UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

DESARROLLO DE SOFTWARE PARA ANÁLISIS GEOESTADÍSTICO CON APLICACIÓN EN GEOTECNIA

TESIS
QUE PARA OBTENER EL TÍTULO DE
INGENIERA EN COMPUTACIÓN PRESENTA:

Claudia Bibiana Martínez Torres

Director: M. en C. Moisés Juárez Camarena

México, D. F. 2014

Agradecimientos

A la Universidad Nacional Autónoma de México por la formación que me proporciono a nivel profesional, personal y cultural desde mis inicios en el Colegio de Ciencias y Humanidades, en la Facultad de Ingeniería, en el CELE y en todas las actividad artísticas realizadas dentro de su campus.

Al Instituto de Ingeniería UNAM y Laboratorio de Geoinformática por brindarme la oportunidad y el apoyo para realizar mi tesis y el servicio social.

A mi tutor M. en C. Moisés Juárez Camarena por su apoyo y tiempo durante la realización de la tesis y el servicio social.

A mis sinodales: Dr. Gabriel Yves Armand Auvinet Guichard, M.I. Aurelio Adolfo Millán Nájera, M.I. Abigail Serralde Ruiz e Ing. Pedro Osnaya Medrano por su disposición para revisar este tema de tesis.

A mis padres por haber apoyado cada una de mis etapas de vida hasta llegar a este punto.

A mi hermano por tantas risas que me ha dado.

A mis tías, tíos, primos y familia por apoyarme una y otra vez.

A mis amigos de la facultad, pero en especial, Juan Carlos, “Brendito”, Sebastián, Taco, Joel, Nan, Pix, Alicia por tantos recuerdos, risas, salidas, los momentos que hemos pasado juntos y que quedan con nosotros.

A mis grandes amigas Dulce y Ana, por su inagotable paciencia y amistad.

A los nuevos amigos que conocí en la estadía en el IINGEN, sin algunos no habría podido entender nada del trabajo y todos hicieron más amena y divertida la estadía; Marco, Beto, Jorge, Christian, Rodrigo, Omar, Arturo y Claudio.

Dedicatoria

Con todo mi cariño para las personas que han hecho todo para motivarme y apoyarme, mis padres, Nicolás y Angelina.

Por ser un ejemplo para mis padres y para mí, gracias a su sabiduría influyeron en mi vida para llegar a ser la persona que he llegado a ser. A mis amados abuelos.

Por ser siempre mi cómplice, y aunque no siempre estemos de acuerdo, has llenado mi vida de risas. Mi hermano, Fernando.

A esas personas importantes en mi vida, que siempre han estado listas para brindarme toda su ayuda, paciencia y simplemente su compañía, les dedico esta tesis ya que nunca me dejaron rendirme.

A Oso.

Índice:

1 Introducción

1.1 Antecedentes

Tradicionalmente en la geotecnia la caracterización del subsuelo se realiza de forma subjetiva, es decir, de acuerdo con la experiencia y criterios del especialista. Las formas convencionales empleadas para la presentación de los datos de las propiedades del suelo: las tablas, los perfiles geotécnicos y los cortes estratigráficos, así como conceptos básicos de la estadística descriptica clásica.

Actualmente, se cuenta con nuevas herramientas que presentan un gran potencial pero que se han usado poco, hasta ahora en la geotecnia, para describir la distribución espacial de las propiedades el subsuelo para fines de la Ing. Civil, una de ellas es la Geoestadística.

Los orígenes de la Geoestadística se encuentran en el campo de la minería. Como antecedentes suelen citarse trabajos de Sichel (1947, 1949), Krige (1951) y Matheron (1962).

La Geoestadística puede definirse como un conjunto de técnicas basadas en la teoría de los campos aleatorios y del tratamiento de las señales aplicadas a la descripción de las condiciones estratigráficas y a la distribución espacial de las propiedades de los materiales geológicos (Auvinet,2002).

Las principales diferencias entre la Geoestadística y la estadística clásica son:

1. La procedencia de los datos: En la estadística clásica no se toma en cuenta la procedencia o ubicación de los datos, mientras que en la Geoestadística es necesario conocer la posición espacial de todos y cada uno de los datos. La posición de los datos está basada en un sistema coordenado convencionalmente definido.

2. La dependencia espacial: En la estadística clásica, los valores se consideran independientes, por el contrario, en Geoestadística se considera de manera implícita que están correlacionados unos con otros, es decir, que existe una dependencia espacial. Intuitivamente esto indica que mientras más cerca se encuentren dos puntos de datos estarán más correlacionados y mientras más separados la correlación será menor. Entre mayor sea la correlación se tendrá un medio geológico más homogéneo. Para evaluar esta correlación se recurre a la teoría de los campos aleatorios.

La Geoestadística ha sido ampliamente aplicada en diversas ramas de las ciencias aplicadas y en las ingenierías, como son:

Petróleo: Elaboración de modelos geológico, petrofísicos de yacimientos, análisis de permeabilidad absoluta y su escalamiento, caracterización de yacimientos, integración de información, análisis de riesgo, evaluación de reservas.

Hidrogeología: Solución de problemas inversos (permeabilidad, transmisividades), Estimaciones de los niveles piezométricos, diseño de redes óptimas de monitoreo, Estimación de los límites de la pluma de un contaminante.

Minería: Estudiar la factibilidad de un yacimiento, cálculo de reservas, cálculo y estimación de la varianza del yacimiento, pronóstico de las variaciones de la mena a través de simulación.

Medio ambiente: Estimación de contaminantes (atmósfera, suelo, cuerpos de agua), estimación de contaminantes in situ, estudios de riesgo e impacto ambiental.

Salud pública: Análisis de la distribución espacial de enfermedades, estimación de la exposición de personas a elementos nocivos (acústicos, químicos, polvos, etc.).

Industria forestal y agrícola: Estudio de la distribución espacial y la afectación de plagas, inventarios forestales, estudio cuantitativo de los suelos y sus propiedades químicas y mecánicas.

Industria pesquera: Estimación in situ de la potencialidad de pesca, relación entre la distribución espacial de especies de peces y diferentes variables (profundidad, temperatura, salinidad, etc.).

En México el método geoestadístico se ha aplicado satisfactoriamente a la minería, específicamente al análisis de la variabilidad de los depósitos de carbón (Auvinet, 1984) y en algunas otras ramas de las ciencias.

Por otra parte, la Geotecnia es una rama de la Ingeniería Civil que se ocupa del estudio de los suelos para fines de análisis y diseño de cimentaciones de edificaciones y de estructuras geotécnicas, con base en el estudio del comportamiento de los suelos y rocas en términos de su resistencia mecánica, deformación y permeabilidad a los fluidos, tanto en condiciones naturales como cuando éstas son modificadas por el hombre. Así mismo, cada una de estas características se expresan por medio de parámetros de resistencia, deformación y permeabilidad respectivamente, que se obtienen a partir de ensayos de campo, laboratorio e instrumentación. De esta forma se determinan, por ejemplo, la resistencia al esfuerzo cortante de un suelo que a su vez se emplea para calcular la capacidad de carga de la cimentación de un edificio, o bien, la compresibilidad del suelo para calcular los asentamientos a largo plazo.

Asimismo, en los últimos años se han presentado varios trabajos sobre la aplicación del método geoestadístico en la Geotecnia. Los primeros trabajos de este tipo fueron aplicaciones a la estructura de medios granulares (Auvinet, 1968), a la estratigrafía de minas de carbón (Auvinet, 1984), al control de materiales compactados (Auvinet y Abaziou, 1993) y a la descripción de la variabilidad espacial de las propiedades de los suelos naturales o estabilizados (Auvinet, 1994), se incluye el estudio realizado en México para la cimentación del puente de Rion-Antirion ubicado en Grecia, elaborado por Auvinet y Medina (1998).

Existen algunos trabajos antecedentes sobre la aplicación de la Geoestadística a la caracterización del subsuelo del valle de México, los cuales han sido realizados con el propósito de evaluar la eficiencia del método aplicado a la Geotecnia (Juárez y Auvinet, 2000; Juárez, 2001) y a la descripción del subsuelo del Valle de México (Medina, 2001; Juárez y Auvinet, 2002), mostrando que el método puede ser empleado para estimar de manera racional los espesores y profundidades de las formaciones típicas del subsuelo y

las propiedades índice o mecánicas del subsuelo a partir de una base de datos de sondeos disponibles. También se han realizado algunos trabajos de caracterización geotécnica para la zona norte del valle de México (Laboratorio de Geoinformática, 2007), pero con algunas limitaciones debido a la escasez de datos.

Los trabajos antes descritos han sido realizados empleando herramientas donde el usuario tiene una participación parcial. Los programas no realizan un análisis geoestadístico completo y es necesario el uso de procedimientos adicionales fuera del programa, provocando un aumento de tiempo para el análisis.

Entre las herramientas existentes se encuentran para realizar el método geoestadístico se encuentran: BLUEPACK (Centro de Geoestadística de Fontainebleau), ISATIS (nueva versión de BLUEPACK, Escuela de Minas de Paris - Geovariances), GEO-EAS (Enviromental Protection Agency, U.S.A.), GSLIB (C. Deutsch y A. Journel, U. de Stanford), GS+ (Gamma Design Software, nueva versión de GSLIB), SGeMS (Nicolas Remy, U. de Stanford).

Estos programas aunque están bien diseñados no cubren las necesidades de aplicación en la Geotecnia, por lo que, resultan limitados y poco versátiles para realizar un análisis geoestadístico completo.

El programa SAAG (Sistema de Apoyo al Análisis Geoestadístico) es un programa que fue desarrollado en lenguaje de programación C-Builder (Dávalos, Medina y Auvinet, 2001). Es un programa que permite calcular la función de correlación espacial, la estimación puntual mediante la técnica de Kriging simple y la simulación de perfiles de la propiedad de interés del subsuelo, a partir de los datos de sondeos. Las limitantes que presenta este programa son: Los análisis se realizan por etapas, en las que el usuario toma participación directa al realizar algunos cálculos en forma personal fuera del programa, lo que amplía el tiempo para realizar un análisis. Otra limitante es que es un programa que fue desarrollado para trabajar en sistemas de 32 bits.

Por lo anterior, la importancia del desarrollo de software aplicado en Geotecnia es que debe cubrir las necesidades del usuario para desarrollar su trabajo de una manera más rápida y eficiente, dicho de otra forma, lo que se busca al incluir un software es tener un adecuado desempeño en un tiempo mínimo. Por esta razón se necesita una herramienta exclusiva para la Geotecnia que facilite la aplicación de las metodologías geoestadísticas integradamente.

1.2 Objetivos

El objetivo es desarrollar una aplicación computacional que implemente algoritmos eficientes para el análisis geoestadístico y que sirva como herramienta para el análisis de las propiedades índice y mecánicas del subsuelo de datos en 2D y 3D. Adicionalmente, se procura que esta aplicación sea amigable para el usuario, permita realizar un análisis completo, paso a paso, y sobre todo que tenga el menor costo de rendimiento de procesador.

1.3 Alcance

El trabajo de tesis se enfoca al desarrollo de un sistema que permita, a investigadores y estudiantes, realizar el análisis geoestadístico con aplicación particular en la Geotecnia de las propiedades índice y mecánicas del subsuelo, para lograr esto, se realizarán las actividades siguientes.

1. Revisar los programas existentes para el análisis geoestadístico.
2. Definir las necesidades de los análisis geoestadísticos con aplicación en geotecnia.
3. Definir el lenguaje o plataforma del programa.
4. Desarrollar los algoritmos optimizados para cada etapa del análisis.
5. Diseñar el diagrama de flujo de la metodología geoestadística.
6. Desarrollar un módulo para el análisis estadístico.
7. Desarrollar un módulo para el análisis estructural (correlación espacial) en 2D y 3D.
8. Desarrollar un módulo para la predicción (estimación y simulación) en 2D y 3D.
9. Elaborar los manuales de instalación y de usuario.

2 Conceptos básicos de la Geoestadística

La Geoestadística es un conjunto de técnicas usadas para analizar y predecir valores de una propiedad distribuida en espacio o tiempo. En contraposición con la estadística clásica o convencional, tales valores no se consideran independientes, por el contrario se suponen de manera implícita que están correlacionados unos con otros, es decir que existe una dependencia espacial

La Geoestadística, a diferencia de la estadística y del enfoque de variable aleatoria simple, permite analizar datos de fenómenos naturales distribuidos en el tiempo o en el espacio, para lo cual, se toma en cuenta la posición relativa que guardan los datos de la muestra.

Los conceptos básicos de esta metodología se presentan en este capítulo.

2.1 Fundamento teórico

2.1.1 Campos aleatorios

La Geoestadística considera que los valores de la variable de interés en diferentes puntos constituyen una familia o campo de variable aleatorias *V(x)*. Estas variables distribuidas dentro de un espacio R^p (p=1, longitud, p=2, superficie y p=3, volumen) pueden ser llamadas *variables aleatorias regionalizadas*, y al fenómeno representado por las variables aleatorias regionalizadas se le denomina *regionalización* o *campo aleatorio* (Auvinet, 1987). La *figura 2.1* muestra la representación gráfica de un campo aleatorio.

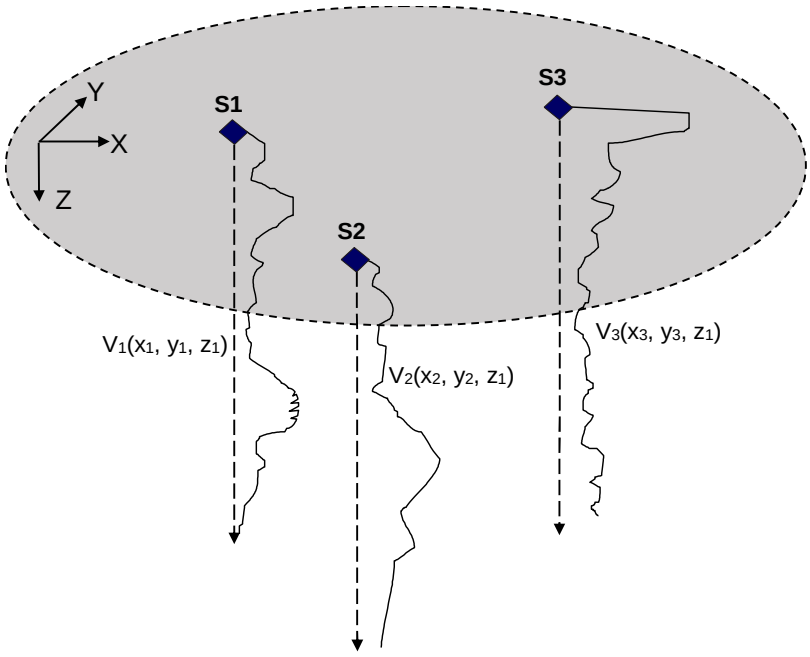


Figura 2.1 Representación gráfica de un campo aleatorio, (Medina, 2000; Auvinet, 2002)

Como se observa en la *figura 2.1*, el valor experimental de la variable de interés se conoce únicamente en algunos puntos del dominio. Además, los valores están referidos a un sistema coordenado, con el fin de establecer la posición de los valores que toma esta variable dentro de un espacio o dominio.

2.1.2 Parámetros descriptivos

Para describir este campo pueden emplearse los siguientes parámetros y funciones (Auvinet, 2002).

En el caso de sondeos continuos el *valor esperado* o *esperanza matemática* puede ser evaluado utilizando la aproximación (Auvinet, 1988):

$$E\{V(X)\} = \mu_V(X) \cong \frac{1}{L}\int_0^L V(x)dx \qquad 2.1$$

donde L es la longitud del sondeo y x una abscisa definida sobre el eje del registro.

Función de autocovarianza

En la misma forma, es posible estimar la covarianza a lo largo de la dirección **u** como:

$$C_V(\lambda\mathbf{u}) \cong \frac{1}{L}\int_0^L V(x)V(x+\lambda\mathbf{u})dx - \mu_V^2 \qquad 2.2$$

donde, **u** es el vector unitario en la dirección en la que se evalúa la covarianza y λ es un escalar.

La autocovarianza representa el grado de dependencia (parecido) lineal existente entre los valores de la propiedad de interés en dos puntos diferentes del medio.

$$C_V(X_1, X_2) = Cov[V(X_1), V(X_2)] = E\{[V(X_1) - \mu_V(X_1)][V(X_2) - \mu_V(X_2)]\} \qquad 2.3$$

Variograma

Otra técnica alternativa para describir la estructura de correlación espacial de las propiedades del suelo es utilizar el momento de segundo orden del incremento del campo aleatorio $V(X)$ o variograma, estimado en la forma siguiente:

$$2\gamma(\lambda\mathbf{u}) \cong \frac{1}{L}\int_0^L [V(X+\lambda\mathbf{u}) - V(X)]^2\, dx \qquad 2.4$$

El *variograma* puede interpretarse como un parámetro que también indica la dependencia lineal existente entre los valores de la propiedad de interés en dos puntos diferentes del medio con base en la varianza.

Coeficiente de autocorrelación

La autocovarianza representa el grado de dependencia lineal existente entre los valores de la propiedad de interés en dos puntos diferentes del medio. Se puede escribir bajo la forma de un *coeficiente de autocorrelación* adimensional, cuyo valor queda siempre comprendido entre -1 y +1. En la misma forma, el coeficiente de correlación puede escribirse como:

$$\rho_V(X_1, X_2) = \frac{C_V(X_1, X_2)}{\sigma_V(X_1)\,\sigma_V(X_2)} = \rho_V(h) \qquad 2.5$$

Correlograma experimental

La función de autocovarianza puede ser normalizada y expresada mediante un coeficiente, para lo cual se estima un valor de $\rho(\lambda\mathbf{u})$ para cada valor de $C(\lambda\mathbf{u})$. Con los valores obtenidos se construye una gráfica con $\rho(\lambda\mathbf{u})$ y $h=(\lambda\mathbf{u})$ en un sistema coordenado; en este trabajo la curva resultante es llamada *correlograma experimental*.

2.1.3 Predicción

Las técnicas que a continuación se describen brevemente son las empleadas para realizar predicciones tomando en cuenta la dependencia espacial de los datos.

Kriging Simple

La técnica del *Kriging Simp*le (Krige, 1962; Matheron, 1965; Vanmarcke, 1983; Deutsch, 1992; Auvinet, 2002) se basa en la hipótesis de que se conoce la media y la covarianza del campo aleatorio. Supóngase que se hacen mediciones de una variable de interés V en los puntos X_i, ($i = 1, 2,..., n$), del campo aleatorio estudiado, es decir, se tienen realizaciones de las variables $V(X_1)$, ..., $V(X_n)$, y se desea predecir o estimar $V^*(X)$, en el punto X donde no hubo medición. En esta circunstancia, el método *kriging simple* propone que el valor de la variable puede predecirse como una combinación lineal de las n variables aleatorias:

$$V^*(X) = \sum_{i=1}^{n} \lambda_i \cdot V(X_i) + \left[1 - \sum_{i=1}^{n} \lambda_i\right] \cdot \mu_V \qquad 2.6$$

Varianza de estimación: Asimismo, es se determina el valor de la varianza del error (minimizado) asociado a la estimación, también conocida como "*varianza de estimación*":

$$\sigma_E^2(X) = \mathrm{Var}[V(X)] + \nu - \sum_{i=1}^{n} \lambda_i \cdot \rho(X - X_i) \qquad 2.7$$

Kriging Ordinario

La técnica del *Kriging Ordinario* (Krige, 1962; Matheron, 1965; Vanmarcke, 1983; Deutsch, 1992; Auvinet, 2002) considera que no se conoce la media μ_V del campo aleatorio. Esto permite generalizar el kriging a situaciones donde esta media no es constante en el espacio: la media puede variar de una región a otra, siempre que sea aproximadamente constante en cada vecindad de *kriging*. Sólo se conoce la función de covarianza $C(h)$ o el variograma $\gamma(h)$.

Es posible encontrar un estimador lineal, sin sesgo y de mínima varianza que no requiera el conocimiento de la media μ_V, imponiendo la condición:

$$\sum_{i=1}^{n} \lambda_i \qquad 2.8$$

Siendo μ_V desconocido, para que este *valor esperado* sea nulo se debe plantear

$$a = 0 \qquad \sum_{i=1}^{n} \lambda_i = 1 \qquad 2.9$$

Varianza de estimación: La varianza del error de estimación (*varianza de estimación*) es:

$$\sigma_E^2(X) = \sigma^2 - \left[\sum_{i=1}^{n} \lambda_i \cdot \rho(X_i - X) - \nu \right] \qquad 2.10$$

2.2 Metodología de aplicación práctica en la Geotecnia

En esta sección se presenta una metodología propuesta para la aplicación particular de la Geoestadística en la Geotecnia (Juárez, 2014).

2.2.1 Análisis exploratorio

Con el fin de alcanzar una descripción satisfactoria de las variaciones espaciales de las propiedades y características estratigráficas del subsuelo, tradicionalmente se recurre al uso de las herramientas de estadística descriptiva tradicional (tablas, histogramas, gráficas correlaciones), las cuales permiten ordenar y procesar los datos disponibles para su interpretación. Los parámetros principalmente empleados son:

Medidas de tendencia central:

-Media. Denotada de la siguiente manera:

$$\bar{x} = \frac{1}{n} \sum_{i=1}^{n} x_i \qquad 2.11$$

-Mediana.
-Moda.

Medidas de dispersión o variación:

-Varianza. La varianza de la muestra de valores $x_1, x_2, \ldots, x_n$ se denota por s^2 y está dada por la siguiente fórmula.

$$s^2 = \frac{1}{n} \sum_{i=1}^{n} (x_i - \bar{x})^2 \qquad 2.12$$

-Desviación estándar. Estará denotada por *s*, y se define como la raíz cuadrada de la varianza. Es decir:

$$s = \sqrt{s^2} \qquad 2.13$$

-Coeficiente de variación. Simbolizado por *Cv*, es igual a la desviación estándar dividida entre la media:

$$c_v = \frac{s}{\bar{x}} \qquad 2.14$$

Análisis gráfico

-Histograma. Gráfica de una tabla de distribución de frecuencias de datos agrupados, *figura 2.2.*

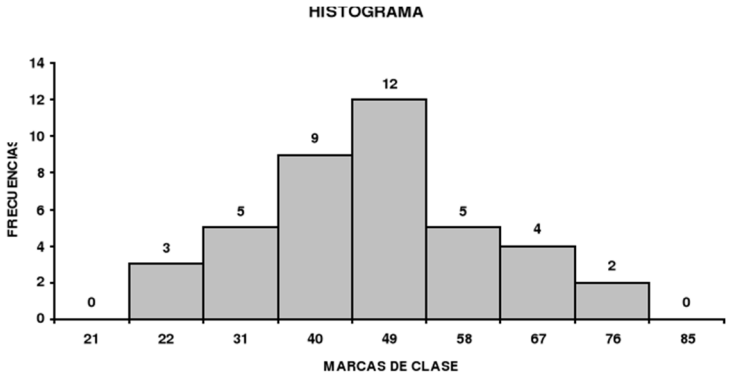


Figura 2.2 Histograma

Tendencia o deriva

Cuando la esperanza matemática no es una función constante de las coordenadas *x, y, z*, se dice que el campo aleatorio presenta una tendencia o deriva. La determinación de la tendencia se realiza generalmente mediante el ajuste de un modelo lineal (hiperplano en los casos de 2D y 3D) de la forma:

$$\begin{array}{cc} V^*(x) = \mathrm{a}x & \text{en una línea (1D)} \\ V^*(x,y) = \mathrm{a}x + \mathrm{b}y + \mathrm{c} & \text{en un plano (2D)} \\ V^*(x,y,z) = \mathrm{a}x + \mathrm{b}y + \mathrm{c}z + \mathrm{d} & \text{en un volumen (3D)} \end{array} \qquad 2.15$$

Donde a,b,c,d son los *coeficientes de regresión*. En el Anexo II, se presenta el desarrollo matemático para el análisis de regresión para los diferentes casos.

Para el caso de 2D, la tendencia puede evaluarse a partir de los *coeficientes de regresión lineal*, se construye una gráfica de regresión (recta o plano) que representa la tendencia. En la *figura 2.3* se presenta un ejemplo de regresión lineal para un análisis de datos distribuidos en un plano (2D). A partir de esta representación gráfica, el criterio para valorar si el campo es o no es estacionario:

1. Si el plano de regresión exhibe una pendiente prácticamente nula se dice que el campo no presenta tendencia y, por tanto, es *estacionario*.
2. Si el plano de regresión exhibe una pendiente considerable se dice que el campo presenta tendencia y, por tanto, es *no estacionario*.

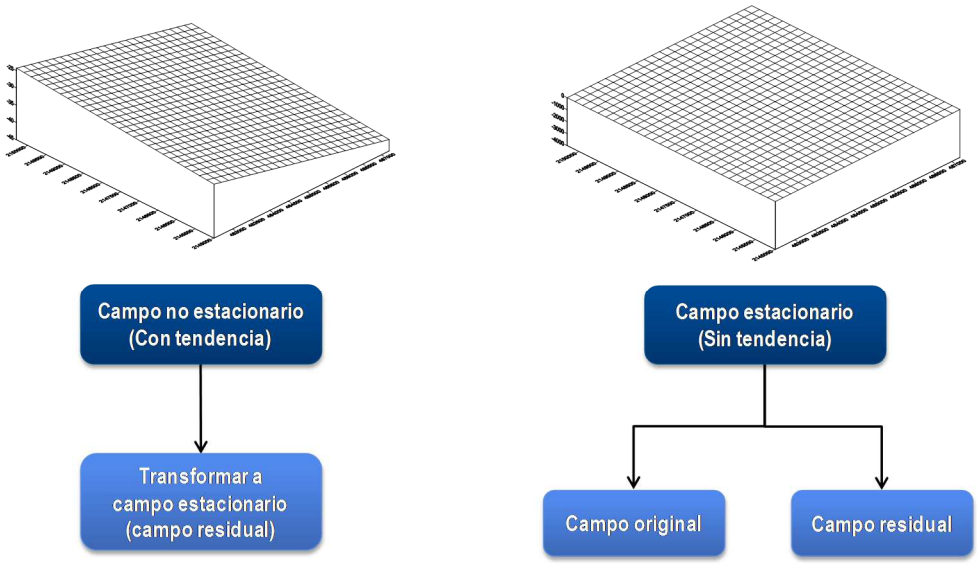


Figura 2.3 Evaluación de la estacionaridad (Juárez, 2014)

2.2.2 Análisis estructural

Consiste en determinar los parámetros y funciones (*autocorrelación, autocovarianza, coeficiente de autocorrelación*) que describen un campo aleatorio, a partir de los datos disponibles. Los parámetros de los campos aleatorios se estiman a parir de los resultados "discretos" (muestras aisladas) o "continuos" (sondeos) de las campañas de exploración.

La determinación de estos parámetros y funciones se explican en el Anexo I.

Las funciones (*autocorrelación, autocovarianza, coeficiente de correlación*) que describen un campo aleatorio se estiman para diferentes incrementos de desplazamiento h_i, tomando como valor inicial la distancia típica que existe entre los datos y hasta un incremento máximo h_{max} del 60% de la distancia que existe entre los datos más distantes.

Las funciones (*autocorrelación, autocovarianza, coeficiente de autocorrelación*) se calculan, en diferentes direcciones preferenciales, de acuerdo con el tipo de dominio o región estudiado:

- Para un plano (2D): en cuatro direcciones referidas al azimut geográfico: Az = 0°, 45°, 90° y 135°.
- Para un volumen (3D): por sencillez se consideran únicamente las direcciones vertical y horizontal.

En caso de que los datos se encuentren distribuidos en forma aleatoria es necesario proporcionar algunos parámetros que definen ciertas tolerancias lineales angulares (*figura 2.4*) para la conformación de pares de datos que permitan calcular las anteriores funciones lo mejor posible.

Finalmente, con base en las distancias de correlación y adoptando una función matemática de tipo exponencial se obtiene un modelo de correlación espacial.

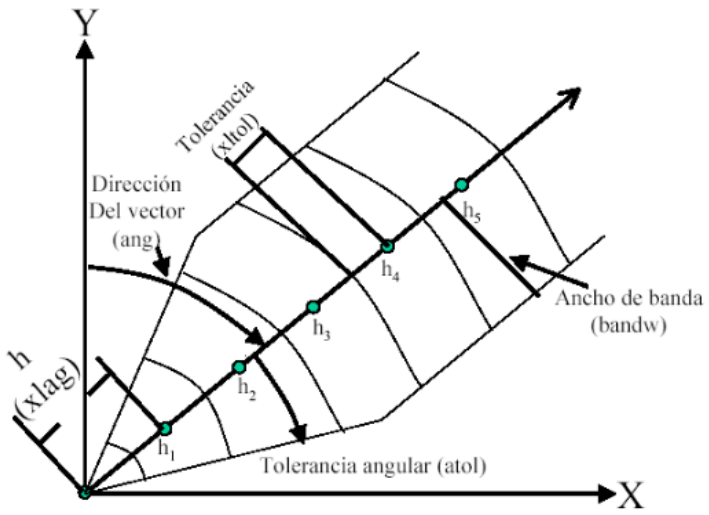


Figura 2.4 Definición de tolerancias (Deutch, 1992)

2.2.3 Predicción

A partir de un conjunto de datos (puntos de medición) es posible realizar predicciones, para las predicciones se requiere (Juárez, 2014):

1. *Definir una malla de estimación*: Si bien no hay restricciones para definir la malla de estimación, para el caso de un análisis en un plano (2D), usualmente se eligen mallas regulares (*figura 2.5*), debido a que su geometría facilita la representación gráfica de los resultados en forma de mapas de contornos, relieves, etc. Para el caso del análisis en un volumen (3D), se requiere definir la posición de los puntos a lo largo de un eje, en los que se estimarán valores a lo largo de la profundidad (*figura 2.6*), es decir, la malla de interpolación definirá la generación de perfiles estimados (virtuales) del parámetro que se analiza. Por tanto, en este caso la malla de estimación está definida por:

   - Las coordenadas de los puntos iniciales (x_1,y_1) y final (x_2,y_2) del eje de estimación.
   - El número de puntos a estimar a lo largo del eje.
   - Distancia o paso de cálculo a los largo de la profundidad, Δz.

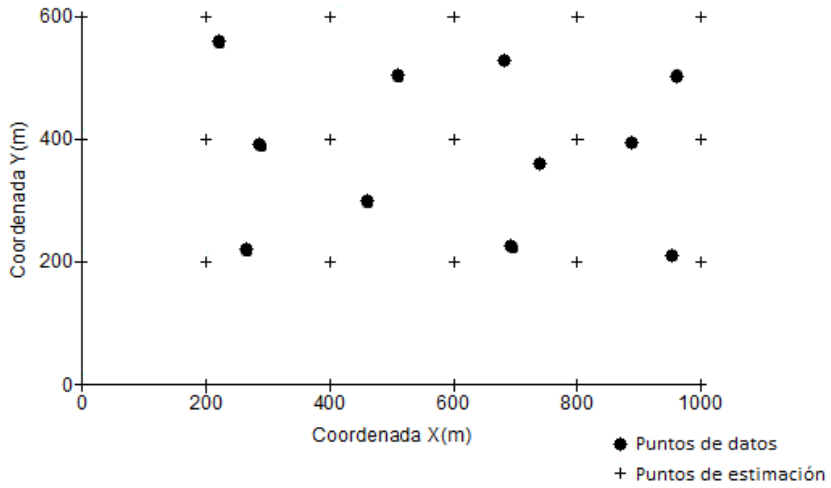


Figura 2.5 Malla de estimación para el caso del análisis en un plano (2D) (Juárez, 2014)

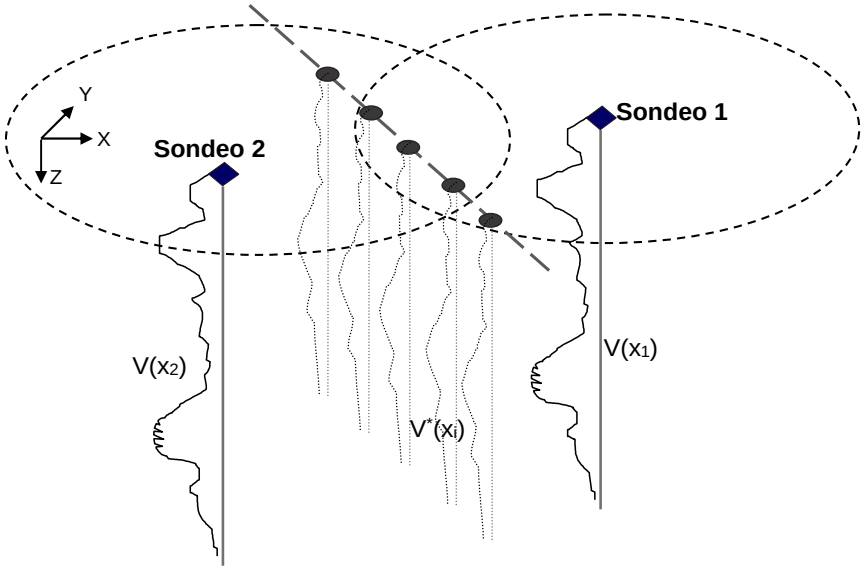


Figura 2.6 Malla de estimación para el caso del análisis en un volumen (3D) (Juárez, 2014)

2. *Definir una vecindad de búsqueda:* La vecindad de búsqueda se define como la distancia máxima, medida a partir del punto a estimar, dentro de la que se encuentran los puntos vecinos que serán tomados en cuenta para la estimación.

3. *Parámetros de correlación espacial:* Se definen las distancias de correlación, en cada una de las direcciones preferenciales de cálculo, así como la función de ajuste del *correlograma experimental*.

En caso de que el análisis se realice con el *campo residual*, la *predicción* del campo se realizar considerando los datos del campo estacionario y, posteriormente, al campo resultante se le agrega la tendencia eliminada.

La predicción se puede realizar de dos formas, estimación o simulación:

a) Estimación

La estimación consiste en obtener el *valor esperado* de una variable en un punto sin medición.

El modelado de las variaciones espaciales de las propiedades del suelo mediante campos aleatorios permite realizar estimaciones del valor de dichas propiedades en puntos en los que se carezca de medición tomando en cuenta la correlación espacial. Para ello, se puede recurrir a la técnica de *Kriging* (anexo I). Esta técnica fue desarrollada por Matheron (1965). Esta técnica permite encontrar el mejor (en el sentido de la mínima varianza) estimador lineal sin sesgo, tomando en cuenta la correlación espacial, supuesta conocida.

b) Simulación

La *simulación*, consiste en obtener el *valor posible* de una variable en un punto sin medición.

La *Simulación* (anexo I) es el proceso por medio del cual se genera una posible configuración del campo aleatorio compatible con sus parámetros descriptivos (simulación incondicional) o con estos parámetros y, además, con los datos disponibles (simulación condiciona). Se genera así una realización o imagen del campo que permite apreciar en particular valores extremos potencialmente problemáticos.

La forma más sencilla de simular un campo aleatorio consiste en considerar que el mismo está representado por n puntos x_1, x_2,...,x_n donde, para cada punto, se deben obtener realizaciones del grupo de variables aleatorias $V(x_1)$, $V(x_2)$,…$V(x_n)$, con la estructura correcta del campo en cuanto a esperanza y matriz de covarianza. La simulación se realiza generalmente sobre una malla de puntos en el dominio de interés y se reduce por tanto, a generar un cierto número de variables aleatorias conjuntamente distribuidas.

> *Simulación incondicional*. Este tipo de simulación requiere inicialmente la generación de una secuencia de valores de *variables aleatorias estándar independientes distribuidas normalmente* (con media cero y varianza unitaria), obtenidas a partir de dos variables aleatorias distribuidas uniformemente entre 0 y 1.
>
> *Simulación condicional* (Anexo I). Se supone que el campo aleatorio $V(x)$ ha sido medido en los puntos x_1, x_2,…,x_n y que será simulado en los puntos x_{p+1}, x_{p+2},…,x_{p+n}. Se desea generar realizaciones de $V(x)$ que igualen de manera exacta los datos en p puntos y que sean aleatorias en los n-p puntos restantes. Este tipo de simulación es la que interesa en efectos de la herramienta desarrollada.

En la *simulación condicional de un campo aleatorio* se usan las esperanzas, varianzas y covarianzas condicionales sobre los datos disponibles.

2.2.4 Visualización

La interpretación de los resultados numéricos de la estimación (interpolación) o simulación ordenados en forma tabular, no es sencilla; por lo que, se recurre a técnicas de graficación avanzadas (programas comerciales) para construir perfiles, secciones transversales (cortes), mapas de contornos o modelos de superficies, a partir de los valores puntalmente estimados. Las representaciones gráficas permiten apreciar visualmente la distribución espacial de las propiedades estudiadas.

2.3 Estudio y comparación de programas existentes

A continuación se presenta una lista de software existente de tipo libre y comercial (*tabla 2.1*), para realizar análisis geoestadísticos.

Tabla 2.1 Programas existentes para análisis geoestadísticos

Software	Alcances
Surfer	Interpolación, Visualización
GS+	Variogramas, Kriging, Simulación, Mapeo, dos y tres dimensiones
Isatis	Manejo de datos, Variogramas, Kriging, Cokriging, Simulación, Mapeo, dos y tres dimensiones
WinGslib	Manejo de datos, Hitogramas, Variogramas, Kriging, Cokriging, Simulación, dos y tres dimensiones
GSLIB	Variogramas, Kriging, Cokriging, Simulaciones, Solo tres dimensiones
Geo-EAS	Variogramas, Kriging, Solo dos dimensiones
GeoPack	Variogramas, Kriging, Cokriging, Solo dos dimensiones
R	Variogramas, Kriging, Cokriging, Simulaciones, Solo tres dimensiones
SADA	Variogramas, Kriging, Otros métodos de estimación, Funciones GIS, Solo tres dimensiones
SAGA GIS	Variogramas, Kriging, Otros métodos de estimación, Funciones GIS, Solo dos dimensiones
Variowin	Variogramas, Solo dos dimensiones
Vesper	Variograma, Kriging, Solo dos dimensiones
GeoDa	Otros métodos de estimación, Solo dos dimensiones
SAAG	Función correlación espacial, Kriging simple, Simulación, dos y tres dimensiones

Como se observa en la *tabla 2.1* prácticamente todos los programas emplean el variograma para definir el modelo de correlación espacial. En este trabajo para el desarrollo de la herramienta se emplea la función de autocovarianza y de autocorrelación por tener este parámetro algunas ventajas sobre el variograma.

Además, se observa que solo algunos programas realizan análisis en dos y tres dimensiones, lo que limita sus alcances de aplicación en la Geotecnia. Así también, la mayoría de ellos están constituidos por módulos que trabajan en forma independiente, lo que constituye una limitante, puesto que, amplía el tiempo para completar un análisis.

Por otra parte, la mayoría de estos programas no tienen formato de archivos compatibles con los programas usuales de trabajo (Excel, Word…), lo que dificulta el manejo e interpretación de los resultados, obligando al uso de otros programas.

Por lo anterior, en el desarrollo de la herramienta se procuró resolver las limitaciones anteriormente expuestas.

3 Fundamentos teóricos

En este capítulo se definen los conceptos relacionados con software, aplicaciones de escritorio, paradigmas y lenguajes de programación, almacenamiento, algoritmos, diagramas de flujo y graficación. Estos conceptos son aplicables a la construcción del proyecto del software desarrollado en esta tesis.

3.1 Software

Es llamado software a la parte lógica de una PC o computador, es decir, es el conjunto de programas que pueden ser ejecutados en un computador. En una definición más amplia el software no son sólo programas, sino también todos los documentos asociados, que sirven para realizar el método lógico, procedimiento o control requerido, y la configuración de datos que se necesitan para hacer que estos programas operen de manera correcta (Aggarwal K. y Sigh Y., 2006). Los componentes del software se muestran en la *figura 3.1*.

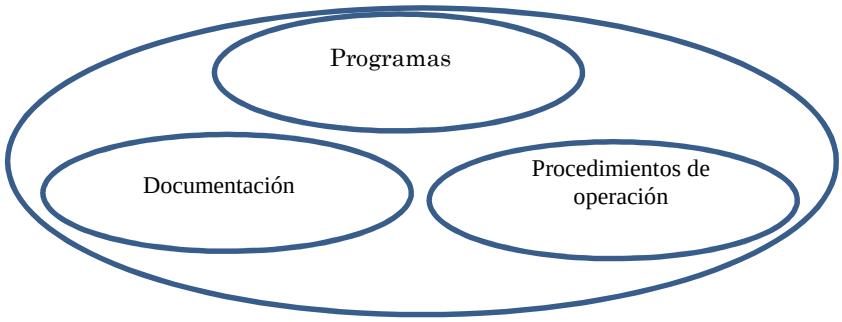


Figura 3.1 Componentes de software

Cualquier programa es un producto de software y se convierte en software solamente si documentación y respectivos manuales son preparados. El programa es una combinación de código fuente y código objeto. La documentación consiste en diferentes tipos de manuales.

El software se conforma de: *1) instrucciones* que al ejecutarse proporcionan las características, funciones y el grado de desempeño deseados; *2) estructuras de datos* que permiten que los programas manipulen información de manera adecuada; *3) documentos* que describen la operación y el uso de los programas.

Sus características son:

a. *El software es desarrollado; no se construye en el sentido clásico.* Aunque existen ciertas similitudes entre desarrollo de software y la manufacturación de hardware, las dos actividades son fundamentalmente diferentes. En ambas actividades se depende de gente, requieren construcción de un “producto”. Pero los enfoques son diferentes. El costo del software está concentrado en la ingeniería, esto significa que los proyectos de software no puede ser manejados como proyectos de manufacturación.

b. *El software no se desgasta.* Existe una curva conocida como "curva bañera" en estudios de confiabilidad en los productos de hardware, *figura 3.2*. Esta curva representa a la tasa de fracaso en función del tiempo para el hardware.

Figura 3.2 Curva de Hardware

La curva para los proyectos de software es como la que se presenta en la *figura 3.3*.

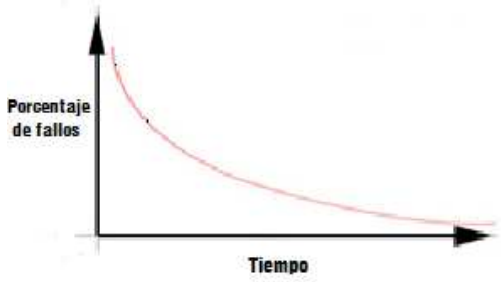


Figura 3.3 Curva de Software

Un aspecto importante es que el software se vuelve fiable con el tiempo en lugar de desgastarse. Sin embargo, se vuelve obsoleto si el ambiente para el que se desarrolló cambia.

c. *La mayoría del software aún se construye a la medida, en lugar de ensamblado a partir de componentes existentes*. La mayor parte del software continua siendo a la medida, aunque desarrollo reciente tiende a ser basado en componentes.

d. *El software es flexible.* Se puede desarrollar un programa para hacer prácticamente casi cualquier cosa. Algunas veces esta característica nos puede ayudar a dar cabida a cualquier tipo de cambio. Reuso de componentes de librerías ayuda a reducir el esfuerzo. Actualmente, se reúsan no solamente algoritmos sino también estructuras de datos.

3.1.1 Clasificación

El software se puede clasificar en muchas categorías usando diferentes criterios. Una clasificación puede ser la siguiente:

1. *Programas de control*, que controlan y supervisan la ejecución de todas las tareas y procesos que tienen lugar en la computadora.

2. *Programas de proceso*, que sirven para que el usuario cree sus propios programas.

3. *Programas de aplicación*, que son los desarrollados por el usuario de la computadora para resolver problemas específicos.

Los dos primeros tipos reciben el nombre de “software de sistema” porque son programas desarrollados para el correcto funcionamiento de la computadora. Mientras que al tercer tipo se le denomina “software de aplicación” porque son programas desarrollados para resolver los problemas de los usuarios.

Existen tres tipos de programas de aplicación:

1. *Productos genéricos*. Son sistemas aislados producidos por una organización de desarrollo y que se venden al mercado.

2. *Productos personalizados (o hechos a medida).* Son sistemas requeridos por un cliente particular.

3. *Sistemas genéricos adaptados a clientes específicos.*

3.2 Aplicación de escritorio

Una aplicación de escritorio es un programa que se utiliza como herramienta para una operación o tarea específica a personas individuales, grupos de trabajo y empresas. Se contrasta con el software del sistema que gestiona e integra las capacidades de un ordenador, pero normalmente no las aplican directamente a las tareas que benefician al usuario.

Una aplicación tiene connotaciones diferentes según su entorno pero generalizando, es un subsistema de información que trata de resolver una problemática, con una misma finalidad y una fuerte cohesión entre sus componentes. A menudo se componen de varios archivos que interactúan entre sí con instrucciones que rigen a la computadora.

Las aplicaciones de escritorio corren en el escritorio de la computadora del usuario final en una ventana separada que permite interactuar con esa aplicación particular. Estas aplicaciones tienen una interfaz de usuario mucho más completa que su versión contraparte aplicación Web. Estas aplicaciones también se ejecutan más rápido en términos de refrescamiento de pantalla (Kanalakis, 2003)

Dos puntos importantes deben tenerse presentes para tomar la decisión de desarrollar una aplicación de escritorio:

- **Acceso global:** Se puede dividir en dos partes (conectividad y plataforma objetivo). En si la conectividad no es tan importante ya que casi todo tipo de aplicaciones se puede tener conectividad, lo importante es la plataforma objetivo. Si es conocido que la plataforma objetivo es ciertamente una plataforma de Microsoft Windows, entonces una aplicación de escritorio es ideal.

- **Acceso a los recursos locales de la máquina:** Si los requerimientos de la aplicación es que necesita acceso a recursos locales tales como archivos de datos, registros, u objetos (COM), entonces una aplicación de escritorio será requerida.

El proceso de desarrollo de aplicaciones típicamente se divide en seis pasos específicos: captura de requerimientos, prototipo, diseño, desarrollo, testeo y liberación de la aplicación.

3.2.1 Características

- Son interactivas. El plan de ejecución de programas interactivos puede ser menos predecible que los caminos de aplicaciones de lotes ya que el usuario proporciona la entrada que dirige la ejecución durante el tiempo de vida de la aplicación.

- Son gráficas. Mientras que los resultados las aplicaciones de referencia tradicionales son archivos de texto que consisten en unas pocas páginas de salida, muchas aplicaciones de escritorio muestran cientos de widgets gráficos y fotos en la pantalla, además de generar la producción convencional de archivos.

- Multiproceso: Tienden a ser multiproceso, tanto como un mecanismo de estructuración y como una manera de ocultar operaciones de latencia desde el usuario.

- Las aplicaciones de escritorio son generalmente diseñadas específicamente para el sistema operativo donde está destinado a ejecutarse, por lo que, tienen la ventaja de basarse en los servicios o funciones característicos del sistema operativo.

- Son ejecutadas directamente de una computadora local y no necesariamente necesitan conexión a internet.

- El acceso a los datos almacenados localmente es mediante el acceso de archivos.

- Pueden tener funciones más avanzadas que interactúan y se integran mejor con el sistema operativo local.

3.2.2 Ventajas

Al crear una aplicación de escritorio se puede tomar ventaja del hardware que se encuentra en la máquina.

Con las aplicaciones de escritorio, el rendimiento es generalmente más rápido, en este caso la pantalla se dibuja una sola vez y solamente los datos se cambian. Esto evita una gran cantidad de datos de pantalla que se obtienen desde el servidor hasta el cliente, lo que aumenta el tiempo necesario para visualizar los datos en pantalla.

Al programar una aplicación de escritorio, se tiene un conjunto más rico de controles para la construcción de la interfaz de usuario. Esta característica tiene la ventaja de que la interfaz de usuario pueda tener la apariencia que se desee.

Si una aplicación de escritorio necesita integrarse con cualquier hardware especial, máquinas de control numérico, impresoras y escáneres, es mucho más fácil producir una interfaz ordenada utilizando una aplicación de escritorio que desde un navegador.

Si se necesita integrar con otros productos, será mucho más fácil de hacer a partir de una aplicación de escritorio que desde un navegador.

Brindan una aplicación personalizada y se crean a partir de la necesidad de resolver un problema o de simplificar una operación compleja o tardada.

3.2.3 Desventajas

Las aplicaciones de escritorio proveen un amplio soporte de usuario, por lo que ofrecen, una amplia gama de aplicaciones. En la actualidad hay muchos controles disponibles que es posible desviarse del objetivo principal de la aplicación de escritorio. Esto puede dar lugar a costos adicionales de capacitación, y puede causar una cierta confusión cuando los usuarios utilizan una variedad de aplicaciones.

Las aplicaciones de escritorio son difíciles de utilizar en algunos casos, sobre todo cuando el usuario trabaja desde una ubicación remota.

Uno de los mayores inconvenientes de una aplicación de escritorio se presenta cuando la aplicación se implementa para cientos o miles de usuarios y cada vez que se tiene que hacer una actualización, se tiene que instalar nuevamente por completo.

Cuando los usuarios se mueven de un sitio de trabajo a otro, tienen que preocuparse acerca de que la aplicación ha sido o no instalada.

3.3 Paradigmas de programación

Un paradigma de programación es un modelo básico de construcción de programas. Un modelo que permite producir programas conforme con unas directrices específicas, tales como diseñar un programa mediante una secuencia de instrucciones que operan sobre unos datos de entrada y producen un resultado de salida, entre otros (Alonso, Martínez, Segovia; 2012).

Un paradigma es una colección de modelos conceptuales que juntos modelan el proceso de construcción de un programa y determinan, al final, su estructura (Ambler,1992).

Existen diversos lenguajes y paradigmas de programación que se han diseñado para facilitar la tarea de la programación en diferentes ámbitos. De acuerdo con el paradigma seleccionado, el programador modifica su conceptualización de los problemas a resolver, de los resultados esperados y de los posibles algoritmos.

Para este trabajo se tomaron 4 de los paradigmas y se estudiaron más a fondo. Los paradigmas elegidos son: programación estructurada, programación orientada a objetos, orientada a eventos y programación orientada a aspectos.

Programación estructurada

Emerge en 1960s, particularmente del trabajo de Böhm, Jacopini y una carta,"Go to statement considered Harmful", de Edsger Dijkstra en 1968 (Dijkstra Edsger, 1968) y fue reforzado por la aparición de lenguajes tales como ALGOL con estructuras de control adecuadas.

En este paradigma se visualiza la solución de un problema, un algoritmo, con base en una jerarquía de procedimientos que manipulan estructuras de datos y se comunican con el medio ambiente externo. Cada procedimiento es responsable de realizar una tarea y se especifica mediante una secuencia de procedimientos de menor nivel, donde el nivel más bajo son las instrucciones básicas proporcionadas por el lenguaje. En la programación procedural, un procedimiento es una nueva instrucción que puede formar parte de un procedimiento de mayor nivel, y un programa es una secuencia de instrucciones (procedimientos).

Las tres estructuras básicas de control son:

- **Secuencia:** el bloque secuencial de instrucciones, instrucciones ejecutadas sucesivamente, una detrás de otra.
- **Selección:** la instrucción condicional con doble alternativa, de la forma "*if* condición *then* instrucción-1 *else* instrucción-2".
- **Iteración:** el bucle condicional "*while* condición *do* instrucción", que ejecuta la instrucción repetidamente mientras la condición se cumpla.

El uso de procedimientos cortos es otra idea fundamental de la programación estructurada. Un programa bien estructurado debe entregar una solución a un solo problema. La división de un problema en subproblemas se debe ver reflejada en un número de procedimientos.

"La programación estructurada está estrictamente orientada hacia la resolución de un problema particular, se realiza mediante la descomposición gradual de la funcionalidad. La desventaja es que es los sistemas reales no tienen soló una cima, pueden tener varias cimas" (Meyer, 1988).

Se dice que un programa es estructurado si cumple los principios de la programación estructurada:

Estructura jerárquica top-down.
Diseño modular.
Salidas y entradas claramente definidas.
Un único punto de entrada y un único punto de salida
Utilización exclusiva de los constructores de programas estructurados: secuencias, selección, iteración y llamada a procedimientos.

Programación orientada a objetos

Una definición de este tipo de programación puede ser la siguiente: *"Es un enfoque que provee una forma de modularizar programas creando áreas de memoria particionada para los datos y funciones respectivamente que puede ser usada como una plantilla para crear copias de módulos en demanda."* (Keda y Seema, 2008)

En el paradigma de programación orientada a objetos (POO), el procedimiento es desplazado por el objeto como centro de los programas. La organización de éstos gira ahora alrededor de los **objetos**, que encapsulan estructuras de datos junto con los procedimientos (**métodos**) que los manipulan.

Un programa está poblado por objetos independientes que modelan entidades del mundo real, cada uno responsabilizándose por sí mismo y comunicándose con otros por medio de mensajes, pidiéndose unos a otros que efectúen tareas o que informen sobres su estado, para cooperar y lograr un fin común. La unidad básica de operación sigue siendo la instrucción imperativa y la ejecución de un programa una secuencia de acciones. Sin embargo, en este paradigma las acciones son los métodos de los objetos que responden al paso de mensajes, de acuerdo con la acción más apropiada determinada por su estado.

Los objetos son, en realidad, instancias activas de clases que abstraen las características de objetos similares, tanto su estado, determinado por sus tipos de datos, como su comportamiento, especificado por sus métodos.

Las clases son tipos definidos por el usuario, que contienen datos y funciones que operan en los datos. Una vez que se crea una clase se pueden crear cualquier número de objetos pertenecientes a la clase. Generalmente, la especificación de la clase tiene dos partes: declaración de la clase y definiciones de las funciones de la clase.

Al proceso de envolver los datos en una clase es conocido como *encapsulación*. En la encapsulación los datos no son accesibles al mundo exterior. Las funciones que se envuelven en la clase pueden ser solamente accesadas mediante la clase. Esto provee seguridad a los datos, este aislamiento de los datos de acceso directo por el programa es conocido como *ocultación de información*.

La abstracción se refiere a la acción de representar características esenciales sin incluir detalles o explicaciones. Las clases usan el concepto de abstracción de información.

La programación orientada a objetos explota una estructura jerárquica de clases mediante la herencia de datos y métodos. Esta herencia permite reutilizar el código ya escrito para crear nuevas clases, provee de la idea de reusabilidad. Además de la herencia, puede utilizarse también el mecanismo de composición para construir clases complejas a partir de clases simples.

Otra característica de la programación orientada a objetos directamente relacionada con la herencia es el *Poliformismo*, esta característica significa la habilidad de tomar más de una forma, esto quiere decir que un operador puede exhibir diferentes comportamientos en diferentes instancias.

Khoshafian (1990) señala que los tres aspectos fundamentales que definen la programación orientada a objetos son:

- Los tipos abstractos de datos, que son implementados por las clases.
- La herencia como estructura jerárquica de generalización y especialización de objetos.
- La identidad de los objetos, invariante al objeto durante toda su vida.

Programación dirigida por eventos

Consiste en diseñar un programa en el que los eventos dirijan la ejecución del programa; es decir, el programa no solicita datos, ni un orden de petición de los mismos, es el usuario el que emite unos eventos que exigen al programa realizar determinadas acciones. El programa está pendiente del evento que introduce el usuario y según sea éste ejecuta una u otra operación, que puede implicar la lectura de datos o no.

Un *evento* es un estímulo ocurrido durante la ejecución de un programa, como seleccionar con el ratón o presionar una tecla. Un evento también puede ser disparado (iniciado) por actividades internas, por ejemplo, con el reloj de la PC.

Estos programas se basan en escribir código para un evento particular, el programa responderá a él en el momento que ocurra en tiempo de ejecución. Sin embargo, si no se escribe código para un evento particular y ese evento ocurre, el programa lo ignorará. El código aparece y actúa, no como una larga lista de texto, sino como varias secciones cortas de código. Cada sección está escrita para responder a determinados eventos. Cada una de estas secciones queda estática, hasta que ocurre su evento; en ese momento, el programa comienza a ejecutar el código de ese evento (proceso que se conoce como *disparar un evento*).

Un programa dirigido por eventos se inicia, espera continuamente la ocurrencia de eventos para procesarlos y termina cuando ocurre otro evento en particular. No sólo se deben manejar eventos aleatorios: si se ejecuta más de un programa a la vez, cada programa necesita analizar y responder a los eventos.

Los componentes básicos de los programas dirigidos por eventos son:

- *Sensores* que detectan y recogen eventos.
- *Distribuidores* o *Despachadores* de eventos, que los envían a los procedimientos encargados de procesarlos.
- *Manejadores* de eventos, que procesan los eventos recibidos.

El principal patrón estructural que constituye la esencia del paradigma dirigido por eventos es el llamado patrón Handler (manejador), *figura 3.4*, el cual tiene su origen en los diagramas de análisis de transacciones definidos por DeMarco, Yourdon y Constantine (1979).

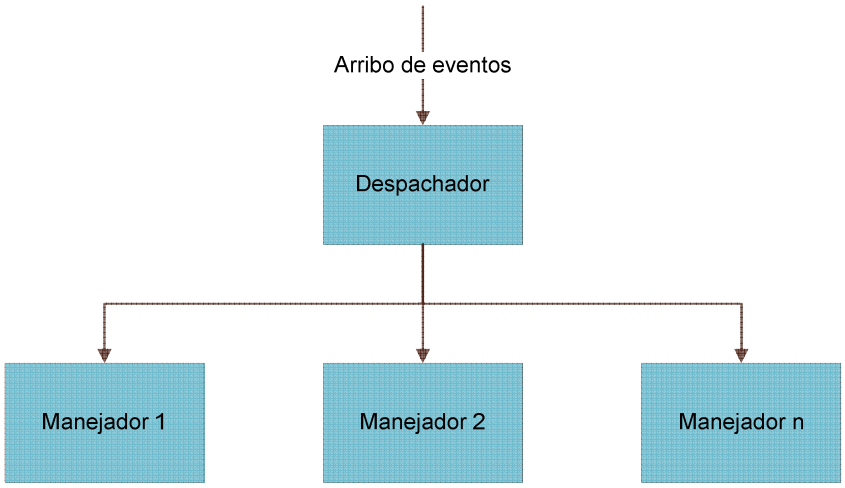


Figura 3.4 Patrón Handler

Los elementos estructurales de este patrón son: un flujo de datos llamados eventos, un despachador de eventos y un conjunto de manejadores.

La función del despachador es tomar cada uno de los eventos que van llegando a él, analizar cada evento para determinar su tipo y finalmente enviar cada uno de éstos a los respectivos manejadores, quienes realizan alguna función de acuerdo al tipo de evento que puedan procesar.

En una interfaz gráfica (GUI), se refleja la estructura de este tipo de sistemas. Donde cada elemento de la interfaz gráfica como los botones, cajas de selección, cajas de edición entre otros, pueden estar en modo de espera hasta la llegada de un evento como lo es el oprimir algún botón del mouse sobre uno de los elementos de la GUI, una vez arribado el evento éstos realizan alguna acción. Los módulos que componen a este tipo de sistemas son: un elemento que representa los eventos generados externamente por el usuario, un despachador responsable de analizar los eventos que llegan a éste y una serie de manejadores que efectúan alguna operación dependiendo del tipo de evento generado por el usuario. El patrón Handler aplicado en un sistema GUI se muestra en la *figura 3.5.*

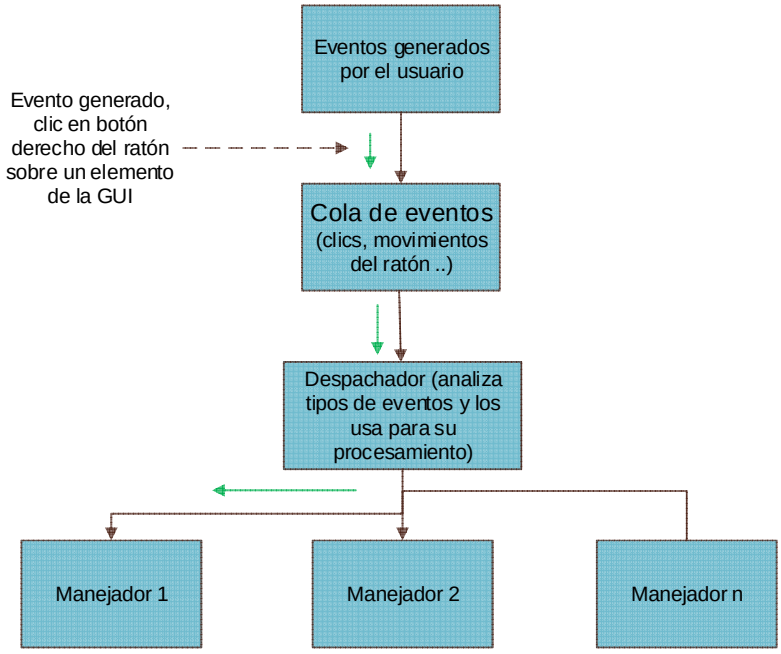


Figura 3.5 Patrón Handler aplicado en sistemas GUI

Las características más relevantes de los programas dirigidos por eventos son:

- *¿Quién decide qué hace el programa en cada momento?*: es la secuencia de eventos que se han introducido junto con sus datos asociados quién marca la secuencia de operaciones que el programa ha realizado hasta el momento.

- *¿Qué decisiones puede tomar el usuario en cada momento?*: puede provocar aquellos eventos relacionados con la interfaz y, por tanto, marcar una secuencia de proceso.

- *¿Qué carga supone al procesador este tipo de programación?*: un programa dirigido por eventos es *reactivo*, sólo trabaja cuando se recibe un evento, por lo que, el procesador no es acaparado por un solo proceso. Debe señalarse que un programa de este tipo se diseña para procesar los eventos rápidamente, dejando el procesador libre cuanto antes.

- *¿Dónde están las rutinas de entrada de datos en el programa?*: lo que existe son rutinas de tratamiento de eventos, que a su vez llaman a rutinas de procesamientos de datos asociados.

- *¿Qué implicaciones tendría cambiar el formato de entrada de algún dato?*: mínimas, al estar las rutinas de entrada concentradas en un solo sitio.

Programación orientada a aspectos

La programación orientada a aspectos (POA) es muy reciente y surge al constatar que los programas normalmente tienen funcionalidades que no se limitan a un solo módulo (procedimiento o clase), o a un conjunto de módulos relacionados.

La idea básica de la POA es proporcionar soporte avanzado para modularidad combinada con separación de intereses. La idea de la modularidad no es nueva y prácticamente todos los paradigmas de programación de la última década se han esforzado para soportar la modularidad, mientras que la programación orientada a objetos ha tenido un gran impacto. Los partidarios de la POA creen que el concepto de aspecto puede resolver algunos de los mayores problemas del desarrollo orientado a objetos que no han sido completamente tratados por algunas de los recientes avances en la POO, tales como patrones diseño.

La programación orientada a aspectos mueve a un módulo separado el código que no se relaciona directamente con la solución básica del problema y que muchas veces se encuentra repetido en módulos que tienen funciones principales diferentes. De esta forma, el código de la aplicación ya no contiene segmentos de funciones transversales dispersas en otros módulos.

Los programadores separan las funciones transversales en unidades independientes denominadas *aspectos*, facilitando el diseño, implementación y mantenimiento de las funciones de base y de las transversales. Los aspectos de un sistema pueden cambiarse, insertarse, removerse y reutilizarse. La expectativa de la programación orientada a aspectos es permitir que una aplicación pueda adoptar nuevos aspectos en cualquier momento, sin tener que modificar su funcionalidad básica. La separación es sólo a nivel sintáctico ya que el código corre integrado en tiempo de ejecución. Un lenguaje AO debe proporcionar la posibilidad de especificar la definición del aspecto fácilmente y sobre todo en un solo punto.

3.3.1 Comparación

En la *tabla 3.1* se presenta una comparación entre los paradigmas anteriormente descritos.

Tabla 3.1 Comparación de paradigmas

	Estructurado	**Orientado a objetos**	**Dirigido por eventos**	**Orientado a aspectos**
Basado	Máquina de Von Neumann	Modelar objetos del mundo real como forma de encapsular algoritmos y datos en una unidad.	Sucesos detectados por el sistema tendrán la posibilidad de ser un evento para el programa, y desencadene un procesamiento y posiblemente más eventos.	Aspectos diseminados por todo el sistema como entidades por separado.
Características importantes	Considera únicamente subrutinas, y tres estructuras: secuencia, selección e iteración. Impone restricciones en diseño de algoritmos que facilitan su posterior modificación y mantenimiento.	Trata la información como objetos abstractos manipulados mediante métodos pre-definidos. Conceptos de herencia y poliformismo.	El flujo del programa es determinado principalmente por eventos, como clics del mouse o el timer. A diferencia con la programación estructurada donde el programador define el flujo del programa, el usuario definirá el flujo.	Separa funcionalidades: funcionalidades a lo largo de la aplicación, funcionalidades de cada módulo, y las encapsula en unidades llamadas aspectos. Conceptos de join point, pointcut, weaving.

3.3.2 Selección

Para el desarrollo de la aplicación de este trabajo se escogió la programación dirigida por eventos, puesto que, como se expuso anteriormente es frecuentemente usada para las interfaces gráficas. Como se puede ver en la *figura 3.5* se tienen eventos generados por el usuario que pasan a través de una cola de eventos al despachador, y éste al manejador correspondiente, esto desencadenará una acción. Este procedimiento es conveniente ya que el usuario podrá enviar distintos tipos de eventos en la interfaz gráfica.

También se escogió este tipo de programación debido a que el usuario tendrá diferentes opciones a realizar durante el análisis. Aunque la idea principal sea un análisis guiado, en un inicio tendrá diferentes opciones a escoger y no siempre será el mismo procedimiento a seguir; por ejemplo, al iniciar el programa se debe escoger entre 2 dimensiones o 3 dimensiones, otro ejemplo puede ser que quiera crear un archivo, modificarlo o pasar directamente al análisis.

3.4 Lenguajes de programación

Independientemente del paradigma de ingeniería del software que se adopte, el lenguaje de programación tendrá impacto en la planificación, el análisis, el diseño, la codificación, la prueba y el mantenimiento de un proyecto. El papel que representa el lenguaje de programación ha de tenerse presente en todo momento. Los lenguajes proporcionan los medios de la traducción hombre-máquina.

Un lenguaje de programación es un formalismo artificial en el que los algoritmos pueden ser expresados. A pesar de su artificialidad, este formalismo permanece como un lenguaje. Su estudio puede hacer un buen uso de los muchos conceptos y herramientas desarrolladas en el último siglo en la lingüística (que estudia los lenguajes tanto naturales como artificiales) (Gabbrielli y Martini, 2010).

Un lenguaje de programación es un lenguaje no-natural, creado para la comunicación con las maquinas. Son los medios de expresión en el arte de la programación de computadora. Un lenguaje de programación ideal facilitará al programador para escribir programas en forma clara y sencilla, puesto que, los programas deben ser capaces de entenderse, modificarse, y mantenerse a través de su ciclo de vida. Un adecuado lenguaje de programación ayudará a otros desarrolladores a leer los programas y entender cómo trabajan.

Hoy en día existen los lenguajes de alto y bajo nivel, los lenguajes de programación de alto nivel tienen un fuerte parecido a un lenguaje humano. Los lenguajes de bajo nivel están justamente por encima de los lenguajes máquina, ellos no necesitan de mucha traducción como los lenguajes de alto nivel, pero son más difíciles de entender.

Para que una máquina “entienda” y opere las instrucciones de los leguajes de alto nivel, existen entre los lenguajes de programación y un lenguaje máquina una serie de traducciones. Estas traducciones se llevan a cabo por un traductor, los traductores convierten lenguajes de bajo y alto nivel en lenguaje máquina que puede ser entendido

por el procesador en el CPU. Trabajan de dos formas, como intérpretes o compiladores. Los *intérpretes* traducirán una línea de código a la vez y generan mensajes de error inmediatamente. Los *compiladores* generarán mensajes de error hasta que todo el código ha sido traducido. El programa que el programador escribió se llama código fuente. El código objeto es el resultado de la traducción y es la versión de lenguaje máquina del programa original. Un ejemplo de lenguaje compilado es C++, y un ejemplo de interpretado es BASIC.

La tendencia actual es buscar lenguajes que se parezcan en mayor medida al lenguaje natural de los humanos. Los lenguajes han evolucionado con el tiempo, actualmente su evolución se encuentra en lenguajes de cuarta generación (4GL). Las llamadas *técnicas de cuarta generación* están cambiando las ideas del término "lenguaje de programación". Más que codificar, los que desarrollan algún tipo de sistema de gestión pueden hoy en día describir los resultados deseados, en lugar del procesamiento deseado, en un *lenguaje no procedimental.*

3.4.1 Clasificación

A través de la historia de la computación existen cientos de lenguajes de programación que han sido aplicados, en uno u otro momento, en un esfuerzo de desarrollo de software. Existen varias maneras de clasificar los lenguajes de programación, una de las más usadas es la clasificación por generaciones.

Generaciones

1. **Primera generación.** Son los lenguajes de programación directa sobre los computadores, los lenguajes máquina. Estos lenguajes dependientes de la máquina muestran el menor nivel de abstracción con el que se puede representar un programa. Estos, en sus inicios, eran introducidos en el computador mediante interruptores del tablero frontal del aparato.

   El código máquina y su equivalente más humanamente legible, que es el lenguaje ensamblador, representan la primera generación de lenguajes. Existen tantos lenguajes ensambladores como arquitecturas de procesadores con sus correspondientes conjuntos de instrucciones. Desde un punto de vista de la ingeniería del software, estos lenguajes sólo se deben usar cuando un lenguaje de alto nivel no satisfaga las necesidades.

   En esta generación todas las unidades de hardware eran construidos usando relés y tubos de vacío. Las computadoras resultantes eran extremadamente grandes de equipamiento, capaces de realizar miles de cálculos por segundo y costando millones de dólares.

2. **Segunda generación**. Desarrollados a finales de los 50's y principios de los 60's, han servido como base para los lenguajes de programación de la tercera generación. Esta generación se caracteriza por su amplio uso, la enorme cantidad de bibliotecas de software, la familiaridad y aceptación. Los lenguajes de base de esta generación son: *FORTRAN*, *COBOL*, *ALGOL*, *BASIC*.

Durante esta generación el hardware de propósito general se volvió más frecuente. Por otra parte, el software era hecho a la medida para cierta aplicación y su distribución era limitada. La mayoría del software era desarrollado y usado por la misma persona u organización. En esta generación se introducen los transistores, propiciando una reducción del tamaño y el costo del hardware.

3. **Tercera generación**. Los lenguajes de tercera generación están caracterizados por sus potentes posibilidades procedimentales y de estructuración de datos. Los lenguajes de esta clase se pueden dividir en tres amplias categorías, *lenguajes de alto nivel de propósito general, lenguajes de alto nivel orientados a los objetos* y *lenguajes especializados.*

   Durante esta era de la evolución de los sistemas de computación se introdujeron nuevos conceptos de interacción humano-máquina. Por lo que, se abrió un nuevo mundo de aplicaciones a nuevos niveles de sofisticación de hardware y software.

4. **Cuarta generación.** Los lenguajes de cuarta generación, al igual que los lenguajes de inteligencia artificial, contienen una sintaxis distinta para la representación del control y para la representación de las estructuras de datos. Sin embargo, un L4G representa estas estructuras en un mayor nivel de abstracción, eliminando la necesidad de especificar los detalles algorítmicos. Estos lenguajes combinan características procedimentales y no procedimentales.

   La gran mayoría de los lenguajes de cuarta generación se han desarrollado para ser usados conjuntamente con aplicaciones de bases de datos.

Otra forma de clasificar los lenguajes es de acuerdo al paradigma de programación, esto quiere decir, el estilo de programación mayormente usado, ya que, algunos lenguajes soportan varios paradigmas.

3.4.2 Estructurados

Incluyen lenguajes como COBOL, BASIC, PASCAL, y C, estos lenguajes reflejan el hecho de que las instrucciones disponibles permiten a los programadores concentrarse en los procedimientos que ellos están usando para resolver un problema, sin considerar el hardware donde se correrá el programa.

Típicamente, un grupo de declaraciones se combinan para crear un conjunto lógico de instrucciones llamado procedimiento, que es usado para producir un resultado específico. Un programa completo se puede componer de múltiples procedimientos que en conjunto cumplen con la programación del objetivo deseado. Regularmente, se crean siguiendo secuencialmente los pasos presentados en la *figura 3.6.*

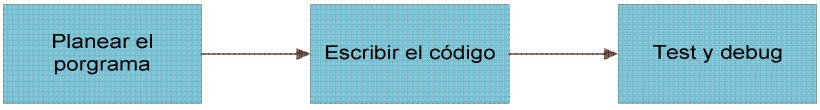


Figura 3.6 Pasos para construir programa estructurado

Un lenguaje estructurado se distingue por su uso de bloques. Un bloque es un conjunto

de sentencias que están relacionadas de forma lógica. Un lenguaje estructurado permite muchas posibilidades en programación. Así también, soporta el concepto de subrutinas con variables locales, donde una variable local es simple ante una variable que sólo es conocida dentro de la subrutina en la que está declarada.

Puede decirse que los lenguajes siguen un procedimiento, en el que las secuencias de control, conceptos de variables y operaciones de asignación son los componentes fundamentales.

3.4.3 Orientado a objetos

Algunos lenguajes orientados a objetos, como C++ y Object LISP son exténsiones de lenguajes convencionales que fueron diseñados básicamente como imperativos, por lo que, estas extensiones no consideran a los objetos como entidades activas que reciben mensajes, más bien invocan a procedimientos o funciones que pasan objetos. No obstante, incorporan las características más relevantes de la POO, en particular, la encapsulación de datos, el polimorfismo con enlace dinámico, la genericidad y la herencia.

El lenguaje paradigmático de este tipo de programación es SMALLTALK, que fue diseñado a principios de los años 70 con el fin de disponer de un potente sistema informático donde el usuario pudiera almacenar y manipular la información ágilmente.

No obstante, dada una relevancia que ha adquirido este tipo de programación, es usual que todos los lenguajes de propósito general y de una utilización específica, incorporen como lo han hecho los lenguajes C, PASCAL y LISP, los conceptos que definen la POO, quedando SMALLTALK como lenguaje puramente académico.

Actualmente, los lenguajes más utilizados en el desarrollo de software orientado a objetos son Java, C++ y C# (Alonso, Martínez y Segovia, 2012). Java (Arnold, 2001) ocupa, con el auge de la programación en Internet, un lugar destacado en este tipo de programación ya que está totalmente orientado a objetos, es multiplataforma e incorpora además el tratamiento de eventos. C++ (Stroustroup, 2000), precursor del anterior, es un estándar y su ambivalencia imperativa y OO ha permitido que tenga una gran difusión. Por último, C# (Liberty, 2003) es un lenguaje similar a Java definido por Microsoft que ésta teniendo cada vez más aceptación por sus características de fiabilidad, portabilidad y eficiencia.

3.4.4 Orientado a eventos

Los lenguajes orientados a eventos son los lenguajes que permiten al programador manipular objetos gráficos directamente, y subsecuentemente con el lenguaje de programación. Proveen el código necesario para soportar el objeto seleccionado y su interfaz. Visual Basic es un ejemplo de este tipo de lenguajes.

En estos programas lo más notable es la manera en que el usuario interactúa con el programa cuando se ejecuta. En este caso, el usuario tiene una variedad de opciones a elegir. El usuario puede elegir una de las opciones en cualquier secuencia. La elección de una opción disparara un evento, que es un procedimiento específico ligado a un objeto.

Un aspecto importante de estos lenguajes como Visual Basic es que proveen un set básico

de objetos que pueden ser colocados en un formulario durante el desarrollo de un programa. Esto se realiza dentro de un entorno de desarrollo integrado (IDE), que facilita la creación de interfaces gráficas. Como resultado, cuando se usa Visual Basic, el programador no se ocupa de la escritura del código para producir los objetos gráficos, o de reconocer cuando ocurren ciertos eventos, como “el clic del ratón”. Una vez que los objetos deseados son seleccionados y colocados en el formulario Visual Basic se hace cargo de crear el objeto y reconocer apropiadamente los eventos del objeto.

Usar este tipo de lenguajes deja al programador la responsabilidad de:

- En un inicio seleccionar y colocar objetos en un formulario donde el programa se desarrolla.
- Escribir e incluir código procedural a los eventos que pueden ser disparados por la interacción del usuario con los objetos del programa cuando se ejecuta.

3.4.5 Comparación

En la *tabla 3.2* se presenta una comparación de algunos lenguajes característicos de los tipos de lenguajes descritos en el subcapítulo anterior.

Tabla 3.2 Lenguajes de programación

	PASCAL	**C++**	**Visual Basic**	**Java**
¿Qué es?	•Lenguaje de programación estructural derivado de ALGOL-60	•Lenguaje de programación orientado a objetos creado por Bjarne Stroustrup.	•Lenguaje de programación dirigido por eventos, desarrollado por Alan Cooper para Microsoft.	•Es un lenguaje orientado a objetos, desarrollado por Sun Microsystems.
Características	•La gran expresividad debida a la particular estructuración de sus datos y de sus instrucciones evita la necesidad de recurrir a otros mecanismos. •Permite crear módulos que extienden el lenguaje.	•Potente en cuanto a lo que se refiere a creación de sistemas complejos ya que es un lenguaje robusto.	•Posee una curva de aprendizaje muy rápida. •Integra el diseño e implementación de formularios de Windows. •Permite usar con facilidad la plataforma de los sistemas Windows. •Es uno de los lenguajes de uso más extendido, por lo que resulta fácil encontrar información, documentación y fuentes para los proyectos. •Fácilmente extensible mediante librerías DLL y componentes ActiveX de otros lenguajes.	•Se pueden realizar distintos aplicativos, como son applets, que son aplicaciones, que se ejecutan dentro de un navegador al ser cargada una página HTML en un servidor WEB. •Puede desarrollar aplicaciones como software que se ejecutan en forma independiente.

Estructura básica	Programa nombre_programa;uses ……;
	declaraciones
	BEGIN
	•(* instrucciones *)
	•END.
		Directivas de preprocesador
		Declaraciones globales (varaibles globales, funciones, …)
		función main()
		{
		declaraciones e instrucciones
		}
		función1()
		{
		declaraciones e
			Declaraciones (localidades de memoria, funciones externas)
			Class NombreClase
			Sub NombreDeProcedimiento Sub()
			{Código de instrucciones}
			End Sub
			…
			Function NombreDeProcedimiento
				public class NombredeClase {
				public static void main(String[] args)
				{
				sentencia_1;
				sentencia_2;
				// ...
				sentencia_N;
				}
				}

		instrucciones
		}
		funciónN()
		{
		declaraciones e instrucciones
		}
			Function()
			{Código de instrucciones}
			End Function
			End Class	

Funciones	Las funciones tienen en Pascal la limitación de devolver únicamente valores pertenecientes a tipos simples.	Las funciones dividen un programa en subprogramas con finalidad concreta, los valores de retorno pueden ser valores numéricos, direcciones o estructuras, pero no matrices ni vectores.	Una función es un cálculo preprogramado que se puede realizar bajo petición desde cualquier lugar de un programa. Puesto que una función adopta uno o más argumentos y devuelve un único valor, esta se puede incluir en una expresión. La forma general es: Function NombreFunción([parámetros]) As TipoDeDato '{Instrucciones} NombreFunción = ValorDevuelto End Function	En Java una función es un módulo de un programa separado del cuerpo principal, que realiza una tarea específica y que puede regresar un valor a la parte principal del programa u otra función o procedimiento que la invoque. La forma general de una función es: tipodatoregresa Nom_fun(parametros) { cuerpo de instrucciones; intruccion return; }
Datos	INTEGER CHAR BOOLEAN Enumerativos Subcampo REAL ARRAY RECORD FILE SET Punteros	int (# enteros) float (# reales) double (# reales más grandes que float) bool (false, true) char (Caracteres y cualquier cantidad de 8 bits) void (Nada. Sirve para indicar que una función no devuelve valores)	Byte Integer Long Dec Single Double String Boolean:True y False Variant (cualquier tipo)	Int Short Long Byte Float Double Char Boolean(false, true)

3.4.6 Selección

Para el desarrollo de la aplicación se optó trabajar con Visual Basic (VB) por las siguientes razones:

- Como se describió en la sección 3.3.2 se optó por trabajar con el paradigma orientado a eventos, ya que, se planeó una aplicación donde el usuario y sus acciones tuvieran el control sobre el programa. Visual Basic es uno de los lenguajes más representativos de dicho paradigma.

- El IDE de VB es fácil de utilizar y cuenta con herramientas destinadas para ayudar a escribir y modificar el código del programa, así como, a detectar y corregir errores en el programa.

- Para la elaboración de los algoritmos y del software se consideró que se recibiría ayuda de profesionales expertos para la programar las expresiones matemáticas de la Geoestadística, con conocimientos de VB pero no de otros lenguajes, por tanto, para agilizar el desarrollo del programa se decidió proceder con VB.

- Su sintaxis, derivada del antiguo BASIC que ha sido ampliada con el tiempo, posee características típicas de los lenguajes estructurados modernos, haciendo posible usar en sus rutinas las estructuras básicas de la programación estructurada.

- VB es un lenguaje tanto interpretado como compilado, ya que, se puede probar el código y eliminar errores de manera inmediata. Posteriormente, se pueden compilar para que sea ejecutable y rápido.

- La principal desventaja que se hubiera presentado al usar VB es que es un lenguaje desarrollado para plataforma Windows, pero en el laboratorio donde se implementará el software solo trabaja bajo esta plataforma, por ende, esto dejo de representar un problema para proceder con VB.

- Para resolver el problema de las matrices de correlación en el análisis geostadístico, se encontró una biblioteca compatible con VB llamada MaPack.dll (Geosoft Inc, 2010) que incluye las funciones necesarias para trabajar con matrices que se requieren en el análisis, ahorrando así tiempo y código al trabajar con las matrices necesarias. Mapack.dll pertenece al módulo Dapple desarrollado por Geosoft Inc. El módulo Dapple es un explorador global de información diseñado para proveer un ambiente óptimo y abierto para visualizar, presentar y compartir cantidades masivas de información geocientífica en computadoras de escritorio. El proyecto Dapple es un proyecto de código abierto derivado del proyecto de código abierto NASA World Wind.

3.5 Almacenamiento

Bajo el concepto de almacenamiento se agrupa a los dispositivos y software dedicados al archivo o preservación de datos e información.

3.5.1 Archivos

Los datos son la materia prima de la que se deriva la información. La información está compuesta por datos que se han recopilado y procesado de manera significativa. Un conjunto de bits se combinan para formar un carácter; los caracteres se unen para representar los valores de los elementos dato, también llamados campos; los elementos dato relacionados entre sí se agrupan para formar registros; los registros que contienen los mismos elementos dato se combinan para formar un archivo (Long, L. 1995).

Un registro es un conjunto lógicamente conectado de uno o más campos que describe una persona, lugar o cosa. Mientras un campo es un carácter o grupo de caracteres que tienen un significado específico. Los campos son utilizados para definir y guardar datos.

Un archivo está compuesto de registros homogéneos que contienen información sobre un tema específico. Tratado como una unidad de almacenamiento y organizado de forma

estructurada para la búsqueda de un dato individual. Es un conjunto de datos que tienen entre sí algo de común. Es decir, que los datos poseen un criterio de pertenencia que les permite ser identificados como elementos o miembros del conjunto que constituye al archivo, cada archivo tiene un nombre asociado.

Los archivos se utilizan desde 1960s como un mecanismo para asegurar la permanencia de los datos y cuando el volumen de los datos es bastante significativo para almacenarse en la memoria principal de la computadora, teniendo que almacenarse estos en el disco duro o en algún otro medio de almacenamiento secundario.

Los archivos se clasifican, de acuerdo al contenido en: archivos texto plano, archivos binarios, archivos de datos, archivos de control, archivos de transacción, archivos de registro, archivos de imagen, archivos de sonido, archivos de video, etc.

3.5.1.1 Características

Un archivo posee un nombre y una extensión, con las cuales es posible identificarlo. Poseen una serie de características entre las que se encuentran:

- **Volatilidad:** Se refiere a la frecuencia con que se realizan adiciones y eliminaciones en un archivo. Cuando esta frecuencia e baja, el archivo se llama *archivo estático*, cuando la frecuencia es alta, el archivo se llama *archivo volátil*.

- **Actividad:** Se refiere al porcentaje de registros de un archivo que son procesados en una ejecución dada.

- **Tamaño:** Se refiere a la cantidad de información almacenada en el archivo.

- **Personales:** Los archivos son personales, esto quiere decir que no son compartidos entre usuarios.

- **Independencia:** Los archivos son independientes de los programas, y es probable que un mismo archivo creado por un programa puede ser usado por otros.

- **Almacenamiento:** Gran capacidad de almacenamiento.

3.5.1.2 Ventajas

Algunas ventajas del uso de archivos son:

- **Almacenamiento sencillo**: Ya que solo es una colección lógica de información, a la que dé inicio solo se le asocia un nombre.

- **Costo:** Tienen un menor costo ya que no necesitan personal capacitado ni hardware adicional.

- **Accesibilidad:** Todos los usuarios podrían leer y modificar el contenido del archivo, aunque esto significa una desventaja también.

- **Ahorro en recursos:** No necesitan de un motor de almacenamiento, ni un manejador específico ni un lenguaje de consulta específico.

3.5.1.3 Desventajas

Algunas desventajas del uso de archivos son:

- **Acceso:** Resulta complicado el compartir la información, ya que, hace mayor uso de recursos y servicios de cómputo. Además permitir el acceso desde distintas aplicaciones frecuentemente necesita ser programada.

- **Seguridad:** Los sistemas de archivos no ofrecen la seguridad de los datos, e implementar mecanismos de seguridad necesita de mayor programación y consumo de recursos de cómputo.

- **Concurrencia:** Los archivos no permiten acceder a múltiples usuarios a la vez.

- **Redundancia**: Los archivos permiten repetición de datos. La característica de que cada programa defina y maneje sus propios datos da oportunidad a que se presente su principal debilidad, la redundancia de datos, y por consecuencia, la posible fuente de inconsistencias. El hecho de que haya duplicidad también requiere de almacenamiento adicional.

- **Mantenimiento:** La realización de actualizaciones puede resultar costosa cuando se tiene información parcialmente duplicada en varios archivos.

- **Inconsistencia**: Fácilmente los datos pueden quedar en un estado de inconsistencia, esto quiere decir, que existirán diferentes valores para un mismo objeto.

- **Diccionario de datos**: No provee de diccionarios de datos.

3.5.2 Bases de datos

Una base datos es un conjunto de datos organizados de modo tal que resulte fácil acceder a ellos, gestionarlos y actualizarlos. Es una colección de datos interrelacionados y almacenados permanentemente en una computadora tal que:

a. Los datos son compartidos por diferentes usuarios y programas de aplicación, pero existe un mecanismo común para la inserción actualización, borrado y consulta de los datos.
b. Tanto los usuarios finales como los programas de aplicación no necesitan conocer los detalles de las estructuras de almacenamiento.

3.5.2.1 Características

Las bases de datos poseen una serie de características entre las que se encuentran:

- **Seguridad:** Los datos deben permanecer resguardados y seguros, no susceptibles a accesos o modificaciones no autorizados.

- **Independencia lógica y física**: Se refiere a la capacidad de modificar una definición de esquema en un nivel de la arquitectura sin que esta modificación afecte al nivel inmediatamente superior. Para ello un registro externo en un esquema externo no tiene por qué ser igual a su registro correspondiente en el esquema conceptual

- **Concurrencia:** Múltiples usuarios acceden a la base de datos al mismo tiempo.

- **Integridad:** Esto quiere decir que todos los datos deben ser correctos. Normalmente se expresa mediante restricciones o reglas que no se pueden violar.

- **Lenguajes de consulta:** estos lenguajes facilitan al usuario el acceso a los datos.

- **Fiabilidad:** Existe una protección contra fallos.

3.5.2.2 Ventajas

Algunas ventajas del uso de bases de datos son:

- **Minimiza redundancia:** Disminuye el margen de error e inconsistencia. Si no se elimina completamente, la redundancia se controla y minimiza, también la duplicación de datos es controlada.

- **Consistencia:** Ya que se controla la redundancia de datos se reduce el riesgo de que exista inconsistencia. Si un dato está almacenado una sola vez, cualquier actualización se debe realizar sólo una vez, y está disponible para todos los usuarios inmediatamente.

- **Independencia:** Entre datos y hardware, entre datos y programas de aplicación. Adicionalmente a estas independencias también está el hecho de que usuarios finales no necesitan preocupar por la localidad donde se encuentra almacenada la información.

- **Concurrencia:** Múltiples usuarios acceden a la base de datos al mismo tiempo.

- **Mantenimiento:** Reduce gastos de mantenimiento gracias a la independencia de datos. Esto simplifica el mantenimiento de las aplicaciones que acceden a la base de datos.

- **Seguridad:** Mecanismos de seguridad superiores. Permisos y derechos de acceso pueden ser definidos para un usuario o grupo de usuarios fácil y eficientemente, así un usuario desautorizado podrá actualizar o leer información sensible.

- **Respuesta rápida:** En caso de los archivos para extraer información directamente se necesita programar una aplicación, mientras que en las bases de datos se pueden extraer directamente.

3.5.2.3 Desventajas

Algunas desventajas del uso de base de datos son:

- **Complejidad:** Pueden llegar a ser complejas con una gran funcionalidad. Así que es preciso comprender esta funcionalidad para poder realizar un buen uso de ellos. Asimismo, desde el punto de vista de infraestructura se requiere más equipo de cómputo en caso de requerir la base.

- **Personal capacitado:** Las bases de datos requieren un manejador que garantice seguridad, consistencia, concurrencia e integridad. Si no se cuenta con el personal especializado se debe contratar nuevo personal o capacitar al existente.

- **Costo:** Tienen un mayor costo, ya que para la implementación se requiere de software especializado. Además que es poco rentable a corto plazo. El costo también implica, el gasto en nuevo equipo de cómputo si se requiere.

- **Implantación:** La implantación de una base de datos regularmente es larga y difícil. Requiere de un consumo de mayores recursos, estructura dedicada, que demanda mayores recursos para una aplicación de escritorio.

- **Hardware:** Para alcanzar las prestaciones necesarias muy probablemente se necesite hardware adicional, así como más memoria tanto primaria como secundaria.

- **Vulnerabilidad a fallos:** En caso dado de que el sistema sea centralizado el sistema es más vulnerable ante fallos que puedan producirse.

3.6 Algoritmos

Un algoritmo es una sucesión finita ordenada y sin ambigüedades de instrucciones o pasos que llevan a resolver un problema (Baldwin y Scragg, 2003), esta demostración se ilustra en la *figura 3.7*. Un algoritmo es una secuencia de instrucciones que describe como la información debe ser procesada para producir las salidas deseadas. La palabra *"algoritmo"* proviene etimológicamente del árabe *"al-juarizmi"* debido al lugar de procedencia del matemático Mahomed ben Musa, quien vivió en el siglo IX en la zona que ahora es Uzbekistán. Se puede decir que la acuñación de la palabra "algoritmo" es tardía, pues los algoritmos han existido desde mucho tiempo atrás.

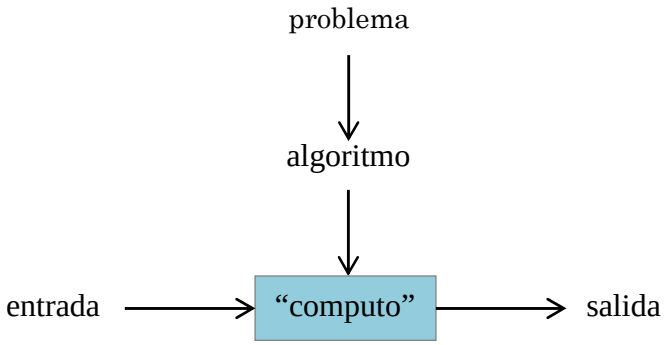


Figura 3.7 Noción de algoritmo

Para diseñar algoritmos es necesario disponer de una notación llamada ‘*notación algorítmica*’, que permita:

- Describir las operaciones puestas en juego (acciones).
- Describir los objetos manipulados por el algoritmo (datos/informaciones)
- Controlar la realización de las acciones descritas, indicando la forma en que estas se organizan en el tiempo.

3.6.1 Estructura

La estructura de un algoritmo se encuentra dada de la siguiente forma:

- Cabecera
- Inicio
- Bloque de Declaraciones
- Bloque de Instrucciones
- Fin

Para describir cualquier tipo de acción dentro del bloque de instrucciones de las que intervienen en un algoritmo, se han propuesto el uso de un conjunto de construcciones lógicas (secuencia, decisión e iteración) con las que es posible escribir cualquier programa.

Acción Elemental

Se entienden por *acciones elementales* aquellas que el ordenador es capaz de realizar y que serán de dos tipos:

- *Aritmético – lógicas*: Operaciones que, a partir de unos determinados datos, realizan un cálculo aritmético (suma, resta, multiplicación,…) o un cálculo lógico (mayor que, menor que, igual que,…). Las primeras devuelven un valor numérico y las segundas un valor lógico.
- *De entrada – salida:* Acciones que permiten capturar datos para su posterior tratamiento y guardar los resultados de dicho tratamiento.

Secuencia de acciones elementales (composición secuencial)

Cuando en un algoritmo se deben ejecutar varias acciones sucesivamente, estás se describen una detrás de otra según el orden en que deban ejecutarse. Las acciones se

suceden de tal modo que tras la salida (final) de una se efectúa la entrada (principio) en la siguiente y así sucesivamente hasta el fin del proceso.

Ejecución condicional de una acción (if- else)

Cuando en un algoritmo se quiere indicar que cierta acción sólo se debe ejecutar bajo cierta condición se indica como:

```
If      <condición>     Then
        hacer <declaración1>
Else
        hacer <declaración2>
End If
```

Solo si la condición es verdadera se ejecutará la declaración. En este caso, la declaración puede referirse tanto a una acción elemental como a un conjunto de ellas.

Otra caso es el siguiente:

```
If      <condición1>    Then
        hacer <declaración1>
ElseIf  <condición2>    Then
        hacer <declaración2>
End If
```

Todas las condiciones son evaluadas secuencialmente. Si es verdad, la declaración correspondiente es ejecutada.

Ejecución condicional de una de varias opciones (Case)

Cuando a la hora de especificar la ejecución de una acción haya que escoger ésta entre varias dependiendo del valor de una determinada variable. Este caso se e expresa como:

```
Select Case indicador
        Case indicador1
                hacer <declaración1>
        Case inidicador2
                hacer <declaración2>
        …
        Case indicadorn
                hacer <declaraciónn>
End Select
```

Ejecución múltiple de una acción (Do While)

El computador está especialmente diseñado para aplicaciones en las que una operación o un conjunto de ellas deben repetirse. Cuando una acción o conjunto de acciones debe ejecutarse varias veces se recurre a una estructura iterativa o bucle. En estas estructuras se necesita una condición que determine cuando terminan las iteraciones, ya que si éstos no existen, el bucle puede convertirse en un proceso infinito.

```
Do While        <condición>
                hacer <declaración>
```

Loop

En este caso se realiza el bucle y la declaración es ejecutada mientras la condición sea verdad.

De acuerdo con lo anterior, una estructura iterativa debe constar de:

- Condición para repetición de bucle
- Cuerpo de bucle (instrucciones que se repetirán)
- Salida del bucle

Ejecución de ciclo (For)

Este bucle cuenta con un contador que se irá incrementando en uno cada vez que el bucle se repita. El bucle es ejecutado hasta que el valor del contador es igual a aún valor final que también ha sido asignado.

For <contador> **To** <fin>
hacer <declaración>
Next

El contador se incrementa en uno por default, más el usuario puede incrementarlo especificándolo con **Step<**no.>. Ejemplo:

For N=1 **to** **Step** 2
hacer <declaración>
Next

Ejecución Con-Fin (With-End)

Ejecuta todas las declaraciones entre el **With-End** con un objeto particular.

With <nombre objeto>
hacer <declaración>
End With

3.6.2 Características

Para encontrar solución a un problema planteado existe más de un proceso, pero no todos los procesos son algoritmos, para ser un algoritmo, un proceso debe tener las siguientes características:

- **No ambiguo:** En otras palabras, tiene que ser capaz de describir cada paso del proceso en suficiente detalle como para que cualquiera (hasta una máquina) pueda ejecutar el algoritmo en la forma que el diseñador previo. Esto requiere no solamente saber exactamente como debe desarrollarse cada paso, sino también el orden exacto en que los pasos se presentaran.

- **Debe resolver el problema:** En otras palabras, una persona (o máquina) que ejecute el algoritmo para resolver una instancia del problema debe ser capaz de llegar al final

con la respuesta correcta después de ejecutar un número finito de pasos. El usuario debe llegar al resultado correcto sin importar en que instancia del problema se inicie.

3.6.3 Ventajas

Los algoritmos poseen una serie de ventajas entre las que se encuentran:

- Un desarrollo del procedimiento mismo, involucra identificación de procesos, puntos de decisión e identificación de las variables necesarias para resolver el problema. La identificación de los procesos y puntos de decisión reduce las tareas en pasos más pequeños y fáciles de manejar.

- Al desarrollar el algoritmo permite y obliga examinar la solución de forma racional, disminuyendo el riesgo de errores.

- Al usar algoritmos, la toma de decisiones se vuelve un proceso más racional, eficiente y consistente.

- Funciona como un dispositivo nemotécnico, y ayuda a asegurar que no se ignoren variables o partes del problema.

- Ayuda a resolver más fácil y rápido los problemas ya que gracias a la separación del procedimiento en pasos hay una división de trabajo.

- A medida que pasa el tiempo y los resultados se comparan con los objetivos, la existencia de un proceso de solución especificada (algoritmo) permite la identificación de las debilidades y los errores en el proceso.

3.7 Diagramas de flujo

Un diagrama de flujo es la representación gráfica de un algoritmo que muestra los pasos que deben seguirse para alcanzar la solución de un problema. Los distintos tipos de paso se dibujan de manera diferente, permitiendo una rápida interpretación de su significado. Al escribir en código un algoritmo, expresado mediante un diagrama de flujo, la traducción a cualquier lenguaje de programación se puede lograr fácilmente.

3.7.1 Estructura

Los diagramas de flujo se construyen utilizando ciertos símbolos de uso especial, tales como rectángulos, óvalos, círculos y rombos, los cuales se conectan entre sí por flechas, mismas que son conocidas como líneas de flujo. Cada símbolo representa una acción específica y las flechas el orden de realización de las acciones. Por tanto, cada símbolo tendrá al menos una flecha que conduzca a él y una flecha que parta de él, exceptuando el comienzo y el final del diagrama.

El diagrama de flujo aporta una definición más clara del problema en estudio, puesto que, describe una solución por medio de una expresión lógica. Lo anterior es importante, ya que, muchas veces se tienen dificultades para asignar códigos que guíen el trabajo de la computadora, debido a que frecuentemente existen ambigüedades entre el planteamiento

mismo del problema y las instrucciones que recibe el programador para elaborar el código.

Los elementos gráficos empleados para elaborar diagramas de flujo se muestran en la *figura. 3.8*:

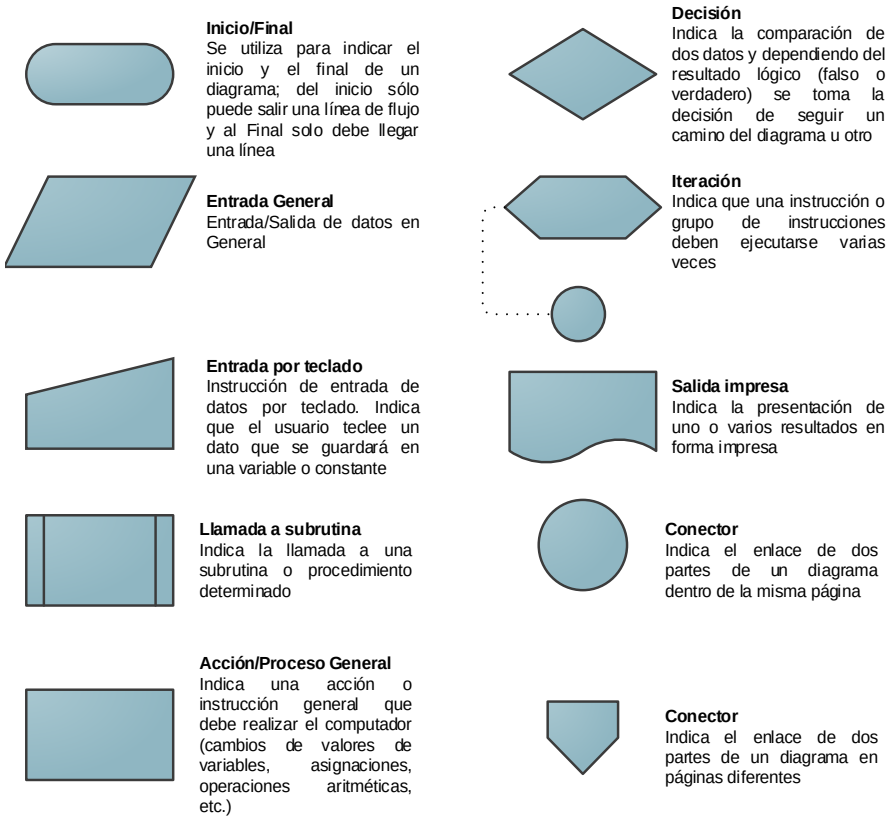


Figura 3.8 Elementos de los diagramas de flujo.

La forma de representación de las construcciones de los algoritmos o "estructuras de control" explicadas en el apartado 3.6.1 se muestran en las *figuras 3.9* a *3.13*.

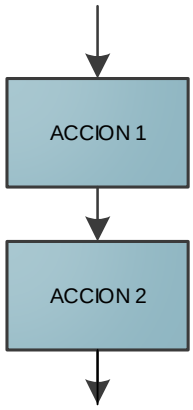


Figura 3.9 Esquema de estructura secuencial.

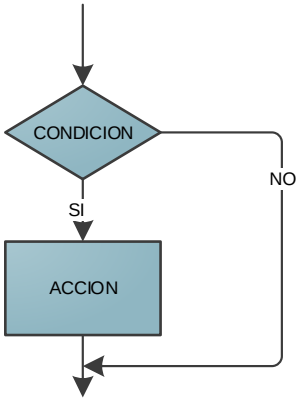


Figura 3.10 Esquema de estructura condicional de una acción.

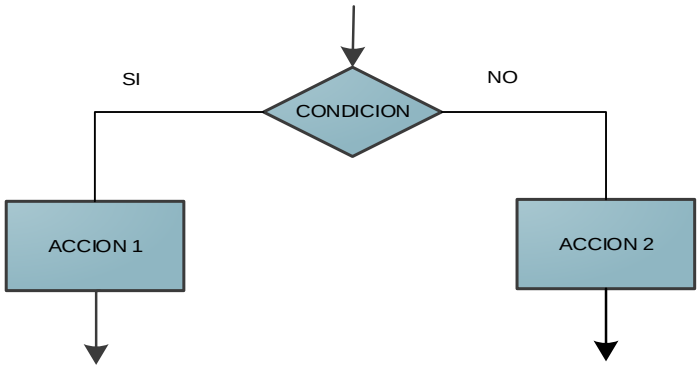


Figura 3.11 Esquema de estructura condicional de dos acciones.

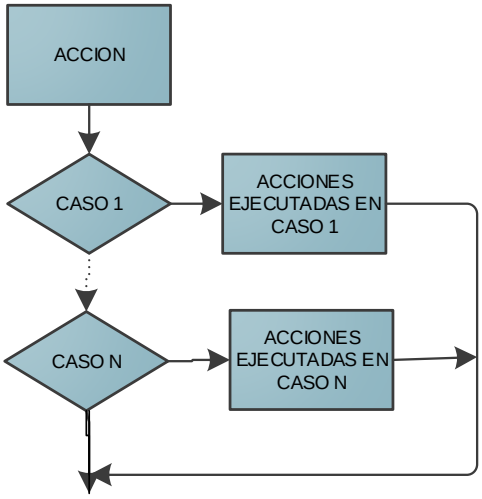


Figura 3.12 Esquema de estructura condicional de dos acciones.

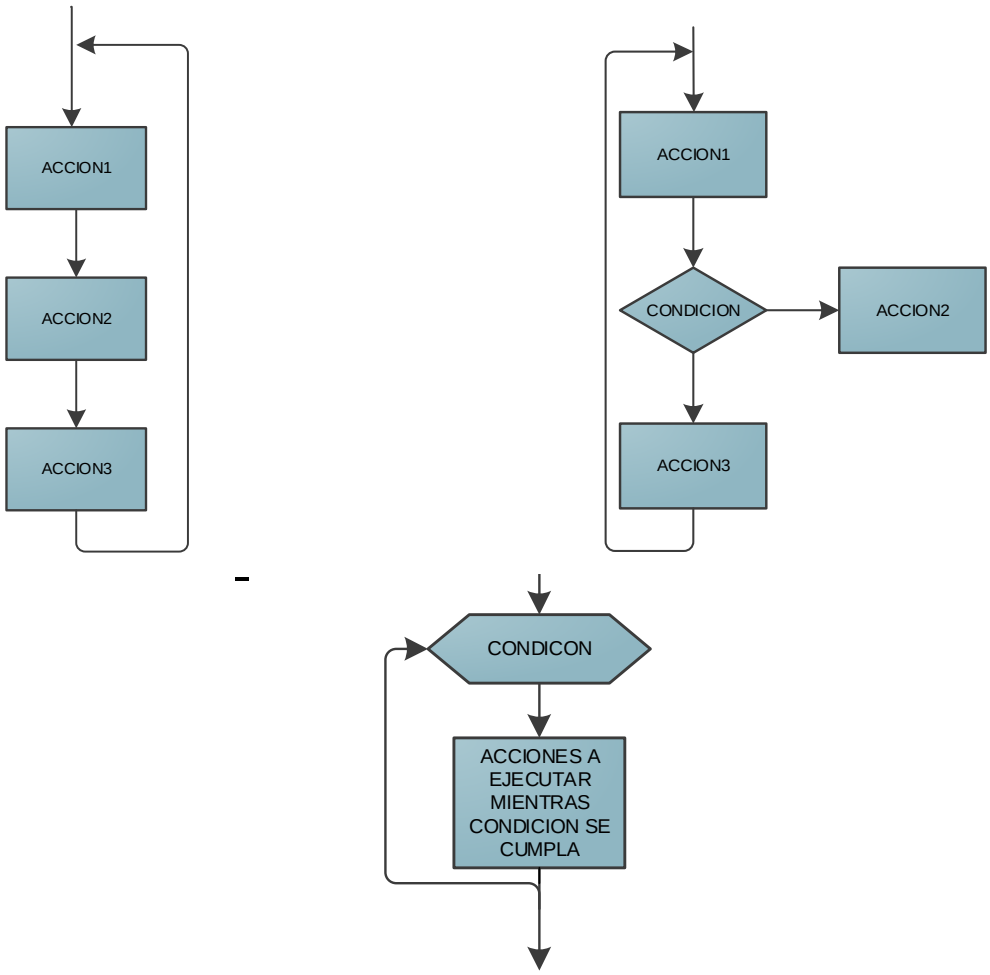


Figura 3.13 Esquemas de estructura condicional de múltiples acciones.

3.7.2 Características

Los diagramas de flujo poseen una serie de características entre las que se encuentran:

- **Capacidad de Comunicación:** Permite la puesta en común del flujo de control de un programa.

- **Claridad:** Proporciona información sobre el flujo de control de un programa de forma clara, ordenada y concisa; ya que, usa símbolos para mostrar las operaciones y decisiones a seguir por una computadora para resolver un problema. Además, las instrucciones deben ser escritas usando declaraciones claras.

- **Inicio – Fin:** Los diagramas de flujo siempre tienen un único punto de inicio y un único punto de término. Además, todo camino de ejecución debe permitir llegar desde el inicio hasta el término. Indicando así al programa el inicio y final de una tarea dada.

- **Representación:** Representa el flujo de la información entre las operaciones y decisiones conectadas por líneas solidas con flechas, indicando el flujo de operación en una secuencia que debe seguir el programa.

- **Descomposición:** Permiten la descomposición en submodelos, estos submodelos son fragmentos del flujo de control de un programa.

3.7.3 Ventajas

Por otra parte, los diagramas de flujo poseen una serie de ventajas entre las que se encuentran:

- **Síntesis**: Los diagramas de flujo son usados como modelos de trabajo en el diseño de nuevos programas y sistemas de software.

- **Documentación**: La documentación de un programa consiste en actividades, como recolectar, organizar, almacenar, y dar mantenimiento a todos los archivos del programa. Parte de esta documentación son los diagramas de flujo.

- **Codificación**: Los diagramas de flujo guían al programador para escribir el código en un lenguaje de alto nivel, lo que presupone debe generar un programa libre de errores.

- **Depuración**: Los errores de un programa son detectados después de su ejecución en una computadora, estos errores se conocen como *bugs* y el proceso de removerlos se llama depuración. En el proceso de depuración, un diagrama de flujo actúa como una herramienta importante para detectar, localizar, y remover *bugs* de un programa.

- **Comunicación**: Un diagrama de flujo es una representación pictórica de un programa. Por tanto, puesto que se utilizan símbolos universales es un excelente medio de comunicación para explicar la lógica de un programa.

- **Análisis**: Con ayuda de un diagrama de flujo se puede lograr fácilmente un análisis efectivo de un problema lógico.

- **Testeo**: Un diagrama de flujo es una herramienta importante en manos de un programador, este le ayudara en el diseño de las pruebas sistemáticas de los programas.

3.8 Graficación

El concepto de graficación se introdujo en los años 70, donde se designaba únicamente los gráficos que tenían la intención de representar en forma de imagen información generalmente estática por medio de diagramas, mapas o esquemas. Posteriormente, el concepto de gráfico por computadora se extendió rápidamente a todos los gráficos producidos por medios informáticos.

El campo de la graficación computarizada abarca todos los aspectos relacionados con el

uso del computador para generar imágenes. Además, realiza el estudio, diseño y trabajo del despliegue de imágenes en dos y tres dimensiones en la pantalla de un computador, a través de herramientas proporcionadas por la matemática, la física, etc.

En el caso de la Geotecnia, una de las aplicaciones de la graficación es la descripción estadística de las propiedades del suelo por medio de las técnicas de gráficación, siendo los más usuales el histograma, el polígono de frecuencias y la distribución de frecuencia relativa.

Así también, se usan gráficas de dispersión para tener una referencia visual sobre la magnitud de variación de los datos (sondeos y propiedades). El polígono de frecuencias y el histograma se usan para conocer la variación del conjunto de datos; muestra rango de datos, los valores de los datos que ocurren con mayor frecuencia y el grado de dispersión alrededor de los valores medios del conjunto de datos. También pueden indicarse los valores de la media, mediana y moda.

4 Análisis de la información

4.1 Planificación y consideración del diseño

El propósito de este capítulo es describir los elementos involucrados para el desarrollo de un software que ayude al usuario a realizar paso a paso un análisis geoestadístico, teniendo al final un conjunto de archivos de resultados en cada etapa del análisis.

En el capítulo se pretende dar a conocer cada aspecto sobre el análisis y planificación del proyecto de software, así como las consideraciones empleadas durante el proceso de diseño.

Primeramente, se hace el reconocimiento del problema, se explica la necesidad de crear un software para uso exclusivo en Geotecnia, así como, los requerimientos del usuario. Posteriormente, se describe brevemente la infraestructura disponible para el desarrollo de software. Después, se explica la metodología para el diseño y desarrollo del software, así como, su diagrama de actividades. Finalmente, se explican los criterios de la elección de plataforma y lenguaje de desarrollo.

4.2 Definición del problema

4.2.1 Importancia social

Una de las actividades de la Ingeniería Geotécnica es el diseño y construcción de cimentaciones de las obras civiles, partiendo siempre del conocimiento adecuado de las características y propiedades naturales del subsuelo. Los datos que permiten llegar a un conocimiento adecuado del subsuelo provienen de campañas de exploraciones geotécnicas de campo (sondeos) y ensayes de laboratorio. En la práctica, las técnicas convencionales empleadas en la Geotecnia para la presentación de los datos obtenidos en las campañas de exploración geotécnica consisten en tablas, perfiles geotécnicos y secciones transversales estratigráficas elaboradas en forma subjetiva o artística.

Por otra parte, la información que se obtiene como resultado del reconocimiento del subsuelo es frecuentemente insuficiente y no da una idea precisa sobre la variación espacial de las condiciones estratigráficas y de sus propiedades. Estas deficiencias son frecuentemente producto de las limitaciones económicas y prácticas que se encuentran al realizar la exploración y caracterización del subsuelo.

Aun cuando existen técnicas de interpolación, éstas prácticamente no son utilizadas para estos fines. Actualmente, se cuenta con herramientas estadísticas con gran potencial pero que hasta ahora en Geotecnia prácticamente no se han usado. Una de estas herramientas es la Geoestadística, nombre bajo el cual se entiende la aplicación de la teoría de las funciones aleatorias (en este caso espaciales) y del tratamiento de las señales a la descripción de las condiciones estratigráficas y a la distribución espacial de las propiedades de los materiales geológicos. La Geoestadística permite tomar en cuenta la dependencia espacial entre propiedades en puntos cercanos a través del concepto de “función de autocovarianza” o de “variograma.” Con estas herramientas es posible resolver en forma racional problemas tales como la estimación de espesores de estratos típicos del subsuelo, o de valores de las propiedades del subsuelo en un sitio particular o

en una zona determinada, a partir de la información de los sondeos existentes con base en técnicas de estimación optimizadas como el “Kriging” (obtención de estimadores lineales de mínima varianza).

Sin embargo, para lograr que la Geoestadística sea empleada en forma sistemática en la Geotecnia es necesario contar con una herramienta de ayuda que facilite su aplicación práctica para tener un mejor aprovechamiento de la información obtenida sin tener que adoptar hipótesis que llevan a una gran incertidumbre en los resultados arrojados por los modelos matemáticos con los que se pretende predecir el comportamiento de las obras geotécnicas. Contar con una herramienta que facilite su aplicación práctica ayudaría a reducir costos ya que se evitarían medidas de seguridad exageradas a causa de la incertidumbre asociada a la caracterización del subsuelo.

Actualmente, existen varios programas de computadora para la aplicación de la Geoestadística, mencionados con anterioridad en la *tabla 2.1* del capítulo 2. Los programas existentes son útiles para ciertas aplicaciones específicas pero ninguno de ellos ha sido diseñado específicamente para su aplicación en Geotecnia.

4.2.2 Requerimientos técnicos

Los requerimientos técnicos que debió cubrir el programa desarrollado son:

- Una interfaz sencilla que lleve de la mano al usuario durante todo el proceso del análisis para disminuir el tiempo de realización del análisis y no cause confusiones en el orden de las etapas a seguir del análisis.

- Tener la capacidad de realizar análisis geoestadísticos para una, dos y tres dimensiones de la forma más eficiente posible, ya que estos son los tipos de análisis geoestadísticos de mayor utilidad en Geotecnia.

- Los parámetros de la estadística descriptiva calculados se deben mostrar en la ventana del programa por cada análisis que se realice, esto con el fin cubrir una primera etapa del análisis, ya que, el software permite ordenar y procesar los datos disponibles para una rápida interpretación y cálculo de los principales parámetros estadísticos.

- El análisis de tendencia debe ser realizado por medio del método de regresión lineal, ya que, es el método más sencillo que se ha encontrado y con buenos resultados.

- El proceso de estimación debe ser realizado con el método de Kriging Ordinario, puesto que, es el método que mejor se adapta a los datos que se analizan.

- La creación de archivos debe ser posible desde la interfaz del programa. El usuario debe tener la posibilidad de introducir los valores representantes a las variables por medio del teclado; para tal tarea el programa debe contar con la forma de presentarle al usuario una tabla con columnas para introducir los datos y guardarlo posteriormente.

- El programa debe adquirir datos desde un archivo, este requisito tiene la finalidad de consultar los datos previamente guardados y con la posibilidad de poder cambiar algún valor (edición) de los datos.

- Los archivos de resultados deben crearse en cada fase del análisis, con la intención de tener un registro completo de los resultados durante cada fase del análisis.

- Los resultados se almacenan en columnas, teniendo cada columna un encabezado y su nombre debe estar relacionado con los valores asociados a cada columna.

- La generación de archivos de interpolación durante el análisis en todos los casos debe ser automático.

- Se requiere que el sistema trabaje en sistema operativo Windows XP o Windows 7, ya que, son las plataformas que se utilizan en el Laboratorio de Geoinformática del Instituto de Ingeniería, UNAM.

- Graficación de los resultados mediante perfiles (1D), mapas de contornos (2D) y secciones geotécnicas de propiedades del subsuelo (3D), empleando escala de colores en todos los casos.

4.3 Infraestructura necesaria

4.3.1 Ambiente de desarrollo

El ambiente de desarrollo contiene el conjunto más amplio de tareas, que abarca desde el comienzo del ciclo de vida del software hasta la obtención de una versión mínimamente estable de la aplicación, o de un subconjunto de la misma (módulos).

En este ambiente se realizan las siguientes tareas:

Toma de requerimientos: Se concretan las necesidades del cliente durante varias reuniones.

Análisis de arquitectura y diseño técnico: De acuerdo a los requerimientos especificados se realiza la elección de la tecnología que abarque las necesidades del proyecto con una adecuada arquitectura y diseño de los componentes, la localización de las fases más críticas. Con lo anterior, se aportan soluciones de contención ante posibles problemas que pudieran presentarse.

Implementación: Indica el comienzo del desarrollo con la tecnología adecuada, elegida en la fase de análisis.

Pruebas de unidad y módulos: Estas pruebas se deben realizar sobre los módulos y componentes que se vayan finalizando. Se realizan en un entorno local para comprobar su correcto funcionamiento y poder integrarlas para obtener el producto final.

En el ambiente de desarrollo de esta aplicación, los integrantes del equipo de desarrollo emplearon cada uno con una computadora personal en las que se instalaron todas las

herramientas requeridas a fin de que cada integrante fuera autosuficiente, la configuración de las computadoras se presentan en la *tabla 4.1*.

Tabla 4.1 Configuración de las computadoras de desarrollo

Sistema Operativo	Windows 7 Professional o Enterprise
Tipo de sistema	Sistema operativo de 64 bits
IDE (entorno de desarrollo)	Visual Studio 2010 Ultimate
Procesador	Intel Xeon E5645 2.40GHz 2.39GHz (2 procesadores)
Memoria RAM mínima	4 GB
Disco	700 GB

4.3.2 Ambiente de producción

Las pruebas finales de la aplicación son realizadas por los desarrolladores antes del paso final del ambiente de producción, donde se pone en funcionamiento bajo un escenario real. En esta etapa se tiene en todo momento la versión activa de la aplicación y los usuarios finales tienen acceso a la aplicación implementada.

Cuando se llega a este ambiente, el producto ya se encuentra trabajando con datos reales, obtenidos de estudios previamente realizados. Así mismo, en este ambiente se usan computadoras que trabajan con diferente sistema operativo, en este caso Windows XP, así como, con diferentes capacidades de memoria.

4.4 Desarrollo de metodología

Como se expuso en el apartado 2.3, el análisis geoestadístico se divide en varias etapas, por lo que, se debe elegir una metodología que tenga la posibilidad de entregas en cada una de las etapas para tener revisión de los resultados obtenidos en cada etapa del análisis.

Para el desarrollo de la aplicación se eligió una **Metodología de Entrega por Etapas**, la cual no entrega el producto total al final del proyecto, sino que, muestra al usuario etapas funcionales sucesivamente. Primero, se realiza la definición de concepto del software, el análisis de requerimientos y la creación del diseño global de la arquitectura del modelo en cascada. Posteriormente, se procede a realizar el diseño detallado, la codificación, depuración y prueba dentro de cada etapa.

El modelo de entrega por etapas es útil para el desarrollo de aplicaciones debido a que su uso se recomienda para problemas que pueden ser tratados descomponiéndolos en problemas más reducidos.

Con esta metodología se reduce el tiempo necesario para la construcción de un producto; entre sus beneficios se tienen:

- Detección continúa de problemas por fase y no hasta la entrega final del proyecto.
- Eliminación del tiempo en informes debido a que cada versión es un avance.
- Estimación de tiempo por entrega.

Usando esta metodología se produce un intercambio de conocimiento, corrección y crítica, debido a que en cada entrega se realizan pruebas y en caso de que se presenten problemas deben corregirse en entregas posteriores, lo que hace que el proyecto sea mejorado en

cada etapa. Permite la entrega de versiones cada vez más completas que llevan a lograr un producto final conforme a los requisitos y especificaciones planteados. La metodología se ilustra en la *figura 4.1*.

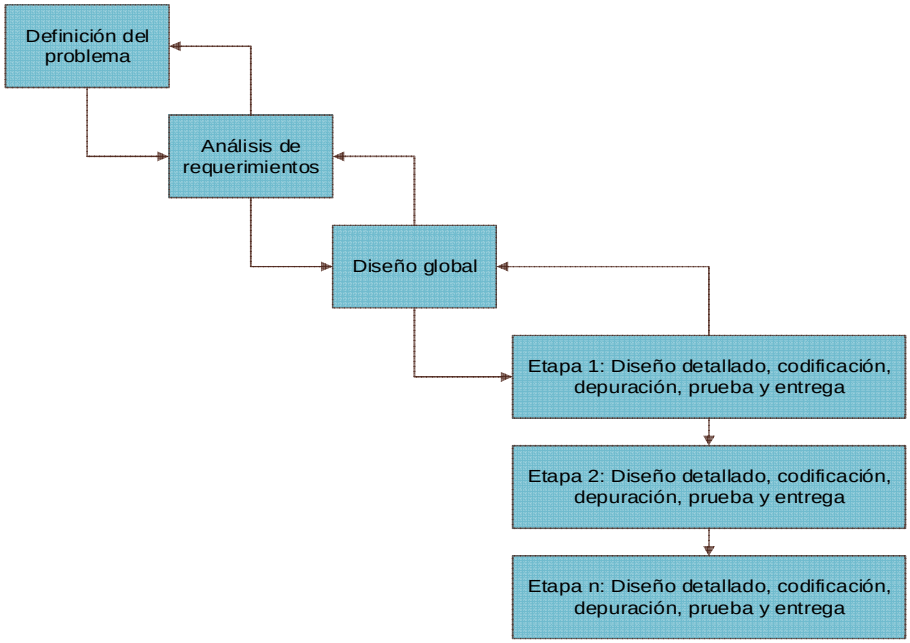


Figura 4.1 Metodología entrega por etapas

El análisis geoestadístico, como se explicó en el capítulo 2, se divide en varias etapas y se implementaron de la siguiente manera:

Etapa 1: Implementación de la etapa de análisis exploratorio. En esta etapa se implementan los módulos para la creación, edición y actualización de archivos creados desde la aplicación.

También se implementan los módulos principales donde se realizará el análisis geostadístico llamado *Análisis*, uno para 2D y otro para 3D. En estos módulos se implementan las regiones para la carga de un archivo o archivos de datos (*x,y,val* o *x,y,z,val*), permitiendo la visualización de la ubicación en planta de los datos con base en las coordenadas *x, y*. Asimismo, se visualizan los parámetros descriptivos descritos en el capítulo 2.

Por otra parte, se implementa la clase para la creación de un árbol que despliega los archivos de la carpeta seleccionada donde se encuentra el archivo o archivos que contienen los datos para el análisis, este árbol a través del análisis se actualiza con la creación de nuevos archivos.

En los análisis se implementa la clase para generar un histograma, que en caso necesario puede modificarse.
Para el caso del análisis en tres dimensiones, se implementan procedimientos para la corrección de elevación por medio de dos métodos: por *sondeo de referencia*, o bien, por *elevación de referencia*. La corrección de elevación crea nuevos archivos a partir de los cuales se continúa con el análisis geostadístico y los procedimientos correspondientes a la creación de archivos son también implementados.

Etapa 2: Implementación de la etapa de análisis de tendencia. En esta etapa se implementa al módulo *Análisis* las funciones y procedimientos para realizar la regresión lineal. Por tanto, se añaden procedimientos para crear archivos de diferente extensión a los originales para facilitar la distinción entre los diferentes archivos generados durante el análisis.

En el caso de dos dimensiones se integran procedimientos que muestran los resultados obtenidos en pantalla, así como, los coeficientes de la regresión lineal. Para el caso de tres dimensiones se integran procedimientos que solo muestran los coeficientes de la regresión lineal.

En esta etapa se optimizó la carga de archivos y despliegue de la gráfica de ubicación en planta de los datos, con base en las observaciones de los usuarios.

Etapa 3: Implementación de la etapa de análisis de correlación. En esta etapa se implementa las funciones y procedimientos para la creación de correlogramas, cálculo de las distancias de correlación y cálculo del número de pares de datos tomados en cuenta para el cálculo del correlograma.

En el caso del análisis de dos dimensiones se visualizan 2 ó 4 correlogramas (dependiendo de la selección del usuario). En estos correlogramas el usuario tiene la oportunidad de modificar la distancia de correlación teórica para tener un mejor ajuste al correlograma experimental. La modificación en la distancia de correlación se ve reflejada en la gráfica correspondiente.

En el caso del análisis de tres dimensiones se muestran solo dos correlogramas representativos uno para la dirección horizontal y otro para la dirección vertical, así como, el cálculo de las respectivas distancias de correlación.

Por último, se implementa una función para que el usuario tenga la posibilidad de guardar como imagen cualquiera de los correlogramas visualizados para su posterior uso o referencia. Asimismo, es posible guardar cualquiera de las gráficas visualizadas con formato de imagen (*.jpg).

Etapa 4: Implementación de la etapa de estimación/simulación. En esta etapa se implementan las funciones y procedimientos para la estimación y simulación en dos y tres dimensiones.

Por la similitud de parámetros de entrada para el cálculo de ambos procedimientos se decidió presentar ambos en la misma sección y el usuario elige el procedimiento que desea realizar.

En el caso de dos dimensiones se asigna un paso de cálculo, con este paso de cálculo se implementa una función para crear una malla de puntos, donde en cada punto se obtendrá un valor estimado o simulado. La malla creada puede ser modificada si el usuario lo requiere, por lo que, se implementa la función para la modificación de la malla de predicción.

En el caso de tres dimensiones se implementan funciones para crear y cargar un archivo donde se especifican las coordenadas de los puntos donde se desea estimar o simular. Cuando se realiza el cálculo se llama a una función que crea un archivo de resultados.

En esta etapa también se optimizaron los procedimientos para la presentación de las gráficas de los correlogramas y modificación de las distancias de correlación, así también, para ayudar al usuario se asignaron cajas de ayuda referente al tipo de dato que se deben ingresar para realizar el cálculo.

4.5 Diagrama de actividades

El desarrollo de la metodología se ilustra en el diagrama de la *figura 4.2*.

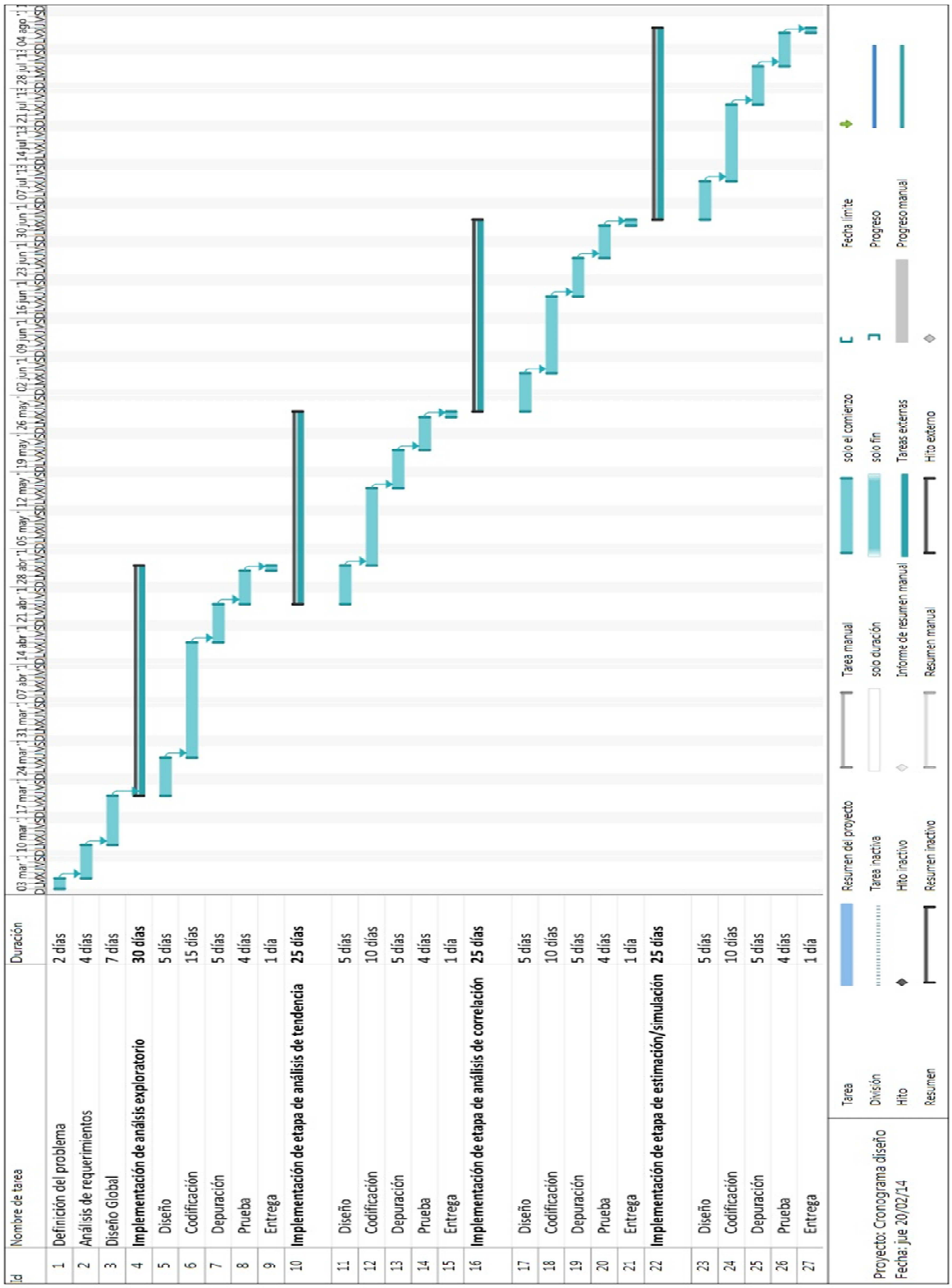


Figura 4.2 desarrollo de la metodología propuesta

4.6 Plataforma de desarrollo

Microsoft Visual Studio 2010 Ultimate

Microsoft Visual Studio 2010 Ultimate es un entorno de desarrollo integrado (IDE) para sistemas operativos basados en Windows, con desarrollo multiplataforma. Es una plataforma que soporta varios lenguajes de programación tales como Visual C++, Visual C#, Visual J#, ASP:NET y Visual Basic:NET, entre otros.

Microsoft Visual Studio 2010 Ultimate presenta las siguientes características (MSDN, Microsoft, 2010):

- **Apariencia y Comportamiento:** Legibilidad, Compatibilidad con distintos monitores, las ventanas de documento, como la ventana Editor de código y Vista de diseño, pueden situarse fuera de la ventana del IDE**.**

- **Administración del ciclo de vida de las aplicaciones**: La creación de aplicaciones de éxito requiere un proceso de ejecución uniforme que beneficie a todos los componentes del equipo. Las herramientas de ALM integradas en Visual Studio 2010 Ultimate contribuyen a que las organizaciones colaboren y se comuniquen de forma efectiva en todos los niveles, y a que se hagan una idea precisa del estado real del proyecto, lo que garantiza que se ofrezcan soluciones de gran calidad al tiempo que se reducen los costos.

- **Depuración y diagnóstico**: Visual Studio 2010 Ultimate presenta IntelliTrace, una valiosa característica de depuración que hace que el argumento "no reproducible" sea cosa del pasado. Los evaluadores pueden archivar errores enriquecidos y modificables para que los desarrolladores puedan reproducir siempre el error del que se informe en el estado en el que se encontró.

- **Herramientas de prueba:** Visual Studio 2010 Ultimate incorpora herramientas avanzadas de pruebas para ayudarle a garantizar la calidad del código en todo momento.

- **Arquitectura y modelado:** El explorador de arquitectura de Visual Studio 2010 Ultimate ayuda a entender los activos de código existentes y otras interdependencias. Los diagramas por capas ayudan a garantizar el cumplimiento de la arquitectura y permiten validar artefactos de código con respecto al diagrama. Además, Visual Studio 2010 Ultimate admite los cinco diagramas de UML más comunes que conviven junto con su código.

- **Desarrollo de bases de datos**: Visual Studio 2010 Ultimate proporciona potentes herramientas de implementación y administración de cambios que garantizan que la base de datos y la aplicación estén siempre sincronizadas.

- **Entorno de desarrollo integrado:** Permite el aprovechamiento de las características personalizables como, por ejemplo, compatibilidad con varios monitores, de modo que se pueda organizar y administrar el trabajo como se quiera.

- **Compatibilidad con la plataforma de desarrollo:** Visual Studio 2010 Ultimate permite trabajar en una gran variedad de plataformas, entre las que se incluyen Windows, Windows Server, Web, Cloud, Office y SharePoint, entre otras, todo en un único entorno de desarrollo integrado.

- **Team Foundation Server:** Team Foundation Server (TFS) es la plataforma de colaboración sobre la que se asienta la solución de administración de ciclo de vida de aplicaciones de Microsoft. TFS automatiza y simplifica el proceso de entrega de software, y proporciona rastreabilidad completa y la posibilidad de comprobar en tiempo real el estado de los proyectos (para todos los miembros del equipo) con potentes herramientas de elaboración de informes y paneles.

- **Lenguaje Visual Basic**: Visual Basic dispone de nuevas características que acortan la sintaxis y permiten escribir código con más rapidez. Estas características contienen propiedades implementadas automáticamente, continuación de línea implícita, inicializadores de colección y expresiones lamdba de una y varias instrucciones. Además, Visual Basic ahora es compatible con la implementación simplificada a través de la equivalencia de tipos. Para obtener más información, vea Novedades de Visual Basic 2010.

4.7 Lenguaje de desarrollo

Visual Basic

Visual Basic es una herramienta de diseño de aplicaciones para Windows; un lenguaje de fácil aprendizaje pensado tanto para programadores principiantes como para expertos. Su sintaxis, derivada del BASIC, ha sido ampliada con el tiempo al agregarse las características típicas de los lenguajes estructurados modernos. Para entender mejor este concepto es necesario descomponer su nombre en dos partes:

> VISUAL: propone crear de forma sencilla la interfaz de usuario mediante objetos (controles) prediseñados, reduciendo el tiempo de trabajo.
>
> BASIC: Esta palabra es la abreviación de la sigla en inglés Beginners All-Purpose Symbolic Instruction Code. Se refiere al lenguaje de programación más utilizado en la historia de la informática y la programación. En principio era más sencillo, actualmente muchas más instrucciones y métodos se han adaptado para crear aplicaciones bajo ambiente Windows.

Visual Basic (VB) permite a los desarrolladores apuntar a Windows, Web y dispositivos móviles.

Con VB se crean aplicaciones que corren en Windows, es un ambiente *orientado a lo visual* que proporciona librerías de herramientas y objetos para aplicaciones de programación.

Los errores de programación no se generan tan frecuentemente y, en caso de presentarse, son más fáciles de depurar. Además, incluye dos conceptos importantes:

Un **método visual** de creación de aplicaciones, incluyendo formularios, controles y componentes del formulario.

Habilidad de asociar código directamente a cada evento de cada elemento del diseño visual.

Las aplicaciones creadas con Visual Basic están basadas en objetos y emplean un modelo de programación manejada por eventos, centrado en un motor de formularios que facilita el rápido desarrollo de aplicaciones gráficas. No es programación orientada a objetos como C++ o Java. Visual Basic utiliza objetos con propiedades y métodos, y aunque admite el *polimorfismo* mediante el uso de interfaces, no admite la *herencia* de mecanismos propio de los lenguajes orientados a objetos.

Una aplicación desarrollada con VB se compone por una parte de código puro, y por otra de partes asociadas a los objetos que forman la interfaz gráfica. No requiere de manejo de punteros y posee un manejo muy sencillo de cadenas de caracteres. Posee varias bibliotecas para manejo de bases de datos.

VB es frecuentemente utilizado debido a la rapidez con la que puede hacerse un programa, pero no se pueden comparar sus facilidades con otros lenguajes cuando se desea llegar al fondo de la máquina y controlar uno a uno sus registros. La finalidad de VB no es llegar a esas precisiones, en tal caso será necesario utilizar otro lenguaje que permita bajar el nivel de programación o crear librerías (DLLs) que lo realicen. En la mayoría de las aplicaciones las herramientas aportadas por VB son suficientes para lograr un programa fácil de realizar y de altas prestaciones.

4.8 Pruebas

Existen pruebas indicadoras para determinar que la aplicación desarrollada se encuentra lista para el uso del usuario, en el caso de esta aplicación se planteó realizar las siguientes pruebas.

Pruebas de unidad

Se planeó realizar pruebas de caja negra con el objetivo de encontrar las clases válidas para ciertas variables. La forma que se empleó para conocer estas clases válidas fue completando una tabla con el formato de la *tabla 4.2*.

Tabla 4.2 Tabla de prueba

Variables	Clases válidas	Clases inválidas

Ejemplos de algunas variables que se revisaron en estas pruebas son: *Pasos de cálculo*, *Tolerancias lineales*, *Ancho de banda* y parámetros de modificación de malla.

Se planeó realizar pruebas de caja blanca ejecutando al menos una vez todos los caminos independientes de cada módulo con el fin de:

- Probar las decisiones en su parte verdadera y en su parte falsa.
- Ejecutar todos los bucles en sus límites.
- Utilizar todas las estructuras de datos internas.

Ambos tipos de pruebas permitieron conocer el límite máximo de datos con los que podrá trabajar la herramienta.

Pruebas de integración

Se planteó realizar este tipo de pruebas cada vez que se añadieron cada una de las etapas del análisis, finalmente se realizó también esta misma prueba después de integrar los análisis en 2D y 3D en una interfaz.

Se realizaron pruebas de regresión con la intención de probar los nuevos componentes integrados, así como, los componentes ya existentes. Esto se realizó con el fin de asegurar que la introducción de los nuevos módulos no produjeran cambios que necesitaran ser ajustados.

Pruebas de validación

Puesto que, la herramienta desarrollada está basada en cálculos matemáticos, con este tipo de pruebas se pretendió validar que los resultados entregados por la herramienta fueran correctos. En estas pruebas se planteó comprobar los resultados comparándolos con resultados de análisis previamente desarrollados usando otras herramientas.

5 Desarrollo de los algoritmos para diseño de diagramas de flujo.

Como se explicó en el capítulo 2, existen algoritmos que se ejecutan durante un análisis geoestadístico. En este capítulo se describe el desarrollo de los principales algoritmos implementados y sus correspondientes diagramas de flujo con los acoplamientos realizados para el desarrollo de la aplicación.

5.1 Desarrollo de algoritmos

5.1.1 Desarrollo del algoritmo para el análisis geoestadístico en 2D

A continuación, se presentan los principales algoritmos desarrollados para cada etapa del análisis geoestadístico en 2D.

Análisis de tendencia

Como se explicó anteriormente, para realizar el análisis de tendencia se recurre al método de regresión lineal, el algoritmo correspondiente es:

calculo_Residuos
Entrada: Conjunto *n* de puntos en el plano con coordenada *x, y* y un valor *V*
Salida: Regresión lineal de *n* puntos y coeficientes *a, b, c* de la regresión

For i=0 **to** i <= *n*-1
hacer $x_p = x_p + x(i)$
hacer $y_p = y_p + y(i)$
hacer $V_p = V_p + V(i)$
Next
hacer $x_p = x_p/n$
hacer $y_p = y_p/n$
hacer $V_p = V_p/n$
For i=0 **to** i <= *n*-1
hacer $K_{xx} = K_{xx} + (x(i) - x_p)^2$
hacer $K_{yy} = K_{yy} + (y(i) - y_p)^2$
hacer $K_{xy} = K_{xy} + (x(i) - x_p) * (y(i) - y_p)$
hacer $C_x = C_x + (x(i) - x_p) * (V(i) - V_p)$
hacer $C_y = C_y + (y(i) - y_p) * (V(i) - V_p)$
Next
hacer $K_{xx} = K_{xx} / n$
hacer $K_{yy} = K_{yy} / n$
hacer $K_{xy} = K_{xy} / n$
hacer $C_x = C_x / n$
hacer $C_y = C_y / n$
K(2,2)
Donde K(0,0) = $K_{xx,}$ K(0,1) =$K_{xy,}$ K(1,0) =$K_{xy,}$ K(1,1) =K_{yy}
CC(2,1)
Donde CC(0,0)=$C_{x;}$ CC(1,0)C_y
AB(2,1)
hacer K.inversa
AB = K.inversa*CC
Donde a = AB(0,0)
b = AB(1,0)
c = $V_p - a * x_p - b * y_p$

Análisis estructural

Para realizar el análisis estructural se elaboró el siguiente algoritmo:

calculoDistCorr
Entrada: Conjunto *n* de puntos en el plano con coordenada *x, y* y un valor *V*
If campo es estacionario T**hen**
    If datos distribuidos en forma regular **Then**
        hacer Cálculo del correlograma experimental
        hacer Obtención distancia de correlación
        hacer Definición del modelo de correlación
    Else
        hacer Definir tolerancias lineales y angulares
    End If
Else
    hacer *calculo_Residuos*
End If

Estimación

Como se explicó anteriormente, para la estimación, se empleó la técnica de Kriging Ordinario, el algoritmo es:

kriging_Ordinario
Entrada: Conjunto *n* de puntos en el plano con coordenada *x, y* y un valor *V*
Salida: Conjunto de puntos estimados con coordenadas x, *y* y un valor *V*
    hacer identificación distancia correlación más corta
    hacer obtención de sistema de ecuaciones de Kriging
    For i=1 **To** *n*
        For j =1 **To** *n*
            hacer identificación de *n* coeficientes λ
            donde
                $$\lambda = C(x_i - x_j)^{-1} * C(x_i - x)$$
        Next
    Next
    hacer obtención multiplicador Lagrange, *v*
    Temp(*n*)
        For i=1 **To** *n*
            hacer Temp(i)=$\lambda_i C_v(x, x_i)$
        Next
    hacer obtención varianza de estimación $\sigma_E^2(x) = Var[V(x)] + v -$Temp

Simulación

De igual manera, para la simulación se empleó la técnica de simulación condicional, el algoritmo de la simulación condicional es:

simulacion
Entrada: Conjunto *n* de puntos en el plano con coordenada *x, y* y un valor *V*
Salida: conjunto de puntos simulados con coordenadas x, *y* y un valor *V*
    hacer *kriging_Ordinario* (*V**)
    hacer realización incondicional de campo aleatorio. (*V***)
    hacer secuencia de variables aleatorias (*Z*)
    hacer descomposición de matriz de correlación (*L*)
    A partir de *L* obtener campo aleatorio normal estándar correlacionado (*G*)
        For i=0 **To** i<=n-1
            For j = 0 **To** j = i

hacer obtener campo aleatorio normal estándar correlacionado (*G*)
donde

$$G(x_i) = G(x_i) + L_{ij}Z_j$$

Next
Next

Conociendo la media y varianza

For i=0 **To** i<=n-1
hacer $V^{**}(x_i) = \mu V(x_i) + \sigma V(x_i) * G(x_i)$
Next

hacer *kriging_Ordinario* de la simulación incondicional. (*V****)
For i=0 **To** i<=n-1
hacer combinación de los tres campos $V^C(x_i)$
donde

$$V^C(x_i) = V^*(x_i) + [V^{**}(x_i) - V^{***}(x_i)]$$

Next

5.1.2 Desarrollo del algoritmo para el análisis geoestadístico en 3D

A continuación se presentan los principales algoritmos desarrollados para cada etapa del análisis en tres dimensiones.

Análisis de Tendencia

A continuación se muestra el algoritmo correspondiente del cálculo de residuos en 3D.

calculo_Residuos
Entrada: Conjunto *n* de puntos en el plano con coordenada *x*, *y*, *z* y un valor *V*
Salida: Regresión lineal de *n* puntos y coeficientes *a*, *b*, *c* y *d* de la regresión

For i=0 **To** i <= n-1
hacer $x_p = x_p + x(i)$
hacer $y_p = y_p + y(i)$
hacer $z_p = z_p + z(i)$
hacer $V_p = V_p + V(i)$
Next
hacer $x_p = x_p/n$
hacer $y_p = y_p/n$
hacer $z_p = z_p/n$
hacer $V_p = V_p/n$
For i=0 **To** i <= n-1
hacer $K_{xx} = K_{xx} + (x(i) - x_p)^2$
hacer $K_{yy} = K_{yy} + (y(i) - y_p)^2$
hacer $K_{zz} = K_{zz} + (z(i) - z_p)^2$
hacer $K_{xy} = K_{xy} + (x(i) - x_p) * (y(i) - y_p)$
hacer $K_{xz} = K_{xz} + (x(i) - x_p) * (z(i) - z_p)$
hacer $K_{yz} = K_{yz} + (y(i) - y_p) * (z(i) - z_p)$
hacer $C_x = C_x + (x(i) - x_p) * (V(i) - V_p)$
hacer $C_y = C_y + (y(i) - y_p) * (V(i) - V_p)$
hacer $C_z = C_z + (z(i) - z_p) * (V(i) - V_p)$
Next
hacer $K_{xx} = K_{xx} / n$
hacer $K_{yy} = K_{yy} / n$
hacer $K_{zz} = K_{zz} / n$
hacer $K_{xy} = K_{xy} / n$
hacer $K_{xz} = K_{xz} / n$
hacer $K_{yz} = K_{yz} / n$
hacer $C_x = C_x / n$
hacer $C_y = C_y / n$
hacer $C_z = C_z / n$

K(3,3)
Donde K(0,0) = K_{xx}; K(0,1) =K_{xy}; K(0,2) =K_{xz}; K(1,0) =K_{xy}; K(1,1) =K_{yy}; K(1,2) =K_{yz}; K(2,0) =K_{xz}; K(2,1) =K_{yz}; K(2,2) =K_{zz}
CC(3,1)
Donde CC(0,0)=C_x; CC(1,0)=C_y; CC(2,0)=C_z
AB(3,1)
hacer K.inversa
AB =K.inversa*CC
Donde a = AB(0,0)
b = AB(1,0)
c = AB(2,0)
d =$V_p - a * x_p - b * y_p - c * z_p$

Análisis estructural

Para realizar el análisis estructural se presenta el siguiente algoritmo.

calculoDistCorr
Entrada: Conjunto *n* de puntos en el plano con coordenada *x*, *y*, *z* y un valor *V*
If campo es estacionario **Then**
If datos distribuidos en forma regular **Then**
hacer Cálculo del correlograma experimental
hacer Obtención distancia de correlación
hacer Definición del modelo de correlación
Else
hacer Definir tolerancias lineales y angulares
End If
Else
hacer *calculo_Residuos*
End If

Estimación

El algoritmo para el Kriging ordinario es el siguiente.

kriging_Ordinario
Entrada: Conjunto n de puntos en el plano con coordenada x, y, z y un valor V
Salida: Conjunto de puntos estimados con coordenadas x, *y, z* y un valor *V*
hacer identificación distancia más corta
hacer obtención de sistema de ecuaciones de Kriging
For i=1 **To** *n*
For j =1 **To** *n*
hacer identificación de *n* coeficientes λ
donde
$$\lambda = C(x_i - x_j)^{-1} * C(x_i - x)$$
Next
Next
hacer obtención multiplicador Lagrange, *v*
Temp(*n*)
For i=1 **To** *n*
hacer Temp(i)=$\lambda_i C_v(x, x_i)$
Next
hacer obtención varianza de estimación $\sigma_E^2(x) = Var[V(x)] + v -$Temp

Simulación

El algoritmo de la simulación condicional se presenta de la siguiente forma.

simulacion
Entrada: Conjunto *n* de puntos en el plano con coordenada *x*, *y*, *z* y un valor *V*
Salida: Conjunto de puntos simulados con coordenadas x, *y*, *z* y un valor *V*

hacer *kriging_Ordinario* (*V**)
hacer realización incondicional de campo aleatorio. (*V***)
hacer secuencia de variables aleatorias (*Z*)
hacer descomposición de matriz de correlación (*L*)
A partir de *L* obtener campo aleatorio normal estándar correlacionado (*G*)

For i=0 **To** i<=n-1
For j = 0 **To** j = i
hacer obtener campo aleatorio normal estándar correlacionado (*G*)
donde

$$G(x_i) = G(x_i) + L_{ij}Z_j$$

Next
Next

Conociendo la media y varianza

For i=0 **To** i<=n-1
hacer $V^{**}(x_i) = \mu V(x_i) + \sigma V(x_i) * G(x_i)$
Next

hacer *kriging_Ordinario* de la simulación incondicional. (*V****)

For i=0 **To** i<=n-1
hacer combinación de los tres campos $V^C(x_i)$
donde

$$V^C(x_i) = V^*(x_i) + [V^{**}(x_i) - V^{***}(x_i)]$$

Next

5.2 Acoplamiento o ajuste para la implementación

Para implementar los algoritmos descritos anteriormente fue necesario realizar algunos ajustes, quedando finalmente como acontinuación se explica.

A los algoritmos correspondientes a la regresión lineal en 2D y 3D no se les realizó algún ajuste.

El algoritmo del análisis estructural 2D fue ajustado, quedando de la siguiente forma.

calculoDistCorr
Select Case *puntos*

Case *pOrig*
Entrada: Conjunto *n* de puntos en el plano con coordenada *x*, *y* y un valor *V*, paso de cálculo *h*, tolerancia lineal *xtol*, ancho de banda *bandw*, tolerancia angular *atol*
hacer Cálculo del correlograma experimental
hacer Obtención distancia de correlación
hacer Definición del modelo de correlación

Case *pRes*
Entrada: Conjunto *n* de puntos en el plano con coordenada *x*, *y* y un valor de regresión *V*, paso de cálculo *h*, tolerancia lineal *xtol*, ancho de banda *bandw*, tolerancia angular *atol*
hacer Cálculo del correlograma experimental
hacer Obtención distancia de correlación
hacer Definición del modelo de correlación

End Select

En el análisis 2D, el algoritmo para realizar la estimación y la simulación quedó de la siguiente manera.

estimSimu
Entrada: paso de cálculo *h*
Salida*:* Conjunto de puntos en el plano con coordenada *x*, *y*, un valor *V*, y dependiendo, valores de estimación y dispersión o de simulación.

Select Case *Proc*
Case *estimación*
hacer creación de malla
hacer *kriging_Ordinario*
Case *simu*
hacer creación de malla
hacer *simulacion*
End Select

El algoritmo del análisis estructural 3D fue ajustado, quedando de la siguiente forma.

calculoDistCorr
Select Case *campo*
Case *campoEst*
Entrada: Conjunto *n* de puntos en el plano con coordenada *x*, *y*, *z* y un valor *V*, paso de cálculo
hacer Cálculo del correlograma experimental horizontal
hacer Cálculo del correlograma experimental vertical
hacer Obtención distancia de correlación horizontal
hacer Obtención distancia de correlación vertical
hacer Definición del modelo de correlación horizontal
hacer Definición del modelo de correlación vertical
Case *campoNoEst*
Entrada: Conjunto *n* de puntos en el plano con coordenada *x*, *y*, *z* y un valor *V*, paso de cálculo
hacer Cálculo del correlograma experimental horizontal
hacer Cálculo del correlograma experimental vertical
hacer Obtención distancia de correlación horizontal
hacer Obtención distancia de correlación vertical
hacer Definición del modelo de correlación horizontal
hacer Definición del modelo de correlación vertical
End Select

En el análisis 3D, el algoritmo para realizar la estimación y la simulación, quedó de la siguiente manera.

estimSimu
Entrada: Conjunto *n* de puntos en el plano con coordenada *x*, *y*, *z* y un valor *V*, distancia correlación horizontal *distCorrH*, distancia de correlación vertical *distCorrV*, nodos *nodos*
Salida: Conjunto de puntos simulados con coordenadas x, *y, z* y un valor *V*

Select Case *Proc*
Case *estimacion*
hacer creación de malla
hacer *kriging_Ordinario*
Case *simu*
hacer creación de malla
hacer *simulacion*
End Select

5.3 Diagramas de flujo

A continuación se presentan los diagramas de flujo de los algoritmos ajustados para su implementación.

Los diagramas de flujo para el análisis de tendencia enen 2D y 3D se muestran en las *figuras 5.1* y *5.2*.

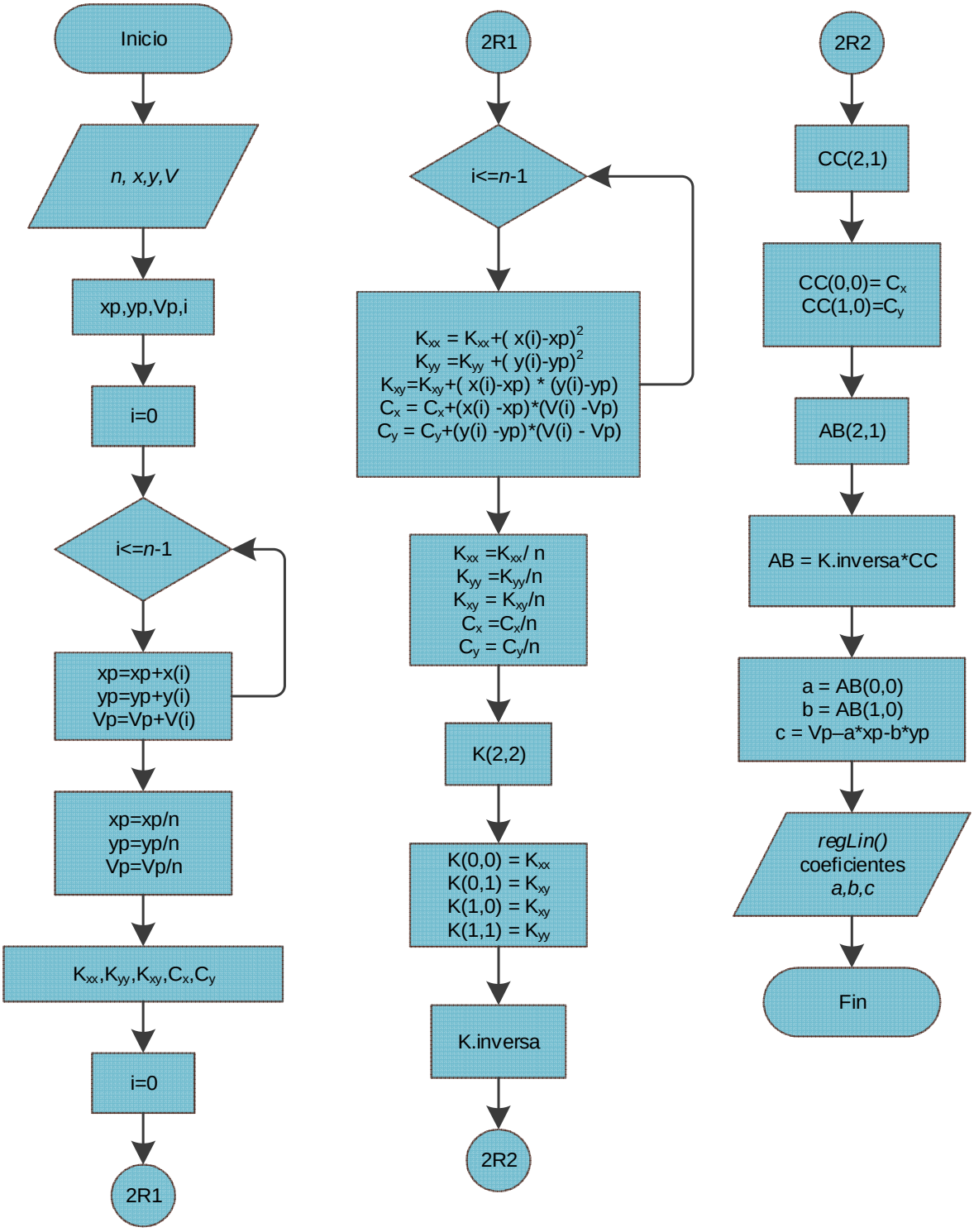


Figura 5.1 Regresión lineal 2 dimensiones.

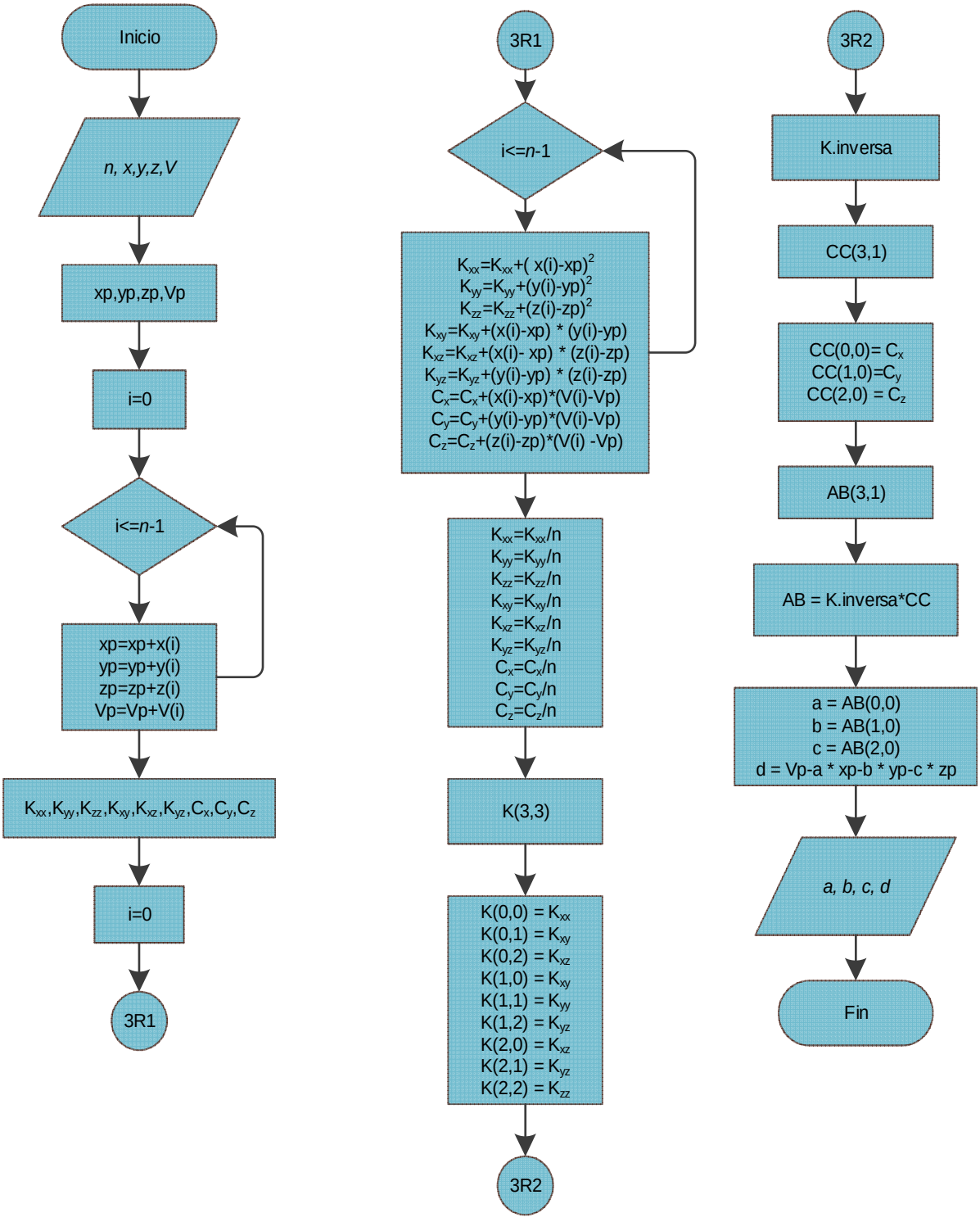


Figura 5.2 Regresión lineal 3 dimensiones.

Los algoritmos para el análisis estructural en 2D y 3D con su acoplamiento se muestran en las *figuras 5.3* y *5.4.*:

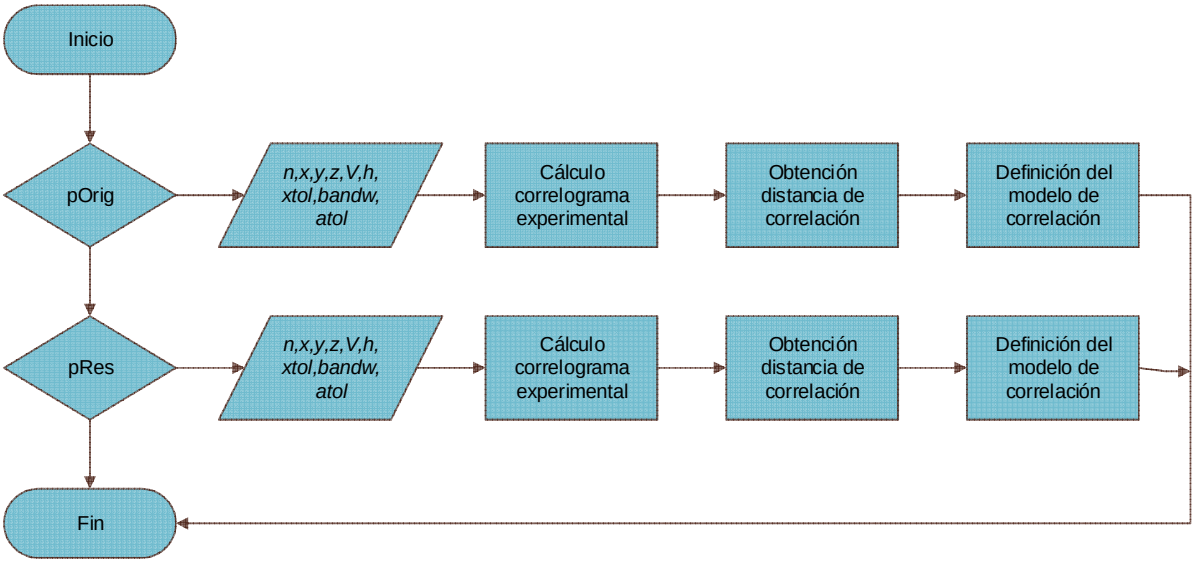


Figura 5.3 Análisis Estructural 2 dimensiones.

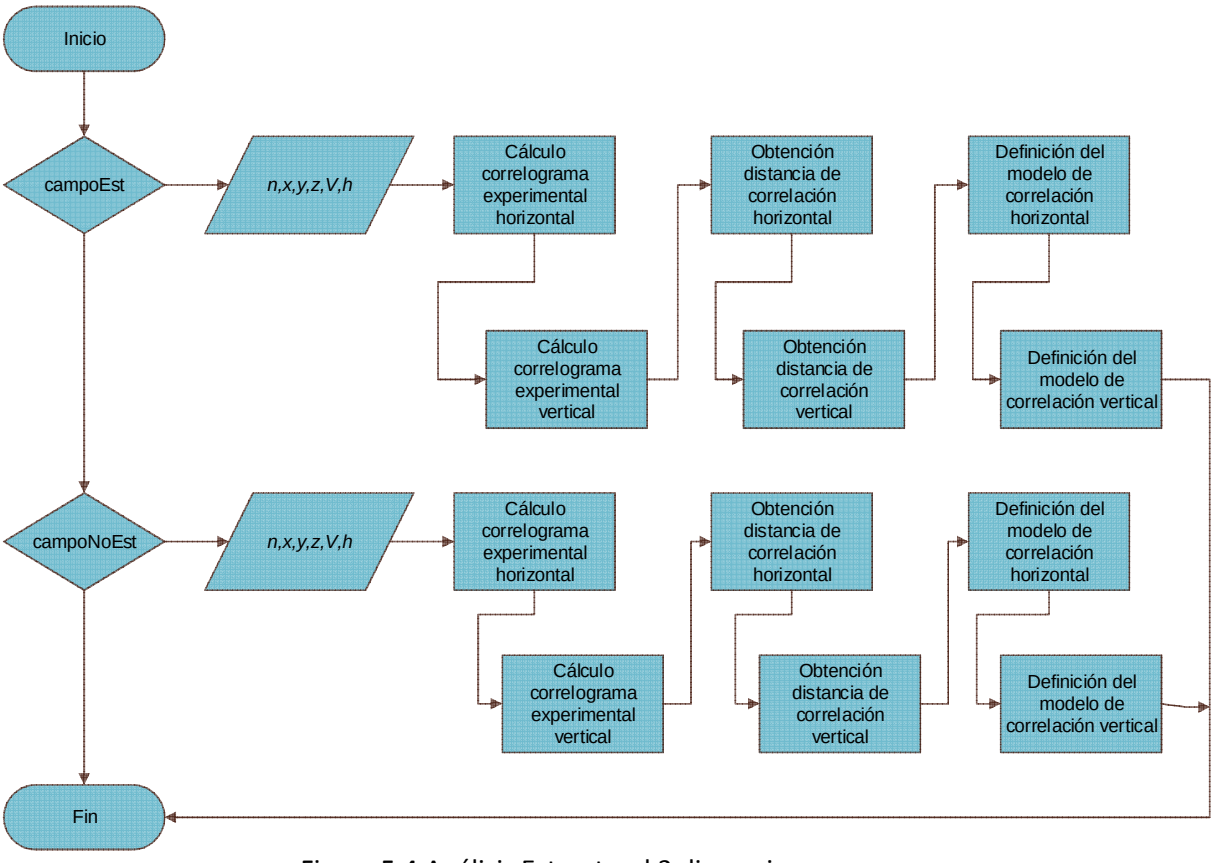


Figura 5.4 Análisis Estructural 3 dimensiones.

Los algoritmos para la estimación y simulación en 2D y 3D con su acoplamiento se muestran en las *figuras 5.5* y *5.6.*:

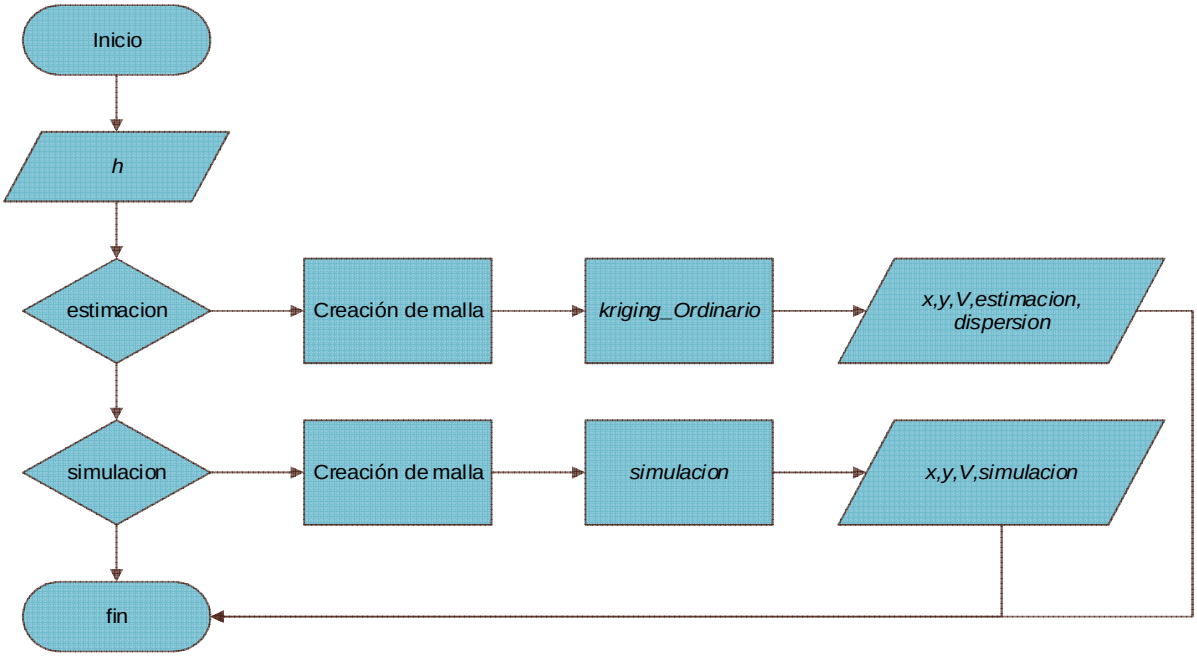


Figura 5.5 Estimación/Simulación 2 dimensiones

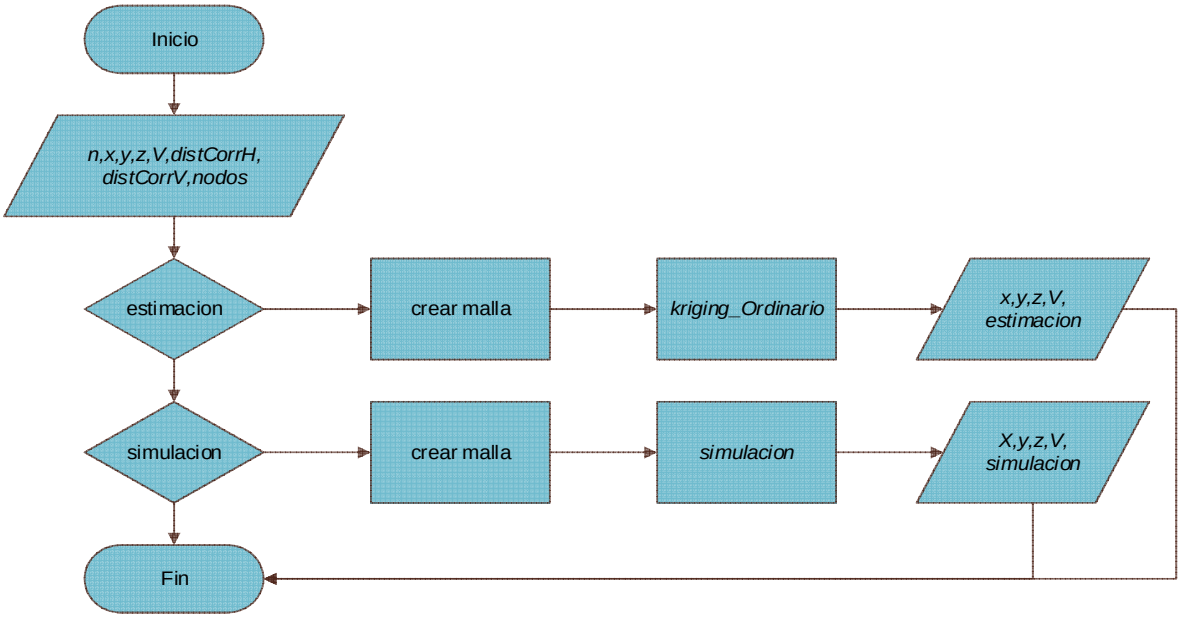


Figura 5.6 Estimación/Simulación 3 dimensiones.

6 Desarrollo de la aplicación

En este capítulo se presenta la arquitectura general del software desarrollado. El sistema cuenta con una serie de estructuras, clases, módulos y regiones principales; que se describen también en este apartado.

6.1 Arquitectura de la aplicación

La arquitectura de la aplicación se presenta en la *figura 6.1*.

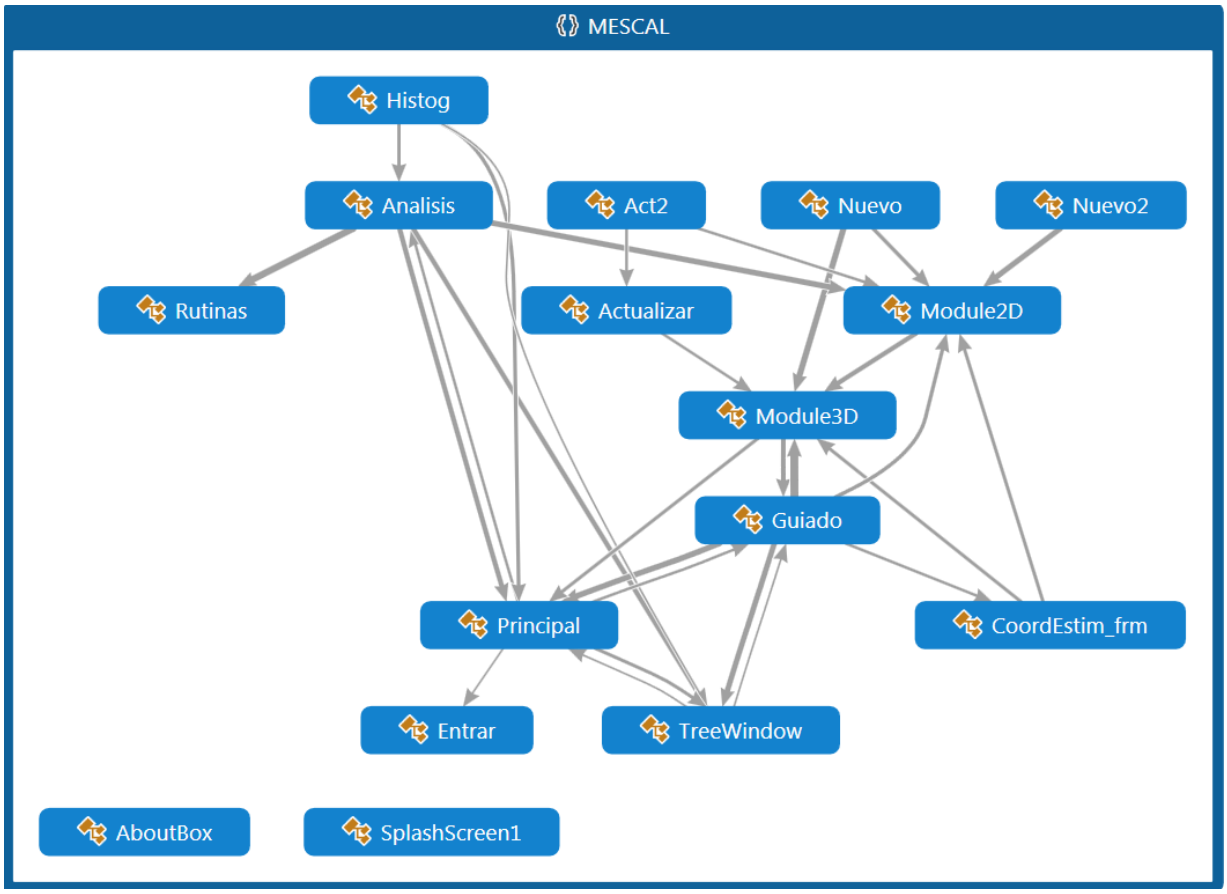


Figura 6.1 Arquitectura de la aplicación

6.1.1 Estructuras principales

Las principales estructuras creadas y utilizadas en el programa se describen a continuación.

La estructura *plano* sirve para dar estructura a los archivos de planos que se utilizan en el proceso del análisis en 2D, consta de una serie de declaraciones de variables y una rutina para dimensionar los arreglos de *x*, *y* y *val*. La declaración es de la siguiente manera.

```
Public Structure plano
    Dim id As String
    Dim np As Integer
    Dim var As String
    Dim unit As String
```

```
        Dim x() As Double
        Dim y() As Double
        Dim val() As Double
        Dim nomArchivo As String
        Dim coefHip() As Double

        Sub iniciarCaptura(ByVal np As Integer)
            ReDim x(np - 1)
            ReDim y(np - 1)
            ReDim val(np - 1)
        End Sub
    End Structure
```

La estructura *sondeo* sirve para dar estructura a los archivos de los sondeos que se utilizan en el proceso del análisis en 3D. Al igual que la estructura *plano* consta de una serie de declaraciones de variables y una rutina para dimensionar los arreglos de *z* y *val*. Se declara de la siguiente manera.

```
Public Structure sondeo
        Dim id As String
        Dim x As Double
        Dim y As Double
        Dim np As Integer
        Dim var As String
        Dim unit As String
        Dim elev As Double
        Dim broc As Boolean
        Dim z() As Double
        Dim val() As Double
        Dim nomArchivo As String
        Dim disTipica As Double
        Dim coefHip() As Double
        Dim corrElev As Double
        Dim distCorr As Double
        Dim indCorr As Double

        Sub iniciarCaptura(ByVal np As Integer)
            ReDim z(np - 1)
            ReDim val(np - 1)
        End Sub
    End Structure
```

La estructura *punto* sirve para dar estructura a los puntos para la estimación/simulación. Se define de la siguiente manera:

```
    Public Structure punto
        Dim x As Double
        Dim y As Double
        Dim z As Double
        Sub newPunto(ByVal x As Double, ByVal y As Double, ByVal z As Double)
            Me.x = x
            Me.y = y
            Me.z = z
        End Sub
    End Structure
```

6.1.2 Clases

Las principales clases creadas y utilizadas en el programa, con sus regiones principales son las que a continuación se describen.

La clase A*nalisis* es la encargada de realizar el análisis en 2D. Entre sus regiones se encuentran cada parte del análisis, estas regiones se encuentran indicadas como: *proyecto, análisis de tendencia, análisis estructural* y *estimación*.

La clase *Guiado* es la encargada de realizar el análisis en 3D. Entre sus regiones se encuentran cada parte del análisis, estas regiones se encuentran indicadas como: *datos proyecto, análisis estructural* y *estim/simulación*.

La clase *Act2* se encarga de la actualización de archivos *.po* que son los datos originales para el caso del análisis en 2D. Se encarga de mostrar los datos del archivo en una tabla y sobre esta realizar las modificaciones necesarias y guardar los cambios.

La clase *Actualizar* se encarga de la actualización de archivos *.so* que son los datos de los sondeos originales para el caso del análisis en 3D. Muestra los datos en una tabla en la que se hacen los cambios necesarios y posteriormente se guardan los cambios.

La clase *Nuevo2* se encarga de crear los archivos *.po* para el análisis en 2D. Se escriben los datos en una tabla y se verifica de que todos los valores sean valores válidos y que todos los campos estén completos.

La clase *Nuevo* se encarga de crear los archivos *.so* para el análisis en 3D. Se escriben los datos en una tabla y se verifica de que todos los valores sean valores válidos y que todos los campos estén completos.

6.1.3 Módulos

Los principales módulos creados y utilizados en el programa son los siguientes:

El módulo *Module1* contiene las funciones y rutinas utilizadas para el análisis en 3D (contiene todo el desarrollo matemático utilizado para el análisis). Se divide en regiones: *Parámetros estadísticos, Simulación, Estimación, Análisis estructural, Regresión lineal, Análisis de cotas* (corrección elevación) y *Lectura y escritura de archivos.* Estas funciones y rutinas son llamadas desde la clase *Guiado, Nuevo* y *Actualizar* (en el caso de estas dos últimas solo invocaran funciones de la región de *Lectura y escritura de archivos*). Se utilizan funciones de la biblioteca MaPack.dll.

El módulo *Rutinas* contiene las funciones utilizadas para el análisis en 2D, esta se encarga de contener todo lo referente a los cálculos matemáticos que se realizan durante el análisis. Se divide en regiones: *Estructuras, Análisis estructural, Regresion lineal, Estimación.* Estas funciones son invocadas desde la clase *Analisis*, adicionalmente, en el módulo se utilizan funciones de la biblioteca MaPack.dll.

El módulo *Module2* contiene las funciones y rutinas utilizadas para tratamiento de archivos para el análisis en 2D. Funciones para la lectura, escritura y modificación de los planos originales *.po* y creación de los archivos *.por* (planos sin tendencia), *.pe* (planos estimados) y *.psi* (planos simulados).

6.1.4 Regiones

Con base en lo anteriormente descrito, se seleccionaron las regiones más importantes que se explican a continuación:

En la clase *Analisis* destacan las regiones:

- *Análisis de tendencia*: En esta región se encuentran los procedimientos y funciones para realizar el análisis de tendencia en 2D y los procedimientos para mostrar los resultados de forma tabular.

- *Análisis estructural*: En esta región se encuentran los procedimientos para verificar que se hayan dado todos los parámetros para los cálculos necesarios y verificar también que solo se ingresen valores válidos en estos campos. Contiene el procedimiento para realizar el cálculo de los correlogramas teóricos y experimentales, así como, para la graficación de ambos correlogramas. También, contiene los procedimientos para modificar los correlogramas teóricos y verificar que solo se ingresen valores validos para estos campos.

- *Estimación*: En esta región se encuentran los procedimientos para identificar si se ha seleccionado realizar estimación o simulación, así como, para crear la malla para estimar/simular y que sea un valor válido. Contiene los procedimientos para modificar los valores de la malla original, verificar que solo sean ingresados valores validos en los campos y que se guarden estos nuevos cambios. También contiene los procedimientos para calcular la estimación y la simulación.

En la clase *Guiado* destacan las regiones:

- *Análisis Estructural*: Esta región incluye los siguientes procedimientos: Procedimiento que invoca función para realizar el análisis de tendencia encontrado en Module1.vb y muestra los resultados. Procedimiento que invoca procedimientos y funciones de Module1.vb para la revisión e interpolación de sondeos. Procedimientos para calcular las funciones de correlación y graficarlas. Procedimiento para modificar la gráfica de la correlación horizontal dependiendo del ancho de clase.

- *Estim/Simulación:* Esta región incluye los siguientes procedimientos: Procedimiento para cargar los datos sondeos a los que se le realizara estimación o simulación. Procedimiento para cargar el archivo de coordenadas de referencia si este ha sido creado. Procedimiento que genera un nuevo formulario para crear los archivos de coordenadas para la estimación o simulación. Rutinas que manejan los eventos que se generan en la sección de Estimación/Simulación. Procedimiento que invoca procedimientos y funciones de Module1.vb para

realizar estimación o simulación, así como, escribir los archivos de resultados de estimación/simulación.

En el módulo *Module1* destacan las regiones:

- *Regresión lineal*: Esta región contiene la función de la regresión lineal.

- *Análisis estructural*: En esta región se incluyen las funciones y rutinas que son invocadas desde la clase *Guiado*; las funciones y procedimientos son: las funciones que interpolan sondeos, la función que revisa la interpolación lineal entre sondeos, la función que calcula las distancias de correlación vertical y horizontal, el procedimiento para generar los archivos de las funciones de correlación y los modelos de correlación horizontal y vertical.

- *Estimación*: En esta región se incluyen las funciones que son invocadas desde la clase *Guiado* para la realizar la estimación.

- *Simulación*: En esta región se incluyen las funciones que son invocadas desde la clase *Guiado* para la realizar la simulación.

- *Lectura y escritura de archivo*: En esta región se encuentran las funciones y procedimientos para la lectura (individual y de grupo de datos de sondeos) y escritura de todos los tipos de archivos utilizados en el programa para el análisis en 3D. Procedimientos para la actualización, así como, para la lectura del encabezado y la actualización de formatos de archivos.

En el módulo *Rutinas* destacan las regiones:

- *Regresión lineal*: Esta región contiene la función de la regresión lineal para el análisis en 2D.

- *Análisis estructural:* Esta región contiene la función para el cálculo de pares de datos del análisis estructural (autocorrelación) y la función para calcular distancias y coeficientes de correlación.

- *Estimación*: Esta región contiene funciones que son invocadas en la región *Estim/Simulación* de la clase *Analisis*. Contiene: La función para el cálculo de correlación de una matriz, la función para calcular el vector B de la matriz de correlación entre un nodo por estimar y el conjunto de datos conocidos.

6.1.5 Funciones y procedimientos

Las funciones y procedimientos más importantes en las regiones anteriormente descritas son:

El procedimiento de *escrituraPlano* es el procedimiento principal para la escritura de los archivos utilizados en el análisis 2D. Este procedimiento escribe los datos de los proyectos en diferentes formatos. Como parámetros recibe: la ubicación del archivo, los

datos del proyecto, tipo de formato; y como parámetros opcionales: los datos de la regresión lineal. El procedimiento se define de la siguiente manera:

```
Public Sub escrituraPlano(ByVal path_name As String, ByVal planoA As plano, ByVal
formato As String, _
                          Optional ByVal f1() As Double = Nothing, _
                          Optional ByVal f2() As Double = Nothing)
        Dim sw As IO.StreamWriter
        Select Case formato
            Case ".po"
                sw = New IO.StreamWriter(path_name.Replace(".po", formato))
            Case ".por"
                sw = New IO.StreamWriter(path_name.Replace(".po", formato))
            Case ".pe"
                sw = New IO.StreamWriter(path_name.Replace(".po", formato))
            Case ".psi"

        Select Case formato
            Case ".po"
                sw.Write(planoA.id.ToUpper)
            Case ".por"
                sw.Write(planoA.id.ToUpper)
            Case ".pe"
                sw.Write(planoA.id.ToUpper)
            Case ".psi"
        End Select
        sw.WriteLine("                        /Nombre del proyecto.")
        sw.Write(planoA.np)
        sw.WriteLine("                                  /Número de datos que
contiene el proyecto.")
        sw.Write(planoA.var)
        sw.WriteLine("                                     /Variable de interés del
proyecto.")
        sw.Write(planoA.unit)
        sw.WriteLine("                                     /Unidad dimensional de la
variable de interés.")
        Select Case formato

            Case ".por"
                sw.WriteLine("/ #  CoordX        CoordY     Residuos    Regresión
" + planoA.var + "[" + planoA.unit + "]")
                For i = 0 To planoA.np - 1
                    sw.WriteLine(String.Format("{0,3:#0}", i + 1) + "," + _
                                 String.Format("{0,10:0.00}", planoA.x(i)) + "," + _
                                 String.Format("{0,10:0.00}", planoA.y(i)) + "," + _
                                 String.Format("{0,10:0.00}", f1(i)) + "," + _
                                 String.Format("{0,13:0.00}", f2(i)) + "," + _
                                 String.Format("{0,13:0.00}", planoA.val(i)))
                Next

            Case ".po"
                sw.WriteLine("/ #  CoordX        CoordY        " + planoA.var + "[" +
planoA.unit + "]")
                For i = 0 To planoA.np - 1
                    sw.WriteLine(String.Format("{0,3:#0}", i + 1) + "," + _
                                 String.Format("{0,10:0.00}", planoA.x(i)) + "," + _
                                 String.Format("{0,10:0.00}", planoA.y(i)) + "," + _
                                 String.Format("{0,13:0.00}", planoA.val(i)))
                Next
            Case ".pe"
```

```
                sw.WriteLine("/ #  CoordX        CoordY     Estimación    Disperción
")
                For i = 0 To planoA.np - 1
                    sw.WriteLine(String.Format("{0,3:#0}", i + 1) + "," + _
                                 String.Format("{0,10:0.00}", planoA.x(i)) + "," + _
                                 String.Format("{0,10:0.00}", planoA.y(i)) + "," + _
                                 String.Format("{0,13:0.00}", planoA.val(i)) + "," +
_
                                 String.Format("{0,10:0.00}", f1(i)))
                Next
        End Select
        sw.Write("/END_FILE")
        sw.Close()
    End Sub
```

La función *regLineal* realiza el cálculo de regresión lineal en 2D, obteniendo los coeficientes (a,b,c) del hiperplano V=ax+by+c. Los parámetros que recibe son: *x*, vector de coordenadas en x; *y*, vector de coordenadas en y; *V*, vector de datos y regresa un vector con los coeficientes a,b,c. La función se define de la siguiente forma.

```
    Public Function regLineal(ByVal x() As Double, ByVal y() As Double, ByVal V() As
Double) As Double()
        Dim xp, yp, Vp, a, b, c As Double
        Dim Kxx, Kyy, Kxy, Cx, Cy As Double
        Dim K As New Matrix(2, 2)
        Dim AB As New Matrix(2, 1)
        Dim CC As New Matrix(2, 1)
        Dim n As Integer = x.Length

        For i = 0 To n - 1
            xp += x()
            yp += y()
            Vp += V()
        Next

        xp /= n
        yp /= n
        Vp /= n
        For i = 0 To n - 1
            Kxx += (x(i) - xp) ^ 2
            Kyy += (y(i) - yp) ^ 2
            Kxy += (x(i) - xp) * (y(i) - yp)
            Cx += (x(i) - xp) * (V(i) - Vp)
            Cy += (y(i) - yp) *
        Next

        Kxx /= n
        Kyy /= n
        Kxy /= n
        Cx /= n
        Cy /= n
        K(0, 0) = Kxx
        K(0, 1) = Kxy
        K(1, 0) = Kxy
        K(1, 1) = Kyy
        CC(0, 0) = Cx
        CC(1, 0) = Cy

        AB = Matrix.Multiply(K.Inverse, C)
        a = AB(0, 0)
        b = AB(1, 0)
```

```
    c = Vp - a * xp - b * yp

    regLineal = Double() {a, b, c}
End Function
```

El procedimiento *estim* se encarga de realizar la estimación a partir de un conjunto de valores conocidos, este procedimiento es utilizado para el análisis en 2D. Este procedimiento revisa que existan todos los datos para crear la malla de puntos a estimar, realiza todo el proceso de estimación y finalmente, muestra los resultados en forma tabular en pantalla, y crea el archivo correspondiente con los resultados. El procedimiento está definido de la siguiente forma.

```
Private Sub estim()
    Dim ALPHA, distcorrel(4), ellipse(2), estim2(500) As Decimal
    Dim PORTEE, angle, vect1(2), invW(500, 500) As Decimal
    Dim distance, Maille(500, 500), x, y, series(500), lambda(500) As Decimal
    Dim estim(500), variance, disp(500, 500), moyenne, teta As Decimal
    Dim t, r, p, l As Integer
    Checar()
    If cambios = False Then
        Try
            If DisMod0TB.Text = vbNullString Then
                distcorrel(0) = CDec(DisCal0TB)
            Else
                distcorrel(0) = CDec(DisMod0TB)
            End If

            If DisMod90TB.Text = vbNullString Then
                distcorrel(2) = CDec(DisCal90TB)
            Else
                distcorrel(2) = CDec(DisMod90TB)
            End If

            If DisMod45TB.Text = vbNullString Then
                distcorrel(1) = CDec(DisCal45TB)
            Else
                distcorrel(1) = CDec(DisMod45TB)
            End If

            If DisMod135TB.Text = vbNullString Then
                distcorrel(3) = CDec(DisCal135TB)
            Else
                distcorrel(3) = CDec(DisMod135TB)
            End If

            'Identificación de la correlación distancia más corta
            p = 0
            For i = 0 To 3    ' cambio aqui 22-05-12
                If distcorrel(i) < distcorrel(p) Then
                    p = i
                End If
            Next

            'Características de la elipse
            If p = 0 Or p = 1 Then
                ellipse(1) = distcorrel(p)
                ellipse(2) = distcorrel(p + 2)
            Else
                ellipse(1) = distcorrel(p)
                ellipse(2) = distcorrel(p - 2)
```

```
                End If

                angle = p * 45
                angle = CDec(angle * 3.14159265 / 180)

                If p <= 2 Then
                    teta = p * 45.0
                Else
                    teta = (p * 45.0) - 90.0
                End If

                distcorrel(p) = (ellipse(1) * ellipse(2)) / Sqrt((ellipse(1) ^ 2 *
(Cos(teta)) ^ 2) + (ellipse(2) ^ 2 * (Sin(teta)) ^ 2))

                For i = 1 To numDatos
                    TAB2(t, 1) = Cos(angle) * TAB(t, 1) + Sin(angle) * TAB(t, 2)
                    TAB2(t, 2) = -Sin(angle) * TAB(t, 1) + Cos(angle) * TAB(t, 2)
                Next

                ReDim Maille(n1(1), n1(2))

                For i = 1 To numDatos
                    For j = 1 To numDatos
                        vect1(1) = Abs(TAB(j, 1) - TAB(i, 1))
                        vect1(2) = Abs(TAB(j, 2) - TAB(i, 2))
                        distance = CDec(Sqrt(vect1(1) * vect1(1) + vect1(2) *
vect1(2)))
                    Next
                Next

                invW=compKrig(numDatos, distance, distcorrel)

                ReDim estim(numDatos + 1)
                ReDim estim2(numDatos + 1)
                ReDim disp(n1(1), n1(2))

                For i = 0 To n1(1)
                    For j = 0 To n1(2)
                        moyenne = 0
                        variance = 0
                        disp(i, j) = 0
                        x = min1 + i * pas
                        y = min2 + j * pas

                        'Creación de vectores
                        For t = 1 To numDatos
                            vect1(1) = Abs(TAB(t, 1) - x)
                            vect1(2) = Abs(TAB(t, 2) - y)

                            If vect1(1) <= 0.001 Then
                                ALPHA = CDec(90 * 3.14159265 / 180)
                            Else
                                ALPHA = CDec(Atan(Abs(vect1(2)) / vect1(1)))
                                If vect1(2) < 0.001 Then
                                    ALPHA = 0
                                End If
                            End If

                            distance = CDec(Sqrt(vect1(1) * vect1(1) + vect1(2) *
vect1(2)))
```

```
                            PORTEE = ellipse(1) * ellipse(2) / (ellipse(2) ^ 2 *
Cos(ALPHA) ^ 2 + ellipse(1) ^ 2 * Sin(ALPHA) ^ 2)

                            estim(t) = Exp(-2 * distance / distcorrel(p))
                        Next

                        estim(numDatos + 1) = 1

                        'Cálculo de coeficientes LAMBDA
                        For t = 1 To numDatos + 1
                            lambda(t) = 0
                            For r = 1 To numDatos + 1
                                lambda(t) = lambda(t) + invW(t, r) * estim(r)
                            Next
                        Next

                        'estimación lineal
                        For t = 1 To numDatos
                            Maille(i, j) = Maille(i, j) + TAB(t, 3) * lambda(t)
                            disp(i, j) = disp(i, j) + lambda(t) * estim(t)
                        Next

                        ProgressBar1.PerformStep()

                        disp(i, j) = -disp(i, j) + 1 + lambda(numDatos + 1)
                    Next
                Next

            Catch ex As Exception
                MsgBox(ex.Message, MsgBoxStyle.Critical, "Error")
            End Try
        End If
    End Sub
```

El procedimiento *simu* calcula la simulación condicional a partir de un conjunto de datos, esta función es utilizada en el análisis en 2D. El procedimiento revisa también que existan todos los datos para crear la malla de puntos donde se simulará, realiza todo el proceso de simulación y finalmente, muestra los resultados en forma tabular, y genera un archivo de resultados. El procedimiento está definido en la forma.

```
Private Sub simu()
        Dim somme, ALPHA, distcorrel(4), ellipse(2), n As Decimal
        Dim estim2(500), angle, vect1(2), W(500, 500), W1(500, 500), invW(500, 500)
As Decimal
        Dim distance, Maille(500, 500), Maille2(500, 500), x, y, series(500) As
Decimal
        Dim lambda(500), estim(500), variance, disp(500, 500), moyenne, LU(500, 500)
As Decimal
        Dim somme2, G(500, 500), Yi(500), Maille3(500, 500) As Decimal
        Dim t, r, p, l As Integer
        Dim aleatoire1, aleatoire2, zeta, teta As Decimal

        Checar()
        If cambios = False Then
            Try
                'cambio de indices 22-05-12
                If DisMod0TB.Text = vbNullString Then
                    distcorrel(0) = CDec(DisCal0TB)
                Else
                    distcorrel(0) = CDec(DisMod0TB)
```

```
                End If

                If DisMod90TB.Text = vbNullString Then
                    distcorrel(2) = CDec(DisCal90TB)
                Else
                    distcorrel(2) = CDec(DisMod90TB)
                End If

                If DisMod45TB.Text = vbNullString Then
                    distcorrel(1) = CDec(DisCal45TB)
                Else
                    distcorrel(1) = CDec(DisMod45TB)
                End If

                If DisMod135TB.Text = vbNullString Then
                    distcorrel(3) = CDec(DisCal135TB)
                Else
                    distcorrel(3) = CDec(DisMod135TB)
                End If

                'Identificación de la distancia de correlación más corta
                p = 0
                For i = 1 To 3
                    If distcorrel(i) < distcorrel(p) Then
                        p = i
                    End If
                Next

                'Características de la elipse
                If p = 0 Or p = 1 Then
                    ellipse(1) = distcorrel(p)
                    ellipse(2) = distcorrel(p + 2)
                Else
                    ellipse(1) = distcorrel(p)
                    ellipse(2) = distcorrel(p - 2)
                End If

                angle = p * 45
                angle = CDec(angle * 3.14159265 / 180)

                If p <= 2 Then
                    teta = p * 45.0
                Else
                    teta = (p * 45.0) - 90.0
                End If

                distcorrel(p) = (ellipse(1) * ellipse(2)) / Sqrt((ellipse(1) ^ 2 *
(Cos(teta)) ^ 2) + (ellipse(2) ^ 2 * (Sin(teta)) ^ 2))

                n = (1 + n1(1)) * (1 + n1(2))

                ReDim Maille(n1(1), n1(2))

                ProgressBar1.Visible = True
                ProgressBar1.Minimum = 1
                ProgressBar1.Maximum = 11
                ProgressBar1.Step = 1
                ProgressBar1.Value = 1

                For i = 1 To numDatos
                    For j = 1 To numDatos
                        vect1(1) = Abs(TAB(j, 1) - TAB(i, 1))
```

```
                        vect1(2) = Abs(TAB(j, 2) - TAB(i, 2))
                        distance = CDec(Sqrt(vect1(1) * vect1(1) + vect1(2) *
vect1(2)))
                    Next
                Next
                invW = compKrig(numDatos, distance, distcorrel)
                ProgressBar1.PerformStep()

                'Creación de la malla
                ReDim estim(numDatos + 1)
                ReDim estim2(numDatos + 1)
                ReDim disp(n1(1), n1(2))
                ReDim TAB2(n, 4)

                l = 1

                For i = 0 To n1(1)
                    For j = 0 To n1(2)
                        moyenne = 0
                        variance = 0
                        disp(i, j) = 0
                        x = min1 + i * pas
                        y = min2 + j * pas

                        'Creacion del vector
                        For t = 1 To numDatos
                            vect1(1) = Abs(TAB(t, 1) - x)
                            vect1(2) = Abs(TAB(t, 2) - y)
                            If vect1(1) <= 0.001 Then
                                ALPHA = CDec(90 * 3.14159265 / 180)
                            Else
                                ALPHA = CDec(Atan(Abs(vect1(2)) / vect1(1)))
                                If vect1(2) < 0.001 Then
                                    ALPHA = 0
                                End If
                            End If

                            distance = CDec(Sqrt(vect1(1) * vect1(1) + vect1(2) *
vect1(2)))

                            estim(t) = Exp(-2 * distance / distcorrel(p))
                        Next

                        estim(numDatos + 1) = 1

                        'Calculo de los coeficientes LAMBDA
                        For t = 1 To numDatos + 1
                            lambda(t) = 0
                            For r = 1 To numDatos + 1
                                lambda(t) = lambda(t) + invW(t, r) * estim(r)
                            Next
                            If t <> numDatos + 1 Then
                                moyenne = moyenne + TAB(t, 3)
                            End If
                        Next

                        moyenne = moyenne / numDatos

                        'Estimación lineal
                        For t = 1 To numDatos
                            Maille(i, j) = Maille(i, j) + TAB(t, 3) * lambda(t)
                            disp(i, j) = disp(i, j) + lambda(t) * estim(t)
```

```
                        variance = variance + (TAB(t, 3) - moyenne) ^ 2
                    Next

                    variance = variance / numDatos
                    TAB2(l, 1) = min1 + (i - 1) * pas
                    TAB2(l, 2) = min2 + (j - 1) * pas
                    TAB2(l, 3) = Maille(i, j)
                    disp(i, j) = -disp(i, j) + 1.0 + lambda(numDatos + 1)
'Modif. 25-05-12
                    l = l + 1
                Next
            Next

            ReDim W(n, n)

            For i = 1 To n
                For j = 1 To n
                    vect1(1) = Abs(TAB2(i, 1) - TAB2(j, 1))
                    vect1(2) = Abs(TAB2(i, 2) - TAB2(j, 2))
                    distance = CDec(Sqrt(vect1(1) * vect1(1) + vect1(2) *
vect1(2)))
                    W(i, j) = Exp(-2 * distance / distcorrel(p))
                Next
            Next

            'CHOLESKY
            ReDim LU(n, n)

            somme = 0
            somme2 = 0
            LU(1, 1) = Sqrt(W(1, 1))

            For i = 2 To n
                LU(i, 1) = W(i, 1) / LU(1, 1)
            Next

            For k = 2 To n - 1
                For i = 1 To k - 1
                    somme = somme + LU(k, i) ^ 2
                Next

                LU(k, k) = Sqrt(W(k, k) - somme)
                somme = 0

                For i = k + 1 To n
                    For j = 1 To k - 1
                        somme2 = somme2 + LU(i, j) * LU(k, j)
                    Next
                    LU(i, k) = (W(i, k) - somme2) / LU(k, k)
                    somme2 = 0
                Next
            Next

            somme = 0
            For i = 1 To n
                somme = somme + LU(n, i) ^ 2
            Next

            LU(n, n) = Sqrt(W(n, n) - somme)
            Randomize()

            ReDim Yi(n)
```

```
                For i = 1 To n
                    aleatoire1 = Rnd()
                    aleatoire2 = Rnd()
                    zeta = Sqrt(-2 * Log(aleatoire1)) * Cos(2 * 3.14159265 *
aleatoire2)
                    Yi(i) = zeta
                Next

                l = 1
                ReDim Maille2(n1(1), n1(2))
                ReDim TAB2(n, 4)

                For i = 0 To n1(1)
                    For j = 0 To n1(2)
                        For k = 1 To l
                            G(i, j) = G(i, j) + LU(l, k) * Yi(k)
                        Next
                        Maille2(i, j) = Maille(i, j) + G(i, j) * disp(i, j)
                        l = l + 1
                    Next
                Next

                'KRIGING SIMULATION
                ReDim TAB2(numDatos, 4)
                moyenne = 0

                For i = 1 To numDatos
                    moyenne = moyenne + TAB(i, 3)
                Next

                moyenne = moyenne / numDatos
                variance = 0

                For i = 1 To numDatos
                    variance = variance + (TAB(i, 3) - moyenne) ^ 2
                Next

                variance = variance / numDatos
                Randomize()

                For i = 1 To numDatos
                    aleatoire1 = Rnd()
                    aleatoire2 = Rnd()
                    zeta = Sqrt(-2 * Log(aleatoire1)) * Cos(2 * 3.14159265 *
aleatoire2)
                    TAB2(i, 3) = moyenne + variance * zeta
                Next

                ReDim Maille3(n1(1), n1(2))
                ReDim W(numDatos + 1, numDatos + 1)

                For i = 1 To numDatos
                    For j = 1 To numDatos
                        vect1(1) = Abs(TAB(i, 1) - TAB(j, 1))
                        vect1(2) = Abs(TAB(j, 2) - TAB(i, 2))
                        distance = CDec(Sqrt(vect1(1) * vect1(1) + vect1(2) *
vect1(2)))
                        W(i, j) = Exp(-2 * distance / distcorrel(p))
                    Next
                Next
```

```
                For i = 1 To numDatos
                    W(numDatos + 1, i) = 1
                    W(i, numDatos + 1) = 1
                Next

                W(numDatos + 1, numDatos + 1) = 0

                invW = compSimu(numDatos, W)


                ReDim estim(numDatos + 1)

                For i = 0 To n1(1)
                    For j = 0 To n1(2)
                        moyenne = 0
                        variance = 0
                        disp(i, j) = 0
                        x = min1 + i * pas
                        y = min2 + j * pas

                        For t = 1 To numDatos
                            vect1(1) = Abs(TAB2(t, 1) - x)
                            vect1(2) = Abs(TAB2(t, 2) - y)
                            distance = CDec(Sqrt(vect1(1) * vect1(1) + vect1(2) *
vect1(2)))
                            estim(t) = Exp(-2 * distance / distcorrel(p))
                        Next

                        estim(numDatos + 1) = 1

                        For t = 1 To numDatos + 1
                            lambda(t) = 0
                            For r = 1 To numDatos + 1
                                lambda(t) = lambda(t) + invW(t, r) * estim(r)
                            Next
                        Next

                        'Estimación lineal
                        For t = 1 To numDatos
                            Maille3(i, j) = Maille3(i, j) + TAB2(t, 3) * lambda(t)
                        Next
                    Next
                Next

            Catch ex As Exception
                MsgBox(ex.Message, MsgBoxStyle.Critical, "Error")
            End Try
        End If
    End Sub
```

El procedimiento *escrituraSondeo* es el procedimiento principal para escribir los archivos utilizados en el análisis en 3D. Este procedimiento escribe los datos de sondeos en diferentes formatos. Como parámetros recibe: la ubicación del archivo, los datos de sondeos, formato del archivo, y como parámetros opcionales: los datos de otro sondeo. El procedimiento se define de la siguiente manera:

```
    Public Sub escrituraSondeo(ByVal path_name As String, ByVal sondeoA As sondeo,
ByVal formato As String, _
                               Optional ByVal sondeoB As sondeo = Nothing, _
```

```
                                Optional ByVal f1() As Double = Nothing, _
                                Optional ByVal f2() As Double = Nothing, _
                                Optional ByVal f12() As Double = Nothing)
        Dim sw As IO.StreamWriter

        Select Case formato
            Case ".so"
                sw = New IO.StreamWriter(path_name.Replace(".so", formato))
            Case ".sor"
                sw = New IO.StreamWriter(path_name.Replace(".so", formato))
            Case ".sce"
                sw = New IO.StreamWriter(path_name.Replace(".so", formato))
            Case ".si"
                sw = New IO.StreamWriter(path_name.Replace(".so", formato))
            Case ".sir"
                sw = New IO.StreamWriter(path_name.Replace(".so", formato))
            Case ".fcv"
                sw = New IO.StreamWriter(path_name.Replace(".si", formato))
            Case ".fch"
                sw = New IO.StreamWriter(path_name.Replace(".si", formato))
            Case ".cev"
                sw = New IO.StreamWriter(path_name.Replace(".si", formato))
            Case ".ceh"
                sw = New IO.StreamWriter(path_name.Replace(".si", formato))
        End Select

        'Linea 1   /Identificador, clave o nombre del sondeo.
        Select Case formato
            Case ".so"
                sw.Write(sondeoA.id.ToUpper) '+ " - SONDEO ORIGINAL")
            Case ".sor"
                sw.Write(sondeoA.id.ToUpper) ' + " - SONDEO ORIGINAL RESIDUAL")
            Case ".sce"
                sw.Write(sondeoA.id.ToUpper) ' + " - CON CORRECCION DE ELEVACIONES")
            Case ".si"
                sw.Write(sondeoA.id.ToUpper) ' + " - SONDEO INTERPOLADO")
            Case ".sir"
                sw.Write(sondeoA.id.ToUpper) ' + " - SONDEO INTERPOLADO RESIDUAL")
            Case ".fcv"
                sw.Write(sondeoA.id.ToUpper) ' + " - FUNCION DE CORRELACION
VERTICAL")
            Case ".fch"
                sw.Write(sondeoA.id.ToUpper) ' + " - FUNCION DE CORRELACION
HORIZONTAL")
            Case ".cev"
                sw.Write(sondeoA.id.ToUpper) ' + " - FUNCION DE CORRELACION MODELO
VERTICAL")
            Case ".ceh"
                sw.Write(sondeoA.id.ToUpper) ' + " - FUNCION DE CORRELACION MODELO
HORIZONTAL")
        End Select
        sw.WriteLine("                          /Nombre del proyecto.")
        sw.Write(String.Format("{0,0:0.000}", sondeoA.x) + "," +
String.Format("{0,0:0.0000}", sondeoA.y))
        If formato = ".si" Or formato = ".sir" Then
            sw.Write("," + String.Format("{0,0:0.000}", sondeoB.x) + "," +
String.Format("{0,0:0.000}", sondeoB.y))
        End If
        sw.WriteLine("                           /Posición en planta (x,y) del
sondeo.") 'ínicar la comentario en 40
        sw.Write(sondeoA.np)
```

```
        sw.WriteLine("                                      /Número de datos que
contiene el sondeo.")
        sw.Write(sondeoA.var)
        sw.WriteLine("                                        /Variable de interés del
sondeo.")
        sw.Write(sondeoA.unit)
        sw.WriteLine("                                        /Unidad dimensional de la
variable de interés.")
        sw.Write(String.Format("{0,0:0.00}", sondeoA.elev) + "," +
String.Format("{0,0:0.00}", sondeoA.corrElev))
        sw.WriteLine("                           /Elevación de brocal y valor actual
de corrección topográfica.")
        sw.Write(sondeoA.broc)
        sw.WriteLine("                                    /Corrección
topográfica:[True=Si];[False=No]")
        sw.Write(sondeoA.disTipica)
        sw.WriteLine("                                     /Paso de cálculo
vertical.")
        'Línea 9   /Coeficientes de la ec. del hiperplano para retirar tendencia
        sw.Write(CStr(Format(sondeoA.coefHip(0), "0.000")) + "," +
CStr(Format(sondeoA.coefHip(), "0.000")) + "," + CStr(Format(sondeoA.coefHip(),
"0.000")) + "," + CStr(Format(sondeoA.coefHip(), "0.000")))
        sw.WriteLine("                                    /Coeficientes  de regresión
lineal.")
        Select Case format
            Case ".sce"
                'Línea 13, encabezado de los datos: / n  Profundidad      valor
                sw.WriteLine("/ #  Profundidad     " + sondeoA.var + "[" +
sondeoA.unit + "]")
                For i = 0 To sondeoA.np - 1
                    sw.WriteLine(String.Format("{0,3:#0}", i + 1) + "," + _
                                 String.Format("{0,10:0.000}", sondeoA.z(i)) + "," +
_
                                 String.Format("{0,13:0.000}", sondeoA.val(i)))
                Next

            Case ".fcv"
                'Línea 13, encabezado de los datos: / n  Profundidad      valor
                sw.WriteLine("/ #    DistCorr    Coef. de correlación")
                For i = 0 To f1.Length - 1
                    sw.WriteLine(String.Format("{0,3:#0}", i + 1) + "," + _
                                 String.Format("{0,10:0.000}", sondeoA.disTipica * i)
+ "," + _
                                 String.Format("{0,10:0.000}", f1(i)))
                Next
            Case ".fch"
                'Línea 13, encabezado de los datos: / n  Profundidad      valor
                sw.WriteLine("/ #    DistCorr    Coef. de correlación")
                For i = 0 To f1.Length - 1
                    sw.WriteLine(String.Format("{0,3:#0}", i + 1) + "," + _
                                 String.Format("{0,10:0.000}", f1(i)) + "," + _
                                 String.Format("{0,13:0.000}", f2(i)))
                Next
            Case ".si"
                'Línea 13, encabezado de los datos: / n  Profundidad      valor
                sw.WriteLine("/ #  Profundidad     " + sondeoA.var + "[" +
sondeoA.unit + "]" + "      " + sondeoB.var + "[" + sondeoA.unit + "]")
                For i = 0 To sondeoA.np - 1
                    sw.WriteLine(String.Format("{0,3:#0}", i + 1) + "," + _
                                 String.Format("{0,10:0.000}", sondeoA.z(i)) + "," +
_
```

```
                                String.Format("{0,13:0.000}", sondeoA.val(i)) + ","
+ _
                                String.Format("{0,13:0.000}", sondeoB.val(i)))
                Next
            Case ".sor"
                'Línea 13, encabezado de los datos: / n  Profundidad      valor
                sw.WriteLine("/ #  Profundidad     " + sondeoA.var + "[" +
sondeoA.unit + "]")
                For i = 0 To sondeoA.np - 1
                    sw.WriteLine(String.Format("{0,3:#0}", i + 1) + "," + _
                                String.Format("{0,10:0.000}", sondeoA.z(i)) + "," +
_
                                String.Format("{0,13:0.000}", sondeoA.val(i)))
                Next

            Case ".so"
                'Línea 13, encabezado de los datos: / n  Profundidad      valor
                sw.WriteLine("/ #  Profundidad     " + sondeoA.var + "[" +
sondeoA.unit + "]")
                For i = 0 To sondeoA.np - 1
                    sw.WriteLine(String.Format("{0,3:#0}", i + 1) + "," + _
                                String.Format("{0,10:0.000}", sondeoA.z(i)) + "," +
_
                                String.Format("{0,13:0.000}", sondeoA.val(i)))
                Next
            Case ".cev"
                'Línea 13, encabezado de los datos: / n  Profundidad      valor
                sw.WriteLine("/ #  Dist Corr      Coef. de correlación")
                For i = 0 To sondeoA.val.Length - 1
                    sw.WriteLine(String.Format("{0,3:#0}", i + 1) + "," + _
                                String.Format("{0,10:0.000}", sondeoA.disTipica * i)
+ "," + _
                                String.Format("{0,13:0.000}", sondeoA.val(i)))
                Next
            Case ".ceh"
                'Línea 13, encabezado de los datos: / n  Profundidad      valor
                sw.WriteLine("/ #  Dist Corr      Coef. de correlación")
                For i = 0 To sondeoA.val.Length - 1
                    sw.WriteLine(String.Format("{0,3:#0}", i + 1) + "," + _
                                String.Format("{0,10:0.000}", sondeoA.z(i)) + "," +
_
                                String.Format("{0,13:0.000}", sondeoA.val(i)))
                Next
        End Select
        sw.Write("/END_FILE")
        sw.Close()
    End Sub
```

En este caso la función *regLineal* realiza el cálculo de regresión lineal en 3D y obtiene los coeficientes (a,b,c,d) del hiperplano V=ax+by+cz+d. Los parámetros que recibe son: *x*, vector de coordenadas en x; *y*, vector de coordenadas en y; *z*, vector de coordenadas en z; *V*, vector de datos y regresa un vector con los coeficientes a,b,c,d. La función se define de la siguiente manera.

```
    Public Function regLineal(ByVal x() As Double, ByVal y() As Double, ByVal z() As
Double, ByVal V() As Double) As Double()
        Dim Kxx, Kyy, Kzz As Double
        Dim Kxy, Kxz, Kyz As Double
        Dim Cx, Cy, Cz As Double
        Dim K As New Matrix(3, 3)
```

```
    Dim AB As New Matrix(3, 1)
    Dim CC As New Matrix(3, 1)
    Dim n As Integer = x.Length
    Dim xp, yp, zp, Vp As Double
    Dim a, b, c, d As Double

    For i = 0 To n - 1
        xp += x()
        yp += y()
        zp += z()
        Vp += V()
    Next
    xp /= n
    yp /= n
    zp /= n
    Vp /= n

    For i = 0 To n - 1
        Kxx += (x(i) - xp) ^ 2
        Kyy += (y(i) - yp) ^ 2
        Kzz += (z(i) - zp) ^ 2
        Kxy += (x(i) - xp) * (y(i) - yp)
        Kxz += (x(i) - xp) * (z(i) - zp)
        Kyz += (y(i) - yp) * (z(i) - zp)
        Cx += (x(i) - xp) * (V(i) - Vp)
        Cy += (y(i) - yp) * (V(i) - Vp)
        Cz += (z(i) - zp) * (V(i) - Vp)
    Next
    Kxx /= n
    Kyy /= n
    Kzz /= n
    Kxy /= n
    Kxz /= n
    Kyz /= n
    Cx /= n
    Cy /= n
    Cz /= n

    K(0, 0) = Kxx
    K(0, 1) = Kxy
    K(0, 2) = Kxz
    K(1, 0) = Kxy
    K(1, 1) = Kyy
    K(1, 2) = Kyz
    K(2, 0) = Kxz
    K(2, 1) = Kyz
    K(2, 2) = Kzz

    CC(0, 0) = Cx
    CC(1, 0) = Cy
    CC(2, 0) = Cz

    AB = Matrix.Multiply(K.Inverse, CC)
    a = AB(0, 0)
    b = AB(1, 0)
    c = AB(2, 0)
    d = Vp - a * xp - b * yp - c *
    regLineal = New Double() {a, b, c, d}
End Function
```

La función *funcionesCorrelacion* calcula la función de correlación y genera los archivos de las funciones de correlación, así también genera los modelos de correlación horizontal y vertical. La función se define de la siguiente manera:

```
    Function funcionesCorrelacion() As Double()
        Dim sondeosI As sondeo()
        Dim val2()() As Double
        Dim x2(), y2() As Double
        Dim i, k, j, ji, jj As Integer
        Dim nSond, nSondeos, lastI As Integer
        Dim numSuma() As Integer
        Dim temp As String
        Dim corrVert(0), corrHorz(0), disCorrH(0), disCorrV(0) As Double
        Dim ind() As String
        Dim distCorH, clase, distCorV, prom As Double

        nSondeos = nombSondeos.Length
        ReDim sondeosI(nSondeos - 1)

        ReDim val2(nSondeos - 1)
        ReDim x2(nSondeos - 1)
        ReDim y2(nSondeos - 1)

        For i = 0 To nSondeos - 1
            sondeosI(i) = leerSondeo(nombSondeos(i), ".si", val2(i), x2(i), y2(i))
            sondeosI(i).nomArchivo = nombSondeos(i)
        Next

        nSond = sondeosI.Length
        ReDim dist(nSond - 1)
        ReDim f12(nSond - 1)

        For i = 0 To nSond - 1
            lastI = sondeosI(i).nomArchivo.LastIndexOf("_") + 1
            temp = sondeosI(i).nomArchivo.Substring(lastI,
sondeosI(i).nomArchivo.Length - lastI).Replace(".si", "")
            ind = temp.Split(New Char() {"-"}, StringSplitOptions.RemoveEmptyEntries)
            ji = Val(ind(0))
            jj = Val(ind(1))
            If ji = jj Then
                Console.WriteLine(temp)
                f1 = CorrelacionExperimental(sondeosI(i).val)
                sondeosI(i).np = f1.Length
                escrituraSondeo(sondeosI(i).nomArchivo, sondeosI(i), ".fcv", , f1)
                If f1.Length > corrVert.Length Then
                    ReDim Preserve corrVert(f1.Length - 1)
                    ReDim Preserve disCorrV(f1.Length - 1)
                    ReDim Preserve numSuma(f1.Length - 1)
                End If

                For m = 0 To f1.Length - 1
                    corrVert(m) += f1(m)
                    numSuma(m) += 1
                Next
            Else
                Console.WriteLine(temp)
                f12(k) = FuncionCorrExp(sondeosI(i).val, val2(i))(0)
                dist(k) = Sqrt((x2(i) - sondeosI(i).x) ^ 2 + (y2(i) - sondeosI(i).y)
^ 2)
                k += 1
            End If
```

```
        Next
        ReDim Preserve dist(k - 1)
        ReDim Preserve f12(k - 1)

        sondeosI(0).np = f12.Length
        escrituraSondeo(sondeosI(0).nomArchivo, sondeosI(0), ".fch", , dist, f12)

        'calculo del correlograma experimental vetical
        'se obtiene como el promedio de las funciones de correlacion vertical (.fcv)
        For i = 0 To corrVert.Length - 1
            corrVert(i) /= numSuma(i)
        Next
        sondeosI(0).val = corrVert
        sondeosI(0).np = corrVert.Length
        escrituraSondeo(sondeosI(0).nomArchivo, sondeosI(0), ".cev")

        For i = 0 To disCorrV.Length - 1
            disCorrV(i) = i * sondeosI(0).disTipica
        Next

        'calculo del correlograma experimental horizontal
        'se obtiene como definiendo un ancho de clase y los promedios de las
correlaciones resprectivas

        Array.Sort(dist, f12)
        clase = dist.Max / (14 - guiado.anchoClaseTrB.Value)
        guiado.clase_LB.Text = Round(clase, 1)

        k = 0
        j = 0
        i = 0
        Do
            ReDim Preserve corrHorz(k + 1)
            ReDim Preserve disCorrH(k + 1)
            Do While dist(i) <= clase * (k + 1)
                prom += f12(i)
                i += 1
                j += 1
                If i = dist.Length Then Exit Do
            Loop
            prom /= j
            If j = 0 Then prom = 0

            disCorrH(k + 1) = clase * 0.5 + clase * k
            corrHorz(k + 1) = prom
            prom = 0
            j = 0
            k += 1
        Loop While i < f12.Length

        disCorrH(0) = 0
        corrHorz(0) = 1
        sondeosI(0).iniciarCaptura(k + 1)
        sondeosI(0).val = corrHorz
        sondeosI(0).z = disCorrH
        sondeosI(0).np = k + 1

        escrituraSondeo(sondeosI(0).nomArchivo, sondeosI(0), ".ceh")
        distCorH = distanciasCorrelacion(sondeosI(0).nomArchivo.Replace(".si",
".ceh"))
        distCorV = distanciasCorrelacion(sondeosI(0).nomArchivo.Replace(".si",
".cev"))
```

```
        funcionesCorrelacion = {distCorH, clase, distCorV, disCorrH.Max,
disCorrV.Max}
    End Function
```

La función *generaEstimacion* realiza la estimación en los nodos de la malla de estimación, a partir de un conjunto de los datos, esta función se utiliza en el análisis en 3D. Como parámetros de entrada recibe: un grupo de datos de sondeos, las coordenadas del punto o malla de estimación (estos dos parámetros son estructuras creadas por los programadores) y dos distancias de correlación. La función está definida de la siguiente forma:

```
    Public Function generaEstimacion(ByVal sondeos() As sondeo, ByVal coordEsti() As
punto, _   ByVal distH As Double, ByVal distV As Double)
        Dim numSond As Integer = sondeos.Length 'numero de sondeos
        Dim numRows As Integer
        Dim k, numsOrig As Integer
        Dim est, desv As Double
        Dim matB(), matX() As Double
        Dim nodosMalla As Integer = coordEsti.Length 'numero de nodos de la malla

        ReDim estimacion(nodosMalla - 1)
        ReDim desvEst(nodosMalla - 1)
        coordEstim = coordEsti

        For i = 0 To numSond - 1
            numRows += sondeos(i).np
            ReDim Preserve sondOrig(numRows - 1)
            For j = 0 To sondeos(i).np - 1
                sondOrig(k) = sondeos(i).val(j)
                k += 1
            Next
        Next

        var = VarianzaDatos(sondeos)
        matA = MatrizCorrelacion(distH, distV, sondeos, coords) 'crea matriz de
correlacion
        numsOrig = Math.Sqrt(matA.Length) ' numero de datos originales
        ReDim matAInv(numsOrig - 1, numsOrig - 1)
        ReDim matX(numsOrig - 1)
        Array.Copy(matA, matAInv, matA.Length)
        alglib.rmatrixinverse(matAInv, info, rep) 'inversa de la matriz de
correlacion

        For i = 0 To nodosMalla - 1
            matB = VecBCorr_nodo_conocidos(distH, distV, coords, coordEstim(i))
            alglib.rmatrixmv(numsOrig, numsOrig, matAInv, 0, 0, 0, matB, 0, matX, 0)
            est = 0
            desv = 0

            For j = 0 To numsOrig - 2
                est += sondOrig(j) * matX(j)
                desv += matB(j) * matX(j)
            Next
            estimacion(i) = est
            desvEst(i) = Math.Sqrt((1 - desv - matX(numsOrig - 1)) * var)
        Next
        generaEstimacion = est
    End Function
```

La función *GeneraSimulacion* es la función que realiza la simulación en los nodos de una malla, a partir de un conjunto de datos, esta función es utilizada en el análisis en 3D. Como parámetros de entrada recibe: un grupo de datos de sondeos, las coordenadas del punto o malla de simulación (estos dos parámetros son estructuras creadas por los programadores) y dos distancias de correlación. La función se define en la siguiente forma:

```
    Public Function generaSimulacion(ByVal sondeos() As sondeo, ByVal coordEsti() As
punto, _   ByVal distH As Double, ByVal distV As Double) As Double()
        Dim matCoorSimu(,) As Double
        Dim m As Integer
        Dim vectZ(), vectG(), Vx(), Vxx(), Vxxx() As Double
        Dim suma As Double
        Dim rnd As New Random()

        generaEstimacion(sondeos, coordEsti, distH, distV)
        matCoorSimu = matrizCorrelacion(distH, distV, coordEstim)
        m = coordEstim.Length

        'vectZ y vectG deben de ser del tamaño de la matriz rho de correlacion = m
        ReDim vectG(m - 1)
        ReDim simulacionIncond(m - 1)
        ReDim Vx(m - 1)
        ReDim Vxx(m - 1)
        ReDim Vc(m - 1)

        alglib.spdmatrixcholesky(matCoorSimu, m, True)
        vectZ = VectorZ(m, rnd.Next)

        For i = 0 To m - 1
            suma = 0
            For j = 0 To i
                suma += matCoorSimu(j, i) * vectZ(j)
            Next
            vectG(i) = suma
        Next

        'estimacion(i) -----> V*(X)
        'simulacionIncond --->  V**(X)
        For i = 0 To m - 1
            simulacionIncond(i) = estimacion(i) + desvEst(i) * vectG(i)
        Next
        Array.Copy(estimacion, Vx, m)
        Array.Copy(simulacionIncond, Vxx, m)
        Vxxx = generaEstimacion(simulacionIncond, coordEsti, distH, distV)

        'Vc=Vx+[Vxx-Vxxx]
        For i = 0 To m - 1
            Vc(i) = Vx(i) + Vxx(i) - Vxxx(i)
        Next

        generaSimulacion = Vc
    End Function
```

7 Ejemplos para la validación del programa con aplicación en Geotecnia

En este capítulo se presentan un conjunto de aplicaciones prácticas que ilustran la aplicación del software desarrollado (MESCAL). A través de estos ejemplos prácticos, también se muestra la aplicación de la Geoestadística en la geotecnia. Asimismo, se evalúa el beneficio de utilizar el software desarrollado como herramienta cotidiana para el análisis de parámetros geotécnicos. Para la validación, se elaboraron algunos ejemplos con el software desarrollado y se compararon los resultados obtenidos con los resultados obtenidos con otras herramientas como son *Microsoft Excel* (Parámetros estadísticos), módulos de la librería llamada *GSLIB* (Geostatistic Software Library; Deutch, 1992) y programa SAAG (Sistema de Ayuda al Análisis Geoestadístico; Auvinet *et al*, 2002).

Los ejemplos que se presentan corresponden a:

- Análisis de la distribución espacial de la profundidad del límite superior de la Capa Dura (2D).
  La secuencia estratigráfica superficial típica del subsuelo de la zona lacustre en la Ciudad de México incluye (Marsal y Mazari, 1959): una costra superficial seca delgada llamada Costra Superficial (CS), un estrato de arcilla de varias decenas de metros de espesor llamada Formación Arcillosa Superior (FAS), una primera capa resistente llamada Capa Dura (CD), un segundo estrato de arcilla llamado Formación Arcillosa Inferior (FAI) y un estrato resistente denominado Depósitos profundos (DP).
  Una de las capas que más interesa conocer su configuración superficial es la CD, puesto que, por ser esta una capa de suelo resistente sirve para apoyar las cimentaciones profundas (pilas y pilotes) que generalmente son usadas como solución de cimentación para edificios altos y algunas obras importantes, como es el caso de la Línea 12 del Metro.

- Análisis de la distribución espacial del contenido de agua (3D).
  Para ilustrar el análisis de distribución espacial de las propiedades geotécnicas del subsuelo en un volumen de suelo (3D), se emplean los datos del contenido de agua tomados de los sondeos geotécnicos contenidos en el Sistema de Información Geográfica para Sondeos Geotécnicos, SIG-SG (Laboratorio de Geoinformática, Instituto de Ingeniería UNAM, 2010). El contenido de agua, w (%), es la propiedad índice de los suelos que más destaca, especialmente para materiales cohesivos (arcilla blanda), debido a las correlaciones que presenta con el tipo de material y con las propiedades mecánicas. Así como también, es la propiedad que se mide experimentalmente en un alto número de ensayes de laboratorio.

Los ejemplos se elaboraron siguiendo la metodología propuesta en el apartado 2.3.

7.1 Análisis de la distribución espacial de la profundidad del límite superior de la Capa Dura (2D)

7.1.1 Definición del campo

Los valores de la profundidad de la frontera superior de la CD se considera un campo aleatorio $V(X)$, distribuidos dentro de un espacio R^p, con $p = 2$ (área, 2D). El conjunto de valores medidos dentro del dominio R^p, constituye una muestra de ese campo aleatorio. Para este análisis se consideraron los datos de 88 sondeos geotécnicos distribuidos en forma irregular (aleatoria).

El ejemplo se inició estableciendo un sistema de referencia cartesiano para ubicar los datos experimentales. En la *figura 7.1* se presenta la ubicación en planta generado en MESCAL y en la *figura 7.2* se muestra la ubicación en planta generada en *Excel*, que fue el programa utilizado para validar esta primera fase del análisis.

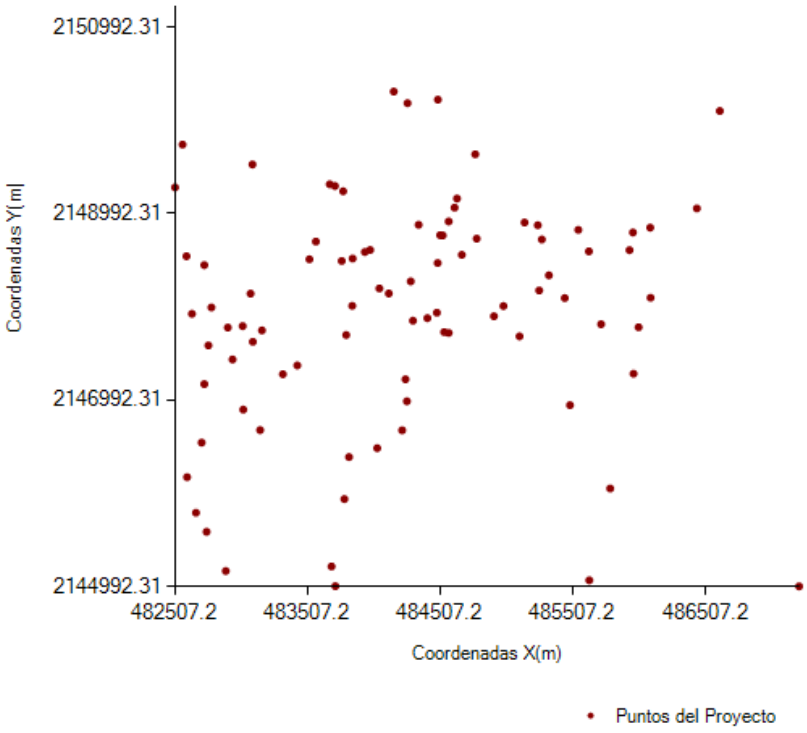


Figura 7.1 Mapa de ubicación de los datos (MESCAL).

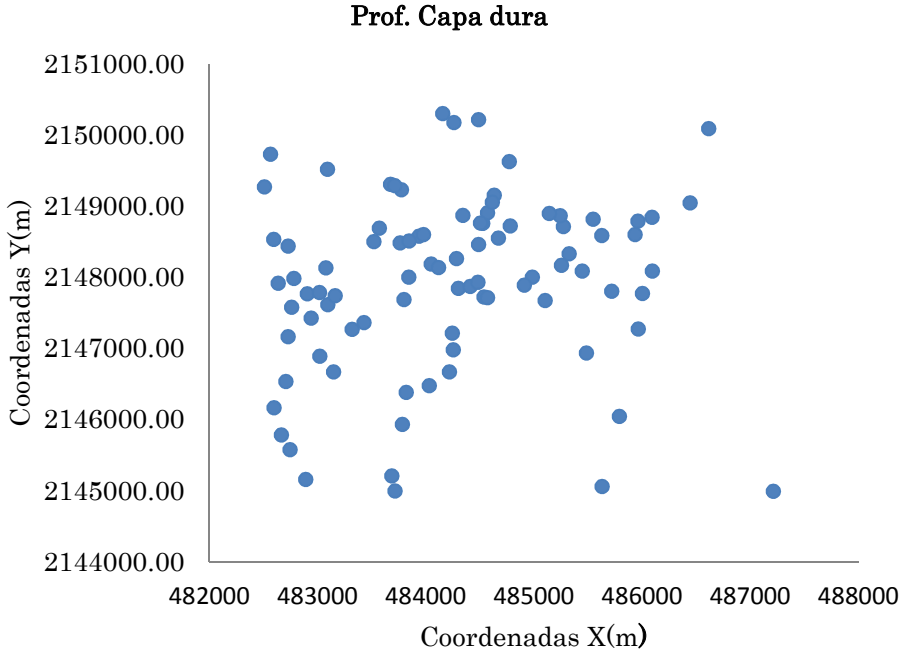


Figura 7.2 Mapa de ubicación de los datos (Excel).

7.1.2 Descripción estadística

Los principales parámetros estadísticos de la profundidad de la capa dura estimados con MESCAL y *Excel* se presentan en la *tabla 7.1*. Así también, en las *figuras 7.3 y 7.4* se presentan los histogramas de frecuencias de las profundidades de la capa dura obtenidos con las mismas herramientas.

Tabla 7.1 Parámetros estadísticos de la profundidad de la CD.

Parámetros	**Valor**	
	MESCAL	**Excel**
Media	-31.904	-31.9045455
Desviación estándar	3.804	3.80362529
Varianza de la muestra	14.468	14.4675653
Número de datos	88	88

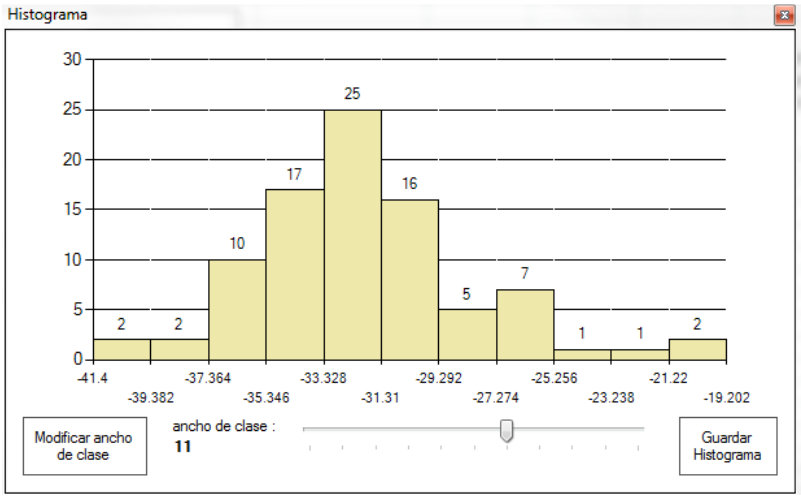


Figura 7.3 Histograma de la profundidad de la CD (MESCAL).

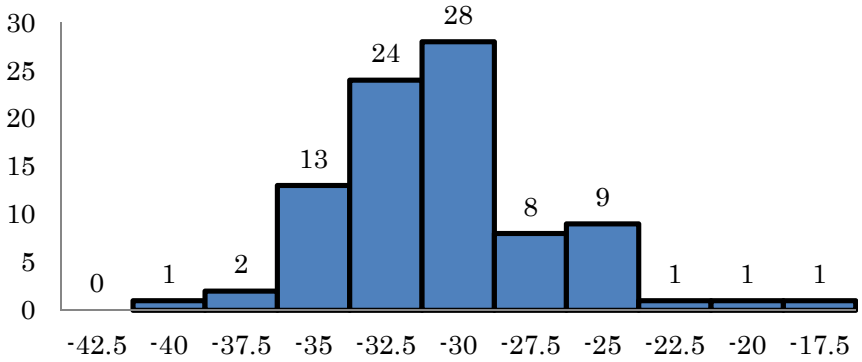


Figura 7.4 Histograma de la profundidad de la CD (Excel).

7.1.3 Análisis de tendencia

El siguiente pasó de la metodología es realizar un análisis de tendencia, a fin de evaluar si el campo aleatorio estudiado es estacionario o no estacionario. La tendencia se evalúa realizando un análisis de regresión lineal buscando ajustar un plano. Para validar los resultados de regresión lineal, obtenidos por el programa MESCAL, se utilizó la herramienta *Solver* de *Excel*. En el Anexo 2, se presentan las tablas con los resultados (regresión y residuos) obtenidos con las dos herramientas. Los coeficientes de regresión lineal de la profundidad de la CD se presentan en la *tabla 7.2*.

Tabla 7.2 Resultados de la Regresión Lineal.

Coeficientes de regresión lineal	**Valor**	
	MESCAL	**Excel**
a	-0.00207	-0.00207226
b	0.00109	0.00109485
c	-1380.11634	-1380.11634

7.1.4 Análisis estructural

El siguiente paso de la metodología es realizar el análisis estructural con el propósito de definir los modelos de correlación espacial. A partir de los datos experimentales, se calculan los correlogramas experimentales direccionales para el campo analizado (profundidad de la CD). Para realizar este cálculo, la herramienta MESCAL requiere como parámetros: paso de cálculo, tolerancia lineal, ancho de banda, tolerancia angular y número de direcciones. Para el ejemplo, se consideraron los siguientes valores:

paso de cálculo = 250 m
tolerancia lineal = 125 m
ancho de banda = 125 m
tolerancia angular = 22.5 m
número de direcciones = 4

Los correlogramas experimentales direccionales del campo generados con la herramienta MESCAL se muestran en la *figura 7.5*.

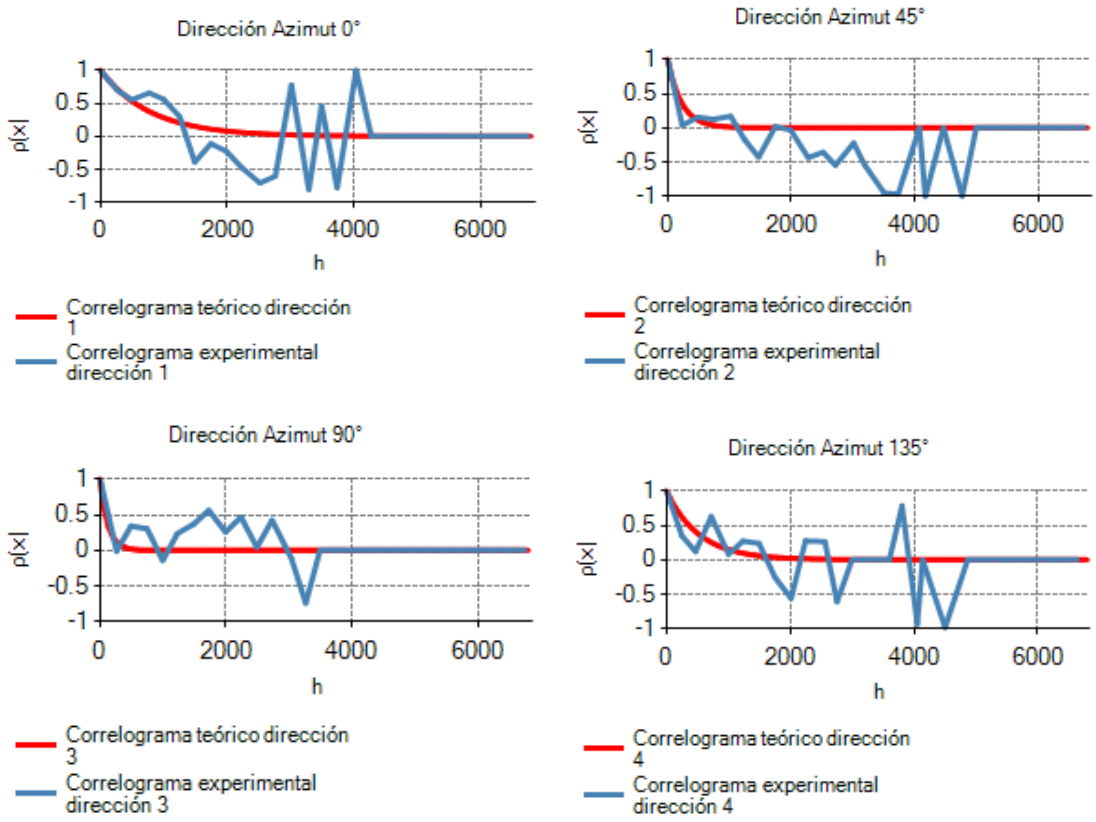


Figura 7.5 Correlogramas experimentales direccionales (MESCAL).

A partir de los correlogramas direccionales experimentales se estiman las distancias de correlación espacial mostradas en la *tabla 7.3*.

Tabla 7.3 Distancias de correlación.

Dirección	Distancia de correlación, δ(m)
0°	1570.93
90°	247.07
45°	449.61
135°	1052.86

Para validar el análisis estructural se utilizó el programa *GAMV2* que es una librería de *GSLIB*. Para la ejecución de este programa se requiere contar con un archivo de datos, el cual contiene el nombre del archivo con los datos experimentales y los parámetros de cálculo. Para el ejemplo, se utilizan los valores antes indicados. Al concluir el cálculo se genera un archivo de resultados, estos se grafican en *Excel* para obtener los correlogramas experimentales direccionales correspondientes ilustrados en la *figura 7.6*. Así también, a partir de los correlogramas experimentales se estiman las distancias de correlación espacial que se muestran en la *tabla 7.4*.

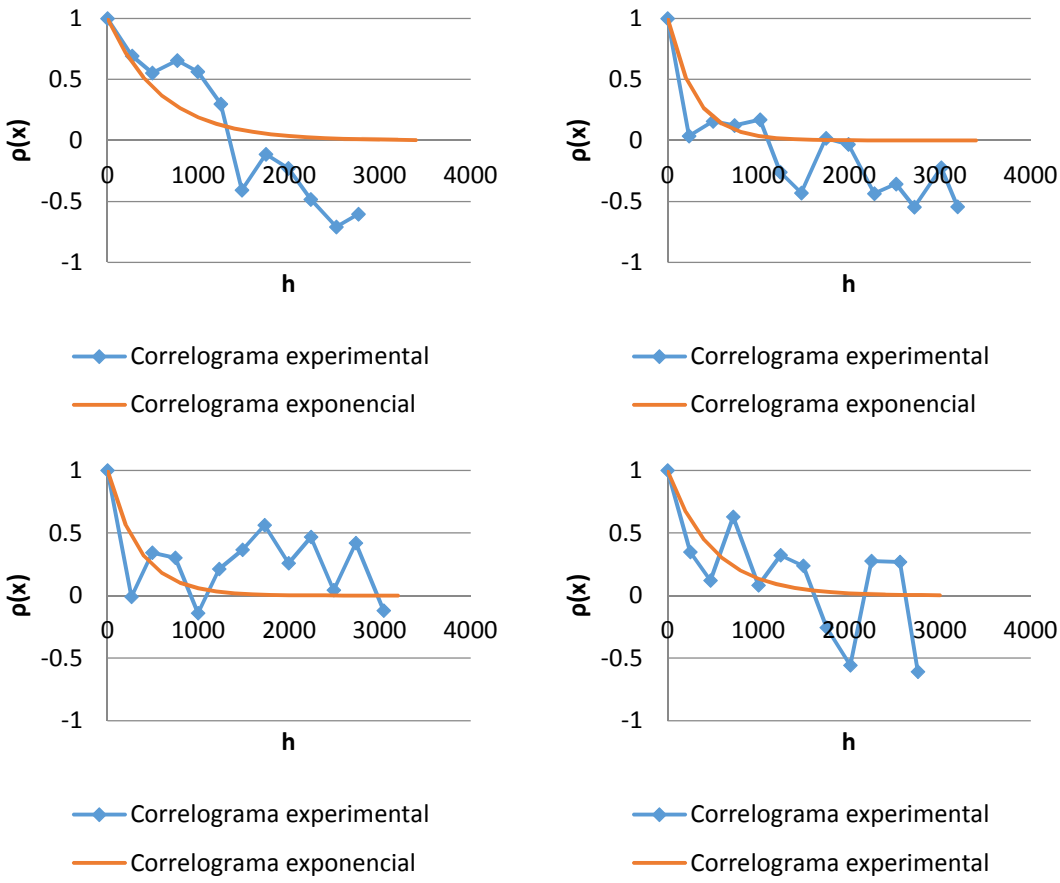


Figura 7.6 Correlogramas experimentales direccionales (Excel).

Tabla 7.4 Distancias de correlación.

Dirección	Distancia de correlación, δ(m)
0°	1200
90°	700
45°	600
135°	1000

Las distancias de correlación anteriormente definidas, permiten construir las elipses de anisotropía mostradas en las *figuras 7.7* a *7.9*. El grado de anisotropía del campo aleatorio puede definirse a través de un factor de anisotropía. Este factor se expresa como el cociente de la distancia de correlación menor entre la distancia de correlación mayor, pueden tenerse dos alternativas, la primera es considerando únicamente dos direcciones perpendiculares y la segunda considerando las cuatro direcciones en forma simultánea.

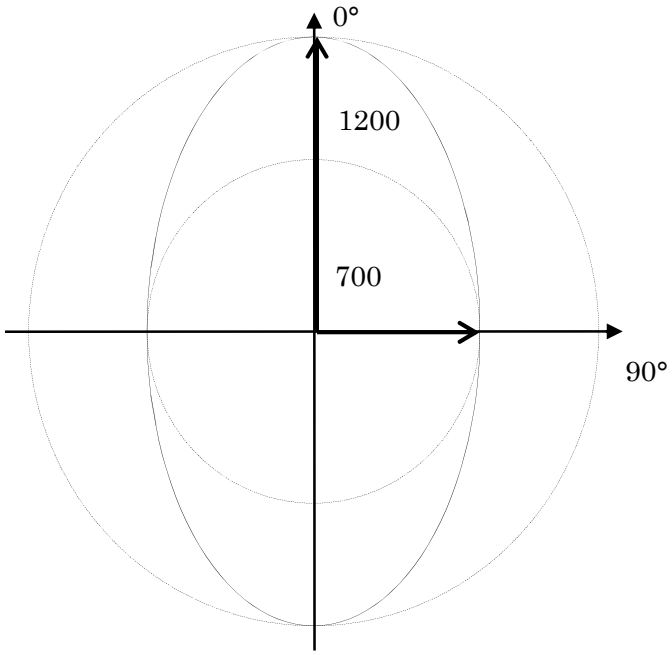


Figura 7.7 Elipses de anisotropía con direcciones 0° y 90°.

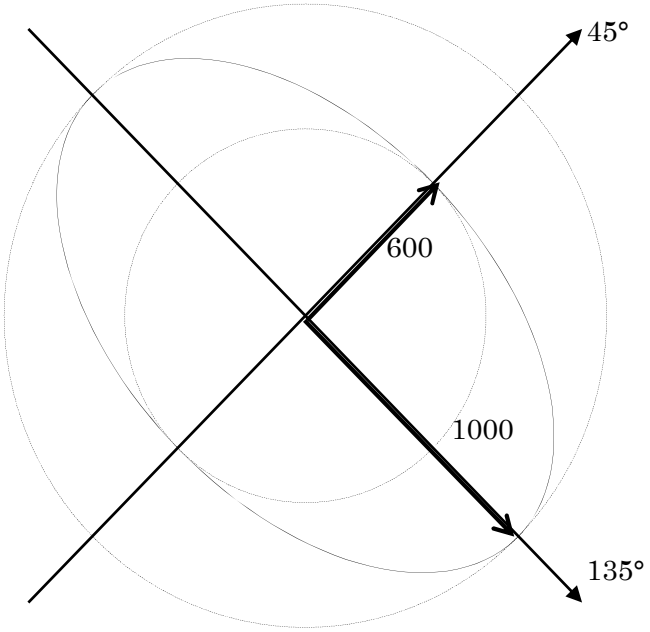


Figura 7.8 Elipses de anisotropía con direcciones 45° y 135°.

0°

45°

90°

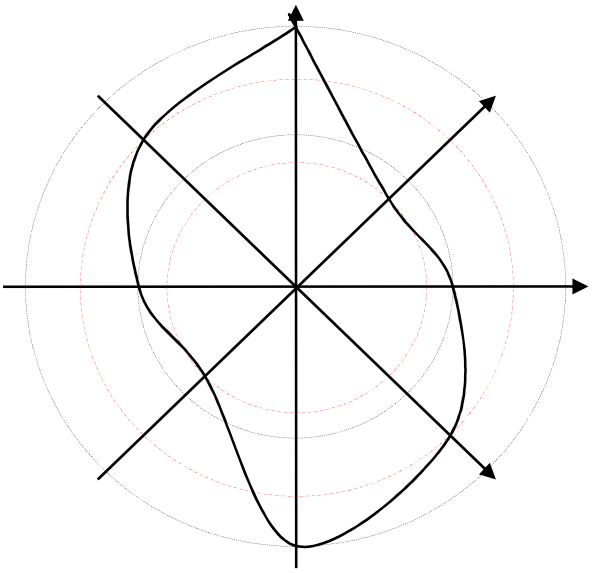


Figura 7.9 Elipses de anisotropía de las profundidades de la capa dura.

Con base en las elipses de anisotropía se obtienen los coeficientes de anisotropía mostrados en la *tabla 7.5*.

Tabla 7.5 Coeficientes de anisotropía.

Dirección	δ(m)	C.A.
0°	1200	0.58333333
90°	700	
45°	600	0.6
135°	1000	

Debe señalarse que este último proceso de la obtención de las elipses de anisotropía y definición de los coeficientes de anisotropía, según la estructura deseada, la herramienta MESCAL lo hace automáticamente, pasando desapercibido por el usuario.

7.1.5 Predicción

A partir de los datos experimentales del campo residual y los valores obtenidos en el análisis estructural, se estiman puntualmente valores del parámetro analizado, empleando la técnica de *Kriging Ordinario*.

En el programa MESCAL es necesario definir el paso de cálculo, el origen de la malla se define en forma automática, tomando como base las coordenadas X e Y más bajas. Para el ejemplo, el paso de cálculo usado es de 100 m, adicionalmente se hizo un ajuste de la malla, quedando definida a partir de Xmin = 482500 y Ymin = 2145000, con un máximo de incrementos de 50 en dirección horizontal y de 56 en dirección vertical.

Para la validación de la predicción se utilizó el programa *OKB2D* que es una librería de *GSLIB*, para la ejecución de este programa se requiere contar con un archivo de datos con los siguientes parámetros:

-nombre del archivo con los datos experimentales.
-nombre de archivo de salida, donde se presentaran los resultados obtenidos.
-definición de la malla (en ejes *X* e *Y*), se utiliza la misma que se utilizó para MESCAL.
-el radio de búsqueda (7000 m).
-la distancia de correlación, el ángulo y el coeficiente de anisotropía de la estructura seleccionada previamente y la varianza de la muestra (estos coeficientes fueron obtenidos en el apartado anterior).

Al realizar el cálculo con *OKB2D*, el programa genera un archivo con solamente los valores de la estimación y la varianza de estimación en cada punto de estimación. Para agregar las coordenadas *X* e *Y* a las estimaciones se utiliza el programa *ADDCOORD* que también es una librería de *GSLIB*.

7.1.6 Mapeo

Finalmente, con el propósito interpretar en forma visual los resultados numéricos, se recurre a herramientas de graficación especializada. Para este trabajo, se empleó el programa *Surfer* (Golden Software, 2002) programa comercial. Para la graficación de los resultados debe tenerse cuidado de no realizar una nueva interpolación a partir de los valores interpolados, esto se logra haciendo coincidir los nodos de la malla de estimación con los nodos de la malla de graficación. Los resultados de la graficación se muestran en las *figuras 7.10* a *7.13*.

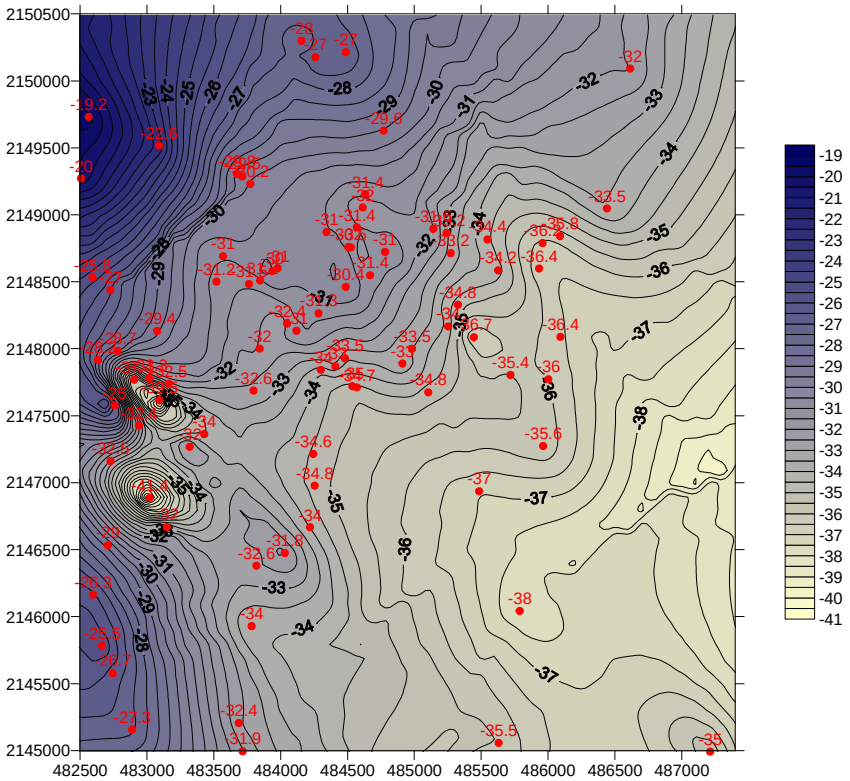


Figura 7.10 Mapa contornos de la profundidad estimada de la CD (MESCAL).

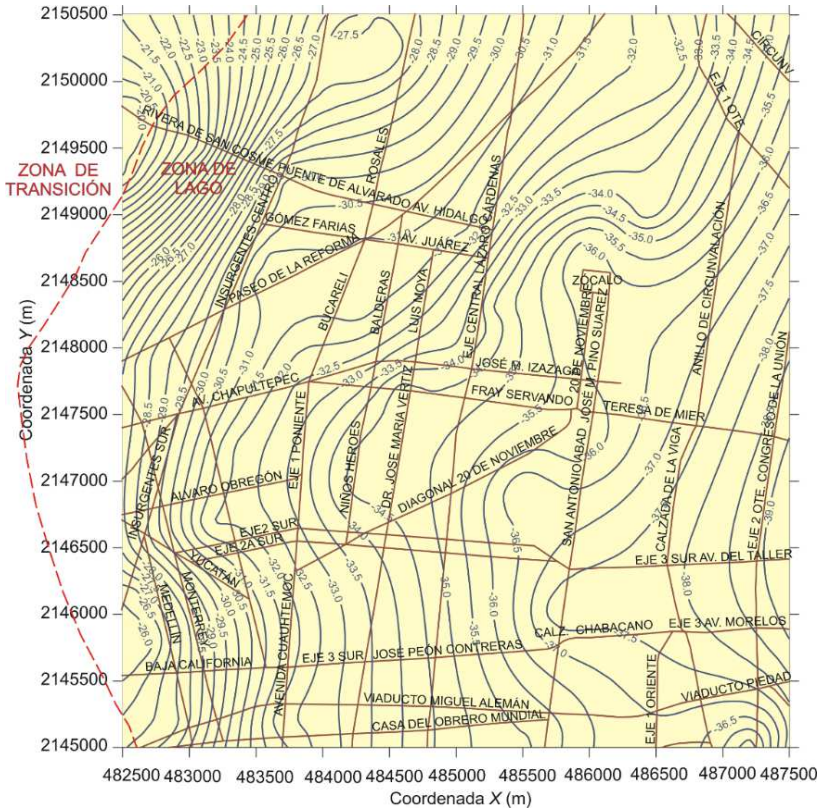


Figura 7.11 Mapa contornos de la profundidad estimada de la CD (GSLIB); (Auvinet y Juárez, 2003; Juárez, 2014).

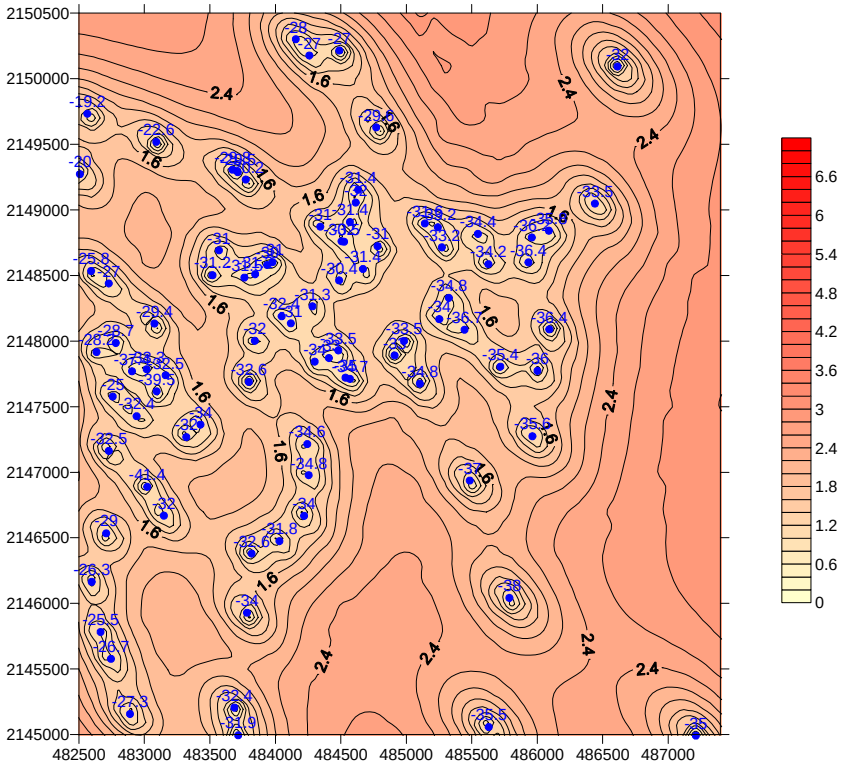


Figura 7.12 Mapa contornos de la varianza de estimación de la profundidad de la CD (MESCAL).

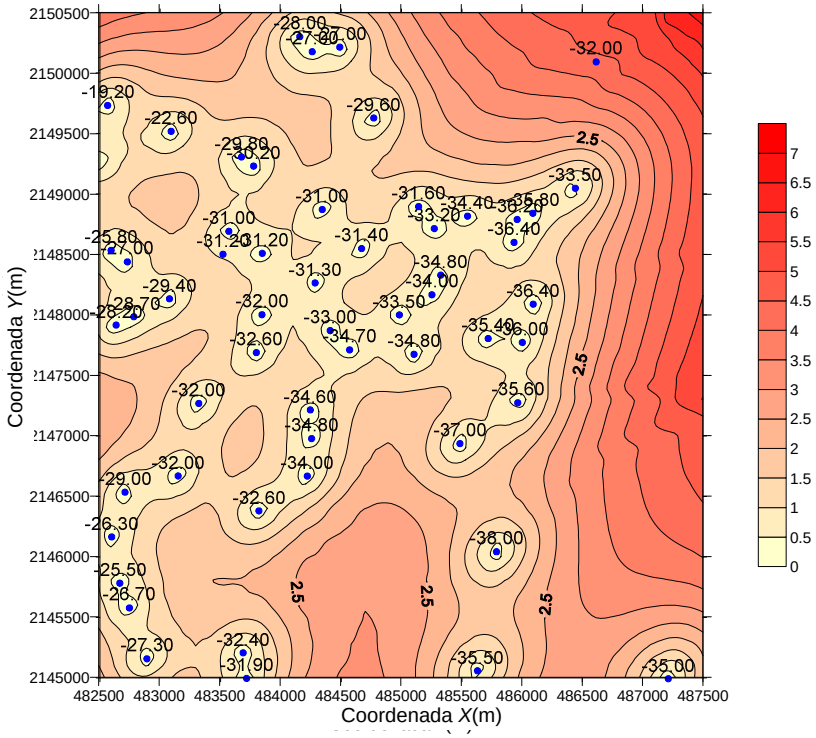


Figura 7.13 Mapa contornos de la varianza de estimación de la profundidad de la CD (GSLIB); (Auvinet y Juárez, 2003; Juárez, 2014).

7.2 Análisis geostadistico de la distribución espacial del contenido de agua del subsuelo a lo largo del trazo de la Línea 12 del Sistema de Transporte Colectivo Metro

Para validar la herramienta MESCAL se realizó también un análisis en 3D siguiendo la misma metodología.

7.2.1 Definición del campo

Los valores del contenido de agua, w (%), del subsuelo se considera un campo aleatorio $V(X)$, distribuidos dentro de un espacio R^p, con $p = 3$ (volumen de suelo, 3D). El conjunto de valores medidos dentro del dominio R^p, constituye una muestra de ese campo aleatorio. Para este análisis se consideraron 102 sondeos geotécnicos distribuidos en forma aleatoria.

En las *figuras 7.14* y *7.15* se muestra la ubicación en planta de los sondeos geotécnicos considerados para el análisis del contenido de agua a lo largo del trazo de la *Línea 12* del *Sistema de Transporte Colectivo Metro*.

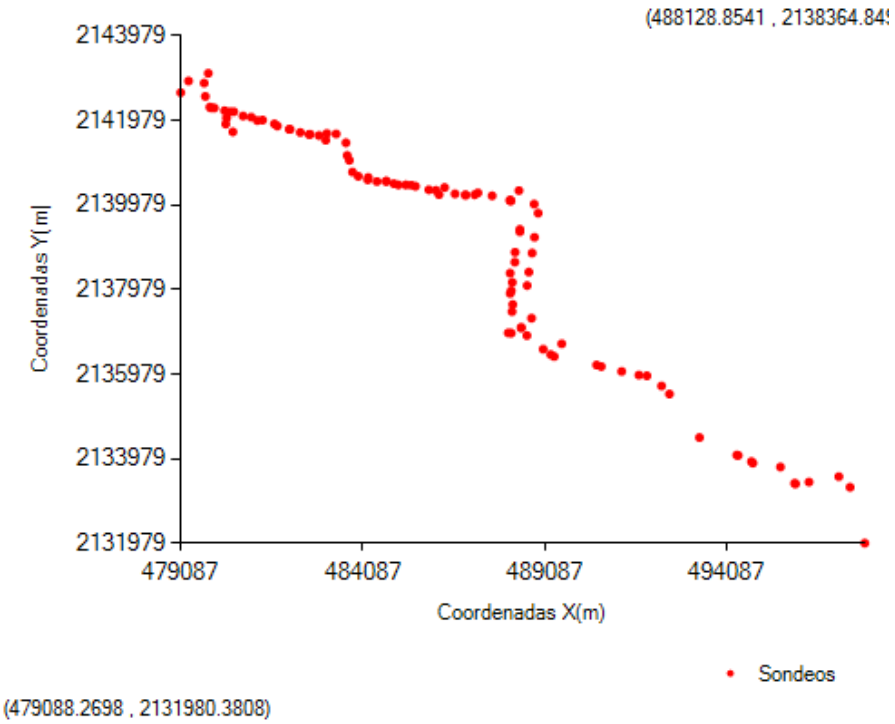


Figura 7.14 Ubicación en planta sondeos (MESCAL).

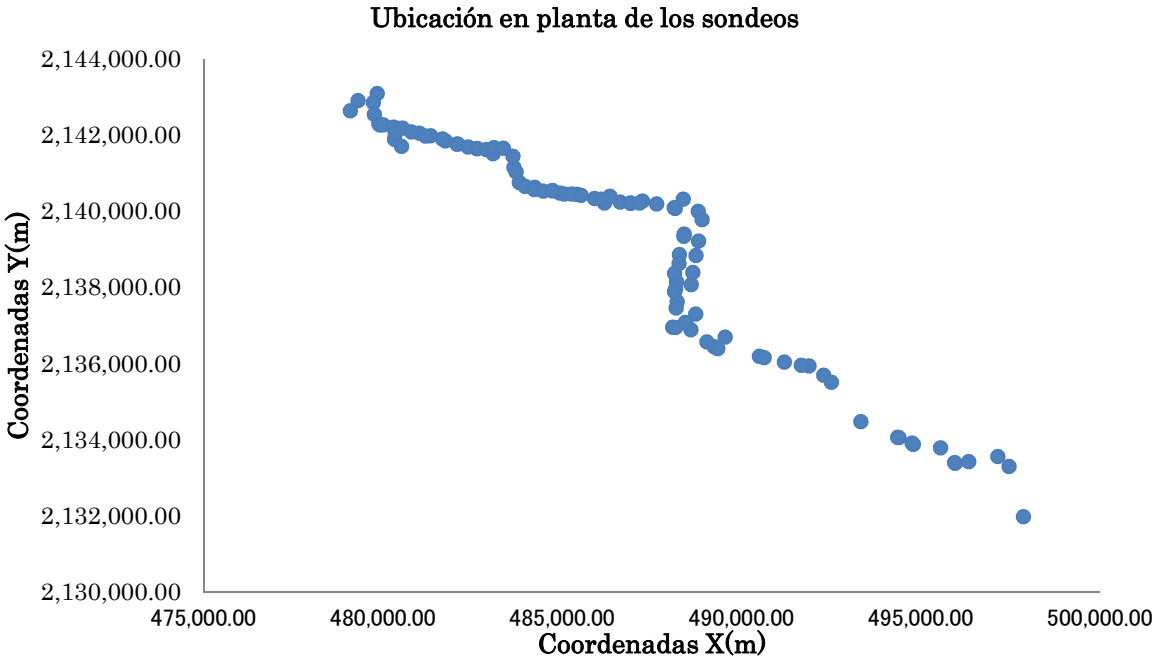


Figura 7.15 Ubicación de los sondeos en planta (Excel).

7.2.2 Descripción estadística

Los principales parámetros estadísticos de contenido de agua, *w* (%), se estiman a partir de los datos experimentales, los resultados se presentan en la *tabla 7.6*, se observa que para los dos casos los parámetros presentan prácticamente la misma magnitud.

Tabla 7.6 Parámetros estadísticos.

Parámetros	Valor	
	MESCAL	Excel
Media	105.713754	105.684259
Desviación estándar	97.355425	97.3734205
Coeficiente de variación	0.9209343	0.92136162
Mínimo	0.1	0.1
Máximo	662	662

7.2.3 Análisis de tendencia

La tendencia se evalúa mediante un análisis de regresión lineal. Los coeficientes de regresión obtenidos con la herramienta MESCAL y con *Excel* se presentan en la *tabla 7.7*, se observa que para los dos casos los coeficientes de regresión lineal presentan prácticamente la misma magnitud.

Tabla 7.7 Resultados de la Regresión Lineal.

Coeficientes de regresión lineal	Valor	
	MESCAL	Excel
a	0.01908	0.01911087
b	0.01941	0.01945989
c	-1.35333	-1.34856754
d	-50666.78632	-50666.2777

7.2.4 Análisis estructural

El campo aleatorio está definido en tres dimensiones; sin embargo, por sencillez se analiza solamente en dos direcciones: vertical y horizontal. Los correlogramas experimentales direccionales para el contenido de agua se calculan en las direcciones vertical y horizontal considerando el campo residual.

La correlación vertical se obtiene calculando la función del coeficiente de autocorrelación asociado a cada sondeo consigo mismo conforme se desfasa el perfil verticalmente una distancia constante; este coeficiente se estimó con un paso de cálculo de 0.20 m. El correlograma teórico horizontal del contenido de agua se obtiene evaluando la correlación cruzada entre todos los sondeos. Los correlogramas experimentales resultantes se presentan en las *figuras 7.16* y *7.17.*

A partir de los correlogramas experimentales se estiman las distancias de correlación espacial mostradas en la *tabla 7.8.*

Los modelos teóricos de correlación espacial horizontal y vertical se obtienen ajustando una función de tipo exponencial simple, los cuales se muestran en la *figuras 7.16* y *7.17.*

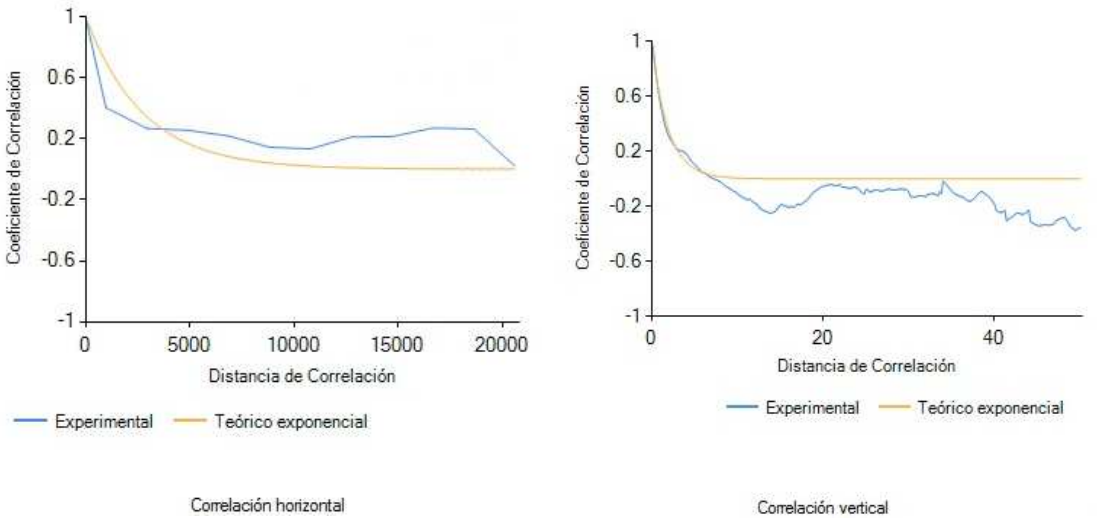


Figura 7.16 Correlograma horizontal y vertical (MESCAL).

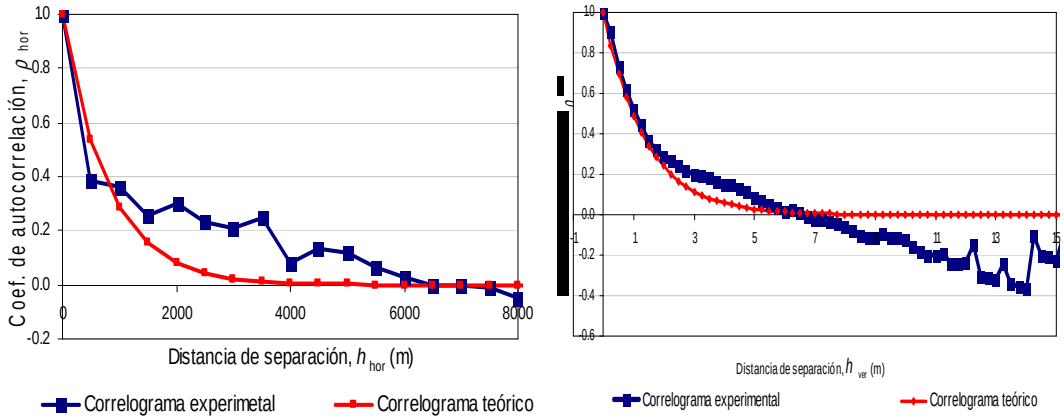


Figura 7.17 Correlograma horizontal y vertical (?).

Tabla 7.8 Distancias de correlación del contenido de agua.

Dirección	Distancia de correlación, δ(m)	
	MESCAL	SAAG
Vertical	3.84584	2.8
Horizontal	5533.34	1600

Se observa una variación en la magnitud de los resultados, esto se debe a la diferencia de datos considerados en cada caso.

7.2.5 Predicción

Con las distancias de correlación espacial δ_{ver} y δ_{hor}, los modelos teóricos y los datos experimentales de los 102 perfiles geotécnicos disponibles, se estimaron perfiles de contenido de agua, w(%) a lo largo del trazo (*figura 7.18*). La propiedad se estimó puntualmente hasta una profundidad de 60 m, con un paso de cálculo de 0.2 m.

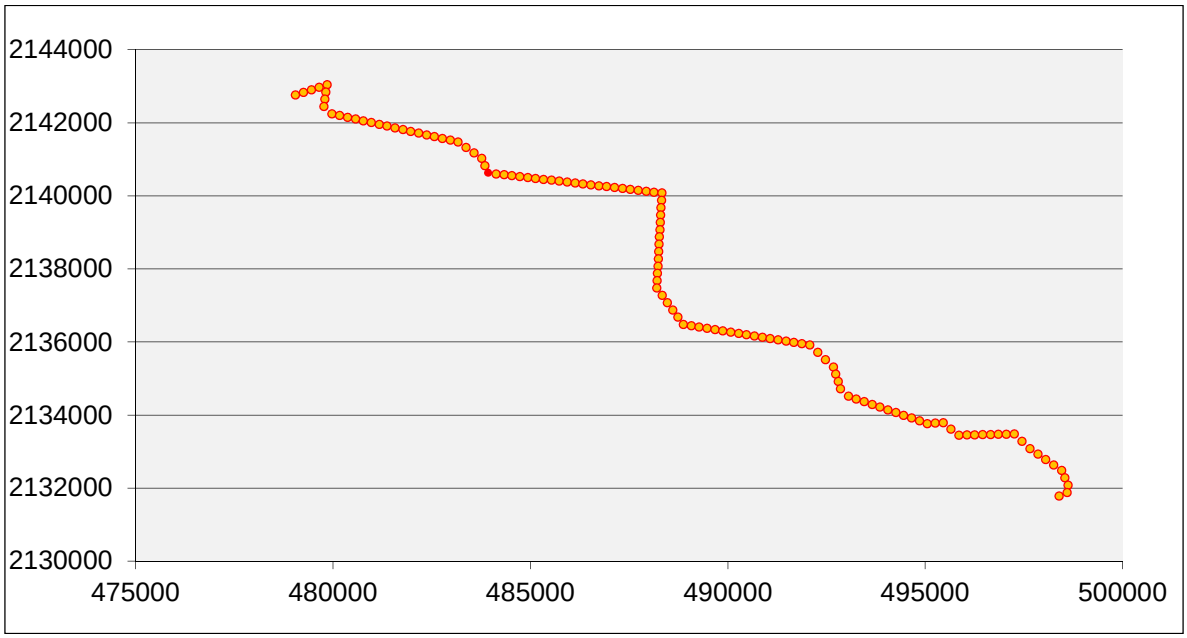


Figura 7.18 Trazo de estimación.

7.2.6 Mapeo

Para facilitar la interpretación de los resultados, a partir de la integración de los perfiles estimados sobre el eje de trazo se construyó un corte que representa la distribución espacial del contenido de agua a lo largo del trazo de la Línea 12. En la *figura 7.19* se presenta la sección transversal (virtual) del contenido de agua estimado a lo largo del trazo. Así también, a partir de la desviación estándar de estimación se construye el corte de la desviación estándar de la estimación en la *figura 7.20.* En la *figura 7.21* se presenta una vista en perspectiva (3D) del corte de contenido de agua estimado.

Por otra parte, con el propósito de validar los resultados, en la *figura 7.22* se presenta el mismo modelo elaborado por el Laboratorio de Geoinformática (2010). Este modelo se elaboró empleando la herramienta SAAG.

Con base en las *figuras 7.21* y *7.22* se observa que existe una aparente diferencia, lo cual se debe a que los dos modelos se elaboraron con diferente soporte de datos, siendo mayor en el modelo elaborado en este trabajo. Esto permite contar con un modelo más confiable, el cual puede ser actualizado fácilmente conforme se agreguen nuevos datos.

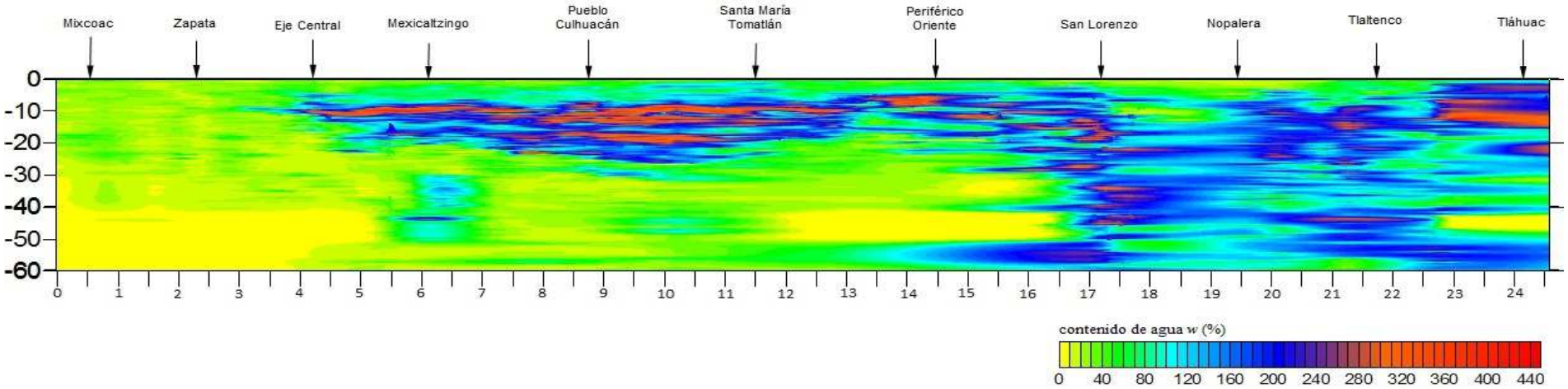


Figura 7.19 Sección transversal (virtual) contenido de agua estimado a lo largo del trazo de la Línea 12 del Metro.

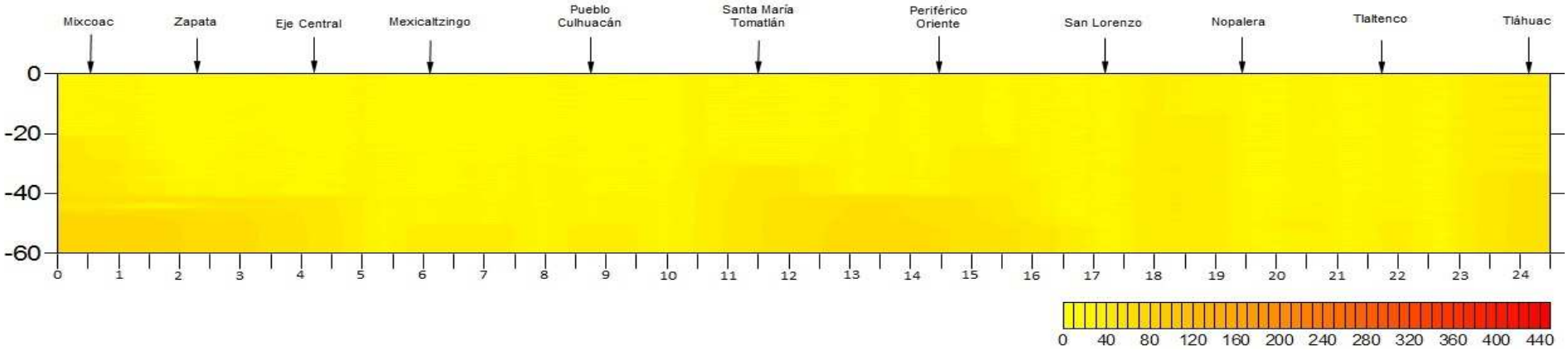


Figura 7.20 Sección transversal de la desviación estándar de estimación.

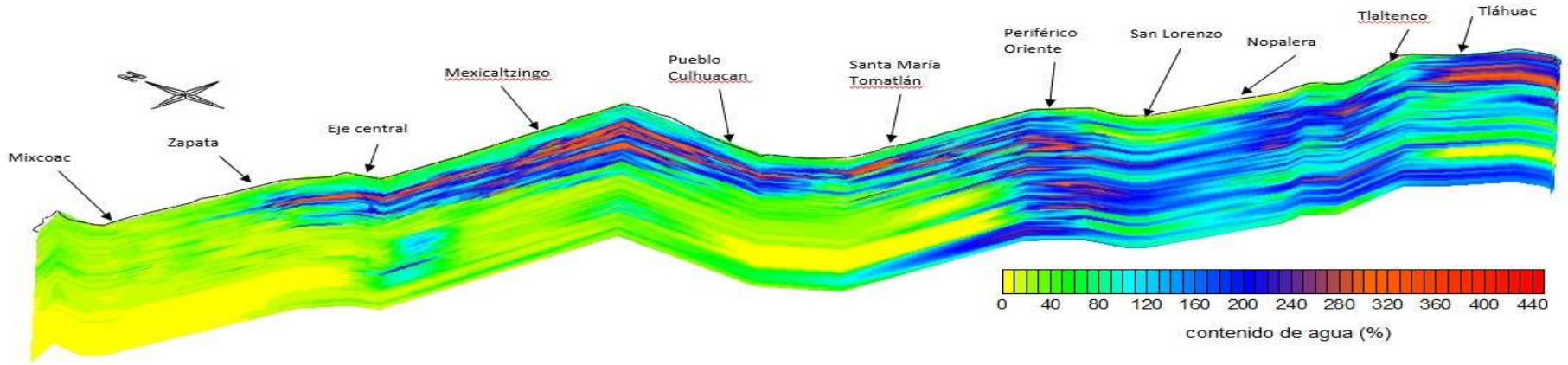


Figura 7.21 Mapa estimación de contenido de agua, Línea 12 del Metro.

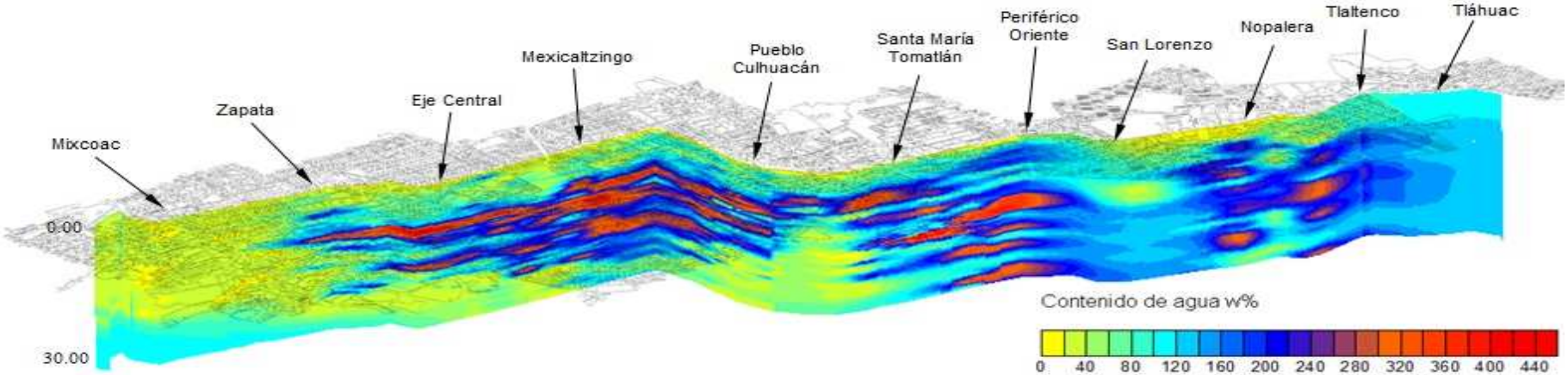


Figura 7.22 Mapa estimación de contenido de agua, Línea 12 del Metro (Auvinet, Juárez y Méndez, 2010).

7.2.7 Interpretación de resultados

Las diferencias apreciables entre los modelos de las *figuras 7.21* y *7.22* puede deberse a:

1. La herramienta SAAG, utilizada para la estimación del modelo de la *figura 7.22*, emplea la técnica de probabilidad condicional, que es equivalente a la técnica de *Kriging Simple* (toma en cuenta la media global del campo); mientras que, la herramienta MESCAL utiliza la técnica de *Kriging Ordinario* (no toma en cuenta la media global del campo).

2. Los datos empleados para la elaboración del modelo de referencia (*figura 7.21*) son en menor número que en el ejemplo de aplicación presentado en este trabajo (*figura 7.22*).

Desde el punto de vista geotécnico, los valores altos de contenido de agua (de 100% y más) se asocian a suelos de arcilla lacustre de baja resistencia y alta compresibilidad y los valores bajos de contenido de agua (menores a 100%) se asocian a materiales rígidos (limo y arena) de alta resistencia y baja compresibilidad.

Tomando en cuenta la consideración anterior, los modelos de las *figuras 7.21* y *7.22* permiten identificar estratos altamente compresibles y de baja resistencia potencialmente críticos que pueden influir desfavorablemente en comportamiento de las construcciones.

La ventaja que presenta la herramienta desarrollada es que permite el cálculo de la estimación a lo largo de diferentes segmentos en forma simultánea, a diferencia de la herramienta usada en trabajos anteriores (SAAG) que en cada ejecución realiza el cálculo en un solo segmento. La herramienta automatiza gran parte de la metodología geoestadística con aplicación en la geotecnia permitiendo realizar análisis completos en forma guiada.

8 Documentación

8.1 Elaboración de manual de instalación

8.1.1 Objetivo

El objetivo de la realización del manual de instalación es mostrar paso a paso la forma en que debe ser instalada la aplicación, y ayudar así al usuario en dudad respecto a la instalación.

A continuación se presentan las instrucciones de instalación del software MESCAL y los requerimientos mínimos para su instalación.

8.1.2 Requerimientos previos

La computadora deberá tener como mínimo:

- Sistema operativo superior a Windows XP.
- Sistema operativo de 64 bits.
- RAM de 2 GB.
- Velocidad de procesador 1GHz.
- Permisos de administrador.
- Microsoft .Network 4.0 (si no está instalado se instala a la hora de instalarse el programa).
- Windows Installer 3.1 (si no está instalado se instala a la hora de instalarse el programa).

8.1.3 Instalación

La instalación se realiza de la siguiente forma.

1. Para realizar la instalación, se debe ejecutar el paquete de instalación llamado "*MescalSetup*" (*figura 8.1*).

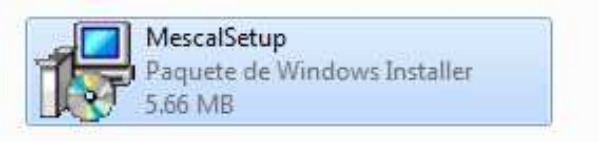


Figura 8.1 Paquete instalación.

2. Una vez que el instalador se pone en marcha, aparecerá una ventana (*figura 8.2*) indicando que es el asistente de instalación de MESCAL, seleccionar la opción *Siguiente*.

Figura 8.2 Asistente instalación MESCAL.

3. A continuación aparecerá una nueva ventana (*figura 8.3*) donde deberá seleccionarse la carpeta en la que se desea instalar MESCAL, además si desea instalar para todos los usuarios o únicamente para el que está instalando. Una vez terminado se acepta la opción *Siguiente*.

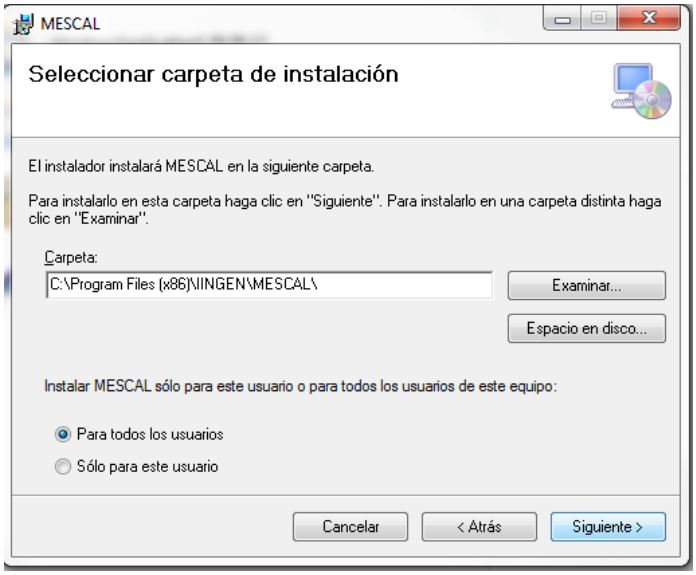


Figura 8.3 Selección carpeta de instalación.

4. Enseguida aparece otra ventana (*figura 8.4*) solo para confirmar la aceptación de la instalación bajo las especificaciones previamente establecidas, en caso contrario seleccionar la opción *Atrás*.

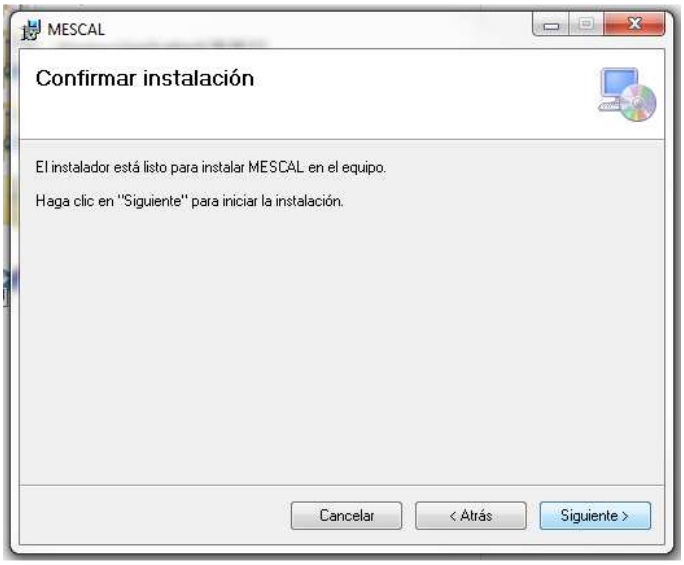


Figura 8.4 Confirmar instalación.

5. Consecuentemente, aparece una ventana (*figura 8.5*) donde se muestra el proceso de instalación de MESCAL en el equipo, por tanto, se deberá esperar a que la instalación termine.

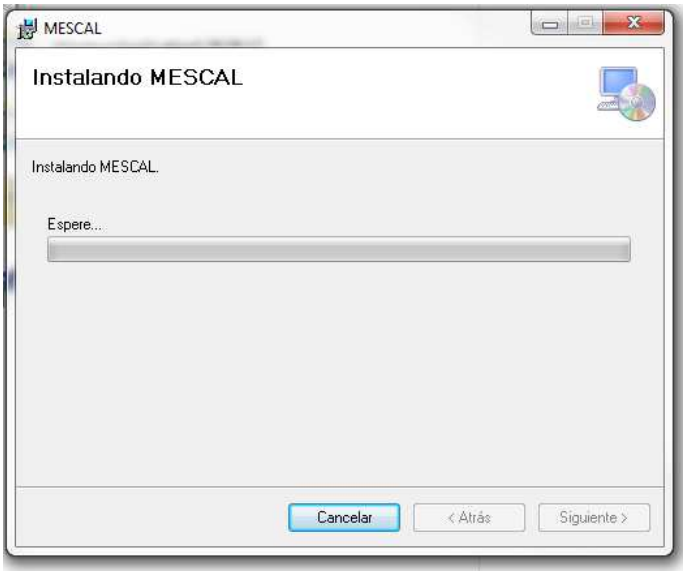


Figura 8.5 Instalación.

Al finalizar la instalación aparecerá una ventana indicando que la instalación ha finalizado (*figura 8.6*).

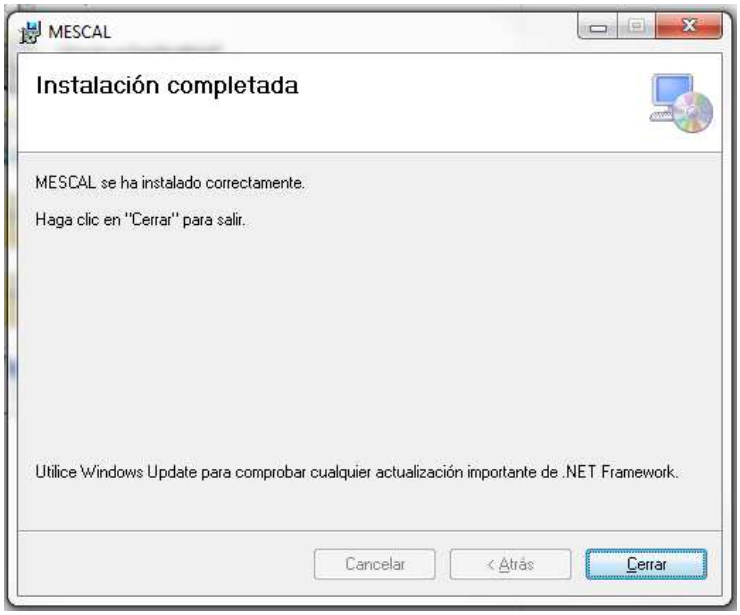


Figura 8.6 Instalación completa.

8.2 Elaboración de manual de usuario

El programa| se desarrolló de forma que su uso sea de manera intuitiva y sencilla para los usuarios, por lo que, sólo es necesario un conocimiento básico de computación, pero con amplios conocimientos en Geoestadística. No obstante, se elaboró un manual de usuario ilustrativo para guiar al usuario durante el desarrollo de algún análisis.

8.2.1 Objetivo

El objetivo de éste manual es contar con un documento que ayude y guíe a los usuarios en el uso de la aplicación MESCAL. El manual se divide en dos secciones:

- Análisis para 2 dimensiones.
- Análisis para 3 dimensiones.

Particularmente, el manual está dirigido a los usuarios del laboratorio de Geoinformática del Instituto de Ingeniería, UNAM. Debe señalarse que usuarios deben tener amplios conocimientos de las etapas que constituyen un análisis geoestadístico: *Análisis de tendencia*, *análisis estructural*, *estimación* y *simulación*.

8.2.2 Inicio de la aplicación

Para realizar cualquier análisis es necesario acceder a la aplicación, la cual se identifica con el icono asociado a la aplicación (*figura 8.7*).

Figura 8.7 Icono MESCAL.

Una vez ejecutado, aparecerá una ventana de presentación (*figura 8.8*) seguida de una ventana donde se debe elegir el tipo de análisis que se desea realizar (*figura 8.9*).

Figura 8.8 Ventana de presentación.

Figura 8.9 Elección de tipo de análisis.

8.2.3 Análisis 2D

1. Al seleccionar la opción "*2D*" (*figura 8.10*) aparecerá la ventana del menú principal y los términos de privacidad de la aplicación (*figura 8.11*).

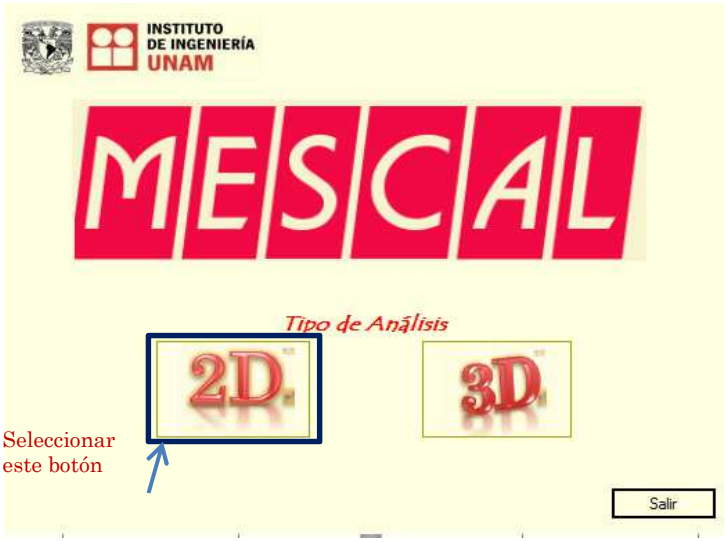


Figura 8.10 Elección análisis 2D.

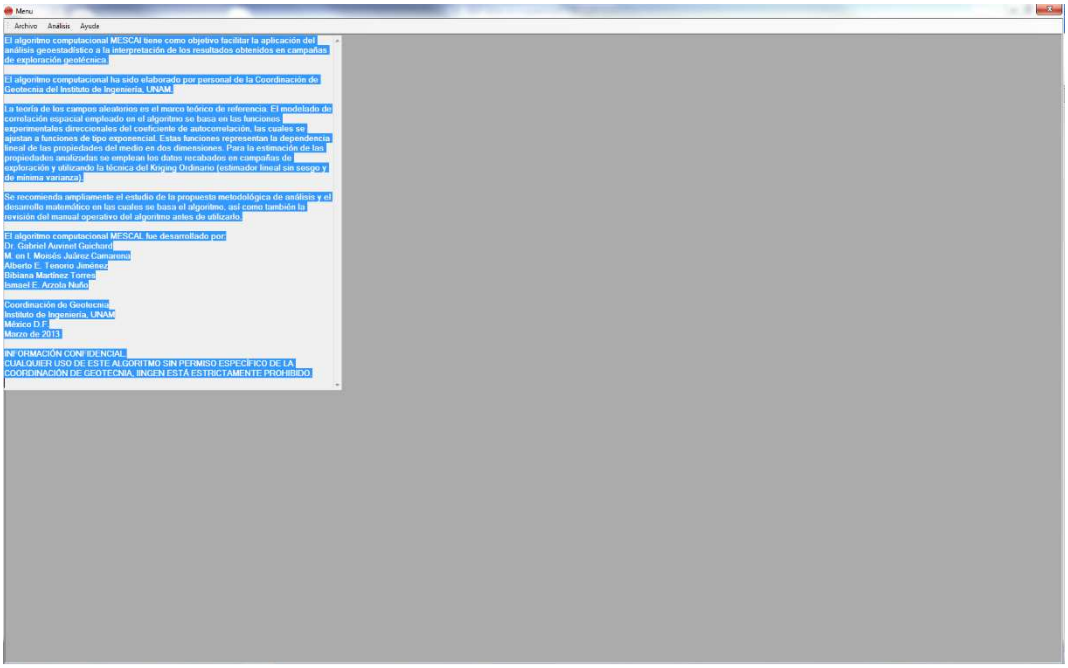


Figura 8.11 Ventana Principal.

2. En caso de un nuevo proyecto, deberá seleccionar en el menú la opción "*Análisis*", enseguida se desplegaran dos ventanas (*figura 8.12*) llamadas "*Explorador de archivos*" y "*Análisis*".

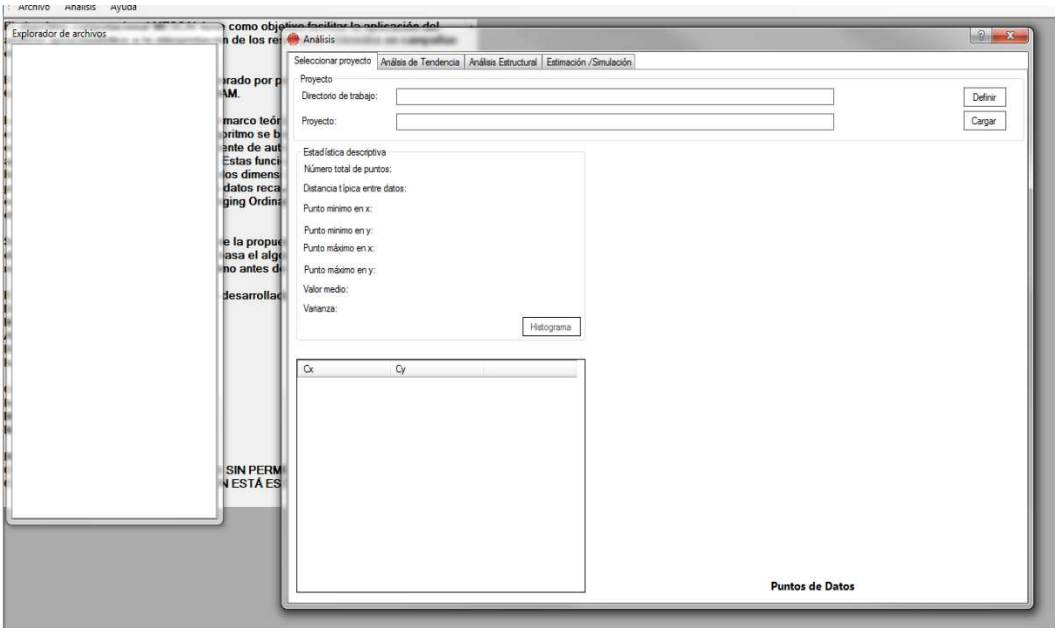
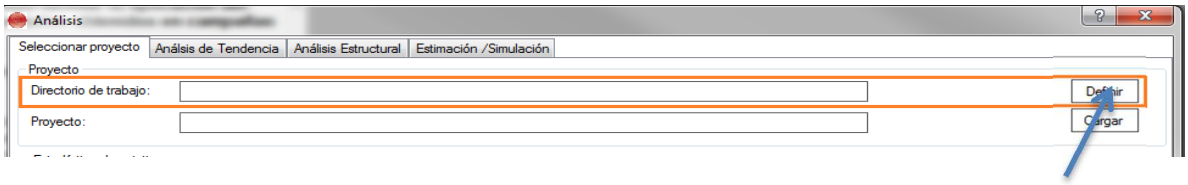


Figura 8.12 Ventanas utilizadas para en análisis.

A continuación se presenta el paso a paso de cada pestaña del *Análisis.*

I. Selección de proyecto

1. Escoger el directorio de trabajo (*figura 8.13*), donde se encuentran los archivos de los planos originales (*.po).



Abre explorador de carpetas para seleccionar directorio de trabajo

Figura 8.13 Definir directorio de trabajo.

2. Elegir un proyecto sobre el cual se trabajara (*figura 8.14*). Se calculan automáticamente los principales parámetros estadísticos, se muestra una gráfica de dispersión con los puntos de datos del proyecto, asimismo, se despliega una tabla de coordenadas (*Cx*), (*Cy*) y el valor de la variable de interés *V*(*X*). Adicionalmente, en caso necesario el usuario podrá visualizar el histograma, el cual podrá ser ajustado por el usuario en cuanto al número de clases.

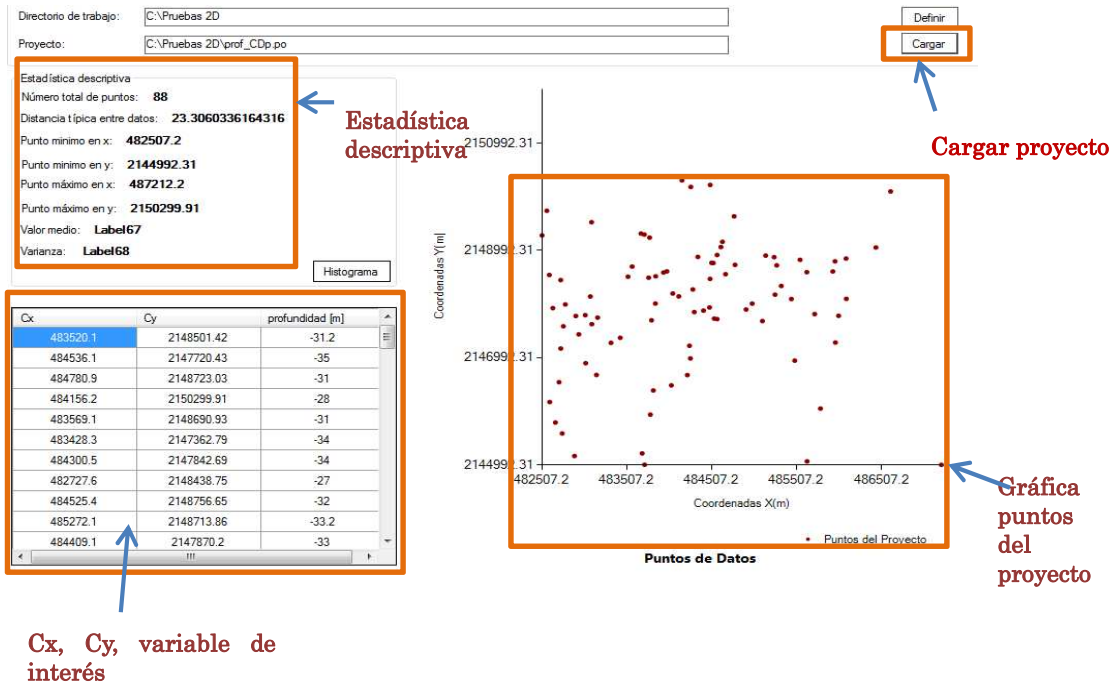


Figura 8.14 Carga de la vista de proyecto.

II. Análisis de Tendencia

1. Seleccionar la pestaña de "*Análisis de Tendencia*" y seleccionar la opción *Calcular*. Se despliegan los resultados de la regresión lineal (*figura 8.15*). A su vez aparece un mensaje indicando que se generó un archivo con residuos.

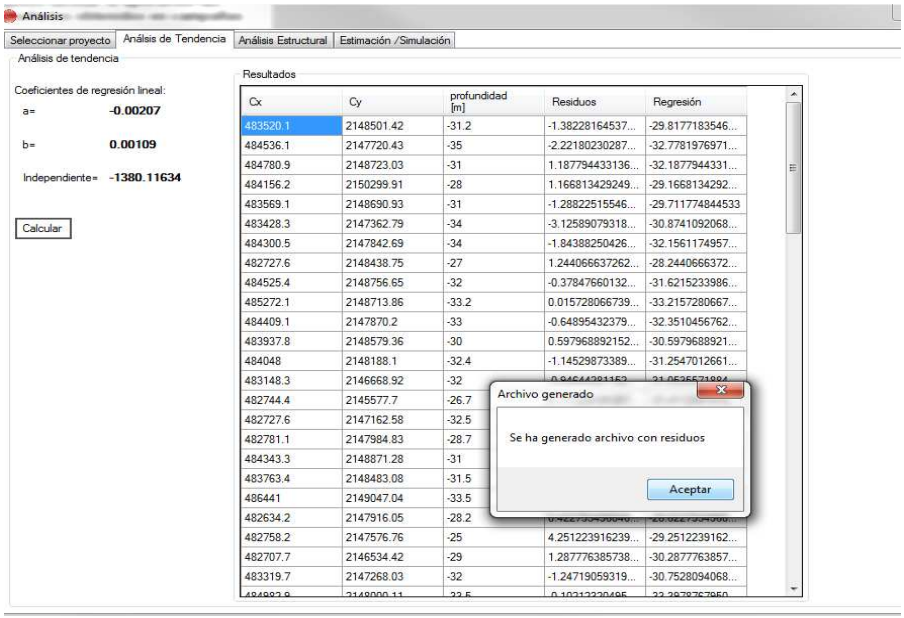


Figura 8.15 Análisis de tendencia.

III. Análisis Estructural

1. Seleccionar la siguiente pestaña llamada "*Análisis de Tendencia*".

2. Seleccionar con que datos se desea trabajar; si con los puntos originales o residuales (*figura 8.16*).

3. El usuario debe ingresar los parámetros para el cálculo de la función de autocorrelación (*figura 8.16*). Los valores a proporcionar son: tolerancia lineal, paso de cálculo, tolerancia angular y número de direcciones, donde:

   Paso de cálculo, h (xlag).- es la distancia vectorial unitaria de desfase; un criterio recomendable para definir esta variable consiste en considerarla aproximadamente igual a la distancia típica entre los datos disponibles.

   Tolerancia lineal (xtol).- es el margen lineal aceptable para el cálculo. Generalmente se considera igual a la mitad de xlag o menor.

   Ancho de banda (bandw).- Normalmente es el mismo valor que para la tolerancia lineal.

   Numero de direcciones (n).- número de direcciones acimutales que son consideradas para el cálculo.

   Tolerancia angular (atol).- es el margen angular aceptable para el cálculo respecto al vector direccional considerado. Comúnmente se considera igual a 180°/2n, donde n es igual al número de direcciones tomadas en cuenta para el cálculo.

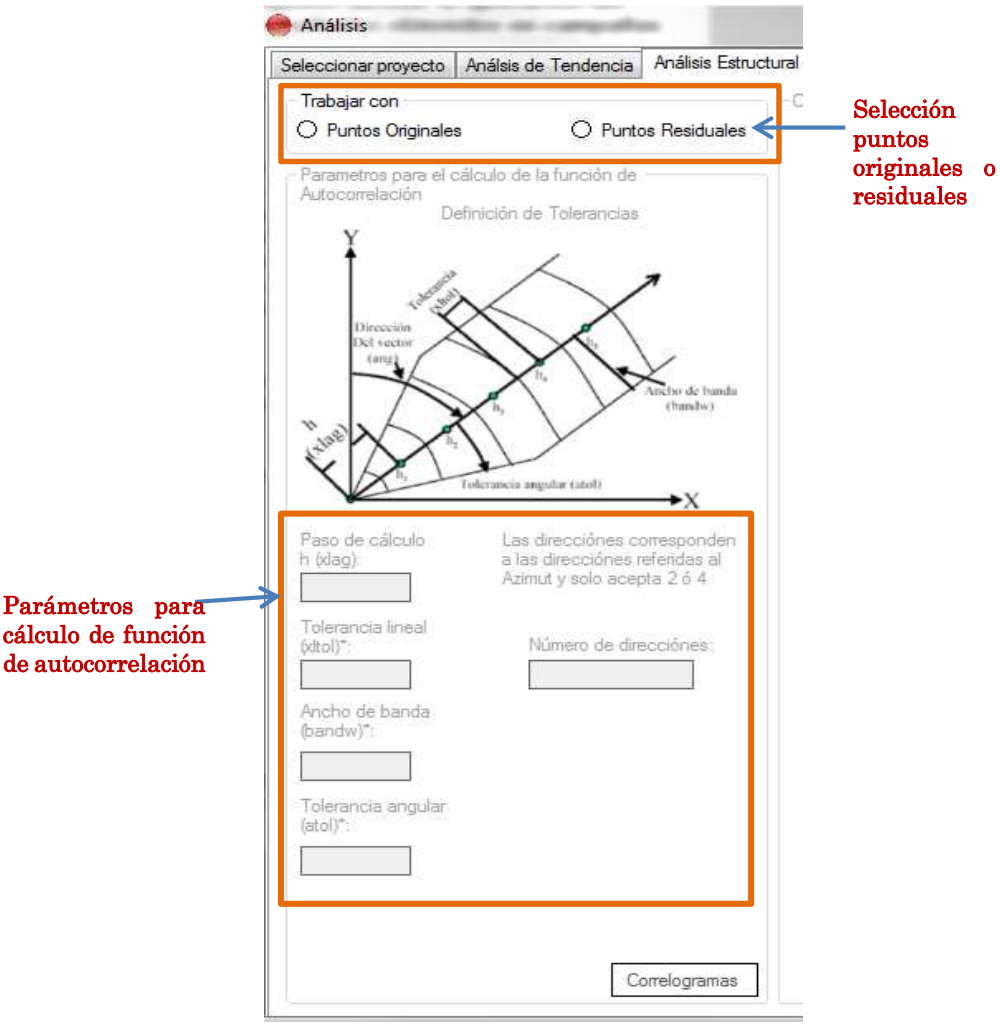


Figura 8.16 Parámetros para función de Autocorrelación.

4. Seleccionar la opción "*Correlogramas*". Al seleccionar esta opción dependiendo de las direcciones seleccionadas en la ventana anterior se mostraran 2 ó 4 gráficas de los correlogramas teóricos y experimentales (*figura 8.17*). Debajo de los gráficos aparecen indicadas la distancia de correlación calculada y un espacio para indicar la distancia de correlación modificada.

5. Si el usuario desea modificar la distancia de correlación y el correlograma teórico deberá ingresar la distancia de correlación modificada y dar *Enter* en el teclado, se actualizará automáticamente la gráfica del correlograma teórico (*figura 8.18*).

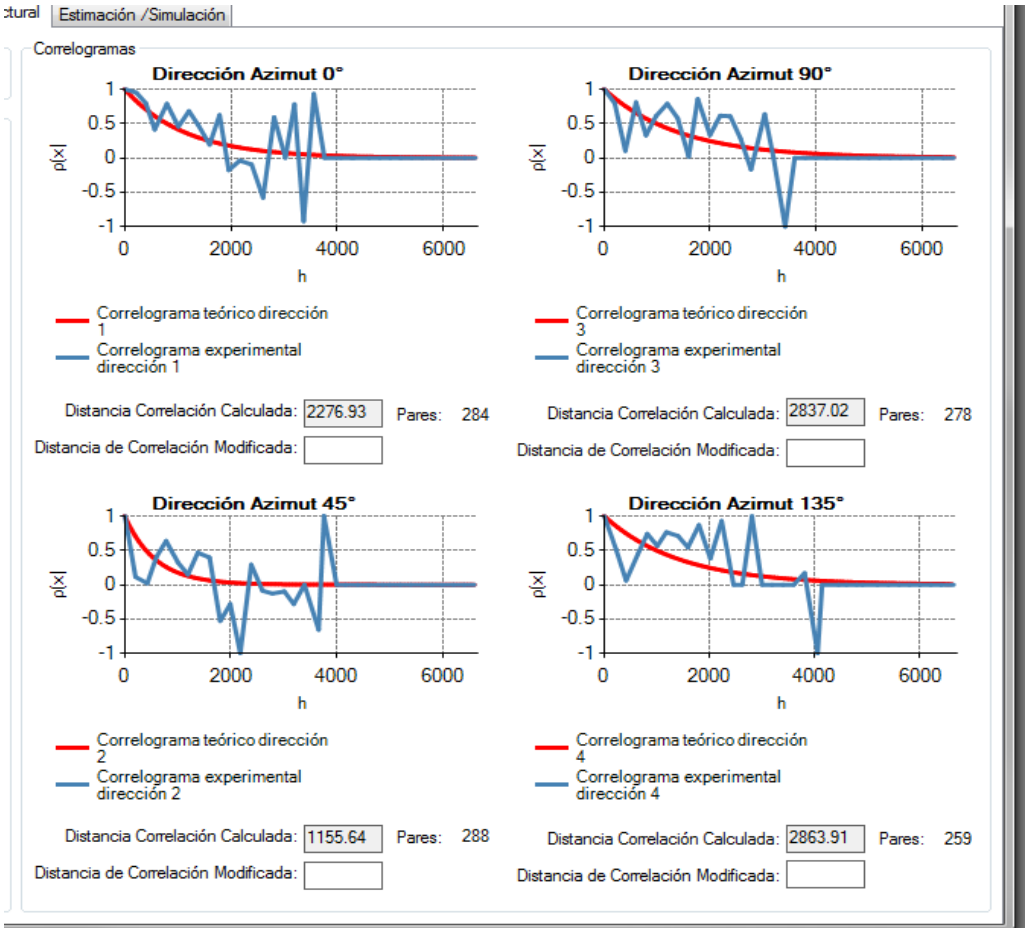


Figura 8.17 Correlogramas.

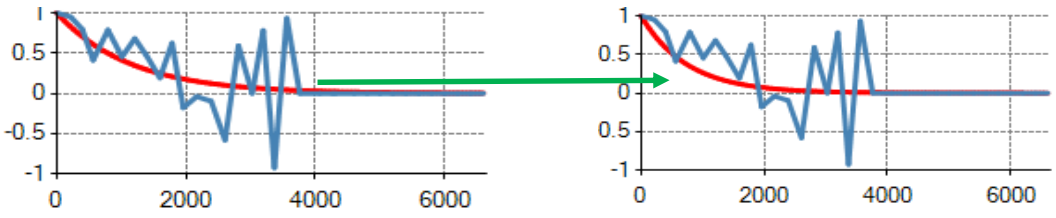


Figura 8.18 Modificación de correlograma teórico.

IV. Estimación/Simulación

1. Localizarse en la última pestaña llamada "*Estimación/Simulación*" (*figura 8.19*) donde debe indicar que es lo que realizara, simulación o estimación.

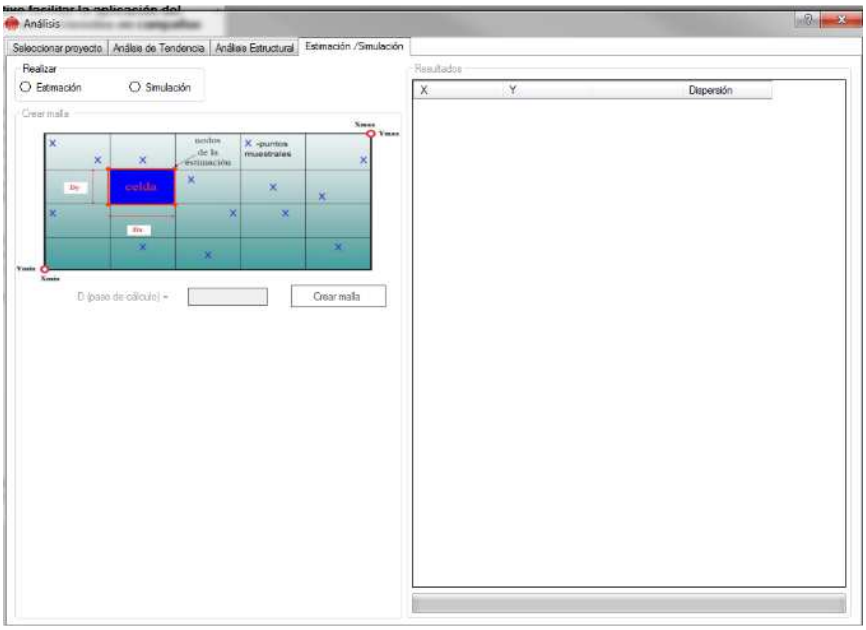


Figura 8.19 Estimación/Simulación.

2. El usuario deberá asignar un paso de cálculo de la malla (distancia entre dos nodos consecutivos de la malla) y seleccionar la opción "*Crear malla*".

3. Al generar la malla aparece la opción "*Calcular*". Además, de mostrarse en pantalla esta opción aparecen los valores mínimos y máximos de las coordenadas en *X* y *Y* de la malla regular de estimación.

   Si el usuario así lo desea podrá modificar los valores mencionados anteriormente seleccionando la opción "*Modificar Malla*" (*figura 8.20*).

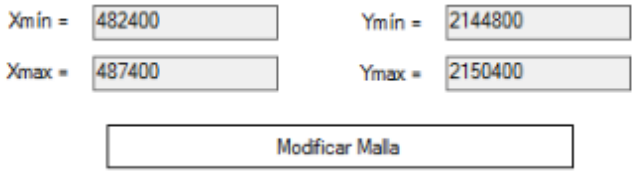


Figura 8.20 Modificación de malla.

Si el usuario decide modificar los valores de la malla se deberán definir los valores solicitados (*figura 8.21*) que corresponden a:

- Xmín (modif).- es la nueva coordenada mínima en X de la nueva malla de estimación.
- Ymín (modif).- es la nueva coordenada mínima en Y de la nueva malla de estimación.
- #Dx.- es el número de intervalos (pasos) en el sentido horizontal para la generación de una nueva malla de estimación.
    - #Dy.- es el número de intervalos (pasos) en el sentido vertical para la generación de una nueva malla de estimación.

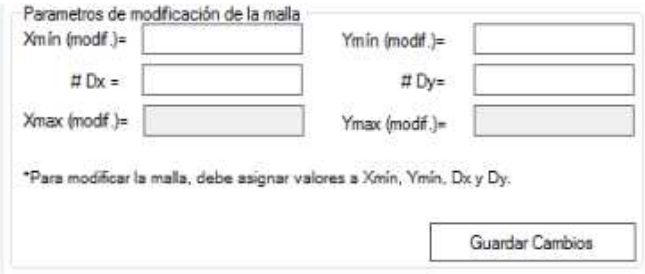


Figura 8.21 Parámetros para modificación de malla.

Seleccionar la opción "*Guardar Cambios*".

4. Seleccionar la opción "*Calcular*", al hacerlo, aparecerán los resultados del cálculo en forma tabular (*figura 8.22*). Y se habrá generado un archivo con los resultados.

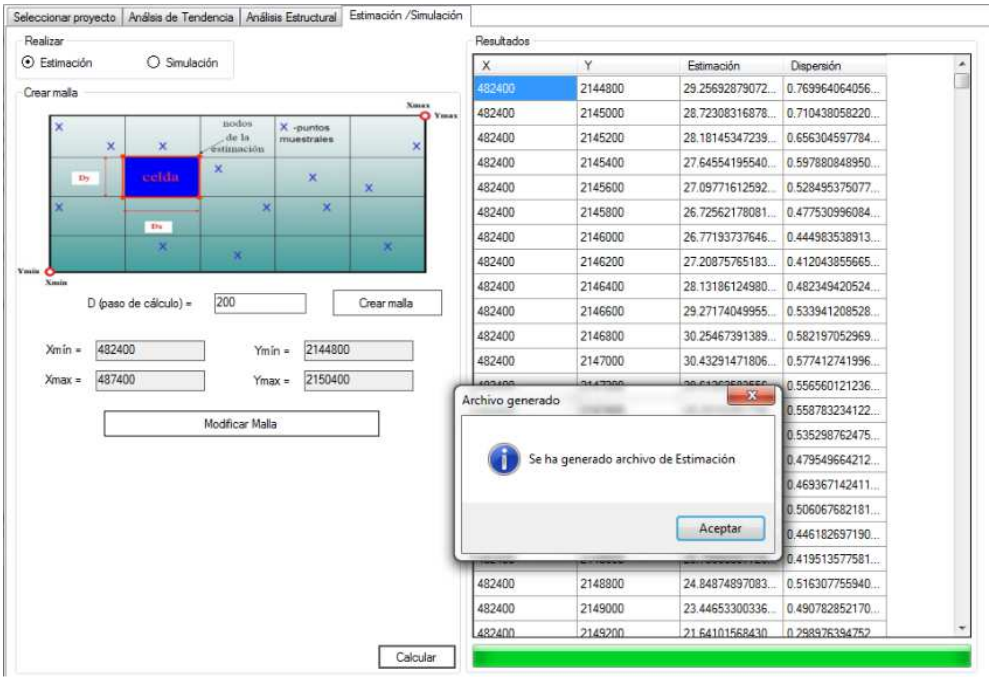


Figura 8.22 Cálculo terminado.

8.2.4 Análisis 3D

1. Accesar a la aplicación de MESCAL como ya se ha explicado en el análisis 2D y seleccionar el botón de "*3D*" (*figura 8.23*), aparecerá la ventana del menú principal y los términos de privacidad de la aplicación.

2. Seleccionar en el menú la opción de *"Análisis"*, a continuación se desplegaran en pantalla dos ventanas (*figura 8.24*), "*Explorador de archivos*" y "*Análisis*".

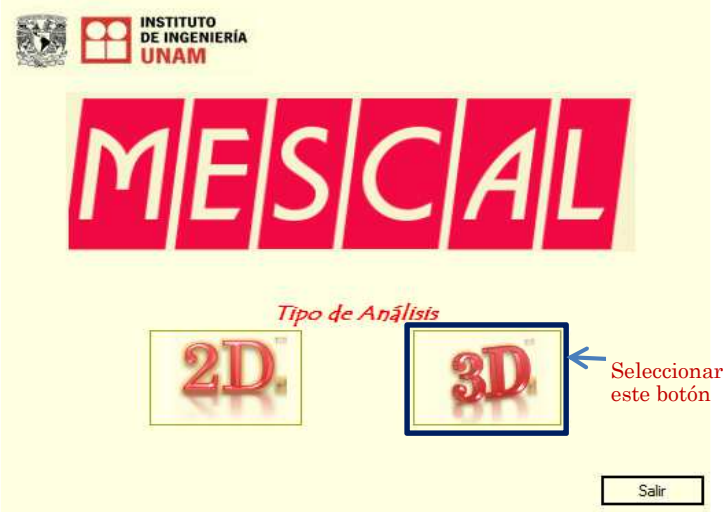


Figura 8.23 Selección análisis 3D.

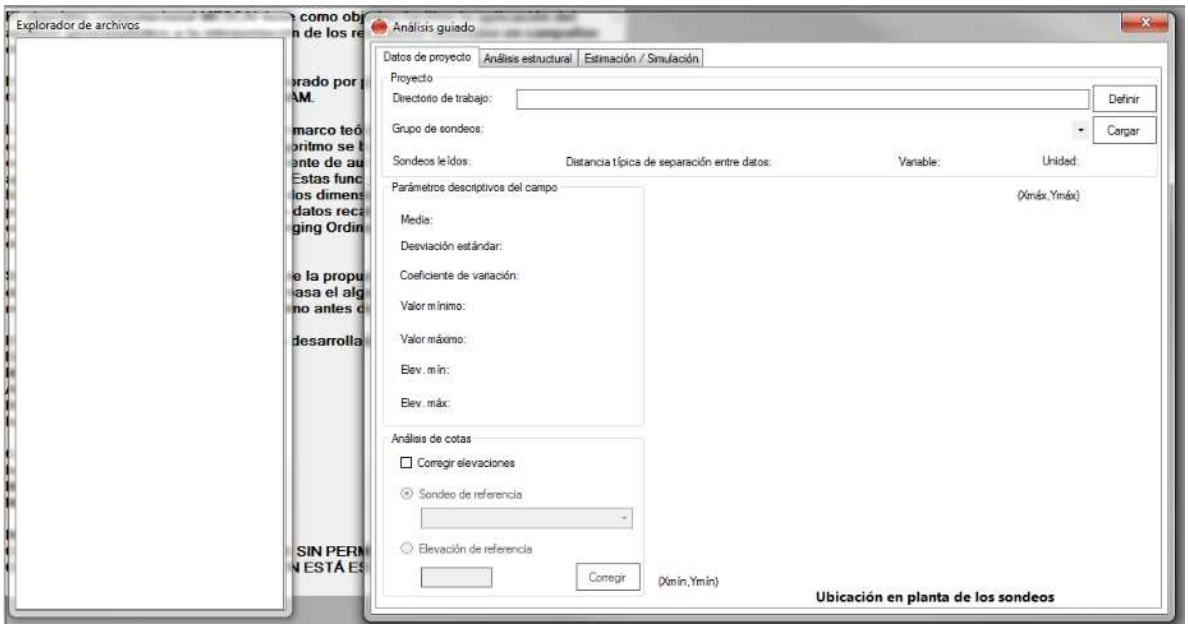


Figura 8.24 Presentación del análisis.

I. Selección de proyecto

1. Para iniciar, definir el directorio de trabajo donde se encuentren los sondeos originales (**.so*) seleccionando el botón "*Definir*", si no selecciona un directorio no podrá continuar. Al seleccionar el directorio de trabajo se actualiza automáticamente el explorador de archivos (*figura 8.25*).

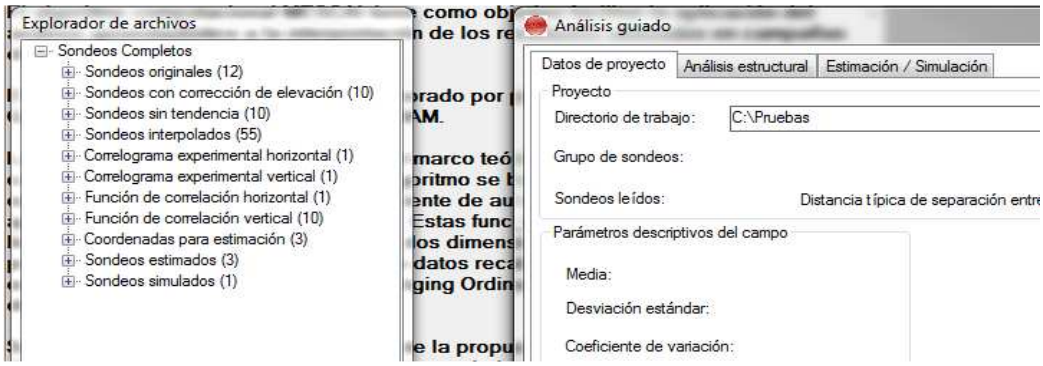


Figura 8.25 Actualización explorador de archivos.

2. Elegir el grupo de sondeos sobre el cual se trabajara (*figura 8.26*). Se calcula y se muestra la estadística descriptiva del campo, también se muestran la gráfica con la ubicación en planta de los sondeos y una descripción del grupo de sondeos que incluye el número de sondeos leídos, distancia de separación entre datos, la variable y la unidad (*figura 8.27*).

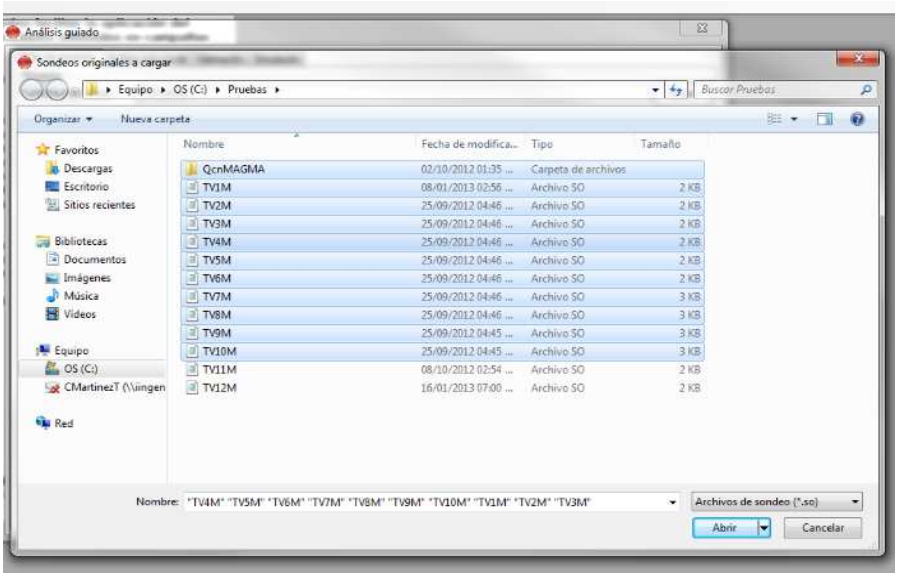


Figura 8.26 Selección del grupo de sondeos.

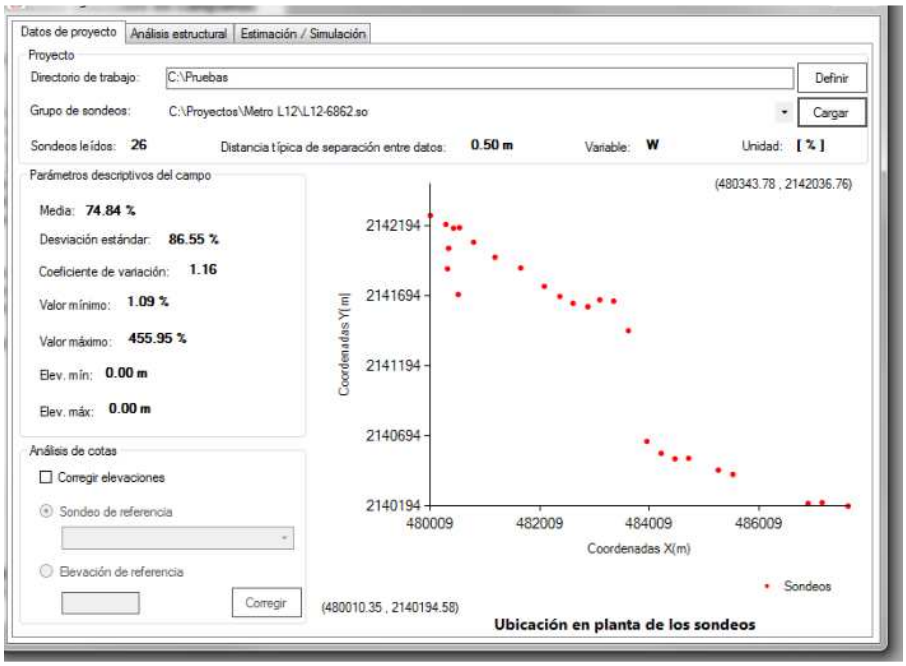


Figura 8.27 Datos del proyecto.

Adicionalmente en esta pestaña se podrá realizar el análisis de cotas; para realizarlo se deberá seleccionar la opción de "*Corregir elevaciones*", y seleccionar el tipo de corrección a realizar (*figura 8.28*).

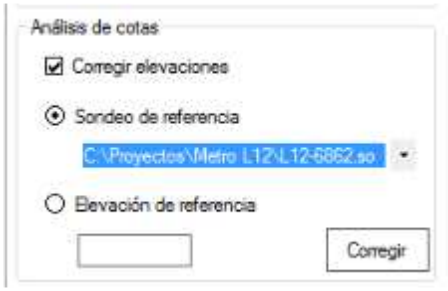


Figura 8.28 Corrección de elevación.

Si se selecciona la opción "*Sondeo de referencia*", debajo de la opción aparecerá una caja combo donde el usuario debe seleccionar el sondeo que desea utilizar como referencia. Si se desea corregir por elevación se deberá seleccionar la opción "*Elevación de referencia*" e introducir la elevación que desea utilizar para la .corrección.

Seleccionar la opción "*Corregir*".

II. <u>Análisis Estructural</u>

1. Seleccionar la segunda pestaña del *Análisis guiado* llamada *Análisis Estructural* (*figura 8.29*), en esta pestaña se podrá realizar el *Análisis de tendencia* y el *Análisis de correlación*.

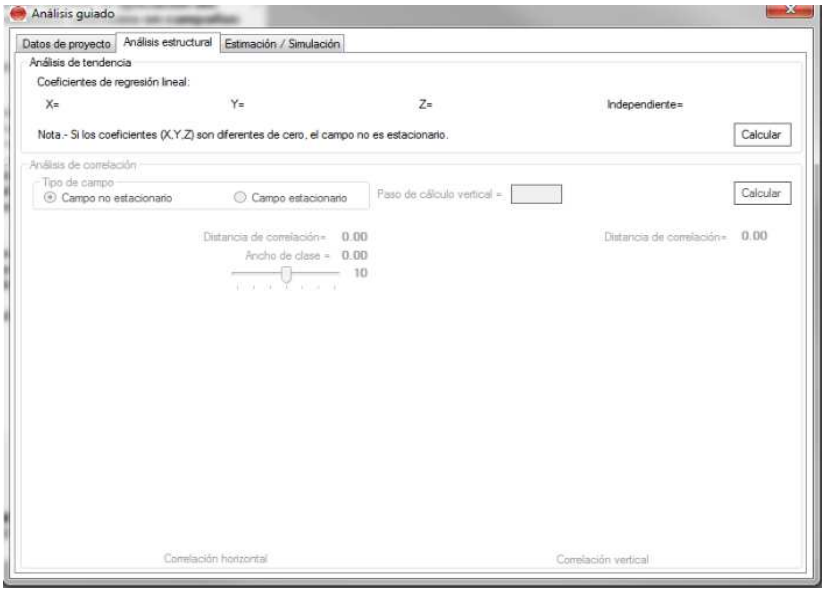


Figura 8.29 Análisis Estructural.

- Análisis de Tendencia

2. Seleccionar la opción "*Calcular*" del grupo de *Análisis de tendencia*, aparecerá un mensaje indicando que se generaron sondeos sin tendencia *.sor* y se mostraran en pantalla los coeficientes de la regresión lineal (*figura 8.30*).

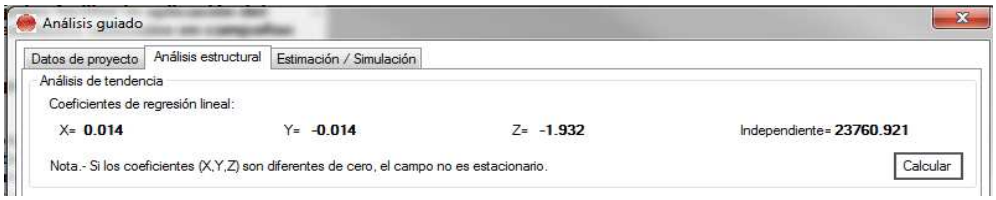


Figura 8.30 Análisis de Tendencia.

- Análisis de Correlación

3. Seleccionar el tipo de campo que se está analizando y asignar un paso de cálculo vertical para el análisis (*Figura 8.31*).

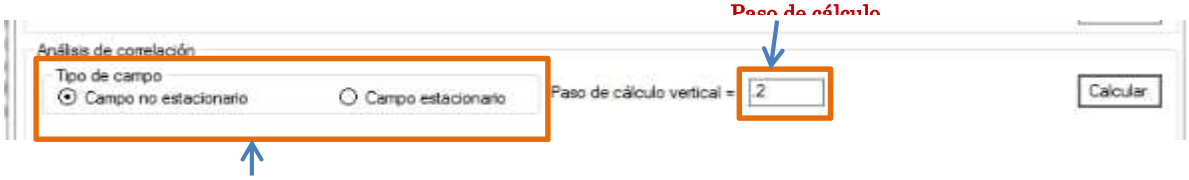


Figura 8.31 Parámetros para análisis de correlación.

4. Seleccionar la opción "*Calcular*" del área de *Análisis de Correlación*, a continuación se mostraran los correlogramas horizontal y vertical (*figura 8.32*).

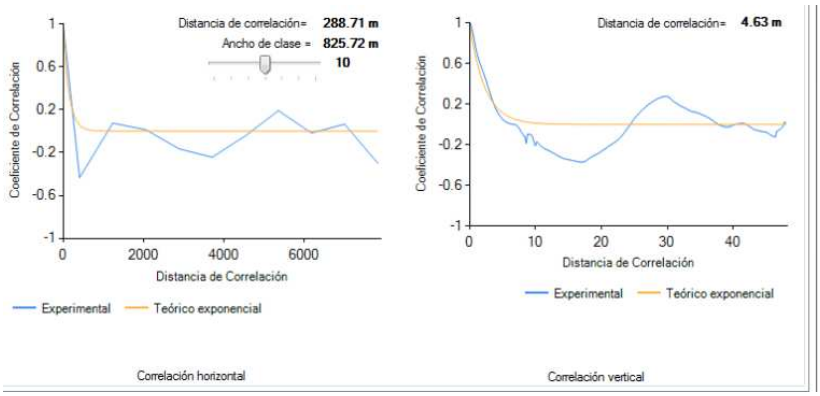


Figura 8.32 Correlogramas experimental y exponencial, vertical y horizontal.

Cada correlograma tiene indicada la distancia de correlación en la parte superior derecha, adicionalmente, el correlograma horizontal indicara el ancho de clase, la cual podrá ser modificada, cuando el usuario ajuste el ancho de clase automáticamente se ajustara la gráfica del correlograma experimental horizontal (*figura 8.33*).

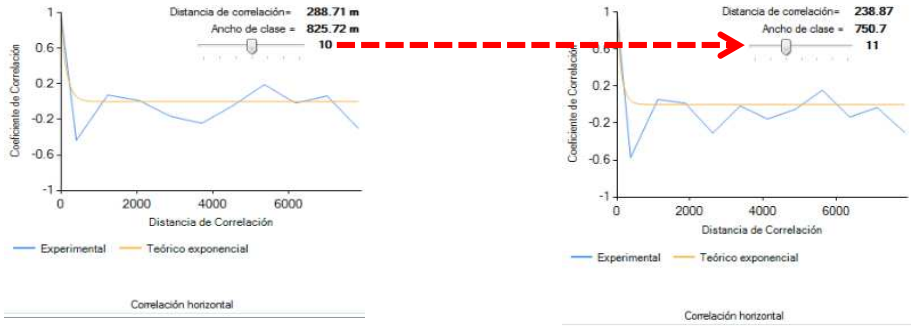


Figura 8.33 Ajuste correlación horizontal.

III. Estimación/Simulación

1. Seleccionar la tercera pestaña de *Análisis guiado* llamada *Estimación/Simulación* (*figura 8.34*), en esta pestaña se podrán realizar estimación y simulación condicional.

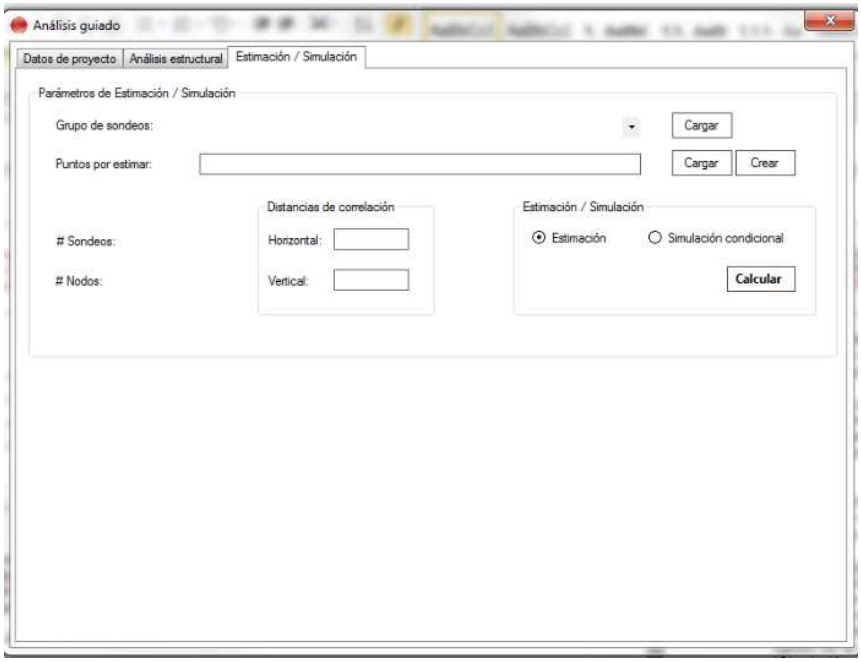


Figura 8.34 Estimación/Simulación.

En esta pestaña no se trabajara con los sondeos cargados en la pestaña *Datos de proyecto*, lo único necesario es que se haya definido el directorio de trabajo.

2. Cargar los sondeos originales a los que se desea realizar la simulación o estimación con la opción "*Cargar*" de *Grupo de sondeos.*

3. Asignar los puntos por estimar, para realizar esto existen dos opciones, cargar las coordenadas de estimación previamente creadas o crear unas nuevas.

   - Para cargar solo se deberá seleccionar la opción "*Cargar*" de *Puntos por estimar*, se mostrara una ventana donde se mostraran los archivos con extensión *.cpe*, que son los archivos de coordenadas para estimación.

4. En el recuadro llamado *Distancias de correlación* se deben asignar las distancias de correlación horizontal y vertical, estas distancias son las distancias que se calcularon en la pestaña de A*nálisis estructural.*

5. En el recuadro llamado *Estimación / Simulación* ubicado en la parte derecha inferior el usuario debe escoger que opción ("*Estimación*", "*Simulación condicional*") es la que desea realizar y seleccionar la opción "*Calcular*" (*figura 8.35*). En caso de seleccionar "*Simulación condicional*" aparecerá un recuadro que pedirá la *Semilla* que debe ser un número entero positivo.

6. Al terminar el cálculo se presenta un mensaje indicando que se ha generado un archivo de resultados (*figura 8.36*).

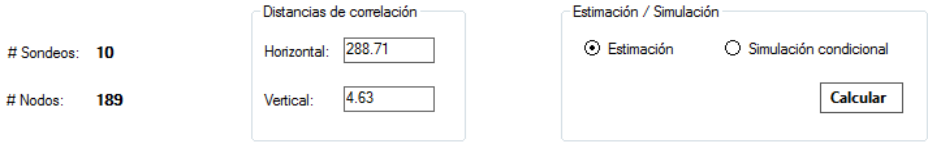


Figura 8.35 Vista previa de Estimación/Simulación.

Figura 8.36 Cálculo finalizado con éxito.

- <u>Archivo puntos por estimar</u>

1. En caso de querer crear un nuevo archivo de puntos por estimar, se selecciona la opción "*Crear*" de *Puntos por estimar*, aparecerá una ventana (*figura 8.37*), donde el usuario podrá seleccionar entre PUNTO, LINEA y LIBRE (*figura 8.38*) para asignar las coordenadas de estimación.

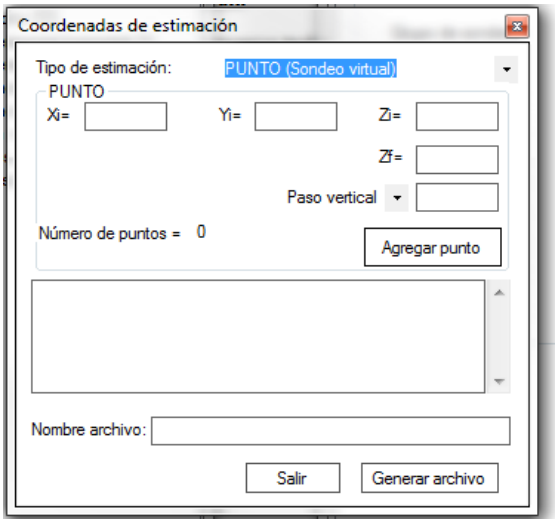


Figura 8.37 Coordenadas de estimación.

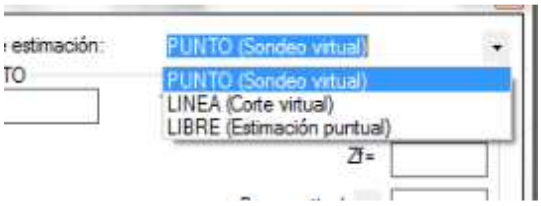


Figura 8.38 Tipo de estimación.

2. Dependiendo de la forma en que se decida crear el tipo de estimación se deberán dar las coordenadas delos puntos (*x*,*y*,*z*) y seleccionar la opción de "*Agregar punto*",

"*Agregar línea*" o "*Agregar nodo*", según sea el caso. Aparecerá debajo en un cuadro de texto los datos que se han ingresado (*figura 8.39*).

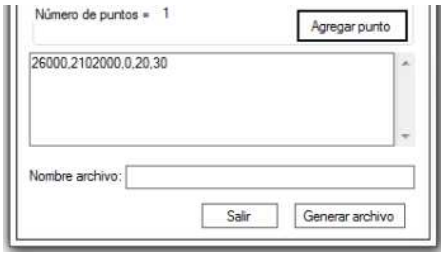


Figura 8.39 Datos para archivo de coordenadas de estimación.

3. Debajo, en el cuadro de texto donde dice *Nombre archivo*, debe asignar el nombre con el que se quiere guardar el archivo.

4. Por ultimo debe seleccionar la opción "*Generar archivo*", esto generara un archivo con extensión **.cpe* (*figura 8.40*) que se podrá cargar y continuar a partir de la instrucción 4 para realizar estimación o simulación.

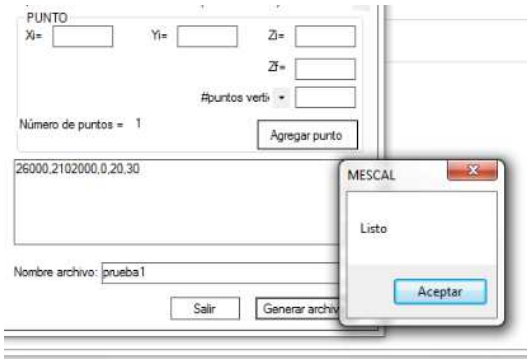


Figura 8.40 Archivo *.cpe generado.

9 Conclusiones y recomendaciones

9.1 Conclusiones

Con base en los resultados obtenidos, las conclusiones de este trabajo son:

- Se estudió el fundamento teórico de la Geoestadística con aplicación a la Geotecnia, puesto que, el trabajo principal consistió en desarrollar una herramienta para aplicar este fundamento teórico en la práctica geotécnica.

- Se repasó el fundamento teórico de la ingeniería de software para elegir el paradigma y lenguaje de programación a emplear en el desarrollo de la aplicación. Así también, se recordaron los conceptos del fundamento para el desarrollo de algoritmos y diagramas de flujo.

- Se realizó un análisis para definir los requerimientos específicos de los usuarios, así como, seleccionar la metodología a seguir y su desarrollo. Asimismo, se identificaron las condiciones de la infraestructura necesaria para tener control del ambiente de trabajo de la aplicación durante su desarrollo.

- Se desarrollaron varios algoritmos matemáticos:
  - Para el cálculo de los parámetros estadísticos de las propiedades geotécnicas ubicadas en el espacio (x,y,z).
  - Para el cálculo de la función de autocovarianza y autocorrelación (2D y 3D).
  - Para la definición de la malla de estimación en el espacio (2D y 3D).
  - Para la estimación de las propiedades geotécnicas del subsuelo en el espacio (2D y 3D) mediante la técnica del Kriging Ordinario.
  - Para la simulación condicional de las propiedades geotécnicas del subsuelo en el espacio (2D y 3D).

- Con base en los algoritmos desarrollados, se construyó una herramienta de software llamado MESCAL, basada en Visual Basic, en la que se implementan los algoritmos matemáticos previamente desarrollados como apoyo al análisis geoestadístico. Durante la implementación de los algoritmos se presentó un problema para el manejo de matrices de correlación, este problema se resolvió cuando se encontró la biblioteca compatible con VB llamada MaPack.dll que incluye funciones para trabajar con matrices durante el análisis, ahorrando tiempo y código.

- Con la herramienta desarrollada, se ha automatizado gran parte de la metodología geoestadística con aplicación en la geotecnia permitiendo realizar análisis completos en forma guiada, lo cual facilita y reduce el tiempo y recursos de los análisis, siendo esta la diferencia principal con otras herramientas.

- Se desarrollaron los manuales de instalación y de usuario como apoyo a los usuarios para facilitar el uso de la aplicación.

- El código del software desarrollado se documentó para tener control durante el desarrollo y, a su vez, para que en un futuro pueda ser fácilmente modificado y se requiera añadir nuevas funcionalidades.

- Mediante una serie de pruebas con datos reales se comprobó cada una de las etapas del análisis. Estas pruebas sirvieron para revisar el cumplimiento de los requerimientos planteados en la definición del problema. Así también, los resultados de estas pruebas fueron comparados con resultados obtenidos anteriormente con otras herramientas. Los resultados fueron coherentes, considerando la diferencia entre los datos empleados en cada caso.

- Por la limitación del tiempo disponible para desarrollar este trabajo no fue posible incluir un módulo de visualización grafica de los resultados. La graficación es el último paso de la metodología del análisis geoestadístico. Al realizar la validación del programa y graficar los resultados se planteó la necesidad de integrar a corto plazo este módulo al programa.

9.2 Recomendaciones

Al concluir este trabajo y con base en las conclusiones, antes mencionadas, se plantean las siguientes sugerencias con la finalidad de desarrollar un software más completo:

- *Adición de técnicas geoestadísticas.*

  - Existen otros métodos para realizar el cálculo de tendencia además de la regresión lineal que podrían incluirse.
  - Existen otras funciones menos robustas para el cálculo del análisis estructural basado en el variograma que podrían incluirse.
  - Existen otros métodos de estimación además del kriging ordinario, entre ellos se puede mencionar el kriging simple, kriging universal, kriging indicador y cokriging estos métodos podrían también incluirse.
  - También existen otras técnicas comúnmente en la Geoestadística que pueden ser aplicadas en la Geotecnia que podrían ser añadidos para tener una mayor funcionalidad dentro de la aplicación. Entre estas técnicas se encuentran la validación cruzada y el concepto de ganancia.

- *Adición de módulo de graficación.*

  - En este proyecto solo se llegó hasta la obtención de los resultados numéricos de estimación y simulación. El siguiente paso es la visualización de los resultados en forma gráfica, actualmente esto se realiza recurriendo a programas comerciales de graficación especializada. Por tanto, se recomienda continuar trabajando en la aplicación para añadir un módulo para que el programa realice la graficación de los resultados mediante mapas de contornos (2D) y secciones geotécnicas de propiedades del subsuelo (3D), empleando escalas de colores en todos los casos.

- *Integración a Sistemas de Información Geográfica (SIG).*

  - Integrar este tipo de Software a SIGs, de tal forma que se pueda extender la potencialidad de estos. Aunque, actualmente existen módulos de análisis geoestadístico integrados a los SIGs pero con limitaciones para su aplicación en la Geotecnia. Al integrar el programa desarrollado daría como resultado un programa exclusivo para su uso particular en la Geotecnia.

- *Generación de layouts o informes.*

  - Generación de reportes al finalizar cada análisis, resumiendo los datos principales obtenidos durante el análisis, por ejemplo, los parámetros de la estadística descriptiva e histogramas; los coeficientes y gráficas de regresión lineal; correlogramas, entre otros.
  - Generación de layouts con gráficas, mapas de contornos y modelos al finalizar cada análisis con elementos de ayuda definidos por el usuario, como son: escalas gráficas, coordenadas, mallas de referencia cartesiana, flechas de norte, mapas cartográficos, entre otros para una rápida y fácil generación de reportes.

Estas recomendaciones se sugieren realizar a corto plazo para lograr que el usuario ya no tenga que utilizar distintas herramientas de software para cada fase del análisis. De tal forma que todo quede integrado en un solo entorno de trabajo.

10 Referencias

1. Acuña, Luis, 2004, "*Visual Basic como segundo lenguaje*", Ed. Tecnológica de Costa Rica, Cartago, Costa Rica.
2. Agarwal, K. y Singh, Y., 2005, "*Software Engineering*", Pack Printers, New Delhi, India.
3. Agarwal, B., Tayal, S. y Gupta, M., 2010, "*Software Engineering & Testing*", Jones and Bartlett Publishers, USA.
4. Alonso, F., Martínez, L. y Segovia, F., 2012, "*Introducción a la ingeniería del software. Modelos de desarrollo de programas*", Delta Publicaciones, Madrid, España.
5. Arzuza, Johan, 2004, "*Algoritmo para reconstrucción digital 3D y visualización de superficies a partir de una muestra de datos*", Universidad Industrial de Santander.
6. Auvinet, G., 1976, "*Probabilidad y Estadística*", Notas del curso de Probabilidad y Estadística, División de Estudios de Posgrado, Facultad de Ingeniería, UNAM, México.
7. Auvinet, G., 1984, "*Variabilidad de los depósitos de carbón. Un enfoque estocástico*", Estudio realizado para Minera Carbonífera de Río Escondido, Coahuila, México.
8. Auvinet, G., 2002, "*Incertidumbre en Geotecnia*". Decimosexta Conferencia Nabor Carrillo, Sociedad Mexicana de Mecánica de Suelos, Querétaro, México.
9. Auvinet, Guichard, G., Juárez Camarena, M., Méndez Sánchez, E. y Barranco Eyssautier, A., 2010, "*Estudios consistentes en revisar, analizar y evaluar los aspectos relacionados con la Ingeniería Geotécnica y la Ingeniería Geosísmica asociados con el análisis, diseño y construcción de la Línea 12 del Metro. Tercera Etapa: Caracterización de detalle del subsuelo*", informe interno del II-UNAM, elaborado para la Dirección General del Proyecto Metro Línea 12, México.
10. Bhushan, B. y Prakash, S., 2007, "*Software Engineering*", Laxmi Publications, New Delhi, India.
11. Campusano, Juan, 2009, "*Diseño e implementación de un software de adquisición, graficación, y análisis de datos experimentales*", Tesis de Licenciatura, Facultad de Ingeniería, UNAM, México.
12. Casario, Marco, 2009, "*The Essential Guide to Flash CS4 AIR Development*", Friendsof, New York, USA.
13. Deutch, C. y Journel, A., 1992, "*Geostatistical Software Library, GSLIB*", Oxford University Press, New York, USA.
14. Dijkstra E., 1968 "*Go To Statement Considered Harmful"*, Communications of the ACM, Vol. 11, USA.
15. Evans, M. y Rosenthal, J., 2004, "*Probabilidad y estadística*", Ed. Reverté, Barcelona, España.
16. Farrell, Mary, 2002, "*Learning Computer Programming. It´s not about languages*", Charles River Media, USA.
17. Fernández, S. et al., 2002, "*Estadística descriptiva*", Ed. ESIC, Madrid, España.
18. Gabbrielli, M., Martini S., 2010, "*Programming Languages: Principles and Paradigms*", Springer London Ltd, United Kingdom.
19. Juárez, E. y Rico, A., 1970, "*Mecánica de suelos*", México.
20. Juárez, M., Auvinet, G., Barranco, A. y Méndez, E., 2010, "*Contribución a la caracterización geoestadística del subsuelo del sur del valle de México*", Memorias técnicas, XXV Reunión Nacional de Mecánica de Suelos e Ingeniería Geotécnica, Sociedad Mexicana de Mecánica Ingeniería Geotécnica, Acapulco, Gro., México.

21. Juárez, Moisés, 2001 "*Aplicación de la Geoestadística a la descripción del Subsuelo del Valle de México*", Tesis de Maestría, ESIA, IPN, México.
22. Juárez, Moisés, 2014, "*Análisis Geostadístico del Subsuelo de la Zona Lacustre del Valle de México*", Tesis de Doctorado, DEPFI, UNAM, México.
23. Juárez, M. y Auvinet, G., 2002, "*Avances en la Caracterización Geoestadística del Subsuelo de la Zona Lacustre del Valle de México*" Memorias técnicas de la XXI Reunión Nacional de Mecánica de Suelos, Sociedad Mexicana de Mecánica de Suelos, Querétaro, México.
24. Kanalakis, John, 2003, "*Developing .NET Enterprise Applications*", Springer-Verlag, New York, USA.
25. Kedar y Seema, 2008, "*Programming Paradigms And Methodology*", Vikram Printers, India.
26. Kelkar, S., 2007, "*Software Engineering: A Concise Study*", Prentice-Hall, New Delhi, India.
27. Khoshafian, S., Abnous, R., "*Object Orientation.- Concepts, Language, Databases, User Interfaces*", John Wiley & Sons, USA.
28. Krige, D. G., 1951, "*Statistical application in mine valuation*", J. Institute Mine Survey, South Africa.
29. Leite, Mario, 2006, "*Técnicas de Programação – Uma Abordagem Moderna"*, Brasport Livros, Rio de Janeiro, Brasil.
30. Levitin, Anany, 2005, "*The Design & Analysis of Algorithms 2nd Edition*", Pearson Education, Boston, USA.
31. Marsal, R. J. y Mazari, M., 1969, "*El Subsuelo de la Ciudad de México*", Facultad de Ingeniería, UNAM, México.
32. Matheron, G., 1965, "*Les variables généralisées et leur estimation*", Masson et Cie, France.
33. McMillan, Michael, 2005, "*Data Structures and Algorithms Using Visual Basic. NET*", Cambridge University Press, New York, USA.
34. Medina, Z., 2001, "*Interpretación Geoestadística de Campañas de Reconocimiento del Subsuelo*", Tesis de Maestría, DEPFI, UNAM, México.
35. Meyer, Bertrand, 1988, "*Object-Oriented Software Construction*" Prentice-Hall, USA.
36. Microsoft, 2014, "*Lo más destacado de Visual Studio 2010*" Recuperado el 30 de Enero de 2014, de la fuente http://msdn.microsoft.com/es-es/library/vstudio/dd547188(v=vs.100).aspx
37. Mitchell, John, 2003, "*Concepts in Programming Languages*", Cambridge University Press, New York, USA.
38. Rodríguez, J., Santamaría, L. Rabasa, A., 2003, "*Introducción a la programación. Teoría y práctica*", Editorial Club Universitario, España.
39. Salvador, Omar, 2007, "*Paradigma de programación dirigido por eventos"*. Recuperado el 3 de Febrero de 2014, de la fuente http://www.osgg.net/omarsite/resources/papers/event_driven_p.pdf.
40. Sommerville, Ian, 2005, "*Ingeniería del Software, séptima edición*", Pearson Educación, Madrid, España.
41. Stroustrup, Bjarne, 2000, "*The C++ Programming Language*", Addison Wesley, USA.
42. Valencia, Diego, 2013, "*Software para construir repositorios digitales*", Facultad de Ingeniería, UNAM, México.
43. Vanmarcke, E. 1983, "*Random fields, analysis and synthesis"*, The Massachusetts Institute of Technology Press, Cambridge, Massachusetts, USA.

44. Vick, Paul, 2004, "*The Visual Basic .NET programming language*", Pearson Education, Boston, USA.
45. Winder, Russel, 1995, "*Desarrollo de software con C++*", Ed. Díaz de Santos, Madrid, España.

ANEXOS

Anexo I. Fundamento teórico de los Campos Aleatorios

Parámetros descriptivos

Sea $V(x)$ una variable geotécnica de interés definido en los puntos X del dominio R^0 (p = 1,2 ó 3). Si, en cada punto del dominio, esta variable se considera como aleatoria, el conjunto de estas variables aleatorias constituye un campo aleatorio. Para describir este campo pueden emplearse los siguientes parámetros (Auvinet, 2002):

-*Valor esperado*:

$$\mu_V(X) = E\{V(X)\} \tag{I.1}$$

-*Varianza*:

$$\sigma_V^2(X) = Var[V(X)] \tag{I.2}$$

La raíz cuadrada $\sigma_V(X)$ de la varianza se le suele llamar *desviación estándar* y el cociente $CV(X) = \sigma_V(X)/E\{V(X)\}$ se conoce como *coeficiente de variación*.

-*Función de autocorrelación*, definida sobre R^P x R^P:

$$R_V(X_1, X_2) = E\{V(X_1)\, V(X_2)\} \tag{I.3}$$

Esta función es un momento de segundo orden mixto que puede centrarse introduciendo el concepto de función de autocovarianza:

-*Función de autocovarianza*

En la misma forma, es posible estimar la covarianza (y a partir de ella el coeficiente de autocorrelación) a lo largo de la dirección **u** como:

$$C_V(\lambda \mathbf{u}) = \frac{1}{L}\int_0^L V(x)V(x + \lambda \mathbf{u})dx - \mu_V^2 \tag{I.4}$$

donde, **u** es el vector unitario en la dirección en la que se evalúa la covarianza, λ es un escalar. y L es la longitud del sondeo.

La función de autocovarianza representa el grado de dependencia lineal entre los valores de la propiedad de interés en dos puntos diferentes del medio.

$$\begin{aligned} C_V(X_1, X_2) &= Cov[V(X_1), V(X_2)] \\ &= E\{[V(X_1) - \mu_V(X_1)][V(X_2) - \mu_V(X_2)] \end{aligned} \tag{I.5}$$

Su estimador estadístico es:

-*Coeficiente de autocorrelación*

La función de autocovarianza se puede escribir bajo la forma de un *coeficiente de autocorrelación* adimensional, cuyo valor queda siempre comprendido entre -1 y +1:

$$\rho_V(X_1, X_2) = \frac{C_V(X_1, X_2)}{\sigma_V(X_1)\sigma_V(X_2)} \qquad \text{I.6}$$

-*Variograma*

Es el momento estadístico de segundo orden del incremento $V(X)$-$V(X+h)$, es una herramienta generalmente equivalente a la función de autocovarianza, se estima de la siguiente forma:

$$2\gamma(\lambda\mathbf{u}) \cong \frac{1}{L}\int_0^L [V(X + \lambda\mathbf{u}) - \mathrm{V(X)}]^2\, dx \qquad \text{I.7}$$

El estimador estadístico está dado por la semivarianza:

$$\gamma(h) = \gamma(X_1, X_2) = \frac{1}{2N}\sum_{i=1}^{N}[V(X_1) - V(X_2)]^2 \qquad \text{I.8}$$

-*Correlograma experimental*

La función de autocovarianza puede ser normalizada y expresada mediante un coeficiente, para lo cual se estima un valor de ρ(λ**u**) para cada valor de $C(\lambda\mathbf{u})$. Con los valores obtenidos se construye una gráfica con ρ(λ**u**) y $h = \lambda\mathbf{u}$ en un sistema coordenado; en este trabajo la curva resultante es llamada *correlograma experimental*.

-*Distancia de correlación*

Para estimar la correlación espacial en forma cuantitativa se introduce el término *distancia de correlación* (también conocida como *alcance*, *influencia* o *rango*), que es la distancia a partir de la cual las variables aleatorias regionalizadas $V(X_1)$ y $V(X_2)$ son independientes para cualquier $X \in \Omega \subset R^p$, de aquí que se interprete como *zona de influencia*.

La distancia de correlación, $\delta=2a$, se estima a partir del correlograma experimental, definiéndose a como:

$$a = \int_0^{\lambda_C} \rho(\lambda\boldsymbol{u})d\lambda \qquad \text{I.9}$$

siendo: λ_C es el valor crítico de λ donde ρ se anula por primera vez.

Correlograma teórico

Los correlogramas teóricos se definen ajustando los correlogramas experimentales a una función matemática que incluya las correspondientes distancias de correlación. En la *tabla I.1* se describen las funciones comúnmente empleadas.

Tabla I.1. Funciones del coeficiente de autocorrelación (Vanmarcke, 1983)

Tipo	Función
1. Exponencial	$\exp - \left\{\left(\frac{d_x}{x}\right)^k + \left(\frac{d_z}{z}\right)^k\right\}$
I. Exponencial simple	$\exp\left\{-2\left(\frac{d_x}{\delta_x} + \frac{d_z}{\delta_z}\right)\right\}$
II. Exponencial cuadrada	$\exp\left\{-\pi\left(\frac{d_x{}^2}{\delta_x{}^2} + \frac{d_z{}^2}{\delta_z{}^2}\right)\right\}$
III. Modelo autoregresivo de segundo orden	$\exp\left\{-4\left(\frac{d_x}{\delta_x} + \frac{d_z}{\delta_z}\right)\right\}\left(1 + \frac{4d_x}{\delta_x}\right)\left(1 + \frac{4d_y}{\delta_y}\right)$
IV. Exponencial cosenoidal	$\exp\left\{-\left(\frac{d_x}{\delta_x} + \frac{d_z}{\delta_z}\right)\right\}\cos\frac{d_x}{\delta_x}\cos\frac{d_z}{\delta_z}$

Se ha demostrado que una función de tipo exponencial simple es la que mejor se ajusta para las propiedades del subsuelo (Juárez, 2014). Eventualmente esta distancia de correlación puede ser modificada ligeramente para mejorar el ajuste de la función teórica con el correlograma experimental.

Estimación

Considérese una función $V(X)$: $R^P \rightarrow R$ que representa una propiedad aleatoria del medio. Se suponen conocidos los valores de $V(X)$: $V_1, V_2, \ldots, V_n$ en n puntos $X_1, X_2, \ldots, X_n$.

El problema consiste en estimar:
a) el valor $V(X)$ en cualquier punto X (estimación puntual):
b) o bien, el valor medio de V sobre un dominio de R^P (estimación global).

El estimador $V^*(X)$ de $V(X)$ en un punto X será un campo aleatorio, función de X y de la información disponible, es decir de los valores $V_i = V(X_i)$, $i = 1$ a n.

La técnica de kriging consiste en recurrir a estimadores que sean combinaciones lineales de los datos:

$$V^*(X) = \sum_{i=1}^{n} \lambda_i V_i \qquad \text{I .10}$$

Donde los λ_i son reales.

El problema consiste en estimar los parámetros λ_i que den un estimador satisfactorio de $V(X)$. El kriging recurre al *"mejor estimador sin sesgo"*; es decir, a un estimador que tenga las dos propiedades siguientes:

1. Ausencia de sesgo: $E\{V(X) - V^*(X)\} = 0$
2. Mínimo valor de la varianza de estimación definida como:

$$\sigma_E^2 = Var[V(X) - V^*(X)] = E\left\{\left(V(X) - V^*(X)\right)^2\right\}$$

Introduciendo la variable aleatoria *"error"* definida como $V^*(X) - V(X)$, los dos criterios anteriores implican que esta variable tiene:

- una esperanza nula
- una dispersión mínima alrededor de esta esperanza

Kriging *Ordinario*

La técnica del *Kriging ordinario* (Krige, 1962; Matheron, 1965; Vanmarcke, 1983; Deutsch, 1992; Auvinet, 2002) considera que no se conoce la media μ_V del campo aleatorio. Esto permite generalizar el kriging a situaciones donde esta media no es constante en el espacio: la media puede variar de una región a otra, siempre que sea aproximadamente constante en cada vecindad de *Kriging*. Sólo se conoce la función de covarianza $C(h)$ o el variograma $\gamma(h)$.

Cuando el campo es estacionario, $V^*(X)$ puede escribirse:

$$V^*(X) = \sum_{i=1}^{n} \lambda_i V_i + \left[1 - \sum_{i=1}^{n} \lambda_i\right] \mu_V \qquad \text{I .11}$$

donde μ_v es la esperanza constante del campo.

Partiendo de esta ecuación es posible encontrar un estimador lineal, sin sesgo y de mínima varianza que no requiera el conocimiento de la media μ_v, imponiendo la condición:

$$\sum_{i=1}^{n} \lambda_i = 1 \qquad \text{I .12}$$

Condición de ausencia de sesgo: El valor esperado del error de estimación es:

$$E\{V^*(X) - V(X)\} = a + \sum_{i=1}^{n} \lambda_i E\{V(X_i)\} - \{V(X)\} = a + \sum_{i=1}^{n} \lambda_i \mu_V - \mu_V \qquad \text{I .13}$$

Siendo μ_v desconocido, para que este *valor esperado* sea nulo se debe plantear

$$a = 0 \qquad \sum_{i=1}^{n} \lambda_i = 1 \qquad \text{I .14}$$

Condición de varianza mínima: La varianza del error de estimación se expresa en función de la covarianza.

$$var[V^*(X) - V(X)] = \sum_{i=1}^{n}\sum_{j=1}^{n} \lambda_i \lambda_j C(X_i - X_j) - 2\sum_{i=1}^{n} \lambda_i C(X - X_i) + C(0) \qquad \text{I .15}$$

El sistema de ecuaciones del *kriging ordinario* (KO).

$$\begin{cases} \sum_{j=1}^{n} \lambda_j C(X_i - X_j) + v = C(X - X_i), i = 1, \dots n \\ \sum_{i=1}^{n} \lambda_i = 1 \end{cases} \qquad \text{I .16}$$

En notación matricial:

$$\begin{pmatrix} C(X_1 - X_1) & \cdots & C(X_1 - X_n) & 1 \\ \vdots & \ddots & \vdots & \vdots \\ C(X_n - X_1) & \cdots & C(X_n - X_n) & 1 \\ 1 & \dots & 1 & 0 \end{pmatrix} \begin{pmatrix} \lambda_1 \\ \vdots \\ \lambda_n \\ v \end{pmatrix} = \begin{pmatrix} C(X_1 - X) \\ \vdots \\ C(X_n - X) \\ 1 \end{pmatrix} \qquad \text{I .17}$$

Simulación

.La simulación se realiza generalmente sobre una malla de puntos en el dominio de interés y se reduce por tanto, a generar un cierto número de variables aleatorias conjuntamente distribuidas.

Simulación condicional

Se supone que el campo aleatorio $V(X)$ ha sido medido en los puntos $X_1, X_2, \dots, X_p$ y que será simulado en los puntos $X_{p+1}, X_{p+2}, \dots, X_{p+n}$. Se desea general realizaciones de $V(X)$ que igualen de manera exacta los datos en *p* puntos y que sean aleatorios en los *n-p* puntos restantes. La *simulación de un campo aleatorio condicional* se lleva a cabo dando los siguientes pasos:

1) Con los datos conocidos, se calcula por el método del *kriging* el valor estimado en los puntos desconocidos $X_{p+1}, X_{p+2}, \dots, X_{p+n}$. Se denomina a este campo $V^*(X)$. En los puntos conocidos, el campo $V^*(X)$ es igual a los datos.

2) Se genera una realización incondicional del campo aleatorio, respetando la media, la varianza y la estructura de correlación del mismo, de acuerdo con lo indicado en el inciso anterior; y se denomina este campo $V^{**}(X)$.

3) Se realiza la estimación por *kriging* del campo en los puntos desconocidos, utilizando como datos los valores determinados en el paso anterior en los puntos $X_1, X_2, \dots, X_p$, es decir, se genera un campo con el *kriging* de la simulación incondicional. Se denomina a este campo $V^{**}(X)$.

4) Se combinan los tres campos para generar la realización condicional $V^c(X_i)$ de acuerdo con:

$$V^c(X_i) = V^*(X_i) + [V^{**}(X_i) - V^{**}(X_i)] \qquad \text{I .18}$$

En los puntos conocidos $V^{**}(X)$ es igual a $V^{***}(X)$, por tanto, el campo simulado se ajusta exactamente a los valores conocidos. Por otra parte, el término $[V^{**}(X_i) - V^{***}(X_i)]$ representa una desviación aleatoria que, sumada a los valores estimados por *kriging*, sirve

para estimar $V^c(X)$ de tal manera que la esperanza de $V^c(X)$ sea $V^*(X)$, con un incremento en la varianza lejos de los puntos conocidos.

Anexo II. Análisis de Regresión Lineal

II.1 Regresión Lineal

Caso bidimensional

Sean (x_p, y_p) las coordenadas de los puntos M_p.

Se quiere encontrar el plano de la ecuación:

$$V = a + bx + cy$$

Que representa lo mejor posible la tendencia general de los puntos de datos. Los reales *a,b* y *c* se eligen de manera que si $e_p = V_p - ax - by - c$, la suma $\sum e_p^2$ sea mínima.

El sistema lineal se escribe:

$$\begin{bmatrix} K_{xx} & K_{xy} \\ K_{xy} & K_{yy} \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} C_x \\ C_y \end{bmatrix}$$

$$c = \overline{V} - a.\overline{x} - b.\overline{y}$$

Con:

$$\overline{x} = \frac{1}{n}\sum_{p=0}^{n} x_p$$

$$\overline{y} = \frac{1}{n}\sum_{p=0}^{n} y_p$$

$$\overline{V} = \frac{1}{n}\sum_{p=0}^{n} V_p$$

$$K_{xx} = \frac{1}{n}\sum_{p=0}^{n} (x_p - \overline{x})^2$$

$$K_{yy} = \frac{1}{n}\sum_{p=0}^{n} \left(y_p - \overline{y}\right)^2$$

$$K_{xy} = \frac{1}{n}\sum_{p=0}^{n} (x_p - \overline{x})(y_p - \overline{y})$$

$$C_x = \frac{1}{n}\sum_{p=0}^{n} \left(x_p - \overline{x}\right)\left(V_p - \overline{V}\right)$$

$$C_y = \frac{1}{n}\sum_{p=0}^{n} (y_p - \overline{y})(V_p - \overline{V})$$

Caso tridimensional

Sean (x_p, y_p, z_p) las coordenadas de los puntos M_p.

Se quiere encontrar el plano de la ecuación:

$$V = ax + by + cz + d$$

Que representa lo mejor posible la tendencia general de los datos. Los reales *a*, *b*, *c* y *d* se eligen de manera que, si $e_p = V_p - ax_p - by_p - cz_p - d$, la suma $\sum e_p^2$ sea mínima.

El sistema lineal se escribe:

$$\begin{bmatrix} K_{xx} & K_{xy} & K_{xz} \\ K_{xy} & K_{yy} & K_{yz} \\ K_{xz} & K_{yz} & K_{zz} \end{bmatrix} \begin{bmatrix} a \\ b \\ z \end{bmatrix} = \begin{bmatrix} C_x \\ C_y \\ C_z \end{bmatrix}$$

$$d = \overline{V} - a.\overline{x} - b.\overline{y} - c.\overline{z}$$

Con:

$$\overline{x} = \frac{1}{n}\sum_{p=0}^{n} x_p$$

$$\overline{y} = \frac{1}{n}\sum_{p=0}^{n} y_p$$

$$\overline{z} = \frac{1}{n}\sum_{p=0}^{n} z_p$$

$$\overline{V} = \frac{1}{n}\sum_{p=0}^{n} V_p$$

$$K_{xx} = \frac{1}{n}\sum_{p=0}^{n} (x_p - \overline{x})^2$$

$$K_{yy} = \frac{1}{n}\sum_{p=0}^{n} \left(y_p - \overline{y}\right)^2$$

$$K_{zz} = \frac{1}{n}\sum_{p=0}^{n} \left(z_p - \overline{z}\right)^2$$

$$K_{xy} = \frac{1}{n}\sum_{p=0}^{n} (x_p - \overline{x})(y_p - \overline{y})$$

$$K_{xz} = \frac{1}{n}\sum_{p=0}^{n} \left(x_p - \overline{x}\right)\left(z_p - \overline{z}\right)$$

$$K_{yz} = \frac{1}{n}\sum_{p=0}^{n} (y_p - \overline{y})(z_p - \overline{z})$$

$$C_x = \frac{1}{n}\sum_{p=0}^{n} \left(x_p - \overline{x}\right)\left(V_p - \overline{V}\right)$$

$$C_y = \frac{1}{n}\sum_{p=0}^{n} (y_p - \overline{y})(V_p - \overline{V})$$

$$C_z = \frac{1}{n}\sum_{p=0}^{n} (z_p - \overline{z})(V_p - \overline{V})$$

II.2 Resultados de la regresión lineal obtenidos en la etapa de validación

Resultados del análisis geoestadístico de la distribución espacial de la profundidad de la capa dura en la ciudad de México, análisis en 2D.

En las *tablas II.1* e *II.2* se muestran los resultados obtenidos con MESCAL y con Excel.

Tabla II.1 Resultados regresión lineal (MESCAL).

Cx	Cy	Profundidad [m]	Residuos	Regresión
483520.1	2148501.42	-31.2	-1.38228164537527	-29.8177183546247
484536.1	2147720.43	-35	-2.22180230287222	-32.7781976971278
484780.9	2148723.03	-31	1.18779443313679	-32.1877944331368
484156.2	2150299.91	-28	1.16681342924949	-29.1668134292495
483569.1	2148690.93	-31	-1.28822515546699	-29.711774844533
483428.3	2147362.79	-34	-3.12589079318377	-30.8741092068162
484300.5	2147842.69	-34	-1.84388250426309	-32.1561174957369
482727.6	2148438.75	-27	1.24406663726245	-28.2440666372624
484525.4	2148756.65	-32	-0.378476601320926	-31.6215233986791
485272.1	2148713.86	-33.2	0.0157280667392996	-33.2157280667393
484409.1	2147870.2	-33	-0.648954323797852	-32.3510456762021
483937.8	2148579.36	-30	0.597968892152494	-30.5979688921525
484048	2148188.1	-32.4	-1.14529873389156	-31.2547012661084
483148.3	2146668.92	-32	-0.946442811520228	-31.0535571884798
482744.4	2145577.7	-26.7	4.71128919125777	-31.4112891912578
482727.6	2147162.58	-32.5	-2.85872399046207	-29.6412760095379
482781.1	2147984.83	-28.7	0.15190493217715	-28.8519049321771
484343.3	2148871.28	-31	0.118662757432048	-31.118662757432
483763.4	2148483.08	-31.5	-1.15802142150756	-30.3419785784924
486441	2149047.04	-33.5	1.77321154479296	-35.273211544793
482634.2	2147916.05	-28.2	0.422793496846726	-28.6227934968467
482758.2	2147576.76	-25	4.25122391623972	-29.2512239162397
482707.7	2146534.42	-29	1.28777638573865	-30.2877763857387
483319.7	2147268.03	-32	-1.24719059319432	-30.7528094068057
484982.9	2148000.11	-33.5	-0.102123204956115	-33.3978767950439
483977.6	2148600.76	-31	-0.342984877471736	-30.6570151225283
487212.2	2144992.31	-35	6.31064225518412	-41.3106422551841
486001.9	2147772.4	-36	-0.241183371320858	-35.7588166286791
485718.9	2147804.49	-35.4	-0.262766429690281	-35.1372335703097
484911	2147890.07	-33	0.369358164927689	-33.3693581649277
484481	2147928.28	-33.5	-1.06354750559967	-32.4364524944003
483797.1	2147688.33	-32.6	-1.31805756988124	-31.2819424301188
482940.2	2147426.98	-32.4	-2.60763883838754	-29.7923611616125
482905	2147769.33	-37.4	-8.05540283671362	-29.3445971632864
484569.8	2147711.26	-34.7	-1.84192741954276	-32.8580725804572
485445	2148085.7	-36.7	-2.43823991322866	-34.2617600867713
484669.2	2148548.79	-31.4	0.747088973142219	-32.1470889731422
483021.3	2146888.99	-41.4	-10.8505624926096	-30.5494375073904
482592.9	2148533.51	-25.8	2.06118568548818	-27.8611856854882
483846	2148509.06	-31.2	-0.715296875608328	-30.4847031243917
484283.6	2148264.52	-31.3	0.359257502758918	-31.6592575027589
483076.4	2148133.08	-29.4	-0.0984677094822004	-29.3015322905178
485627	2148585.47	-34.2	-0.108259761160255	-34.0917402388397
484118.4	2148134.61	-31	0.459151641539393	-31.4591516415394
483843	2148001.64	-32	-0.965966995405324	-31.0340330045947
485484.8	2146936.39	-37	-1.39744674928033	-35.6025532507197
485325.6	2148328.71	-34.8	-1.05172618442593	-33.7482738155741
482664.8	2145782.5	-25.5	5.52211290893388	-31.0221129089339
483716	2144995.37	-31.9	2.16225814131904	-34.062258141319
483686.9	2145206.28	-32.4	1.37104145907424	-33.7710414590742
483783.3	2145929.19	-34	-0.820667700644208	-33.1793322993558
483818.5	2146380.06	-32.6	0.158642707425905	-32.7586427074259
484031.2	2146474.81	-31.8	1.29567567644854	-33.0956756764485
484219.4	2146667.39	-34	-0.725170479036478	-33.2748295209635
484254.6	2146977.64	-34.8	-1.79190285372287	-33.0080971462771
484243.9	2147214.54	-34.6	-1.87344500249888	-32.7265549975011

484487.1	2148461.68	-30.4	1.4651025248023	-31.8651025248023
484569.8	2148906.43	-31.4	0.149545718469291	-31.5495457184693
484632.5	2149152.5	-31.4	0.0100676856082842	-31.4100676856083
484770.2	2149627.81	-29.6	1.57502666743166	-31.1750266674317
485630.1	2145058.03	-35.5	2.460167133494	-37.960167133494
485787.7	2146042.3	-38	-0.790868640804547	-37.2091313591955
485963.7	2147274.15	-35.6	0.625163236207889	-36.2251632362079
486092.2	2148088.76	-36.4	-0.800423751936613	-35.5995762480634
485933.1	2148599.23	-36.4	-1.68900618708212	-34.7109938129179
485959.1	2148788.75	-36.2	-1.64262261559061	-34.5573773844094
485547.5	2148816.25	-34.4	-0.725672911568289	-33.6743270884317
485241.5	2148866.7	-33.2	-0.215019308895265	-32.9849806911047
485142	2148895.73	-31.6	1.14700749079397	-32.747007490794
484614.1	2149054.68	-32	-0.520964073613868	-31.4790359263861
483774.1	2149230.44	-30.2	-0.654092205770393	-29.5459077942296
483673.1	2149305.33	-29.8	-0.545383424090961	-29.254616575909
483090.2	2149517.77	-22.6	5.214107433348	-27.814107433348
482563.8	2149731.74	-19.2	7.28900584203739	-26.4890058420374
482597.5	2146164.56	-26.3	4.1643530516646	-30.4643530516646
482889.7	2145157.37	-27.3	4.87258504347228	-32.1725850434723
485252.2	2148166.71	-34	-0.226465015805843	-33.7735349841942
485103.8	2147674.6	-34.8	-0.79520376541727	-34.0047962345827
484505.5	2148759.71	-30.5	1.07693520520002	-31.5769352052
483093.2	2147616.5	-39.5	-9.59807830233967	-29.9019216976603
484259.8	2150175.7	-27	2.51749031758845	-29.5174903175885
486089.1	2148840.71	-35.8	-1.03011706042462	-34.7698829395754
483016.7	2147784.62	-33.3	-3.74067163585168	-29.5593283641483
483162.1	2147738.76	-32.5	-2.58915546666299	-29.910844533337
484488.3	2150213.8	-27	2.94928800262551	-29.9492880026255
482507.2	2149271.71	-20	6.8753778668929	-26.8753778668929
486613.9	2150091.78	-32	2.48767601592454	-34.4876760159245
483712.9	2149286.99	-29.5	-0.142828021383821	-29.3571719786162

Tabla II.2. Resultados regresión lineal (Excel).

Coor_X	*Coor_Y*	*Prof_CD*	*RL*	*Residuos*
483520.10	2148501.42	-31.2	-29.8	-1.4
484536.10	2147720.43	-35.0	-32.8	-2.2
484780.90	2148723.03	-31.0	-32.2	1.2
484156.20	2150299.91	-28.0	-29.2	1.2
483569.10	2148690.93	-31.0	-29.7	-1.3
483428.30	2147362.79	-34.0	-30.9	-3.1
484300.50	2147842.69	-34.0	-32.2	-1.8
482727.60	2148438.75	-27.0	-28.2	1.2
484525.40	2148756.65	-32.0	-31.6	-0.4
485272.10	2148713.86	-33.2	-33.2	0.0
484409.10	2147870.20	-33.0	-32.4	-0.6
483937.80	2148579.36	-30.0	-30.6	0.6
484048.00	2148188.10	-32.4	-31.3	-1.1
483148.30	2146668.92	-32.0	-31.1	-0.9
482744.40	2145577.70	-26.7	-31.4	4.7
482727.60	2147162.58	-32.5	-29.6	-2.9
482781.10	2147984.83	-28.7	-28.9	0.2
484343.30	2148871.28	-31.0	-31.1	0.1
483763.40	2148483.08	-31.5	-30.3	-1.2
486441.00	2149047.04	-33.5	-35.3	1.8

482634.20	2147916.05	-28.2	-28.6	0.4
482758.20	2147576.76	-25.0	-29.3	4.3
482707.70	2146534.42	-29.0	-30.3	1.3
483319.70	2147268.03	-32.0	-30.8	-1.2
484982.90	2148000.11	-33.5	-33.4	-0.1
483977.60	2148600.76	-31.0	-30.7	-0.3
487212.20	2144992.31	-35.0	-41.3	6.3
486001.90	2147772.40	-36.0	-35.8	-0.2
485718.90	2147804.49	-35.4	-35.1	-0.3
484911.00	2147890.07	-33.0	-33.4	0.4
484481.00	2147928.28	-33.5	-32.4	-1.1
483797.10	2147688.33	-32.6	-31.3	-1.3
482940.20	2147426.98	-32.4	-29.8	-2.6
482905.00	2147769.33	-37.4	-29.3	-8.1
484569.80	2147711.26	-34.7	-32.9	-1.8
485445.00	2148085.70	-36.7	-34.3	-2.4
484669.20	2148548.79	-31.4	-32.1	0.7
483021.30	2146888.99	-41.4	-30.5	-10.9
482592.90	2148533.51	-25.8	-27.9	2.1
483846.00	2148509.06	-31.2	-30.5	-0.7
484283.60	2148264.52	-31.3	-31.7	0.4
483076.40	2148133.08	-29.4	-29.3	-0.1
485627.00	2148585.47	-34.2	-34.1	-0.1
484118.40	2148134.61	-31.0	-31.5	0.5
483843.00	2148001.64	-32.0	-31.0	-1.0
485484.80	2146936.39	-37.0	-35.6	-1.4
485325.60	2148328.71	-34.8	-33.7	-1.1
482664.80	2145782.50	-25.5	-31.0	5.5
483716.00	2144995.37	-31.9	-34.1	2.2
483686.90	2145206.28	-32.4	-33.8	1.4
483783.30	2145929.19	-34.0	-33.2	-0.8
483818.50	2146380.06	-32.6	-32.8	0.2
484031.20	2146474.81	-31.8	-33.1	1.3
484219.40	2146667.39	-34.0	-33.3	-0.7
484254.60	2146977.64	-34.8	-33.0	-1.8
484243.90	2147214.54	-34.6	-32.7	-1.9
484487.10	2148461.68	-30.4	-31.9	1.5
484569.80	2148906.43	-31.4	-31.5	0.1
484632.50	2149152.50	-31.4	-31.4	0.0
484770.20	2149627.81	-29.6	-31.2	1.6
485630.10	2145058.03	-35.5	-38.0	2.5
485787.70	2146042.30	-38.0	-37.2	-0.8
485963.70	2147274.15	-35.6	-36.2	0.6
486092.20	2148088.76	-36.4	-35.6	-0.8
485933.10	2148599.23	-36.4	-34.7	-1.7
485959.10	2148788.75	-36.2	-34.6	-1.6
485547.50	2148816.25	-34.4	-33.7	-0.7
485241.50	2148866.70	-33.2	-33.0	-0.2
485142.00	2148895.73	-31.6	-32.7	1.1
484614.10	2149054.68	-32.0	-31.5	-0.5

483774.10	2149230.44	-30.2	-29.5	-0.7
483673.10	2149305.33	-29.8	-29.3	-0.5
483090.20	2149517.77	-22.6	-27.8	5.2
482563.80	2149731.74	-19.2	-26.5	7.3
482597.50	2146164.56	-26.3	-30.5	4.2
482889.70	2145157.37	-27.3	-32.2	4.9
485252.20	2148166.71	-34.0	-33.8	-0.2
485103.80	2147674.60	-34.8	-34.0	-0.8
484505.50	2148759.71	-30.5	-31.6	1.1
483093.20	2147616.50	-39.5	-29.9	-9.6
484259.80	2150175.70	-27.0	-29.5	2.5
486089.10	2148840.71	-35.8	-34.8	-1.0
483016.70	2147784.62	-33.3	-29.6	-3.7
483162.10	2147738.76	-32.5	-29.9	-2.6
484488.30	2150213.80	-27.0	-29.9	2.9
482507.20	2149271.71	-20.0	-26.9	6.9
486613.90	2150091.78	-32.0	-34.5	2.5
483712.90	2149286.99	-29.5	-29.4	-0.1

Anexo III. Creación de archivos

Creación de archivos para 2D

Para realizar el análisis 2D el programa necesita archivos con extensión **.po*, para crearlos se realiza el siguiente procedimiento.

1. En la ventana del menú principal se debe seleccionar la opción *Archivo*, después la opción *Nuevo* (*figura III.1*). También desde el teclado se puede oprimir la teclas Ctrl + N para accesar a la ventana *Nuevo*.

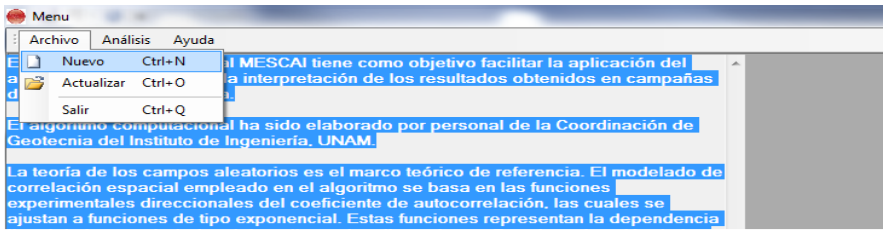


Figura III.1 Selección para la generación de archivos nuevos.

2. Una nueva ventana se despliega (*figura III.2*) donde se presenta la opción de escoger el origen de los datos, si son desde teclado o desde portapapeles.

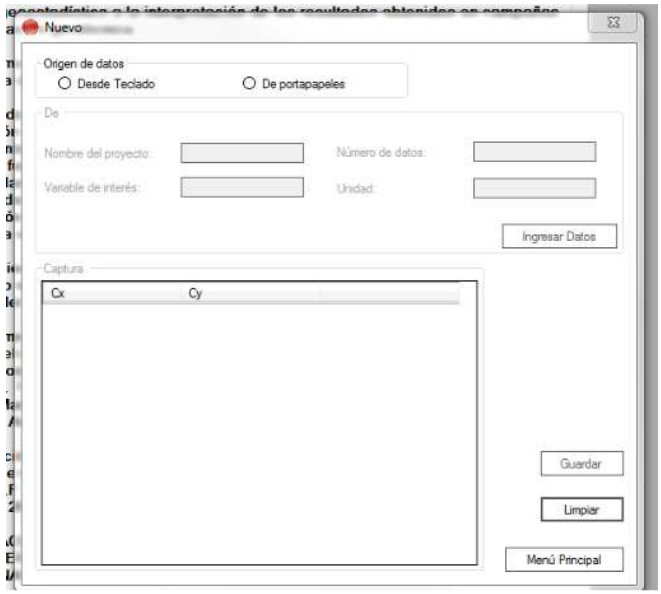


Figura III.2 Ventana para captura de datos.

3. De acuerdo con de la opción seleccionada se habilitan las cajas de texto para ingresar los datos del parámetro por analizar. La caja de texto de *Número de datos* se habilitara o deshabilitara según sea el caso (*figura III.3*).

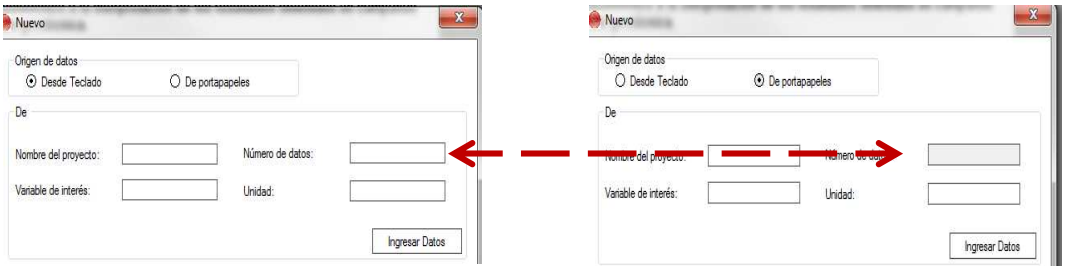


Figura III.3 Vista de las opciones de la forma de ingresar datos.

4. Al terminar de ingresar los datos del parámetro analizado se debe seleccionar la opción *Ingresar Datos*, según sea el caso, los datos se pueden ingresar de dos formas:

- Manualmente: Los datos del proyecto se ingresan y se selecciona la opción *Ingresar Datos* (*figura III.4*), posteriormente se llena la tabla con los datos del proyecto. En caso de quedar celdas vacías, al intentar guardar el proyecto aparecerá un mensaje de advertencia y no permitirá guardar (*figura III.5*).

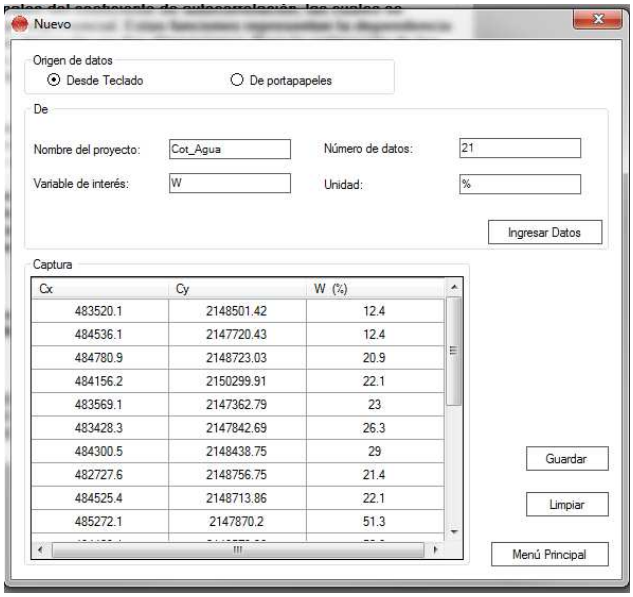


Figura III.4 Ingreso de datos de forma manual.

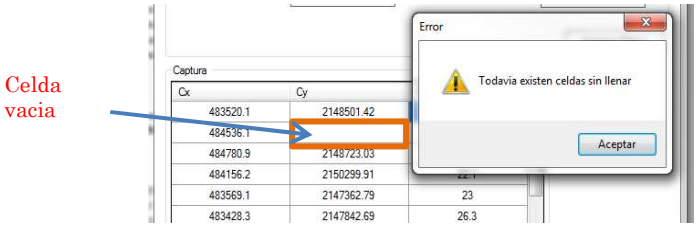


Figura III.5 Mensaje de error, celdas con celdas vacías.

- De portapapeles: se hace clic en el botón derecho del mouse sobre la tabla, se despliega un menú donde se selecciona la opción de pegar; habiendo en una hoja de Excel copiado tres columnas correspondientes a la los datos: coordenada *X*, coordenada *Y*, y *variable*; se pegaran en la tabla los datos copiados y se actualizara el valor de la caja de texto *Número de datos* (*figura III.6*).

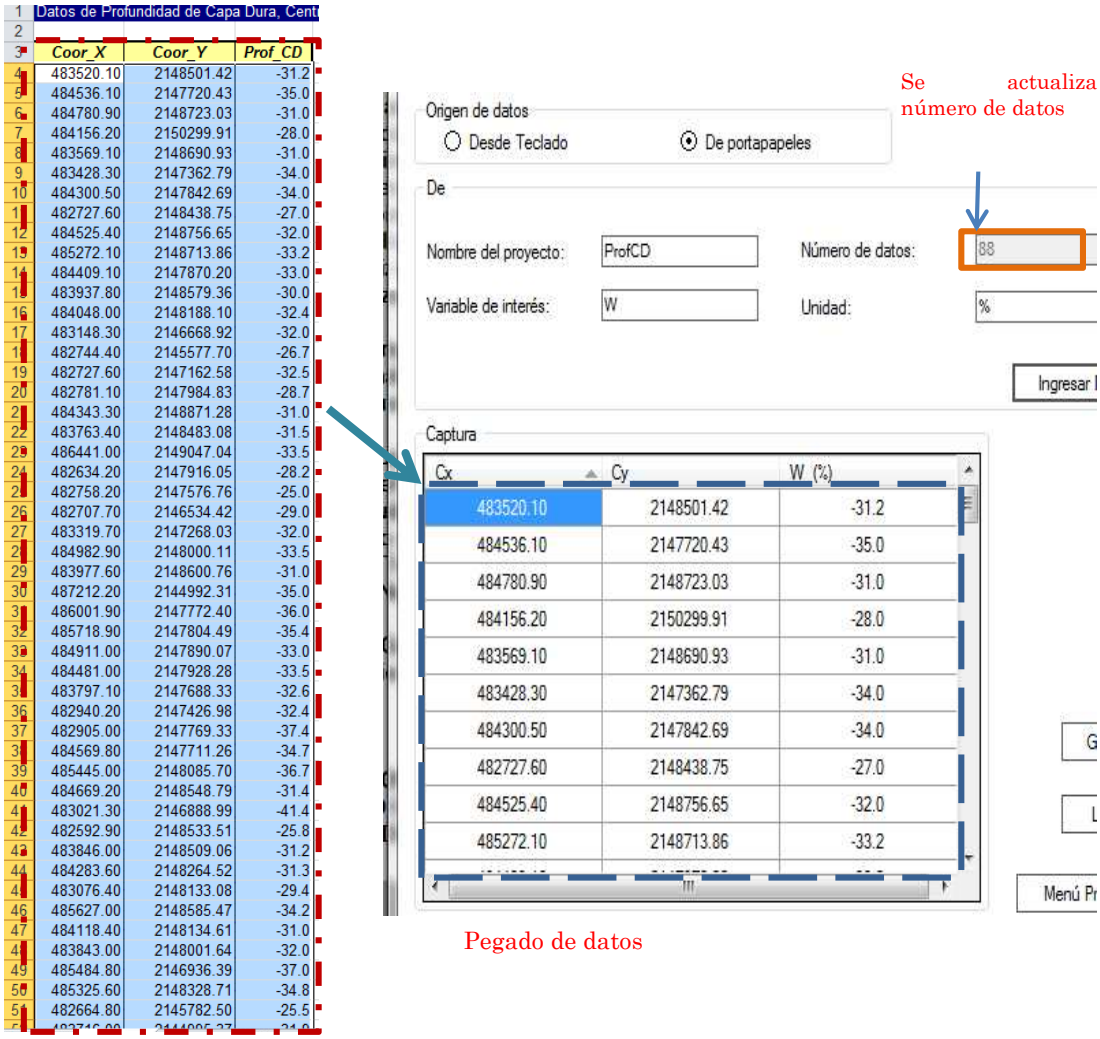


Figura III.6 Ingreso de datos desde portapeles.

5. Cuando se llena todo el formulario con los datos necesarios, y si no falta ningún campo, seleccionar la opción *Guardar*, aparece entonces una ventana (*figura III.7*) donde se asignara nombre al nuevo archivo.

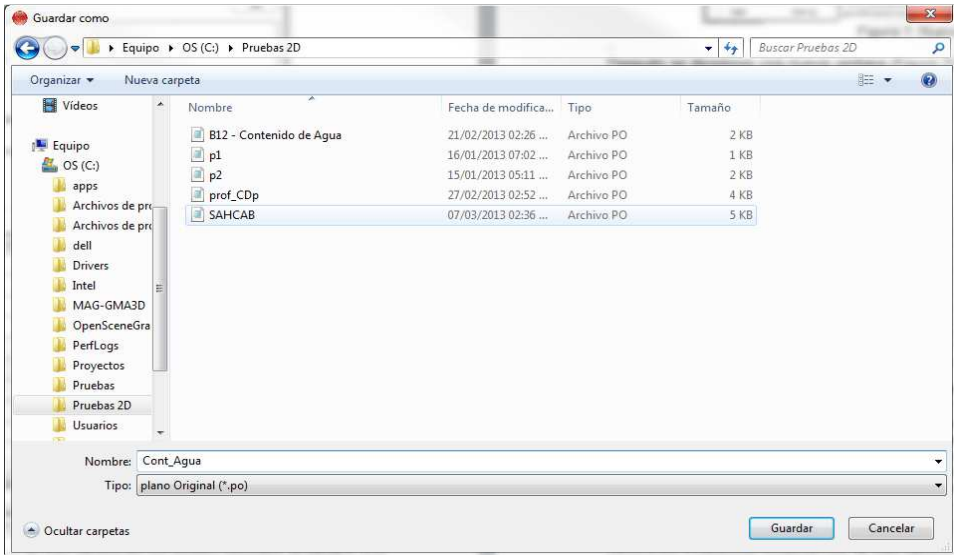


Figura III.7 Guardar como.

6. Si todo se realiza satisfactoriamente se desplegará un mensaje indicando que se ha generado un archivo con la ubicación del mismo (*figura III.8*)

.

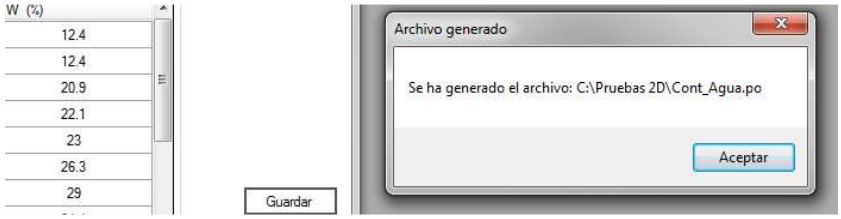


Figura III.8 Generación archivo de proyecto nuevo.

Creación de archivos para 3D

Para realizar el análisis 3D el programa necesita archivos de extensión ***.so***, para crearlos se realiza el siguiente procedimiento.

1. En la ventana del menú principal debe seleccionar la opción *Archivo*, después la opción *Nuevo* (*figura III.9*). También desde el teclado podrá oprimir la teclas Ctrl + N para accesar a la ventana *Nuevo*.

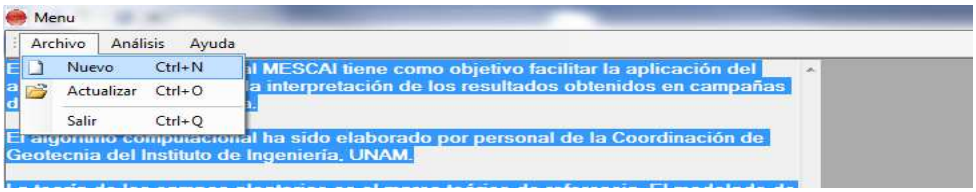


Figura III.9 Selección para generación de archivos nuevos.

2. Una nueva ventana se despliega donde se tendrá la opción de escoger el origen de los datos (*figura III.10*), si son desde teclado o desde portapapeles.

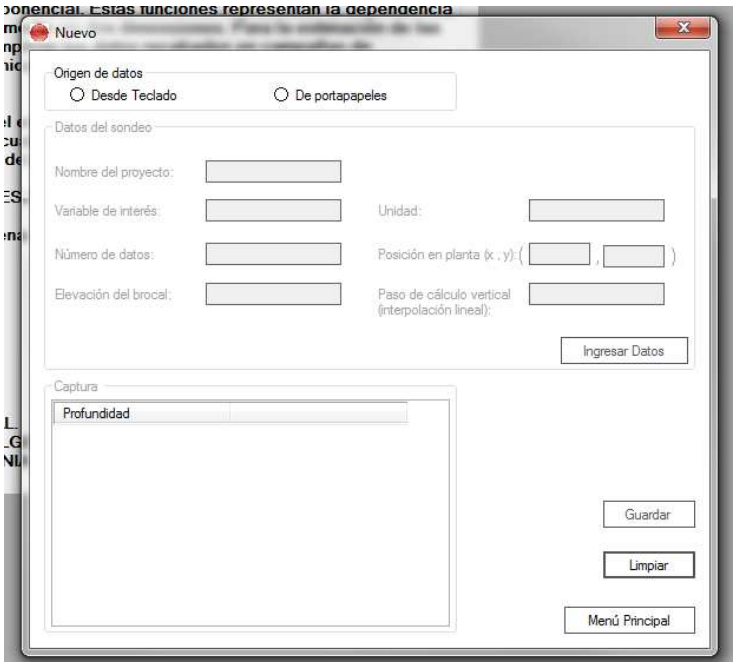


Figura III.10 Ventana para captura de datos.

3. Dependiendo de la opción seleccionada se habilitaran las cajas de texto para ingresar los datos del parámetro que se analiza. La caja de texto de *Número de datos* se habilitara o deshabilitara según sea el caso (*figura III.11*).

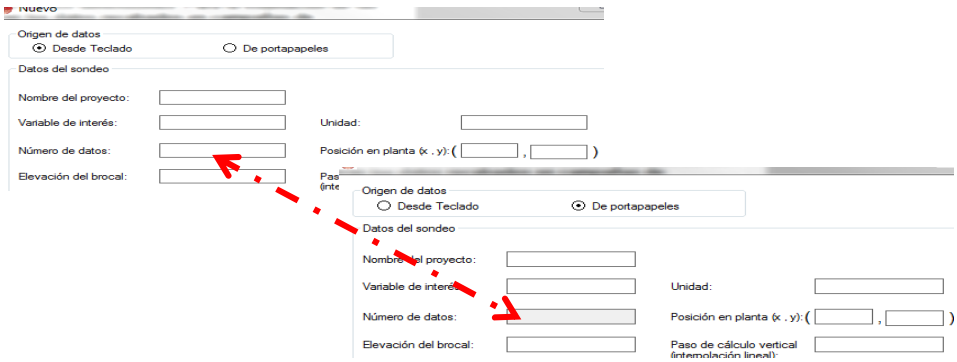


Figura III.11 Vista de las opciones de la forma de ingresar datos.

4. Obligatoriamente debe ingresar todos los datos a excepción de *Elevación del brocal* y *Paso de cálculo vertical*, esos son opcionales si se cuenta con tales datos. Al terminar de ingresar los datos del sondeo y seleccionar la opción de *Ingresar Datos* según sea el caso se podrán ingresar los datos de dos formas:

- Manualmente: Después de seleccionar *Ingresar Datos*, llenará la tabla con los datos de profundidad y la variable de interés (*figura III.12*). En caso de no llenar todas las celdas y querer guardar el proyecto aparecerá un mensaje de advertencia y no permitirá guardar (*figura III.13*).

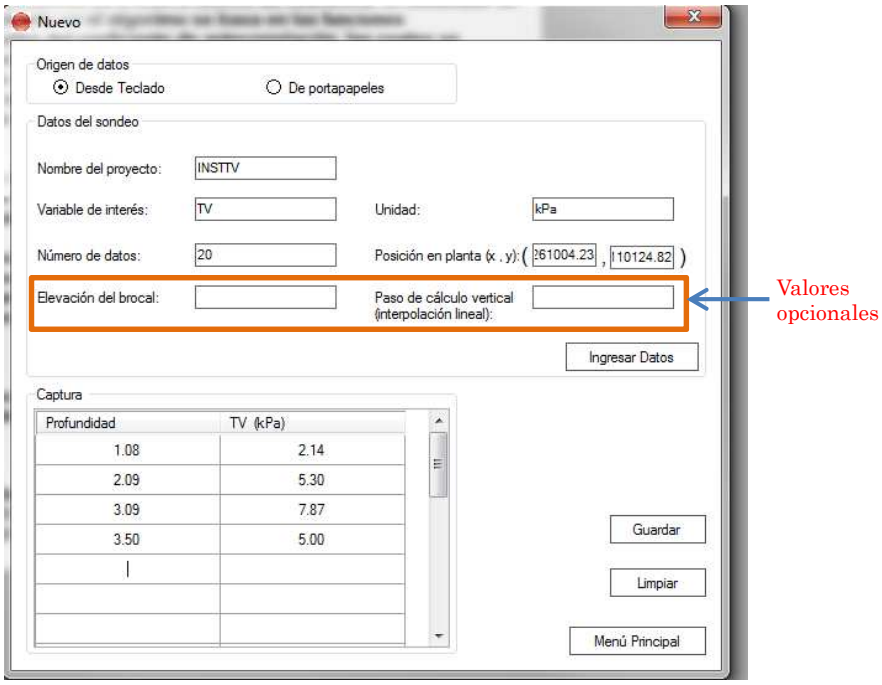


Figura III.12 Ingreso de datos de forma manual.

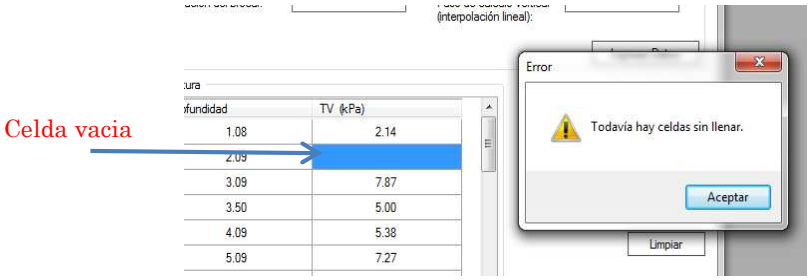


Figura III.13 Mensaje de error, celdas sin llenar.

- De portapapeles: Deberá hacer clic derecho sobre la tabla y se despliega un menú donde se podrá seleccionar la opción de pegar, habiendo copiado en una hoja de Excel los datos de dos columnas correspondientes a la profundidad y la variable (*figura III.14*), se pegarán en la tabla los datos copiados y se actualizara el valor en la caja de texto de *Número de datos*.

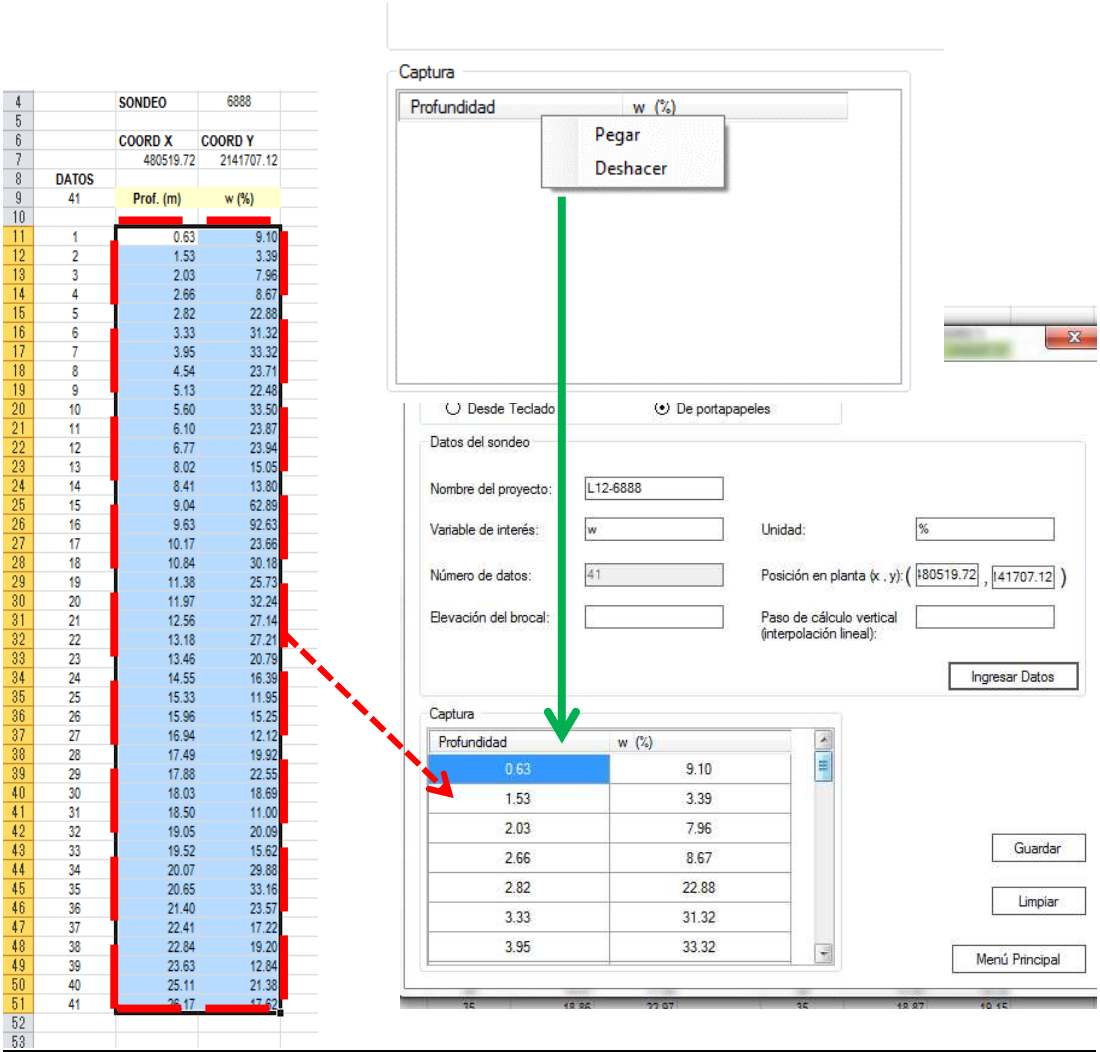


Figura III.14 Ingreso de datos desde portapeles.

5. Habiendo llenado completamente el formulario con los datos necesarios y si no falta ningún campo se selecciona la opción *Guardar*, aparecerá una ventana donde se asignara nombre al nuevo archivo (*figura III.15*).

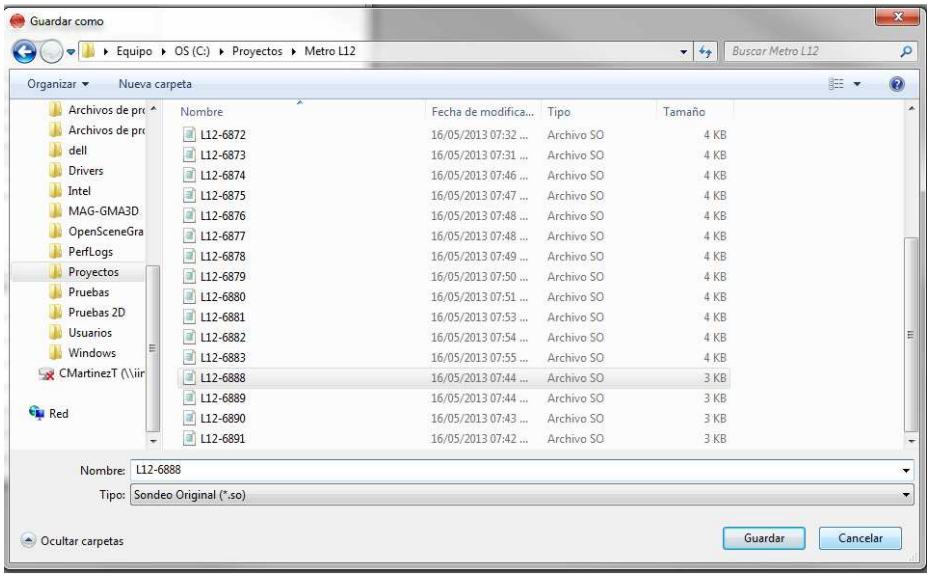


Figura III.15 Guardar como.

6. Si todo se realiza correctamente se desplegará un mensaje indicando que se ha generado un archivo con la ubicación del mismo (*figura III.16*).

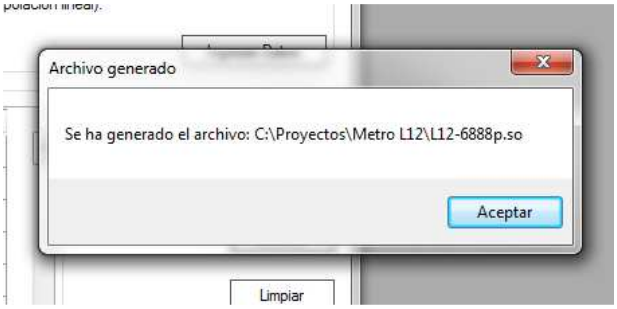


Figura III.16 Generación archivo de proyecto nuevo.