TRABAJO ESPECIAL DE GRADO

DISEÑO DE UN PROGRAMA PARA LA CONVERSIÓN DE UNIDADES DE TIEMPO A UNIDADES DE PROFUNDIDAD EN HORIZONTES SÍSMICOS INTERPRETADOS

Presentado ante la Ilustre
Universidad Central de Venezuela
Por el Br. Da Encarnação Q.,
Colombo C.
Para optar al Título
de Ingeniero Geofísico

Caracas, 2008

TRABAJO ESPECIAL DE GRADO

DISEÑO DE UN PROGRAMA PARA LA CONVERSIÓN DE UNIDADES DE TIEMPO A UNIDADES DE PROFUNDIDAD EN HORIZONTES SÍSMICOS INTERPRETADOS

TUTOR ACADÉMICO: Prof. José Cavada
TUTOR INDUSTRIAL: Ing. Yaraixa Pérez

Presentado ante la Ilustre
Universidad Central de Venezuela
Por el Br. Da Encarnação Q.,
Colombo C.
Para optar al Título
de Ingeniero Geofísico

Caracas, 2008

Caracas, noviembre de 2008

Los abajo firmantes, miembros del Jurado designado por el Consejo de Escuela de Geología, Minas y Geofísica, para evaluar el Trabajo Especial de Grado presentado por el Bachiller Colombo Da Encarnação, titulado:

"Diseño de un programa para la conversión de unidades de tiempo a unidades de profundidad en Horizontes sísmicos interpretados"

Consideran que el mismo cumple con los requisitos exigidos por el plan de estudios conducente al Título de Ingeniero Geofísico, y sin que ello signifique que se hacen solidarios con las ideas expuestas por el autor, lo declaran APROBADO.

Prof. Yaraixa Pérez	Prof. Juan Infante
Jurado	Jurado

Prof. José Cavada
Tutor Académico

AGRADECIMIENTOS

Antes que todo agradezco a la ilustre Universidad Central de Venezuela por ser punto de encuentro del conocimiento, las culturas e ideologías. Las cuales son características de fundamentales para educar a un profesional íntegro, con criterio propio, moral y sin prejuicios en un momento donde la actualidad venezolana promueve las diferencias.

Agradezco al profesor José Cavada por haber sido tutor académico de este trabajo. Y por saber donde encontrar cualquier tipo de información.

A la profesora Yaraixa Pérez, la tutora industrial y de quien nace la idea de realizar este proyecto. Gracias amiga.

A Teikoku Oil & Gas Venezuela por ser la empresa que acoge la idea y me da los medios económicos y la experiencia para finalizar el proyecto.

A mis padres y hermana. Por darme la vida, la educación, los valores casi extintos en la actualidad caraqueña. Por ser fuente infinita de paciencia. Sin ustedes este documento no existiría.

A Ana Karina por acompañarme todos estos años, dejarme trabajar cuando tengo que hacerlo y estar en frente de cada salón después de mis clases. Te amo.

A todos los que estudiaron conmigo la carrera de ingeniero geofísico. Ahora somos competencia laboral, pero al fin amigos. Nos veremos en la empresa.

Al Chino, a Chuky y al concierto de Metallica en Venezuela que ninguno de nosotros pudo ir. Por que las frustraciones endurecen el alma. Los quiero.

A Jesús el pana de la fotocopiadora por tener todo el material de estudio de la carrera y por la extensión que me presto para poder defender este proyecto.

A la parroquia. Posiblemente el mejor centro psiquiátrico del país y punto de encuentro de los más reconocidos vagos y maleantes del recinto universitario. A todos gracias.

Da Encarnação Q., Colombo C.

DISEÑO DE UN PROGRAMA PARA LA CONVERSIÓN DE UNIDADES DE TIEMPO A UNIDADES DE PROFUNDIDAD EN HORIZONTES SÍSMICOS INTERPRETADOS

Tutor Académico: Prof. José Cavada. Tutor Industrial: Ing. Yaraixa Pérez. Tesis. Caracas, U.C.V. Facultad de Ingeniería. Escuela de Geología, Minas y Geofísica. 2008, XXX pp.

Palabras claves: Sísmica de reflexión, relación tiempo profundidad, modelos de velocidad, geoestadística, programación, lenguaje C/C++.

Resumen. El método sísmico de reflexión es la técnica de exploración geofísica más usada en la industria del petróleo y el gas. Con los datos obtenidos a partir de los estudios de reflexión es posible cartografiar estructuras geológicas en gran detalle. Uno de los pasos importantes dentro de esta técnica es la conversión de unidades de tiempo a profundidad en las interpretaciones realizadas. Por ello el propósito fundamental de este trabajo es diseñar una herramienta computacional que facilite mencionada tarea. En principio se eligió el lenguaje de programación C/C++ por su probada portabilidad, potencialidad y robustez en el diseño de software. Luego se inició el diseño de software mediante pseudocódigo. Se creo tipos de datos estructurados para pozos y superficies interpretadas. Se realizó funciones de lectura de archivos, almacenamiento de datos, generación de cuadriculas para las superficies en sus distintos dominios (tiempo, velocidad y profundidad), funciones de relación tiempo profundidad, y finalmente una función de impresión de resultados. Para probar la herramienta, se realizó experimentos en datos teóricos para observar el comportamiento de los datos. Se elaboró histogramas de frecuencia ante distintas geometrías con lo que se logro identificar comportamientos ante patrones ideales. Posteriormente se utilizó el software en datos reales. Finalmente se pudo concluir que las superficies se ajustan mejor mediante el método de promedios dentro de un radio de búsqueda adecuado y el modelo de velocidad ajusta mejor a métodos de poca variación lateral. Además aunque las curvas de TZ se aproximan más a los datos con la profundidad, el error de la conversión aumenta con la profundidad.

ÍNDICE GENERAL

Página

Capítulo I.....1

Planteamiento del problema.....1

Objetivos.....2

Objetivo General.....2

Objetivos Específicos.....2

Justificación.....3

Capítulo II.....4

Marco teórico.....4

Métodos Sísmicos.....4

Conversión tiempo-profundidad.....4

Propagación de las ondas sísmicas.....5

Constantes elásticas.....5

Tensión y deformación.....5

Coeficiente de Poisson.....6

Modulo de Young.....7

Modulo volumétrico y compresibilidad 7

Ondas elásticas.....8

Ondas de cuerpo.....9

Ondas P 9

Método de reflexión sísmica.....9

Corrección por NMO (NORMAL MOVEOUT).....11

Resolución sísmica vertical.....13

Determinación de la velocidad media (velocidad average).....14

Tiro de verificación sísmica.....14

Registros continuos de velocidad (registros sónicos) 15

Velocidad interválica (intervallic velocity).....15

Modelos de velocidad.....16

Métodos matemáticos.....17

.....17

Ajuste de datos.....17

Ajuste por mínimos cuadrados.....17

Estimadores lineales ponderados.....18

Estimador del vecino más cercano.....18

Estimador de valor promedio.....19

Inverso de la distancia lineal.....19

Programación.....20

Lenguaje C y C++.....21

Programación estructurada y programación orientada a objetos 21

Funciones y bibliotecas.....22

Archivos en C/C++.....23

Capítulo III.....23

Marco metodológico.....23

Página

Selección de herramientas de desarrollo....24
Desarrollo de código fuente....25
Archivos UnitMain....25
Tipos de dato....26
Archivo Superficie.h....27
Archivo Velocidades.h....27
Prueba de eficiencia del programa....27
Histogramas de frecuencia de datos sintéticos e ideales....28
Prueba con datos artificiales y reales....29

Capítulo IV....35

Análisis de resultados....36

Modelos de histogramas de frecuencia de datos sintéticos....36
Pruebas en datos sintéticos....46
Capas planas inclinadas....46
Capas elípticas....49
Pruebas en datos reales....52
Ambiente sedimentario de transición....52
Geología estructuralmente compleja....57

Capítulo V....61

Conclusiones y recomendaciones....61

Bibliografía....63

Anexos....65

Anexo 1: Cuadrículas generadas con los distintos métodos de ajuste por ponderación....65
Anexo 2: Mapas en profundidad de la prueba en datos reales....66
Anexo 3: Estructura para manipular velocidades....67
Anexo 4: Estructura para manipular y crear cuadriculas....70
Anexo 5: Estructura para calcular coeficientes de segundo grado por mínimos cuadrados....78
Anexo 6: Estructura que define los tipos de datos a utilizar....89

ÍNDICE DE FIGURAS

Página

Figura 1. Diferencia entre compresión y tracción (DIMUDA, 2006).....6
Figura 2: Esquema del método sísmico de reflexión.....10
Figura 3. Configuración de una reflexión para un medio homogéneo.....10
Figura 4. Tiempo de viaje de la reflexión de una onda.....11
Figura 5. Geometría NMO para un reflector horizontal 12
Figura 6. Corrección NMO.....12
Figura 7. Reflexión en varias capas.....13
Figura 8. Diagrama de adquisición del tiro de verificación sísmica.....14
Figura 9. Ejemplo de curva T-Z.....15
Figura 10. Algoritmo del estimador del vecino más cercano. 19
Figura 11. Algoritmo del ajuste por el promedio dentro de un área.....19
Figura 12. Ambiente de trabajo del Borland Builder C++ v.6 ®.....24
Figura 13. Ambiente de trabajo del Bloodshed Dev-C++.....25
Figura 14. Configuración de una capa plana inclinada.....28
Figura 16. Distribución aleatoria de datos sintéticos.....29
Figura 20. Carga de pozo en el programa de conversión.....33
Figura 21. Distribución de los datos dentro de la cuadricula.....33
Figura 22. Histogramas de frecuencia de las cuadriculas generadas.....34
Figura 23. Configuración de de una capa plana horizontal.....36
Figura 24. Histograma de de una capa plana horizontal. Solo posee una moda....37
Figura 25. Configuración de una capa plana horizontal fallada.....37
Figura 26. Histograma de una capa plana horizontal fallada.....38
Figura 27. Configuración de una capa plana inclinada.....38
Figura 29. Configuración de una capa plana inclinada fallada.....40
Figura 31. Configuración de una capa plana horizontal fallada varias veces.....41
Figura 33. Configuración de una capa parabólica.....42
Figura 35. Configuración de una capa parabólica fallada.....43
Figura 37. Configuración de una capa elipsoidal.....44
Figura 38. Histograma de una capa elipsoidal.....44

Figura 39. Configuración de una capa elipsoidal fallada.....45
Figura 41. Distribución de los datos sintéticos en capas planas inclinadas.....46
Figura 42. Esquema de los datos tiempo profundidad para cada superficie.....47
Figura 43. Distribución de datos sintéticos en capas elipsoidales.....49
Figura 44. Modelo sintético de capas elipsoidales.....49
Figura 45. Distribución las tablas TZ en capas elipsoidales.....50
Figura 46. Distribución las tablas TZ en capas elipsoidales.....51
Figura 47. Modelo en profundidad de las superficies elipsoidales.....52
Figura 48. Imagen de líneas sísmicas utilizadas y capas usadas en esta prueba..…53
Figura 49. Configuración en tiempo doble de viaje.....53
Figura 50. Histograma de frecuencia de los datos.....54
Figura 51. Curvas TZ del área de estudio.....55
Figura 52. Modelos de velocidad para cada una de las superficies geológicas.....55
Figura 53. Mapa en profundidad de la máxima superficie de inundación 57

ÍNDICE DE TABLAS

Página

Tabla 1. Rango de valores del coeficiente de Poisson en ciertos tipos de roca (DIMUDA, 2006).....7

Tabla 2. Constantes mecánicas para algunas rocas.....8

Tabla 3. Tipos de velocidades.....16

Tabla 4. Estadísticas de la capa plana inclinada.....28

Tabla 5. Muestra de los datos generados.....30

Tabla 6. Velocidades de los datos sintéticos.....30

Tabla 7. Estadísticas de los datos de entrada y las superficies de ajuste.....34

Tabla 8. Velocidades obtenidas a partir del polinomio de segundo grado.....35

Tabla 9. Error porcentual entre la profundidad hallada por velocidades promedio y velocidades interválicas.....35

Tabla 10. Estadísticas de la capa plana horizontal.....36

Tabla 11. Estadísticas de la capa plana horizontal fallada.....37

Tabla 12. Estadísticas de la capa plana inclinada.....38

Tabla 13. Estadísticas de la capa plana inclinada fallada.....39

Tabla 14. Estadísticas de la capa plana horizontal fallada varias veces.....41

Tabla 15. Estadísticas de la capa parabólica.....42

Tabla 16. Estadísticas de la capa parabólica fallada.....43

Tabla 17. Estadísticas de la capa elipsoidal.....44

Tabla 18. Estadísticas de la capa elipsoidal fallada.....45

Tabla 19: Correlaciones de Pearson entre los resultados y los datos de entrada.…48

Tabla 20: Estadísticas de las velocidades halladas 51

Tabla 21. Comparación de la profundidad obtenida con estudio de pozo y la profundidad calculada.....56

Tabla 22. Promedio de velocidades resultante para cada capa.....59

CAPÍTULO I

PLANTEAMIENTO DEL PROBLEMA

Los métodos sísmicos han permitido estudiar el subsuelo mediante sus propiedades mecánicas y generar modelos de la distribución de capas sedimentarias y estructuras geológicas en el espacio a un nivel de detalle tan elevado, que se han convertido en los medios de exploración geofísica de mayor uso y confianza en la industria petrolera. Este hecho ha generado que las empresas inviertan grandes cantidades de su capital en el desarrollo, compra o alquiler de instrumental especializado en las distintas fases de aplicación de dichos métodos: adquisición, procesamiento e interpretación.

A nivel de la fase de interpretación, un importante requerimiento es la conversión de unidades de tiempo a unidades de longitud en los datos adquiridos (Zijlstra, 1991), lo cual ha dado pie al desarrollo de diversas teorías, metodologías y tecnologías. Varias empresas han tomado el rol de diseñar y desarrollar paquetes computacionales que realicen estas tareas, pero esta tecnología muestra resultados que omiten el procedimiento seguido para su obtención; esto implica ciertas desventajas: excluye a usuarios, no es amigable y no permite interactuar y evaluar el algoritmo con el que se ejecutan determinadas tareas.

Es por ello que en el presente trabajo se llevó cabo una revisión de distintos procedimientos de conversión tiempo a profundidad, modelos de velocidad, métodos geoestadísticos y herramientas de desarrollo de software para elaborar un instrumento computacional, que mediante tablas tiempo profundidad de pozos y mapas de horizontes en tiempo (isócronos), permita convertir horizontes sísmicos en el dominio del tiempo al dominio de profundidad, visualice datos de entrada y resultados obtenidos, posea un alto nivel de confianza y cuya implementación se refleje en la reducción de costos operativos.

OBJETIVOS

Objetivo General

Elaborar una herramienta computacional que permita convertir los datos de horizontes interpretados sobre líneas sísmicas en tiempo a profundidad.

Objetivos Específicos

- Seleccionar las herramientas que permitan la elaboración del código y la visualización de los datos y resultados.
- Realizar una investigación de los métodos de conversión de tiempo a profundidad en la sísmica petrolera, y el contorneo de datos en geologías estructuralmente variables.
- Diseñar un algoritmo para la transformación de unidades de tiempo a profundidad.
- Construir modelos de velocidad que serán expresados en funciones numéricas (X, Y, Z) a lo largo de capas geológicas a partir de datos de pozo mediante técnicas de interpolación.
- Generar mapas en profundidad (usando algún programa especializado).
- Probar la herramienta con datos sintéticos y reales, y su efectividad tanto en áreas sencillas como aquellas con alta complejidad geológica.

JUSTIFICACIÓN

En el presente proyecto, se realizó una investigación basada en distintos procedimientos para la transformación de unidades de tiempo a profundidad en horizontes sísmicos interpretados, con el fin, de diseñar una herramienta computacional capaz de cumplir la mencionada tarea, esto permitiría satisfacer la necesidad de la Universidad Central de Venezuela y la empresa petrolera Teikoku Oil & Gas Venezuela, C.A., de poseer herramientas propias, eficientes, que agilicen la producción y cuya implementación se refleje en la reducción de costos.

Adicionalmente, se planteó una metodología que servirá de guía, promoverá y facilitará el futuro desarrollo de otras herramientas computacionales, así como, despertar el interés en el área de la geofísica instrumental que devenga no sólo en el desarrollo de software, sino también, en la elaboración de equipos mecánicos y electrónicos necesarios en el estudio de las geociencias.

Esto, con el tiempo, se traducirá en un desarrollo tecnológico del departamento de Geofísica de la Universidad Central de Venezuela, lo cual ayudará al fortalecimiento de la autonomía universitaria, al reducir la dependencia financiera del aporte gubernamental y privado con fines técnicos. Asimismo, proporcionará al departamento la posibilidad de ofrecer nuevos servicios a empresas, que aumenten las actividades de extensión y se traduzcan en ingresos económicos para el recinto universitario.

CAPÍTULO II

MARCO TEÓRICO

MÉTODOS SÍSMICOS

Los métodos sísmicos son un tipo de método geofísico, y constituyen pruebas realizadas para la determinación de las características mecánicas del subsuelo mediante la propagación de ondas sísmicas a través de un medio. La velocidad de propagación de las ondas sísmicas en el terreno depende de sus características de deformabilidad. En la hipótesis de suponer un comportamiento elástico para el terreno, la velocidad de las ondas longitudinales y transversales es en función de las constantes elásticas, por lo que con ambas expresiones pueden obtenerse dichos parámetros. (Dobrin, 1961)

Conversión tiempo-profundidad

Un paso importante en la interpretación sísmica es la conversión de unidades de tiempo a profundidad, esto porque las medidas realizadas en los estudios sísmicos son hechas en tiempo. La cantidad física que permite relacionar el tiempo con la profundidad es la velocidad. En la sísmica de reflexión petrolera, nos interesa conocer la velocidad de la onda longitudinal (onda P) a través del medio, motivado a que el procesamiento sísmico convencional elimina toda la energía sísmica existente excepto la representación de la reflexión de la onda P. La velocidad necesaria para la conversión es la que tiene dirección vertical, y esta puede ser obtenida a través de estudios de pozo que permitan generar tablas que relacionen tiempo con profundidad, (Brown, 2003).

Propagación de las ondas sísmicas

Según Dobrin (1961) los métodos sísmicos suministran información de los contrastes entre las propiedades elásticas de las rocas y para poder entenderlos es necesario conocer los principios de elasticidad y de la propagación de ondas elásticas. La velocidad de onda depende de las constantes elásticas del medio, y sobre el comportamiento de las mismas en lugares del suelo donde esta puede cambiar. Estos cambios están asociados a contactos geológicos son los que la prospección sísmica trata de detectar.

Constantes elásticas

Una constante elástica es cada uno de los parámetros físicamente mesurables que caracterizan el comportamiento elástico de un sólido deformable elástico-lineal. En los materiales elásticos homogéneos e isótropos las constantes elásticas presentan el mismo comportamiento mecánico para cualquier dirección de estiramiento alrededor de un punto. (Wikipedia, 2007)

Tensión y deformación

La *tensión* S se define como la fuerza F por unidad de área A, donde F es la fuerza aplicada uniformemente a una pequeña superficie de área. La *presión* o tracción se refiere a la tensión dirigida perpendicularmente al área, según sea ejercida hacia el cuerpo sobre que actúa (presión) o en sentido contrario (tracción). La presión causa un acortamiento en el interior del cuerpo, en cambio, la tracción produce un alargamiento en el interior del cuerpo.

La deformación longitudinal (ε_l) es la relación entre el alargamiento o el acortamiento (Δl) causado por una tensión y la longitud original (l) antes de la aplicación de la tensión: $\varepsilon_l = \Delta l/l$. La deformación transversal (ε_a) se define como la relación entre la variación del ancho (Δa) causada por una tensión y el ancho primitivo a del cuerpo antes de la aplicación de la tensión: $\varepsilon_a = \Delta a/a$. (DIMUDA, 2006)

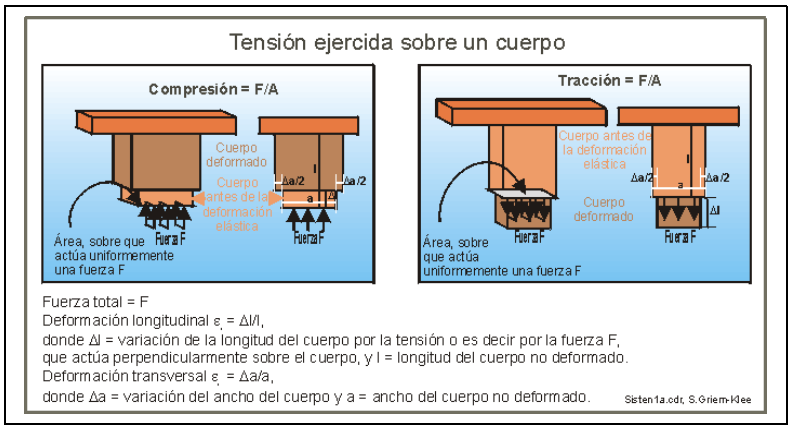


Figura 1. Diferencia entre compresión y tracción (DIMUDA, 2006).

Coeficiente de Poisson

Cuando un cuerpo se acorta por efecto de una compresión, se alarga en la dirección perpendicular a la compresión. Un cuerpo alargado por efecto de una tracción, disminuye su ancho en la dirección perpendicular a la tensión. La relación entre la deformación longitudinal (ε_l) y la deformación transversal (ε_a) se denomina coeficiente de Poisson (σ).

$$\sigma = \varepsilon_a / \varepsilon_l = (\Delta a/a) / (\Delta l/l).$$

Cuando una tensión actúa en un cuerpo en una dirección y el volumen del cuerpo es constante, el coeficiente de Poisson tiene su valor máximo igual a 0,5. (DIMUDA, 2006)

Tabla 1. Rango de valores del coeficiente de Poisson en ciertos tipos de roca (DIMUDA, 2006)

Tipo de roca	**Rango del coeficiente de Poisson**
Roca consolidada, no alterada	0,2 - 0,3
Roca sedimentaria clástica	0,02 - 0,05

Modulo de Young

Según Wikipedia (2007) el módulo de elasticidad o módulo de Young es un parámetro que caracteriza el comportamiento de un material elástico, según la dirección en la que se aplica una fuerza. Para un material elástico lineal e isótropo, el módulo de Young tiene el mismo valor para una tracción que para una compresión, siendo una constante independiente del esfuerzo siempre que no exceda de un valor máximo denominado límite elástico, y es siempre mayor que cero.

Para un material elástico el módulo de elasticidad longitudinal es una constante y se puede expresar mediante la siguiente relación:

$$\boldsymbol{E} = \sigma/\varepsilon$$

Donde: $\boldsymbol{E}$ es el módulo de Young, σ es la tensión sobre la barra usada para determinar el módulo de elasticidad y ε es la deformación unitaria en cualquier punto de la barra.

Modulo volumétrico y compresibilidad

Según Dobrin (1961) si un cuerpo de volumen V está sometido a una fuerza de compresión uniforme en todas direcciones, su volumen disminuirá en una cantidad ΔV. El módulo volumétrico (k) se define como la presión dividida por el correspondiente cambio proporcional de volumen, es decir:

$$k = S/(\Delta V/V)$$

El valor recíproco del módulo volumétrico se llama compresibilidad.

Tabla 2. Constantes mecánicas para algunas rocas (SCHMIDT-THOMÉ, 1965)

Tipo de roca	**Resistencia a la compresión en Kg./cm² -valores limites-**	**Resistencia a la tracción en Kg./cm²**	**Resistencia al cizallamiento en Kg./cm²**	**Módulo de YOUNG en Kg./cm² -valores medios-**
Basalto	1500 - 4500	150	300	800000
Granito de grano fino	2000 - 2500	65 - 115	150 - 160	
Granito de grano grueso y en general	370 - 3790	30 - 80	100 - 300	100000 - 400000
Pórfido de cuarzo	1500 - 3500	65	150	
Cuarcita	260 - 3200			100000 - 450000
Mármol	310 - 3000	30 - 90	100 - 300	800000
Caliza en general	60 - 3600	10 - 117	35- 200	100000 - 800000
Caliza porosa	400 - 800			
Dolomía	790 - 1300	16 - 28	70 -75	<100000 - 710000
Arenisca en general	100 - 3000	Oct-43	46 -150	<20000 - 636000
Arenisca calcárea	900 - 3000			30000 - 60000
Arcilla esquistosa	600 - 3130	250	50 - 250	40000 - 200000
Gneis	810 - 3270		<650	<100000 - 400000

Ondas elásticas

Las ondas elásticas son la propagación de perturbaciones temporales del campo de esfuerzos que generan pequeños movimientos en un medio. Las ondas sísmicas pueden ser generadas por movimientos telúricos naturales, los más grandes

de los cuales pueden causar daños en zonas donde hay asentamientos urbanos. Existe toda una rama de la sismología que se encarga del estudio de este tipo de fenómenos físicos. Las ondas sísmicas pueden ser generadas también artificialmente (en general por explosiones). La sísmica es la rama de la sismología que estudia estas ondas artificiales para, por ejemplo, la exploración del petróleo.

Ondas de cuerpo

Las ondas de cuerpo viajan a través del interior de la Tierra. Siguen caminos curvos debido a la variada densidad y composición del interior de la Tierra. Este efecto es similar al de refracción de ondas de luz. Las ondas de cuerpo transmiten los temblores preliminares de un terremoto pero poseen poco poder destructivo. Las ondas de cuerpo son divididas en dos grupos: ondas primarias (P) y secundarias (S).

Ondas P

Las ondas P u ondas primarias son ondas de cuerpo longitudinales o compresionales, lo cual significa que el suelo es alternadamente comprimido y dilatado en la dirección de la propagación. Estas ondas pueden viajar a través de cualquier tipo de material. Velocidades típicas son 330 m/s en el aire, 1450 m/s en el agua y cerca de 5000 m/s en el granito. (Wikipedia, 2007)

Según Dobrin (1961) la velocidad de las ondas longitudinales V_L está relacionada con las constantes elásticas y la densidad mediante la siguiente relación:

$$V_L = ((k + \mu 4/3)/\rho)^{1/2}$$

Donde ρ es la densidad del medio, k es el módulo volumétrico y μ es el modulo de rigidez.

Método de reflexión sísmica

Según Dobrin (1961) el método sísmico de reflexión es una técnica que levanta el mapa de la estructura del subsuelo haciendo uso de los tiempos de llegada de una onda sísmica generada en el suelo por alguna fuente próxima a la superficie (figura 2), para volver a ésta después de ser reflejada en las formaciones. Las

reflexiones son registradas por sensores colocados en superficie. Las variaciones en los tiempos de reflexión entre dos puntos dependen de las características estructurales de las rocas en el subsuelo. Las profundidades hasta las superficies reflectoras pueden ser determinadas en base a los tiempos de viaje, siempre que sea posible conocer la rapidez de las ondas sísmicas en la zona situadas encima de ellas.

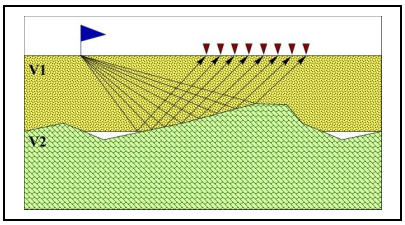


Figura 2: Esquema del método sísmico de reflexión.

El método de reflexión sísmica es el más usado de los métodos prospección geofísica por ser el que aporta un cuadro más directo y detallado de la geología del subsuelo. Sin embargo, este método es empleado casi exclusivamente para la industria petrolera, debido a su poca aplicabilidad en zonas someras.

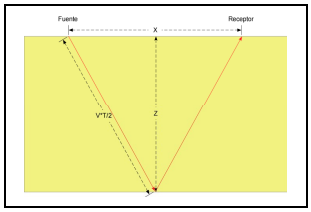


Figura 3. Configuración de una reflexión para un medio homogéneo, de una capa, una fuente y un solo receptor (Dobrin, 1961).

El *tiempo de viaje* de una reflexión sísmica depende de la velocidad de propagación de onda en un medio y de la trayectoria que esta recorra. Para la configuración más simple (figura 3) la relación matemática viene dada por:

$$T = (2/V)*(Z^2 + (X/2)^2)^{1/2};$$

Donde T es el tiempo total recorrido, V es la velocidad del medio, Z es la profundidad y X es la separación horizontal entre la fuente y el receptor. De la

fórmula se puede observar que para una velocidad constante, estaremos en el caso de una hipérbola (figura 4). Y en el caso particular X = 0, es decir, la fuente y el receptor ubicados en el mismo punto podemos extraer:

$$T(0) = 2*Z/V$$

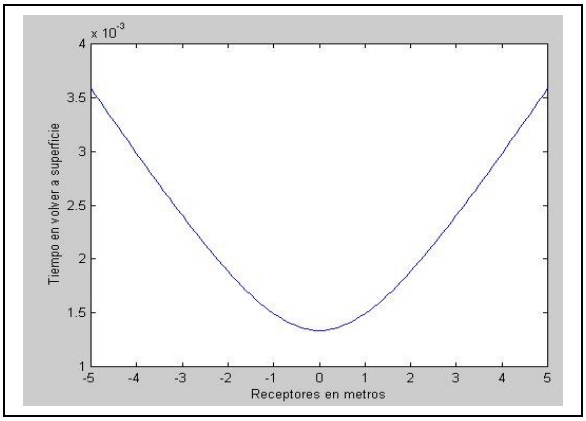


Figura 4. Tiempo de viaje de la reflexión de una onda en un medio de una capa horizontal con velocidad constante

Corrección por NMO (NORMAL MOVEOUT)

Según Yilmaz (1987) otra forma de calcular el tiempo de viaje de una onda reflejada en un medio sencillo de capas horizontales es; dado un punto medio M, el tiempo de viaje a lo largo del rayo proveniente de S hasta el punto en profundidad D, y llegue al receptor G (Figura 5) es T(x). Usando el teorema de Pitágoras, la ecuación del tiempo de viaje es:

$$T^2(x) = T^2(0) + x^2/v^2;$$

Donde x es la distancia (alejamiento) entre la posición de la fuente y el receptor, v es la velocidad de propagación del frente de onda y T(0) es dos veces el tiempo de viaje a lo largo de la vertical MD. La ecuación anterior describe el tiempo de viaje doble (two-way time) en función del "alejamiento".

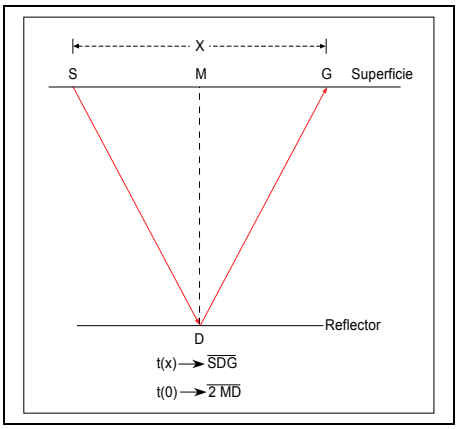


Figura 5. Geometría NMO para un reflector horizontal (extraído de Yilmaz, 1987)

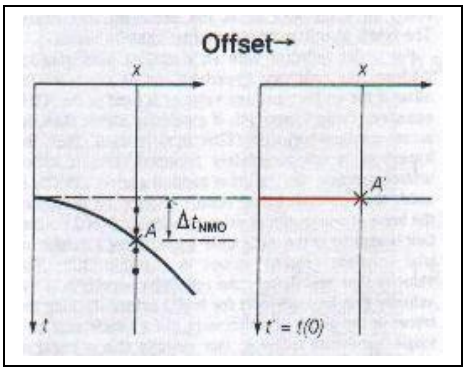


Figura 6. Corrección NMO. En la izquierda se ve el tiempo de la onda reflejada sin la corrección, y a la derecha el tiempo de llegada es haciendo t = T(0) (extraído de Yilmaz, 1987)

La diferencia entre el "two-way time" y dado por el "alejamiento" y el "two-way time" en la posición cero es llamada NMO. La corrección NMO es dada por la ecuación:

$$\Delta t_{NMO} = T(x) - T(0) = T(0)*\{[1 + (x/(V_{NMO}*T(0)))^2]^{1/2} - 1\}$$

El objetivo de esta corrección es convertir las hipérbolas en planos (figura 6), es decir, las superficies reflectoras en el dominio del tiempo, ajustarlas a los tiempos de llegada de la onda a alejamiento cero.

Ahora si consideramos un medio compuesto por varias capas (figura 7), cada una con su velocidad y espesor. Todas estas capas pueden definirse en términos de su tiempo doble en alejamiento cero. Las capas tendrán velocidades (v_1, v_2, … , v_n) donde N es el número de capas. Considerando el rayo que viaja en la trayectoria

SDG, asociado con el alejamiento X y el punto medio M, el tiempo de viaje se puede expresar mediante la ecuación:

$$T^2(x) = C_0 + C_1x^2 + C_2x^4 + \ldots + C_nx^{2n}$$

Donde $C_0 = T^2(0)$, $C_1 = 1/V_{rms}^2$ y C_2, C_3,… son funciones complicadas que dependen del espesor y la velocidad del intervalo. La velocidad rms es definida como:

$$V_{rms}^2 = \sum_{i=1}^{N} V_i^2 \Delta T_i(0);$$

Donde $\Delta T_i(0)$ es el tiempo doble de viaje a través del intervalo i y V_i es la velocidad en dicho intervalo, (Yilmaz, 1987).

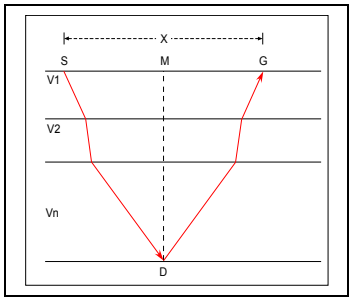


Figura 7. Reflexión en varias capas (extraído de Yilmaz, 1987)

Terminada la corrección por NMO la sección sísmica debe estar representada por tiempos de llegada para reflexiones con tiempos dobles de viaje en alejamiento cero. Y estos tiempos serán iguales a multiplicar la profundidad de cada superficie reflectora (tope o base del intervalo) por su *velocidad promedio*. Según Dobrin (1961) la velocidad promedio se define como velocidad interválica a través de una sección geológica integrada por más de una formación o litología y expresa el promedio ponderado de todas las velocidades que integran dicha formación. Esta velocidad se puede obtener por medio de tiros de verificación sísmica.

Resolución sísmica vertical

Según Sheriff (1991) es la menor separación que puede apreciarse entre dos eventos geológicos presentes en una imagen sísmica. El criterio de Rayleigh establece que el límite de la resolución sísmica es de ¼ de la longitud de onda.

Determinación de la velocidad media (velocidad average).

Según Dobrin (1961) para conocer la profundidad a partir de los tiempos de reflexión es necesario conocer la velocidad media del perfil desde la superficie hasta la capa u horizonte reflector. El intervalo de velocidad y, por ende, la velocidad promedio aumentan generalmente con la profundidad. Para ello se nombran tres técnicas; (1) el tiro de verificación sísmica; (2) registros continuos de velocidad; y (3) análisis de los registros de reflexión. Para los fines de este trabajo especial de grado solo se describirán las dos primeras técnicas.

Tiro de verificación sísmica

Según Schlumberger (2007) los tiros de verificación sísmica (check-shot survey) son un tipo de estudio sísmico hecho en pozo, el cual es diseñado para medir el tiempo de viaje de una onda desde la superficie hasta una profundidad conocida. Al producir una señal sísmica en la superficie de la Tierra y grabando el tiempo de llegada de onda en geófonos en posición conocida a lo largo de un pozo, se puede establecer relaciones que permiten conocer la velocidad.

Adicionalmente, los datos obtenidos pueden ser corregidos mediante registros sónicos y correlacionados con la sísmica de superficie para generar sismogramas sintéticos que permiten confirmar o corregir la interpretación sísmica.

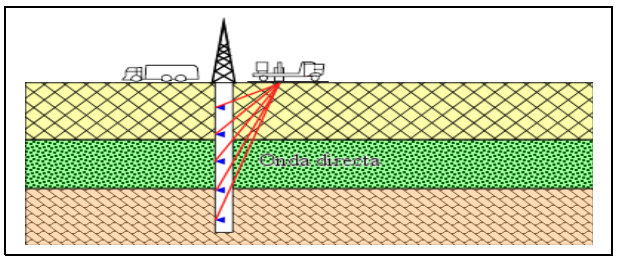


Figura 8. Diagrama de adquisición del tiro de verificación sísmica

Con los datos obtenidos de los tiros de verificación sísmica se pueden construir curvas para relacionar tiempo con la profundidad directamente, o mediante el cálculo de la pendiente de estas curvas podemos obtener la lentitud o su inverso que es la velocidad promedio. A estas curvas se les conoce como curvas T-Z (T-D en ingles) o curvas tiempo profundidad.

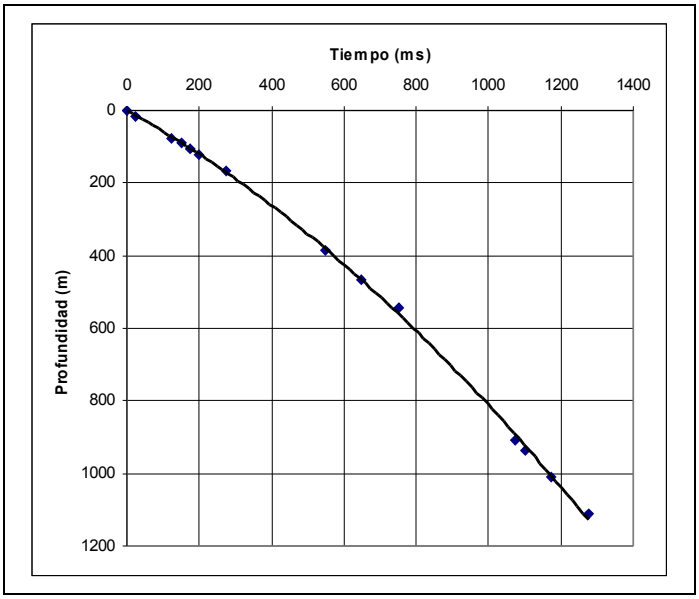


Figura 9. Ejemplo de curva T-Z

Registros continuos de velocidad (registros sónicos)

Los registros recogen, como función de la profundidad, los tiempos de recorrido de las ondas sísmicas a través de la formación, entre una fuente de sonido en un extremo del aparato y un receptor en el otro extremo, o entre dos receptores distanciados de 0.3 a 3 metros (Dobrin, 1961)

La herramienta sónica mide la velocidad interválica del sonido en las formaciones penetradas por un pozo, en intervalos dados por la longitud de dicha herramienta.

Velocidad interválica (intervallic velocity)

Según Sheriff (1991) la velocidad interválica es la velocidad promedio de la onda sísmica a través de un medio de características similares, es decir, es la velocidad con que se desplaza una onda sísmica desde el tope hasta la base de una formación de interés. Esta se puede obtener mediante la integración del registro sónico a lo largo de todo el espesor de la formación o bien, mediante el conocimiento de las velocidades promedio y profundidades del tope y la base de la formación obtenidos del estudio de verificación sísmica.

Tabla 3. Tipos de velocidades.

Tipo	Concepto	Aplicación
Velocidad promedio $V = P/T$	Es la velocidad media de propagación de la onda P desde la fuente hasta una capa geológica.	Conversiones tiempo a profundidad en áreas con poca variación lateral.
Velocidad interválica $V_n = (P_n – P_{(n-1)})/(T_n – T_{(n-1)})$	Es la velocidad media de propagación de onda entre dos capas geológicas.	Permite calcular los espesores de capas geológicas. Así como definir dominios de velocidad a determinadas litologías.
Velocidad NMO $V_{nmo} = V/\cos\theta$ Donde θ es el buzamiento del la capa.	Es la velocidad utilizada en procesamiento para la corrección por NMO.	Se usa en procesamiento para la corrección del efecto hiperbólico producido por la distancia de la fuente al receptor.

Modelos de velocidad

Una conversión exitosa de unidades de tiempo a unidades de profundidad tiene como clave la construcción de un modelo de velocidad (Söllner et al, 2004).

Además los modelos de velocidad proveen la relación entre las unidades de tiempo y unidades de profundidad (Bartel et al, 2006). Los modelos de velocidad son formas de representar la distribución de velocidades en el subsuelo y estos pueden ser expresados en 1-D como es el caso de los registros de velocidad en pozo, por secciones (2-D), por cubos de velocidad o mapas de velocidad en distintas profundidades (3-D).

MÉTODOS MATEMÁTICOS

Para el ajuste de datos de las tablas tiempo-profundidad y la generación de los modelos de velocidad como funciones en 3D, es importante considerar ciertas operaciones y métodos matemáticos. En esta sección se definió los métodos matemáticos utilizados para el ajuste, así como consideraciones teóricas de la estadística descriptiva necesaria para las pruebas de funcionamiento del programa.

Ajuste de datos

Con frecuencia en la ciencia y la ingeniería se presenta el problema de ajustar un conjunto de datos proveniente de mediciones que se desea ajustar a funciones que describan la relación entre variable dependientes e independientes, es decir, hallar funciones $y = f(x)$ para ajustes bidimensionales y funciones $z = f(x,y)$ para ajustes tridimensionales. Para ello se han desarrollado modelos lineales y no lineales que permiten el ajuste y que dependen del problema y la naturaleza de los datos.

Para resolver el problema de ajuste de datos en este trabajo, se utilizo para datos en dos dimensiones el ajuste por mínimos cuadrados. Y para el problema superficial se utilizó estimadores lineales ponderados.

Ajuste por mínimos cuadrados

Supóngase el conjunto de puntos (x_k,y_k), siendo k = 1, 2, …,n. Sea una base de m funciones linealmente independientes $f_j(x)$, con j = 1, 2, …, m. Queremos encontrar una función *f* combinación lineal de las funciones base tal que $f(x_k) = y_k + \varepsilon$; donde ε es el error. Esto es:

$$f(x) = \sum_{j=1}^{m} c_j f_j(x)$$

Se trata de hallar los m coeficientes $\boldsymbol{c_j}$ que aproximen la función f(x) lo mejor posible a los puntos $\boldsymbol{(x_k,y_k)}$. El criterio de mejor aproximación puede variar, pero en general se trata de minimizar el error, de forma tal que $\varepsilon_k = y_k - f(x_k)$.

Estimadores lineales ponderados

La idea básica es estimar el valor de una variable regionalizada en una posición *u* donde no conocemos el valor verdadero. Si planteamos:

$$Z^*(\mathbf{u}) = a + \sum_{i=1}^{n} \lambda_i \cdot Z(\mathbf{u}_i)$$

Donde Z*(*u*) es el valor estimado para la posición u, $\{Z(u_i), i=1...n\}$ son los valores de los datos en las posiciones $\{u_i, i=1...n\}$, *a* es un coeficiente aditivo y $\{\lambda_i, i=1...n\}$ son ponderadores.

Los factores que afectan la asignación de los ponderadores son: la cercanía que le da peso a la distancia del dato; la redundancia entre los valores de datos que estudia las repeticiones y genera promedios ponderados; continuidad o variabilidad espacial de alguna propiedad; y la anisotropía que es la propiedad general de la materia según la cual determinadas propiedades físicas varían según la dirección en que son examinadas.

Los siguientes estimadores ponderados utilizan los tres primeros factores nombrados anteriormente.

Estimador del vecino más cercano

Este estimador atribuye toda la ponderación al dato más cercano al sitio a estimar ($\lambda i = 1$ para el dato más cercano, $\lambda i = 0$ para los otros datos, $a = 0$) Supóngase el conjunto de puntos (x_k,y_k,z_k), siendo k = 1, 2, …,n. podemos definir una función z = f(x,y), que estudie las distancias del conjunto y asigne a f(x,y) el z_k mas cercano.

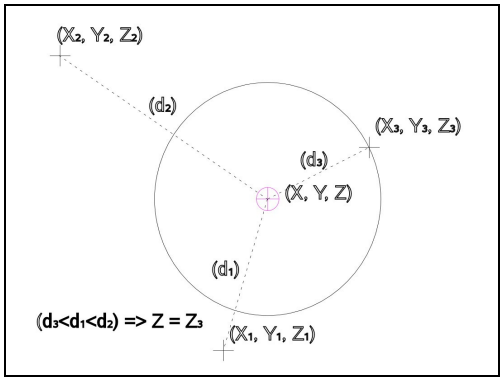


Figura 10. Algoritmo del estimador del vecino más cercano.

Estimador de valor promedio

Es un estimador que calcula el promedio de los datos ubicados alrededor de z a conocer. Este estimador debe evaluarse usando un radio de búsqueda para evitar que todos los datos de la muestra sean calculados.

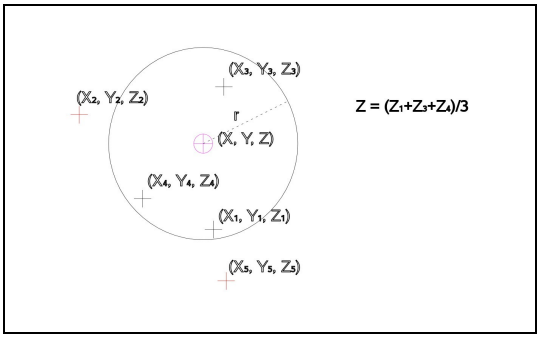


Figura 11. Algoritmo del ajuste por el promedio dentro de un área.

Inverso de la distancia lineal

Estimador del inverso de la distancia atribuye a cada dato una ponderación proporcional al inverso de su distancia al sitio a estimar. Existen variantes, donde se eleva la distancia a una cierta potencia.

$$a=0 \qquad \lambda_i = \frac{\frac{1}{c+d_i^w}}{\sum_{i=1}^{n} \frac{1}{c+d_i^w}}$$

Donde d_i es la distancia entre el dato n_i y la posición que se está estimando, c es una constante pequeña, y w es una potencia usualmente comprendida entre 1 y 3.

PROGRAMACIÓN

El computador es para el ingeniero una fundamental herramienta de trabajo la cual puede servirse para resolver múltiples problemas que se presentan en la elaboración de proyectos. En gran número de situaciones se dispone de programas ya diseñados, pero en otros casos hay que diseñar herramientas propias y es aquí donde se asume el papel de programador.

Según Deitel y Deitel (2004) existen diversos lenguajes de programación, pero en general se pueden clasificar en tres tipos: Lenguajes de máquina, que son aquellos que pueden entender las computadoras de manera directa, está definido por el diseño del hardware y consisten en cadenas de números [unos (1) y ceros (0)]; lenguajes ensambladores, este lenguaje se basa en abreviaturas en inglés que permiten una escritura mas clara de un programa y luego son traducidos a lenguaje de máquina; y finalmente los lenguajes de alto nivel, que permiten una escritura más humana y facilitan la creación de tareas de alta complejidad. Los leguajes de alto nivel son traducidos a lenguaje de maquina mediante programas conocidos como compiladores.

Lenguaje C y C++

"C es un lenguaje de programación creado en 1969 por Ken Thompson y Dennis M. Ritchie en los Laboratorios Bell como evolución del anterior lenguaje B, a su vez basado en BCPL. Al igual que B, es un lenguaje orientado a la implementación de Sistemas Operativos, concretamente Unix. C es apreciado por la eficiencia del código que produce y es el lenguaje de programación más popular para crear software de sistemas, aunque también se utiliza para crear aplicaciones." (Wikipedia, 2007)

Deitel y Deitel (2004) describen a al lenguaje C++ como un C mejorado, desarrollado por Bjarne Stroustrup en los laboratorios Bell. Entre varias características que pulen al lenguaje C, destaca la capacidad de realizar programación orientada a objetos. Actualmente C++ se ha convertido en el lenguaje dominante en la industria y en las universidades.

Entorno integrado de desarrollo (Integrated Development Environment)

Un Entorno integrado de desarrollo (IDE) Es una aplicación compuesta por un conjunto de herramientas útiles para la programación. Un entorno IDE puede ser exclusivo para un lenguaje de programación o bien, poder utilizarse para varios. Suele consistir de un editor de código, un compilador, un depurador (debugger) y un constructor de interfaz gráfica (GUI).

Programación estructurada y programación orientada a objetos

La programación estructurada es una forma de escribir programación de ordenador de forma clara, para ello utiliza únicamente tres estructuras: secuencial, selectiva e iterativa; siendo innecesario y no permitiéndose el uso de la instrucción o instrucciones de transferencia incondicional (GOTO).

El principal inconveniente de este método de programación, es que se obtiene un único bloque de programa, que cuando se hace demasiado grande puede resultar problemático su manejo, esto se resuelve empleando la programación modular, definiendo módulos interdependientes programados y compilados por separado. Un

método un poco más sofisticado es la programación por capas, en la que los módulos tienen una estructura jerárquica muy definida y se denominan capas.

La Programación Orientada a Objetos (POO u OOP según siglas en inglés) es un paradigma de programación que define los programas en términos de "clases de objetos", objetos que son entidades que combinan *estado* (es decir, datos), *comportamiento* (procedimientos o métodos) e *identidad* (propiedad del objeto que lo diferencia del resto). La programación orientada a objetos expresa un programa como un conjunto de estos objetos, que colaboran entre ellos para realizar tareas. Esto permite hacer los programas y módulos más fáciles de escribir, mantener y reutilizar. De esta forma, un objeto contiene toda la información, que permite definirlo e identificarlo frente a otros objetos pertenecientes a otras clases. A su vez, dispone de mecanismos de interacción que favorecen la comunicación entre objetos de una misma o de distintas clases, y en consecuencia, el cambio de estado en los propios objetos. Esta característica lleva a tratarlos como unidades indivisibles, en las que no se separan Los datos y los métodos (Wikipedia, 2007).

Funciones y bibliotecas

Según Deitel y Deitel (2004) la mejor manera de desarrollar y mantener un programa grande es construirlo a partir de piezas pequeñas o módulos, los cuales son más manejables que el programa original. En el lenguaje C a estos módulos se les llama funciones. Las funciones cumplen tareas determinadas y se construyen a partir de instrucciones simples o de otras funciones con el objetivo de crear funciones capaces de realizar tareas más complejas.

Las bibliotecas son un conjunto de funciones elaboradas por el usuario u otros usuarios y cuyo objetivo es proveer al programador de funciones preescritas, esto permite concentrar los esfuerzos de la programación a resolver la tarea principal sin la necesidad de escribir funciones ya existentes.

Archivos en C/C++

“El almacenamiento de datos dentro de variables y arreglos es temporal; todos esos datos se pierden cuando termina el programa. Los archivos se utilizan para retener permanentemente grandes cantidades de datos. Las computadoras almacenan los archivos en dispositivos secundarios de almacenamiento, en especial en dispositivos de disco.” (Deitel y Deitel, 2004)

Deitel y Deitel (2004) mencionan que el lenguaje C no impone una estructura a un archivo. Por lo tanto, el registro de un archivo no existe en el leguaje C, esto implica que el programador debe proporcionar la escritura del archivo para poder leer y cumplir con los objetivos de una aplicación en especial. El lenguaje C cuenta con una biblioteca especial para manejar (crear, leer y escribir) archivos. Esto nos permite crear programas para abrir cualquier tipo de archivo, siempre y cuando se conozca la estructura del mismo.

CAPÍTULO III

MARCO METODOLÓGICO

Para la finalización exitosa del proyecto se cumplió dos áreas. Una de desarrollo del programa, que inició desde la selección de las herramientas de desarrollo, el diseño de las variables que representan el problema, diseño de las funciones que representen los fenómenos físicos involucrados y culmina en la compilación del programa. Y otra, la prueba del programa. En esta fase primero se observó la expresión de histogramas de frecuencia de coordenadas de tiempo o profundidad datos sintéticos e ideales. Luego se genero dos ejemplos sintéticos con configuraciones distintas y muestreos aleatorios. Finalmente se repitió las pruebas con datos reales, una geología carente de dominio estructural y otro ejemplo con una estructura geológica compleja.

Selección de herramientas de desarrollo

En la selección de las herramientas de desarrollo utilizadas para la elaboración del trabajo se considero las siguientes características:

- Ser actuales.
- De uso sencillo.
- Con capacidad de manejo de archivos. Lectura y escritura de datos.
- Potencial para programación orientada a objetos.
- Alta portabilidad, lo cual permite la compilación en distintos sistemas operativos con pequeñas modificaciones al código.

El lenguaje C/C++ cumple con las características de robustez, versatilidad y portabilidad. Para el desarrollo del entorno grafico en Windows se utilizó el IDE Borland Builder C++ v.6 (figura 12) y el código fue escrito usando el Bloodshed **Dev-C++** (figura 13) que es un IDE para programar en lenguaje C/C++.

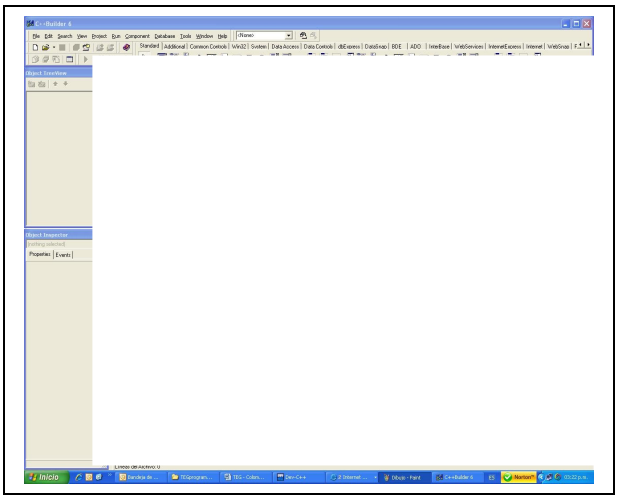


Figura 12. Ambiente de trabajo del Borland Builder C++ v.6 ®

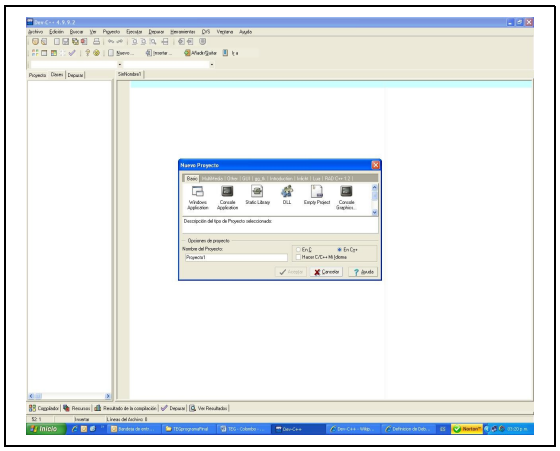


Figura 13. Ambiente de trabajo del Bloodshed Dev-C++

Desarrollo de código fuente

La elaboración del código fuente se realizó mediante la técnica del pseudocódigo. En esta etapa se escribió la idea básica del funcionamiento del programa en términos humanos, es decir, no se usó palabras reservadas por el lenguaje de programación ni se creo funciones ni datos.

Archivos UnitMain

El archivo UnitMain.cpp contiene la definición principal del programa. Aquí se escribió la función main que es imprescindibles en cualquier programa C/C++, esta representa el punto de inicio y fin de su ejecución. En este archivo se escribió las funciones de lectura y escritura de datos. Adicionalmente se declararon las variables globales y temporales necesarias para realizar los cálculos. En el archivo UnitMain.h se generó, con ayuda del IDE Borland Builder C++ v.6 ®, la definición de la forma visual (botones, cuadros, gráficas, etc). La integración de los archivos UnitMain.cpp y UnitMain.h definen la interacción con las funciones de los archivos de cabecera.

Tipos de dato

Para modelar la información de pozo y superficies a trabajar se creó estructuras de datos en un archivo de cabecera distinto al código fuente principal. Este archivo llamado Estructuras.h define dos tipo de dato, uno Tipo_Pozo y otro Tipo_Superficie.

La Estructura Tipo_Pozo fue diseñada de la siguiente manera:

1. Una variable tipo char llamada Nombre. Esta variable contiene el nombre del pozo, el cual puede contener 16 caracteres. Esta variable no es obligatoria para el desarrollo del programa.
2. Un arreglo tipo double llamado TablaTZ. Este arreglo posee la información que relaciona el tiempo con la profundidad. Esta información puede ser de datos de pozo o una tabla generada mediante la interpretación de pozo y sísmica.
3. Dos variable tipo float llamadas Este y Norte. En estas variables se almacena la ubicación del pozo en coordenadas rectangulares.
4. Dos variables tipo int. Una de ellas llamada Numero indica un numero identificador del pozo dentro de la estructura Tipo_Pozo. La otra llamada Muestras indica la cantidad de datos que contiene el arreglo TablaTZ.

La estructura Tipo_Superficie fue diseñada usando:

1. Una variable tipo char. Esta contiene el nombre de la superficie y contiene hasta 32 caracteres.
2. Seis variables tipo double, dos de ellas X y Y representan las coordenadas planas de cada dato; T, Z, Vpromedio y Vintervalica representan el tiempo, la profundidad, la velocidad promedio y la velocidad interválica en función de las coordenadas X y Y.

Archivo Superficie.h

Este archivo contiene la cabecera de todas las funciones necesarias para la manipulación del dato Tipo_Superficie. Aquí se redacto el código para la generación de cuadriculas, con límites que pueden ser estipulados por el usuario o las datos de entrada. Además de la generación de las cuadrículas se escribió las operaciones básicas (suma, resta y multiplicación) que permiten la interacción entre ellas. Con las operaciones escritas aquí se pudo generar arreglos para el control de datos en superficie, así como operaciones pata obtener las superficies en tiempo, superficies de velocidades, superficies en profundidad y superficies isópacas.

Archivo Velocidades.h

Este archivo fue escrito con el objetivo de tener funciones que controlen los parámetros asociados al tiempo, velocidad y profundidad. Además tiene funciones que permiten la conversión entre unidades (metros a pies o segundos a milisegundos) y pasa de tiempo doble a tiempo simple. Esta función también tiene el control de la curva tiempo-profundidad extraída de la información de pozo.

Prueba de eficiencia del programa

El programa fue sometido a cuatro pruebas para comprobar su funcionamiento: dos de ellas se realizó con datos sintéticos, es decir, datos artificiales de los cuales se conocen su respuesta tanto en tiempo como en profundidad; y otras dos pruebas que fueron realizadas con datos reales previamente interpretados y convertidos en profundidad; estas últimas pruebas se realizarán para enfrentar la eficiencia del programa con trabajos de interés para la empresa.

La características probadas fueron: eficiencia en zonas geológicamente sencillas y complejas, el éxito con respecto a la distribución de los pozos, y relación cantidad y distribución de pozos contra complejidad geológica.

Para ello se realizó estudios de estadísticas descriptivas en los resultados obtenidos a partir del programa elaborado contra los resultados anteriores.

Histogramas de frecuencia de datos sintéticos e ideales

Esta prueba fue realizada con el objetivo de establecer patrones de histogramas de frecuencias a datos ideales. Con esto poder comparar e interpretar los histogramas obtenidos en las pruebas de conversión de tiempo profundidad.

Esta prueba se realizó con datos generados con funciones Y=f(X) en una hoja de cálculo; donde X = {1, 2, …, 4000} y Y fue una función de X (Apéndice 1).

A cada caso de Y se le realizó un estudio de estadística descriptiva (tabla 4) y se le generó un histograma de frecuencia (figura 15)

En este ejemplo, se trabajó con Y simulando una capa plana inclinada. Podemos observar en sus estadísticas descriptivas que la mediana es similar a la media aritmética.

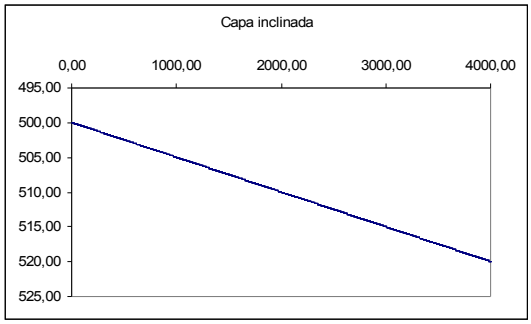


Figura 14. Configuración de una capa plana inclinada.

Tabla 4. Estadísticas de la capa plana inclinada.

Capa inclinada	
Media	510,0025
Mediana	510,0025
Moda	#N/A
Desviación estándar	5,774224335
Varianza de la muestra	33,34166667

En el histograma de frecuencia se aprecia que en la configuración de la capa plana inclinada el histograma de frecuencia presenta una distribución uniforme, es decir, no presenta modas ni subpoblaciones dentro de la muestra.

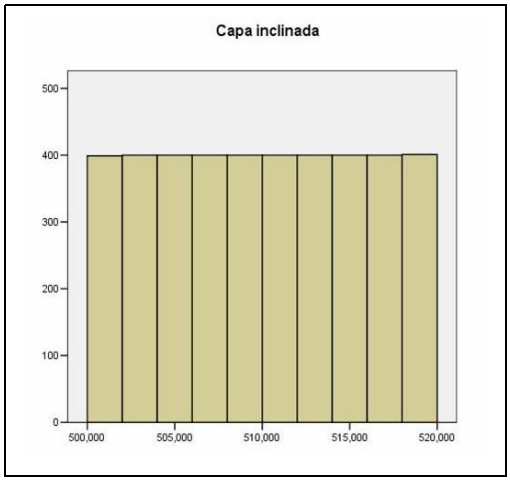


Figura 15. Histograma de una capa plana inclinada. El histograma presenta una distribución uniforme.

Prueba con datos artificiales y reales

Usando hoja de cálculo se produjo una distribución aleatoria de coordenadas X y Y. Para cada uno de los ejemplos se genero tres capas con tiempo, velocidad y profundidad conocida y se construyó el mapa de cada ejemplo.

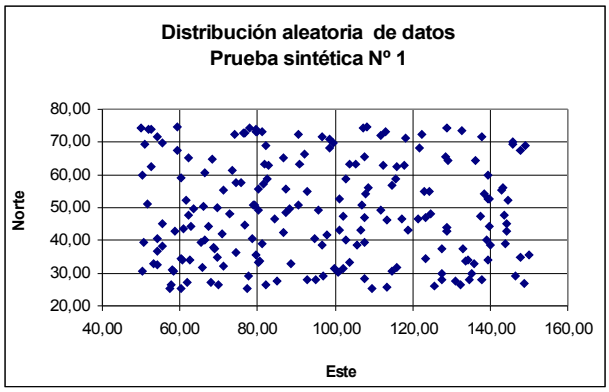


Figura 16. Distribución aleatoria de datos sintéticos.

Tabla 5. Muestra de los datos generados.

X	Y	TWT1 (ms)	Zteorico1(m)	TWT2 (ms)	Zteorico2(m)	TWT3 (ms)	Zteorico3(m)
96,69	38,56	366,73	220,04	733,45	513,42	1350,18	1130,14
58,41	30,58	271,02	162,61	542,04	379,43	1063,07	900,45
68,41	64,92	296,02	177,61	592,05	414,43	1138,07	960,46
68,80	37,60	296,99	178,20	593,99	415,79	1140,98	962,78
123,31	34,53	433,27	259,96	866,53	606,57	1549,80	1289,84
127,64	29,73	444,10	266,46	888,20	621,74	1582,30	1315,84

Tabla 6. Velocidades de los datos sintéticos

	Velocidades (m/s)
V1	1200
V2	1600
V3	2000

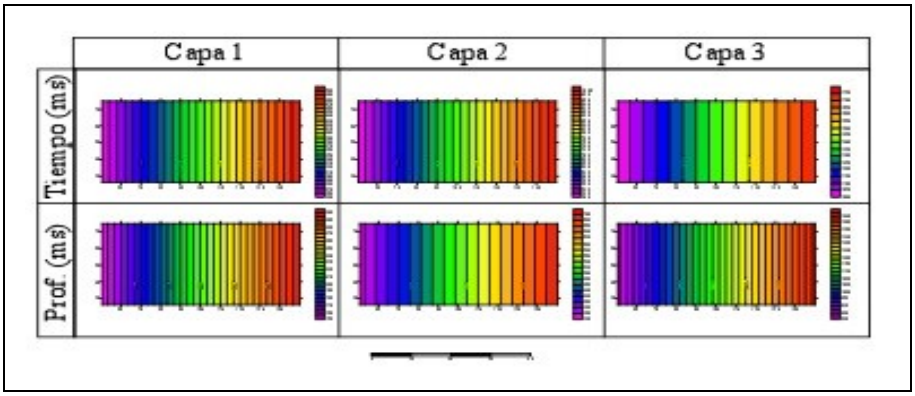
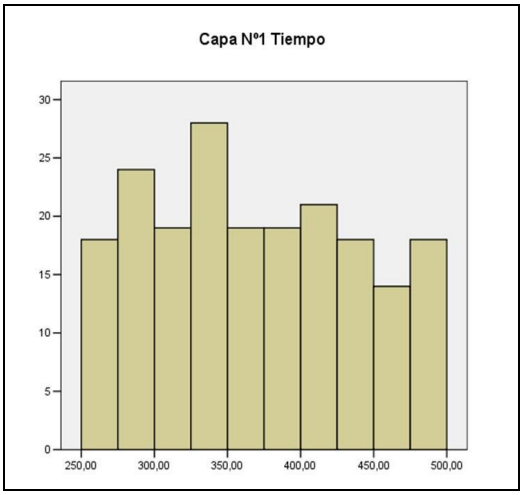
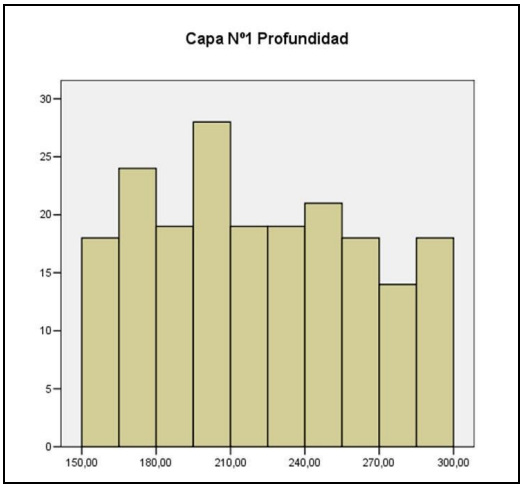


Figura 17. Prueba sintética, mapas en tiempo y profundidad de las tres capas.

Posteriormente se generó los histogramas de frecuencia de cada superficie. Y se comprobó la relación entre los histogramas de tiempo y profundidad (Figura 18). En esta etapa no es necesario ver el comportamiento de la velocidad, ya que esta es asignada.



18.a



18.b

Figura 18. Histogramas de frecuencia de los datos sintéticos. Arriba histograma de frecuencia en tiempo (ms), abajo histograma de frecuencia en profundidad (m).

Usando los datos generados. Se construyó tablas de tiempo-profundidad sintéticas para iniciar un estudio de velocidad.

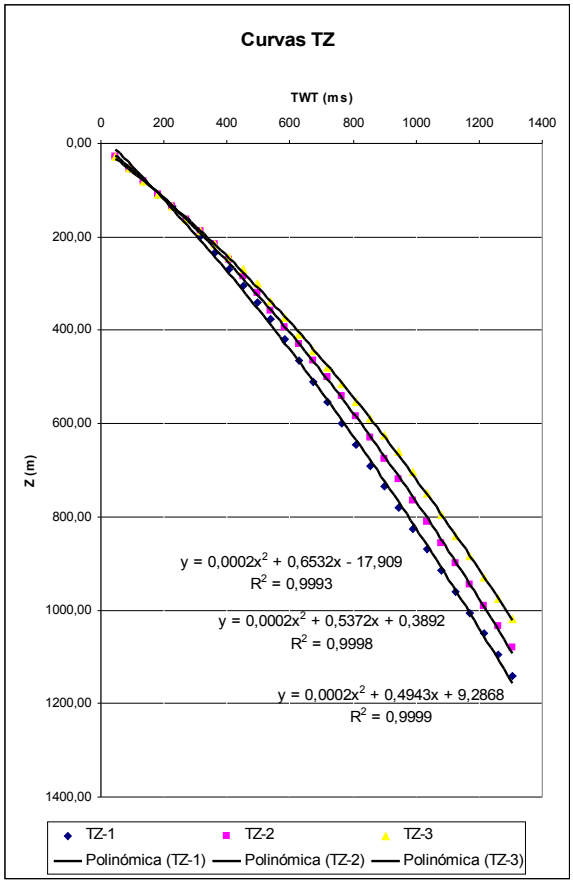


Figura 19. Curva TZ de los datos sintéticos.

Luego se usó el programa de conversión tiempo profundidad. Se cargó en el programa las tablas tiempo profundidad con lo que se obtuvo los polinomios de segundo grado que mejor se ajustan a los datos.

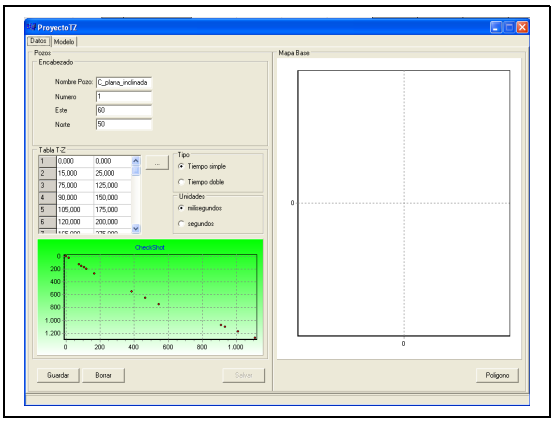


Figura 20. Carga de pozo en el programa de conversión.

Luego se cargó en el programa los datos de las superficies en tiempo y aplicando técnicas de estimadores lineales ponderados se generó una nueva cuadrícula con un nuevo muestreo figura 21). Se generó histogramas de frecuencia (figura 22) a las cuadriculas en tiempo y se comparó con el histograma de los datos originales.

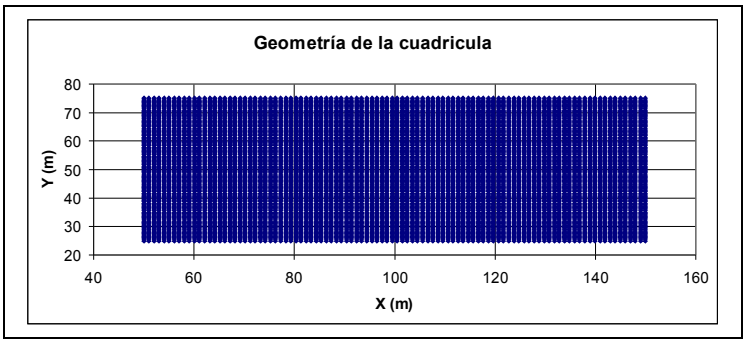


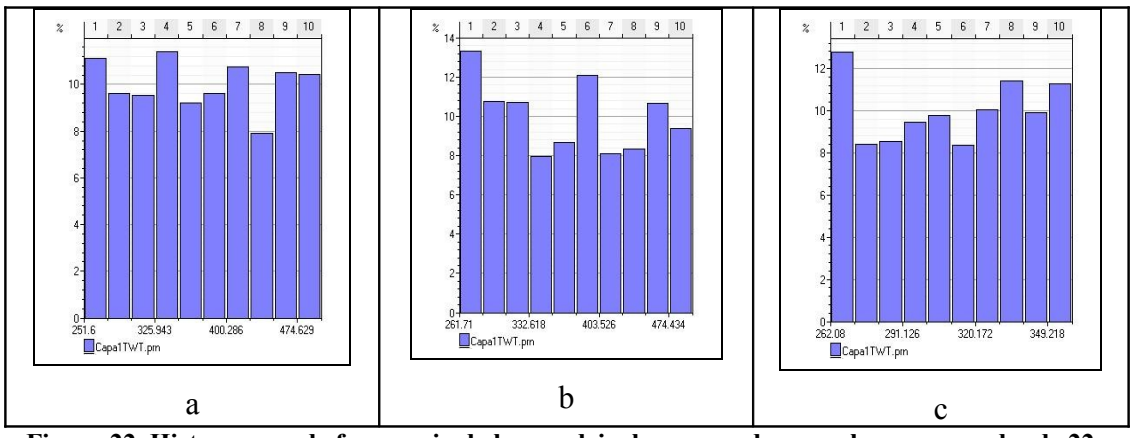
Figura 21. Distribución de los datos dentro de la cuadricula.

Mediante las estadísticas y la visualización cualitativa de los histogramas de los datos originales y las cuadriculas que se ajustaron, se decidió el mejor arreglo que modelará los datos de entrada. Para ello no solo se tomó como criterio el histograma que mejor ajustara a lo datos, también se tomó en cuenta que la cuadrícula tenía que aproximarse a una capa inclinada. Considerando los modelos de histogramas en condiciones ideales, y que este caso es el de capas planas inclinadas, se decidió

trabajar con el ajuste del vecino más cercano ya que es el que presentó la distribución más uniforme (figura 22.a).

Tabla 7. Estadísticas de los datos de entrada y las superficies de ajuste

Estadísticas	Datos Originales	Vecino más cercano	Promedios dentro de un área	Inverso de la distancia
Media	373,04	374,97	374,47	310,97
Mediana	374,07	374,36	376,02	311,69
Moda	258,52	498,07	482,03	263,01
Desviación Estándar	70,45	72,93	69,97	29,19
Varianza	4963,11	5318,59	4895,87	852,06
Mínimo	254,48	251,60	261,71	262,08
Máximo	498,78	499,41	498,07	358,90



a b c

Figura 22. Histogramas de frecuencia de las cuadriculas generadas con el programa, donde 22.a es la cuadricula generada con el método del vecino más cercano, 22.b el promedio dentro de un área y 22.c es el estimador del inverso de la distancia.

Los modelos de velocidad fueron construidos generando cuadriculas a partir de los datos de velocidad extraídos de pozo. En los casos necesarios se realizó el cambio de unidades de metros a pies o viceversa.

Las profundidades fueron obtenidas mediante la relación Pn = Vn x Tn para las velocidades promedio y la relación Pn = P(n-1) + Vn x (Tn-T(n-1)) (Tabla 8),

luego se generó distintas cuadriculas de velocidad usando los tres métodos de ponderación para ajustar datos y distintos radios de búsqueda.

Tabla 8. Velocidades obtenidas a partir del polinomio de segundo grado

Capa 1						
X	Y	TWT	Profundidad	Profundidad tabla TZ	Velocidad Promedio	Velocidad Interválica
60,00	46,00	275,00	165,00	179,38	1304,56	1304,56
100,00	64,00	375,00	225,00	229,96	1226,48	1226,48
140,00	74,00	475,00	285,00	289,20	1217,70	1217,70
Capa2						
X	Y	TWT	Profundidad	Profundidad tabla TZ	Velocidad Promedio	Velocidad Interválica
60,00	46,00	550,00	165,00	405,31	1473,86	1643,1533
100,00	64,00	750,00	225,00	515,79	1375,44	1524,4
140,00	74,00	950,00	285,00	659,37	1388,15	1558,6
Capa 3						
X	Y	TWT	Profundidad	Profundidad tabla TZ	Velocidad Promedio	Velocidad Interválica
60,00	46,00	1075,00	345,00	916,19	1704,54	1946,1965
100,00	64,00	1375,00	405,00	1117,16	1624,97	1924,4
140,00	74,00	1675,00	465,00	1398,36	1669,69	2038,6

Finalmente se calculó la diferencia entre los métodos de conversión, es decir, la diferencia entre la profundidad calculada usando las velocidades promedio y la profundidad hallada por las velocidades interválicas. Se calculo el error y se graficaron ambas superficies.

Tabla 9. Error porcentual entre la profundidad hallada por velocidades promedio y velocidades interválicas.

Capa	Error
1	0,0000001548%
2	0,0001632618%
3	0,2281237532%

CAPÍTULO IV

ANÁLISIS DE RESULTADOS

Modelos de histogramas de frecuencia de datos sintéticos

En este experimento se observó la respuesta de los histogramas de frecuencia a datos sintéticos de distintas geometrías dadas. A su vez se consideró los resultados de las medidas de tendencia central y dispersión.

En el primer caso podemos observar un comportamiento ideal de una capa plana y horizontal. Se observó que coincide el valor de las medidas de tendencia central y las medidas de dispersión tuvieron valor 0. El histograma de frecuencia (figura 24) indicó una barra, es decir, el 100 % de los datos pertenecen a una clase.

Tabla 10. Estadísticas de la capa plana horizontal.

Capa plana horizontal	
Media	500
Mediana	500
Moda	500
Desviación estándar	0
Varianza de la muestra	0

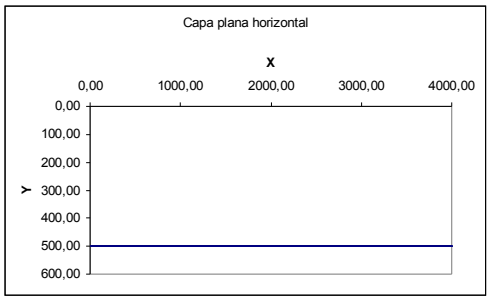


Figura 23. Configuración de de una capa plana horizontal.

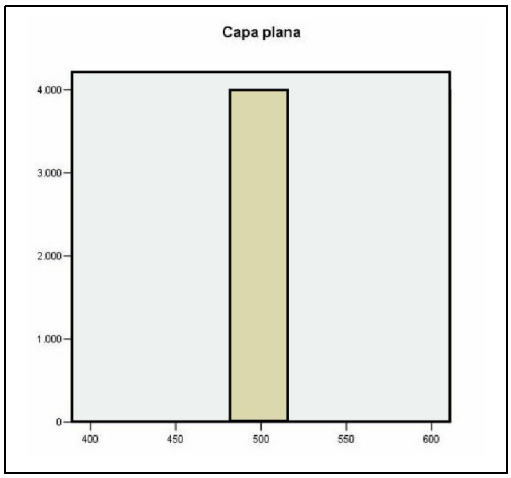


Figura 24. Histograma de de una capa plana horizontal. Solo posee una moda.

El segundo caso se estudió una capa plana, horizontal y fallada. Se observó que coincide el valor de la moda y la mediana. El histograma de frecuencia (figura 26) mostró dos barras, lo que representa dos poblaciones bien identificadas.

Tabla 11. Estadísticas de la capa plana horizontal fallada.

Capa plana fallada	
Media	500,05
Mediana	600
Moda	600
Desviación estándar	100,0124898
Varianza de la muestra	10002,49812

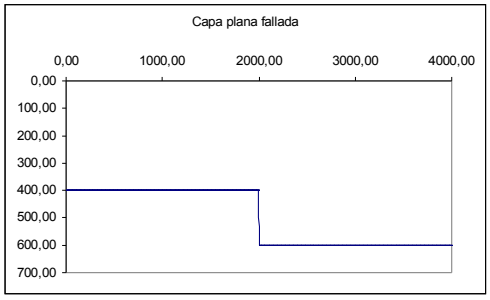


Figura 25. Configuración de una capa plana horizontal fallada.

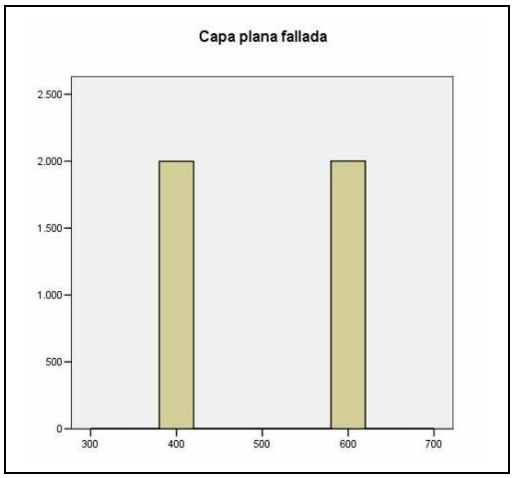


Figura 26. Histograma de una capa plana horizontal fallada. El histograma presenta dos modas.

El tercer caso estudió una capa plana e inclinada. Se observó que coincide el valor de la media aritmética y la mediana, mientras que no existe una moda ya que todos los datos poseen la misma frecuencia de aparición. El histograma de frecuencia (figura 28) una distribución uniforme, es decir, todos los eventos tienen la misma probabilidad de aparición.

Tabla 12. Estadísticas de la capa plana inclinada.

Capa inclinada	
Media	510,0025
Mediana	510,0025
Moda	#N/A
Desviación estándar	5,774224335
Varianza de la muestra	33,34166667

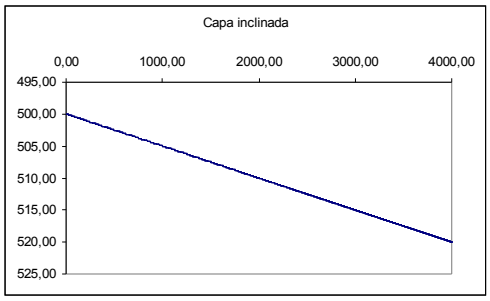


Figura 27. Configuración de una capa plana inclinada.

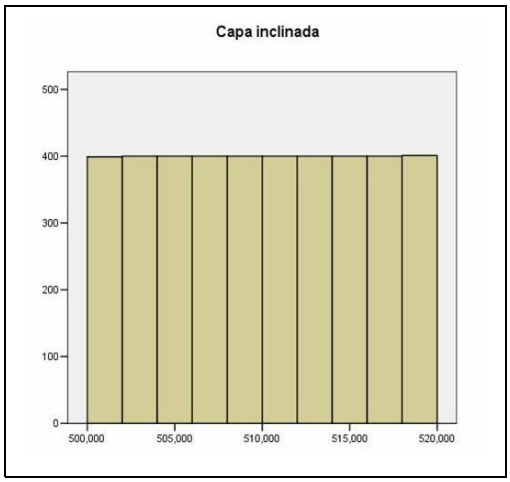


Figura 28. Histograma de una capa plana inclinada. El histograma presenta una distribución uniforme.

El caso una capa plana, inclinada y fallada. Se observó que coincide el valor de la media aritmética y la mediana, mientras que no existe una moda ya que todos los datos poseen la misma frecuencia de aparición. El histograma de frecuencia (figura 30) dos distribuciones uniformes separadas por un vació en los datos. Ese fenómeno se debe a la aparición de la falla en los datos sintéticos.

Tabla 13. Estadísticas de la capa plana inclinada fallada.

Capa inclinada fallada	
Media	700,1
Mediana	800,05
Moda	#N/A
Desviación estándar	208,1926056
Varianza de la muestra	43344,16104

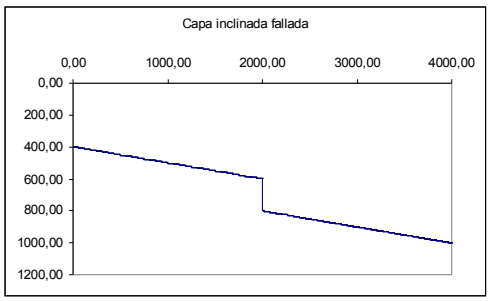


Figura 29. Configuración de una capa plana inclinada fallada.

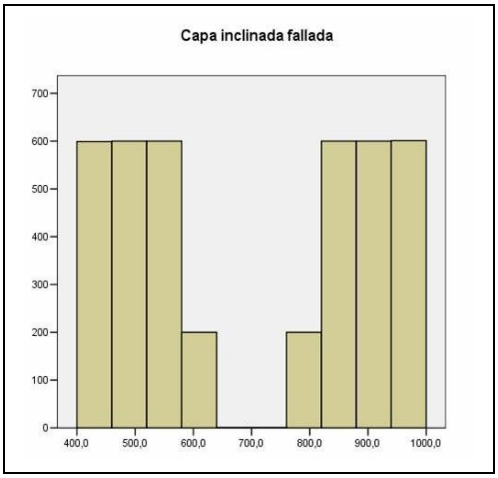


Figura 30. Histograma de una capa plana inclinada fallada. El histograma presenta dos modas, cada moda posee una distribución uniforme.

Cuando se repitió el mismo experimento en una capa horizontal fallada múltiples veces, se observó que el histograma presenta múltiples poblaciones. Con este fenómeno y los experimentos anteriores, así como los próximos experimentos sometidos a fallas (parábola y elipse sometidas a fallas), se pudo observar que las distribuciones de las poblaciones por ende los histogramas son sensibles a las deformaciones frágiles.

Tabla 14. Estadísticas de la capa plana horizontal fallada varias veces.

Capa horizontal múltiplemente fallada	
Media	509,7025
Mediana	500
Moda	500
Desviación estándar	56,45525478
Varianza de la muestra	3187,195793

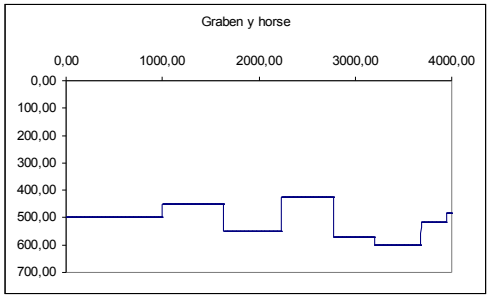


Figura 31. Configuración de una capa plana horizontal fallada varias veces.

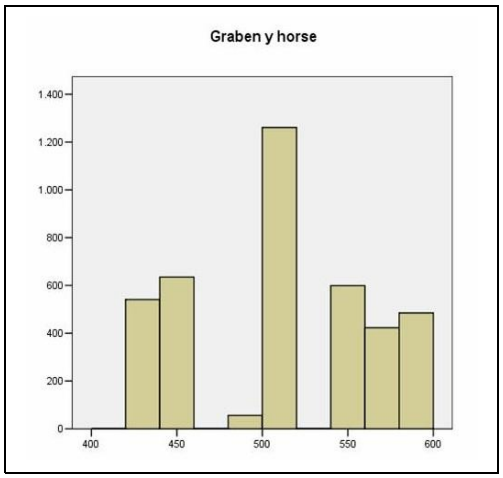


Figura 32. Histograma de una capa plana horizontal fallada varias veces. El histograma presenta una moda representativa y otras de menor importancia.

La distribución de la capa parabólica presenta un aumento en la dirección de crecimiento. En la naturaleza es difícil conseguir este modelo pero se puede comparar parcialmente con cuencas o terminaciones sísmicas en secuencias sedimentarias. Cuando se observa la distribución en las parábolas falladas, es posible distinguir el

rompimiento en el histograma y además es comparable con facies marinas asociadas a sistemas encadenados de bajo nivel.

Tabla 15. Estadísticas de la capa parabólica.

Parábola	
Media	542,171556
Mediana	544,726949
Moda	#N/A
Desviación estándar	14,90353621
Varianza de la muestra	222,1153915

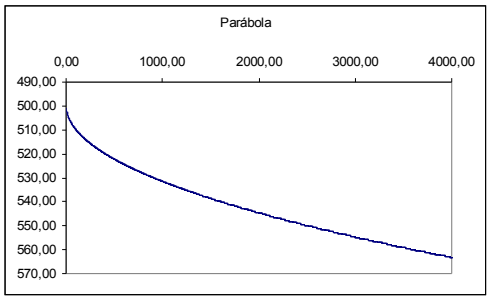


Figura 33. Configuración de una capa parabólica.

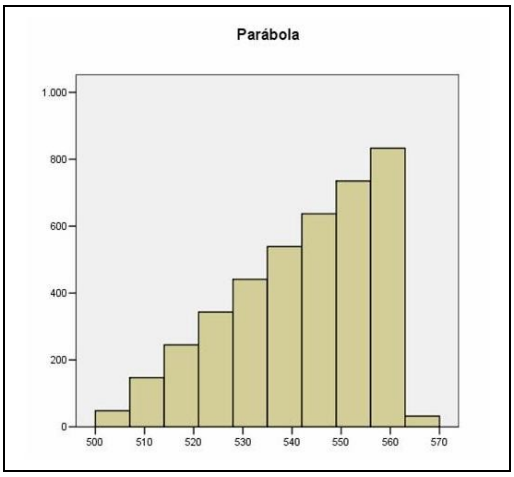


Figura 34. Histograma de una capa parabólica. Se observa crecimiento en la dirección que crece la parábola.

Tabla 16. Estadísticas de la capa parabólica fallada.

Parábola Fallada	
Media	549,675306
Mediana	559,726949
Moda	#N/A
Desviación estándar	21,53226587
Varianza de la muestra	463,6384736

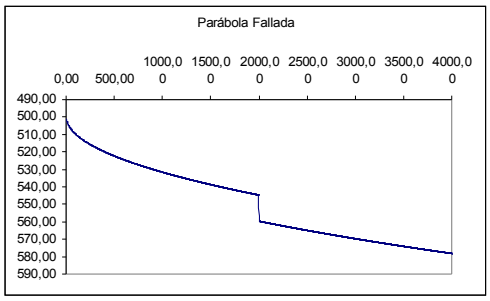


Figura 35. Configuración de una capa parabólica fallada.

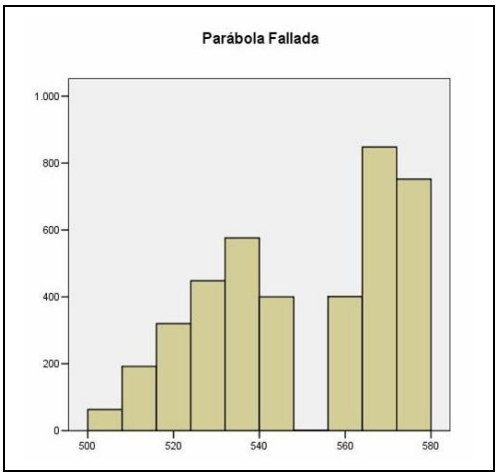


Figura 36. Histograma de una capa parabólica fallada. Se observa crecimiento en la dirección que crece la parábola, adicionalmente de observan dos modas en la distribución.

Con la elipse se puede modelar una estructura en la dirección de su orientación. Observamos que tiene un comportamiento exponencial positivo en su distribución aunque su geometría es limitada a diferencia de otras geometrías que se han visto. Las altas frecuencias se deben a las zonas bajas de la elipse, es decir, en su

zona más achatada. También muestra sensibilidad a las fallas, pero en el modelo no se generó una falla lo suficientemente grande para romper el histograma en dos poblaciones.

Tabla 17. Estadísticas de la capa elipsoidal.

Elipse	
Media	157,0789753
Mediana	173,2050808
Moda	6,323764702
Desviación estándar	44,6470476
Varianza de la muestra	1993,35886

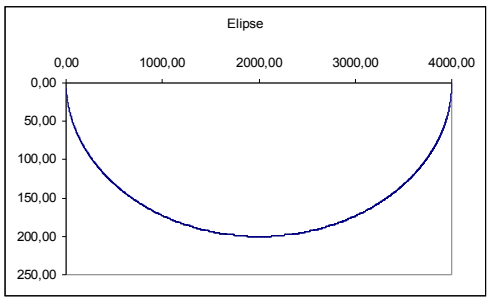


Figura 37. Configuración de una capa elipsoidal.

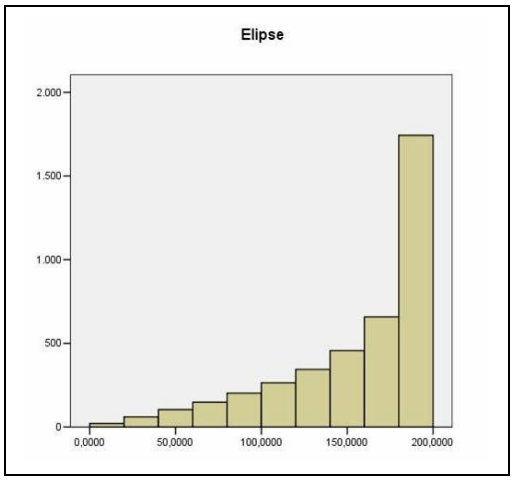


Figura 38. Histograma de una capa elipsoidal. Se observa una distribución exponencial.

Tabla 18. Estadísticas de la capa elipsoidal fallada.

Elipse fallada	
Media	164,5827253
Mediana	180,0619183
Moda	#N/A
Desviación estándar	45,2680319
Varianza de la muestra	2049,194712

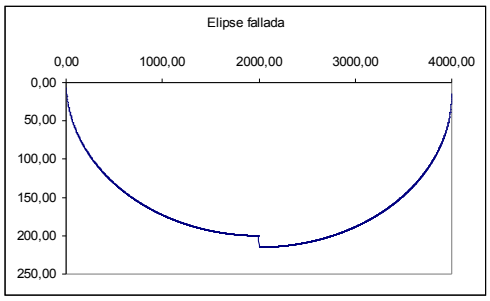


Figura 39. Configuración de una capa elipsoidal fallada.

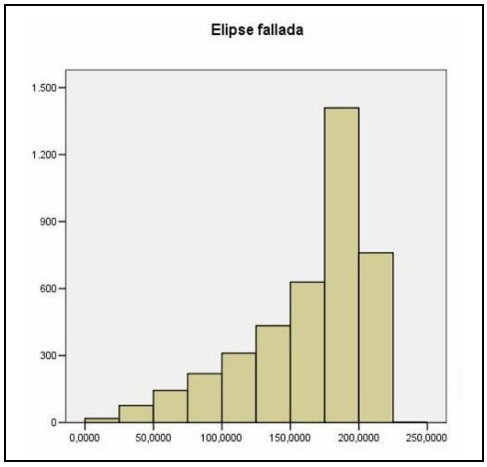


Figura 40. Histograma de una capa elipsoidal fallada. Se observa una distribución exponencial.

De las observaciones realizadas a los histogramas se puede extraer que: (1) Ninguna configuración generó distribuciones normales; (2) las capas planas horizontales generaron modas que contienen el 100 % de los datos; (3) Las capas planas inclinadas generaron distribuciones uniformes.

Pruebas en datos sintéticos

Capas planas inclinadas

Esta prueba estudia el comportamiento del programa en tres capas planas inclinadas con tres velocidades distintas y crecientes en función de la profundidad. Con esta prueba se modeló una zona de capas planas originalmente horizontales y alta rigidez flexural que han sido sometidas a eventos tectónicos sinsedimentarios. Pero también puede utilizarse para modelar otro tipo de acuñamientos presentes en la geología.

Los datos X y Y fueron generados de forma aleatoria (figura 41) y la coordenada Z se construyó usando funciones planas, tal que Z(X, Y) = A*X + C; donde A es la máxima pendiente del plano y C es una constante.

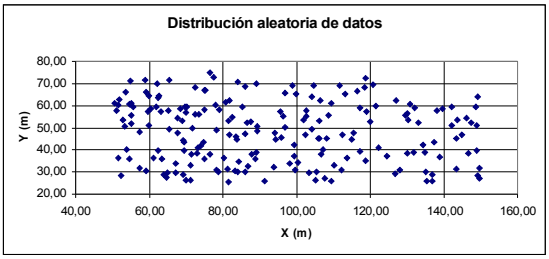


Figura 41. Distribución de los datos sintéticos en capas planas inclinadas.

Al generar una cuadricula de cada una de las superficies usando los tres métodos de ajuste de datos, se observó; que el método del inverso de la distancia disminuye la pendiente de los planos originales y su histograma de frecuencias se alejó del comportamiento ideal de una capa plana. El ajuste por vecinos mas cercanos mostró buenos resultados con la salvedad de que aumentó el peso de los datos en áreas que aleatoriamente fueron mayor muestreadas y regresó valores nulos en zonas sin datos. Finalmente se realizó el ajuste utilizando el promedio de los datos dentro de un radio alrededor de cada punto de la cuadricula. Este método presentó tres variantes, una cuando el radio de búsqueda fue muy pequeño no se tomo valores y el programa retornó ceros, en radios de búsqueda muy amplios el programa retorno superficies de pendiente mucho menor y zonas horizontales, y finalmente usando un

radio de búsqueda intermedio (luego de pruebas cíclicas) se generó un buen resultado, el cual se ajustó a los datos de entrada y simuló el comportamiento ideal de una capa plana inclinada.

El cálculo de las velocidades promedio realizado mediante las curvas TZ teóricas mostró: (1) un aumento en las zonas más someras de la primera capa con respecto a las velocidades interválicas originales, el cual se va contrayendo a medida que la capa se vuelve más profunda. Este fenómeno es provocado por que la velocidad se halló mediante el polinomio de segundo grado que mejor se ajusta a los datos y no las tablas TZ originales. En la figura 42 se puede observar la distribución de datos en cada capa y que las curvas se ajustan mejor a los datos con la profundidad. (2) En la segunda y tercera capa la velocidad es representativamente menor que sus velocidades y este fenómeno es más notable en la capa más profunda. Esto es provocado por que la velocidad promedio de una capa *N* contiene información de todas las capas más someras, y a medida que esas capas posean mayor espesor, aumentaran su peso en el cálculo de la velocidad promedio.

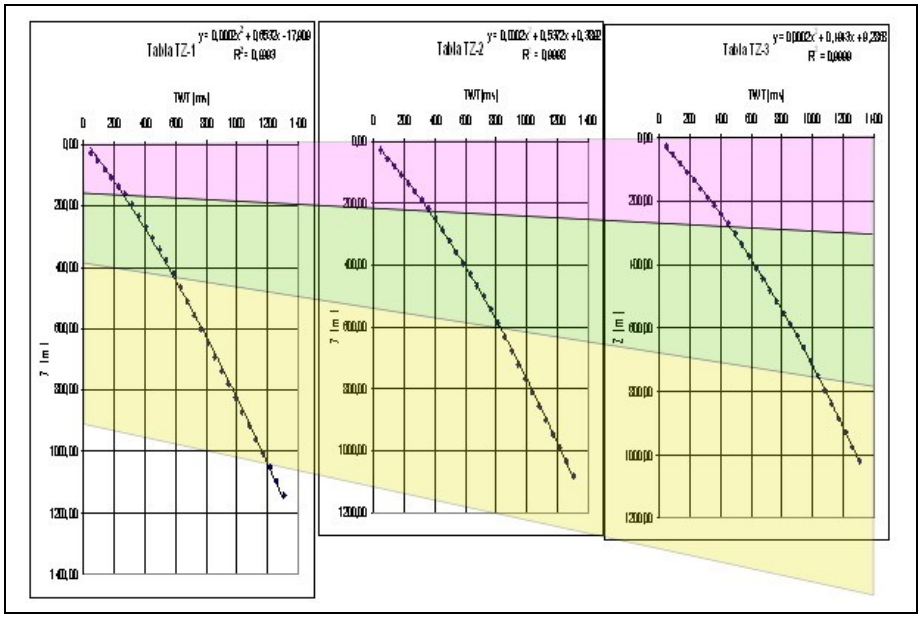


Figura 42. Esquema de los datos tiempo profundidad para cada superficie.

Con respecto a las velocidades interválicas, la primera capa tiene las velocidades promedio e interválica coinciden, dado el inicio de esta capa en la superficie. La segunda y tercera capa, poseen velocidades muy próximas a la velocidad original en algunas zonas por defecto y en otras por exceso. Esto es provocado por que la velocidad es la razón de cambio de entre la distancia y la profundidad, y para los datos originales la expresión matemática de esto fue una recta y no una parábola como la que se usó en el modelo. Sin embargo, Los resultados dieron buen ajuste y la parábola se aproxima bien a los datos en su parte más profunda, y se pudo interpretar que para modelos con más niveles, por ejemplo, una columna estratigráfica con velocidades crecientes se tendría mejor ajuste.

El método ponderado del vecino más cercano dio los mejores resultados para la conversión tiempo profundidad en los dos tipos de velocidad. Este método permitió distribuir los valores de velocidad extraído de las curvas TZ a todos los puntos de la cuadricula evitando que los mismos se corrompieran o cambiaran su distribución. El método de los promedios dentro de un radio dejó muchos puntos de la cuadricula con valor nulo o combinaba las velocidades de los tres pozos. El método del inverso de la distancia tuvo problemas de borde o problemas computacionales de punto flotante.

Finalmente se observó un buen ajuste de los resultados con respecto al modelo sintético original. Presentándose mejor ajuste el mejor ajuste en la capa más somera y aumentando el error en las capas subyacentes. El crecimiento de error con la profundidad se logró explicar considerando que los estudios de velocidad se realizan de los eventos más someros a lo más profundos, esto quiere decir que cada superficie inferior tendrá su error de cálculo más el error que arrastra la superficie anterior.

Tabla 19: Correlaciones de Pearson entre los resultados y los datos de entrada.

	twt1	twt2	twt3
z1 Promedio	0,996		
Z1 Interválica	0,996		
z2 Promedio		0,993	
Z2 Interválica		0,993	
z3 Promedio			0,991
z3 Interválica			0,992

Capas elípticas

Esta prueba anterior, este ensayo tiene el objetivo de someter al programa al estudio de tres capas, las cuales tendrán forma de elipsoide y concavidad positiva, sus velocidades son distintas y crecientes con la profundidad. Y simula cuencas sin fallamientos o estructuras con deformaciones elásticas sinclinales.

Las coordenadas X y Y fueron generadas mediante funciones aleatorias de una hoja de cálculo y para el tiempo se usó funciones $T(X,Y) = raiz(R^2-(X-100)^2-r^2*(Y-50)^2)$; donde R es el radio mayor, R/r es el radio menor y las elipses están centradas en (100, 50).

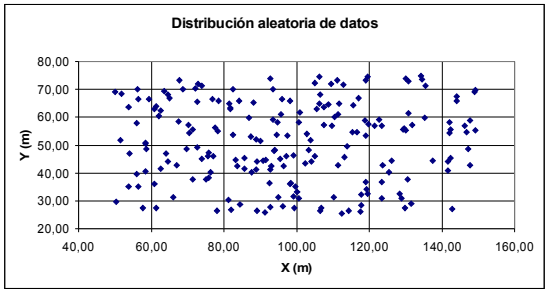


Figura 43. Distribución de datos sintéticos en capas elipsoidales.

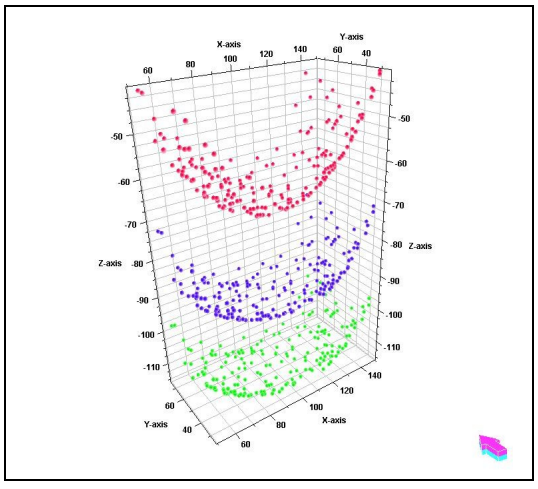


Figura 44. Modelo sintético de capas elipsoidales.

A diferencia de la prueba anterior, en esta el tiempo también es dependiente de la coordenada Y, además la geometría es simétrica con respecto al centro del

elipsoide en los dos ejes. Por ello se decidió tomar cinco curvas de tiempo profundidad (figura 45) donde; TZ-2 está en el centro, TZ-1 a la izquierda y TZ-3 a la derecha simétricas con respecto al eje Y y TZ-4 arriba y TZ-5 abajo simétricas con respecto al eje X, pero con distinto muestreo en profundidad.

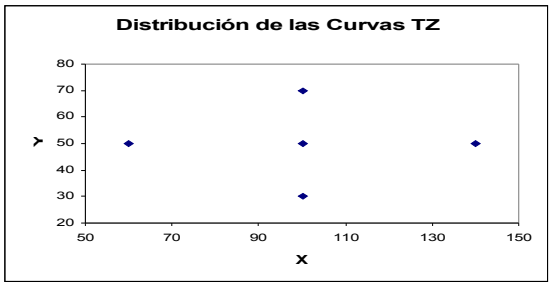


Figura 45. Distribución las tablas TZ en capas elipsoidales.

En la figura 46, el resultado más evidente que se observó es que aún habiendo un parecido en las curvas, estas son distintas. En los polinomios de segundo grado se observa un buen ajuste pero no hay similitud entre ellos. Esto nos indica que aún teniendo los mismos datos y la misma configuración la generación de curvas depende del muestreo.

Los métodos que mejor ajustaron para generar las cuadriculas fueron el método de los promedios para la superficie en tiempo y el método del vecino más cercano para el modelo de velocidades. En la tabla 17 se puede observar las estadísticas de las velocidades obtenidas y se aprecia el mismo fenómeno que ocurrió en la prueba de capas planas. Las velocidades promedio someras son mayores a la real y las más profundas son menores a las reales. La velocidad interválica ajusta muy bien en la primera capa, pero tiene mayor error en la tercera.

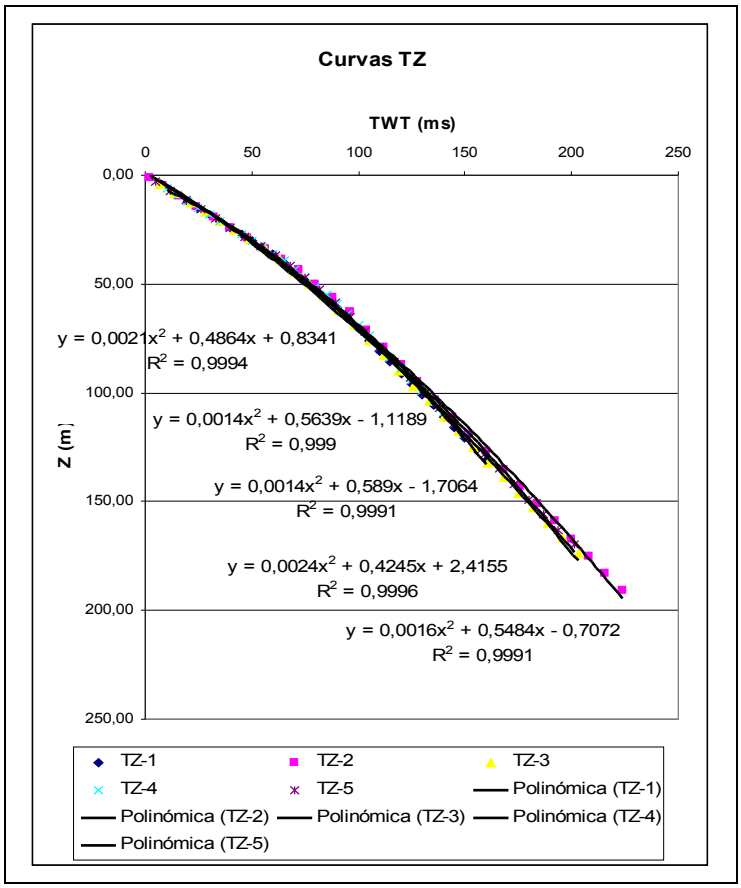


Figura 46. Distribución las tablas TZ en capas elipsoidales.

Tabla 20: Estadísticas de las velocidades halladas

	Vp1	Vp2	Vp3	Vi2	Vi3
Media	1265,46	1373,61	1458,12	1608,41	1819,86
Mediana	1243,11	1360,51	1450,09	1582,78	1803,63
Moda	1243,11	1360,51	1450,09	1582,78	1803,63
Desviación estándar	29,37	16,77	9,53	31,32	20,06
Varianza	862,50	281,30	90,87	980,95	402,37
Vp: Velocidad promedio, Vi Velocidad interválica					

En el modelo en profundidad (figura 47) se pudo observar que los mayores errores están en la zona de borde. Adicionalmente se pueden apreciar pequeñas variaciones en las zonas simétricas. Esto es debido a la generación de distintas tablas TZ, es decir el uso de un muestreo con otras características.

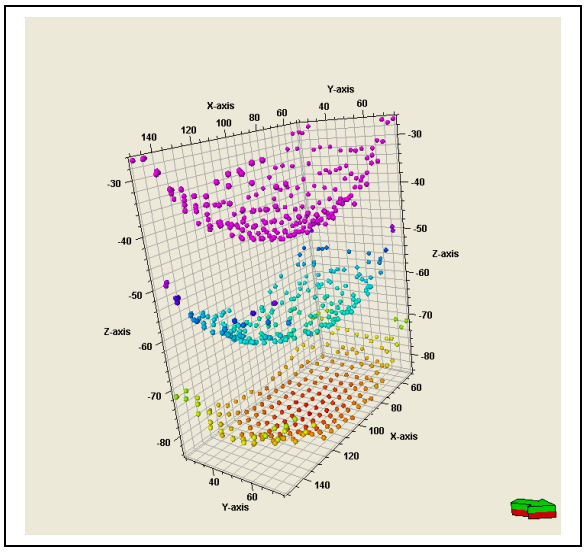


Figura 47. Modelo en profundidad de las superficies elipsoidales.

Pruebas en datos reales

Ambiente sedimentario de transición

El campo usado en esta prueba está ubicado en La Cuenca Oriental de Venezuela, Subcuenca de Guarico, situado en parte nororiental del estado Guarico. Se han perforados 32 pozos con profundidad promedio de 7700 pies, cuatro de ellos tienen estudios tiempo profundidad (tres tiros de verificación sísmica y un perfil sísmico vertical), 27 líneas sísmicas 2D regionales y una sísmica 3D. El campo contiene una importante cantidad de areniscas tipo barra, las cuales se formaron durante el Oligoceno en un ambiente costero. Se estima que estas areniscas de tipo barra tienen una orientación suroeste-noreste y que se acuñan hacia el suroeste. Esta orientación, en combinación con la estructura de la Formación – un monoclinal que buza hacia el noreste – da origen a un entrampamiento estratigráfico. Una segunda formación posee areniscas de tipo canal, en trampas estratigráficas.

El yacimiento principal se interpreta como arena de barra costera y podría depositarse bajo un ambiente de delta dominado por olas en una dirección NE-SW a la línea paleo-costera. Así se presume que la orientación de esta arena de barra está distribuida paralela al NE-SW. Por otro lado, las facies de canal también se reconocen y podría desarrollar la dirección de NW, (reporte interno TOGV, 2004).

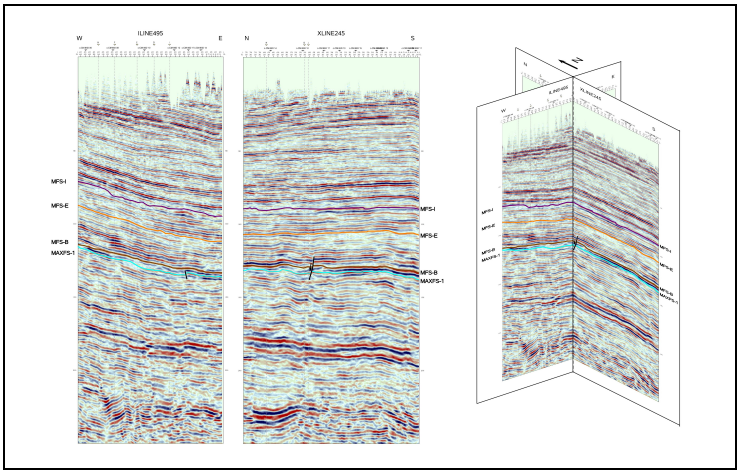


Figura 48. Imagen de líneas sísmicas utilizadas y capas usadas en esta prueba.

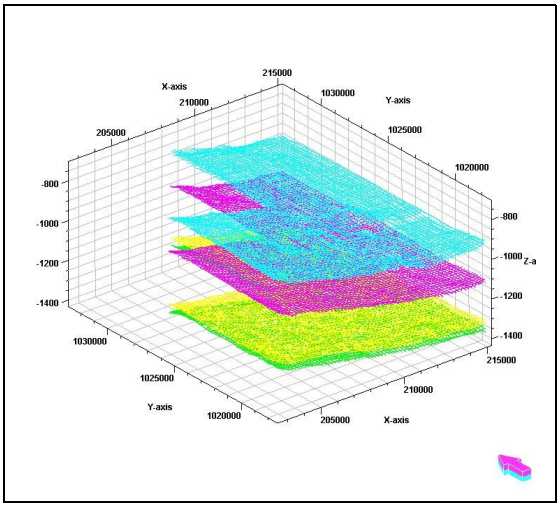


Figura 49. Configuración en tiempo doble de viaje de las capas usadas en esta prueba.

Esta prueba se realizó sobre cuatro superficies de inundación. Una de ellas, MAXFS1, es una superficie de máxima inundación regional y MFS-B, MFS-E, MFS-I son superficies de inundación locales. En la figura 49 se observa la configuración la distribución en tiempo y su buzamiento con tendencia NE.

En el histograma de frecuencia (Figura 50.a) de los datos en tiempo la capa MAXFS1 se observo una tendencia semejante a una elipse. Los otros histogramas (Figura 50.b, 50.c y 50.d) pertenecientes a la reconstrucción de los datos de

MAXFS1 mediante cuadriculas, Se pudo observar que el mejor ajuste a una superficie de máxima inundación lo generó el método del vecino más cercano, seguido del promedio dentro de un área. Pero el método del inverso de la distancia presentó un histograma semejante al de un plano inclinado. A diferencia de las pruebas sintéticas cuyos datos tenían una distribución aleatoria, el mejor ajuste es por el método del vecino más cercano, lo que hace concluir que el mejor ajuste para superficies con datos bien distribuidos nos permite usar este método y en superficies con distribuciones aleatorias es mejor usar el método de los promedios ajustando bien el radio de búsqueda.

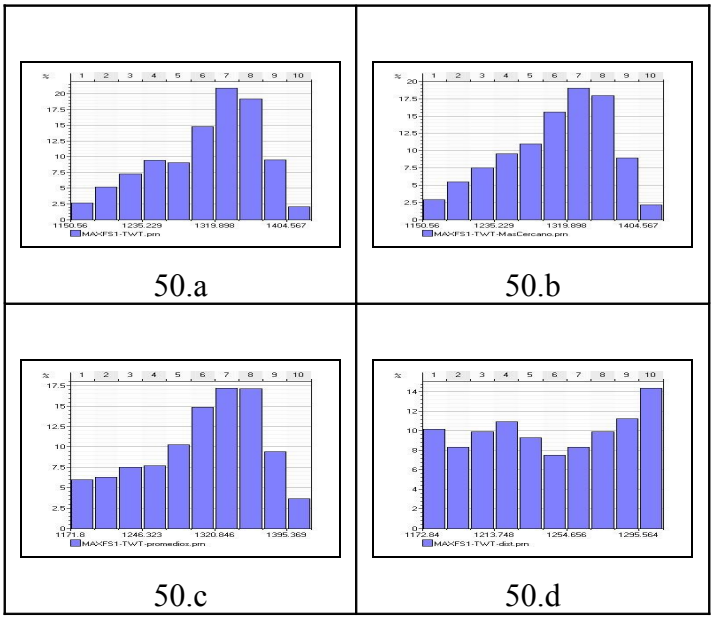


Figura 50. Histograma de frecuencia de los datos originales (a) y de las cuadriculas generadas con el programa, donde (b) es la cuadricula generada con el método del vecino más cercano, (c) el promedio dentro de un área y (d) es el estimador del inverso de la distancia.

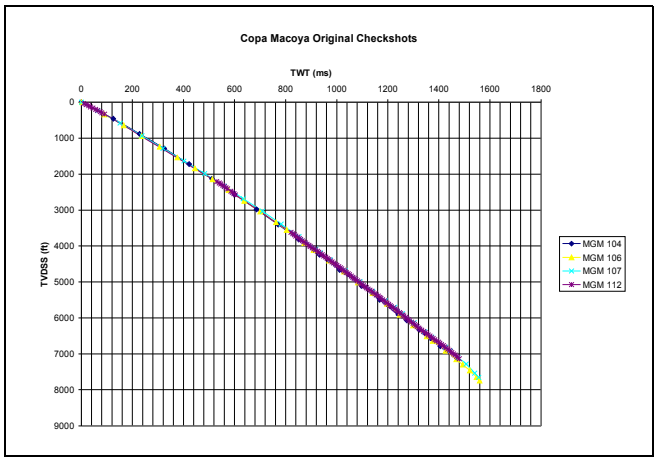


Figura 51. Curvas TZ del área de estudio.

En la figura 51 se pudo observar el buen ajuste de las curvas TZ en el área de estudio. Esto es consecuente con la geología sencilla de las formaciones y la gran continuidad lateral de la geometría. Así mismo se observo en los mapas de velocidad que respetan el comportamiento geométrico del medio, teniendo un aumento en la dirección este.

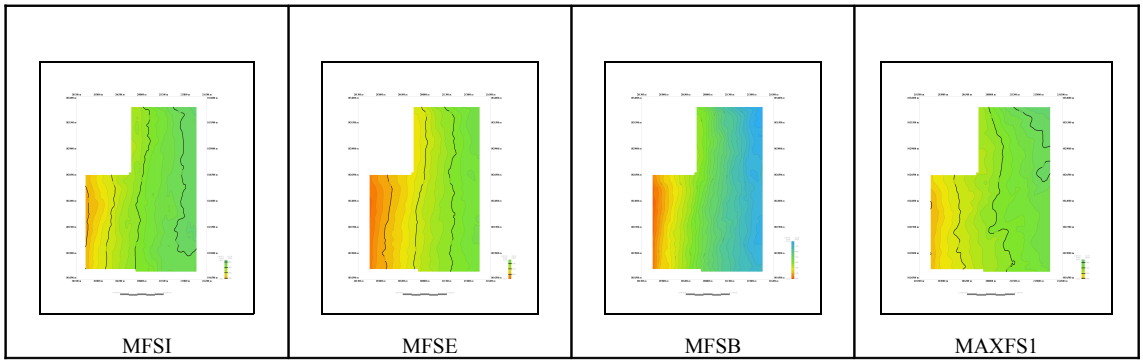


Figura 52. Modelos de velocidad para cada una de las superficies geológicas.

Según un reporte interno, la resolución sísmica calculada en el área de interés va desde 91,2 pies hasta 140,4 pies en función de la profundidad.

Tabla 21. Comparación de la profundidad obtenida con estudio de pozo y la profundidad calculada.

Pozo	Superficie	Profundidad del pozo (ft)	Profundidad Calculada (ft)	Diferencia (ft)
1	MFS-I	4232	4205	27
2	MFS-I	4084	4035	49
3	MFS-I	3898	3860	38
4	MFS-I	3439	3320	119
5	MFS-I	4040	3942	98
1	MFS-E	5117	5113	4
2	MFS-E	4994	4890	104
3	MFS-E	4769	4700	69
4	MFS-E	4312	4187	125
5	MFS-E	4860	4754	106
1	MFS-B	6263	6232	31
2	MFS-B	6291	6145	146
3	MFS-B	6055	5948	107
4	MFS-B	5640	5552	88
5	MFS-B	6087	5975	112
1	MAXFS-1	6438	6425	13
2	MAXFS-1	6467	6345	122
3	MAXFS-1	6230	6205	25
4	MAXFS-1	5839	5713	126
5	MAXFS-1	6276	6150	126

Se apreció en la tabla 18 la diferencia entre la profundidad calculada con el programa y la profundidad obtenida de la interpretación de pozo. Los pozos 1, 2, 3 y 4 poseen estudios de velocidad mediante tiro de verificación sísmica y perfil sísmico vertical. El pozo 5 es el pozo de control. Se observó que el pozo de control no da buen resultado aunque se mantiene dentro de la resolución sísmica.

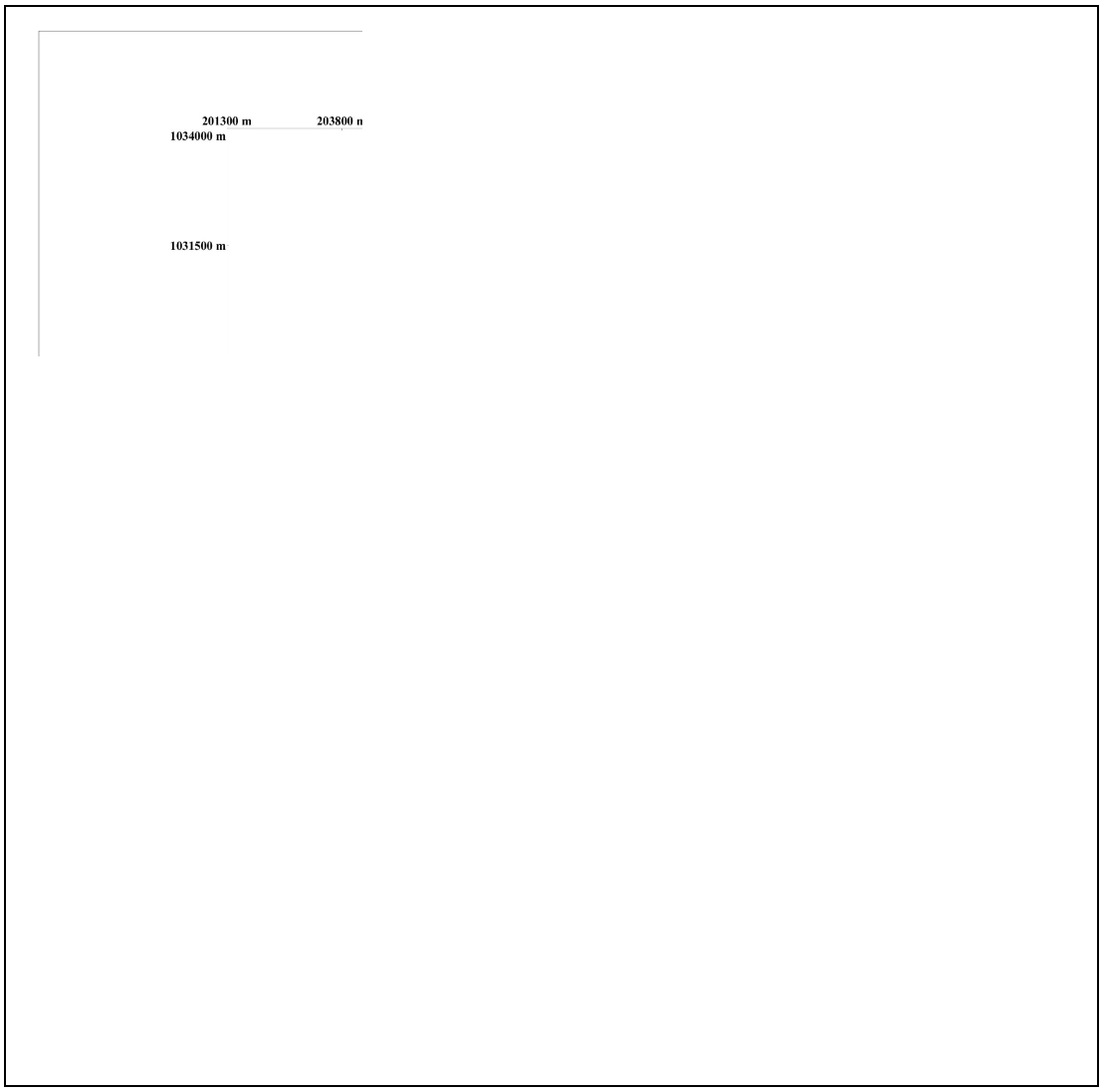


Figura 53. Mapa en profundidad de la máxima superficie de inundación (MAXFS-1). En la figura se observa la ubicación de los pozos con estudios de velocidad y el pozo de control.

Geología estructuralmente compleja

Esta prueba tuvo como objetivo evaluar el programa en zonas con cierta complejidad geológica. Además de que fue realizada con datos de sísmica 2D, es decir no poseyó una interpretación regular en un mallado, lo que se traduce en áreas no muestreadas o vacío de datos.

El área de estudio fue una zona de relleno, con una falla normal que atraviesa a todas las capas y otra falla normal que corta las dos superficies inferiores.

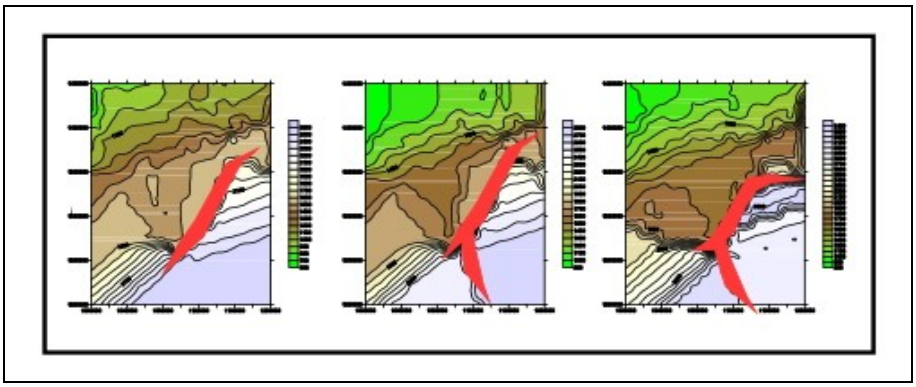


Figura 54. Mapas en tiempo de la zona de estudio. A la izquierda la más somera y a la derecha la más profunda.

Cada una de las superficies geológicas contó con interpretaciones que ocupaban aproximadamente 250000 datos cada una. Esto provocó deficiencias en el programa para generar modelos de velocidad. Así que se generó nuevas cuadriculas de 200 x 250 mediante el método del vecino más cercano. La figura 54 muestra el resultado de este ajuste, el cual, acopló bien con los datos originales respetando la estructura en su totalidad.

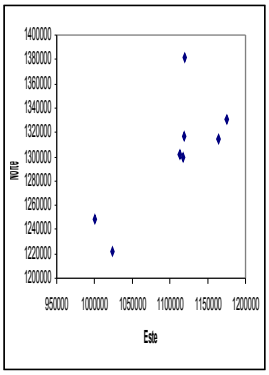


Figura 55. Distribución de pozos con datos tiempo profundidad.

En la figura 55 se muestra la ubicación de los pozos con curvas de tiempo velocidad. De los pozos ubicados en la zona SW se obtuvo la información de velocidad del bloque deprimido y los pozos en la zona NE se extrajo la velocidad del otro bloque. En la figura 55 muestra las curvas de velocidad de todos los pozos, de estas se pudo observar las pequeñas variaciones de velocidad del área. Esto se puede

traducir en que a pesar del dominio estructural, la litología posee comportamientos similares lateralmente.

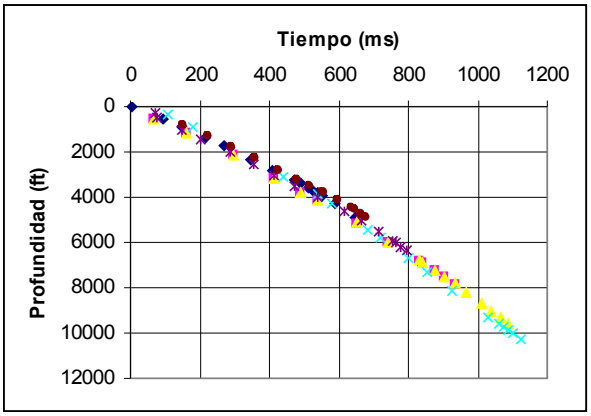


Figura 55. Curvas TZ de los pozos.

Tabla 22. Promedio de velocidades resultante para cada capa

Capa	Velocidad promedio (ft/s)	Velocidad Interválica (ft/s)
1	7996,67663	--
2	8211,22296	10660,5738
3	8526,26814	11202,1312

La figura 56 muestra la distribución de velocidad obtenida usando el método del promedio en un radio de búsqueda. Se logró observar la intensa variación en la zona del bloque deprimido aunque se mantiene continuidad lateral, esto fue debido a que la velocidad se trabajó como una función continua.

Finalmente en la figura 57 se observó que el dominio estructural en tiempo se mantiene. Este hecho se debió a la continuidad lateral de la velocidad como operador lineal de la conversión.

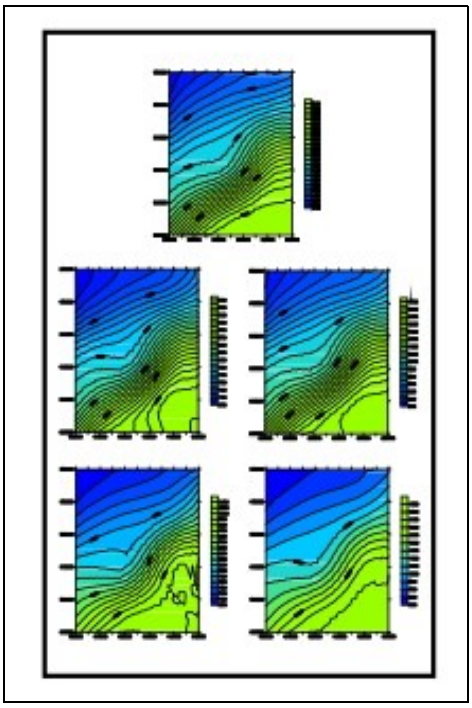


Figura 56. Mapas de velocidades para la geología compleja. Arriba la capa más somera y abajo la más profunda.

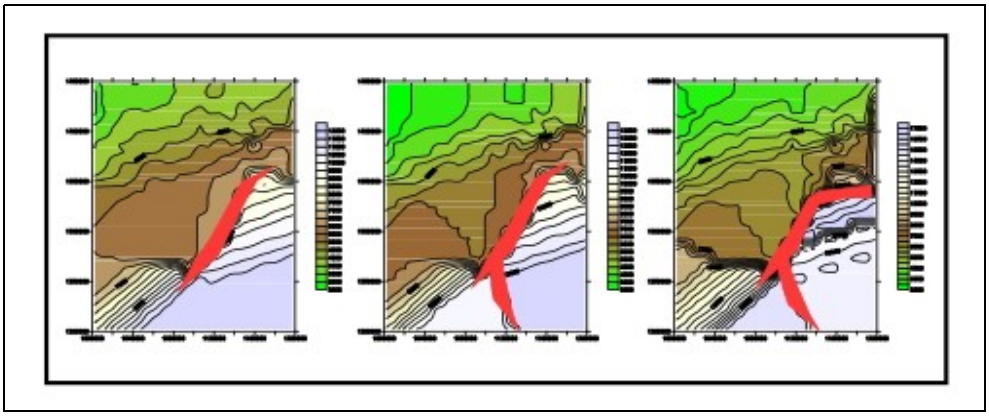


Figura 57. Mapas en tiempo de la zona de estudio. A la izquierda la más somera y a la derecha la más profunda.

CAPÍTULO V

CONCLUSIONES Y RECOMENDACIONES

- Los histogramas de frecuencia generados con datos de geometrías ideales, mostraron comportamientos que combinados pueden modelar a datos reales mediante combinaciones lineales. Adicionalmente se observó que los histogramas son sensibles a la distribución de los datos, y un remuestreo modifica su expresión. De la misma manera se notó la multiplicidad de fallas expresa multiplicidad de modas. Finalmente, existe relación lineal entre los histogramas de tiempo y los histogramas de profundidad.
- El uso de estimadores lineales ponderados permite la generación de cuadriculas de datos de superficies. En superficies geológicas, el método del vecino más cercano representa mejor ajuste a distribuciones de datos regulares, geologías sencillas y estructuras más complejas. El método del promedio de los datos dentro de un radio da buenos resultados en datos aleatoriamente distribuidos siempre que se defina un radio adecuado. El método de inverso de la distancia permite suavizar las superficies y puede ser usado en áreas con poca variación lateral de la velocidad.
- Las variaciones laterales de velocidad en zonas estructuralmente complejas son altas. En contraparte, en zonas sencillas hay baja variación o incluso nula.
- Los modelos de velocidad son funciones continuas en el espacio, de forma que el control estructural lo tiene la interpretación sísmica en tiempo.
- El uso de velocidades promedio es recomendable en áreas con alta continuidad lateral. La velocidad interválica tiene mejor respuesta en zonas estructuralmente complejas o alejadas de información de pozo.
- Finalmente los parámetros que controlan una exitosa conversión de unidades de tiempo a unidades de profundidad dependen de la velocidad del medio y su variación en el espacio, de la geometría del área de estudio, del muestreo de

los datos y posterior remuestreo en situaciones computacionales que se requiera. Todo esto se traduce en que la conversión tiempo profundidad es un proceso cíclico donde se debe tomar en cuenta la rigidez del dato versus la geología de la zona de estudio.

- El manejo de matrices de superficies de muchos datos es dependiente del sistema operativo y la cantidad de memoria de acceso aleatorio que contiene el computador. De todas formas, este problema puede ser parcialmente resuelto usando tipos de datos estructurados definidos por el programador.
- A pesar de que el programa esta escrito en lenguaje C/C++, la interfaz grafica fue diseñada para Windows, se recomienda tomar la parte funcional del programa y elaborar las interfaces para otros sistemas operativos.

BIBLIOGRAFÍA

BARTEL, D. C., BUSBY, M., NEALON, J. Y ZASKE, J. (2006) ***Time to Depth Conversion and Uncertainty Assessment using Average Velocity Modelling***. Chevron Energy Technology CO. SEG/New Orleans 2006 Annual Meeting.

BROWN, ALISTAIR R. (2003) *Interpretation of 3-seismic Data.* Tercera edición, AAPG Memoir 42

DEITEL, H. M. Y DEITEL P. J. (2004). ***Cómo Programar en C/C++ y Java***. Cuarta edición, PEARSON Educación, México.

DOBRIN, M. (1961). ***Introducción a la Prospección Geofísica.*** Ediciones Omega, Barcelona. 438 p. España.

DEPARTAMENTO DE INGENIERÍA DE MINAS DE LA UNIVERSIDAD DE ATACAMA (DIMUDA, 2006) html//www.plata.uda.cl.

GERALD, C. F. Y WHEATLEY, P. O. (2000). ***Análisis Numérico con Aplicaciones***. Sexta edición. Prentice Hall.

GROSSMAN, S. I. (1999). ***Álgebra Lineal***. Quinta edición. McGraw-Hill.

NAKAMURA, S. (1997). ***Análisis Numérico y Visualización Gráfica con Matlab***. Pearson Educación.

NAVIDI, W. (2006). ***Estadística para ingenieros y Científicos***. McGraw-Hill.

REGUEIRO, P. ***Métodos Sísmicos***. www.gc.usb.ve/Pepe_Regueiro/Metodos1.html.

SERWAY, R. (1999) *Física. Tomos I y II.* McGraw-Hill.

SCHMIDT-THOMÉ, P. (1972). ***Tektonik***. - Vol. II de BRINKMANN, R. (Ed): Lehrbuch der Allgemeinen Geologie, Enke (Stuttgart).

SHARMAN, P.V. (1986). ***Geophysical Methods in Geology***. Second Edition. Elseiver Science Publishing Co., Inc. New York, U.

SHERIFF, R. E. (1991). ***Encyclopedic Dictionary of Exploration Geophysics***. Third Edition. Geophysical Reference Series. Society of Exploration Geophysicists. U.S.A.

SÖLLNER, W., ANDERSEN, E. Y LIMA, J. (2004). ***Fast time-to-depth mapping by ray transformation in a 3-D visualization system***. SEG Int'l Exposition and 74th Annual Meeting, Denver, Colorado.

SPIEGEL, M. R. (1991). ***Estadística***. Segunda edición. McGraw-Hill.

TEIKOKU OIL & GAS VENEZUELA (TOGV). (2004). *Reporte interno.*

TROLLTECH (2006) ***Qt 4.2 Whitepaper***. Noruega.

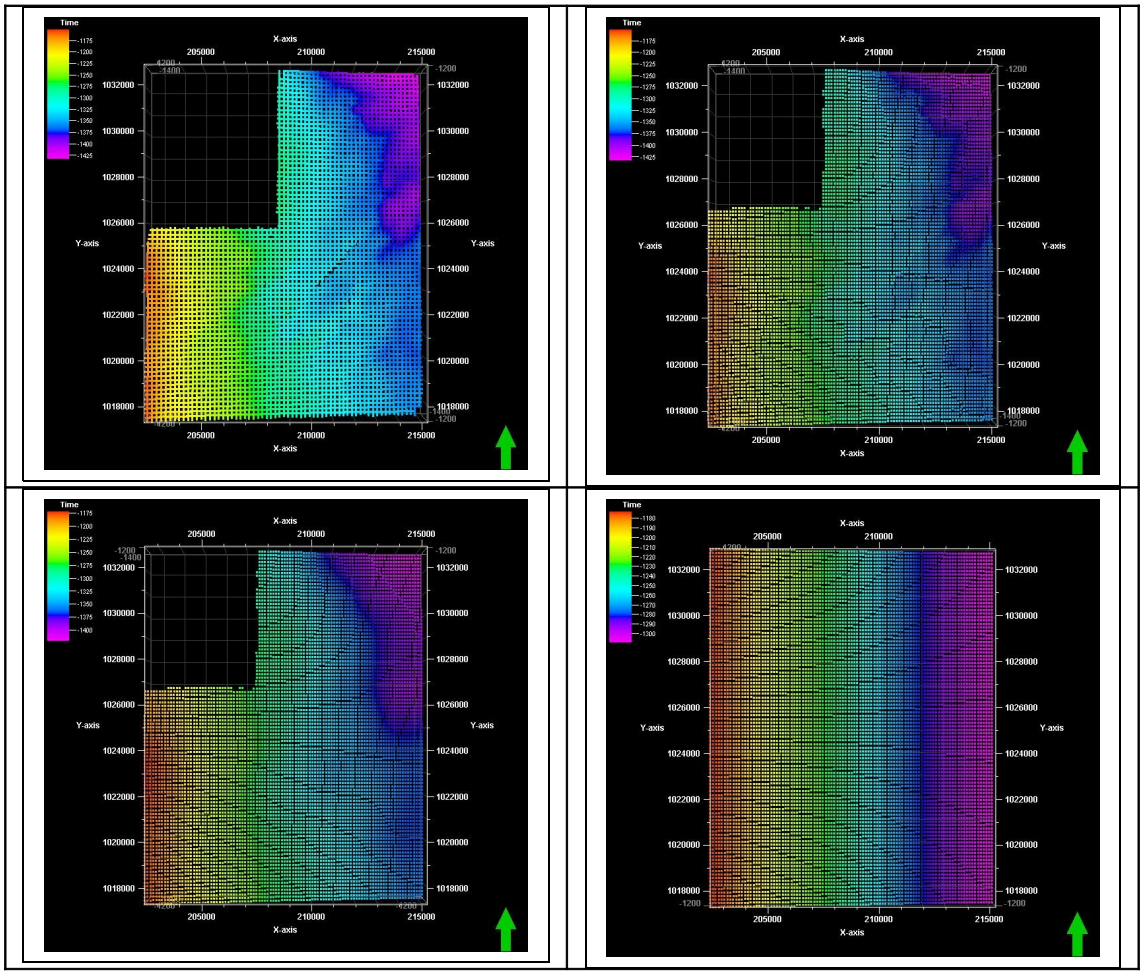
YILMAZ, Ö. (1987). ***Seismic data processing***. Society of Exploration Geophysicists, Tulsca, EEUU.

WIKIPEDIA, la enciclopedia libre. (2007). http://es.wikipedia.org

ZIJLSTRA, D. M., VAN DER MADE, P. M. BUSSEMAKER, F. Y VAN RIEL, P. (1991). ***Effective Depth Conversion: A North Sea Case Study***. Applications to Exploration Problems, Jason Geosystems.

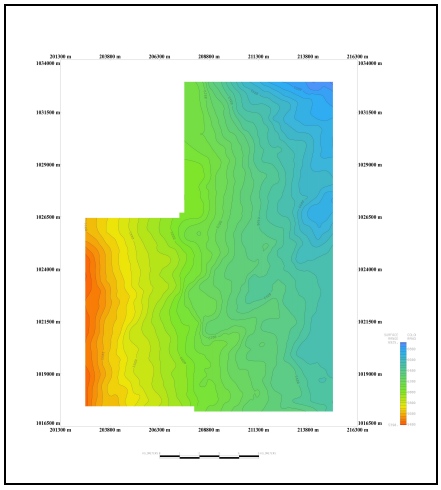
ANEXOS

Anexo 1: Cuadrículas generadas con los distintos métodos de ajuste por ponderación

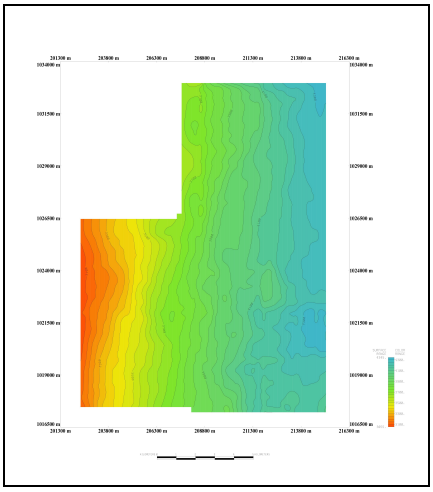


Reconstrucción de los datos en tiempo para la conversión tiempo profundidad. Arriba a la izquierda en el modelo original. A la derecha se usó el método del vecino más cercano. Abajo a la izquierda se usó el método del promedio de los datos en un radio de búsqueda. Y abajo a la derecha el método del estimador del inverso de la distancia.

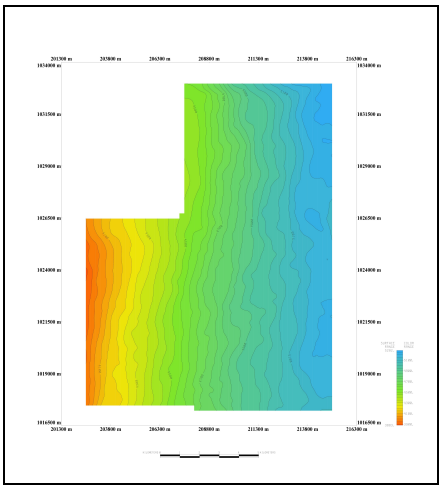
Anexo 2: Mapas en profundidad de la prueba en datos reales



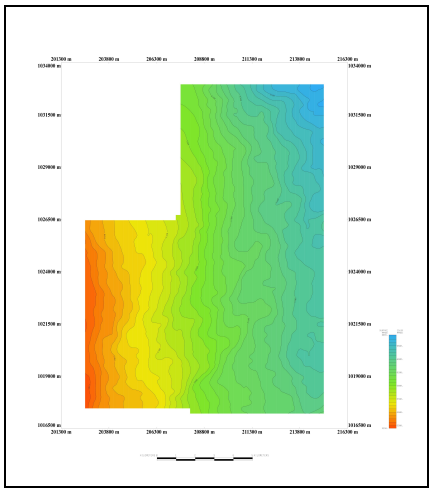
MAXFS-1



MFS-I



MFS-E



MFS-B

Mapas en profundidad de la prueba con datos reales. Arriba a la izquierda una superficie de máxima inundación, el resto son superficies de inundación de menor nivel.

Anexo 3: Estructura para manipular velocidades

```
//---------------------------------------------------------------------------
//    Librería para calcular las velocidades promedio o interválicas en un
//    pozo o ubicación con tablas tiempo profundidad.
//---------------------------------------------------------------------------
#include <math.h>                //Inclusión de la librería matemática.
//---------------------------------------------------------------------------
double distancia (double velocidad, double tiempo); //Relación M.R.U. para
distancia.
double velocidad (double tiempo, double distancia); //Relación M.R.U. para
velocidad.
double tiempo (double velocidad, double distancia); //Relación M.R.U. para tiempo.
//---------------------------------------------------------------------------
double Polinomio2grado(double A, double B, double C, double X); //Calcula el valor
de un polinomio de segundo grado con
                                        //Coeficientes A, B, Y en un X dado.
double TDobleATSimple(double T);  //Pasa tiempo doble a tiempo simple.
double TSimpleATDoble(double T);  //Pasa tiempo simple a tiempo doble.
double SegAMseg(double T);        //Pasa segundos a milisegundos.
double DistanciaEtrePuntos(double X1,double X2,double Y1,double Y2); // Calcula
la distancia entre puntos.
//---------------------------------------------------------------------------
double Vintervalica(double Z1,double Z2,double T1,double T2);
     //Velocidad por intervalos para una coordenada dada. Z son las
     //profundidades del intervalo, y T tiempos del intervalo.
//---------------------------------------------------------------------------
double distancia(double velocidad, double tiempo)
{
```

```
    return velocidad*tiempo;
}
//---------------------------------------------------------------------------
double velocidad(double tiempo, double distancia)
{
    return distancia/tiempo;
}
//---------------------------------------------------------------------------
double tiempo(double velocidad, double distancia)
{
    return distancia/velocidad;
}
//---------------------------------------------------------------------------
double polinomio2grado(double A,double B,double C,double X)
{
    return (A*X*X + B*X + C);
}
//---------------------------------------------------------------------------
double TDobleATSimple(double T)
{
    return T/2;
}
//---------------------------------------------------------------------------
double TSimpleATDoble(double T)
{
    return 2*T;
}
//---------------------------------------------------------------------------
double SegAMseg(double T)
{
```

```
    return T*1000;
}
//---------------------------------------------------------------------------
double DistanciaEtrePuntos(double X1,double X2,double Y1,double Y2)
{
    double XX, YY;
    XX = X2-X1;
    YY = Y2-Y1;
    return sqrt(XX*XX + YY*YY);
}
//---------------------------------------------------------------------------
double Vintervalica(double Z1,double Z2,double T1,double T2)
{
    double Zt, Tt;      //Zt: espesor. Tt: diferencia de tiempo.
    Zt = labs(Z2-Z1);
    Tt = labs(T2-T1);
    return velocidad(Tt, Zt);
}
//---------------------------------------------------------------------------
```

Anexo 4: Estructura para manipular y crear cuadriculas

```
//---------------------------------------------------------------------------
//     Estructura con las variables y funciones para la cuadrícula
//---------------------------------------------------------------------------
#include <cstdlib>
#include <iostream>
#include <vector>
#include <cmath>
//---------------------------------------------------------------------------
#define max_filas 250000
//---------------------------------------------------------------------------
using namespace std;
//---------------------------------------------------------------------------
typedef struct superficie{

      double Datos[max_filas][3];             //Matriz de datos.
      int NDatos;                      //Número de datos.
      double Grid[max_filas][3];              //Matriz de la cuadricula.
      int NGrid;                       //Número de resultados de la cuadrícula.
      double X[max_filas], Y[max_filas];           //Vectores originales de las
posiciones.
      double minX, max_filasX, minY, max_filasY;     //máximos y mínimos que
serán usados en el grid.
      double deltaX, deltaY;           //Remuestreo para la cuadricula.
      int Nf, Nc;                      //Número de filas y columnas de la cuadrícula
      double DeltaX, DeltaY;           //Muestreo de la cuadricula.
      double radio;                    //Radio de búsqueda.
      //--------------------------
```

```
        void copiaX(double Matriz[max_filas][3], double X[max_filas], int N);
        void copiaY(double Matriz[max_filas][3], double Y[max_filas], int N);
        void imprimeVector(double V[max_filas], int N);
        double max_filasimo(double V[max_filas], int N);
        double Minimo(double V[max_filas], int N);
        void Genera_coordenadas(double X[], int Nx, double Y[], int Ny, double grid[]
[3], int NGrid);
        void Calcula_nuevo_Z(double A[][3], double grid[][3], int N, int Ngrid, double
Radio);
        double DistEntrePuntos(double a[], double b[]);
        double promedio(double matriz[max_filas], int N);
        void Zpromedios(double A[][3], double grid[][3], int N, int Ngrid, double
Radio);
        void ZDistaLinealInversa(double A[][3], double grid[][3], int N, int Ngrid,
double Radio);
        void ZmasCercano(double A[][3], double grid[][3], int N, int Ngrid, double
Radio);
        //---------------------------
        }Tipo_surfer;
//------------------------------------------------------------------------------
void copiaX(double Matriz[max_filas][3], double X[max_filas], int N)
{
      for(int i=0; i < N; i++)
      {
         X[i] = Matriz[i][0];
      }
}
//------------------------------------------------------------------------------
void copiaY(double Matriz[max_filas][3], double Y[max_filas], int N)
{
```

```
    for(int i=0; i < N; i++)
    {
       Y[i] = Matriz[i][1];
    }
}
//---------------------------------------------------------------------------
void imprimeVector(double V[max_filas], int N)
{
    for(int i=0; i < N; i++)
       std::cout << "\t" << V[i] << std::endl;
    std::cout << std::endl;
}
//---------------------------------------------------------------------------
double max_filasimo(double V[max_filas], int N)
{
    double temp = V[0];
    for(int i=1; i < N; i++)
       if(temp < V[i])
            temp = V[i];
    return temp;
}
//---------------------------------------------------------------------------
double Minimo(double V[max_filas], int N)
{
    double temp = V[0];
    for(int i=1; i < N; i++)
       if(temp > V[i])
            temp = V[i];
    return temp;
}
```

```
//---------------------------------------------------------------------------
void Genera_coordenadas(double X[], int Nx, double Y[], int Ny, double grid[][3],
int NGrid)
{
    int z = 0;
    for(int i=0; i<Nx; i++)
        for(int j=0; j<Ny; j++)
        {
            grid[z][0] = X[i];
            grid[z][1] = Y[j];
            z++;
        }
}
//---------------------------------------------------------------------------
double DistEntrePuntos(double a[], double b[])
{
    double g = sqrt((a[0]-b[0])*(a[0]-b[0])+(a[1]-b[1])*(a[1]-b[1]));
    return g;                    //Calcula la distancia entre los puntos
}
//---------------------------------------------------------------------------
double promedio(double matriz[max_filas], int N)
{
     double suma = 0;
     for(int i=0; i<N; ++i)
     {
          suma += matriz[i];      //Suma de valores de la matriz
     }
     return suma/N;               //Retorna la media aritmetica
}
//---------------------------------------------------------------------------
```

```
void Zpromedios(double A[][3], double grid[][3], int N, int Ngrid, double Radio)
{
    double Z_en_Area[max_filas];     //Vector de los Z contenidos dentro del radio.
    double a[2], b[2];         //Matrices temporales para medir las distancias.
    int contadorZ = 0;         //Cuenta los elementos en la matriz de z en área.
    int tipo;                  //tipo indica el tipo de ajuste que se desea.

    for(int i=0; i<Ngrid; ++i)
    {
        a[0] = grid[i][0];                  //X del grid
        a[1] = grid[i][1];                  //Y del grid
        for(int j=0; j<N; ++j)
        {
          b[0] = A[j][0];                   //X de los datos
          b[1] = A[j][1];                   //Y de los datos
          if(DistEntrePuntos(a, b)<=Radio)
          {
             Z_en_Area[contadorZ] = A[j][2];
             contadorZ ++;
          }
        }
        grid[i][2] = promedio(Z_en_Area, contadorZ);
        contadorZ = 0;
    }
}
//---------------------------------------------------------------------------
void ZDistaLinealInversa(double A[][3], double grid[][3], int N, int Ngrid, double
Radio)
{
    double a[2], b[2];         //Matrices temporales para medir las distancias.
```

```
    int tipo;               //tipo indica el tipo de ajuste que se desea.

    double sumaNumerador = 0;
    double sumaDenominador = 0;

    for(int i=0; i<Ngrid; ++i)
    {
        a[0] = grid[i][0];              //X del grid
        a[1] = grid[i][1];              //Y del grid
        for(int j=0; j<N; ++j)
        {
          b[0] = A[j][0];               //X de los datos
          b[1] = A[j][1];               //Y de los datos

          double Distancia = DistEntrePuntos(a, b);
          if(Distancia<=Radio)
          {
             sumaNumerador += A[j][2]/Distancia;
             sumaDenominador += 1/Distancia;
          }
        }
        grid[i][2] = sumaNumerador / sumaDenominador;
    }
}
//---------------------------------------------------------------------------
void ZmasCercano(double A[][3], double grid[][3], int N, int Ngrid, double Radio)
{
    double a[2], b[2];        //Matrices temporales para medir las distancias.
```

```
    double temp_Z;          //Valor temporal de Z hasta encontrar al más
                            //cercano.
    double temp_Dist;       //Temporal de la distancia, actualiza a la
                            //más cercana.


    for(int i=0; i<Ngrid; ++i)
    {
        temp_Dist = Radio;                  //Inicialización

        a[0] = grid[i][0];                  //X del grid
        a[1] = grid[i][1];                  //Y del grid
        for(int j=0; j<N; ++j)
        {
          b[0] = A[j][0];                   //X de los datos
          b[1] = A[j][1];                   //Y de los datos
          double Distancia = DistEntrePuntos(a, b);
          if(Distancia <= temp_Dist)
          {
             grid[i][2] = A[j][2];
             temp_Dist = Distancia;
          }
        }
    }
}
//---------------------------------------------------------------------------
void Calcula_nuevo_Z(double A[][3], double grid[][3], int N, int Ngrid, double
Radio)
{
    int tipo;
```

```
    cout << "\nSeleccione el tipo de regreción: "<<endl;

    do
    {
       cout << "\t1.- Promedio dentro del area"<<endl;
       cout << "\t2.- Estimador lineal distancia inversa"<<endl;
       cout << "\t3.- Vecino cercano"<<endl;
       cin >> tipo;
       if(tipo<1 || tipo>3)
          cout << "Error 7569: Seleccione una opcion Valida!!!";
    }
    while(tipo<1 || tipo>3);

    switch(tipo)
    {
       case 1:
           Zpromedios(A, grid, N, Ngrid, Radio);
           break;
       case 2:
           ZDistaLinealInversa(A, grid, N, Ngrid, Radio);
           break;
       case 3:
           ZmasCercano(A, grid, N, Ngrid, Radio);
           break;
       default:
           ZmasCercano(A, grid, N, Ngrid, Radio);
    }
}
//---------------------------------------------------------------------------
```

Anexo 5: Estructura para calcular coeficientes de segundo grado por mínimos cuadrados

```
//---------------------------------------------------------------------------
//        Programa para ajustar por mínimos cuadrados a partir de una matriz
//        de datos. Ver. 1
//---------------------------------------------------------------------------
#include <cstdlib>
#include <iostream>
#include <math.h>
//#include <vector>
//---------------------------------------------------------------------------
#define Maxd 250
//---------------------------------------------------------------------------
//      Función principal, CoeficienteAjuste[3] es el vector donde se
//      devuelve los coeficientes del polinomio.
//---------------------------------------------------------------------------
void coeficientes(double Datos[][2], int N, double CoeficienteAjuste[3]);
//---------------------------------------------------------------------------
void CreaA(double Completa[][2], double A[][3], int tipo, int d, int N);
void CreaB(double Completa[][2], double B[], int tipo, int d, int N);
//---------------------------------------------------------------------------
//      Encabezado para las funciones necesarias para la transposición
//---------------------------------------------------------------------------
void transpuesta(double M[][3], double Mt[][Maxd], int filas, int columnas);
//---------------------------------------------------------------------------
//      Encabezado para las funciones necesarias para la multiplicación
//---------------------------------------------------------------------------
```

```
void Mult_Matrices(double M1[][Maxd], double M2[][3], double M3[][3], int NFA,
int NCA, int NCB);
void Mult_Matrices2(double M1[][3], double M2[][Maxd], double M3[][Maxd], int
NFA, int NCA, int NCB);
void Mult_Matrice_vector(double M[][Maxd], double V[], double Va[], int ajuste, int
N);
//---------------------------------------------------------------------------
//       Encabezado para las funciones necesarias para la inversa
//---------------------------------------------------------------------------
void Inversa(int orden, double A[][3], double IA[][3]);  //Función principal para
hallar la inversa
//---------------------------------------------------------------------------
void copiaMatriz(double M[][3], double cM[][3], int Orden);         //Genera una copia
de la matriz que se desea invertir.
void identidad(double iden[][3], int Orden);                        //Genera una matriz
identidad.
void muestra(double M[][3], double cM[][3], double iden[][3], int Orden);     //
Presenta las matrices creadas
//-------------------------------------------------------------
void calcula_inversa(double IM[][3], double cM[][3], double iden[][3], int Orden); //
Función para calcular inversa
void Remiendalo(int k, int N, bool *Falla, double A[][3], double B[][3]);
void CambiaFilas(int k, int p, int N, double A[][3], double B[][3]);
void Actualiza_las_ecuaciones(int k, int N, double A[][3], double B[][3]);
void DivideFila(int k, int N, double A[][3], double B[][3]);
void restaFilasPorEscalar(double M[][3], int Orden, double escalar, int i, int k);
//---------------------------------------------------------------------------
void coeficientes(double Datos[][2], int N, double CoeficienteAjuste[3])
{
   int ajuste = 2;            //Identifica el tipo de ajuste.
```

```
   int D = 2;                //Siempre es plano.
   //int N = 0;                //Numero de datos disponibles.
   //double Datos[Maxd][2];      //Matriz de datos. Esta puede ser datos de CheckShot
(t,z)
                        //o datos de velocidad (x,y,v).

   double A[Maxd][3];              //Matriz que contiene las x o (x,y) y su arreglo para
el ajuste.
   double TranspuestaA[3][Maxd];     //Transpuesta de A.
   double Producto1[3][3];         //Producto de TranspuestaA * A.
   double InversaProducto[3][3];   //Inversa del producto.
   double PseudoinversaA[3][Maxd];   //Producto de Inversa * TranspuestaA.
   double B[Maxd];                 //Vector que contiene las F(x) de los datos.

   // 1.- construir A =>
   CreaA(Datos, A, ajuste, D, N);
   CreaB(Datos, B, ajuste, D, N);
   // 2.- construir Atranspuesta (Transponer A) =>
   ajuste += 1;
   transpuesta(A, TranspuestaA, N, ajuste);
   // 3.- Multiplicar Atranspuesta X A =>
   Mult_Matrices(TranspuestaA, A, Producto1, ajuste, N, ajuste);
   // 4.- Invertir esa matriz con la función Inversa =>
   Inversa(ajuste, Producto1, InversaProducto);
   // 5.- Multiplicar ese monstruo por la Transpuesta de A =>
   Mult_Matrices2(InversaProducto, TranspuestaA, PseudoinversaA, ajuste, ajuste,
N);
   // 6.- Finalmente multiplicar esa "cosa" que quedo por el vector B (en donde están
las Y o Z) =>
   Mult_Matrice_vector(PseudoinversaA, B, CoeficienteAjuste, ajuste, N);
```

```
}
/*****************************************************************/
void CreaA(double Completa[][2], double A[][3], int tipo, int d, int N)
{
   int i, j;
   double copiaX[Maxd];
   double copiaY[Maxd];

   for(i=0; i<N; ++i)                //Copia el vector de X.
      copiaX[i] = Completa[i][0];

   if(d==2)                          //Si el ajuste es 2d => y = f(x)
      for(j=0;j<=2;++j)
         for(i=0; i<N; ++i)
         {
               if(j==0)
                  A[i][j] = 1;
               else
                if(j==1)
                  A[i][j] = copiaX[i];
                else
                  A[i][j] = copiaX[i]*copiaX[i];
         }
}
/
*****************************************************************
**********/
void CreaB(double Completa[][2], double B[], int tipo, int d, int N)
{
```

```
    int i;

    for(i=0; i<N;++i)
      B[i] = Completa[i][1];
}
/
**********************************************************************
**********/
/*      Definición de las funciones necesarias para calcular la inversa       */
/
**********************************************************************
**********/
void Inversa(int orden, double A[][3], double IA[][3])
{
   double iden[3][3], cA[3][3];   //matriz identidad y copia de A.
   copiaMatriz(A, cA, orden);
   identidad(iden, orden);
   calcula_inversa(IA, cA,iden, orden);
}
/
**********************************************************************
**********/
void copiaMatriz(double M[][3], double cM[][3], int Orden)
{
    int i, j;                  //Contadores de filas i columnas
    for(i=0; i<Orden; ++i)
      for(j=0; j<Orden; ++j)
        cM[i][j] = M[i][j];
}
```

```
/
**********************************************************************
**********/
void identidad(double iden[][3], int Orden)
{
    int i, j;                   //Contador de filas i columnas
    for(i=0; i<Orden; ++i)
      for(j=0; j<Orden; ++j)
      {
        if(i==j)
            iden[i][j] = 1;
        else
            iden[i][j] = 0;
      }
}
/
**********************************************************************
**********/
void calcula_inversa(double IM[][3], double cM[][3], double iden[][3], int Orden)
{
    int i,j,k;            //Contadores; i representa la fila, j representa la
                          //columna y k representa la zona resuelta.

    bool falla=0;         //Detecta si el pivote es igual a cero. si el pivote
                          //es igual a cero, entonces falla = true.

    copiaMatriz(iden, IM, Orden);     //Inicializa la matriz inversa como identidad.

          for(k=0; k<Orden && !falla ; ++k)
          {
```

```
        if(k==Orden-1)
           Actualiza_las_ecuaciones(k, Orden, cM, IM);
        else
        {
                falla = cM[k][k]==0;
                    if(falla)                   //Si falla es true, cambia la posición de las
filas.
              Remiendalo(k, Orden, &falla, cM, IM);
                    if(!falla)                  //Arregla la matriz vuelve cero a los
elementos arriba y abajo del pivote.
              Actualiza_las_ecuaciones(k, Orden, cM, IM);
        }
           }
           if(!falla)
        falla = (cM[Orden-1][Orden-1]==0);
}
/
**********************************************************************
**********/
void Remiendalo(int k, int N, bool *Falla, double A[][3], double B[][3])
{
   int p;          //Variable auxiliar para apoyar a k en las columnas.
          p=k;
          do
          {
         ++p;
         *Falla = (A[p][k]==0);
          }
          while(*Falla && p<N-1);
          if(!*Falla)
```

```
   CambiaFilas(k, p, N, A, B);
}
/
**********************************************************************
**********/
void CambiaFilas(int k, int p, int N, double A[][3], double B[][3])
{
   int j;
   double Temp;        //temporales para los elementos de las matrices

        for(j=k; j<N; ++j)
   {
     Temp = A[p][j];          //Cambia matriz Original.
     A[p][j] = A[k][j];
     A[k][j] = Temp;

     Temp = B[p][j];          //Cambia matriz identidad.
     B[p][j] = B[k][j];
     B[k][j] = Temp;
   }
}
/
**********************************************************************
**********/
void Actualiza_las_ecuaciones(int k, int N, double A[][3], double B[][3])
{
   int i, j;
   double pivote = 0;

   DivideFila(k, N, A, B);    //Para volver el pivote igual a uno (1).
```

```
   for(i=0; i<N; ++i)
   {
      pivote = A[i][k];
      if(i != k)
      {
         restaFilasPorEscalar(A, N, pivote, i, k);
         restaFilasPorEscalar(B, N, pivote, i, k);
      }
   }
}
/
**********************************************************************
**********/
void DivideFila(int k, int N, double A[][3], double B[][3])
{
   int j;
   double Pivote = A[k][k];
   for(j=0; j<N; ++j)
   {
      A[k][j] /= Pivote;       //divide fila k de matriz A entre pivote.
      B[k][j] /= Pivote;       //divide fila k de matriz A entre pivote.
   }
}
/
**********************************************************************
**********/
void restaFilasPorEscalar(double M[][3], int Orden, double escalar, int i, int k)
{
    int j;
    for(j = 0; j < Orden; ++j)
```

```
    {
        M[i][j] -= escalar*M[k][j];
    }
}
/
**********************************************************************
**********/
/*      Definición de las funciones necesarias para la multiplicación        */
/
**********************************************************************
**********/
void Mult_Matrices(double M1[][Maxd], double M2[][3], double M3[][3], int NFA,
int NCA, int NCB)
{
   int i, j, k;

         for(i=0; i<NFA; ++i)
                  for(j=0; j<NCB; ++j)
                  {
                  M3[i][j]=0;
                           for(k=0; k<NCA; ++k)
                                    M3[i][j] += M1[i][k] * M2[k][j];
                  }
}
/
**********************************************************************
**********/
void Mult_Matrices2(double M1[][3], double M2[][Maxd], double M3[][Maxd], int
NFA, int NCA, int NCB)
{
```

```
   int i, j, k;

        for(i=0; i<NFA; ++i)
                for(j=0; j<NCB; ++j)
                {
        M3[i][j]=0;
                        for(k=0; k<NCA; ++k)
                                M3[i][j] += M1[i][k] * M2[k][j];
                }
}
/
**********************************************************************
**********/
void Mult_Matrice_vector(double M[][Maxd], double V[], double Va[], int NFA, int
NCA)
{
   int i, j, k;

        for(i=0; i<NFA; ++i)
                {
        Va[i]=0;
                        for(k=0; k<NCA; ++k)
                                Va[i] += M[i][k] * V[k];
           }
}
/
**********************************************************************
**********/
/*      Definición de la función para transponer matrices
```

```
/
**********************************************************************
**********/
void transpuesta(double M[][3], double Mt[][Maxd], int filas, int columnas)
{
        int i, j;

   for(i=0; i < filas; ++i)
      for(j=0; j < columnas; ++j)
         Mt[j][i] = M[i][j];
}
/*********************************************************************/
```

Anexo 6: Estructura que define los tipos de datos a utilizar.

```
//---------------------------------------------------------------------------
#define maxDatos 250
#define maxXYZ 250000
//---------------------------------------------------------------------------
typedef struct Pozo{
     double TablaTZ[maxDatos][2];    //Tabla tiempo(simple en milisegundos)
profundidad
     float Este;                 //Este.
     float Norte;                 //Norte.
     int Numero;                  //Número identificador del pozo.
     int Muestras;                //Número de muestras del pozo.
     char Nombre[16];             //Nombre del pozo.
```

```
    }Tipo_Pozo;
//---------------------------------------------------------------------------
typedef struct Polinomio{
    float Este;              //Este.
    float Norte;             //Norte.
    double coeficientes[3];  //coeficientes del polinomio de 2do grado.
    }Tipo_Polinomio;
//---------------------------------------------------------------------------
typedef struct Superficie{
    double X[maxXYZ];        //Este.
    double Y[maxXYZ];        //Norte.
    double T[maxXYZ];        //Tiempo doble en milisegundos.
    double Z[maxXYZ];        //Profundidad.
    double Vpromedio[maxXYZ];      //Velocidad promedio asociada a la superficie
    double Vintervalica[maxXYZ];   //Velocidad interválica asociada a la superficie.
    char Nombre[32];         //Nombre de la superficie.
    }Tipo_Superficie;
//---------------------------------------------------------------------------
```