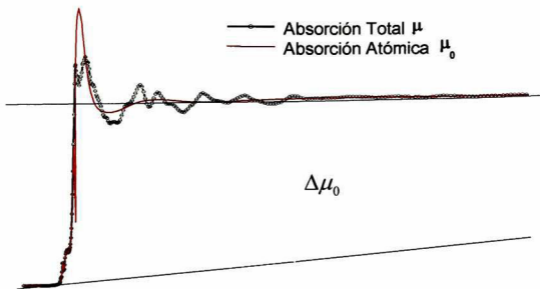


Figura I.14 Espectro de XAS donde se muestra la diferencia en la absorción de luz de átomos aislados y dentro de una estructura. Las oscilaciones son resultado de la presencia de vecinos. Por esta razón, EXAFS provee información sobre la estructura local, desglosado donde se muestra la contribución del átomo que absorbe.

Si graficamos el espectro de XAS y llamamos a μ_0 a la absorción del átomo central y a $\mu(k)$ a la absorción total, se tiene que las oscilaciones de EXAFS normalizadas son obtenidas mediante¹⁵:

$$(I.1) \quad \dots\dots\dots \chi(k) = \frac{\mu - \mu_0}{\Delta\mu_0},$$

donde k es el vector de onda, obtenido en función de la energía (E) y de la energía del borde de absorción (E_0)¹⁵

$$(I.2) \quad \dots\dots\dots k = \sqrt{\frac{2m}{\hbar^2}(E - E_0)},$$

donde m es la masa del electrón y $\hbar = h / 2\pi$, con h la constante de Plank. Cuando se hace la transformación con la ecuación anterior estamos en el espacio k (o espacio recíproco), esto no es una transformación de espacios sino simplemente un cambio de variable. Si desglosamos los gráficos se vería algo similar a lo que se muestra en la Figura I.15.

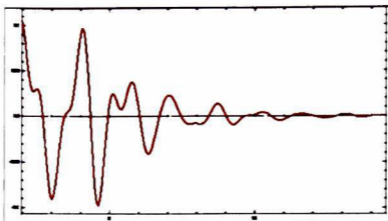


Figura 1.15 Gráfico de EXAFS normalizado, correspondiente a la muestra sin tratamiento térmico.

Las oscilaciones aisladas constituyen el espectro de EXAFS y son resultado de las reflexiones del fotoelectrón con los átomos vecinos. La excitación de este electrón es el proceso que da origen a XAS. Como se discutió en la Sección I.C, el tratamiento teórico se realiza a través de una aproximación semi-clásica. Por un lado el fotón es representado como un campo electromagnético y el electrón es tratado cuánticamente. Al realizar una aproximación dipolar, la ecuación resultante ha sido modificada empíricamente a través de los años y actualmente tenemos una expresión (I.3) **heurística** pues de algún modo la experiencia en análisis de EXAFS ha sido fundamental en su formulación.¹⁶

$$(I.3) \quad \dots\dots\dots \chi(k, \Gamma) = \frac{N_{\Gamma} S_0^2 F_{\Gamma}}{k R_{\Gamma}} e^{-2\sigma_{\Gamma}^2 k^2} e^{-2\frac{R_{\Gamma}}{\lambda}} \sin(2kR_{\Gamma} + \Phi_{\Gamma}),$$

$$(I.4) \quad \chi(k) = \sum_{\Gamma} \chi(k, \Gamma).$$

Donde:

- k Es el vector de onda, obtenido a partir de la energía y calculado con la ecuación (I.2)
- N Es el número de vecinos, estos pueden ser primeros, segundos o terceros vecinos al átomo que absorbe
- S_0 Es un factor de corrección que toma en cuenta la presencia de otros cuerpos, ya que en principio la ecuación fue escrita para un átomo aislado, este factor es un reductor de las amplitudes de dispersión, por lo que su valor debe ser menor que 1.0.
- F Es la amplitud de dispersión de la trayectoria Γ
- R Es la distancia radial de un átomo, al átomo que absorbe

- σ^2 Son los factores de Debye Waller
- λ Es la longitud de onda del fotoelectrón, obtenido también a través de los FEF
- Φ Es el cambio de fase de la onda del fotoelectrón y que esta intrínsecamente ligada a la distancia del recorrido de la trayectoria

I.D. Cálculos teóricos de los parámetros EXAFS

Tradicionalmente, la evaluación de los parámetros que aparecen en la Ecuación (I.3) había sido a través del análisis de materiales de referencia. Afortunadamente, se desarrolló un software (FEFF) basado en Teoría Atómica que calcula estos parámetros en función de la energía del electrón.¹⁶ A través de éste es posible considerar, además, trayectorias de más de dos ramales, las cuales pueden representar una fuerte contribución. El programa FEFF fue usado extensivamente en el análisis.

I.E. Estudios de SIMS

SIMS (Secondary Ion Mass Spectroscopy) es usado para determinar los tipos de elementos químicos o impurezas en un material, todos los elementos incluso el hidrógeno son perceptibles por SIMS. La técnica consiste en colocar la muestra en ultra alto vacío, a la cual se le bombardea la superficie con átomos de oxígeno, cesio u otro elemento químico, de manera que se va haciendo un cráter, el material pulverizado, ionizado y acelerado es recogido por un analizador de masas que detecta la proporción en que se encuentra cada masa atómica.

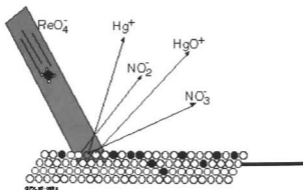


Figura I.16 Obsérvese como un ion de oxígeno de alta energía, incide sobre la superficie y desprende iones de las especies presentes, que son detectadas por un analizador de masas.

I.F. Descripción del proyecto

I.F.1. Elaboración y caracterización de las muestras

Las muestras en estudio fueron elaboradas en Laurence Berkeley Labs (horneo con láser). Y la compañía Customized Wafer Services realizó todo el proceso de impurificación, que consistió en una oblea de silicio con orientación (100), a la cual se le implantaron iones de arsénico (aproximadamente a 500Å de profundidad) equivalente a una dosis de $4 \times 10^{16}/\text{cm}^2$, posteriormente se hizo una licuefacción de los primeros 4000Å con láser para uniformizar el dopaje, así como el daño causado por la implantación. Toda la información obtenida para los procesos de implantación fue utilizada para obtener una homogénea y con las características deseadas, es decir, para lograr obtener un implante hasta los 500Å de profundidad se hicieron pruebas previas para obtener el mejor ángulo de implante y evitar el tuneo, la energía utilizada fue de 10 keV con un ángulo de 65° con respecto a la normal al plano (001).

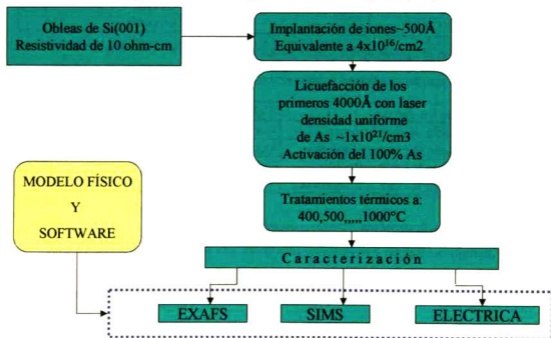


Figura I.17 Diagrama de flujo general para el desarrollo del proyecto

La oblea fue cortada extrayendo muestras que fueron sometidas a tratamientos térmicos por un tiempo suficientemente prolongado hasta lograr la desactivación eléctrica en el semiconductor.

Se obtuvieron datos de caracterización por EXAFS y SIMS obtenidos en el Sincrotrón Brookhaven National Laboratories de Nueva York y en los laboratorios de la Universidad de Stanford respectivamente, los cuales fueron empleados en el análisis presentado en esta tesis.

Para un mejor control del tipo de muestra que se maneje durante este documento, se estableció la siguiente nomenclatura:

Tabla I.1 Nomenclatura de las muestras a estudiar

Tipo de Muestra	Nomenclatura Datos Experimentales	Nomenclatura Curvas de Ajuste
Sin Tratamiento Térmico	STxxx	STxxxfit
Tratamiento a 400 °C	400xxx	400xxxfit
Tratamiento a 500 °C	500xxx	500xxxfit
Tratamiento a 600 °C	600xxx	600xxxfit
Tratamiento a 700 °C	700xxx	700xxxfit
Tratamiento a 800 °C	800xxx	800xxxfit
Tratamiento a 900 °C	900xxx	900xxxfit
Tratamiento a 1000 °C	1000xxx	1000xxxfit

El comodín **xxx**, según el tipo de ajuste, corresponderá a las siguiente simbología:

Tabla I.2 Nomenclatura adicional de acuerdo al espacio de ajuste de los datos experimentales

Espacio de los datos	Siglas correspondiente a:
En el espacio K (curva única)	xxx = AbsKspace
En el espacio R (curva absoluta)	xxx = AbsRspace
En el espacio R (curva Real)	xxx = Re
En el espacio R (curva Imaginaria)	xxx = Im

I.F.2. Descripción general del trabajo a desarrollar

El diagrama de flujo del trabajo a desarrollar para el análisis se muestra en la Figura I.18.

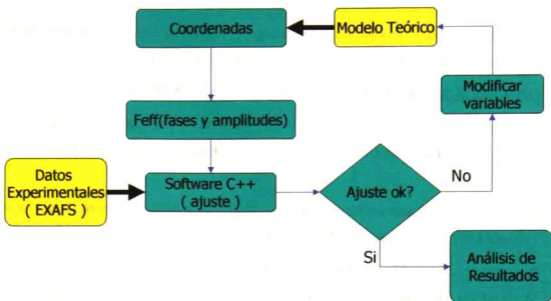


Figura I.18 Diagrama de flujo general del trabajo a desarrollar durante la ejecución del proyecto de tesis

El modelo incorporado al software corresponde al propuesto por Cargil III,⁹ En esta tesis, no se considerarán otros modelos.

Bibliografía

- ¹ J.A. Calderón-Guillén, A. Herrera-Gómez y S. Aguirre-Tostado, "Software para análisis de EXAFS"
- ² M.A. Berding and A. Sher, "Electronic Quasichemical Formalism: Application to arsenic deactivation in silicon. Phys. Rev. B 58, 3853 (1998).
- ³ Cooperative Reseach Between SSRL, Intel, HP, and CIS, Draft Only 11/1/93
- ⁴ TCAD Roadmap: A Supplemement to the National Technology Roadmap for Semiconductors
- ⁵ S. Luning, P.M. Rousseau, P.B Griffin, P.G. Carey, "Kinetics of High Concentration Arsenic deactivation at Moderate to Low Temperature"
- ⁶ John L. Altrip, Alan G.R.Evans, Nigel D.Yong, and John R. Logan, "The Nature of Electrically Inactive implanted Arsenic in Silicon After Rapid Thermal Annealing."

-
- ⁷ Leonard J. Borucki, "Evidence and Modelling of Anomalous Low Concentration Arsenic Inactivation." Motorola Corporatio, Advanced technology Center
- ⁸ D. J. Chadi, P. H. Citrin, C.H. Park, et al. Phys. Rev. Lett. **79**, 4834 (1997).
- ⁹ K.C. Pandey, A.Erbil, G.S. Cargill III, R.F. Boehme, "Annealing of Heavily Arsenic-Doped Silicon: Electrical Deactivation and a New Defect Complex", Phys. Rev. Lett. **61**, 1282 (1988).
- ¹⁰ D.J. Chadi, A. Antonelli y Efthimios Kaxiras, "Vacancy in Silicon Recisted: Structure and Pressure Effect", Phys. Rev. Lett. **81**, 2088 (1998).
- ¹¹ Ravi Subrahmanyam, Marius Orlowski, and Gary Huffman, "Simulation and Experimental Study of the Dynamics of Arsenic Clustering and Precipitation including Ramp-up and Ramp-down Conditions"
- ¹² M. Ramamoorthy and S. T. Pantelides, Phys. Rev. Lett. **76**, 4753 (1996).
- ¹³ D. Novili, A. Carabelas, G. Celotti, and S. Solmi, "Precipitation as the Phenomenon Responsible for the Electrically Inactive Arsenic in Silicon", Phys. Rev. B **49**, 2477 (1994)
- ¹⁴ A. Herrera-Gómez, W.E. Spicer, "Evolution of the Crystallographic position of As impurities in heavily doped Si crystals as their electrical activity changes", Appl. Phys. Lett. **68**, 3090 (1996).
- ¹⁵ Bruce Ravel, EXAFS analysis with FEFF and FEFFIT. 12-Abril-2001
- ¹⁶ J. Mustre de Leon, J. J. Rehr, S. I. Zabinsky, and R. C. Albers. Phys. Rev. B **44**, 4146-4156 (1991).

II. Objetivos

II.A. Objetivo General

Establecer la dependencia de estructura de los agregados de arsénico con la temperatura, identificando y midiendo los diferentes defectos que forman parte en los procesos de desactivación.

II.B. Objetivos específicos

- Crear un modelo físico apropiado para la cinética de desactivación eléctrica del silicio altamente dopado con arsénico, basándose en el modelo de Cargil III
- Elaborar un software que permita realizar ajuste de los datos de EXAFS con base en el modelo físico.
- Analizar los datos existentes de EXAFS con el software desarrollado, y establecer la dependencia de la concentración de los diferentes defectos mostrados en la Figura I.1

III. Resultados Experimentales

III.A. Resultados de EXAFS

Los datos obtenidos del estudio XAS, se muestran en los siguientes gráficos.

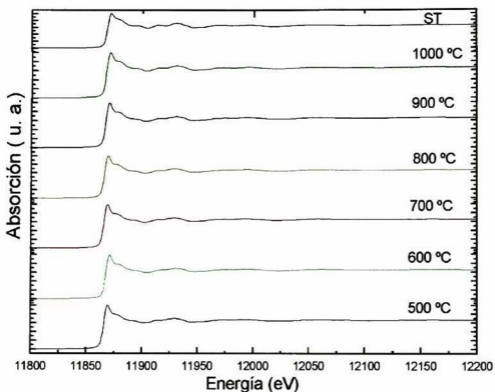


Figura III.1 Datos de XAS obtenidos directamente en el sincrotrón de Brookhaven National Laboratories en New York.

Es difícil apreciar diferencias significativas en la región de EXAFS en estos gráficos. Además, cuando la longitud de onda del electrón es muy pequeña (altas energías), las oscilaciones son muy pequeñas. Sin embargo, es posible observar la parte oscilatoria de manera más detallada si se realizan los siguientes cambios:

- Cambiar de variable E (energía del fotoelectrón) a k (número de onda del fotoelectrón)
- En el eje vertical se grafica el valor de absorción normalizada multiplicada una potencia en k con la finalidad de hacer más importantes las amplitudes de las oscilaciones en la región lejana al borde de absorción. En el caso de la Figura III.2 se utilizó una potencia igual a dos.

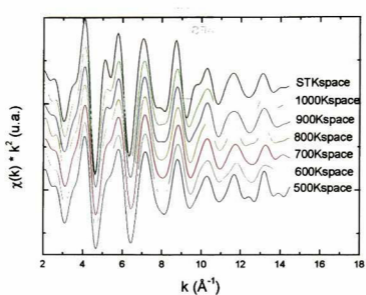


Figura III.2 Datos obtenidos de los estudios de XAS que corresponden a la región extendida oscilatoria (EXAFS), la primera sin tratamiento térmico (S/T) y el resto con tratamientos de 500-1000 °C.

III.B. Tratamiento de los datos experimentales de XAS

Los datos experimentales obtenidos a través del estudio XAS contempla el espectro completo de absorción, el cual se puede dividir en dos partes la correspondiente a XANES y la de EXAFS, esta última es la que por el momento nos interesa obtener.

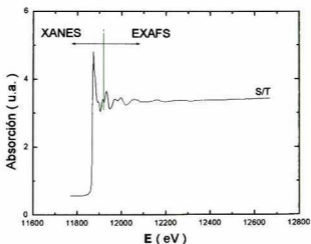


Figura III.3 Espectro completo de un estudio XAS del As K-edge

III.B.2. El espacio k

Aplicando las ecuaciones dadas en el Capítulo I, podemos eliminar la absorción del átomo de arsénico y normalizar obteniendo la región ondulatoria de EXAFS, estos datos pueden llevarse al espacio k (propriadamente no es una transformación de espacio) para su ajuste con un simple cambio de variable, tal como se describe en la Sección I.C.2. Además de obtener estas curvas muchas de las veces es conveniente **pesar** los datos experimentales con la intención de darle más importancia al final del espectro donde las amplitudes de las oscilaciones son muy pequeñas, es común **pesar** los datos en una potencia de k , esto es: $\chi(k) \cdot (k^{\text{kweight}})$, donde **kweight** puede ser 1,2,3.....etc. aunque generalmente se usa las potencias de 1 o 2. A continuación mostramos los datos experimentales en el espacio k realizados con el software desarrollado que se anexa en el Apéndice B.

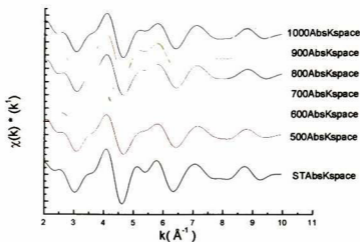


Figura III.4 Datos de EXAFS en el espacio k , pesados con k^{-1}

Obsérvese como se ve un marcado cambio de estructura en el material de la muestra sin tratamiento térmico y la muestra tratada a 500°C.

III.B.3. El espacio R

Mediante una transformada de Fourier podemos pasar del espacio “ k ” al espacio “ R ” (Recordemos que la Transformada de Fourier nos muestra un espacio de frecuencias, que en este caso corresponden por analogía a las distancias R), al hacer esto obtenemos tres clases de curvas, la parte Real e Imaginaria de la transformada de Fourier y la curva absoluta o la magnitud de la transformada que se obtiene a partir de las dos primeras. A continuación mostramos las curvas obtenidas a partir de los datos en el espacio “ k ”

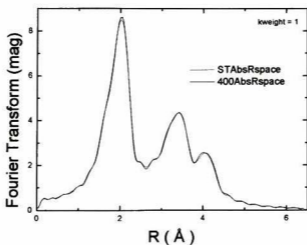


Figura III.5 Datos de EXAFS en el espacio R, muestra Sin Tratamiento térmico y muestra tratada a 400°C

Aunque a simple vista no se aprecia gran diferencia estructural entre ambos espectros (ST y 400) lo que si puede apreciarse es la magnífica definición del primer pico correspondiente a los primeros y segundos vecinos al arsénico y que concuerda muy bien con gráficos reportados por G. S. Cargill III, Chadi y Pandey.

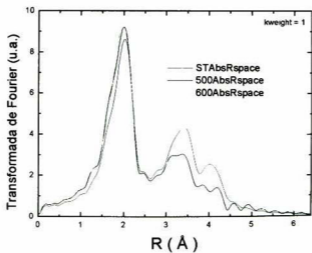


Figura III.6 Datos de EXAFS en el espacio R, imágenes sobrepuestas de las muestras a ST, 500 y 600°C de tratamiento térmico

Estas imágenes superpuestas muestran claramente (mucho mejor que en el espacio K) un cambio de estructura entre las muestras ST y 500°C de tratamiento térmico, sin embargo el pico de los primeros y segundos vecinos siguen bien definidos.

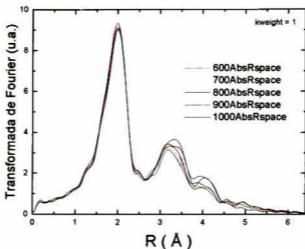


Figura III.7 Datos de EXAFS en el espacio R de muestras tratadas térmicamente de 600-1000°C.

Se superpusieron las imágenes con la intención de mostrar como de manera gradual se van definiendo cada vez mejor los segundos vecinos al arsénico, sin embargo en los primeros vecinos no hay cambios apreciables después de la transición que sufre en los tratamientos térmicos de ST a 500°C.

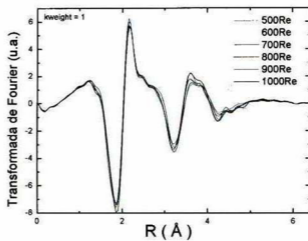
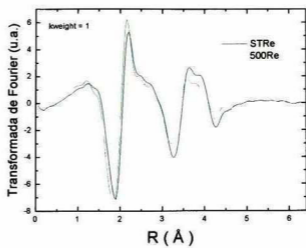


Figura III.8 Datos de EXAFS, parte real de la transformada de Fourier

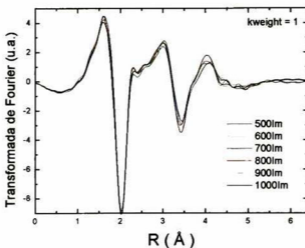
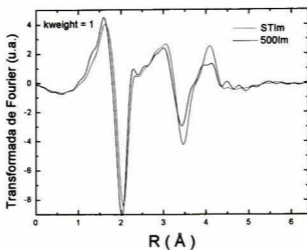


Figura III.9 Datos de EXAFS, parte imaginaria de la transformada de Fourier

Es importante ajustar la parte real e imaginaria de la transformada de Fourier, pues si hiciéramos el ajuste directamente sobre la magnitud, como son valores absolutos puede ser que la parte real o la imaginaria este desfasada y aun así el ajuste sería perfecto, lo cual nos llevaría a errores en la interpretación de los resultados, es por eso que los ajustes de los datos experimentales los haremos con la parte real e imaginaria de la transformada de Fourier, y por supuesto que la suma de estas contribuciones deben ajustar a la magnitud de la transformada en el espacio de Fourier

III.C. Estudios de SIMS

Estos estudios se realizan para determinar el tipo y cantidad de un tipo especial de átomos presentes en un material desde la superficie hasta la profundidad deseada. En la Figura III.10 se muestran los resultados de estos estudios para el caso de los átomos de arsénico distribuidos en el sustrato. Aquí se puede observar como la magnitud de la implantación significativa la podemos encontrar hasta los 4000Å de profundidad, con valores que oscilan entre 1.00E+021 hasta 7.00E+020.

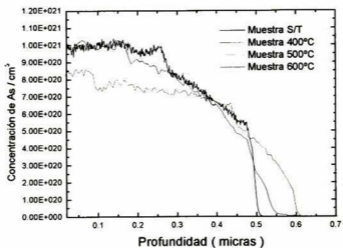
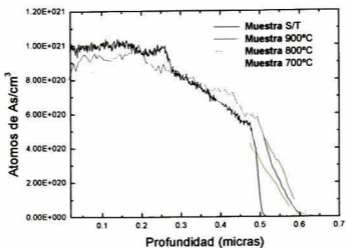
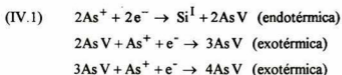


Figura III.10 Figura que muestra los datos obtenidos por SIMS y como se observa a 4000Å de la superficie tenemos una cantidad significativa de dopantes por arriba de $7.00E+020$ As/cm³

IV. Descripción del Modelo Físico

IV.A. Proceso de desactivación eléctrica

Al incorporar impurezas de arsénico (As^+) sobre silicio se crea un semiconductor tipo n , ya que en principio el arsénico activo contribuye con un electrón a la banda de conducción, modelo 4 de la Figura IV.1. Sin embargo cuando se dan tratamientos térmicos a las muestras, los átomos de arsénico empiezan a moverse y a fijarse alrededor de vacancias, y con esto se vuelven inactivos, dejando de contribuir con su electrón de valencia, este es el caso de los modelos 1,2 y 3 de la Figura IV.1



donde Si^{I} representa un átomo de silicio intersticial. La movilidad del silicio intersticial en cristales de silicio es sumamente alta, por lo que estos migran rápidamente a la superficie o al bulto. Esto provoca que, a bajas temperaturas, cuando la disponibilidad de silicio intersticial es poca, la primera reacción sea en una sola dirección. Las siguientes dos reacciones son en una dirección porque son exotérmicas.

IV.B. Modelos Teóricos

Para encontrar los parámetros en la ecuación de EXAFS es necesario tener una idea previa de la estructura local en la vecindad del átomo que emite. A través de los datos EXAFS es posible encontrar los detalles de la estructura del modelo propuesto. Es por ello que se propone una serie de modelos dinámicos, esto es, las configuraciones estructurales que se proponen cuentan con parámetros que determinan la geometría local, pudiendo manejar una infinidad de deformaciones. En principio los modelos propuestos están basados en las investigaciones reportadas hasta la fecha. La forma de denotarlos y llamarlos está mostrada en la Figura IV.1.

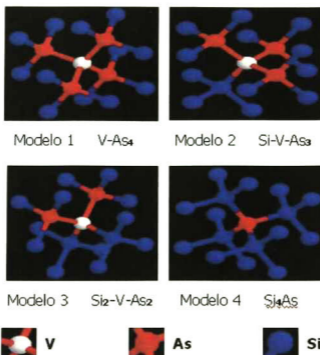


Figura IV.1 Modelos físicos propuestos fundamentados por Pandey et al.

De acuerdo a las investigaciones reportadas por Cargill,¹ Berding,² y otros, los modelos energéticamente favorables presentan una vacancia. Es de esperarse que existirán deformaciones locales de los átomos alrededor de la vacancia. En principio se suponen que estas deformaciones producirán desplazamientos en los primeros y segundos vecinos a la vacancia, gobernados por los parámetros α y β respectivamente. Con cada par de valores de α y β es posible representar infinidad de posiciones en los átomos y con esa posición encontrar las fase y amplitudes de nuestra ecuación de EXAFS para con ellos ajustar los datos experimentales.

IV.C. Formulación Matemática

IV.C.1. Sistema de Referencia

Una de las hipótesis de este proyecto es que existen distorsiones locales en la red debido a dos causas, la primera es que el arsénico entra de manera sustitucional en la red del silicio³ siendo el arsénico de mayor tamaño tenderá a expandir la red en su entorno, la segunda es la presencia de vacancias. Se a supuesto que existen desplazamientos axiales a la vacancia (o arsénico) de los primeros y segundos vecinos a la misma, esto es:

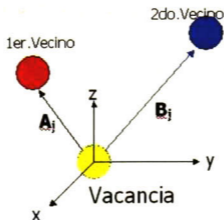


Figura IV.2 Localización de los primero y segundos vecinos a una Vacancia (o arsénico) respecto al sistema X-Y-Z.

$$(IV.2) \quad \dots\dots\dots \vec{A} = (1 + \alpha) \vec{a}$$

$$(IV.3) \quad \dots\dots\dots \vec{B} = (1 + \beta) \vec{b}$$

Donde α y β , son parámetros adimensionales, un valor negativo de ellos indica que la red se comprime, \vec{a} y \vec{b} son los vectores que localizan a los primeros y segundos vecinos cuando la deformación es nula ($\alpha = \beta = 0$).

En los estudios de EXAFS el átomo de arsénico es el que absorbe por lo que es necesario realizar una traslación de ejes que va de la vacancia a uno de sus primeros vecinos arsénicos, a los cuales llamaremos X'-Y'-Z' Respecto a este sistema de referencia se localizarán la posición de los átomos considerando los desplazamientos de los átomos vecinos a la vacancia, en el caso especial del modelo 4 (Si₄As) no es necesario esta traslación de ejes.

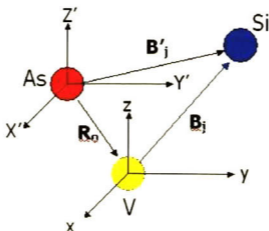


Figura IV.3 Traslación de ejes de referencia de la vacancia al átomo de arsénico que absorbe.

La localización de los átomos respecto al sistema X'-Y'-Z' se hace mediante la siguiente transformación lineal.

$$(IV.4) \quad \dots\dots\dots \vec{B}^j = \vec{R}_0 + \vec{B}_j \dots\dots\dots j = 1, 2, \dots, n$$

con

$$(IV.5) \quad \dots\dots\dots \vec{R}_0 = -\vec{A}_i$$

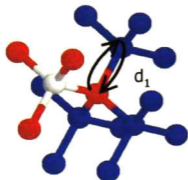
La localización de los átomos se realizó en función de los parámetros α y β y se usó el software **Feff.exe** (Universidad de Washington⁴) para determinar las fases y amplitudes de cada una de las trayectorias que sigue un fotoelectrón durante la dispersión y que depende por supuesto de cada uno de los modelos propuestos. Estos resultados están disponibles en el Apéndice A

IV.C.2. Trayectorias Modelo 1 (V-As₄)

En este modelo el parámetro α está referenciado a los arsénicos que son los primeros vecinos a la vacancia y β esta referenciado a los segundos vecinos a la vacancia o sea a silicios, un valor negativo de estos parámetros indicará que los átomos referenciados se acercan a la vacancia. Para un mejor entendimiento de las figuras siguientes emplearemos la siguiente nomenclatura:

$d_i^j \dots\dots i = 1, 2, 3, 4$ y nos indica el tramo de recorrido de la trayectoria j (A, B o C), el superíndice j se emplea cuando se tienen varias trayectorias con diferentes ecuaciones para un mismo feff, en caso contrario se omite el superíndice.

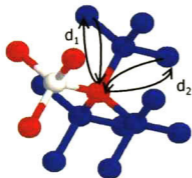
$$d_T^j = \frac{\sum_i d_i^j}{2} \quad \text{que nos da distancia total promedio del recorrido.}$$



$$d_1 = \frac{a}{4} \sqrt{3(1+\alpha)^2 - 8(1+\alpha)(1+\beta) + 8(1+\beta)^2} \text{ Deg} \quad (3)$$

$$d_T = \frac{2d_1}{2}$$

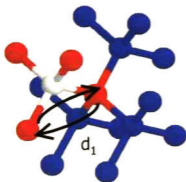
Figura IV.4 Trayectoria 1. Nombre: Feff0001. Contribución: 100%. Número de ramales: 2.



$$d_1 = \frac{a}{4} \sqrt{3(1+\alpha)^2 - 6(1+\alpha) + 11} \text{ Deg} \quad (6)$$

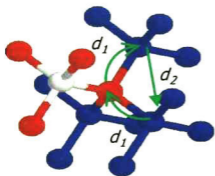
$$d_2 = \frac{a}{4} \sqrt{3(1+\alpha)^2 - 14(1+\alpha) + 19} \text{ Deg} \quad (3)$$

Figura IV.5 Trayectoria 2. Nombre Feff0002. Contribución 71.91%. Número de ramales: 2.



$$d_1 = \frac{(1+\alpha)\alpha}{4} \sqrt{8} \text{ Deg} \quad (3)$$

Figura IV.6 Trayectoria 3. Nombre Feff0003. Contribución 29.78%. Número de ramales: 2.



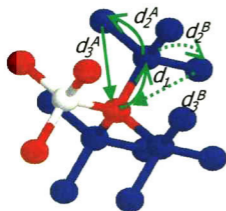
$$d_1 = \frac{a}{4} \sqrt{3(1+\alpha)^2 - 8(1+\alpha)(1+\beta) + 8(1+\beta)^2}$$

$$d_2 = \frac{a}{4} (1+\beta) \sqrt{8}$$

$$d_T = 2d_1 + d_2$$

Deg
(6)

Figura IV.7 Trayectoria 4. Nombre Feff0004. Contribución 3.875%. Número de ramales: 3. d_T es la distancia total promedio del recorrido de la trayectoria



$$d_1 = \frac{a}{4} \sqrt{3(1+\alpha)^2 - 8(1+\alpha)(1+\beta) + 8(1+\beta)^2}$$

$$d_2^A = \frac{a}{4} \sqrt{8(1+\beta)^2 - 16(1+\beta) + 11}$$

$$d_2^B = \frac{a}{4} \sqrt{8(1+\beta)^2 - 24(1+\beta) + 19}$$

$$d_3^A = \frac{a}{4} \sqrt{3(1+\alpha)^2 - 6(1+\alpha) + 11}$$

$$d_3^B = \frac{a}{4} \sqrt{3(1+\alpha)^2 - 14(1+\alpha) + 19}$$

$$d_T^A = d_1 + d_2^A + d_3^A$$

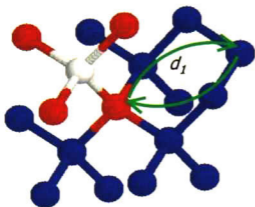
$$d_T^B = d_1 + d_2^B + d_3^B$$

Deg

(12)

(6)

Figura IV.8 Trayectoria 5. Nombre Feff0005. Contribución 27.9%. Número de ramales: 3



$$d_1 = \frac{a}{4} \sqrt{3(1+\alpha)^2 + 8(1+\beta)^2}$$

Deg
(6)

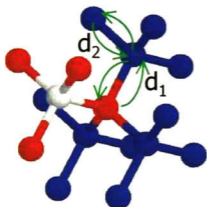
$$d_2 = \frac{a}{4} \sqrt{3(1+\alpha)^2 - 8(1+\alpha) + 16}$$

(3)

$$d_3 = \frac{a}{4} \sqrt{8(1+\alpha)^2 - 16(1+\alpha) + 24}$$

(3)

Figura IV.9 Trayectoria 6. Nombre Feff0006. Contribución 58.41%. Número de ramales: 2



$$d_1 = \frac{a}{4} \sqrt{3(1+\alpha)^2 - 8(1+\alpha)(1+\beta) + 8(1+\beta)^2}$$

$$d_2^A = \frac{a}{4} \sqrt{8(1+\beta)^2 - 16(1+\beta) + 11}$$

$$d_2^B = \frac{a}{4} \sqrt{8(1+\beta)^2 - 24(1+\beta) + 19}$$

Deg

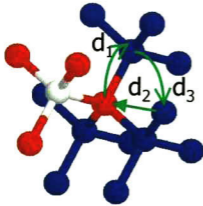
$$d_r = 2 [d_1 + d_2^A]$$

(6)

$$d_r = 2 [d_1 + d_2^B]$$

(3)

Figura IV.10 Trayectoria 8. Nombre Feff0008. Contribución 4.48%. Número de ramales: 4



$$d_1 = \frac{\alpha}{4} \sqrt{3(1+\alpha)^2 - 8(1+\alpha)(1+\beta) + 8(1+\beta)^2}$$

$$d_2^A = \frac{\alpha}{4} \sqrt{3(1+\alpha)^2 - 6(1+\alpha) + 11}$$

$$d_2^B = \frac{\alpha}{4} \sqrt{3(1+\alpha)^2 - 14(1+\alpha) + 19}$$

$$d_3^A = \frac{\alpha}{4} \sqrt{8(1+\beta)^2 - 8(1+\beta) + 11}$$

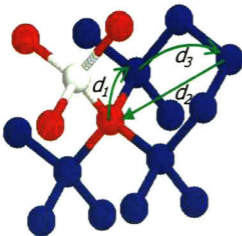
$$d_3^B = \frac{\alpha}{4} \sqrt{8(1+\beta)^2 - 16(1+\beta) + 19}$$

Deg

$$d_r^A = d_1 + d_2^A + d_3^A \quad (6)$$

$$d_r^B = d_1 + d_2^B + d_3^B \quad (3)$$

Figura IV.11 Trayectoria 9. Nombre Feff0009. Contribución 3.80%. Número de ramales: 3



$$d_1 = \frac{\alpha}{4} \sqrt{3(1+\alpha)^2 - 8(1+\alpha)(1+\beta) + 8(1+\beta)^2}$$

$$d_2^A = \frac{\alpha}{4} \sqrt{3(1+\alpha)^2 + 8(1+\beta)^2}$$

$$d_2^B = \frac{\alpha}{4} \sqrt{3(1+\alpha)^2 - 8(1+\alpha) + 16}$$

$$d_2^C = \frac{\alpha}{4} \sqrt{3(1+\alpha)^2 - 16(1+\alpha) + 24}$$

$$d_3^A = \frac{\alpha}{4} \sqrt{8(1+\beta)^2 - 24(1+\beta) + 24}$$

$$d_3^B = \frac{\alpha}{4} \sqrt{8(1+\beta)^2 - 16(1+\beta) + 16}$$

$$d_3^C = \frac{\alpha}{4} (1+\beta) \sqrt{8}$$

Deg

$$d_r^A = d_1 + d_2^A + d_3^A \quad (12)$$

$$d_r^B = d_1 + d_2^B + d_3^B \quad (12)$$

$$d_r^C = d_1 + d_2^C + d_3^C \quad (12)$$

Figura IV.12 Trayectoria 10. Nombre Feff0010. Contribución 12.20%. Número de ramales: 3.

Aunque se generaron las ecuaciones para los primeros 20 feff, únicamente se muestran los primeros 10 ya que en primer lugar son los que tienen mayor probabilidad de ocurrir y segundo porque el radio efectivo para el último feff es de 5.34Å y se encuentra en el límite de la resolución de EXAFS.

IV.C.3. Trayectorias Modelo 2 (As₃-V-Si)

En este modelo el parámetro α_1 está referenciado a los arsénicos que son los primeros vecinos a la vacancia y α_2 está direccionado al silicio que está como primer vecino a la vacancia, β está referenciado a los segundos vecinos a la vacancia o sea a silicios, un valor negativo de estos parámetros indicará que los átomos referenciados se acercan a la vacancia.

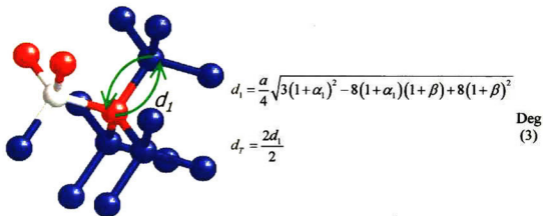


Figura IV.13 Trayectoria 1. Nombre Feff0001. Contribución 100%. Número de ramales: 2

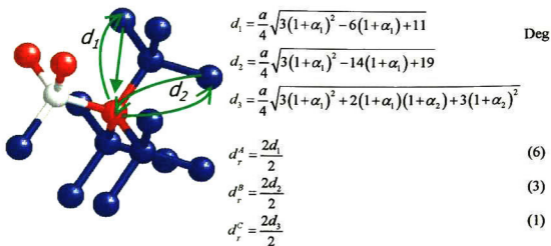


Figura IV.14 Trayectoria 2. Nombre Feff0002. Contribución 79.91%. Número de ramales: 2

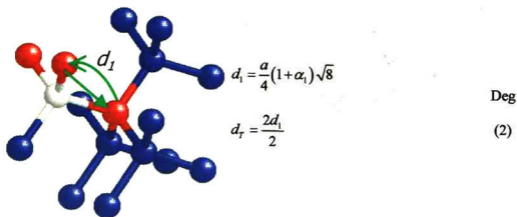
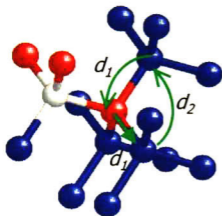


Figura IV.15 Trayectoria 3. Nombre Feff0003. Contribución 19.856%. Número de ramales: 2



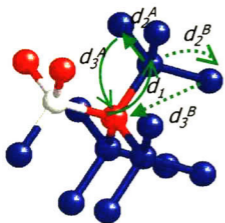
$$d_1 = \frac{a}{4} \sqrt{3(1+\alpha_1)^2 - 8(1+\alpha_1)(1+\beta) + 8(1+\beta)^2}$$

$$d_2 = \frac{a}{4} (1+\beta) \sqrt{8}$$

$$d_7 = 2d_1 + d_2$$

Deg
(6)

Figura IV.16 Trayectoria 4. Nombre Feff004. Contribución 3.872%. Número de ramales: 3



$$d_1 = \frac{a}{4} \sqrt{3(1+\alpha_1)^2 - 8(1+\alpha_1)(1+\beta) + 8(1+\beta)^2}$$

$$d_2^A = \frac{a}{4} \sqrt{8(1+\beta)^2 - 16(1+\beta) + 11}$$

$$d_2^B = \frac{a}{4} \sqrt{8(1+\beta)^2 - 24(1+\beta) + 19}$$

$$d_3^A = \frac{a}{4} \sqrt{3(1+\alpha_1)^2 - 6(1+\alpha_1) + 11}$$

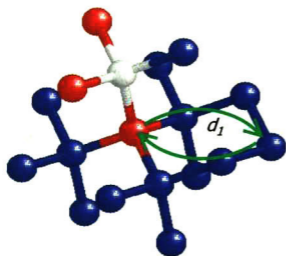
$$d_3^B = \frac{a}{4} \sqrt{3(1+\alpha_1)^2 - 14(1+\alpha_1) + 19}$$

$$d_7^A = d_1 + d_2^A + d_3^A \tag{12}$$

$$d_7^B = d_1 + d_2^B + d_3^B \tag{6}$$

Deg

Figura IV.17 Trayectoria 5. Nombre: Feff005. Contribución: 27.877%. Número de ramales: 3.



$$d_1 = \frac{a}{4} \sqrt{3(1+\alpha_1)^2 + 8(1+\beta)^2}$$

Deg

$$d_2 = \frac{a}{4} \sqrt{3(1+\alpha_1)^2 - 8(1+\alpha_1) + 16}$$

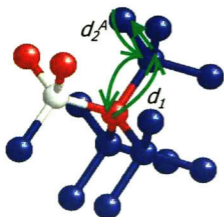
$$d_3 = \frac{a}{4} \sqrt{3(1+\alpha_1)^2 - 16(1+\alpha_1) + 24}$$

$$d_r^A = \frac{2d_1}{2} \quad (6)$$

$$d_r^B = \frac{2d_2}{2} \quad (3)$$

$$d_r^C = \frac{2d_3}{2} \quad (3)$$

Figura IV.18 Trayectoria 6. Nombre: Feff006. Contribución: 58.41%. Número de ramales: 2.



$$d_1 = \frac{a}{4} \sqrt{3(1+\alpha_1)^2 - 8(1+\alpha_1)(1+\beta) + 8(1+\beta)^2}$$

$$d_2^A = \frac{a}{4} \sqrt{8(1+\beta)^2 - 16(1+\beta) + 11}$$

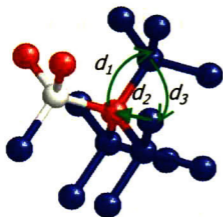
$$d_2^B = \frac{a}{4} \sqrt{8(1+\beta)^2 - 24(1+\beta) + 19}$$

Deg

$$d_r^A = 2[d_1 + d_2^A] \quad (6)$$

$$d_r^B = 2[d_1 + d_2^B] \quad (3)$$

Figura IV.19 Trayectoria 8. Nombre: Feff008. Contribución: 4.472%. Número de ramales: 4.



$$d_1 = \frac{a}{4} \sqrt{3(1+\alpha_1)^2 - 8(1+\alpha_1)(1+\beta) + 8(1+\beta)^2}$$

$$d_2^A = \frac{a}{4} \sqrt{3(1+\alpha_1)^2 - 6(1+\alpha_1) + 11}$$

$$d_2^B = \frac{a}{4} \sqrt{3(1+\alpha_1)^2 - 14(1+\alpha_1) + 19}$$

$$d_2^C = \frac{a}{4} \sqrt{3(1+\alpha_1)^2 + 2(1+\alpha_1)(1+\alpha_2) + 3(1+\alpha_2)^2}$$

$$d_3^A = \frac{a}{4} \sqrt{8(1+\beta)^2 - 8(1+\beta) + 11}$$

$$d_3^B = \frac{a}{4} \sqrt{8(1+\beta)^2 - 16(1+\beta) + 19}$$

$$d_3^C = \frac{a}{4} \sqrt{3(1+\alpha_2)^2 + 8(1+\beta)^2}$$

$$d_T^A = d_1 + d_2^A + d_3^A$$

$$d_T^B = d_1 + d_2^B + d_3^B$$

$$d_T^C = d_1 + d_2^C + d_3^C$$

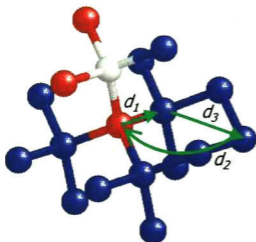
Deg

(12)

(12)

(4)

Figura IV.20 Trayectoria 9. Nombre: Feff009. Contribución: 4.428%. Número de ramales: 3.



$$d_1 = \frac{a}{4} \sqrt{3(1+\alpha_1)^2 - 8(1+\alpha_1)(1+\beta) + 8(1+\beta)^2}$$

$$d_2^A = \frac{a}{4} \sqrt{3(1+\alpha_1)^2 + 8(1+\beta)^2}$$

$$d_2^B = \frac{a}{4} \sqrt{3(1+\alpha_1)^2 - 8(1+\alpha_1) + 16}$$

$$d_2^C = \frac{a}{4} \sqrt{3(1+\alpha_1)^2 - 16(1+\alpha_1) + 24}$$

$$d_3^A = \frac{a}{4} \sqrt{8(1+\beta)^2 - 16(1+\beta) + 16}$$

$$d_3^B = \frac{a}{4} \sqrt{8(1+\beta)^2 - 24(1+\beta) + 24}$$

$$d_3^C = \frac{a}{4} (1+\beta) \sqrt{8}$$

Deg

$$d_7^A = d_1 + d_2^A + d_3^A \quad (12)$$

$$d_7^B = d_1 + d_2^B + d_3^B \quad (12)$$

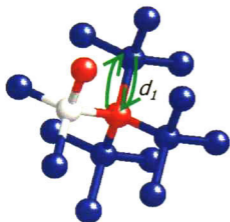
$$d_7^C = d_1 + d_2^C + d_3^C \quad (12)$$

Figura IV.21 Trayectoria 10. Nombre: Feff010. Contribución: 12.19%. Número de ramales: 3. En la figura se colocó únicamente una trayectoria genérica para los tres tipos existentes.

Aunque se generaron las ecuaciones para los primeros 20 feff, únicamente se muestran los primeros 10 ya que en primer lugar son los que tienen mayor probabilidad de ocurrir y segundo porque el radio efectivo para el último feff es de 5.34Å y se encuentra en el límite de la resolución de EXAFS.

IV.C.4. Trayectorias Modelo 3 (As_2-V-Si_2)

En este modelo el parámetro α_1 está referenciado a los arsénicos que son los primeros vecinos a la vacancia y α_2 está direccionado al silicio que está como primer vecino a la vacancia, β está referenciado a los segundos vecinos a la vacancia o sea a silicios, un valor negativo de estos parámetros indicará que los átomos referenciados se acercan a la vacancia.

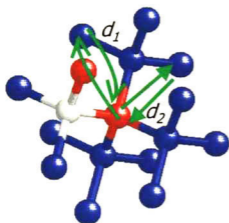


$$d_1 = \frac{a}{4} \sqrt{3(1+\alpha_1)^2 - 8(1+\alpha_1)(1+\beta) + 8(1+\beta)^2}$$

$$d_r = \frac{2d_1}{2}$$

Deg
(3)

Figura IV.22 Trayectoria 1. Nombre Feff0001. Contribución 100%. Número de ramas: 2



$$d_1 = \frac{a}{4} \sqrt{3(1+\alpha_1)^2 - 6(1+\alpha_1) + 11}$$

$$d_2 = \frac{a}{4} \sqrt{3(1+\alpha_1)^2 - 14(1+\alpha_1) + 19}$$

$$d_3 = \frac{a}{4} \sqrt{3(1+\alpha_1)^2 + 2(1+\alpha_1)(1+\alpha_2) + 3(1+\alpha_2)^2}$$

$$d_r^A = \frac{2d_1}{2}$$

Deg

(6)

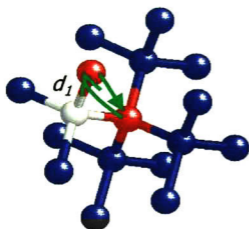
$$d_r^B = \frac{2d_2}{2}$$

(3)

$$d_r^C = \frac{2d_3}{2}$$

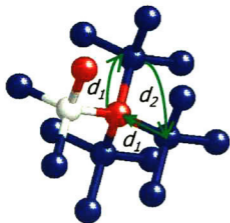
(2)

Figura IV.23 Trayectoria 2. Nombre Feff0002. Contribución 87.905%. Número de ramas: 2



$$d_1 = \frac{a}{4}(1+\alpha_1)\sqrt{8} \quad \text{Deg} \quad (1)$$

Figura IV.24 Trayectoria 3. Nombre Feff0003. Contribución 9.929%. Número de ramales: 2



$$d_1 = \frac{a}{4}\sqrt{3(1+\alpha_1)^2 - 8(1+\alpha_1)(1+\beta) + 8(1+\beta)^2}$$

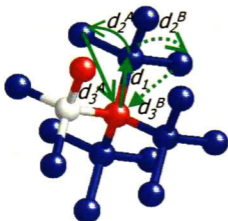
$$d_2 = \frac{a}{4}(1+\beta)\sqrt{8}$$

$$d_7 = 2d_1 + d_2$$

Deg

(6)

Figura IV.25 Trayectoria 4. Nombre Feff0004. Contribución 3.869%. Número de ramales: 3



$$d_1 = \frac{a}{4} \sqrt{3(1+\alpha_1)^2 - 8(1+\alpha_1)(1+\beta) + 8(1+\beta)^2}$$

$$d_2^A = \frac{a}{4} \sqrt{8(1+\beta)^2 - 16(1+\beta) + 11}$$

$$d_2^B = \frac{a}{4} \sqrt{8(1+\beta)^2 - 24(1+\beta) + 19}$$

$$d_3^A = \frac{a}{4} \sqrt{3(1+\alpha_1)^2 - 6(1+\alpha_1) + 11}$$

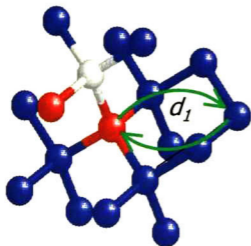
$$d_3^B = \frac{a}{4} \sqrt{3(1+\alpha_1)^2 - 14(1+\alpha_1) + 19}$$

Deg

$$d_7^A = d_1 + d_2^A + d_3^A \quad (12)$$

$$d_7^B = d_1 + d_2^B + d_3^B \quad (6)$$

Figura IV.26 Trayectoria 5. Nombre: Feff005. Contribución: 27.848%. Número de ramales: 3.



$$d_1 = \frac{a}{4} \sqrt{3(1+\alpha_1)^2 + 8(1+\beta)^2}$$

$$d_2 = \frac{a}{4} \sqrt{3(1+\alpha_1)^2 - 8(1+\alpha_1) + 16}$$

$$d_3 = \frac{a}{4} \sqrt{3(1+\alpha_1)^2 - 16(1+\alpha_1) + 24}$$

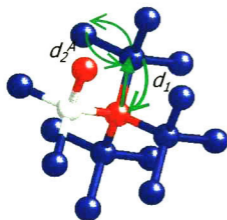
Deg

$$d_r^A = \frac{2d_1}{2} \quad (6)$$

$$d_r^B = \frac{2d_2}{2} \quad (3)$$

$$d_r^C = \frac{2d_3}{2} \quad (3)$$

Figura IV.27 Trayectoria 6. Nombre: Feff006. Contribución: 58.41%. Número de ramales: 2.



$$d_1 = \frac{a}{4} \sqrt{3(1+\alpha_1)^2 - 8(1+\alpha_1)(1+\beta) + 8(1+\beta)^2}$$

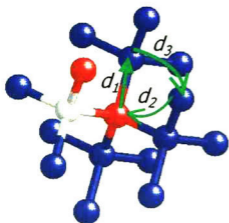
$$d_2^A = \frac{a}{4} \sqrt{8(1+\beta)^2 - 16(1+\beta) + 11}$$

$$d_2^B = \frac{a}{4} \sqrt{8(1+\beta)^2 - 24(1+\beta) + 19}$$

$$d_7^A = 2[d_1 + d_2^A] \quad \text{Deg} \quad (6)$$

$$d_7^B = 2[d_1 + d_2^B] \quad (3)$$

Figura IV.28 Trayectoria 8. Nombre: Feff008. Contribución: 4.472%. Número de ramales: 4.



$$d_1 = \frac{a}{4} \sqrt{3(1+\alpha_1)^2 - 8(1+\alpha_1)(1+\beta) + 8(1+\beta)^2}$$

$$d_2^A = \frac{a}{4} \sqrt{3(1+\alpha_1)^2 - 6(1+\alpha_1) + 11}$$

$$d_2^B = \frac{a}{4} \sqrt{3(1+\alpha_1)^2 - 14(1+\alpha_1) + 19}$$

$$d_2^C = \frac{a}{4} \sqrt{3(1+\alpha_1)^2 + 2(1+\alpha_1)(1+\alpha_2) + 3(1+\alpha_2)^2}$$

$$d_3^A = \frac{a}{4} \sqrt{8(1+\beta)^2 - 8(1+\beta) + 11}$$

$$d_3^B = \frac{a}{4} \sqrt{8(1+\beta)^2 - 16(1+\beta) + 19}$$

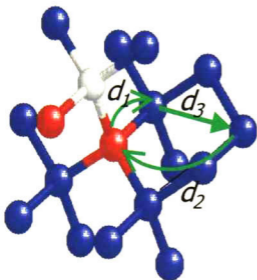
$$d_3^C = \frac{a}{4} \sqrt{3(1+\alpha_2)^2 + 8(1+\beta)^2}$$

$$d_7^A = d_1 + d_2^A + d_3^A \quad \text{Deg} \quad (12)$$

$$d_7^B = d_1 + d_2^B + d_3^B \quad (12)$$

$$d_7^C = d_1 + d_2^C + d_3^C \quad (8)$$

Figura IV.29 Trayectoria 9. Nombre: Feff009. Contribución: 5.058%. Número de ramales: 3.



$$d_1 = \frac{a}{4} \sqrt{3(1+\alpha_1)^2 - 8(1+\alpha_1)(1+\beta) + 8(1+\beta)^2}$$

$$d_2^A = \frac{a}{4} \sqrt{3(1+\alpha_1)^2 + 8(1+\beta)^2}$$

$$d_2^B = \frac{a}{4} \sqrt{3(1+\alpha_1)^2 - 8(1+\alpha_1) + 16}$$

$$d_2^C = \frac{a}{4} \sqrt{3(1+\alpha_1)^2 - 16(1+\alpha_1) + 24}$$

$$d_3^A = \frac{a}{4} \sqrt{8(1+\beta)^2 - 16(1+\beta) + 16}$$

$$d_3^B = \frac{a}{4} \sqrt{8(1+\beta)^2 - 24(1+\beta) + 24}$$

$$d_3^C = \frac{a}{4} (1+\beta) \sqrt{8}$$

Deg

$$d_7^A = d_1 + d_2^A + d_3^A \quad (12)$$

$$d_7^B = d_1 + d_2^B + d_3^B \quad (12)$$

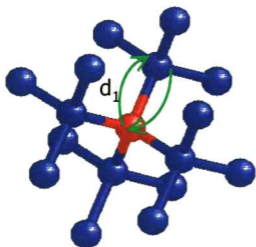
$$d_7^C = d_1 + d_2^C + d_3^C \quad (12)$$

Figura IV.30 Trayectoria 10. Nombre: Feff010. Contribución: 12.18%. Número de ramales: 3.

Aunque se generaron las ecuaciones para los primeros 20 feff, únicamente se muestran los primeros 10 ya que en primer lugar son los que tienen mayor probabilidad de ocurrir y segundo porque el radio efectivo para el último feff es de 5.34Å y se encuentra en el límite de la resolución de EXAFS.

IV.C.5. Trayectorias Modelo 4 (Si_4-As_1)

En este modelo el parámetro α está referenciado a los silicios que son los primeros vecinos al arsénico y β está referenciado a los segundos vecinos silicio del arsénico, un valor negativo de estos parámetros indicará que los átomos referenciados se acercan a la vacancia.



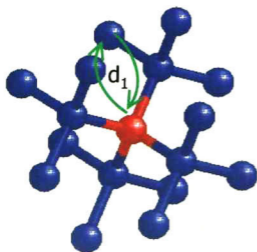
$$d_1 = \frac{a}{4}(1 + \alpha_2)\sqrt{3}$$

Deg

$$d_r = \frac{2d_1}{2}$$

(4)

Figura IV.31 Trayectoria 1. Nombre Feff0001. Contribución 100%. Número de ramales: 2



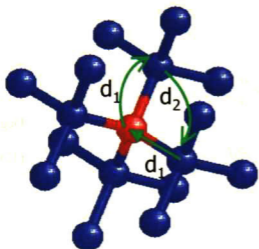
$$d_1 = \frac{a}{4}(1 + \beta)\sqrt{8}$$

Deg

$$d_r = \frac{2d_1}{2}$$

(12)

Figura IV.32 Trayectoria 2. Nombre Feff0002. Contribución 71.9%. Número de ramales: 2



$$d_1 = \frac{a}{4}(1+\alpha_2)\sqrt{3}$$

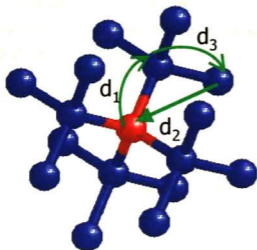
$$d_2 = \frac{a}{4}(1+\alpha_2)\sqrt{8}$$

$$d_T = 2d_1 + d_2$$

Deg

(12)

Figura IV.33 Trayectoria 3. Nombre Feff003. Contribución 5.8%. Número de ramales: 3



$$d_1 = \frac{a}{4}(1+\alpha_2)\sqrt{3}$$

$$d_2 = \frac{a}{4}(1+\beta)\sqrt{8}$$

$$d_3 = \frac{a}{4}\sqrt{3(1+\alpha_2)^2 - 8(1+\alpha_2)(1+\beta) + 8(1+\beta)^2}$$

Deg

$$d_T = d_1 + d_2 + d_3$$

(24)

Figura IV.34 Trayectoria 4. Nombre: Feff004. Contribución: 27.8%. Número de ramales: 3.

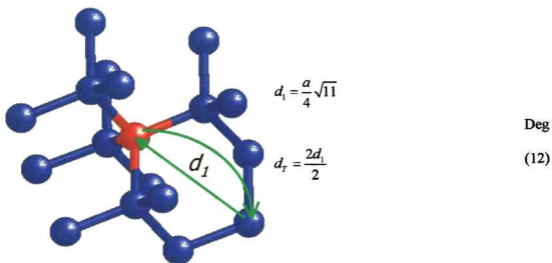


Figura IV.35 Trayectoria 5. Nombre: Feff005. Contribución: 43.81%. Número de ramales: 2.

Aunque se generaron las ecuaciones para los primeros 20 feff, únicamente se muestran los primeros 10 ya que en primer lugar son los que tienen mayor probabilidad de ocurrir y segundo porque el radio efectivo para el último feff es de 5.34Å y se encuentra en el límite de la resolución de EXAFS.

Bibliografía

- ¹ K.C. Pandey, A.Erbil, G.S. Cargill III, R.F. Boehme, "Annealing of Heavily Arsenic-Doped Silicon: Electrical Deactivation and a New Defect Complex" *Phys. Rev. Lett.* **61**, 1282 (1988).
- ² M.A. Berding and A. Sher, "Electronic Quasichemical Formalism: Application to arsenic deactivation in silicon. *Phys. Rev. B* **58**, 3853 (1998).
- ³ A. Herrera-Gómez, W.E. Spicer, "Evolution of the Crystallographic position of As impurities in heavily doped Si crystals as their electrical activity changes" *Appl. Phys. Lett.* **68**, 3090 (1996).
- ⁴ J. Mustre de Leon, J. J. Rehr, S. I. Zabinsky, and R. C. Albers. *Phys. Rev. B* **44**, 4146-4156 (1991).

V. Descripción del Software

Una idea inicial que se tuvo para desarrollar el software es que fuera amigable con el usuario, que tuviera un ambiente gráfico disponible que mostrara los ajustes en tiempo de ejecución, que pudiera a través del ambiente creado la posibilidad de tratar los datos gráficamente, así como analizar sus resultados y establecer sus conclusiones, además pensado en que siempre que uno realiza un análisis de este tipo es necesario elaborar reportes tanto de los datos como de los resultados obtenidos, se crea dentro del mismo ambiente de trabajo la posibilidad de preparar dichos reportes. Un gran reto que se tuvo fue la de estudiar y comprender la plataforma que Visual Borland C++ ofrece, aunque quizás no se ha explotado al máximo la programación sobre objetos que Borland ofrece se ha llegado a un software con un alto nivel de desempeño. Se inicio con un software existente para análisis de datos obtenidos por fotoemisión llamado *AAalyzer*¹, del cual se tomó la estructura principal. Posteriormente se acondicionó para el análisis de datos de EXAFS.

V.A. Esquema de Funcionamiento del Programa

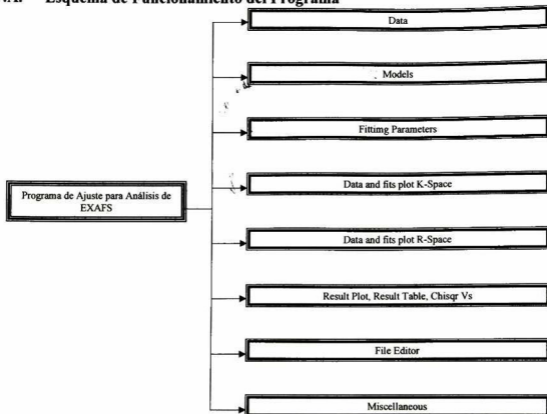


Figura V.1 diagrama de funcionamiento global del software desarrollado

V.A.2. Módulo "Data"

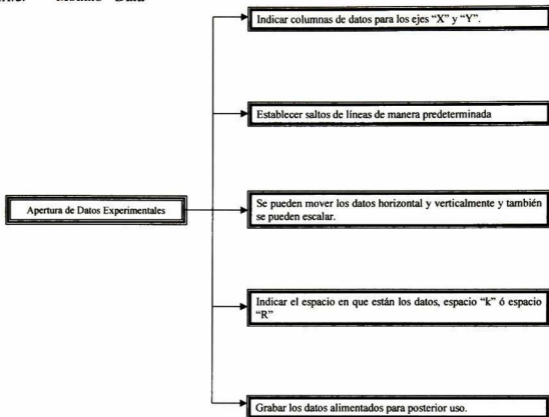


Figura V.2 Esquema que muestra las virtudes del módulo para la captura de los datos experimentales, también se pueden tratar los datos para realizar presentaciones en otro software como por ejemplo "Origin"

V.A.3. Módulo "Models"

Este módulo está pensado para poder ajustar los datos de EXAFS a través de modelos propuestos específicamente para este proyecto, sin embargo también pueden ajustarse en el módulo de "fitting parameters" para el caso general de que no tengan que emplearse propiamente un modelo.

Existen cuatro modelos clasificados como:

- Modelo 1 V-As₄
- Modelo 2 Si-V-As₃
- Modelo 3 Si₂-V-As₂
- Modelo 4 Si₄As

Cada uno tiene asignado los parámetros estructurales α y β , mismos que se ajustarán durante el análisis para determinar las deformaciones locales.

Para cada modelo se calculan las fases y amplitudes en cada una de las posibles trayectorias que puede tener un fotoelectrón durante la dispersión, estos datos están almacenados en archivos llamados feffxxxx, obtenidos a través del programa de la Universidad de Washington (feff.exe)^{2,3}. Cada archivo contiene los parámetros iniciales a usar en la ecuación de EXAFS, la suma de las contribuciones de todos los feffxxxx para un modelo específico forman un pico, en total podemos ajustar los datos experimentales con cuatro picos (uno por modelo). A continuación mostramos las características esenciales de los parámetros de ajuste en cada modelo:

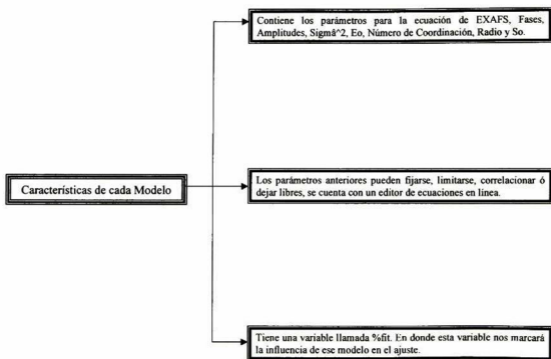


Figura V.3 Esquema que muestra el manejo de las variables que intervienen en la ecuación de EXAFS. Para cada modelo se realizó este mismo módulo.

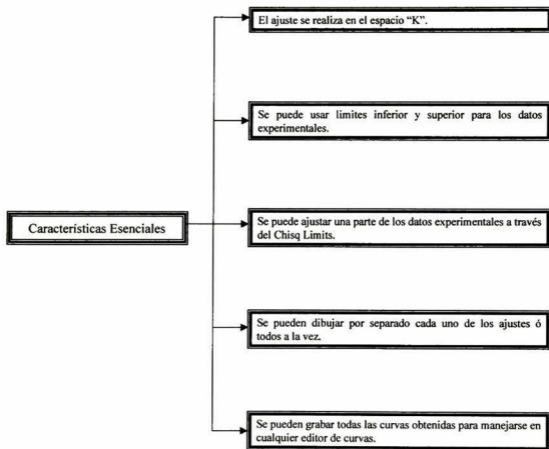
V.A.4. Módulo "Fitting Parameters"

Al igual que el módulo anterior podemos ajustar los datos de EXAFS sin necesidad de utilizar un modelo específico, aquí cada feffxxxx es un pico y contiene también todos los parámetros para la ecuación de EXAFS.(1.4) Las características de manejo para los parámetros son similares al módulo anterior y además en ambos casos se pueden guardar los parámetros alimentados para su posterior uso y realizar el ajuste en el espacio " k " ó en el espacio " R ".

Existe también la posibilidad de pesar los datos experimentales con la finalidad de darle importancia a la región más alejada del espectro.

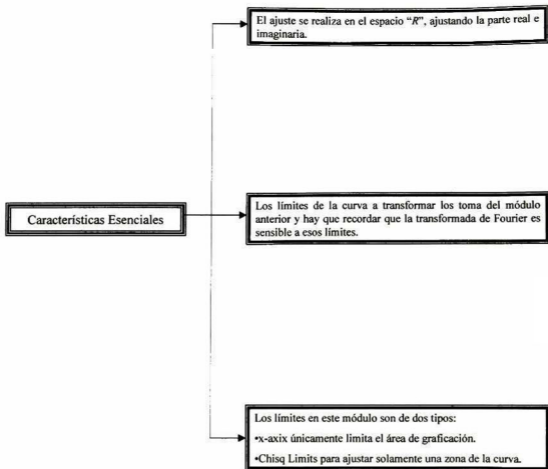
V.A.5. Módulo "Data and Fit Plot K-space"

En esta pantalla se muestra en forma gráfica los cambios durante el ajuste de los datos experimentales y en la parte inferior se grafica el residuo obtenido de la diferencia entre los puntos de la curva de ajuste con los puntos de los datos experimentales. Cuando el ajuste se hace desde esta pantalla automáticamente está direccionada para ajustar en el espacio "k".



V.A.6. Módulo "Data and Fit Plot R-space"

Tiene las mismas virtudes del módulo anterior con la diferencia de que el análisis realizado es en el espacio "R", obtenido a través de una transformada de Fourier del espacio "k", lo que se ajusta es la parte real e imaginaria de la transformada obteniendo la parte absoluta a partir de estas dos.



V.A.7. Módulos "Result Plot" y "Result Table"

En estos módulos se pueden graficar los parámetros que se dejaron libres durante el ajuste contra alguna variable externa. Estos módulos son de gran interés pues con ellos se pueden elaborar las presentaciones de los análisis de los resultados obtenidos a través de gráficos y cuadros de datos.

V.A.8. Módulo "File Editor"

Es un editor de textos dentro del mismo ambiente de trabajo, es simple y útil cuando se desea editar archivos de datos podemos hacerlo en este módulo de manera inmediata.

V.A.9. Módulo "Miscellaneous"

Aquí se podrán encontrar los nombres de las variables que reconoce el editor de ecuaciones al momento de querer correlacionar los parámetros de la ecuación de EXFAS(I.4). Se muestran algunos ejemplos de cómo correlacionar los parámetros entre modelos, picos ó datos.

Al igual que todo programa bajo windows se cuenta con un menú principal consistente en:

File	Edit	Actions	Option	Help
------	------	---------	--------	------

Cada uno de ellos cuentan con submenús que nos auxilia a maniobrar con el ajuste y la presentación de los resultados, asimismo, el programa cuenta con barra de herramientas para un rápido acceso para el usuario.

V.B. Diagrama de flujo de la rutina de ajuste

Es la parte medular del software desarrollado, para el ajuste de los datos experimentales se usó el propuesto por Levenberg⁴, mostraremos los diagramas de flujo de las rutinas principales

Programa principal aquí se identifica el tipo de ajuste, si se usan modelos y llama a la subrutina encargada de realizar el ajuste.

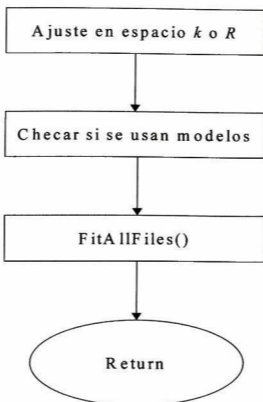


Figura V.4 Diagrama de flujo del programa principal de ajuste de datos

FitAllFiles() es una subrutina que se encarga de ajustar todos los archivos de datos activos a la vez, y claro que primero prepara los datos y picos para el ajuste.

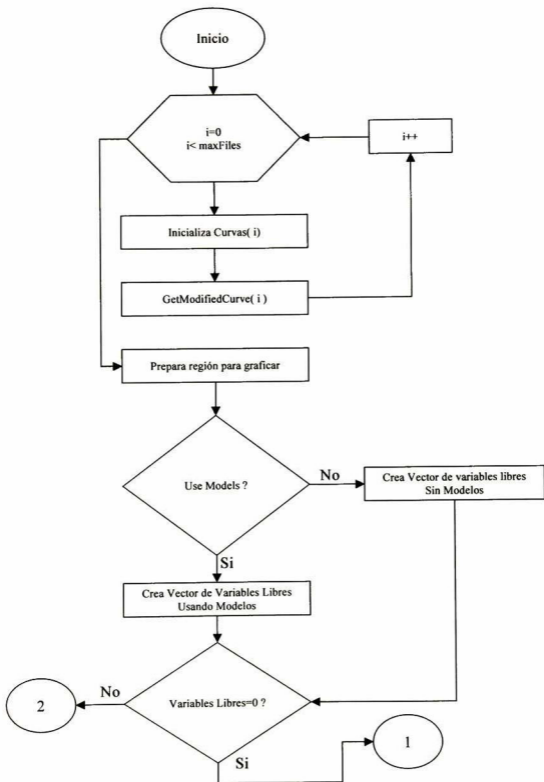


Figura V.5 Diagrama de flujo de FitAllFiles ()

Continuación de *FitAllFiles ()*

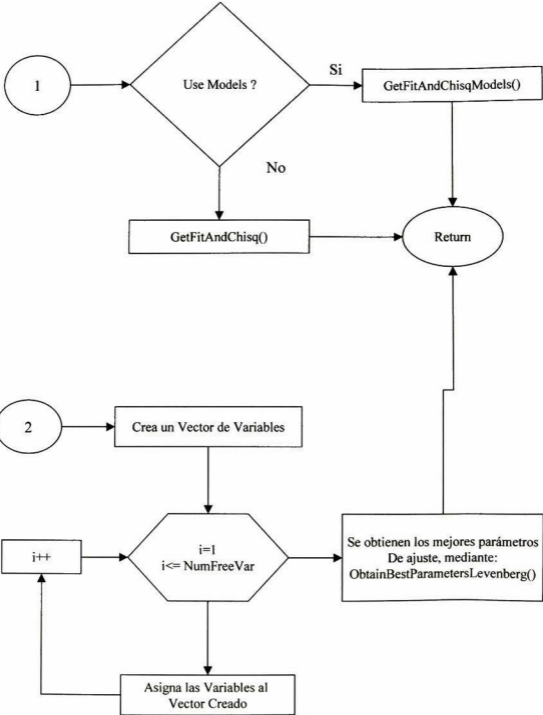


Figura V.6 Continuación del diagrama de flujo de *FitAllFiles()*.

V.B.2. Subrutina *ObtainBestParametersLevenberg()*

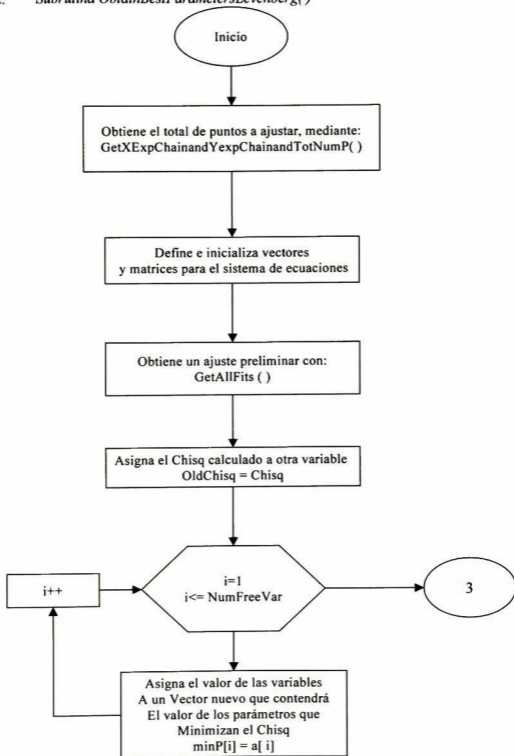


Figura V.7 Diagrama de flujo de la subrutina *ObtainBestParametersLevenberg()*

Continuación de *ObtainbestParametersLevenberg()*

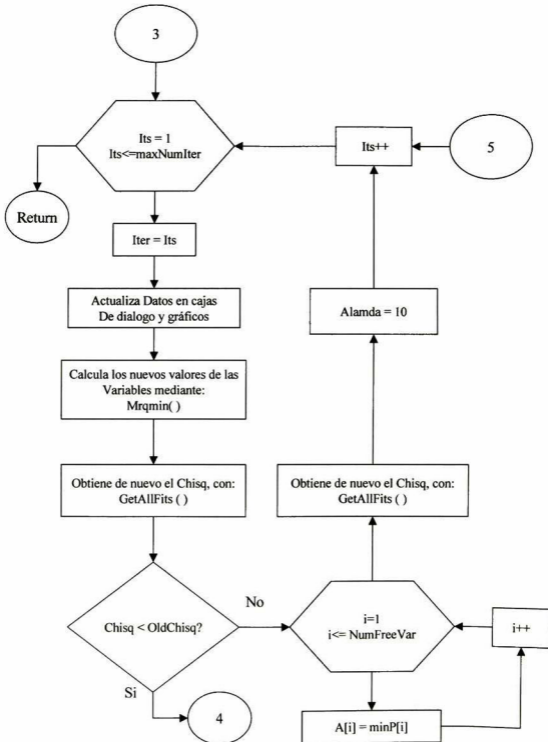


Figura V.8 Continuación del diagrama de flujo de la subrutina ObtainBestParametersLevenberg().

Continuación de *ObtainBestParametersLevenberg()*

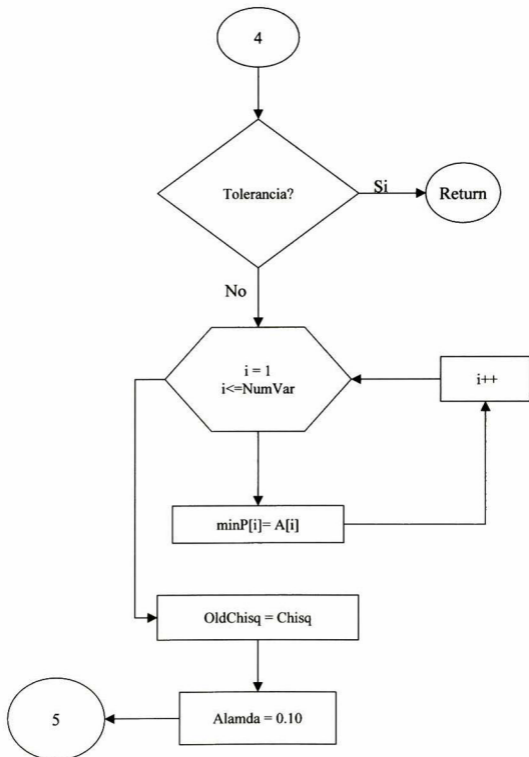


Figura V.9 Continuación del diagrama de flujo de la subrutina *ObtainBestParametersLevenberg()*.

Diagrama de flujo de Subrutina *mrqmin* ()

Obtiene los nuevos valores de los parametros libres, a través de la solución de un sistema de ecuaciones por el metodo de Gauss-Jordan

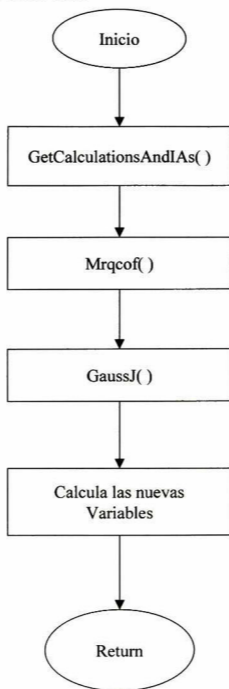


Figura V.10 Diagrama de flujo de subrutina *mrqmin*()

Bibliografía

- ¹ Software AAnalyzer desarrollado por el Dr. Alberto Herrera Gómez, CINVESTAV-QRO
- ² J. Mustre de Leon, J. J. Rehr, S. I. Zabinsky, and R. C. Albers. Phys. Rev. B **44**, 4146-4156 (1991).
- ³ John Rehr, Feff project, Departments of Phisycs, University of washing
- ⁴ Numerical Recipes in C, Second Edition, William H. Press et al, Editorial Cambridge, Cap. 12-15

VI. Análisis y Discusión

En los capítulos anteriores se discutieron:

- los fenómenos más relevantes que están involucrados en el proceso de desactivación eléctrica,
- el modelo matemático y
- el software ad hoc de cerca de 22,000 líneas de código fuente.

En el presente capítulo se aplica lo anterior a los datos experimentales de EXAFS, y se estudia la relevancia de los modelos propuestos.

VI.A. Consideraciones generales

Para ajustar los datos experimentales con la Ecuación (I.3) es necesario proponer las trayectorias posibles de los fotoelectrones. Para ello es necesario conocer la estructura local en la vecindad del átomo que absorbe, lo cual obliga a el uso de **modelos físico**, lo que a su vez obliga a una retroalimentación cíclica recurrente y el uso de información obtenida por otras técnicas. Existen en el mercado software creado exclusivamente para ajuste de datos de EXAFS; un ejemplo de ello es WinXas (uno de los mejores), el cual es muy útil para estructuras más simples. Para el caso en que se tengan asimetrías complejas como las nuestras (ver Capítulo IV), no es posible con los software existentes considerar las distorsiones locales que pudieran existir en un aglomerado de esta naturaleza. Por esta razón hubo la necesidad de desarrollar un software que permitiera el modelado de estructuras atómicas.

Los parámetros libres en la ecuación de EXAFS, tienen que ser cuidados muy meticulosamente pues en principio se obtenían muy buenos ajustes desde el punto de vista visual pero información desastrosa cuando se observaban los parámetros, ya que no eran congruentes con una realidad física. Por ejemplo S_0 es un factor reductor de la amplitud en la ecuación de EXAFS y trata de prever la presencia de muchos cuerpos alrededor del átomo que emite, cuando el átomo está aislado $S_0 = 1$, por tanto cuando en el ajuste se tienen valores mayores a uno no tiene ningún significado físico y se desecha el ajuste por muy bueno que éste halla sido, lo mismo sucede con E_0 , los factores de Debye-Waller o los números de coordinación.

VI.B. Consideraciones especiales durante el ajuste

Aunque se experimentó con bastantes situaciones en el manejo de los parámetros, la siguiente configuración de los parámetros de ajuste son los que representan un significado físico más congruente además de proporcionar un ajuste muy completo, la consideraciones tomadas fueron las siguientes.

- La Universidad de Washington¹ proporciona un software llamado *FEFF.exe* el cual es usado para determinar las fases y amplitudes usadas en el análisis por EXAFS, en los resultados que muestra uno de ellos es la energía E_0 la cual si hacemos ligeras variaciones no influye mucho en el ajuste razón por la que se mantuvo fija durante el ajuste de los datos.
- Las distancias R de los átomos vecinos al arsénico, se mantienen fijas pues estas se mueven de manera indirecta a través de los parámetros α y β para el aglomerado de acuerdo al tipo de modelo usado.
- Los factores de Debye-Waller se mantienen fijas pues se han correlacionado a las distancias radiales R de la forma $\sigma^2 = w R^2$. El software tiene una instrucción en pantalla para realizar esta correlación de manera automática.
- Los números de coordinación están intrínsecamente ligados al tipo de modelo, razón por la cual se escogen desde el mismo modelo durante el ajuste.
- S_0 se mantuvo fijo (e igual a uno) durante el ajuste pues en su lugar se colocaron en cada modelo un factor que regulara la amplitud en la ecuación de EXAFS ese factor es precisamente el porcentaje de la aportación de cada modelo en el ajuste y es el que nos indicará la magnitud de esos aglomerados en cada muestra para los diferentes tratamientos térmicos que se hicieron. Se espera que la suma de estos factores de 1 o muy cercano a ello.
- Otro aspecto importante que hay que resaltar es que las mediciones de EXAFS se hicieron a temperatura ambiente y es de esperarse que los diferentes tipos de aglomerados que estamos proponiendo en el modelo tengan el mismo defecto o estructura y que lo único que puede variar es la cantidad presente en esas muestras.
- El gran reto llega a su fin cuando el ajuste lo hacemos para todas las muestras a la vez, donde lo único que puede variar entre muestra y muestra es el porcentaje de influencia de los modelos utilizados.

A continuación mostramos una serie de gráficos con los ajustes realizados

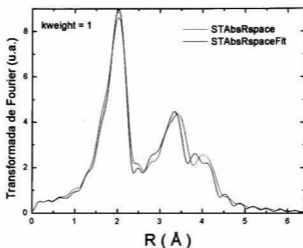


Figura VI.1 Ajuste realizado en el espacio R, existen pequeñas discrepancias entre la curva experimental y la de ajuste, principalmente a segundos y terceros vecinos.

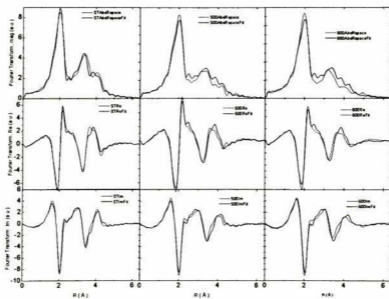


Figura VI.2 Ajuste de cuatro archivos de datos experimentales

VI.C. Discusión de los resultados del análisis

En la siguiente tabla y figura se muestra la tendencia de la población de los distintos defectos en función de la temperatura de equilibrio.

Tabla VI.1 Porcentajes de cada uno de los modelos en el ajuste de los datos experimentales para las muestras de 500-900 °C de tratamiento térmico, comparadas con la de sin tratamiento (ST).

Temp	As4-V	As3-V-Si	As2-V-Si2	Si4-As
500	11	22	25	38
600	13	18	25	34
700	11	17	28	34
800	11	19	27	38
900	3	19	35	38
ST	11	14	8	67

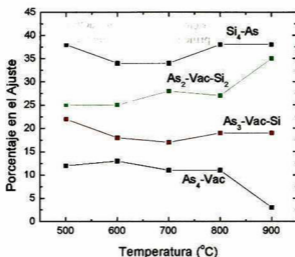


Figura VI.3 En esta figura se muestra la influencia de cada uno de los modelos y su variación con respecto a la temperatura de tratamiento

Los detalles de las estructuras de los defectos creados alrededor de la vacancia están descritos en la siguiente figura.

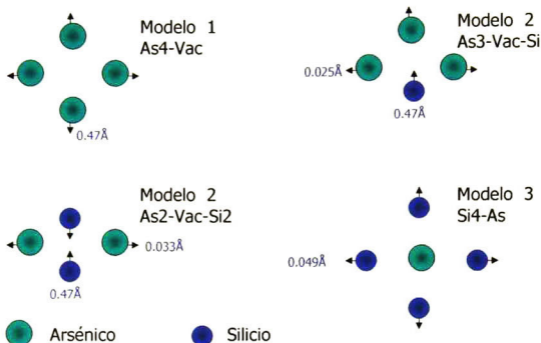


Figura VI.4 Estructura del defecto producido por la vacancia

Obsérvese como en todos los casos, el silicio tiende a ocupar el lugar de la vacancia, mientras que los arsénicos tienden a repelerse entre ellos, esto es quizás debido a la repulsión del par libre que tienen y que apunta hacia la vacancia. En la figura(IV.4) se puede ver como el agregado correspondiente a As2-V-Si2, empieza a incrementarse rápidamente conforme aumenta la temperatura, esto mismo lo había mostrado M. Berding con cálculos ab initio. De igual manera las distancias interatómicas por ejemplo a primeros vecinos son del orden de 2.32 a 2.40 Å, dependiendo del modelo y que no están muy lejos de los obtenidos por Pandey et al. O M. Berding et al. que andan del orden de 2.38-2.40 Å.

VII. Conclusiones y Perspectivas

VII.A. Conclusiones

En este estudio se obtuvieron los siguientes logros y conclusiones:

- Se planteó un modelo estructural para la formación de aglomerados de arsénico en la matriz de silicio. Para introducir el modelo estructural en el análisis cuantitativo de los datos experimentales de EXAFS fue necesario desarrollar un software de 22,000 líneas de código fuente.
- Se cuantificó la variación de la concentración de los aglomerados en función de la temperatura de tratamiento. La presencia del aglomerado As_2-V-Si_2 crece con la temperatura de tratamiento, comprobando experimentalmente la predicción teórica de M. Berding et al.². De igual forma sucede con la conservación del aglomerado Si_4-As el cual se mantiene casi constante a diferentes temperaturas.
- Los detalles de la estructura de los agregados fueron dilucidadas experimentalmente. Las distancias As-Si son del orden de 2.32-2.40 Å (depende del modelo), en ligero desacuerdo con los valores reportados por Pandey et al.³ y Parisini et al.⁴.

VII.B. Perspectivas y trabajo futuro

- Aunque se logró extraer la estructura de los defectos responsables de la desactivación eléctrica de arsénico en silicio, es necesario realizar la caracterización eléctrica para complementar los resultados obtenidos.
- En este trabajo se supuso que la estructura global de los defectos es del tipo propuesta por Cargil III. Con el método aquí desarrollado será posible discriminar la relevancia de otros modelos. Aunque resultados preliminares indicaron que el modelo propuesto por Chadi et al.⁵ (distorsiones angulares) no reproduce adecuadamente los resultados de EXAFS, aún no se le ha estudiado en detalle.
- Existe también la necesidad de hacer unas adecuaciones finales al software desarrollado para hacerlo más versátil y útil para incorporar fácilmente cualquier tipo de estructura.

Apéndice A: Fases y Amplitudes

Las fases y amplitudes para cada modelo propuesto dependen de cada trayectoria y fueron calculadas a partir de un software realizado en la Universidad de Washington,⁶ llamado feff.exe, es un software hecho en lenguaje Fortran especialmente para esta clase de investigaciones

A. Modelo 1 (V-As₄)

En la siguiente tabla se muestra una lista de feffxxx.dat con su respectiva aportación (relación de amplitud), su degeneración (deg) que corresponde al número de vecinos con esa trayectoria, el número de tramos (nlegs) que recorre el fotoelectrón para nlegs=2 corresponde a una dispersión simple(single-scattering) y para nlegs>2 corresponde a una dispersión múltiple.

Tabla VII.1 Concentrado de las diferentes trayectorias que tiene el fotoelectrón para una estructura correspondiente al modelo 1

File	amp ratio	deg	nlegs	r effective
feff0001.dat	100.000	3.000	2	2.3516
feff0002.dat	71.914	9.000	2	3.8401
feff0003.dat	29.781	3.000	2	3.8401
feff0004.dat	3.875	6.000	3	4.2716
feff0005.dat	27.909	18.000	3	4.2716
feff0006.dat	58.407	12.000	2	4.5029
feff0008.dat	4.483	9.000	4	4.7031
feff0009.dat	3.798	24.000	3	5.3473
feff0010.dat	12.203	36.000	3	5.3473
feff0011.dat	13.450	36.000	3	5.3473
feff0012.dat	4.430	12.000	3	5.3473
feff0013.dat	15.869	6.000	2	5.4307
feff0014.dat	4.871	30.000	3	5.7601
feff0015.dat	23.740	12.000	2	5.9180
feff0016.dat	6.126	12.000	3	6.0548
feff0017.dat	2.741	6.000	3	6.0548
feff0018.dat	12.728	18.000	3	6.0548
feff0019.dat	12.722	18.000	3	6.0548
feff0020.dat	4.200	6.000	3	6.0548
feff0021.dat	4.072	18.000	3	6.1426
feff0022.dat	5.908	24.000	3	6.1426
feff0023.dat	3.526	12.000	4	6.1916
feff0024.dat	3.093	9.000	4	6.1916
feff0025.dat	3.816	9.000	4	6.1916
feff0026.dat	6.521	18.000	4	6.1916
feff0027.dat	4.829	54.000	3	6.4229
feff0028.dat	4.129	36.000	3	6.4229

File	amp ratio	deg	nlegs	r effective
feff0029.dat	4.565	36.000	3	6.5554
feff0030.dat	31.629	24.000	2	6.6512
feff0031.dat	6.363	18.000	3	6.7528
feff0032.dat	11.452	18.000	3	6.7528
feff0033.dat	15.567	24.000	3	6.7528
feff0034.dat	4.530	18.000	4	6.8544
feff0035.dat	3.534	9.000	4	6.8545
feff0036.dat	5.705	12.000	4	6.8545
feff0037.dat	17.092	16.000	2	7.0547
feff0038.dat	2.937	36.000	3	7.1305
feff0039.dat	4.215	48.000	3	7.1305
feff0040.dat	9.697	72.000	3	7.1657
feff0041.dat	3.799	24.000	3	7.1657
feff0045.dat	4.370	24.000	3	7.4185
feff0046.dat	5.085	48.000	3	7.4604
feff0047.dat	9.408	12.000	2	7.6802
feff0048.dat	16.522	18.000	3	7.6802
feff0049.dat	5.616	6.000	3	7.6802
feff0050.dat	9.894	9.000	4	7.6802
feff0051.dat	3.596	3.000	4	7.6802
feff0052.dat	2.873	54.000	3	7.6988
feff0053.dat	4.662	54.000	3	7.6988
feff0054.dat	6.511	72.000	3	7.6988
feff0055.dat	5.456	96.000	3	7.8285
feff0057.dat	2.678	48.000	3	7.9610
feff0059.dat	3.322	24.000	3	7.9748
feff0062.dat	4.000	48.000	3	8.0287
feff0063.dat	15.934	24.000	2	8.0321

Aunque se tienen 54 trayectorias calculadas obsérvese que el radio efectivo sobrepasa en mucho a los datos experimentales de EXAFS en el espacio R, esto quiere decir que solamente los feffxxx con valores del radio efectivo menores a 6 Å tienen sentido usarlos en la aproximación de los datos. A continuación como ejemplo pondremos el contenido de un archivo feffxxx.dat, pues es de estos archivos donde el programa lee los datos para la ecuación de EXAFS.

Feff0001.dat

2, 3.000, 2.351, 6, 2.8530, -4.77875 nleg, deg, r_eff, mrmav(bohr), edge

k	real[2*phc]	mag[feff]	phase[feff]	red factor	lambda	real[p]@#
.000	5.4785E+00	0.0000E+00	-5.3371E+00	1.000E+00	1.1209E+01	1.8932E+00
.100	5.4784E+00	3.3356E-02	-5.7982E+00	1.000E+00	1.1224E+01	1.8957E+00
.200	5.4780E+00	6.6178E-02	-6.2409E+00	1.002E+00	1.1268E+01	1.9033E+00
.300	5.4772E+00	9.7928E-02	-6.6657E+00	1.004E+00	1.1338E+01	1.9158E+00

k	real[2*phc]	mag[feff]	phase[feff]	red factor	lambda	real[p]@#
.400	5.4756E+00	1.2806E-01	-7.0730E+00	1.007E+00	1.1431E+01	1.9333E+00
.500	5.4729E+00	1.5602E-01	-7.4637E+00	1.010E+00	1.1540E+01	1.9555E+00
.600	5.4684E+00	1.8128E-01	-7.8391E+00	1.015E+00	1.1658E+01	1.9823E+00
.700	5.4617E+00	2.0333E-01	-8.2006E+00	1.019E+00	1.1778E+01	2.0137E+00
.800	5.4523E+00	2.2176E-01	-8.5503E+00	1.024E+00	1.1890E+01	2.0493E+00
.900	5.4397E+00	2.3631E-01	-8.8908E+00	1.030E+00	1.1985E+01	2.0891E+00
1.000	5.4234E+00	2.4697E-01	-9.2252E+00	1.036E+00	1.2056E+01	2.1329E+00
1.100	5.4029E+00	2.5410E-01	-9.5572E+00	1.042E+00	1.2095E+01	2.1805E+00
1.200	5.3780E+00	2.5863E-01	-9.8909E+00	1.048E+00	1.2096E+01	2.2319E+00
1.300	5.3483E+00	2.6221E-01	-1.0230E+01	1.054E+00	1.2056E+01	2.2869E+00
1.400	5.3136E+00	2.6728E-01	-1.0575E+01	1.061E+00	1.1976E+01	2.3454E+00
1.500	5.2736E+00	2.7690E-01	-1.0923E+01	1.067E+00	1.1858E+01	2.4075E+00
1.600	5.2281E+00	2.9403E-01	-1.1264E+01	1.074E+00	1.1708E+01	2.4731E+00
1.700	5.1769E+00	3.2058E-01	-1.1584E+01	1.080E+00	1.1536E+01	2.5421E+00
1.800	5.1199E+00	3.5663E-01	-1.1870E+01	1.086E+00	1.1353E+01	2.6147E+00
1.900	5.0568E+00	4.0048E-01	-1.2116E+01	1.091E+00	1.1172E+01	2.6910E+00
2.000	4.9874E+00	4.4920E-01	-1.2321E+01	1.096E+00	1.1011E+01	2.7711E+00
2.200	4.8180E+00	5.4499E-01	-1.2607E+01	1.101E+00	1.0819E+01	2.9481E+00
2.400	4.6194E+00	6.8434E-01	-1.2796E+01	1.217E+00	7.2085E+00	3.1427E+00
2.600	4.5543E+00	8.3372E-01	-1.3067E+01	1.298E+00	5.8733E+00	3.2808E+00
2.800	4.5032E+00	9.2004E-01	-1.3321E+01	1.335E+00	5.3534E+00	3.4143E+00
3.000	4.4346E+00	9.4998E-01	-1.3565E+01	1.348E+00	5.1221E+00	3.5587E+00
3.200	4.3594E+00	9.5266E-01	-1.3803E+01	1.343E+00	5.0406E+00	3.7093E+00
3.400	4.2779E+00	9.3962E-01	-1.4026E+01	1.325E+00	5.0479E+00	3.8654E+00
3.600	4.1900E+00	9.1671E-01	-1.4231E+01	1.299E+00	5.1143E+00	4.0264E+00
3.800	4.0964E+00	8.9007E-01	-1.4417E+01	1.267E+00	5.2231E+00	4.1918E+00
4.000	3.9988E+00	8.6036E-01	-1.4586E+01	1.237E+00	5.3642E+00	4.3610E+00
4.200	3.8993E+00	8.3236E-01	-1.4743E+01	1.208E+00	5.5312E+00	4.5336E+00
4.400	3.7992E+00	8.0566E-01	-1.4888E+01	1.181E+00	5.7194E+00	4.7092E+00
4.600	3.6989E+00	7.8115E-01	-1.5024E+01	1.156E+00	5.9257E+00	4.8874E+00
4.800	3.5984E+00	7.5698E-01	-1.5151E+01	1.134E+00	6.1476E+00	5.0678E+00
5.000	3.4979E+00	7.3290E-01	-1.5271E+01	1.114E+00	6.3834E+00	5.2503E+00
5.200	3.3974E+00	7.0785E-01	-1.5386E+01	1.097E+00	6.6315E+00	5.4344E+00
5.400	3.2964E+00	6.7986E-01	-1.5493E+01	1.082E+00	6.8908E+00	5.6201E+00
5.600	3.1950E+00	6.4988E-01	-1.5596E+01	1.069E+00	7.1605E+00	5.8072E+00
5.800	3.0931E+00	6.1776E-01	-1.5696E+01	1.057E+00	7.4397E+00	5.9954E+00
6.000	2.9908E+00	5.8377E-01	-1.5792E+01	1.048E+00	7.7277E+00	6.1847E+00
6.500	2.7353E+00	5.0015E-01	-1.6018E+01	1.027E+00	8.4834E+00	6.6618E+00
7.000	2.4848E+00	4.2924E-01	-1.6222E+01	1.010E+00	9.2848E+00	7.1432E+00
7.500	2.2453E+00	3.7728E-01	-1.6402E+01	9.952E-01	1.0127E+01	7.6280E+00
8.000	2.0197E+00	3.3725E-01	-1.6565E+01	9.820E-01	1.1007E+01	8.1154E+00
8.500	1.8069E+00	3.0050E-01	-1.6724E+01	9.710E-01	1.1922E+01	8.6048E+00
9.000	1.6034E+00	2.6365E-01	-1.6888E+01	9.633E-01	1.2869E+01	9.0958E+00
9.500	1.4064E+00	2.2993E-01	-1.7051E+01	9.582E-01	1.3847E+01	9.5881E+00
10.000	1.2153E+00	2.0308E-01	-1.7198E+01	9.545E-01	1.4854E+01	1.0081E+01
11.000	8.5805E-01	1.6534E-01	-1.7441E+01	9.471E-01	1.6950E+01	1.1070E+01

k	real[2*phc]	mag[feff]	phase[feff]	red factor	lambda	real[p]@#
12.000	5.3277E-01	1.3301E-01	-1.7719E+01	9.379E-01	1.9148E+01	1.2062E+01
13.000	2.2342E-01	1.0868E-01	-1.7972E+01	9.313E-01	2.1442E+01	1.3055E+01
14.000	-6.8139E-02	9.1657E-02	-1.8165E+01	9.270E-01	2.3824E+01	1.4049E+01
15.000	-3.3778E-01	7.7513E-02	-1.8415E+01	9.242E-01	2.6290E+01	1.5045E+01
16.000	-5.9202E-01	6.6311E-02	-1.8600E+01	9.238E-01	2.8836E+01	1.6041E+01
17.000	-8.3186E-01	5.7563E-02	-1.8765E+01	9.241E-01	3.1456E+01	1.7038E+01
18.000	-1.0574E+00	5.0865E-02	-1.8983E+01	9.246E-01	3.4148E+01	1.8035E+01
19.000	-1.2717E+00	4.4353E-02	-1.9127E+01	9.261E-01	3.6908E+01	1.9032E+01
20.000	-1.4762E+00	3.8258E-02	-1.9304E+01	9.277E-01	3.9733E+01	2.0030E+01

Tabla VII.2 Fases y amplitudes calculadas a través del programa feff.exe de la Universidad de Washington

Como puede verse, cada columna corresponde a un valor de los parámetros en la ecuación de EXAFS para cada valor de K (\AA^{-1}), donde K está en función de la energía, existe una línea en los primeros renglones de comentarios del archivo donde se muestran los valores de n_{leg} , deg , r_{eff} y la energía (edge), todos los feffxxx.dat tienen el mismo orden con información similar, por lo que no es necesario mostrar cada uno de ellos.

B. Modelo 2 (Si-V-As₃)

file	amp ratio	deg	nlegs	r effective
feff0001.dat	100.000	3.000	2	2.3516
feff0002.dat	79.910	10.000	2	3.8401
feff0003.dat	19.856	2.000	2	3.8401
feff0004.dat	3.872	6.000	3	4.2716
feff0005.dat	27.877	18.000	3	4.2716
feff0006.dat	58.411	12.000	2	4.5029
feff0008.dat	4.472	9.000	4	4.7031
feff0009.dat	4.428	28.000	3	5.3473
feff0010.dat	12.191	36.000	3	5.3473
feff0011.dat	14.929	40.000	3	5.3473
feff0012.dat	15.869	6.000	2	5.4307
feff0013.dat	5.515	34.000	3	5.7601
feff0014.dat	23.741	12.000	2	5.9180
feff0015.dat	7.145	14.000	3	6.0548
feff0016.dat	12.715	18.000	3	6.0548
feff0017.dat	14.119	20.000	3	6.0548
feff0018.dat	2.796	4.000	3	6.0548
feff0019.dat	4.068	18.000	3	6.1426
feff0020.dat	5.903	24.000	3	6.1426
feff0021.dat	4.107	14.000	4	6.1916
feff0022.dat	3.086	9.000	4	6.1916
feff0023.dat	4.228	10.000	4	6.1916

file	amp ratio	deg	nlegs	r effective
feff0024.dat	6.504	18.000	4	6.1916
feff0025.dat	5.361	60.000	3	6.4229
feff0026.dat	4.124	36.000	3	6.4229
feff0027.dat	5.069	40.000	3	6.5554
feff0028.dat	31.630	24.000	2	6.6512
feff0029.dat	6.362	18.000	3	6.7528
feff0030.dat	11.442	18.000	3	6.7528

Tabla VII.3 Listado de Feffxxx.dat para el modelo, mostrando únicamente las primeras 30 trayectorias

C. Modelo 3 ($\text{Si}_2\text{V-As}_2$)

file	amp ratio	deg	nlegs	r effect
feff0001.dat	100.000	3.000	2	2.3516
feff0002.dat	87.905	11.000	2	3.8401
feff0003.dat	9.929	1.000	2	3.8401
feff0004.dat	3.869	6.000	3	4.2716
feff0005.dat	27.848	18.000	3	4.2716
feff0006.dat	58.414	12.000	2	4.5029
feff0008.dat	4.462	9.000	4	4.7031
feff0009.dat	5.058	32.000	3	5.3473
feff0010.dat	12.181	36.000	3	5.3473
feff0011.dat	16.405	44.000	3	5.3473
feff0012.dat	15.870	6.000	2	5.4307
feff0013.dat	6.483	40.000	3	5.7601
feff0014.dat	23.741	12.000	2	5.9180
feff0015.dat	8.163	16.000	3	6.0548
feff0016.dat	12.702	18.000	3	6.0548
feff0017.dat	15.515	22.000	3	6.0548
feff0018.dat	4.065	18.000	3	6.1426
feff0019.dat	5.899	24.000	3	6.1426
feff0020.dat	4.688	16.000	4	6.1916
feff0021.dat	3.080	9.000	4	6.1916
feff0022.dat	4.639	11.000	4	6.1916
feff0023.dat	6.488	18.000	4	6.1916
feff0024.dat	5.893	66.000	3	6.4229
feff0025.dat	4.120	36.000	3	6.4229
feff0026.dat	5.572	44.000	3	6.5554
feff0027.dat	31.631	24.000	2	6.6512
feff0028.dat	6.360	18.000	3	6.7528
feff0029.dat	11.433	18.000	3	6.7528
feff0030.dat	15.540	24.000	3	6.7528

Tabla VII.4 Listado de feffxxx.dat, obsérvese la variación en la degeneración (deg) y la relación de amplitud con respecto al modelo anterior

D. Modelo 4 (Si_4 -As)

file	sig2	amp ratio	deg	nlegs	r effective
feff0001.dat	.00000	100.000	4.000	2	2.3516
feff0002.dat	.00000	71.914	12.000	2	3.8401
feff0003.dat	.00000	5.803	12.000	3	4.2716
feff0004.dat	.00000	27.856	24.000	3	4.2716
feff0005.dat	.00000	43.811	12.000	2	4.5029
feff0007.dat	.00000	4.463	12.000	4	4.7031
feff0008.dat	.00000	2.996	12.000	4	4.7031
feff0009.dat	.00000	5.694	48.000	3	5.3473
feff0010.dat	.00000	12.189	48.000	3	5.3473
feff0011.dat	.00000	13.436	48.000	3	5.3473
feff0012.dat	.00000	11.904	6.000	2	5.4307
feff0013.dat	.00000	5.840	48.000	3	5.7601
feff0014.dat	.00000	17.808	12.000	2	5.9180
feff0015.dat	.00000	9.182	24.000	3	6.0548
feff0016.dat	.00000	12.681	24.000	3	6.0548
feff0017.dat	.00000	12.671	24.000	3	6.0548
feff0018.dat	.00000	4.067	24.000	3	6.1426
feff0019.dat	.00000	4.427	24.000	3	6.1426
feff0020.dat	.00000	5.383	24.000	4	6.1916
feff0021.dat	.00000	3.742	24.000	4	6.1916
feff0022.dat	.00000	3.079	12.000	4	6.1916
feff0023.dat	.00000	3.768	12.000	4	6.1916
feff0024.dat	.00000	6.485	24.000	4	6.1916
feff0025.dat	.00000	4.826	72.000	3	6.4229
feff0026.dat	.00000	3.095	36.000	3	6.4229
feff0027.dat	.00000	4.565	48.000	3	6.5554
feff0028.dat	.00000	23.723	24.000	2	6.6512
feff0029.dat	.00000	6.359	24.000	3	6.7528
feff0030.dat	.00000	11.432	24.000	3	6.7528

Tabla VII.5 Listado de feffxxx.dat para el modelo Si_4 As

Apéndice B: Código fuente de la rutina de ajuste

A. Programa principal

/*Programa principal, ajusta todos los archivos de datos activos, con los mismos parámetros, pero variando los porcentajes de ajuste para cada modelo usado*/

```
void __fastcall TFirm3::ActionsFitAllTogetherExecute(TObject *Sender)
{
    int typeSpaceFitting;

    youJustFittedLabel->Caption = "all together";
    typeSpaceFitting = checkTypeSpaceFitting();
    checkIfUseModels();
    FitAllFiles(typeSpaceFitting);
}
```

B. FitAllFiles()

/* Sub-rutina encargada de identificar los archivos de datos activos y de acuerdo al tipo de ajuste (typeSpaceFitting), y obtiene en cualquiera de los casos los mejores parámetros para el ajuste de acuerdo al metodo de Levenberg.

```
void TFirm3::FitAllFiles(int typeSpaceFitting)
{
#define FREEALL free_vector(p, 1, numFreeVariables);
    int i;
    float *p;
    int iter;
    float chisqMin;
    int numFreeVariables, numFreeAreas;
    double *freeVariablesPtr[maxNumFreeVar];
    bool tooManyVariables;

    TWaitCursor wait; // show hourglass
                    // cursor restored on exit of block
    singularity = false;
    for (i=0; i<maxFiles; i++)
        if ( Data[i].fit )
        {
            if ( typeSpaceFitting == spaceR )
            {
                InitializeOneCurve( &Data[i].FitCurveAbs[typeSpaceFitting] );
                InitializeOneCurve( &Data[i].FitCurveRe );
                InitializeOneCurve( &Data[i].FitCurveIm );
            }
            else
                InitializeOneCurve( &Data[i].FitCurveAbs[typeSpaceFitting] );

            GetModifiedCurve(i);
        }
}
```

```

}

if( !FindXRegionForFitting( &rectToFit, typeSpaceFitting ) )
    return;
if( useModels )
CreateFreeVariablesVectorWithModel(&numFreeVariables, &numFreeAreas, freeVariablesPtr, typeSpaceFitting,
tooManyVariables);
else
    CreateFreeVariablesVector(&numFreeVariables, &numFreeAreas, freeVariablesPtr,
&tooManyVariables, typeSpaceFitting );

if( tooManyVariables )
{
    Application->MessageBox( "too many free variables!", "AAalyzer --> Warning!", MB_OK);
    tooManyVariables = false;
    return;
}
else if( numFreeVariables == 0 )
{
    if( useModels )
        GetFitAndChisqModels(typeSpaceFitting);
    else
        GetFitAndChisq(typeSpaceFitting);

    if( singularity )
        return;
}
else
{
    p = vector(1, numFreeVariables );
    for( i=1; i<=numFreeVariables; i++ )
        p[i] = *freeVariablesPtr[i-1];
    ObtainBestParametersLevenberg( p, numFreeVariables, numFreeAreas, tolerance, &iter, &chisqMin,
typeSpaceFitting );
    if( singularity )
    {
        FREEALL
        return;
    }
    if( stop )
    {
        stop = false;
        stopAbove = true;
    }
    FREEALL
}

/* if( !individualExponential )
for( i=0; i<maxFiles; i++ )
    Data[i].exponent = exponent; */

UpdateAllBoxes();

MessageBeep(1);
#undef FREEALL
}

```

C. CreateFreeVariablesVectorWithModel()

/*Sub-Rutina encargada de identificar y formar el vector de variables libres usando los modelos propuestos en el presente proyecto, los guarda en un vector.*/

```
void TFirm3::CreateFreeVariablesVectorWithModel(int *numFreeVarPtr, int *numFreeAreasPtr, double *freeVariablesPtr[],
```

```
int typeSpace, bool tooManyVariablesPtr )
{
    int i, j;
    int numFreeVar = 0;

    tooManyVariablesPtr = false;
    for (i=0; i<maxModels; i++)
        if( model[i].modelActive )
            if( model[i].fractionVary.vary == free || model[i].fractionVary.vary == limited )
                {
                    for( j=0; j<maxFiles; j++ )
                        if( Data[j].gotModifiedCurveAbs[typeSpace] && Data[j].fit )
                            {
                                freeVariablesPtr[numFreeVar] = &Data[j].fractionModel[i];
                                numFreeVar++;
                            }
                }
    *numFreeAreasPtr = numFreeVar;
    for (i=0; i<maxModels; i++)
        {
            if( model[i].modelActive )
                {
                    for( j=0; j<maxPeaks; j++ )
                        if( model[i].peak[j].active )
                            {
                                if( model[i].peak[j].energyVary.vary == free || model[i].peak[j].energyVary.vary == limited )
                                    {
                                        freeVariablesPtr[numFreeVar] = &model[i].peak[j].energy;
                                        numFreeVar++;
                                    }
                                if( model[i].peak[j].gaussVary.vary == free || model[i].peak[j].gaussVary.vary == limited )
                                    {
                                        freeVariablesPtr[numFreeVar] = &model[i].peak[j].gauss;
                                        numFreeVar++;
                                    }
                                else if (model[i].peak[j].gaussVary.vary == correlated &&
                                strcmp(model[i].peak[j].gaussVary.correlation,"correlationSigmaOfRatio") == 0 )
                                    {
                                        freeVariablesPtr[numFreeVar] = &model[i].peak[j].factorSigma;
                                        numFreeVar++;
                                    }
                                if( model[i].peak[j].branchingRatioVary.vary == free ||
                                model[i].peak[j].branchingRatioVary.vary == limited )
                                    {
                                        freeVariablesPtr[numFreeVar] = &model[i].peak[j].branchingRatio;
                                        numFreeVar++;
                                    }
                            }
                }
        }
}
```



```

switch (i)
{
case model4AsV:
    if( model[i].alphaAsVary.vary == free || model[i].alphaAsVary.vary == limited )
    {
        freeVariablesPtr[numFreeVar] = &model[i].alphaAs;
        numFreeVar++;
    }
    if( model[i].betaSiVary.vary == free || model[i].betaSiVary.vary == limited )
    {
        freeVariablesPtr[numFreeVar] = &model[i].betaSi;
        numFreeVar++;
    }
    if( model[i].S0Vary.vary == free || model[i].S0Vary.vary == limited )
    {
        freeVariablesPtr[numFreeVar] = &model[i].S0;
        numFreeVar++;
    }
    break;
case model3AsVSi:
    if( model[i].alphaAsVary.vary == free || model[i].alphaAsVary.vary == limited )
    {
        freeVariablesPtr[numFreeVar] = &model[i].alphaAs;
        numFreeVar++;
    }
    if( model[i].alphaSiVary.vary == free || model[i].alphaSiVary.vary == limited )
    {
        freeVariablesPtr[numFreeVar] = &model[i].alphaSi;
        numFreeVar++;
    }
    if( model[i].betaSiVary.vary == free || model[i].betaSiVary.vary == limited )
    {
        freeVariablesPtr[numFreeVar] = &model[i].betaSi;
        numFreeVar++;
    }
    if( model[i].S0Vary.vary == free || model[i].S0Vary.vary == limited )
    {
        freeVariablesPtr[numFreeVar] = &model[i].S0;
        numFreeVar++;
    }
    break;
case model2AsV2Si:
    if( model[i].alphaAsVary.vary == free || model[i].alphaAsVary.vary == limited )
    {
        freeVariablesPtr[numFreeVar] = &model[i].alphaAs;
        numFreeVar++;
    }
    if( model[i].alphaSiVary.vary == free || model[i].alphaSiVary.vary == limited )
    {
        freeVariablesPtr[numFreeVar] = &model[i].alphaSi;
        numFreeVar++;
    }
    if( model[i].betaSiVary.vary == free || model[i].betaSiVary.vary == limited )
    {
        freeVariablesPtr[numFreeVar] = &model[i].betaSi;
    }
}

```

```

        numFreeVar++;
    }
    if( model[i].S0Vary.vary == free || model[i].S0Vary.vary == limited )
    {
        freeVariablesPtr[numFreeVar] = &model[i].S0;
        numFreeVar++;
    }
    break;
case modelAs4Si:
    if( model[i].alphaSiVary.vary == free || model[i].alphaSiVary.vary == limited )
    {
        freeVariablesPtr[numFreeVar] = &model[i].alphaSi;
        numFreeVar++;
    }
    if( model[i].betaSiVary.vary == free || model[i].betaSiVary.vary == limited )
    {
        freeVariablesPtr[numFreeVar] = &model[i].betaSi;
        numFreeVar++;
    }
    if( model[i].S0Vary.vary == free || model[i].S0Vary.vary == limited )
    {
        freeVariablesPtr[numFreeVar] = &model[i].S0;
        numFreeVar++;
    }
}
} //end of switch
} // end of if ( model[i].modelActive )
*numFreeVarPtr = numFreeVar;
if( numFreeVar > maxNumFreeVar )
{
    tooManyVariablesPtr = true;
    return;
}
}
}

```

D. GetXExpChainandYExpChainandTotNumP()

/*En el metodo de Levenberg es necesario colocar todos los archivos activos para ajustar en un vector, esta Sub-Rutina se encarga de establecer la magnitud de este vector de acuerdo a los datos activos en el ajuste, así como del tipo de ajuste a realizar (ajuste en el espacio K o en el espacio R)*/

```

void TFirm3::GetXExpChainandYExpChainandTotNumP( double *x, double *y, int *totalNumPPtr, int typeSpace )
{
    int    i, j, numP;
    int    m = 0;

    switch (typeSpace)
    {
        case spaceK:
            for( i=0; i<maxFiles; i++ )
                if( Data[i].gotModifiedCurveAbs[typeSpace] && Data[i].fit )

```

```

    {
        numP = Data[i].ModifiedCurveAbs[typeSpace].numP;
        for( j=0; j<numP; j++ )
        {
            x[m] = Data[i].ModifiedCurveAbs[typeSpace].x[j];
            y[m] = Data[i].ModifiedCurveAbs[typeSpace].y[j];
            m++;
        }
    }
    *totalNumPPtr = m;
    break;
case spaceR:
    for( i=0; i<maxFiles; i++ )
    {
        if( Data[i].gotModifiedCurveRe && Data[i].fit )
        {
            numP = Data[i].ModifiedCurveRe.numP;
            for( j=0; j<numP; j++ )
            {
                x[m] = Data[i].ModifiedCurveRe.x[j];
                y[m] = Data[i].ModifiedCurveRe.y[j];
                m++;
            }
        }
        if( Data[i].gotModifiedCurveIm && Data[i].fit )
        {
            numP = Data[i].ModifiedCurveIm.numP;
            for( j=0; j<numP; j++ )
            {
                x[m] = Data[i].ModifiedCurveIm.x[j];
                y[m] = Data[i].ModifiedCurveIm.y[j];
                m++;
            }
        }
    }
    *totalNumPPtr = m;
} //end of switch
}

```

E. GetAllFits()

/*Sub-Rutina encargada de copiar los valores de los parámetros libres a un vector y por supuesto este arreglo es distinto en el caso de que se usen o no modelos y de acuerdo también al tipo de ajuste*/

```

void TFirm3::GetAllFits( float a[], int numV, float *chisqPtr, int typeSpace )
{
    if(!useModels)
    {
        CopyNewValuesToFreeParameters(a, typeSpace);
        CheckAllLimits();
        CheckAllCorrelations();
        *chisqPtr = GetFitAndChisq(typeSpace);
    }
}

```

```

        if( singularity )
            return;
    }
    else
    {
        GetChiSqModels( a, typeSpace );
        checkAllModelsLimitsAndCorrelations(typeSpace);
        *chisqPtr = GetFitAndChisqModels(typeSpace);
        if( singularity )
            return;
    }
}

```

F. ObtainBestParametersLevenberg()

/* Subrutina principal del metodo de Levenberg, con la cual se obtienen los mejores parámetros de todos los archivos de datos a ajustar*/

```

void TFirm3::ObtainBestParametersLevenberg( float a[], int numV, int numFreeAreas, float ftol, int *iter, float *chisqPtr, int typeSpaceFitting )

```

```

{
//Given a starting value for the free variables a[1..numV], Levenberg fitting is performed on a
//function (x,y). The convergence tolerance on the function value is input as ftol. Returned
//quantities are a (the location of the minimum), iter (the number of iterations that were
//performed), and chisqPtr.

```

```

#define EPS 1.0e-12

```

```

#define FREEALL free_vector(xp, 1, totalNumP );free_vector(yp, 1, totalNumP );free_vector(sig, 1, totalNumP );
#define FREEALL2 free_matrix(covar, 1, numV, 1, numV );free_matrix(alpha, 1, numV, 1, numV );

```

```

int totalNumP;
double x[maximumNumP], y[maximumNumP];
float *xp, *yp, *sig, **covar, **alpha;
int i, j, its;
float oldChiSq;
float alamda = 0.001;
int numberOfTimesBelowTolerance = 0;
int maximumNumberOfTimesBelowTolerance = 3;
bool stopResponse;

```

```

for( i=0; i<maximumNumP; i++ )
{
    x[i] = 0;
    y[i] = 0;
}

```

```

GetXExpChainandYExpChainandTotNumP( x, y, &totalNumP, typeSpaceFitting );

```

```

xp = vector(1,totalNumP);
yp = vector(1,totalNumP);
sig = vector(1,totalNumP);
covar = matrix( 1, numV, 1, numV );
alpha = matrix( 1, numV, 1, numV );

```

```

for( i=1; i<=numV; i++ )
{
//   ia[i] = 1;
   for( j=1; j<=numV; j++ )
   {
       covar[i][j] = 1;
       alpha[i][j] = 1;
   }
}
for( i=0; i<totalNumP; i++ )
{
   xp[i+1] = x[i];
   yp[i+1] = y[i];
   sig[i+1] = 1;
}

GetAllFits( a, numV, chisqPtr, typeSpaceFitting );
if( singularity )
{
    FREEALL
    FREEALL2
    return;
}
oldChiSq = *chisqPtr;

// remember the "a" that provides the lowest chisq
for( i=1; i<= numV; i++ )
    minP[i] = a[i];

for( its=1; its<=maxNumIterations; its++ )
{
    *iter=its;

    UpdateAllBetweenIterations( its, *chisqPtr, typeSpaceFitting );

    Application->ProcessMessages(); // Allows to react to the STOP instruction
    if( stop )
    {
        stopResponse = Application->MessageBox("stop fitting?", "AAnalyzer --> Alert!", MB_YESNO);
        if( stopResponse == IDYES )
        {
            FREEALL
            FREEALL2
            return;
        }
        else
            stop = false;
    }
}

mrqmin( xp, yp, sig, totalNumP, a, numV, numFreeAreas, covar, alpha,
        chisqPtr, &alamda, typeSpaceFitting );

if( singularity )
{
    FREEALL
    FREEALL2
    return;
}

```

```

}
GetAllFits( a, numV, chisqPtr, typeSpaceFitting );
if( singularity )
{
    FREEALL
    FREEALL2
    return;
}
if( *chisqPtr < oldChiSq )
{
    if( 2.0*fabs(*chisqPtr - oldChiSq) <= ftoi*( *chisqPtr + oldChiSq + EPS ) )
    {
        numberOfTimesBelowTolerance++;
        if( numberOfTimesBelowTolerance == maximumNumberOfTimesBelowTolerance )
            break;
    }
    else
        numberOfTimesBelowTolerance = 0;

    for( i=1; i<= numV; i++ )
        minP[i] = a[i];
    oldChiSq = *chisqPtr;
    alambda *= 0.1;
}
else
{
    for( i=1; i<= numV; i++ )
        a[i] = minP[i];
    GetAllFits( a, numV, chisqPtr, typeSpaceFitting );
    if( singularity )
    {
        FREEALL
        FREEALL2
        return;
    }
    alambda *= 10;
}
}

FREEALL
FREEALL2

#undef FREEALL
#undef FREEALL2
#undef EPS
}

```

G. mrqmin()

/*Esta sub-Rutina es el corazon principal del metodo de Levenberg la cual a través de una solución de un sistema de ecuaciones obtiene los valores de todas las variables en el ajuste.*/

```

void TFirm3::mrqmin( float x[], float y[], float sig[], int ndata, float a[],
    int ma, int numFreeAreas, float **covar, float **alpha, float *chisq, float *alamda, int typeSpace )
{
#define FREEALL free_matrix(onedata,1,mfit,1,1);free_vector(da,1,ma);free_vector(beta,1,ma);free_ivector(ia, 1, ma
);
#define FREEalmostALL free_vector(da,1,ma);free_vector(beta,1,ma);free_ivector(ia, 1, ma );
// free_vector(atry,1,ma);
int      j,k,l;
int      mfit;
float    *da, **onedata;
float    *beta;
int      *ia;
int      i;

da=vector(1,ma);
beta=vector(1,ma);
ia = ivector( 1, ma );

GetCalculationsAndIAS( a, ma, ndata, chisq, ia, typeSpace );
if( singularity )
{
    FREEalmostALL
    return;
}
mrqcof(x,y,sig,ndata,a,ia,ma,numFreeAreas,alpha,beta,chisq,typeSpace);
if( singularity )
{
    FREEalmostALL
    return;
}
for (mfit=0;j=1;j<=ma;j++)
{
    if( ia[j] )
        mfit++;
}

onedata=matrix(1,mfit,1,1);

for (j=1;j<=mfit;j++) //Alter linearize fitting matrix, by augmenting diagonal elements
{
    for (k=1;k<=mfit;k++)
        covar[j][k]=alpha[j][k];
    covar[j][j]=alpha[j][j]*(1.0+(*alamda));
    onedata[j][1]=beta[j];
}

gaussj(covar,mfit,onedata,1); //Matrix solution.
if( singularity )
{
    FREEALL
    return;
}

for(j=1;j<=mfit;j++)
    da[j]=onedata[j][1];

```

```

for (j=0,l=1;l<=ma;l++) //Did the trial succeed?
  if (ia[l])
//    atry[l]=a[l]+da[+j];
    a[l]=a[l]+da[+j];

FREEALL

#undef FREEALL
#undef FREEalmostALL
}

```

H. GetCalculationsAndIAs()

Aquí se determina el grado de influencia que tiene cada una de las variables en el ajuste, así como también a través del gradiente se puede ver si el parámetro libre es un parámetro activo durante el ajuste.

```

void TFirm3::GetCalculationsAndIAs( float *a, int numV, int totalNumP,
                                   float *chisqPtr, int *ia, int typeSpace )
{
  int    i, j;
  double totalGrad, gradSq[maxNumFreeVar];

  CalculateYCalChainAndGradientCalChain( a, numV, totalNumP, chisqPtr, typeSpace );
  if( singularity )
  {
    return;
  }
  for( totalGrad = 0, j=0; j<numV; j++)
  {
    for( gradSq[j] = 0, i=0; i<totalNumP; i++ )
      gradSq[j] += grad[i][j] * grad[i][j];
    totalGrad += gradSq[j];
  }
  for( j=0; j<numV; j++)
  {
    if( gradSq[j] < 1e-10*totalGrad )
      ia[j+1] = 0;
    else
      ia[j+1] = 1;
  }
}

```


I. mrqcof()

Aquí se definen los elementos de los arreglos para el sistema de ecuaciones a resolver por el metodo de Gauss_Jordan en la subrutina mrqmin.

```
void TFirm3::mrqcof(float x[], float y[], float sig[], int ndata, float a[], int ia[],
    int ma, int numFreeAreas, float **alpha, float beta[], float *chisqPtr, int typeSpace )
{
//Used by mrqmin to evaluate the linearize fitting matrix alpha and vector beta as in (15.5.8),
//and calculate chisq.
#define FREEALL free_vector(dyda,1,ma);
    int i,j,k,l,m,mfit=0;
    float ymod,wt,sig2i,dy,*dyda;

    dyda=vector(1,ma);

    for (j=1;j<=ma;j++)
        if (ia[j])
            mfit++;
    for (j=1;j<=mfit;j++) //Initialize (symmetric)alpha beta
    {
        for (k=1;k<=j;k++)
            alpha[j][k]=0.0;
        beta[j]=0.0;
    }
    for (i=1;i<=ndata;i++) // Summation loop over all data.
    {
        for (j=1;j<=ma;j++) // instead of func
            dyda[j] = grad[i-1][j-1];
        ymod = yCal[i-1];
        sig2i=1.0/(sig[i]*sig[i]);
        dy=y[i]-ymod;
        for (j=0,l=1;l<=ma;l++)
        {
            if (ia[l])
            {
                wt=dyda[l]*sig2i;
                for (j++,k=0,m=1;m<=l;m++)
                    if (ia[m])
                        alpha[j][++k] += wt*dyda[m];
                beta[j] += dy*wt;
            }
        }
    }

    for (j=2;j<=mfit;j++) //Fill in the symmetric side.
        for (k=1;k<=j;k++) alpha[k][j]=alpha[j][k];
    FREEALL
    if( treatAreasSimilarly )
    {
        if( !allowForNegativeAreas )
        {
            for( j=1; j<= numFreeAreas; j++ )
                if( a[j] < 0 && ia[j] == 1 )

```

```

    {
        a[j] = 0;
        ia[j] = 0;
        if(!useModels)
            CopyNewValuesToFreeParameters(a, typeSpace);
        else
            GetChiSqModels(a, typeSpace);
        mrqcof(x,y,sig,ndata,a,ia,ma,numFreeAreas,alpha,beta,chisqPtr, typeSpace);
        if( singularity )
        {
//            FREEALL
            return;
        }
        ia[j] = 1;
        break;
    }
}
}
#endif FREEALL
}

```

J. CalculateYCalChainAndGradientCalChain()

```

void TFirm3::CalculateYCalChainAndGradientCalChain( float *a, int numV, int totalNumP,
                                                    float *chisqPtr, int typeSpace )
{
    switch (typeSpace)
    {
        case spaceK:
            CalculateYCalChainAndGradientCalChainAbs( a, numV, totalNumP,
                                                       chisqPtr, spaceK );
            break;
        case spaceR:
            CalculateYCalChainAndGradientCalChainReAndIm( a, numV, totalNumP,
                                                           chisqPtr, spaceR );
    }
}

```

K. CalculateYCalChainAndGradientCalChainAbs()

```

void TFirm3::CalculateYCalChainAndGradientCalChainAbs( float *a, int numV, int totalNumP,
                                                       float *chisqPtr, int typeSpace )
{
    int    i, j, k, numP;
    int    m = 0;
    int    stopResponse;

    GetAllFits(a, numV, chisqPtr, spaceK);
    if( singularity )
        return;
    if(useModels)
    {
        GradChiSqModels(a, numV, spaceK );
    }
}

```

```

    if( singularity )
        return;
    }
    else
    {
        GetAllGradientsKspace(a, numV, spaceK);
        if( singularity )
            return;
    }
    for( i=0; i<maxFiles; i++)
        if( Data[i].gotModifiedCurveAbs[typeSpace] && Data[i].fit )
            {
                numP = Data[i].ModifiedCurveAbs[typeSpace].numP;
                for( j=0; j<numP; j++)
                    {
                        yCal[m] = Data[i].FitCurveAbs[typeSpace].y[j];

                        for(k=0; k<numV; k++)
                            grad[m][k] = Data[i].GradY[j][k];

                        m++;
                    }
            }
    if( m != totalNumP )
    {
        stopResponse = Application->MessageBox("incompatible total number of points,\ncontinue fitting?",
        "AAalyzer --> Alert!", MB_YESNO);
        if( stopResponse == IDNO )
            {
                singularity = true;
                return;
            }
    }
}
}

```

L. CalculateYCalChainAndGradientCalChainReAndIm()

/*En esta subrutina es donde se calcula el gradiente a través de pequeños incrementos de los parámetros libres*/

```

void TFirm3::CalculateYCalChainAndGradientCalChainReAndIm( float *a, int numV, int totalNumP,
float *chisqPtr, int typeSpace )
{
    int    i, j, k, numP;
    int    m = 0;
    int    stopResponse;

    GetAllFits(a, numV, chisqPtr, spaceR);
    if( singularity )
        return;
    if( useModels )
    {
        GradChiSqModels(a, numV, spaceR );
        if( singularity )

```

```

    return;
}
else
{
    GetAllGradientsRspace(a, numV, spaceR);
    if( singularity )
        return;
}
for( i=0; i<maxFiles; i++ )
{
    if( Data[i].gotModifiedCurveRe && Data[i].fit )
    {
        numP = Data[i].ModifiedCurveRe.numP;
        for( j=0; j<numP; j++ )
        {
            yCal[m] = Data[i].FitCurveRe.y[j];

            for(k=0; k<numV; k++ )
                grad[m][k] = Data[i].GradY[j][k];

            m++;
        }
    }
    if( Data[i].gotModifiedCurveIm && Data[i].fit )
    {
        numP = Data[i].ModifiedCurveIm.numP;
        for( j=0; j<numP; j++ )
        {
            yCal[m] = Data[i].FitCurveIm.y[j];

            for(k=0; k<numV; k++ )
                grad[m][k] = Data[i].GradY[ Data[i].ModifiedCurveRe.numP + j ][k];

            m++;
        }
    }
}
if( m != totalNumP )
{
    stopResponse = Application->MessageBox("incompatible total number of points,\ncontinue fitting?",
"AAnalyzer -> Alert!", MB_YESNO);
    if( stopResponse == IDNO )
    {
        singularity = true;
        return;
    }
}
}
}

```

M. :CopyNewValuesToFreeParameters()

/*Subrutina encargada de copiar los valores de los parametros libres a ellos mismos*/

```

void TFirm3::CopyNewValuesToFreeParameters( float a[], int typeSpace )
{

```

```

int          i, k;
int          j = 0;

for( i=0; i<maxPeaks; i++ )
  if( peak[i].active )
    if( peak[i].areaVary.vary == free || peak[i].areaVary.vary == limited )
      for( k=0; k<maxFiles; k++ )
        if( Data[k].fit && Data[k].gotModifiedCurveAbs[typeSpace] )
          {
            j++;
            Data[k].peak[i].area = a[j];
          }
for( i=0; i<maxPeaks; i++ )
  if( peak[i].active )
    if( peak[i].energyVary.vary == free || peak[i].energyVary.vary == limited )
      {
        j++;
        peak[i].energy = a[j];
      }
for( i=0; i<maxPeaks; i++ )
  if( peak[i].active )
    {
      if( peak[i].gaussVary.vary == free || peak[i].gaussVary.vary == limited )
        {
          j++;
          peak[i].gauss = a[j];
        }
      else if( model[i].peak[j].gaussVary.vary == correlated &&
               strcmp(model[i].peak[j].gaussVary.correlation,"correlationSigmaOfRatio") = 0 )
        {
          j++;
          peak[i].factorSigma = a[j];
        }
    }
for( i=0; i<maxPeaks; i++ )
  if( peak[i].active )
    if( peak[i].branchingRatioVary.vary == free ||
        peak[i].branchingRatioVary.vary == limited )
      {
        j++;
        peak[i].branchingRatio = a[j];
      }

if( S0Vary.vary == free || S0Vary.vary == limited )
  {
    j++;
    So = a[j];
  }
if( Exp )
if( exponentVary.vary == free || exponentVary.vary == limited )
  {
    j++;
    exponent = a[j];
  }

```

```

CheckAllLimits();
CheckAllCorrelations();

```

```

}

```

N. GetAllGradientsKspace()

```

void TFirm3::GetAllGradientsKspace( float a[], int numV, int typeSpace )

```

```

{
int i, j;
int k, m;
int freeVarCounter = 0;
double y[maxFiles][maximumNumP];
double y2[maxFiles][maximumNumP];
double deltaArea;
double deltaEnergy;
double deltaGauss;
double deltaBranchingRatio;
double deltaExponent,deltafactorSigma;
double deltaS0;
for( k=0; k<maxFiles; k++ ) // initialize GradY
if( Data[k].fit && Data[k].gotModifiedCurveAbs[typeSpace] )
for( m=0; m < Data[k].FitCurveAbs[typeSpace].numP; m++ )
for( i=0; i < maxNumFreeVar; i++ )
Data[k].GradY[m][i] = 0;

CopyNewValuesToFreeParameters(a, typeSpace);
CheckAllLimits();
CheckAllCorrelations();
GetFitAndChisq(typeSpace);
if( singularity )
return; // make area different from zero
for( k=0; k<maxFiles; k++ )
if( Data[k].fit && Data[k].gotModifiedCurveAbs[typeSpace] )
for( m=0; m < Data[k].FitCurveAbs[typeSpace].numP; m++ )
y[k][m] = Data[k].FitCurveAbs[typeSpace].y[m];

for( i=0; i<maxPeaks; i++ )
if( peak[i].active )
if( peak[i].areaVary.vary == free || peak[i].areaVary.vary == limited )
for( j=0; j<maxFiles; j++ )
if( Data[j].fit && Data[j].gotModifiedCurveAbs[typeSpace] )
{
freeVarCounter++;
deltaArea = max( fabs( Data[j].peak[i].area/50 ), fabs( Data[j].totalArea/200) );
Data[j].peak[i].area += deltaArea;
CheckAllLimits();
CheckAllCorrelations();
GetFitAndChisq(typeSpace);
if( singularity )
return;
for( k=0; k < Data[j].FitCurveAbs[typeSpace].numP; k++ )
{

```

```

        y2[j][k] = Data[j].FitCurveAbs[typeSpace].y[k];
        Data[j].GradY[k][freeVarCounter-1] = ( y2[j][k] - y[j][k] ) / deltaArea;
    }
    Data[j].peak[i].area = a[freeVarCounter]; // a vote of confidence for the counting scheme
}
for( i=0; i<maxPeaks; i++ )
    if( peak[i].active )
        if( peak[i].energyVary.vary == free || peak[i].energyVary.vary == limited )
            {
                freeVarCounter++;
                deltaEnergy = 0.01;
                peak[i].energy += deltaEnergy;
                CheckAllLimits();
                CheckAllCorrelations();
                GetFitAndChisq(typeSpace);
                if( singularity )
                    return;
                for( j=0; j<maxFiles; j++ )
                    if( Data[j].fit && Data[j].gotModifiedCurveAbs[typeSpace] )
                        for( k=0; k < Data[j].FitCurveAbs[typeSpace].numP; k++ )
                            {
                                y2[j][k] = Data[j].FitCurveAbs[typeSpace].y[k];
                                Data[j].GradY[k][freeVarCounter-1] = ( y2[j][k] - y[j][k] ) / deltaEnergy;
                            }
                peak[i].energy = a[freeVarCounter];
            }
for( i=0; i<maxPeaks; i++ )
    if( peak[i].active )
        {
            if( peak[i].gaussVary.vary == free ||
                peak[i].gaussVary.vary == limited )
                {
                    freeVarCounter++;
                    if ( peak[i].gauss==0.1)
                        deltaGauss = -0.001;
                    else
                        deltaGauss = 0.001;
                    peak[i].gauss += deltaGauss;
                    CheckAllLimits();
                    CheckAllCorrelations();
                    GetFitAndChisq(typeSpace);
                    if( singularity )
                        return;
                    for( j=0; j<maxFiles; j++ )
                        if( Data[j].fit && Data[j].gotModifiedCurveAbs[typeSpace] )
                            for( k=0; k < Data[j].FitCurveAbs[typeSpace].numP; k++ )
                                {
                                    y2[j][k] = Data[j].FitCurveAbs[typeSpace].y[k];
                                    Data[j].GradY[k][freeVarCounter-1] = ( y2[j][k] - y[j][k] ) / deltaGauss;
                                }
                    peak[i].gauss = a[freeVarCounter];
                }
            // end of if ( gauss)
//Para el caso en que Sigma2 esté correlacionada con R2 de la forma SIGAM2 = wR2
            else if (peak[i].gaussVary.vary == correlated &&
                strcmp(peak[i].gaussVary.correlation,"correlationSigmaOfRatio") == 0 )

```

```

{
  freeVarCounter++;
  if ( peak[i].factorSigma == 0.002 )
    deltafactorSigma = -0.0001;
  else
    deltafactorSigma = 0.0001;
  peak[i].factorSigma += deltafactorSigma;
  CheckALLimits();
  CheckAllCorrelations();
  GetFitAndChisq(typeSpace);
  if( singularity )
    return;
  for( j=0; j<maxFiles; j++ )
    if( Data[j].fit && Data[j].gotModifiedCurveAbs[typeSpace] )
      for( k=0; k < Data[j].FitCurveAbs[typeSpace].numP; k++ )
        {
          y2[j][k] = Data[j].FitCurveAbs[typeSpace].y[k];
          Data[j].GradY[k][freeVarCounter-1] = ( y2[j][k] - y[j][k] ) / deltafactorSigma;
        }
  peak[i].factorSigma = a[freeVarCounter];
}
}
for( i=0; i<maxPeaks; i++ )
  if( peak[i].active )
    if( peak[i].branchingRatioVary.vary == free || peak[i].branchingRatioVary.vary == limited )
      {
        freeVarCounter++;
        deltaBranchingRatio = 0.005;
        peak[i].branchingRatio += deltaBranchingRatio;
        CheckALLimits();
        CheckAllCorrelations();
        GetFitAndChisq(typeSpace);
        if( singularity )
          return;
        for( j=0; j<maxFiles; j++ )
          if( Data[j].fit && Data[j].gotModifiedCurveAbs[typeSpace] )
            for( k=0; k < Data[j].FitCurveAbs[typeSpace].numP; k++ )
              {
                y2[j][k] = Data[j].FitCurveAbs[typeSpace].y[k];
                Data[j].GradY[k][freeVarCounter-1] = ( y2[j][k] - y[j][k] ) / deltaBranchingRatio;
              }
        peak[i].branchingRatio = a[freeVarCounter];
      }
}
if( S0Vary.vary == free || S0Vary.vary == limited )
  {
    freeVarCounter++;
    deltaS0 = 0.01;
    So = deltaS0;
    CheckALLimits();
    CheckAllCorrelations();
    GetFitAndChisq(typeSpace);
    if( singularity )
      return;
    for( j=0; j<maxFiles; j++ )
      if( Data[j].fit && Data[j].gotModifiedCurveAbs[typeSpace] )
        for( k=0; k < Data[j].FitCurveAbs[typeSpace].numP; k++ )

```



```

        {
            y2[j][k] = Data[j].FitCurveAbs(typeSpace).y[k];
            Data[j].GradY[k][freeVarCounter-1] = ( y2[j][k] - y[j][k] ) / deltaS0;
        }
        So = a[freeVarCounter];
    }
    CheckAllLimits();
    CheckAllCorrelations();
}

```

O. GetAllGradientsRspace()

```

void TFirm3::GetAllGradientsRspace( float a[], int numV, int typeSpace )
{
    int i, j;
    int k, m;
    int freeVarCounter = 0;
    double y[maxFiles][maximumNumP];
    double y2[maxFiles][maximumNumP];
    double deltaArea;
    double deltaEnergy;
    double deltaGauss,deltafactorSigma;
    double deltaBranchingRatio;
    double deltaExponent;
    double deltaS0;

    for( k=0; k<maxFiles; k++) // initialize GradY
        if( Data[k].fit && Data[k].gotModifiedCurveRe )
        {
            for( m=0; m < Data[k].FitCurveRe.numP; m++ )
                for( i=0; i < maxNumFreeVar; i++ )
                    Data[k].GradY[m][i] = 0;
            for( m=0; m < Data[k].FitCurveIm.numP; m++ )
                for( i=0; i < maxNumFreeVar; i++ )
                    Data[k].GradY[Data[k].FitCurveRe.numP + m][i] = 0;
        }
    CopyNewValuesToFreeParameters(a, typeSpace);
    CheckAllLimits();
    CheckAllCorrelations();
    GetFitAndChisq(typeSpace);
    if( singularity )
        return; // make area different from zero
    for( k=0; k<maxFiles; k++ )
        if( Data[k].fit && Data[k].gotModifiedCurveRe )
        {
            for( m=0; m < Data[k].FitCurveRe.numP; m++ )
                y[k][m] = Data[k].FitCurveRe.y[m];
            for( m=0; m < Data[k].FitCurveIm.numP; m++ )
                y[k][Data[k].FitCurveRe.numP + m] = Data[k].FitCurveIm.y[m];
        }
    for( i=0; i<maxPeaks; i++ )
        if( peak[i].active )
            if( peak[i].areaVary.vary == free || peak[i].areaVary.vary == limited )

```

```

for( j=0; j<maxFiles; j++ )
  if( Data[j].fit && Data[j].gotModifiedCurveRe )
  {
    freeVarCounter++;
    deltaArea = max( fabs( Data[j].peak[i].area/50 ), fabs( Data[j].totalArea/200 ) );
    Data[j].peak[i].area += deltaArea;
    CheckAllLimits();
    CheckAllCorrelations();
    GetFitAndChisq(typeSpace);
    if( singularity )
      return;

    for( k=0; k < Data[j].FitCurveRe.numP; k++ )
    {
      y2[j][k] = Data[j].FitCurveRe.y[k];
      Data[j].GradY[k][freeVarCounter-1] = ( y2[j][k] - y[j][k] ) / deltaArea;
    }
    for( k=0; k < Data[j].FitCurveIm.numP; k++ )
    {
      y2[j][Data[j].FitCurveRe.numP + k] = Data[j].FitCurveIm.y[k];
      Data[j].GradY[Data[j].FitCurveRe.numP + k][freeVarCounter-1] =
( y2[j][Data[j].FitCurveRe.numP + k] -
      y[j][Data[j].FitCurveRe.numP + k] ) / deltaArea;
    }
    Data[j].peak[i].area = a[freeVarCounter]; // a vote of confidence for the counting scheme
  }

for( i=0; i<maxPeaks; i++ )
  if( peak[i].active )
    if( peak[i].energyVary.vary == free || peak[i].energyVary.vary == limited )
    {
      freeVarCounter++;
      deltaEnergy = 0.01;
      peak[i].energy += deltaEnergy;
      CheckAllLimits();
      CheckAllCorrelations();
      GetFitAndChisq(typeSpace);
      if( singularity )
        return;
    }

for( j=0; j<maxFiles; j++ )
  if( Data[j].fit && Data[j].gotModifiedCurveRe )
  {
    for( k=0; k < Data[j].FitCurveRe.numP; k++ )
    {
      y2[j][k] = Data[j].FitCurveRe.y[k];
      Data[j].GradY[k][freeVarCounter-1] = ( y2[j][k] - y[j][k] ) / deltaEnergy;
    }
    for( k=0; k < Data[j].FitCurveIm.numP; k++ )
    {
      y2[j][Data[j].FitCurveRe.numP + k] = Data[j].FitCurveIm.y[k];
      Data[j].GradY[Data[j].FitCurveRe.numP + k][freeVarCounter-1] =
( y2[j][Data[j].FitCurveRe.numP + k] -
      y[j][Data[j].FitCurveRe.numP + k] ) / deltaEnergy;
    }
  }
}

```

```

    peak[i].energy = a[freeVarCounter];
}
for( i=0; i<maxPeaks; i++ )
    if( peak[i].active )
    {
        if( peak[i].gaussVary.vary == free ||
            peak[i].gaussVary.vary == limited )
        {
            freeVarCounter++;
            if ( peak[i].gauss==0.1)
                deltaGauss = -0.001;
            else
                deltaGauss = 0.001;
            peak[i].gauss += deltaGauss;
            CheckAllLimits();
            CheckAllCorrelations();
            GetFitAndChisq(typeSpace);
            if( singularity )
                return;
            for( j=0; j<maxFiles; j++ )
                if( Data[j].fit && Data[j].gotModifiedCurveRe )
                {
                    for( k=0; k < Data[j].FitCurveRe.numP; k++ )
                    {
                        y2[j][k] = Data[j].FitCurveRe.y[k];
                        Data[j].GradY[k][freeVarCounter-1] = ( y2[j][k] - y[j][k] ) / deltaGauss;
                    }
                    for( k=0; k < Data[j].FitCurveIm.numP; k++ )
                    {
                        y2[j][Data[j].FitCurveRe.numP + k] = Data[j].FitCurveIm.y[k];
                        Data[j].GradY[Data[j].FitCurveRe.numP + k][freeVarCounter-1] =
                            ( y2[j][Data[j].FitCurveRe.numP + k] -
                              y[j][Data[j].FitCurveRe.numP + k] ) / deltaGauss;
                    }
                }
            peak[i].gauss = a[freeVarCounter];
        }
    }
else if (peak[i].gaussVary.vary == correlated &&
        strcmp(peak[i].gaussVary.correlation,"correlationSigmaOfRatio") == 0 )
{
    freeVarCounter++;
    if ( peak[i].factorSigma == 0.002 )
        deltafactorSigma = -0.0001;
    else
        deltafactorSigma = 0.0001;
    peak[i].factorSigma += deltafactorSigma;
    CheckAllLimits();
    CheckAllCorrelations();
    GetFitAndChisq(typeSpace);
    if( singularity )
        return;

    for( j=0; j<maxFiles; j++ )
        if( Data[j].fit && Data[j].gotModifiedCurveRe )
        {
            for( k=0; k < Data[j].FitCurveRe.numP; k++ )

```

```

    {
        y2[j][k] = Data[j].FitCurveRe.y[k];
        Data[j].GradY[k][freeVarCounter-1] = ( y2[j][k] - y[j][k] ) / deltafactorSigma;
    }
    for( k=0; k < Data[j].FitCurveIm.numP; k++ )
    {
        y2[j][Data[j].FitCurveRe.numP + k] = Data[j].FitCurveIm.y[k];
        Data[j].GradY[Data[j].FitCurveRe.numP + k][freeVarCounter-1] =
        ( y2[j][Data[j].FitCurveRe.numP + k] -
          y[j][Data[j].FitCurveRe.numP + k] ) / deltafactorSigma;
    }
    peak[i].factorSigma = a[freeVarCounter];
}
}
for( i=0; i<maxPeaks; i++ )
    if( peak[i].active )
        if( peak[i].branchingRatioVary.vary == free || peak[i].branchingRatioVary.vary == limited )
        {
            freeVarCounter++;
            deltaBranchingRatio = 0.005;
            peak[i].branchingRatio += deltaBranchingRatio;
            CheckAllLimits();
            CheckAllCorrelations();
            GetFitAndChisq(typeSpace);
            if( singularity )
                return;
            for( j=0; j<maxFiles; j++ )
                if( Data[j].fit && Data[j].gotModifiedCurveRe )
                {
                    for( k=0; k < Data[j].FitCurveRe.numP; k++ )
                    {
                        y2[j][k] = Data[j].FitCurveRe.y[k];
                        Data[j].GradY[k][freeVarCounter-1] = ( y2[j][k] - y[j][k] ) / deltaBranchingRatio;
                    }
                    for( k=0; k < Data[j].FitCurveIm.numP; k++ )
                    {
                        y2[j][Data[j].FitCurveRe.numP + k] = Data[j].FitCurveIm.y[k];
                        Data[j].GradY[Data[j].FitCurveRe.numP + k][freeVarCounter-1] =
                        ( y2[j][Data[j].FitCurveRe.numP + k] -
                          y[j][Data[j].FitCurveRe.numP + k] ) / deltaBranchingRatio;
                    }
                }
            peak[i].branchingRatio = a[freeVarCounter];
        }
}
if( S0Vary.vary == free || S0Vary.vary == limited )
{
    freeVarCounter++;
    if ( So == 1.0 )
        deltaS0 = -0.01;
    else
        deltaS0 = 0.01;
    So += deltaS0;
    CheckAllLimits();
    CheckAllCorrelations();
    GetFitAndChisq(typeSpace);
}

```

```

if( singularity )
    return;
for( j=0; j<maxFiles; j++ )
    if( Data[j].fit && Data[j].gotModifiedCurveRe )
        {
        for( k=0; k < Data[j].FitCurveRe.numP; k++ )
            {
            y2[j][k] = Data[j].FitCurveRe.y[k];
            Data[j].GradY[k][freeVarCounter-1] = ( y2[j][k] - y[j][k] ) / deltaS0;
            }
        for( k=0; k < Data[j].FitCurveIm.numP; k++ )
            {
            y2[j][Data[j].FitCurveRe.numP + k] = Data[j].FitCurveIm.y[k];
            Data[j].GradY[Data[j].FitCurveRe.numP + k][freeVarCounter-1] =
            ( y2[j][Data[j].FitCurveRe.numP + k] -
            y[j][Data[j].FitCurveRe.numP + k] ) / deltaS0;
            }
        }
    So = a[freeVarCounter];
}
CheckAllLimits();
CheckAllCorrelations();
}

```

P. GetChiSqModels()

```

void TFirm3::GetChiSqModels( float p[], int typeSpace )
{
    int                i,k;
    int                j = 0;
    for( i=0; i<maxModels; i++ )
        if( model[i].modelActive )
            if( model[i].fractionVary.vary == free || model[i].fractionVary.vary == limited )
                {
                for( k=0; k<maxFiles; k++ )
                    if( Data[k].fit && Data[k].gotModifiedCurveAbs[typeSpace] )
                        {
                        j++;
                        Data[k].fractionModel[i] = p[j];
                        }
                }
    for( k=0; k<maxModels; k++ )
        if( model[k].modelActive )
            {
            for( i=0; i<maxPeaks; i++ )
                if( model[k].peak[i].active )
                    {
                    if( model[k].peak[i].energyVary.vary == free || model[k].peak[i].energyVary.vary == limited )
                        {
                        j++;
                        model[k].peak[i].energy = p[j];
                        }
                    }
            }
}

```

```

    if( model[k].peak[i].gaussVary.vary == free || model[k].peak[i].gaussVary.vary == limited )
    {
        j++;
        model[k].peak[i].gauss = p[j];
    }
    else if (model[k].peak[i].gaussVary.vary == correlated &&
             strcmp(model[k].peak[i].gaussVary.correlation,"correlationSigmaOfRatio") == 0 )
    {
        j++;
        model[k].peak[i].factorSigma = p[j];
    }
    if( model[k].peak[i].branchingRatioVary.vary == free ||
        model[k].peak[i].branchingRatioVary.vary == limited )
    {
        j++;
        model[k].peak[i].branchingRatio = p[j];
    }
}
switch (k)
{
case model4AsV:
    if( model[k].alphaAsVary.vary == free || model[k].alphaAsVary.vary == limited )
    {
        j++;
        model[k].alphaAs = p[j];
    }
    if( model[k].betaSiVary.vary == free || model[k].betaSiVary.vary == limited )
    {
        j++;
        model[k].betaSi = p[j];
    }
    if( model[k].S0Vary.vary == free || model[k].S0Vary.vary == limited )
    {
        j++;
        model[k].S0 = p[j];
    }
    break;
case model3AsVSi:
    if( model[k].alphaAsVary.vary == free || model[k].alphaAsVary.vary == limited )
    {
        j++;
        model[k].alphaAs = p[j];
    }
    if( model[k].alphaSiVary.vary == free || model[k].alphaSiVary.vary == limited )
    {
        j++;
        model[k].alphaSi = p[j];
    }
    if( model[k].betaSiVary.vary == free || model[k].betaSiVary.vary == limited )
    {
        j++;
        model[k].betaSi = p[j];
    }
    if( model[k].S0Vary.vary == free || model[k].S0Vary.vary == limited )
    {

```

```

        j++;
        model[k].S0 = p[j];
    }
    break;
case model2AsV2Si:
    if( model[k].alphaAsVary.vary == free || model[k].alphaAsVary.vary == limited )
    {
        j++;
        model[k].alphaAs = p[j];
    }
    if( model[k].alphaSiVary.vary == free || model[k].alphaSiVary.vary == limited )
    {
        j++;
        model[k].alphaSi = p[j];
    }
    if( model[k].betaSiVary.vary == free || model[k].betaSiVary.vary == limited )
    {
        j++;
        model[k].betaSi = p[j];
    }
    if( model[k].S0Vary.vary == free || model[k].S0Vary.vary == limited )
    {
        j++;
        model[k].S0 = p[j];
    }
    break;
case modelAs4Si:
    if( model[k].alphaSiVary.vary == free || model[k].alphaSiVary.vary == limited )
    {
        j++;
        model[k].alphaSi = p[j];
    }
    if( model[k].betaSiVary.vary == free || model[k].betaSiVary.vary == limited )
    {
        j++;
        model[k].betaSi = p[j];
    }
    if( model[k].S0Vary.vary == free || model[k].S0Vary.vary == limited )
    {
        j++;
        model[k].S0 = p[j];
    }
} //end of switch
} //end of if()
}

```

Q. ChiSqModels()

```

float TFirm3::ChiSqModels( float p[], int typeSpace )
{
    float chisq=0, chisq2=0;

    GetChiSqModels( p, typeSpace );
}

```

```

checkAllModelsLimitsAndCorrelations(typeSpace);

    chisq2 = GetFitAndChisqModels(typeSpace);
    if( singularity )
        return chisq;
    else
        chisq = chisq2;

return chisq;
}

```

R. GradChiSqModels()

```

void TFirm3::GradChiSqModels( float p[], int numV, int typeSpace )
{
    switch (typeSpace)
    {
        case spaceK:
            GradChiSqModelsAbs( p, numV, typeSpace );
            break;
        case spaceR:
            GradChiSqModelsReAndIm( p, numV, typeSpace );
    }
}

```

S. GradChiSqModelsAbs()

```

void TFirm3::GradChiSqModelsAbs( float p[], int numV, int typeSpace )
{
#define min(a, b) (((a) < (b)) ? (a) : (b))
#define max(a, b) (((a) > (b)) ? (a) : (b))
    int          i,k, m, join;
    int          j= 0;
    int          freeVarCounter = 0;
    double       y[maxFiles][maximumNumP];
    double       y2[maxFiles][maximumNumP];
    double       deltaEnergy;
    double       deltaGauss,deltaFactorSigma;
    double       deltaBranchingRatio;
    double       deltaExponent;
    double       deltaAlphaAs;
    double       deltaAlphaSi;
    double       deltaBetaSi;
    double       deltaFRACTION;
    double       deltaS0;

    for( k=0; k<maxFiles; k++ ) // initialize GradY
        if( Data[k].fit && Data[k].gotModifiedCurveAbs[typeSpace] )

```



```

    for( m=0; m < Data[k].FitCurveAbs[typeSpace].numP; m++ )
        for( i=0; i < maxNumFreeVar; i++ )
            Data[k].GradY[m][i] = 0;
    GetChiSqModels( p, typeSpace); //GetChiSqModels(a, typeSpace);
    checkAllModelsLimitsAndCorrelations(typeSpace);
    GetFitAndChisqModels(typeSpace);
    if( singularity )
        return;
    for( k=0; k<maxFiles; k++ )
        if( Data[k].fit && Data[k].gotModifiedCurveAbs[typeSpace] )
            for( m=0; m < Data[k].FitCurveAbs[typeSpace].numP; m++ )
                y[k][m] = Data[k].FitCurveAbs[typeSpace].y[m];
    for( i=0; i<maxModels; i++ )
        if( model[i].modelActive )
            if( model[i].fractionVary.vary == free || model[i].fractionVary.vary == limited )
                {
                    for( j=0; j<maxFiles; j++ )
                        if( Data[j].fit && Data[j].gotModifiedCurveAbs[typeSpace] )
                            {
                                freeVarCounter++;
                                if ( Data[j].fractionModel[i] <= 0.97 )
                                    deltaFRACTION = 0.01;
                                else if ( Data[j].fractionModel[i] > 0.97 )
                                    deltaFRACTION = -0.01;
                                Data[j].fractionModel[i] += deltaFRACTION;
                                checkAllModelsLimitsAndCorrelations(typeSpace);

                                GetFitAndChisqModels(typeSpace);
                                if( singularity )
                                    return;
                                for( k=0; k < Data[j].FitCurveAbs[typeSpace].numP; k++ )
                                    {
                                        y2[j][k] = Data[j].FitCurveAbs[typeSpace].y[k];
                                        Data[j].GradY[k][freeVarCounter-1] = ( y2[j][k] - y[j][k] ) / deltaFRACTION;
                                    }
                                model[i].fraction = Data[j].fractionModel[i];
                                Data[j].fractionModel[i] = p[freeVarCounter]; // a vote of confidence for the counting scheme
                            }
                }
    }
    for( k=0; k<maxModels; k++ )
        if( model[k].modelActive )
            {
                for( i=0; i<maxPeaks; i++ )
                    if( model[k].peak[i].active )
                        {
                            if( model[k].peak[i].energyVary.vary == free ||
                                model[k].peak[i].energyVary.vary == limited )
                                {
                                    freeVarCounter++;
                                    if(model[k].peak[i].energy >= 0)
                                        deltaEnergy = -0.01;
                                    else if(model[k].peak[i].energy < 0 )
                                        deltaEnergy = 0.01;
                                    model[k].peak[i].energy += deltaEnergy;
                                    checkAllModelsLimitsAndCorrelations(typeSpace);
                                    GetFitAndChisqModels(typeSpace);
                                }
                        }
            }

```

```

if( singularity )
    return;
for( j=0; j<maxFiles; j++ )
    if( Data[j].fit && Data[j].gotModifiedCurveAbs[typeSpace] )
        for( join=0; join < Data[j].FitCurveAbs[typeSpace].numP; join++ )
            {
                y2[j][join] = Data[j].FitCurveAbs[typeSpace].y[join];
                Data[j].GradY[join][freeVarCounter-1] = ( y2[j][join] - y[j][join] ) / deltaEnergy;
            }
model[k].peak[i].energy = p[freeVarCounter];
}
if( model[k].peak[i].gaussVary.vary == free ||
    model[k].peak[i].gaussVary.vary == limited )
{
    freeVarCounter++;
    if( model[k].peak[i].gauss == 0.1 )
        deltaGauss = -0.001;
    else
        deltaGauss = 0.001;
    model[k].peak[i].gauss += deltaGauss;
    checkAllModelsLimitsAndCorrelations(typeSpace);
    GetFitAndChisqModels(typeSpace);
    if( singularity )
        return;
    for( j=0; j<maxFiles; j++ )
        if( Data[j].fit && Data[j].gotModifiedCurveAbs[typeSpace] )
            for( join=0; join < Data[j].FitCurveAbs[typeSpace].numP; join++ )
                {
                    y2[j][join] = Data[j].FitCurveAbs[typeSpace].y[join];
                    Data[j].GradY[join][freeVarCounter-1] = ( y2[j][join] - y[j][join] ) / deltaGauss;
                }
    model[k].peak[i].gauss = p[freeVarCounter];
}
else if( model[k].peak[i].gaussVary.vary == correlated &&
    strcmp(model[k].peak[i].gaussVary.correlation,"correlationSigmaOfRatio") == 0 )
{
    freeVarCounter++;
    if( model[k].peak[i].factorSigma == 0.002 )
        deltaFactorSigma = -0.0001;
    else
        deltaFactorSigma = 0.0001;
    model[k].peak[i].factorSigma += deltaFactorSigma;
    checkAllModelsLimitsAndCorrelations(typeSpace);
    GetFitAndChisqModels(typeSpace);
    if( singularity )
        return;
    for( j=0; j<maxFiles; j++ )
        if( Data[j].fit && Data[j].gotModifiedCurveAbs[typeSpace] )
            for( join=0; join < Data[j].FitCurveAbs[typeSpace].numP; join++ )
                {
                    y2[j][join] = Data[j].FitCurveAbs[typeSpace].y[join];
                    Data[j].GradY[join][freeVarCounter-1] = ( y2[j][join] - y[j][join] ) / deltaFactorSigma;
                }
    model[k].peak[i].factorSigma = p[freeVarCounter];
}
}

```

```

        if( model[k].peak[i].branchingRatioVary.vary == free ||
            model[k].peak[i].branchingRatioVary.vary == limited )
    {
        freeVarCounter++;
        deltaBranchingRatio = 0.01;
        model[k].peak[i].branchingRatio += deltaBranchingRatio;
        checkAllModelsLimitsAndCorrelations(typeSpace);
        GetFitAndChisqModels(typeSpace);
        if( singularity )
            return;
        for( j=0; j<maxFiles; j++ )
            if( Data[j].fit && Data[j].gotModifiedCurveAbs[typeSpace] )
                for( join=0; join < Data[j].FitCurveAbs[typeSpace].numP; join++ )
                    {
                        y2[j][join] = Data[j].FitCurveAbs[typeSpace].y[join];
                        Data[j].GradY[join][freeVarCounter-1] = ( y2[j][join] - y[j][join] ) /
deltaBranchingRatio;
                    }
                model[k].peak[i].branchingRatio = p[freeVarCounter];
            }
        } //end of if()
    switch (k)
    {
    case model4AsV:
        if( model[k].alphaAsVary.vary == free ||
            model[k].alphaAsVary.vary == limited )
        {
            freeVarCounter++;
            if( model[k].alphaAs == -0.10 )
                deltaAlphaAs = 0.02;
            else if( model[k].alphaAs == 0.10 )
                deltaAlphaAs = -0.02;
            else
                deltaAlphaAs = min(0.005, max(0.001, fabs( model[k].alphaAs ) / 10 ));
            model[k].alphaAs += deltaAlphaAs;
            checkAllModelsLimitsAndCorrelations(typeSpace);
            GetFitAndChisqModels(typeSpace);
            if( singularity )
                return;
            for( j=0; j<maxFiles; j++ )
                if( Data[j].fit && Data[j].gotModifiedCurveAbs[typeSpace] )
                    for( join=0; join < Data[j].FitCurveAbs[typeSpace].numP; join++ )
                        {
                            y2[j][join] = Data[j].FitCurveAbs[typeSpace].y[join];
                            Data[j].GradY[join][freeVarCounter-1] = ( y2[j][join] - y[j][join] ) / deltaAlphaAs;
                        }
                    model[k].alphaAs = p[freeVarCounter];
                }
            }
        if( model[k].betaSiVary.vary == free ||
            model[k].betaSiVary.vary == limited )
        {
            freeVarCounter++;
            if( model[k].betaSi == -0.10 )
                deltaBetaSi = 0.02;
            else if( model[k].betaSi == 0.10 )

```

```

        deltaBetaSi = -0.02;
    else
        deltaBetaSi = min(0.005, max(0.001, fabs( model[k].betaSi) / 10 ));
        model[k].betaSi += deltaBetaSi;
        checkAllModelsLimitsAndCorrelations(typeSpace);
        GetFitAndChisqModels(typeSpace);
        if( singularity )
            return;
    for( j=0; j<maxFiles; j++ )
        if( Data[j].fit && Data[j].gotModifiedCurveAbs[typeSpace] )
            for( join=0; join < Data[j].FitCurveAbs[typeSpace].numP; join++ )
                {
                    y2[j][join] = Data[j].FitCurveAbs[typeSpace].y[join];
                    Data[j].GradY[join][freeVarCounter-1] = ( y2[j][join] - y[j][join] ) / deltaBetaSi;
                }
    model[k].betaSi = p[freeVarCounter];
}
if( model[k].S0Vary.vary == free ||
    model[k].S0Vary.vary == limited )
{
    freeVarCounter++;
    if( model[k].S0 == 0.50 )
        deltaS0 = 0.02;
    else if( model[k].S0 == 1 )
        deltaS0 = -0.02;
    else
        deltaS0 = 0.01;
    model[k].S0 += deltaS0;
    checkAllModelsLimitsAndCorrelations(typeSpace);
    GetFitAndChisqModels(typeSpace);
    if( singularity )
        return;
    for( j=0; j<maxFiles; j++ )
        if( Data[j].fit && Data[j].gotModifiedCurveAbs[typeSpace] )
            for( join=0; join < Data[j].FitCurveAbs[typeSpace].numP; join++ )
                {
                    y2[j][join] = Data[j].FitCurveAbs[typeSpace].y[join];
                    Data[j].GradY[join][freeVarCounter-1] = ( y2[j][join] - y[j][join] ) / deltaS0;
                }
    model[k].S0 = p[freeVarCounter];
}
break;
case model3AsVSi:
    if( model[k].alphaAsVary.vary == free ||
        model[k].alphaAsVary.vary == limited )
    {
        freeVarCounter++;
        if( model[k].alphaAs == -0.10 )
            deltaAlphaAs = 0.02;
        else if( model[k].alphaAs == 0.10 )
            deltaAlphaAs = -0.02;
        else
            deltaAlphaAs = min(0.005, max(0.001, fabs( model[k].alphaAs) / 10 ));
        model[k].alphaAs += deltaAlphaAs;
        checkAllModelsLimitsAndCorrelations(typeSpace);
        GetFitAndChisqModels(typeSpace);
    }
}

```

```

    if( singularity )
        return;
    for( j=0; j<maxFiles; j++ )
        if( Data[j].fit && Data[j].gotModifiedCurveAbs[typeSpace] )
            for( join=0; join < Data[j].FitCurveAbs[typeSpace].numP; join++ )
                {
                    y2[j][join] = Data[j].FitCurveAbs[typeSpace].y[jjoin];
                    Data[j].GradY[join][freeVarCounter-1] = ( y2[j][join] - y[j][join] ) / deltaAlphaAs;
                }
    model[k].alphaAs = p[freeVarCounter];
}
if( model[k].alphaSiVary.vary == free ||
    model[k].alphaSiVary.vary == limited )
{
    freeVarCounter++;
    if( model[k].alphaSi == -0.10 )
        deltaAlphaSi = 0.02;
    else if( model[k].alphaSi == 0.10 )
        deltaAlphaSi = -0.02;
    else
        deltaAlphaSi = min(0.005, max(0.001, fabs( model[k].alphaSi) / 10 ));
    model[k].alphaSi += deltaAlphaSi;
    checkAllModelsLimitsAndCorrelations(typeSpace);
    GetFitAndChisqModels(typeSpace);
    if( singularity )
        return;
    for( j=0; j<maxFiles; j++ )
        if( Data[j].fit && Data[j].gotModifiedCurveAbs[typeSpace] )
            for( join=0; join < Data[j].FitCurveAbs[typeSpace].numP; join++ )
                {
                    y2[j][join] = Data[j].FitCurveAbs[typeSpace].y[jjoin];
                    Data[j].GradY[join][freeVarCounter-1] = ( y2[j][join] - y[j][join] ) / deltaAlphaSi;
                }
    model[k].alphaSi = p[freeVarCounter];
}
if( model[k].betaSiVary.vary == free ||
    model[k].betaSiVary.vary == limited )
{
    freeVarCounter++;
    if( model[k].betaSi == -0.10 )
        deltaBetaSi = 0.02;
    else if( model[k].betaSi == 0.10 )
        deltaBetaSi = -0.02;
    else
        deltaBetaSi = min(0.005, max(0.001, fabs( model[k].betaSi) / 10 ));
    model[k].betaSi += deltaBetaSi;
    checkAllModelsLimitsAndCorrelations(typeSpace);
    GetFitAndChisqModels(typeSpace);
    if( singularity )
        return;
    for( j=0; j<maxFiles; j++ )
        if( Data[j].fit && Data[j].gotModifiedCurveAbs[typeSpace] )
            for( join=0; join < Data[j].FitCurveAbs[typeSpace].numP; join++ )
                {
                    y2[j][join] = Data[j].FitCurveAbs[typeSpace].y[jjoin];
                    Data[j].GradY[join][freeVarCounter-1] = ( y2[j][join] - y[j][join] ) / deltaBetaSi;
                }
}

```

```

    }
    model[k].betaSi = p[freeVarCounter];
  }
  if( model[k].S0Vary.vary == free ||
    model[k].S0Vary.vary == limited )
  {
    freeVarCounter++;
    if( model[k].S0 == 0.50 )
      deltaS0 = 0.02;
    else if( model[k].S0 == 1 )
      deltaS0 = -0.02;
    else
      deltaS0 = 0.01;
    model[k].S0 += deltaS0;
    checkAllModelsLimitsAndCorrelations(typeSpace);
    GetFitAndChisqModels(typeSpace);
    if( singularity )
      return;
  }
  for( j=0; j<maxFiles; j++ )
    if( Data[j].fit && Data[j].gotModifiedCurveAbs[typeSpace] )
      for( join=0; join < Data[j].FitCurveAbs[typeSpace].numP; join++ )
        {
          y2[j][join] = Data[j].FitCurveAbs[typeSpace].y[join];
          Data[j].GradY[join][freeVarCounter-1] = ( y2[j][join] - y[j][join] ) / deltaS0;
        }
    model[k].S0 = p[freeVarCounter];
  }
  break;
case model2AsV2Si:
  if( model[k].alphaAsVary.vary == free ||
    model[k].alphaAsVary.vary == limited )
  {
    freeVarCounter++;
    if( model[k].alphaAs == -0.10 )
      deltaAlphaAs = 0.02;
    else if( model[k].alphaAs == 0.10 )
      deltaAlphaAs = -0.02;
    else
      deltaAlphaAs = min(0.005, max(0.001, fabs( model[k].alphaAs ) / 10 ));
    model[k].alphaAs += deltaAlphaAs;
    checkAllModelsLimitsAndCorrelations(typeSpace);
    GetFitAndChisqModels(typeSpace);
    if( singularity )
      return;
  }
  for( j=0; j<maxFiles; j++ )
    if( Data[j].fit && Data[j].gotModifiedCurveAbs[typeSpace] )
      for( join=0; join < Data[j].FitCurveAbs[typeSpace].numP; join++ )
        {
          y2[j][join] = Data[j].FitCurveAbs[typeSpace].y[join];
          Data[j].GradY[join][freeVarCounter-1] = ( y2[j][join] - y[j][join] ) / deltaAlphaAs;
        }
    model[k].alphaAs = p[freeVarCounter];
  }
  if( model[k].alphaSiVary.vary == free ||
    model[k].alphaSiVary.vary == limited )
  {

```

```

freeVarCounter++;
if( model[k].alphaSi == -0.10 )
    deltaAlphaSi = 0.02;
else if( model[k].alphaSi == 0.10 )
    deltaAlphaSi = -0.02;
else
    deltaAlphaSi = min(0.005, max(0.001, fabs( model[k].alphaSi) / 10 ));
model[k].alphaSi += deltaAlphaSi;
checkAllModelsLimitsAndCorrelations(typeSpace);
GetFitAndChisqModels(typeSpace);
if( singularity )
    return;
for( j=0; j<maxFiles; j++ )
    if( Data[j].fit && Data[j].gotModifiedCurveAbs[typeSpace] )
        for( join=0; join < Data[j].FitCurveAbs[typeSpace].numP; join++ )
            {
                y2[j][join] = Data[j].FitCurveAbs[typeSpace].y[jjoin];
                Data[j].GradY[jjoin][freeVarCounter-1] = ( y2[j][join] - y[j][join] ) / deltaAlphaSi;
            }
model[k].alphaSi = p[freeVarCounter];
}
if( model[k].betaSiVary.vary == free || model[k].betaSiVary.vary == limited )
{
    freeVarCounter++;
    if( model[k].betaSi == -0.10 )
        deltaBetaSi = 0.02;
    else if( model[k].betaSi == 0.10 )
        deltaBetaSi = -0.02;
    else
        deltaBetaSi = min(0.005, max(0.001, fabs( model[k].betaSi) / 10 ));
    model[k].betaSi += deltaBetaSi;
    checkAllModelsLimitsAndCorrelations(typeSpace);
    GetFitAndChisqModels(typeSpace);
    if( singularity )
        return;
    for( j=0; j<maxFiles; j++ )
        if( Data[j].fit && Data[j].gotModifiedCurveAbs[typeSpace] )
            for( join=0; join < Data[j].FitCurveAbs[typeSpace].numP; join++ )
                {
                    y2[j][join] = Data[j].FitCurveAbs[typeSpace].y[jjoin];
                    Data[j].GradY[jjoin][freeVarCounter-1] = ( y2[j][join] - y[j][join] ) / deltaBetaSi;
                }
    model[k].betaSi = p[freeVarCounter];
}
if( model[k].S0Vary.vary == free ||
    model[k].S0Vary.vary == limited )
{
    freeVarCounter++;
    if( model[k].S0 == 0.50 )
        deltaS0 = 0.02;
    else if( model[k].S0 == 1 )
        deltaS0 = -0.02;
    else
        deltaS0 = 0.01;
    model[k].S0 += deltaS0;
    checkAllModelsLimitsAndCorrelations(typeSpace);
}

```

```

GetFitAndChisqModels(typeSpace);
if( singularity )
    return;
for( j=0; j<maxFiles; j++ )
    if( Data[j].fit && Data[j].gotModifiedCurveAbs[typeSpace] )
        for( join=0; join < Data[j].FitCurveAbs[typeSpace].numP; join++ )
            {
                y2[j][join] = Data[j].FitCurveAbs[typeSpace].y[join];
                Data[j].GradY[join][freeVarCounter-1] = ( y2[j][join] - y[j][join] ) / deltaS0;
            }
model[k].S0 = p[freeVarCounter];
}
break;
case modelAs4Si:
if( model[k].alphaSiVary.vary == free || model[k].alphaSiVary.vary == limited )
{
    freeVarCounter++;
    if( model[k].alphaSi == -0.10 )
        deltaAlphaSi = 0.02;
    else if( model[k].alphaSi == 0.10 )
        deltaAlphaSi = -0.02;
    else
        deltaAlphaSi = min(0.005, max(0.001, fabs( model[k].alphaSi ) / 10 ));
    model[k].alphaSi += deltaAlphaSi;
    checkAllModelsLimitsAndCorrelations(typeSpace);
    GetFitAndChisqModels(typeSpace);
    if( singularity )
        return;
for( j=0; j<maxFiles; j++ )
    if( Data[j].fit && Data[j].gotModifiedCurveAbs[typeSpace] )
        for( join=0; join < Data[j].FitCurveAbs[typeSpace].numP; join++ )
            {
                y2[j][join] = Data[j].FitCurveAbs[typeSpace].y[join];
                Data[j].GradY[join][freeVarCounter-1] = ( y2[j][join] - y[j][join] ) / deltaAlphaSi;
            }
model[k].alphaSi = p[freeVarCounter];
}
if( model[k].betaSiVary.vary == free || model[k].betaSiVary.vary == limited )
{
    freeVarCounter++;
    if( model[k].betaSi == -0.10 )
        deltaBetaSi = 0.02;
    else if( model[k].betaSi == 0.10 )
        deltaBetaSi = -0.02;
    else
        deltaBetaSi = min(0.005, max(0.001, fabs( model[k].betaSi ) / 10 ));
    model[k].betaSi += deltaBetaSi;
    checkAllModelsLimitsAndCorrelations(typeSpace);
    GetFitAndChisqModels(typeSpace);
    if( singularity )
        return;
for( j=0; j<maxFiles; j++ )
    if( Data[j].fit && Data[j].gotModifiedCurveAbs[typeSpace] )
        for( join=0; join < Data[j].FitCurveAbs[typeSpace].numP; join++ )
            {
                y2[j][join] = Data[j].FitCurveAbs[typeSpace].y[join];
            }
}

```



```

        Data[j].GradY[join][freeVarCounter-1] = ( y2[j][join] - y[j][join] ) / deltaBetaSi;
    }
    model[k].betaSi = p[freeVarCounter];
}
    if( model[k].S0Vary.vary == free ||
        model[k].S0Vary.vary == limited )
{
    freeVarCounter++;
    if( model[k].S0 == 0.50 )
        deltaS0 = 0.02;
    else if( model[k].S0 == 1 )
        deltaS0 = -0.02;
    else
        deltaS0 = 0.01;
    model[k].S0 -= deltaS0;
    checkAllModelsLimitsAndCorrelations(typeSpace);
    GetFitAndChisqModels(typeSpace);
    if( singularity )
        return;
    for( j=0; j<maxFiles; j++ )
        if( Data[j].fit && Data[j].gotModifiedCurveAbs[typeSpace] )
            for( join=0; join < Data[j].FitCurveAbs[typeSpace].numP; join++ )
                {
                    y2[j][join] = Data[j].FitCurveAbs[typeSpace].y[join];
                    Data[j].GradY[join][freeVarCounter-1] = ( y2[j][join] - y[j][join] ) / deltaS0;
                }
    model[k].S0 = p[freeVarCounter];
}
} // end of switch (k)
} //end of if (model[k].modelActive )
checkAllModelsLimitsAndCorrelations(typeSpace);
// CheckAllLimits();
// CheckAllCorrelations();
#undef min
#undef max
}

```

T. GradChiSqModelsReAndIm()

```

void TFirm3::GradChiSqModelsReAndIm( float p[], int numV, int typeSpace )
{
#define min(a, b) (((a) < (b)) ? (a) : (b))
#define max(a, b) (((a) > (b)) ? (a) : (b))
    int i, k, m, join;
    int j= 0;
    int freeVarCounter = 0;
    double y[maxFiles][maximumNumP];
    double y2[maxFiles][maximumNumP];
    double deltaEnergy;
    double deltaGauss,deltaFactorSigma;
    double deltaBranchingRatio;
    double deltaExponent;

```

```

double          deltaAlphaAs;
double          deltaAlphaSi;
double          deltaBetaSi;
double          deltaFRACTION;
double          deltaS0;
int             numP1,numP2;

for( k=0; k<maxFiles; k++ )    // initialize GradY
  if( Data[k].fit && Data[k].gotModifiedCurveRe && Data[k].gotModifiedCurveIm )
  {
    numP1 = Data[k].FitCurveRe.numP;
    numP2 = Data[k].FitCurveIm.numP;
    for( m=0; m < numP1 + numP2; m++ )
      for( i=0; i < maxNumFreeVar; i++ )
        Data[k].GradY[m][i] = 0;
  }
GetChiSqModels( p, typeSpace);
checkAllModelsLimitsAndCorrelations(typeSpace);
GetFitAndChiSqModels(typeSpace);
if( singularity )
  return;
for( k=0; k<maxFiles; k++ )
  if( Data[k].fit && Data[k].gotModifiedCurveRe && Data[k].gotModifiedCurveIm )
  {
    numP1 = Data[k].FitCurveRe.numP;
    numP2 = Data[k].FitCurveIm.numP;
    for( m=0; m < numP1; m++ )
      y[k][m] = Data[k].FitCurveRe.y[m];
    for( m=0; m < numP2; m++ )
      y[k][numP1+m] = Data[k].FitCurveIm.y[m];
  }
for( i=0; i<maxModels; i++ )
  if( model[i].modelActive )
    if( model[i].fractionVary.vary == free || model[i].fractionVary.vary == limited )
    {
      for( j=0; j<maxFiles; j++ )
        if( Data[j].fit && Data[j].gotModifiedCurveRe && Data[j].gotModifiedCurveIm )
        {
          freeVarCounter++;
          if ( Data[j].fractionModel[i] <= 0.97 )
            deltaFRACTION = 0.02;
          else if ( Data[j].fractionModel[i] > 0.97 )
            deltaFRACTION = -0.02;
          Data[j].fractionModel[i] += deltaFRACTION;
          checkAllModelsLimitsAndCorrelations(typeSpace);
          GetFitAndChiSqModels(typeSpace);
          if( singularity )
            return;
          numP1 = Data[j].FitCurveRe.numP;
          for( k=0; k < numP1; k++ )
          {
            y2[j][k] = Data[j].FitCurveRe.y[k];
            Data[j].GradY[k][freeVarCounter-1] = ( y2[j][k] - y[j][k] ) / deltaFRACTION;
          }
          numP2 = Data[j].FitCurveIm.numP;
          for( k=0; k < numP2; k++ )

```

```

        {
            y2[j][numP1+k] = Data[j].FitCurveIm.y[k];
            Data[j].GradY[numP1+k][freeVarCounter-1] = ( y2[j][numP1+k] - y[j][numP1+k] ) /
                                                    deltaFRACTION;
        }
        model[i].fraction = Data[j].fractionModel[i];
        Data[j].fractionModel[i] = p[freeVarCounter]; // a vote of confidence for the counting scheme
    }
}

for ( k=0; k<maxModels; k++ )
if ( model[k].modelActive )
{
    for( i=0; i<maxPeaks; i++ )
        if ( model[k].peak[i].active )
            {
                if ( model[k].peak[i].energyVary.vary == free ||
                    model[k].peak[i].energyVary.vary == limited )
                    {
                        freeVarCounter++;
                        if(model[k].peak[i].energy >= 0)
                            deltaEnergy = -0.01;
                        else if(model[k].peak[i].energy < 0 )
                            deltaEnergy = 0.01;
                        model[k].peak[i].energy += deltaEnergy;
                        checkAllModelsLimitsAndCorrelations(typeSpace);
                        GetFitAndChisqModels(typeSpace);
                        if( singularity )
                            return;
                    }
                for( j=0; j<maxFiles; j++ )
                    if( Data[j].fit && Data[j].gotModifiedCurveRe && Data[j].gotModifiedCurveIm )
                        {
                            numP1 = Data[j].FitCurveRe.numP;
                            for( join=0; join < numP1; join++ )
                                {
                                    y2[j][join] = Data[j].FitCurveRe.y[join];
                                    Data[j].GradY[join][freeVarCounter-1] = ( y2[j][join] - y[j][join] ) / deltaEnergy;
                                }
                            numP2 = Data[j].FitCurveIm.numP;
                            for( join=0; join < numP2; join++ )
                                {
                                    y2[j][numP1+join] = Data[j].FitCurveIm.y[join];
                                    Data[j].GradY[numP1+join][freeVarCounter-1] = ( y2[j][numP1+join] -
                                                                                    y[j][numP1+join] ) / deltaEnergy;
                                }
                        }
                model[k].peak[i].energy = p[freeVarCounter];
            }

            if ( model[k].peak[i].gaussVary.vary == free ||
                model[k].peak[i].gaussVary.vary == limited )
                {
                    freeVarCounter++;
                    if ( model[k].peak[i].gauss == 0.1 )
                        deltaGauss = -0.001;
                    else

```

```

    deltaGauss = 0.001;
    model[k].peak[i].gauss += deltaGauss;
    checkAllModelsLimitsAndCorrelations(typeSpace);
    GetFitAndChisqModels(typeSpace);
    if( singularity )
        return;
    for( j=0; j<maxFiles; j++ )
        if( Data[j].fit && Data[j].gotModifiedCurveRe && Data[j].gotModifiedCurveIm )
            {
                numP1 = Data[j].FitCurveRe.numP;
                for( join=0; join < numP1; join++ )
                    {
                        y2[j][join] = Data[j].FitCurveRe.y[join];
                        Data[j].GradY[join][freeVarCounter-1] = ( y2[j][join] - y[j][join] ) / deltaGauss;
                    }
                numP2 = Data[j].FitCurveIm.numP;
                for( join=0; join < numP2; join++ )
                    {
                        y2[j][numP1+join] = Data[j].FitCurveIm.y[join];
                        Data[j].GradY[numP1+join][freeVarCounter-1] = ( y2[j][numP1+join] -
                                                                    y[j][numP1+join] ) / deltaGauss;
                    }
            }
    }
    model[k].peak[i].gauss = p[freeVarCounter];
}
else if (model[k].peak[i].gaussVary.vary == correlated &&
        strcmp(model[k].peak[i].gaussVary.correlation,"correlationSigmaOfRatio") == 0 )
    {
        freeVarCounter++;
        if (model[k].peak[i].factorSigma == 0.002)
            deltaFactorSigma = -0.0001;
        else
            deltaFactorSigma = 0.0001;
        model[k].peak[i].factorSigma += deltaFactorSigma;
        checkAllModelsLimitsAndCorrelations(typeSpace);
        GetFitAndChisqModels(typeSpace);
        if( singularity )
            return;
        for( j=0; j<maxFiles; j++ )
            if( Data[j].fit && Data[j].gotModifiedCurveRe && Data[j].gotModifiedCurveIm )
                {
                    numP1 = Data[j].FitCurveRe.numP;
                    for( join=0; join < numP1; join++ )
                        {
                            y2[j][join] = Data[j].FitCurveRe.y[join];
                            Data[j].GradY[join][freeVarCounter-1] = ( y2[j][join] - y[j][join] ) / deltaFactorSigma;
                        }
                    numP2 = Data[j].FitCurveIm.numP;
                    for( join=0; join < numP2; join++ )
                        {
                            y2[j][numP1+join] = Data[j].FitCurveIm.y[join];
                            Data[j].GradY[numP1+join][freeVarCounter-1] = ( y2[j][numP1+join] -
                                                                            y[j][numP1+join] ) / deltaFactorSigma;
                        }
                }
    }
    model[k].peak[i].factorSigma = p[freeVarCounter];

```

```

}
if( model[k].peak[i].branchingRatioVary.vary == free ||
    model[k].peak[i].branchingRatioVary.vary == limited )
{
    freeVarCounter++;
    deltaBranchingRatio = 0.01;
    model[k].peak[i].branchingRatio += deltaBranchingRatio;
    checkAllModelsLimitsAndCorrelations(typeSpace);
    GetFitAndChisqModels(typeSpace);
    if( singularity )
        return;
    for( j=0; j<maxFiles; j++ )
        if( Data[j].fit && Data[j].gotModifiedCurveRe && Data[j].gotModifiedCurveIm )
            {
                numP1 = Data[j].FitCurveRe.numP;
                for( join=0; join < numP1; join++ )
                    {
                        y2[j][join] = Data[j].FitCurveRe.y[join];
                        Data[j].GradY[join][freeVarCounter-1] = ( y2[j][join] - y[j][join] ) /
                                                                    deltaBranchingRatio;
                    }
                numP2 = Data[j].FitCurveIm.numP;
                for( join=0; join < numP2; join++ )
                    {
                        y2[j][numP1+join] = Data[j].FitCurveIm.y[join];
                        Data[j].GradY[numP1+join][freeVarCounter-1] = ( y2[j][numP1+join] -
                                                                    y[j][numP1+join] ) / deltaBranchingRatio;
                    }
            }
        model[k].peak[i].branchingRatio = p[freeVarCounter];
    }
}
switch (k)
{
case model4AsV:
    if( model[k].alphaAsVary.vary == free ||
        model[k].alphaAsVary.vary == limited )
        {
            freeVarCounter++;
            if( model[k].alphaAs == -0.10 )
                deltaAlphaAs = 0.02;
            else if( model[k].alphaAs == 0.10 )
                deltaAlphaAs = -0.02;
            else
                deltaAlphaAs = min(0.005, max(0.001, fabs( model[k].alphaAs) / 10 ));
            model[k].alphaAs += deltaAlphaAs;
            checkAllModelsLimitsAndCorrelations(typeSpace);
            GetFitAndChisqModels(typeSpace);
            if( singularity )
                return;
            for( j=0; j<maxFiles; j++ )
                if( Data[j].fit && Data[j].gotModifiedCurveRe && Data[j].gotModifiedCurveIm )
                    {
                        numP1 = Data[j].FitCurveRe.numP;
                        for( join=0; join < numP1; join++ )
                            {

```

```

        y2[j][join] = Data[j].FitCurveRe.y[join];
        Data[j].GradY[join][freeVarCounter-1] = ( y2[j][join] - y[j][join] ) / deltaAlphaAs;
    }
    numP2 = Data[j].FitCurveIm.numP;
    for( join=0; join < numP2; join++ )
    {
        y2[j][numP1+join] = Data[j].FitCurveIm.y[join];
        Data[j].GradY[numP1+join][freeVarCounter-1] = ( y2[j][numP1+join] -
            y[j][numP1+join] ) / deltaAlphaAs;
    }
}
model[k].alphaAs = p[freeVarCounter];

}
if( model[k].betaSiVary.vary == free ||
    model[k].betaSiVary.vary == limited )
{
    freeVarCounter++;
    if( model[k].betaSi == -0.10 )
        deltaBetaSi = 0.02;
    else if( model[k].betaSi == 0.10 )
        deltaBetaSi = -0.02;
    else
        deltaBetaSi = min(0.005, max(0.001, fabs( model[k].betaSi) / 10 ));
    model[k].betaSi += deltaBetaSi;
    checkAllModelsLimitsAndCorrelations(typeSpace);
    GetFitAndChisqModels(typeSpace);
    if( singularity )
        return;
    for( j=0; j<maxFiles; j++ )
        if( Data[j].fit && Data[j].gotModifiedCurveRe && Data[j].gotModifiedCurveIm )
        {
            numP1 = Data[j].FitCurveRe.numP;
            for( join=0; join < numP1; join++ )
            {
                y2[j][join] = Data[j].FitCurveRe.y[join];
                Data[j].GradY[join][freeVarCounter-1] = ( y2[j][join] - y[j][join] ) / deltaBetaSi;
            }
            numP2 = Data[j].FitCurveIm.numP;
            for( join=0; join < numP2; join++ )
            {
                y2[j][numP1+join] = Data[j].FitCurveIm.y[join];
                Data[j].GradY[numP1+join][freeVarCounter-1] = ( y2[j][numP1+join] -
                    y[j][numP1+join] ) / deltaBetaSi;
            }
        }
}
model[k].betaSi = p[freeVarCounter];
}
if( model[k].S0Vary.vary == free ||
    model[k].S0Vary.vary == limited )
{
    freeVarCounter++;
    if( model[k].S0 == 0.50 )
        deltaS0 = 0.02;
    else if( model[k].S0 == 1 )
        deltaS0 = -0.02;
}

```

```

else
    deltaS0 = 0.01;
model[k].S0 += deltaS0;
checkAllModelsLimitsAndCorrelations(typeSpace);
GetFitAndChisqModels(typeSpace);
if( singularity )
    return;
for( j=0; j<maxFiles; j++ )
    if( Data[j].fit && Data[j].gotModifiedCurveRe && Data[j].gotModifiedCurveIm )
    {
        numP1 = Data[j].FitCurveRe.numP;
        for( join=0; join < numP1; join++ )
        {
            y2[j][join] = Data[j].FitCurveRe.y[j][join];
            Data[j].GradY[j][join][freeVarCounter-1] = ( y2[j][join] - y[j][join] ) / deltaS0;
        }
        numP2 = Data[j].FitCurveIm.numP;
        for( join=0; join < numP2; join++ )
        {
            y2[j][numP1+join] = Data[j].FitCurveIm.y[j][join];
            Data[j].GradY[numP1+join][freeVarCounter-1] = ( y2[j][numP1+join] -
                                                            y[j][numP1+join] ) / deltaS0;
        }
    }
model[k].S0 = p[freeVarCounter];
}
break;
case model3AsVSi:
if( model[k].alphaAsVary.vary == free ||
    model[k].alphaAsVary.vary == limited )
{
    freeVarCounter++;
    if( model[k].alphaAs == -0.10 )
        deltaAlphaAs = 0.02;
    else if( model[k].alphaAs == 0.10 )
        deltaAlphaAs = -0.02;
    else
        deltaAlphaAs = min(0.005, max(0.001, fabs( model[k].alphaAs ) / 10 ));
    model[k].alphaAs += deltaAlphaAs;
    checkAllModelsLimitsAndCorrelations(typeSpace);
    GetFitAndChisqModels(typeSpace);
    if( singularity )
        return;
for( j=0; j<maxFiles; j++ )
    if( Data[j].fit && Data[j].gotModifiedCurveRe && Data[j].gotModifiedCurveIm )
    {
        numP1 = Data[j].FitCurveRe.numP;
        for( join=0; join < numP1; join++ )
        {
            y2[j][join] = Data[j].FitCurveRe.y[j][join];
            Data[j].GradY[j][join][freeVarCounter-1] = ( y2[j][join] - y[j][join] ) / deltaAlphaAs;
        }
        numP2 = Data[j].FitCurveIm.numP;
        for( join=0; join < numP2; join++ )
        {
            y2[j][numP1+join] = Data[j].FitCurveIm.y[j][join];

```

```

        Data[j].GradY[numP1+join][freeVarCounter-1] = ( y2[j][numP1+join] -
                                                    y[j][numP1+join] ) / deltaAlphaSi;
    }
}
model[k].alphaAs = p[freeVarCounter];
}
if( model[k].alphaSiVary.vary == free ||
    model[k].alphaSiVary.vary == limited )
{
    freeVarCounter++;
    if( model[k].alphaSi == -0.10 )
        deltaAlphaSi = 0.02;
    else if( model[k].alphaSi == 0.10 )
        deltaAlphaSi = -0.02;
    else
        deltaAlphaSi = min(0.005, max(0.001, fabs( model[k].alphaSi) / 10 ));
    model[k].alphaSi += deltaAlphaSi;
    checkAllModelsLimitsAndCorrelations(typeSpace);
    GetFitAndChisqModels(typeSpace);
    if( singularity )
        return;
for( j=0; j<maxFiles; j++ )
    if( Data[j].fit && Data[j].gotModifiedCurveRe && Data[j].gotModifiedCurveIm )
    {
        numP1 = Data[j].FitCurveRe.numP;
        for( join=0; join < numP1; join++ )
        {
            y2[j][join] = Data[j].FitCurveRe.y[join];
            Data[j].GradY[join][freeVarCounter-1] = ( y2[j][join] - y[j][join] ) / deltaAlphaSi;
        }
        numP2 = Data[j].FitCurveIm.numP;
        for( join=0; join < numP2; join++ )
        {
            y2[j][numP1+join] = Data[j].FitCurveIm.y[join];
            Data[j].GradY[numP1+join][freeVarCounter-1] = ( y2[j][numP1+join] -
                                                            y[j][numP1+join] ) / deltaAlphaSi;
        }
    }
}
model[k].alphaSi = p[freeVarCounter];
}
if( model[k].betaSiVary.vary == free ||
    model[k].betaSiVary.vary == limited )
{
    freeVarCounter++;
    if( model[k].betaSi == -0.10 )
        deltaBetaSi = 0.02;
    else if( model[k].betaSi == 0.10 )
        deltaBetaSi = -0.02;
    else
        deltaBetaSi = min(0.005, max(0.001, fabs( model[k].betaSi) / 10 ));
    model[k].betaSi += deltaBetaSi;
    checkAllModelsLimitsAndCorrelations(typeSpace);
    GetFitAndChisqModels(typeSpace);
    if( singularity )
        return;
for( j=0; j<maxFiles; j++ )

```



```

if( Data[j].fit && Data[j].gotModifiedCurveRe && Data[j].gotModifiedCurveIm )
{
    numP1 = Data[j].FitCurveRe.numP;
    for( join=0; join < numP1; join++ )
    {
        y2[j][join] = Data[j].FitCurveRe.y[join];
        Data[j].GradY[join][freeVarCounter-1] = ( y2[j][join] - y[j][join] ) / deltaBetaSi;
    }
    numP2 = Data[j].FitCurveIm.numP;
    for( join=0; join < numP2; join++ )
    {
        y2[j][numP1+join] = Data[j].FitCurveIm.y[join];
        Data[j].GradY[numP1+join][freeVarCounter-1] = ( y2[j][numP1+join] -
            y[j][numP1+join] ) / deltaBetaSi;
    }
}
model[k].betaSi = p[freeVarCounter];
}
if( model[k].S0Vary.vary == free ||
    model[k].S0Vary.vary == limited )
{
    freeVarCounter++;
    if( model[k].S0 == 0.50 )
        deltaS0 = 0.02;
    else if( model[k].S0 == 1 )
        deltaS0 = -0.02;
    else
        deltaS0 = 0.01;
    model[k].S0 += deltaS0;
    checkAllModelsLimitsAndCorrelations(typeSpace);
    GetFitAndChisqModels(typeSpace);
    if( singularity )
        return;
}
for( j=0; j<maxFiles; j++ )
if( Data[j].fit && Data[j].gotModifiedCurveRe && Data[j].gotModifiedCurveIm )
{
    numP1 = Data[j].FitCurveRe.numP;
    for( join=0; join < numP1; join++ )
    {
        y2[j][join] = Data[j].FitCurveRe.y[join];
        Data[j].GradY[join][freeVarCounter-1] = ( y2[j][join] - y[j][join] ) / deltaS0;
    }
    numP2 = Data[j].FitCurveIm.numP;
    for( join=0; join < numP2; join++ )
    {
        y2[j][numP1+join] = Data[j].FitCurveIm.y[join];
        Data[j].GradY[numP1+join][freeVarCounter-1] = ( y2[j][numP1+join] -
            y[j][numP1+join] ) / deltaS0;
    }
}
model[k].S0 = p[freeVarCounter];
}
break;
case model2AsV2Si:
if( model[k].alphaAsVary.vary == free ||
    model[k].alphaAsVary.vary == limited )

```

```

{
freeVarCounter++;
if( model[k].alphaAs == -0.10 )
    deltaAlphaAs = 0.02;
else if( model[k].alphaAs == 0.10 )
    deltaAlphaAs = -0.02;
else
    deltaAlphaAs = min(0.005, max(0.001, fabs( model[k].alphaAs) / 10 ));
model[k].alphaAs += deltaAlphaAs;
checkAllModelsLimitsAndCorrelations(typeSpace);
GetFitAndChisqModels(typeSpace);
if( singularity )
    return;
for( j=0; j<maxFiles; j++ )
    if( Data[j].fit && Data[j].gotModifiedCurveRe && Data[j].gotModifiedCurveIm )
        {
        numP1 = Data[j].FitCurveRe.numP;
        for( join=0; join < numP1; join++ )
            {
            y2[j][join] = Data[j].FitCurveRe.y[join];
            Data[j].GradY[join][freeVarCounter-1] = ( y2[j][join] - y[j][join] ) / deltaAlphaAs;
            }
        numP2 = Data[j].FitCurveIm.numP;
        for( join=0; join < numP2; join++ )
            {
            y2[j][numP1+join] = Data[j].FitCurveIm.y[join];
            Data[j].GradY[numP1+join][freeVarCounter-1] = ( y2[j][numP1+join] -
                y[j][numP1+join] ) / deltaAlphaAs;
            }
        }
}
model[k].alphaAs = p[freeVarCounter];
}
if( model[k].alphaSiVary.vary == free ||
    model[k].alphaSiVary.vary == limited )
    {
    freeVarCounter++;
    if( model[k].alphaSi == -0.10 )
        deltaAlphaSi = 0.02;
    else if( model[k].alphaSi == 0.10 )
        deltaAlphaSi = -0.02;
    else
        deltaAlphaSi = min(0.005, max(0.001, fabs( model[k].alphaSi) / 10 ));
    model[k].alphaSi += deltaAlphaSi;
    checkAllModelsLimitsAndCorrelations(typeSpace);
    GetFitAndChisqModels(typeSpace);
    if( singularity )
        return;
    for( j=0; j<maxFiles; j++ )
        if( Data[j].fit && Data[j].gotModifiedCurveRe && Data[j].gotModifiedCurveIm )
            {
            numP1 = Data[j].FitCurveRe.numP;
            for( join=0; join < numP1; join++ )
                {
                y2[j][join] = Data[j].FitCurveRe.y[join];
                Data[j].GradY[join][freeVarCounter-1] = ( y2[j][join] - y[j][join] ) / deltaAlphaSi;
                }
            }
    }
}

```

```

numP2 = Data[j].FitCurveIm.numP;
for( join=0; join < numP2; join++ )
{
    y2[j][numP1+join] = Data[j].FitCurveIm.y[join];
    Data[j].GradY[numP1+join][freeVarCounter-1] = ( y2[j][numP1+join] -
                                                    y[j][numP1+join] ) / deltaAlphaSi;
}
}
model[k].alphaSi = p[freeVarCounter];
}
if( model[k].betaSiVary.vary == free || model[k].betaSiVary.vary == limited )
{
    freeVarCounter++;
    if( model[k].betaSi == -0.10 )
        deltaBetaSi = 0.02;
    else if( model[k].betaSi == 0.10 )
        deltaBetaSi = -0.02;
    else
        deltaBetaSi = min(0.005, max(0.001, fabs( model[k].betaSi) / 10 ));
    model[k].betaSi += deltaBetaSi;
    checkAllModelsLimitsAndCorrelations(typeSpace);
    GetFitAndChisqModels(typeSpace);
    if( singularity )
        return;
}
for( j=0; j<maxFiles; j++ )
    if( Data[j].fit && Data[j].gotModifiedCurveRe && Data[j].gotModifiedCurveIm )
    {
        numP1 = Data[j].FitCurveRe.numP;
        for( join=0; join < numP1; join++ )
        {
            y2[j][join] = Data[j].FitCurveRe.y[join];
            Data[j].GradY[join][freeVarCounter-1] = ( y2[j][join] - y[j][join] ) / deltaBetaSi;
        }
        numP2 = Data[j].FitCurveIm.numP;
        for( join=0; join < numP2; join++ )
        {
            y2[j][numP1+join] = Data[j].FitCurveIm.y[join];
            Data[j].GradY[numP1+join][freeVarCounter-1] = ( y2[j][numP1+join] -
                                                            y[j][numP1+join] ) / deltaBetaSi;
        }
    }
}
model[k].betaSi = p[freeVarCounter];
}
if( model[k].S0Vary.vary == free ||
    model[k].S0Vary.vary == limited )
{
    freeVarCounter++;
    if( model[k].S0 == 0.50 )
        deltaS0 = 0.02;
    else if( model[k].S0 == 1 )
        deltaS0 = -0.02;
    else
        deltaS0 = 0.01;
    model[k].S0 += deltaS0;
    checkAllModelsLimitsAndCorrelations(typeSpace);
    GetFitAndChisqModels(typeSpace);
}

```

```

if( singularity )
    return;
for( j=0; j<maxFiles; j++ )
    if( Data[j].fit && Data[j].gotModifiedCurveRe && Data[j].gotModifiedCurveIm )
    {
        numP1 = Data[j].FitCurveRe.numP;
        for( join=0; join < numP1; join++ )
        {
            y2[j][join] = Data[j].FitCurveRe.y[join];
            Data[j].GradY[join][freeVarCounter-1] = ( y2[j][join] - y[j][join] ) / deltaS0;
        }
        numP2 = Data[j].FitCurveIm.numP;
        for( join=0; join < numP2; join++ )
        {
            y2[j][numP1+join] = Data[j].FitCurveIm.y[join];
            Data[j].GradY[numP1+join][freeVarCounter-1] = ( y2[j][numP1+join] -
                y[j][numP1+join] ) / deltaS0;
        }
    }
model[k].S0 = p[freeVarCounter];
}
break;
case modelAs4Si:
if( model[k].alphaSiVary.vary == free || model[k].alphaSiVary.vary == limited )
{
    freeVarCounter++;
    if( model[k].alphaSi == -0.10 )
        deltaAlphaSi = 0.02;
    else if( model[k].alphaSi == 0.10 )
        deltaAlphaSi = -0.02;
    else
        deltaAlphaSi = min(0.005, max(0.001, fabs( model[k].alphaSi) / 10 ));
    model[k].alphaSi += deltaAlphaSi;
    checkAllModelsLimitsAndCorrelations(typeSpace);
    GetFitAndChisqModels(typeSpace);
    if( singularity )
        return;
for( j=0; j<maxFiles; j++ )
    if( Data[j].fit && Data[j].gotModifiedCurveRe && Data[j].gotModifiedCurveIm )
    {
        numP1 = Data[j].FitCurveRe.numP;
        for( join=0; join < numP1; join++ )
        {
            y2[j][join] = Data[j].FitCurveRe.y[join];
            Data[j].GradY[join][freeVarCounter-1] = ( y2[j][join] - y[j][join] ) / deltaAlphaSi;
        }
        numP2 = Data[j].FitCurveIm.numP;
        for( join=0; join < numP2; join++ )
        {
            y2[j][numP1+join] = Data[j].FitCurveIm.y[join];
            Data[j].GradY[numP1+join][freeVarCounter-1] = ( y2[j][numP1+join] -
                y[j][numP1+join] ) / deltaAlphaSi;
        }
    }
}
model[k].alphaSi = p[freeVarCounter];
}

```

```

if( model[k].betaSiVary.vary == free || model[k].betaSiVary.vary == limited )
{
freeVarCounter++;
if( model[k].betaSi == -0.10 )
deltaBetaSi = 0.02;
else if( model[k].betaSi == 0.10 )
deltaBetaSi = -0.02;
else
deltaBetaSi = min(0.005, max(0.001, fabs( model[k].betaSi) / 10 ));
model[k].betaSi += deltaBetaSi;
checkAllModelsLimitsAndCorrelations(typeSpace);
GetFitAndChisqModels(typeSpace);
if( singularity )
return;
for( j=0; j<maxFiles; j++ )
if( Data[j].fit && Data[j].gotModifiedCurveRe && Data[j].gotModifiedCurveIm )
{
numP1 = Data[j].FitCurveRe.numP;
for( join=0; join < numP1; join++ )
{
y2[j][join] = Data[j].FitCurveRe.y[join];
Data[j].GradY[join][freeVarCounter-1] = ( y2[j][join] - y[j][join] ) / deltaBetaSi;
}
numP2 = Data[j].FitCurveIm.numP;
for( join=0; join < numP2; join++ )
{
y2[j][numP1+join] = Data[j].FitCurveIm.y[join];
Data[j].GradY[numP1+join][freeVarCounter-1] = ( y2[j][numP1+join] -
y[j][numP1+join] ) / deltaBetaSi;
}
}
}
model[k].betaSi = p[freeVarCounter];
}
if( model[k].S0Vary.vary == free ||
model[k].S0Vary.vary == limited )
{
freeVarCounter++;
if( model[k].S0 == 0.50 )
deltaS0 = 0.02;
else if( model[k].S0 == 1 )
deltaS0 = -0.02;
else
deltaS0 = 0.01;
model[k].S0 += deltaS0;
checkAllModelsLimitsAndCorrelations(typeSpace);
GetFitAndChisqModels(typeSpace);
if( singularity )
return;
for( j=0; j<maxFiles; j++ )
if( Data[j].fit && Data[j].gotModifiedCurveRe && Data[j].gotModifiedCurveIm )
{
numP1 = Data[j].FitCurveRe.numP;
for( join=0; join < numP1; join++ )
{
y2[j][join] = Data[j].FitCurveRe.y[join];
Data[j].GradY[join][freeVarCounter-1] = ( y2[j][join] - y[j][join] ) / deltaS0;
}
}
}
}

```

```

    }
    numP2 = Data[j].FitCurveIm.numP;
    for( join=0; join < numP2; join++ )
    {
        y2[j][numP1+join] = Data[j].FitCurveIm.y[join];
        Data[j].GradY[numP1+join][freeVarCounter-1] = ( y2[j][numP1+join] -
                                                    y[j][numP1+join] ) / deltaS0;
    }
    }
    model[k].S0 = p[freeVarCounter];
    }
    } // end of switch (k)
    } // end of if (model[k].modelActive )
    checkAllModelsLimitsAndCorrelations(typeSpace);
// CheckAllLimits();
// CheckAllCorrelations();
#undef min
#undef max
}
}

```

U. GetFitAndChisqModels()

```

float TFirm3::GetFitAndChisqModels(int typeSpace)
{
    int          i,j;
    float        chisq;
    float        totalChisq = 0;
    char         label[255];
    bool         noModelsActive = true;

    garCount++;
    for( i=0; i<maxModels; i++ )
        if( model[i].modelActive )
            {
                noModelsActive = false;
                feffGroupBox[i]->Visible = true;
                DisabledToolBar();
            }
    if( noModelsActive )
        {
            EnableToolBar();
            return 0;
        }
    for ( j=0; j<maxFiles; j++)
        if ( Data[j].fit )
            for( i=0; i<maxModels; i++ ) //cambios 15/junio/2002, y 30/may
                if( model[i].modelActive )
                    {
                        GeneratePeakCurveModels( j, i, typeSpace);
                        if (singularity)
                            return 0;
                    }
    }
}

```

```

for( i=0; i<maxFiles; i++ )
{
    if( Data[i].fit && (Data[i].gotModifiedCurveAbs[typeSpace] || Data[i].gotModifiedCurveRe ||
        Data[i].gotModifiedCurveIm))
    {
        chisq = GetOneFitAndChiSqrModels( i, typeSpace );
        if( singularity )
            return 0;
        totalChisq += chisq;
    }
}

sprintf( label, "%.4g\0", totalChisq );
totalChiSqrLabel->Caption = label;
totalChiSqrLabel2->Caption = label;

return totalChisq;
}

```

V. **GetOneFitAndChiSqrModels()**

```

float TFirm3::GetOneFitAndChiSqrModels( int k, int typeSpace )
{
    float chisq;

    switch( typeSpace )
    {
        case spaceK:
            chisq = GetOneFitAndChiSqrModelsAbs(k, typeSpace);
            break;
        case spaceR:
            chisq = GetOneFitAndChiSqrModelsReAndIm(k, typeSpace);
    }
    return chisq;
}

```

W. **GetOneFitAndChiSqrModelsAbs()**

```

float TFirm3::GetOneFitAndChiSqrModelsAbs( int k, int typeSpace )
{
    int          i, j;
    float        chisq, total;
    int          numP=Data[k].ModifiedCurveAbs[typeSpace].numP;
    float        gar;
    double       area;
    double       totalArea=0;

    for(totalArea=0, i=0; i<maxModels; i++ )
        if( model[i].modelActive )
            {
                totalArea += Data[k].fractionModel[i];
            }
}

```

```

    Data[k].peak[i].active = model[i].modelActive;
}
else
    Data[k].peak[i].active = model[i].modelActive;
Data[k].totalArea = totalArea;
Data[k].FitCurveAbs[typeSpace].numP = numP;
Data[k].ResidualCurveAbs[typeSpace].numP = numP;
for( i=0; i<numP; i++ )
{
    Data[k].FitCurveAbs[typeSpace].x[i] = Data[k].ModifiedCurveAbs[typeSpace].x[i];
    Data[k].FitCurveAbs[typeSpace].y[i] = 0;
    for( j=0; j<maxPeaks; j++ )
        if ( Data[k].peak[j].active )
            Data[k].FitCurveAbs[typeSpace].y[i] += Data[k].peak[j].peakCurveAbs.y[i] *
                                                    Data[k].fractionModel[j];
    Data[k].ResidualCurveAbs[typeSpace].x[i] = Data[k].ModifiedCurveAbs[typeSpace].x[i];
    Data[k].ResidualCurveAbs[typeSpace].y[i] = Data[k].ModifiedCurveAbs[typeSpace].y[i] -
                                                    Data[k].FitCurveRe.y[i];
}
}

if( chisqLimits.useLimits[typeSpace] )
for( chisq = 0, total = 0, i=0; i<numP; i++ )
{
    if( Data[k].ModifiedCurveAbs[typeSpace].x[i] >= chisqLimits.leftLimit[typeSpace] &&
        Data[k].ModifiedCurveAbs[typeSpace].x[i] <= chisqLimits.rightLimit[typeSpace] )
        {
            chisq += pow( Data[k].FitCurveAbs[typeSpace].y[i] -
                          Data[k].ModifiedCurveAbs[typeSpace].y[i], 2 );
            total += pow( Data[k].ModifiedCurveAbs[typeSpace].y[i], 2 );
        }
}
else
for(chisq = 0, total = 0, i=0; i<numP; i++ )
{
    chisq += pow( Data[k].FitCurveAbs[typeSpace].y[i] -
                  Data[k].ModifiedCurveAbs[typeSpace].y[i], 2 );
    total += pow( Data[k].ModifiedCurveAbs[typeSpace].y[i], 2 );
}
chisq = chisq / total ;
Data[k].chisq = chisq;
Data[k].gotFitCurveAbs[typeSpace] = true;
Data[k].gotParameters = true;
plotFit[k]->Enabled = true;
// til here
return chisq;
} //borrar lo que sigue

```

X. GetOneFitAndChiSqrModelsReAndIm

```

float TFirm3::GetOneFitAndChiSqrModelsReAndIm( int k, int typeSpace )
{
    int          i, j;
    float        chisq=0;

```



```

float          chisqRe,chisqIm;
float          totalRe,totalIm;
int            numP=Data[k].ModifiedCurveRe.numP;
double        area;
double        totalArea=0;

```

```

for(totalArea=0, i=0; i<maxModels; i++)
  if( model[i].modelActive )
  {
    totalArea += Data[k].fractionModel[i];
    Data[k].peak[i].active = model[i].modelActive;
  }
  else
    Data[k].peak[i].active = model[i].modelActive;

Data[k].totalArea = totalArea;
Data[k].ResidualCurveRe.numP = numP;
Data[k].FitCurveRe.numP = numP;
Data[k].ResidualCurveIm.numP = numP;
Data[k].FitCurveIm.numP = numP;
Data[k].FitCurveAbs[typeSpace].numP = numP;
Data[k].ResidualCurveAbs[typeSpace].numP = numP;
for( i=0; i<numP; i++)
{
  Data[k].FitCurveRe.x[i] = Data[k].ModifiedCurveRe.x[i];
  Data[k].FitCurveRe.y[i] = 0;
  Data[k].FitCurveIm.x[i] = Data[k].ModifiedCurveIm.x[i];
  Data[k].FitCurveIm.y[i] = 0;
  for( j=0; j<maxModels; j++)
    if( model[j].modelActive )
    {
      Data[k].FitCurveRe.y[i] += Data[k].peak[j].peakCurveRe.y[i] * Data[k].fractionModel[j];
      Data[k].FitCurveIm.y[i] += Data[k].peak[j].peakCurveIm.y[i] * Data[k].fractionModel[j];
    }
  Data[k].ResidualCurveRe.x[i] = Data[k].ModifiedCurveRe.x[i];
  Data[k].ResidualCurveRe.y[i] = Data[k].ModifiedCurveRe.y[i] - Data[k].FitCurveRe.y[i];
  Data[k].ResidualCurveIm.x[i] = Data[k].ModifiedCurveIm.x[i];
  Data[k].ResidualCurveIm.y[i] = Data[k].ModifiedCurveIm.y[i] - Data[k].FitCurveIm.y[i];
}

if( chisqLimits.useLimits[typeSpace] )
{
  for( chisqIm = 0, chisqRe = 0, totalRe = 0, totalIm = 0, i=0; i<numP; i++)
  {
    if( Data[k].ModifiedCurveRe.x[i] >= chisqLimits.leftLimit[typeSpace] &&
        Data[k].ModifiedCurveRe.x[i] <= chisqLimits.rightLimit[typeSpace] )
    {
      chisqRe += pow( Data[k].FitCurveRe.y[i] - Data[k].ModifiedCurveRe.y[i], 2 );
      totalRe += pow( Data[k].ModifiedCurveRe.y[i], 2 );
      chisqIm += pow( Data[k].FitCurveIm.y[i] - Data[k].ModifiedCurveIm.y[i], 2 );
      totalIm += pow( Data[k].ModifiedCurveIm.y[i], 2 );
    }
  }
}
}
else

```

```

{
  for(chisqIm = 0, chisqRe = 0, totalRe = 0, totalIm = 0, i=0; i<numP; i++)
  {
    chisqRe += pow( Data[k].FitCurveRe.y[i] - Data[k].ModifiedCurveRe.y[i], 2 );
    totalRe += pow( Data[k].ModifiedCurveRe.y[i], 2 );
    chisqIm += pow( Data[k].FitCurveIm.y[i] - Data[k].ModifiedCurveIm.y[i], 2 );
    totalIm += pow( Data[k].ModifiedCurveIm.y[i], 2 );
  }
}
if ( totalRe!=0 && totalIm!=0 )
{
  chisqRe = chisqRe/totalRe;
  chisqIm = chisqIm/totalIm;
}
else
{
  singularity = true;
  return 0;
}
chisq = chisqRe + chisqIm ;
Data[k].chisq = chisq;

// generate FitCurveAbs from fitCurveRe and FitCurveIm
for ( i=0; i<numP; i++)
{
  Data[k].FitCurveAbs[typeSpace].x[i] = Data[k].ModifiedCurveAbs[typeSpace].x[i];
  Data[k].ResidualCurveAbs[typeSpace].x[i] = Data[k].ModifiedCurveAbs[typeSpace].x[i];
  Data[k].FitCurveAbs[typeSpace].y[i] = sqrt( Data[k].FitCurveRe.y[i]*Data[k].FitCurveRe.y[i] +
    Data[k].FitCurveIm.y[i]*Data[k].FitCurveIm.y[i]);
  Data[k].ResidualCurveAbs[typeSpace].y[i] = Data[k].ModifiedCurveAbs[typeSpace].y[i] -
    Data[k].FitCurveAbs[typeSpace].y[i];
}
Data[k].gotFitCurveAbs[typeSpace] = true;
Data[k].gotFitCurveRe = true;
Data[k].gotFitCurveIm = true;
Data[k].gotParameters = true;
plotFitR[k]->Enabled = true;
// til here

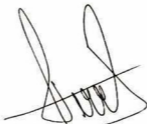
return chisq;
}

```

Bibliografia

- ¹ J. Mustre de Leon, J. J. Rehr, S. I. Zabinsky, and R. C. Albers. Phys. Rev. B **44**, 4146-4156 (1991).
- ² M.A. Berding and A. Sher, "Electronic Quasichemical Formalism: Application to arsenic deactivation in silicon.
- ³ K.C. Pandey, A.Erbil, G.S. Cargill III, R.F. Boehme, "Annealing of Heavily Arsenic-Doped Silicon: Electrical Deactivation and a New Defect Complex"
- ⁴ A. Parisini, A. Armigliato et al, "EXAFS study of the local atomic structure in As⁺ heavily implanted silicon
- ⁵ D.J. Chadi, A. Antonelli y Efthimios Kaxiras, "Vacncy in Silicon Recisted: Structure and Pressure Effect"
- ⁶ J. Mustre de Leon, J. J. Rehr, S. I. Zabinsky, and R. C. Albers. Phys. Rev. B **44**, 4146-4156 (1991).

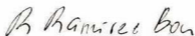
EL JURADO DESIGNADO POR LA UNIDAD QUERÉTARO DEL CENTRO DE INVESTIGACIÓN Y DE ESTUDIOS AVANZADOS DEL INSTITUTO POLITÉCNICO NACIONAL, APROBÓ LA TESIS DEL C. JOEL ANDRÉS CALDERÓN GUILLÉN TITULADA: "DESACTIVACIÓN ELÉCTRICA EN CRISTALES DE SILICIO ALTAMENTE DOPADOS CON ARSÉNICO", FIRMAN AL CALCE DE COMÚN ACUERDO LOS INTEGRANTES DE DICHO JURADO, EN LA CIUDAD DE QUERÉTARO, QRO., A LOS DIECIOCHO DÍAS DEL MES DE OCTUBRE DE 2002.



DR. ALBERTO HERRERA GÓMEZ
DIRECTOR DE TESIS



DR. SERGIO JOAQUÍN JIMÉNEZ SANDOVAL
SINODAL



DR. RAFAEL RAMÍREZ BON
SINODAL



CINVESTAV
BIBLIOTECA CENTRAL



SSIT000003798