#66

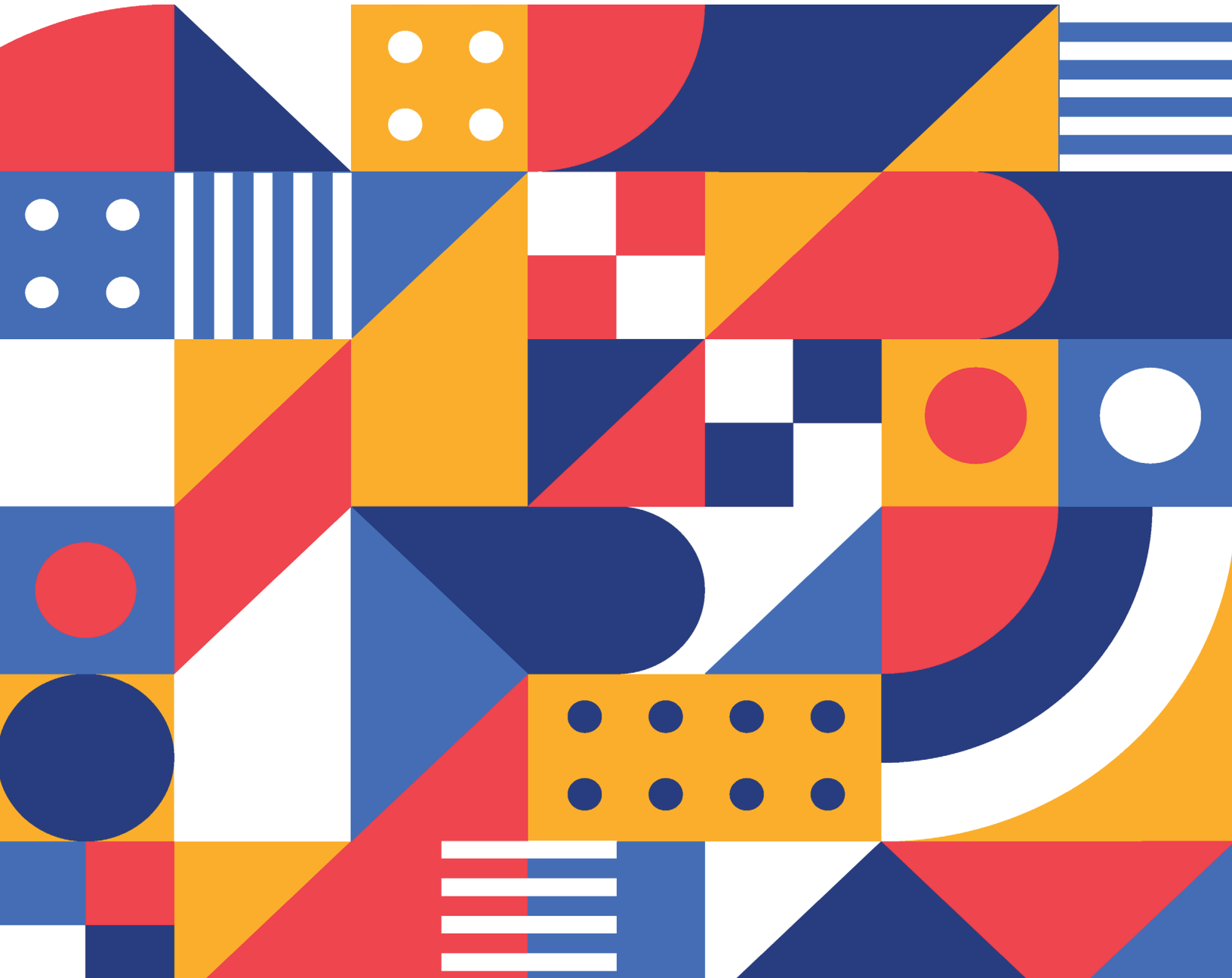
SERIE
MATEMÁTICAS

Reporte de investigación Dic-2005

Genereación numérica de mallas 3D casi ortogonalespara la reconstrucción de yacimientos de hidrocarburos

AÑO
2007

VÍNCULOS
MATEMÁTICOS



FACULTAD DE CIENCIAS

VÍNCULOS MATEMÁTICOS

Generación numérica de mallas 3D casi ortogonales para la reconstrucción de yacimientos de hidrocarburos

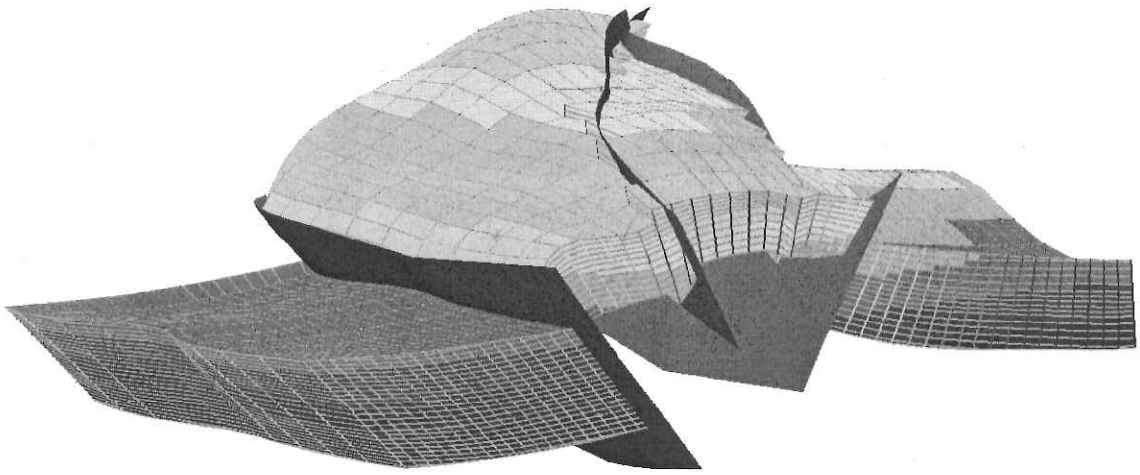
Reporte de Investigación. Diciembre de 2005

VÍNCULOS MATEMÁTICOS NO. 66. 2007

Colaboración del Grupo de Generación Numérica de Mallas y Métodos Numéricos de la Facultad de Ciencias
con el Posgrado de la Facultad de Ingeniería- UNAM

GENERACIÓN NUMÉRICA DE MALLAS 3D CASI ORTOGONALES PARA LA RECONSTRUCCIÓN DE YACIMIENTOS DE HIDROCARBUROS

Reporte de Investigación
Diciembre de 2005



Colaboración del Grupo de Generación Numérica de Mallas y Métodos Numéricos de la Facultad de Ciencias con el Postgrado de la Facultad de Ingeniería - UNAM

Integrantes del GGNM

Dr. Pablo Barrera Sánchez
Responsable por parte del Grupo de Generación Numérica de Mallas.
Facultad de Ciencias - UNAM

M. en C. Martín Carlos Velázquez
Responsable por parte del PEP.
PEMEX - UNAM

Dr. Ángel Albeo Pérez Domínguez
ICIMAF, Cuba.

Dra. Longina Castellanos Noda
ICIMAF, Cuba.

M. en C. Adriana Rivera Hernández
UACM, México.

M. en C. Guilmer González Flores
Facultad de Ciencias, UNAM.

M. en C. Daniel Cervantes
DGSCA, UNAM.

Lic. en Mat. Adriana de la Cruz Uribe
UJAT, México

Act. Luis Carlos Velázquez
Facultad de Ciencias, UNAM.

Generación de Mallas 3D casi Ortogonales para la Reconstrucción de Yacimientos de Hidrocarburos

RESUMEN EJECUTIVO

Un simulador numérico de hidrocarburos requiere información sobre la composición de un yacimiento en cuanto al material de la roca, su compresibilidad, porosidad y propiedades de permeabilidad por mencionar algunas, desde su localización, explotación y abandono. Toda esta información es de gran ayuda a un ingeniero en yacimientos para reproducir el comportamiento de un yacimiento a lo largo de su historia y definir escenarios de recuperación mejorada. Un simulador numérico puede predecir la producción bajo condiciones de operación actuales, o la reacción del yacimiento a cambios en las condiciones, tales como el incremento de la tasa de producción a partir de más o diferentes pozos, respuesta a inyección de agua, químicos, cambios de temperatura, efectos de producción por pozos horizontales, etc.

La cuestión económica de continuar la explotación de un yacimiento y por cuánto tiempo, puede ser obtenida a través de pronósticos de escenarios factibles reproducibles por simuladores numéricos. Es por ello que la importancia de modelar adecuadamente un yacimiento de hidrocarburos es fundamental para poder ofrecer respuestas rápidas y reales a los gerentes de exploración y producción.

Los simuladores numéricos modelan el flujo de fluido móvil a través de las paredes de los bloques resolviendo ecuaciones de flujo-fluido en cada cara del bloque, para esto se requiere de parámetros tales como permeabilidad, porosidad, viscosidad de los fluidos, compresibilidad de la roca, presión capilar, etc. La idea general en la que trabajan los simuladores numéricos de yacimientos es dividir este en bloques, lo que se conoce como bloques de malla o simplemente, la malla 3D de simulación. Cada bloque corresponde a un volumen dentro del yacimiento el cual contiene propiedades de la roca y el fluido de acuerdo a su localización.

Obtener una malla 3D que represente adecuadamente un yacimiento de hidrocarburos es fundamental para operar con cualquier modelo matemático que permita describir el gasto probable de aceite a recuperar de acuerdo a los escenarios posibles: perforar más pozos extractores, perforar pozos inyectores, cerrar pozos, inyectar agua, inyectar gas, probar otras técnicas de recuperación mejorada, etc. Desde luego, el modelo discreto y la malla 3D a emplear deben ser consistentes.

Crear una malla y asignar cada propiedad necesaria para resolver la ecuación de flujo del fluido al bloque de cada celda, es una tarea que consume un tiempo enorme y valioso, por consiguiente no es práctico, es por esto que es necesario desarrollar un procedimiento automático para construir la malla 3D de simulación.

En la práctica, a través de las mediciones de campo realizadas sobre un activo, se cuenta con información de diversas propiedades que conforman un yacimiento de hidrocarburos, como pueden ser: profundidad, porosidad, permeabilidad, por mencionar algunas. Estas propiedades son mediciones que los Ingenieros Geólogos encargados del estudio de la conformación material del

yacimiento proveen para su estudio, información se la entregan al personal encargado de construir una malla 3D de simulación: datos digitalizados. Estos datos son representativos de diferentes capas en que dividen la sección de estudio, distribuidos de manera particular, ya sea por contornos o curvas de nivel o isolíneas de propiedades o bien, datos dispersos o datos provenientes de una retícula rectángular sobre el área de estudio que corresponden a propiedades del yacimiento.

Si el yacimiento fuese una colección de paralepípedos rectangulares, el problema es muy sencillo, este sería modelado por coordenadas cartesianas. Sin embargo, la naturaleza no represesenta lo que se estudia en laboratorio. Por eso el modelo discreto del yacimiento debe conformar las fronteras naturales del yacimiento: fallas, acuíferos y fronteras artificiales de estudio. Otro problema surge cuando modelamos de manera discreta la trayectoria de un pozo, este es una superficie cilíndrica, que las más de las veces sigue una trayectoria espacial muy singular. Por estas características, es conveniente modelar el yacimiento a través de sistemas coordenados curvilíneos de manera que conformen las fronteras.

Una pieza clave para obtener una malla 3D de simulación, es la reconstrucción de cada una de las capas o superficies de propiedades que interesa medir sobre el yacimiento. Por otra parte, la forma tradicional de construir una malla 3D de simulación reportada en la literatura, es primero, construir una malla plana 2D de estudio sobre un plano de referencia a la cima de la sección del yacimiento, segundo proyectar esta malla 2D sobre la cima; y tercero, bajar esta sobre cada capa. Este será el punto de partida para el presente trabajo.

1. Objetivos

En un simulador numérico de yacimientos de hidrocarburos, es necesario representar de manera adecuada la geometría del yacimiento a estudiar. Esto implica que se debe contar con herramientas eficientes para modelar diferentes secciones del yacimiento. Algo importante a enfatizar, es que ningún yacimiento de hidrocarburos es igual, cada uno tiene una formación distinta y una conformación distinta en material rocoso y permeable, sin contar con el hecho de que presenten fallas y por supuesto, la complejidad de los acuíferos. Por esto, es importante diseñar esquemas que nos permitan describir yacimientos complejos.

En este primer año de desarrollo, nos hemos abocado a trabajar con yacimientos homogéneos, sin fallas. Se desarrollaron tareas que integrará un módulo para reconstruir yacimientos que nombraremos a lo largo de este trabajo como GRID. Este módulo es una parte importante dentro del simulador, ya que servirá para reconstruir la complejidad del yacimiento.

Entre las tareas que involucra la descripción discreta de un yacimiento y que debe contar el módulo GRID son:

1. Reconstrucción de las capas que conforman el yacimiento a partir de los datos proporcionados por los Geólogos. Usualmente los datos representan las profundidades o propiedades del yacimiento.
2. Generar una malla 2D sobre la superficie que define la capa.
3. Construir una malla 3D adecuada entre las capas reconstruidas.

Durante este año, fueron desarrolladas rutinas básicas o sub-módulos, que nos permiten generar mallas 3D sobre yacimientos con una geometría no muy complicada y sin fallas.

Durante este año de trabajo, ganamos experiencia en los sub-problemas involucrados, muchos de los cuales fueron detectándose a lo largo del proyecto y que nos permitieron enriquecer el módulo GRID y con ello ofrecer al usuario una serie de opciones que le permitirán modelar adecuadamente el yacimiento a estudiar. A lo largo del desarrollo del proyecto se trabajó en actividades paralelas, algunas dedicadas primero a entender el problema a resolver, plantear una primera forma de atacarlo, construir un sub-módulo que resuelva esa tarea y calibrarlo con problemas tipo. Este esquema de trabajo, aunque largo, nos permitió modificar parámetros involucrados en cada procedimiento, añadir métodos más eficientes e incluso replantear las tareas para lograr cada objetivo.

El proyecto en el que estuvimos trabajando tiene como objetivo principal construir un "Simulador Numérico de Yacimientos Multipropósito", el cual incluye el módulo GRID 1.0 que permite al usuario construir un modelo 3D discreto del yacimiento, la malla 3D de simulación. Para el desarrollo de este módulo se definieron los Requerimientos Funcionales para un sistema de tal envergadura. Para obtener una malla 2D sobre el plano de referencia, se definieron los elementos esenciales pora modelar regiones poligonales usando contornos de contro. La malla primera se obtuvo por interpolación transfinita a partir de los contornos de control. Por otra parte, contando con la malla inicial, el usuario puede optar por mejorarla a través de un suavizamiento por medio del funcional discreto de área-ortogonalidad, que permite obtener mallas suaves y casi-ortogonales.

Durante estas actividades, desarrollamos algunas rutinas de graficación en Matlab, C, y Fortran que nos permitieron visualizar los resultados obtenidos. Estas rutinas sirvieron para contar con una primera forma de mostrar los resultados conforme estos fueron obtenidos. Esos programas fueron desarrollados tanto para las mallas planas, como para visualizar los puntos dispersos y los contornos a partir de los cuales reconstruimos cada capa o superficie por diferentes métodos de interpolación y suavizamiento.

Otra de las fases contempladas en este trabajo, fue reconstruir y visualizar la superficie a partir de los contornos de profundidad. Esto nos permitió obtener la malla 3D como una proyección de la malla 2D sobre el plano de referencia hacia la superficie reconstruida. Este fue nuestro primer intento por atacar el problema, aprendimos las dificultades que esto involucra y propusimos el estudio de métodos que interpolen y suavicen la superficie.

En la segunda fase del año, nos abocamos a la tarea de emplear un método de interpolación bivariada para la reconstrucción de la superficie. Este método, involucra una serie de sub-tareas como la triangulación de la región de estudio la cual es comúnmente usada en diversas técnicas de reconstrucción de superficies, y de la cual somos expertos.

Con la experiencia lograda durante la primera fase, se propuso el diseño de un módulo generador de mallas 3D obtenido a partir de la reconstrucción de las capas o superficies. Esto se logró siguiendo las especificaciones usuales de la Ingeniería de Software. Se planteó el Análisis General para posteriormente describir el Diseño del Sistema. Los rubros que se cubrieron son: Objetivo, Panorama, Descripción del Proyecto, Rango de Alcance, Restricciones Generales para el usuario, Requerimientos Específicos, Requerimientos de Interfase Externa y Requerimientos no Funcionales.

Con la experiencia obtenida a través del uso del paquete GRID del simulador comercial ECLIPSE, observamos las deficiencias para definir los Contornos de Control, esto es, los segmentos de línea que limitan la malla 2D en el plano de referencia, y propusimos usar mejores técnicas para lograr una adecuada parametrización de las curvas de nivel del mapa de contornos y con ello interpolar sobre toda la curva o parte de ella; esto, con el fin de contar con una mejor descripción de la curva

mediante puntos discretos y con esto obtener superficies representativas que eviten tantas oscilaciones en su forma. Durante esa fase, se continuó trabajando en los Requerimientos Funcionales de las tareas involucradas, muchas de las cuales fueron creciendo y/o modificándose conforme el sistema fue siendo calibrado con problemas tipo de la literatura y yacimientos simples.

Debido a que los datos del yacimientos cuentan las más de las veces con erroes de medición, durante la última fase, implementamos un método basado en B-splines del tipo multinivel para suavizar la superficie. Con este trabajo, el módulo ofrece alternativas al usuario dispuesto a experimentar con diferentes técnicas para reconstruir superficies a partir de datos dispersos.

Durante las últimas fases, se trabajó conjuntamente con el grupo de Interfaz y Visualización Gráfica, a fin de diseñar y modificar los requerimientos funcionales, así como definir y calibrar las tareas a realizar y las facilidades gráficas para el Módulo Generador de Mallas 3D GRID 1.0.

En la Figura 1. se puede observar un diagrama esquemático de las tareas desarrolladas y que contempla el módulo GRID.

Organigrama de las Tareas involucradas en el Módulo GRID

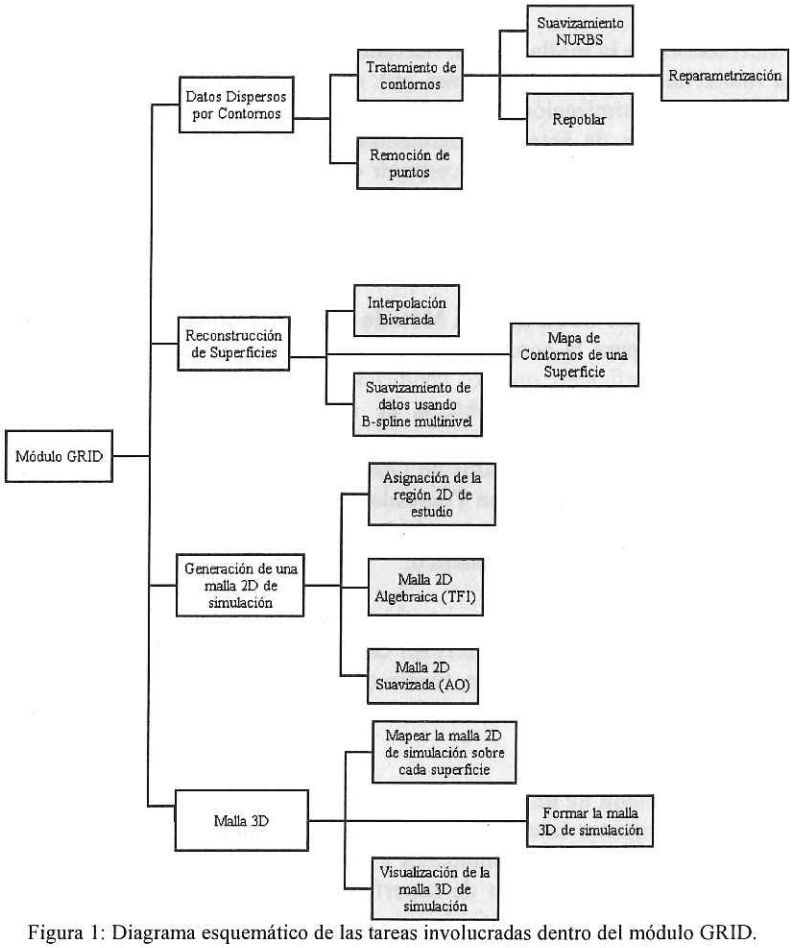


Figura 1: Diagrama esquemático de las tareas involucradas dentro del módulo GRID.

2. Actividades desarrolladas

Para lograr cumplir satisfactoriamente el proyecto, se cubrieron las tareas señaladas en el apartado anterior, mismas que brevemente son descritas por submódulos o subtareas desarrolladas.

Sub-módulo Generador de la malla algebraica

Una malla 2D sobre el plano de referencia está delimitada por lo que se conoce como contornos de control de la malla. Existen contornos de control internos y externos. Los externos corresponden a segmentos de frontera de la región 2D. Los internos son útiles para definir secciones de la región donde se desea seconcentren las líneas curvilíneas de la malla 2D. Contando con los puntos distribuidos sobre cada contorno de control externo, el paso siguiente es llevar a cabo una interpolación entre cada dos contornos de control opuestos, para de esta forma obtener una malla estructurada por interpolación. Esta será la malla inicial con la que se cuente o bien, sobre la que se trabaje si el usuario considera que es suficiente. Este problema fue resuelto usando el Método de

Interpolación Transfinita. Para lograr esto, se parametriza cada segmento de control opuesto y se distribuyen los puntos de manera uniformemente respetando la forma del polígono.

Sub-módulo Suavizador de la malla inicial: Método Discreto AO
A partir de la malla inicial, podemos obtener una malla suave y casi ortogonal a través de un procedimiento directo de optimización donde la función objetivo es un funcional discreto que mide las propiedades geométricas de área y ortogonalidad de cada celda. En esta primera estapa de construcción del simulador, se consideró incorporar esta modalidad como una opción para que el usuario pueda elegir entre la malla algebraica obtenida en el paso anterior, o bien obtener una suave y casi ortogonal con este procedimiento. Este módulo hace uso del módulo de optimizador de Newton Truncado con Región de Confianza.

Sub-módulo Optimizador de Gran Escala: Método Newton Truncado
El planteamiento discreto para la construcción de una malla a través de un funcional, involucra contar con un optimizador eficiente que nos conduzca al óptimo del problema. Cuando la malla es de dimensión grande, se tendrá un problema de optimización de gran escala. Para resolver adecuadamente problemas no lineales de gran escala, es necesario contar con un módulo de optimización eficiente. Uno de los Métodos de Optimización que hemos estudiado para construir mallas por métodos discretos es el Newton Truncado con Región de Confianza. Este optimizador se ha implementado dentro de la rutina `solver.f90`. El optimizador será una pieza clave en algunas tareas que deberá resolver el simulador numérico.

Codificación y documentación de rutinas
Con el fin de generar documentos que describan cada una de las rutinas y módulos involucrados bajo un formato estándar de documentación, fueron documentadas las rutinas siguiendo un Estándar de Codificación y Documentación de Productos provisto por los Ingenieros de Software del proyecto.

Sub-módulo: Reconstrucción de una superficie
Durante la primera fase nuestro interés fue entender el problema de reconstruir una superficie a partir de los mapas de contorno de profundidades y de isopropiedades del yacimiento. Para esto, se investigó una técnica de reconstrucción de superficies a partir de los contornos usando curvas NURBS.
Se desarrolló una versión experimental en C++ con salida en VRML para la visualización de la geometría del yacimiento, a partir del conjunto de contornos de profundidad. Se observó la dificultad para trabajar con esta técnica dado que se requiere que las curvas del mapa de contornos se encuentren ordenadas. El trabajo desarrollado se planteó exclusivamente para el caso en que los datos que describen la cima son mapas de contorno. Los datos que entrega el programa pueden observarse por VRML.
En la segunda parte, se estudiaron técnicas para reconstruir superficies a partir de la interpolación de datos dispersos. Para esto se elaboró un módulo en donde se implementa la técnica de Interpolación Bivariada y el suavizamiento de datos basada en procedimientos de aproximación local, desarrollada por Hiroshi Akima. Este método parte de un conjunto de puntos xi y yi (para i=0,1,...,n) de una malla rectangular y obtiene una función z= z(x,y), en donde las derivadas parciales de primer orden son continuas. Esta técnica permite eliminar excesivas ondulaciones entre los puntos dados de la malla.
Así, con este módulo se logró interpolar cualquier punto (x,y), que pertenezca al dominio de trabajo, al encontrar un triángulo que lo contiene, estimar su primer y segunda derivada parcial. Sin embargo, como se hizo notar a lo largo de ese periodo, los datos que entrega el Geólogo para la descripción del yacimiento provienen de mediciones, mismas que cuentan con errores. Por lo que

una interpolación de los datos dispersos no es suficiente para observar la tendencia de los datos y por ende, la forma de la superficie.
Para resolver eficientemente esta cuestión se planteó que la superficie fuese reconstruida a través del suavizamiento de datos. Se propuso aplicar un método muy interesante de ajuste por B-spline a través de una técnica multinivel. Esto permite elegir un grado de suavizamiento de los datos obteniendo una representación satisfactoria para el nivel 7. Los resultados que obtuvimos de pruebas con diferentes tipos de regiones y datos dispersos obtenidos a través de problemas tipo (como la función de Franke) nos muestran que este método es muy efectivo para el caso cuando la propiedad a medir o la profundidad es un parámetro que presenta variaciones suaves.

Diseño de la interfaz de gráfica con el usuario
Nuestro grupo de trabajo resuelve de manera eficiente el problema de construir mallas 2D sobre regiones muy irregulares. La línea de trabajo se centra en observar propiedades geométricas sobre las celdas como lo son la suavidad en las líneas curvilíneas, la uniformidad en el área de las celdas y la ortogonalidad de las líneas curvilíneas. Para lograr entender el problema de la construcción de la malla 3D para ser usada en un proceso de simulación de yacimientos de hidrocarburos, se estudiaron varios simuladores y sus módulos para construir la malla 3D, algunos de estos paquetes son el ECLIPSE y el CMG. Con el módulo GRID de ECLIPSE aprendimos a reconocer el valor agregado que tiene el Tratamiento de Contornos con el fin de reconstruir eficientemente las capas o superficie sobre el área de trabajo. Con el simulador CMG aprendimos que la interfaz gráfica con el usuario es una pieza clave para el éxito de su uso. El usuario debe poder manejar adecuadamente todas las subtareas que se contempla cuente el simulador para construir la malla 3D, y asignar sus propiedades. Esto debe hacerse de manera automática con opciones predeterminadas. Para esto se elaboraron dos trabajos de suma importancia para el diseño del módulo: una tiene que ver con los requerimientos funcionales para obtener la malla 3D, la otra tiene que ver con la Visualización de la malla 3D. El diseño de Interfaz Gráfica presentado sigue un estilo cercano a Ingeniería de Software, donde se describe primeramente un Análisis General para posteriormente describir el Diseño del Sistema.

Herramientas de Visualización Gráfica previa
Conforme desarrollamos las sub-tareas programadas para cada elemento del Módulo Generador de Mallas 3D, fue necesario programar y diseñar pequeños programas gráficos que nos permitieron visualizar nuestros resultados antes de incorporar las rutinas al simulador para de esta manera, comparar los resultados obtenidos por el simulador y nuestro desarrollo previo. Estos programas fueron definidos conforme el proyecto avanzó a lo largo del año.

Sub-módulo: Tratamiento de los contornos
Al atacar el problema de la reconstrucción de superficies, pudimos notar que la densidad de puntos es un factor determinate de su forma. Por una parte, los datos dispersos del trabajo provenían de los mapas de contornos de las cimas, esto es, contaban con un patrón ordenado, o determinado a partir de la curva de nivel que representan. Logramos observar que esa colección de curvas del mapa de contornos, no siempre se encontraban distribuidos de manera uniforme a lo largo de la curva, o bien concentrados en alguna sección, es decir, la curva así discretizada presentaba fuertes discontinuidades de clase C1. Esto provoca que la superficie así reconstruida presente ondulaciones o rugosidades en su curvatura. Por lo que diseñar y aplicar una técnica que primero nos permita suavizar cada curva del mapa de contorno y luego parametrizarla para distribuir a lo largo de su longitud de arco los puntos, era una tarea que debería ser resuelta de manera automática por el sistema. Para esto se diseñó este sub-módulo que permite de manera automática suavizar por medio de curvas NURBS y una técnica para parametrizar dichas curvas. Esta sub-tarea o módulo se aplica de igual manera para suavizar los contornos de control que definen una región 2D sobre el plano de referencia.

Sub-módulo: Optimizador puntual, Método de Newton Truncado
Cuando se construye la malla 2D, en algunas situaciones, es necesario suavizar las líneas de una sección de la región de estudio, ya sea porque se han introducido algunas modificaciones previas o bien, se desea centrar el estudio en determinada sección. Con este fin, se desarrolló una rutina que optimiza la malla 2D de manera puntual. Esto permite por una parte, suavizar mallas de dimensiones muy grandes, y por la otra, dar una primera forma de resolver el problema de las fallas presentes en yacimientos, al ser consideradas éstas como restricciones puntuales para la malla 2D sobre el plano de referencia. Esta idea de atacar el problema es interesante y novedosa.

Sub-módulo: Generador Experimental de Mallas 3D
Como se comentó en la introducción. Existen varias formas de describir el yacimiento a partir de las capas que lo conforman. Una primera es considerar una malla de referencia, mapearla sobre la superficie reconstruida de la cima del yacimiento y bajarla a un grosor constante. Esto es, se estarán considerando capas intermedias a partir de la cima de grosor constante donde se encontrará mapeada la malla 2D del plano de referencia. Las mallas 2D que generamos por medio del módulo Suavizamiento de Mallas a través del funcional AO, son mallas suaves y casi ortogonales, la malla 3D así reconstruida sigue lo que se conoce como la geometría corner-point, es decir, los vértices del paralepípedo se encuentran sobre una superficie no necesariamente alineada. Existen otras formas de construir la malla 3D: si en lugar de elegir un grosor vertical constante, elegimos un grosor constante pero en la dirección de la normal a la superficie, podremos por lo menos garantizar que las caras verticales con respecto a las correspondientes arriba y debajo de un bloque de celda, sí sean ortogonales. Pero esta idea será algo que recomendaremos como un trabajo posterior.

GENERACIÓN NUMÉRICA DE MALLAS 3D CASI ORTOGONALES PARA LA RECONSTRUCCIÓN DE YACIMIENTOS DE HIDROCARBUROS

Fecha	Propietario	Aprobación
Diciembre 15, 2005	Grupo de Generación Numérica de Mallas	Dr. Pablo Barrera Sánchez Coordinador del GGNM

1. INTRODUCCIÓN

Un simulador numérico contiene mucha información sobre la composición de un yacimiento en cuanto al material de la roca, su compresibilidad, su porosidad, propiedades de permeabilidad por mencionar algunas, desde su localización, todo eso puede ayudarle a un ingeniero en yacimientos a reproducir el comportamiento de un yacimiento. Un simulador numérico puede predecir la producción bajo condiciones de operación actuales, o la reacción del yacimiento a cambios en las condiciones, tales como el incremento de la tasa de producción a partir de más o diferentes pozos, respuesta a inyección de agua, químicos, cambios de temperatura, efectos de producción por pozos horizontales, etc.

El simulador es una herramienta que ayuda a los ingenieros a resolver cuestiones sobre el comportamiento de un yacimiento, ofrecer recomendaciones para un aprovechamiento práctico del mismo.

Una vez determinado el objetivo de la simulación, el siguiente paso es describir el yacimiento en términos del volumen de aceite o gas en el, la cantidad de ello recuperable y la tasa a la cual esta es recuperable. Para estimar la reserva recuperable, debe ser determinado un modelo del yacimiento, que incluya fallas, capas así como las propiedades asociadas a este. Esto es lo que se conoce como el modelo estático del yacimiento, el cual es creado a través de combinar esfuerzos de geólogos, geofísicos, petrofísicos e ingenieros de yacimientos.

La cuestión económica de continuar la explotación de un yacimiento y por cuánto tiempo, puede ser obtenida a través de los simuladores numéricos. Es por ello que la importancia de modelar adecuadamente un yacimiento es fundamental para poder ofrecer respuestas rápidas y reales a los gerentes de explotación y producción.

La idea general en la que trabajan los simuladores numéricos de yacimientos es dividir este en bloques, lo que se conoce como bloques de malla o simplemente, la malla 3D de simulación. Cada bloque corresponde a un volumen dentro del yacimiento el cual contiene propiedades de la roca y el fluido de acuerdo a su localización. Los simuladores modelan el flujo de fluido móvil a través de las paredes de los bloques resolviendo ecuaciones de flujo-fluido en cada cara del bloque, para esto se requiere de parámetros tales como permeabilidad, porosidad, viscosidad de los fluidos, compresibilidad de la roca, presión capilar, etc.

Para llevar a cabo predicciones de producción de un yacimiento, los ingenieros petroleros necesitan entender la compleja estructura 3D de un yacimiento de hidrocarburos, así como el movimiento de fluidos a lo largo de el. Para de esta manera y a través de simulaciones sobre el comportamiento de fluidos y viscosidades, ofrecer recomendaciones en cuanto a abrir nuevos pozos productores, abrir pozos inyectores o bien cambiar la naturaleza de ambos, esto con el fin de recuperar el mayor volumen de hidrocarburos posible, y desde luego la calidad del mismo.

Para entender con claridad las características de un yacimiento, los especialistas requieren analizar el movimiento de los fluidos y la naturaleza de esta a lo largo de diferentes cortes o planos que les permitan observar diferentes escenarios posibles entre los pozos.

La forma tradicional de representar un yacimiento es a través de una malla 3D que proporciona una discretización del mismo, la cual permite llevar a cabo algunas mediciones y simulaciones numéricas sobre el volumen y la calidad de aceite a producir a lo largo del tiempo. La salida obtenida por estas simulaciones, son series de tiempo de muchas variables involucradas: presión, saturación de agua, saturación de aceite, el contacto agua-aceite, etc.

Obtener una malla 3D que represente adecuadamente un yacimiento de hidrocarburos es fundamental para operar con cualquier modelo matemático que permita describir el gasto probable de aceite a recuperar de acuerdo a los escenarios posibles: perforar más pozos extractores, perforar pozos inyectores, cerrar pozos, inyectar agua, inyectar gas, probar otras técnicas de recuperación. Desde luego, el modelo discreto y la malla 3D a emplear deben ser consistentes.

Crear una malla y asignar cada propiedad necesaria para resolver la ecuación de flujo del fluido al bloque de cada celda, es una tarea que consume un tiempo enorme y valioso, por consiguiente no es práctico.

Tradicionalmente los bloques de celda de un modelo discreto del yacimiento son rectilíneos con caras planas tanto en la base superior como en la inferior, lo cual se conoce como la geometría de bloque-centrado. Esta configuración garantiza que la malla sea ortogonal y corresponda a los modelos matemáticos que usan los simuladores.

En la práctica, a través de mediciones de campo, se cuenta con información de diversas propiedades que conforman un yacimiento de hidrocarburos, como pueden ser: profundidad, porosidad, permeabilidad, por mencionar algunas. Estas propiedades son mediciones que los Geólogos encargados del estudio de la conformación material del yacimiento proveen para su estudio.

Usualmente se entregan al personal encargado de construir una malla 3D de simulación, los datos digitalizados sobre diferentes capas en que dividen la sección de estudio, distribuidos de manera particular, por contornos o curvas de nivel o isolíneas de propiedades, o bien, datos dispersos o datos provenientes de una retícula rectángular sobre el área de estudio, que corresponden a propiedades de un yacimiento.

Si el yacimiento fuese una colección de paralepípedos rectangulares, el problema es muy sencillo, este sería modelado por coordenadas cartesianas. Sin embargo, la naturaleza no represesenta lo que se estudia en laboratorio. Por eso el modelo discreto del yacimiento debe conformar las fronteras naturales del yacimiento. De manera similar, cuando modelamos de manera discreta un solo pozo, este es una superficie cilíndrica, la cual representa una frontera para el yacimiento.

Los sistemas coordenados que conforman las fronteras curvas de un yacimiento son los sistemas coordenados curvilíneos.

Una pieza clave para obtener una malla 3D de simulación, es la reconstrucción de cada una de las capas o superficies de propiedades que interesa medir sobre el yacimiento. La forma tradicional de construir una malla 3D de simulación usando esta información es, primero, construir una malla plana 2D de estudio sobre un plano de referencia a la cima

de la sección del yacimiento, segundo proyectar esta malla 2D sobre la cima; y tercero, bajar esta sobre cada capa.

Este será el punto de partida para el presente trabajo.

1.1 Integrantes del GGNM

Dr. Pablo Barrera Sánchez
Responsable por parte del Grupo de Generación Numérica de Mallas.

M.C. Martín Carlos Velázquez
Responsable por parte del PEP.
PEMEX -UNAM

Dr. Ángel Albeo Pérez Domínguez
Responsable de Suavizamiento e Interpolación de Superficies. Desarrollador.

Dr. Longina Castellanos Noda
Responsable de la Optimización Numérica. Desarrollador.

M. en C. Adriana Rivera Hernández
Responsable del Tratamiento de Contornos. Desarrollador.

M. en C. Guilmer González Flores
Coordinador del trabajo de los becarios. Desarrollador.

M. en C. Daniel Cervantes
Becario. Desarrollador.

Lic. en Mat. Adriana de la Cruz Uribe
Becario. Desarrollador.

Act. Luis Carlos Velázquez
Becario. Desarrollador.

2. Descripción del trabajo desarrollado

2.1 Esquema general para construir una malla 3D

Los acuíferos y yacimientos de petróleo son formaciones cuyas fronteras no son fáciles de representar mediante un sistema coordenado cartesiano.

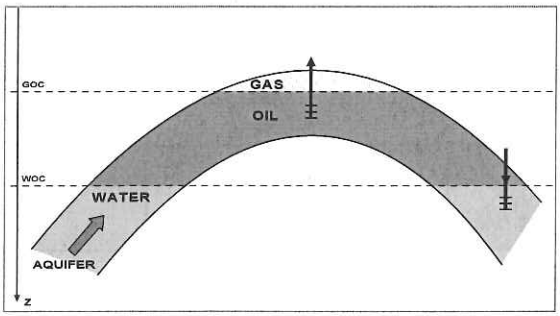


Figura 2.1.1: Ilustración de un yacimiento de hidrocarburos.

Un prototipo de yacimiento podría ser el formado por una colección de elementos simples espaciales como paralepípedos rectángulares rotados con respecto a un eje de referencia, que podría ser el nivel del mar, o el nivel de la superficie del activo. Sin embargo, la gran mayoría de yacimientos solo pueden ser modelados con fronteras (arriba o abajo) y la forma del yacimiento debe observar esto.

En la Figura 2.1.2 se puede apreciar una sección transversal y la forma 3D de una sección de estudio del yacimiento.

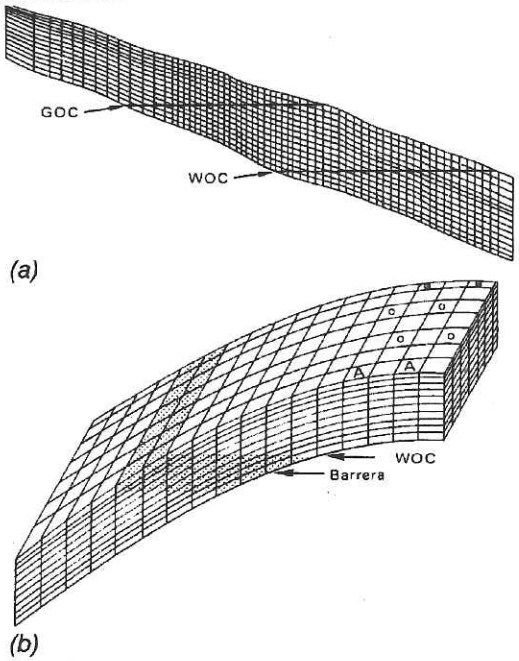


Figura 2.1.2: Ejemplo de una malla 3D. (a) sección tranversal (b) modelo 3D. Tomado de Mattax y Dalton, 1970.

Los datos de la geometría de un yacimiento usualmente se encuentran disponibles a través de mapas de contornos sobre el grosor y la profundidad del yacimiento a partir de la superficie del yacimiento, ver Figura 2.1.3.

Usualmente esos mapas de contornos contienen información del nivel del mar o de la superficie así como de un plano de referencia $\xi-\eta$, o un plano que es aproximadamente paralelo al yacimiento (el yacimento puede estar inclinado, o tener una formación ondulada).

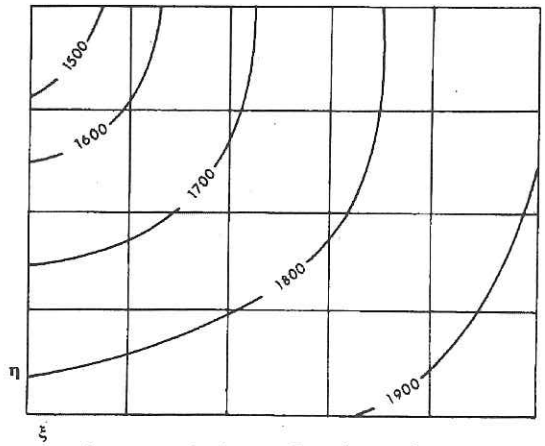


Figura 1.1.3: Mapa de contornos de un yacimiento. En el se observa una malla cartesiana. Tomado de Hirasaki y O'Dell, 1970.

Bajo esto, un sistema de coordenadas cartesianas (ξ,η,ω) puede ser definido de manera que para $\omega=0$ la superficie coincida con el plano de referencia. La posición sobre el plano de referencia puede ser determinada por las coordenadas $(\xi,\eta,0)$. Si el plano de referencia es el nivel del mar o de la superficie donde se encuentre el yacimento, entonces ω representa la profundidad, si es diferente entonces ω es un poco más complejo de determinar. Por otra parte, el grosor y la distancia de la cima del yacimiento desde el plano de referencia es una función de (ξ,η) sobre el mapa de contornos.

Ahora bien, denotemos por (x,y,z) al sistema coordenado curvilíneo del yacimiento. Debemos especificar el sistema coordenado de manera que coincida con las superficies del yacimiento, por ejemplo, sin pérdida de generalidad, podemos considerar que para $z=0.0$ la superficie coincida con la cima y para $z=1.0$ con la base del yacimiento. Observémos la Figura 2.1.4.

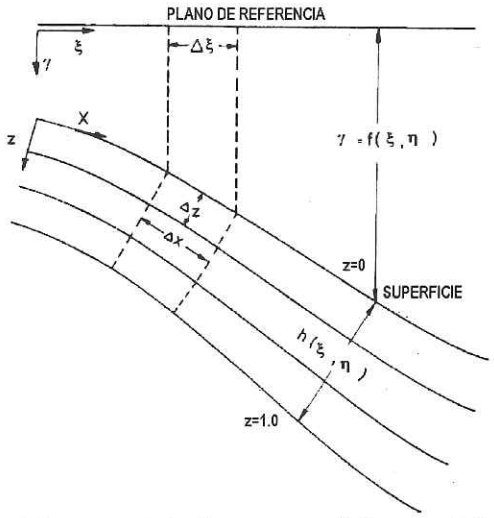


Figura 2.1.4: Sección transversal de un yacimiento para $x = 0.0$ y $x = 1.0$. Tomado de Hirasaki y O'Dell, 1970.

Ahora bien, las coordenadas (ξ, η) están definidas sobre el plano de referencia, necesitamos definir de manera conveniente una relación con las coordenadas (x, y) de la cima de la superficie. Sobre la cima de la superfcie, hagamos

$$x = \xi, y = \eta, z = 0$$

Esto es, definamos las coordenadas curvilíneas sobre la cima, como una proyección de las coordenadas del plano de referencia hacia la cima. Esta definición es conveniente, ya que de esta forma una malla sobre el plano de referencia al ser proyectada sobre la cima se obtiene una malla sobre la cima del yacimiento.

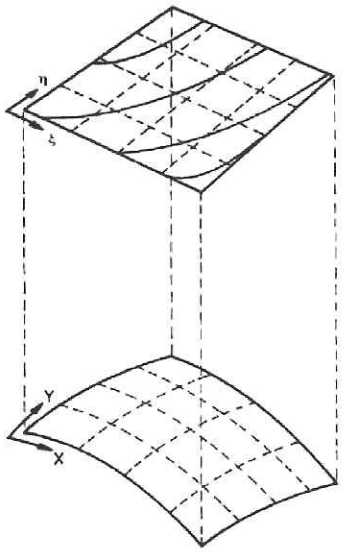


Figura 2.1.5: Procedimiento para obtener una malla 2D sobre la superficie. Tomado de Hirasaki y O'Dell, 1970.

Ahora bien, la malla de simulación debe ser obtenida sobre el interor del yacimiento. La malla de referencia, sobre el plano de referencia (ξ,η) puede ser usada para obtener la malla sobre las coordendas curvilíneas (x,y) de la superficie y a partir de esto, especificar espesores Δz de las capas intermedias. La malla para el sistema (x,y,z) define una colección de *celdas* de la malla o *bloques* de la malla. En la literatura los *puntos* de la malla son los centros de esos bloques.

Observemos que un punto, en la superficie de la cima del yacimiento, está determinado espacialmente en términos del sistema cartesiano del plano de referencia. Por ejemplo, un punto $(x,y,0)$ sobre la superficie de la cima, tiene por coordenadas cartesianas de posición a $(\xi,\eta,D(\xi,\eta))$ donde $D(\xi,\eta)$ es la profundidad de la cima a partir de la superficie (o del plano de referencia).

La forma en que se calcula la posición de los puntos de la malla en el interior, está relacionado con la forma en que se desee construir los grosores de las capas intermedias a partir de $D(\xi,\eta)$. Una forma sería simplemente bajar la malla 2D sobre cada cima intermedia de manera vertical, o bien, considerar la dirección z como la normal a la superficie. Ambas formas se ilustran en la Figura 2.1.6.

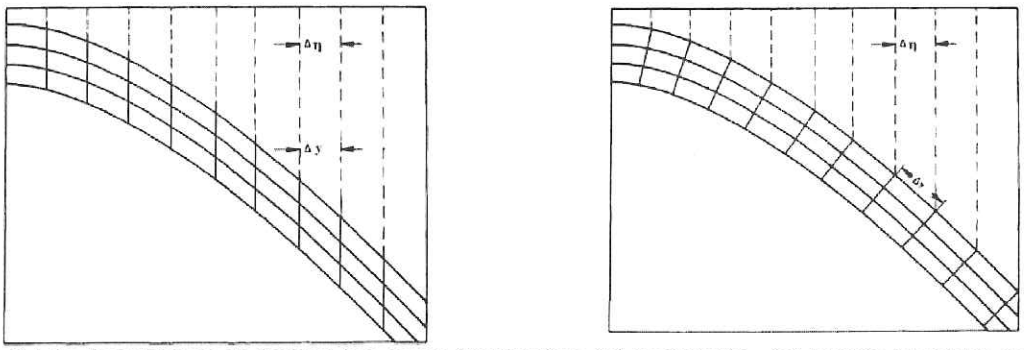


Figura 2.1.6: Dos formas de formar las capas intermedias del yacimiento. (a) Desplazamientos verticales (B) Desplazamientos normales a la superficie.

2.1.1 Objetivos

En un simulador numérico de yacimientos de hidrocarburos, es necesario representar de manera adecuada la geometría del yacimiento a estudiar. Esto implica que se debe contar con herramientas eficientes para modelar diferentes secciones del yacimiento. Algo importante a enfatizar, es que ningún yacimiento de hidrocarburos es igual, cada uno tiene una formación distinta y una conformación distinta en material rocoso y permeable, sin contar con el hecho de que presenten fallas y por supuesto, la complejidad de los acuíferos. Por esto, es importante diseñar esquemas que nos permitan describir yacimientos complejos.

En este primer año de desarrollo, nos hemos abocado a trabajar con yacimientos homogéneos, sin fallas. Se desarrollaron tareas que integrará un módulo para reconstruir yacimientos que nombraremos a lo largo de este trabajo como GRID. Este módulo es una parte importante dentro del simulador, ya que servirá para reconstruir la complejidad del yacimiento.

Entre las tareas que involucra la descripción discreta de un yacimiento y que debe contar el módulo GRID son:

1. Reconstrucción de las capas que conforman el yacimiento a partir de los datos proporcionados por los Geólogos. Usualmente los datos representan las profundidades o propiedades del yacimiento.
2. Generar una malla 2D sobre la superficie que define la capa.
3. Construir una malla 3D adecuada entre las capas reconstruidas.

Durante este año, fueron desarrolladas rutinas básicas o sub-módulos, que nos permiten generar mallas 3D sobre yacimientos con una geometría no muy complicada y sin fallas.

Durante este año de trabajo, ganamos experiencia en los sub-problemas involucrados, muchos de los cuales fueron detectándose a lo largo del proyecto y que nos permitieron enriquecer el módulo GRID y con ello ofrecer al usuario una serie de opciones que le permitirán modelar adecuadamente el yacimiento a estudiar. A lo largo del desarrollo del proyecto se trabajó en actividades paralelas, algunas dedicadas primero a entender el problema a resolver, plantear una primera forma de atacarlo, construir un sub-módulo que resuelva esa tarea y calibrarlo con problemas tipo. Este esquema de trabajo, aunque largo, nos permitió modificar parámetros involucrados en cada procedimiento, añadir métodos más eficientes e incluso replantear las tareas para lograr cada objetivo.

El proyecto en el que estuvimos trabajando tiene como objetivo principal construir un "Simulador Numérico de Yacimientos Multipropósito", el cual incluye el módulo GRID 1.0 que permite al usuario construir un modelo 3D discreto del yacimiento, la malla 3D de simulación. Para el desarrollo de este módulo se definieron los Requerimientos Funcionales para un sistema de tal envergadura. Para obtener una malla 2D sobre el plano de referencia, se definieron los elementos esenciales pora modelar regiones poligonales usando contornos de contro. La malla primera se obtuvo por interpolación transfinita a partir de los contornos de control. Por otra parte, contando con la malla inicial, el usuario puede optar por mejorarla a través de un suavizamiento por medio del funcional discreto de área-ortogonalidad, que permite obtener mallas suaves y casi-ortogonales.

Durante estas actividades, desarrollamos algunas rutinas de graficación en Matlab, C, y Fortran que nos permitieron visualizar los resultados obtenidos. Estas rutinas sirvieron para contar con una primera forma de mostrar los resultados conforme estos fueron obtenidos. Esos programas fueron desarrollados tanto para las mallas planas, como para visualizar los puntos dispersos y los contornos a partir de los cuales reconstruimos cada capa o superficie por diferentes métodos de interpolación y suavizamiento.

Otra de las fases contempladas en este trabajo, fue reconstruir y visualizar la superficie a partir de los contornos de profundidad. Esto nos permitió obtener la malla 3D como una proyección de la malla 2D sobre el plano de referencia hacia la superficie reconstruida. Este fue nuestro primer intento por atacar el problema, aprendimos las dificultades que esto involucra y propusimos el estudio de métodos que interpolen y suavicen la superficie.

En la segunda fase del año, nos abocamos a la tarea de emplear un método de interpolación bivariada para la reconstrucción de la superficie. Este método, involucra una serie de sub-tareas como la triangulación de la región de estudio la cual es comúnmente usada en diversas técnicas de reconstrucción de superficies, y de la cual somos expertos.

Con la experiencia lograda durante la primera fase, se propuso el diseño de un módulo generador de mallas 3D obtenido a partir de la reconstrucción de las capas o superficies. Esto se logró siguiendo las especificicaciones usuales de la Ingeniería de Software. Se planteó el Análisis General para posteriormente describir el Diseño del Sistema. Los rubros que se cubrieron son: Objetivo, Panorama, Descripción del Proyecto, Rango de Alcance, Restricciones Generales para el usuario, Requerimientos Específicos, Requerimientos de Interfase Externa y Requerimientos no Funcionales.

Con la experiencia obtenida a través del uso del paquete GRID del simulador comercial ECLIPSE, observamos las deficiencias para definir los Contornos de Control, esto es, los segmentos de línea que limitan la malla 2D en el plano de referencia, y propusimos usar mejores técnicas para lograr una adecuada parametrización de las curvas de nivel del mapa de contornos y con ello interpolar sobre toda la curva o parte de ella; esto, con el fin de contar con una mejor descripción de la curva mediante puntos discretos y con esto obtener superficies representativas que eviten tantas oscilaciones en su forma. Durante esa fase, se continuó trabajando en los Requerimientos Funcionales de las tareas involucradas, muchas de las cuales fueron creciendo y/o modificándose conforme el sistema fue siendo calibrado con problemas tipo de la literatura y yacimientos simples.

Debido a que los datos del yacimientos cuentan las más de las veces con erroes de medición, durante la última fase, implementamos un método basado en B-splines del tipo multinivel para suavizar la superficie. Con este trabajo, el módulo ofrece alternativas al usuario dispuesto a experimentar con diferentes técnicas para reconstruir superficies a partir de datos dispersos.

Durante las últimas fases, se trabajó conjuntamente con el grupo de Interfaz y Visualización Gráfica, a fin de diseñar y modificar los requerimientos funcionales, así como definir y calibrar las tareas a realizar y las facilidades gráficas para el Módulo Generador de Mallas 3D GRID 1.0.

En la Figura 2.1 se puede observar un diagrama esquemático de las tareas desarrolladas y que contempla el módulo GRID.

Organigrama de las Tareas involucradas en el Módulo GRID

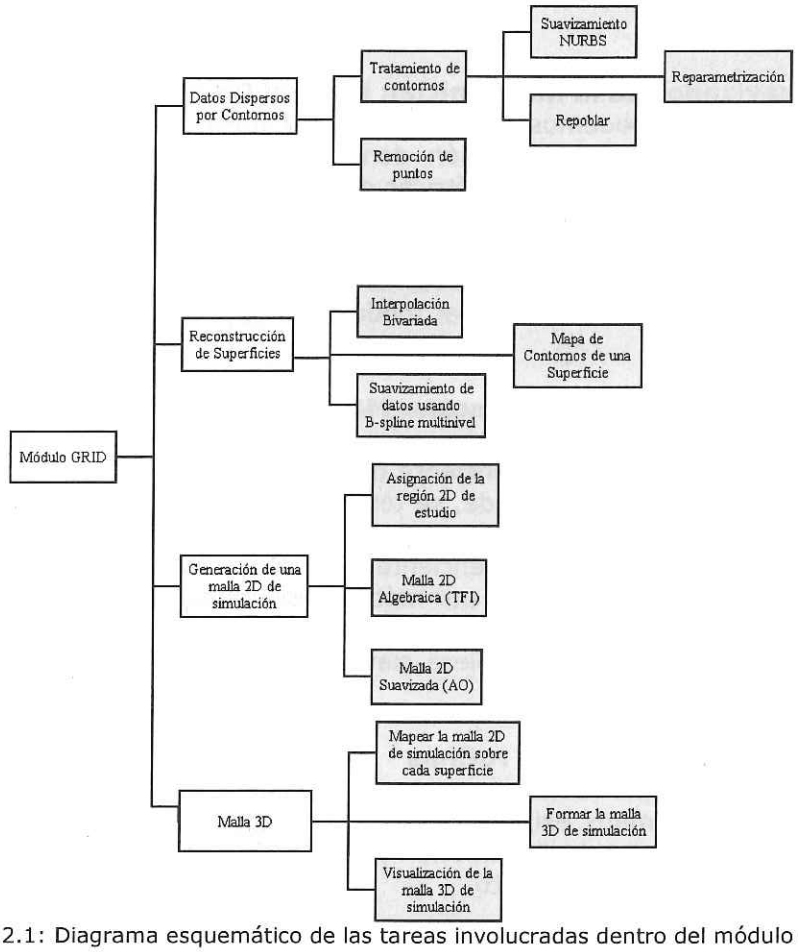


Figura 2.1: Diagrama esquemático de las tareas involucradas dentro del módulo GRID.

2.1.2 Actividades desarrolladas

Para lograr cumplir satisfactoriamente el proyecto, se cubrieron las tareas señaladas en el apartado anterior, mismas que brevemente son descritas por submódulos o subtareas desarrolladas.

Sub-módulo Generador de la malla algebraica
Una malla 2D sobre el plano de referencia está delimitada por lo que se conoce como contornos de control de la malla. Existen contornos de control internos y externos. Los externos corresponden a segmentos de frontera de la región 2D. Los internos son útiles para definir secciones de la región donde se desea seconcentren las líneas curvilíneas de la malla 2D. Contando con los puntos distribuidos sobre cada contorno de control externo, el paso siguiente es llevar a cabo una interpolación entre cada dos contornos de control opuestos, para de esta forma obtener una malla estructurada por interpolación. Esta será la malla inicial con la que se cuente o bien, sobre la que se trabaje si el usuario considera que es suficiente. Este problema fue resuelto usando el Método de

Interpolación Transfinita. Para lograr esto, se parametriza cada segmento de control opuesto y se distribuyen los puntos de manera uniformemente respetando la forma del polígono.

Sub-módulo Suavizador de la malla inicial: Método Discreto AO
A partir de la malla inicial, podemos obtener una malla suave y casi ortogonal a través de un procedimiento directo de optimización donde la función objetivo es un funcional discreto que mide las propiedades geométricas de área y ortogonalidad de cada celda. En esta primera estapa de construcción del simulador, se consideró incorporar esta modalidad como una opción para que el usuario pueda elegir entre la malla algebraica obtenida en el paso anterior, o bien obtener una suave y casi ortogonal con este procedimiento. Este módulo hace uso del módulo de optimizador de Newton Truncado con Región de Confianza.

Sub-módulo Optimizador de Gran Escala: Método Newton Truncado
El planteamiento discreto para la construcción de una malla a través de un funcional, involucra contar con un optimizador eficiente que nos conduzca al óptimo del problema. Cuando la malla es de dimensión grande, se tendrá un problema de optimización de gran escala. Para resolver adecuadamente problemas no lineales de gran escala, es necesario contar con un módulo de optimización eficiente. Uno de los Métodos de Optimización que hemos estudiado para construir mallas por métodos discretos es el Newton Truncado con Región de Confianza. Este optimizador se ha implementado dentro de la rutina `solver.f90`. El optimizador será una pieza clave en algunas tareas que deberá resolver el simulador numérico.

Codificación y documentación de rutinas
Con el fin de generar documentos que describan cada una de las rutinas y módulos involucrados bajo un formato estándar de documentación, fueron documentadas las rutinas siguiendo un Estándar de Codificación y Documentación de Productos provisto por los Ingenieros de Software del proyecto.

Sub-módulo: Reconstrucción de una superficie
Durante la primera fase nuestro interés fue entender el problema de reconstruir una superficie a partir de los mapas de contorno de profundidades y de isopropiedades del yacimiento. Para esto, se investigó una técnica de reconstrucción de superficies a partir de los contornos usando curvas NURBS.
Se desarrolló una versión experimental en C++ con salida en VRML para la visualización de la geometría del yacimiento, a partir del conjunto de contornos de profundidad. Se observó la dificultad para trabajar con esta técnica dado que se requiere que las curvas del mapa de contornos se encuentren ordenadas. El trabajo desarrollado se planteó exclusivamente para el caso en que los datos que describen la cima son mapas de contorno. Los datos que entrega el programa pueden observarse por VRML.
En la segunda parte, se estudiaron técnicas para reconstruir superficies a partir de la interpolación de datos dispersos. Para esto se elaboró un módulo en donde se implementa la técnica de Interpolación Bivariada y el suavizamiento de datos basada en procedimientos de aproximación local, desarrollada por Hiroshi Akima. Este método parte de un conjunto de puntos xi y yi (para i=0,1,...,n) de una malla rectangular y obtiene una función z= z(x,y), en donde las derivadas parciales de primer orden son continuas. Esta técnica permite eliminar excesivas ondulaciones entre los puntos dados de la malla.

Así, con este módulo se logró interpolar cualquier punto (x,y), que pertenezca al dominio de trabajo, al encontrar un triángulo que lo contiene, estimar su primer y segunda derivada parcial. Sin embargo, como se hizo notar a lo largo de ese periodo, los datos que entrega el Geólogo para la descripción del yacimiento provienen de mediciones, mismas que cuentan con errores. Por lo que una interpolación de los datos dispersos no es suficiente para observar la tendencia de los datos y por ende, la forma de la superficie.
Para resolver eficientemente esta cuestión se planteó que la superficie fuese reconstruida a través del suavizamiento de datos. Se propuso aplicar un método muy interesante de ajuste por B-spline a través de una técnica multinivel. Esto permite elegir un grado de suavizamiento de los datos obteniendo una representación satisfactoria para el nivel 7. Los resultados que obtuvimos de pruebas con diferentes tipos de regiones y datos dispersos obtenidos a través de problemas tipo (como la función de Franke) nos muestran que este método es muy efectivo para el caso cuando la propiedad a medir o la profundidad es un parámetro que presenta variaciones suaves.

Diseño de la interfaz de gráfica con el usuario
Nuestro grupo de trabajo resuelve de manera eficiente el problema de construir mallas 2D sobre regiones muy irregulares. La línea de trabajo se centra en observar propiedades geométricas sobre las celdas como lo son la suavidad en las líneas curvilíneas, la uniformidad en el área de las celdas y la ortogonalidad de las líneas curvilíneas. Para lograr entender el problema de la construcción de la malla 3D para ser usada en un proceso de simulación de yacimientos de hidrocarburos, se estudiaron varios simuladores y sus módulos para construir la malla 3D, algunos de estos paquetes son el ECLIPSE y el CMG. Con el módulo GRID de ECLIPSE aprendimos a reconocer el valor agregado que tiene el Tratamiento de Contornos con el fin de reconstruir eficientemente las capas o superficie sobre el área de trabajo. Con el simulador CMG aprendimos que la interfaz gráfica con el usuario es una pieza clave para el éxito de su uso. El usuario debe poder manejar adecuadamente todas las subtareas que se contempla cuente el simulador para construir la malla 3D, y asignar sus propiedades. Esto debe hacerse de manera automática con opciones predeterminadas. Para esto se elaboraron dos trabajos de suma importancia para el diseño del módulo: una tiene que ver con los requerimientos funcionales para obtener la malla 3D, la otra tiene que ver con la Visualización de la malla 3D. El diseño de Interfaz Gráfica presentado sigue un estilo cercano a Ingeniería de Software, donde se describe primeramente un Análisis General para posteriormente describir el Diseño del Sistema.

Herramientas de Visualización Gráfica previa
Conforme desarrollamos las sub-tareas programadas para cada elemento del Módulo Generador de Mallas 3D, fue necesario programar y diseñar pequeños programas gráficos que nos permitieron visualizar nuestros resultados antes de incorporar las rutinas al simulador para de esta manera, comparar los resultados obtenidos por el simulador y nuestro desarrollo previo. Estos programas fueron definidos conforme el proyecto avanzó a lo largo del año.

Sub-módulo: Tratamiento de los contornos
Al atacar el problema de la reconstrucción de superficies, pudimos notar que la densidad de puntos es un factor determinate de su forma. Por una parte, los datos dispersos del trabajo provenían de los mapas de contornos de las cimas, esto es, contaban con un patrón ordenado, o determinado a partir de la curva de nivel que representan. Logramos

observar que esa colección de curvas del mapa de contornos, no siempre se encontraban distribuidos de manera uniforme a lo largo de la curva, o bien concentrados en alguna sección, es decir, la curva así discretizada presentaba fuertes discontinuidades de clase C1. Esto provoca que la superficie así reconstruida presente ondulaciones o rugosidades en su curvatura. Por lo que diseñar y aplicar una técnica que primero nos permita suavizar cada curva del mapa de contorno y luego parametrizarla para distribuir a lo largo de su longitud de arco los puntos, era una tarea que debería ser resuelta de manera automática por el sistema. Para esto se diseñó este sub-módulo que permite de manera automática suavizar por medio de curvas NURBS y una técnica para parametrizar dichas curvas. Esta sub-tarea o módulo se aplica de igual manera para suavizar los contornos de control que definen una región 2D sobre el plano de referencia.

Sub-módulo: Optimizador puntual, Método de Newton Truncado

Cuando se construye la malla 2D, en algunas situaciones, es necesario suavizar las líneas de una sección de la región de estudio, ya sea porque se han introducido algunas modificaciones previas o bien, se desea centrar el estudio en determinada sección. Con este fin, se desarrolló una rutina que optimiza la malla 2D de manera puntual. Esto permite por una parte, suavizar mallas de dimensiones muy grandes, y por la otra, dar una primera forma de resolver el problema de las fallas presentes en yacimientos, al ser consideradas éstas como restricciones puntuales para la malla 2D sobre el plano de referencia. Esta idea de atacar el problema es interesante y novedosa.

Sub-módulo: Generador Experimental de Mallas 3D

Como se comentó en la introducción. Existen varias formas de describir el yacimiento a partir de las capas que lo conforman. Una primera es considerar una malla de referencia, mapearla sobre la superficie reconstruida de la cima del yacimiento y bajarla a un grosor constante. Esto es, se estarán considerando capas intermedias a partir de la cima de grosor constante donde se encontrará mapeada la malla 2D del plano de referencia. Las mallas 2D que generamos por medio del módulo Suavizamiento de Mallas a través del funcional AO, son mallas suaves y casi ortogonales, la malla 3D así reconstruida sigue lo que se conoce como la geometría corner-point, es decir, los vértices del paralepípedo se encuentran sobre una superficie no necesariamente alineada. Existen otras formas de construir la malla 3D: si en lugar de elegir un grosor vertical constante, elegimos un grosor constante pero en la dirección de la normal a la superficie, podremos por lo menos garantizar que las caras verticales con respecto a las correspondientes arriba y debajo de un bloque de celda, sí sean ortogonales. Pero esta idea será algo que recomendaremos como un trabajo posterior.

2.2 Sub-módulo Generador de la Malla Algebraica

2.2.1 Objetivos

El usuario del simulador se encargará de definir la región de estudio sobre la cual desea construir una malla plana, para posteriormente obtener una en 3D a partir de la reconstrucción de la geometría del yacimiento.

La región de estudio está compuesta de cuatro contornos de control, en este caso, curvas poligonales, que define la región plana de estudio sobre el plano de referencia. Para generar la malla plana, es necesario indicar el tamaño de la malla $m \times n$ a construir. La malla generada se logra a través del método de Interpolación Transfinita, mismo que se describe en los apéndices.

La definición de la región de estudio la hará a través de herramientas gráficas diseñadas para tal propósito por el equipo de Interfaz y Visualización Gráfica. El tamaño de la malla será introducido por el usuario a través de opciones de menú o de un panel de opciones. La interfaz se comunicará con el sub-módulo para obtener una malla 2D sobre el plano de referencia.

2.2.2 Descripción del sub-módulo

La subrutina `menugene`, construye una malla por interpolación algebraica TFI. Para lograrlo, la interfaz del sistema debe proveer información del tamaño de la malla a construir, el número de puntos con que cuenta el contorno de la región (la colección de puntos que en total definen los cuatro contornos de control), el tipo de región (simplemente conexa o no conexa, es decir con agujero) el número de puntos que conforman cada contorno de control, un arreglo que contiene todos los puntos de la región, y arreglos de trabajo. Estos arreglos de trabajo representan un espacio de memoria a usar por la rutina, al salir de ella y obtener la malla, la interfaz debe deshacerse de ese espacio de memoria de trabajo. En la salida se obtendrá una bandera `info` que indica si la malla fue construida exitosamente o no, y en caso negativo, la región se considera no admisible. De ser admisible, en el arreglo `red` se obtendrá la malla plana sobre la región. Con esto, la interfaz de programación con esta rutina deberá capturar esta información y de no ser admisible, debe así hacérselo saber al usuario del simulador a través de una ventana de mensaje de error. Una opción que incrementará el potencial del simulador, es informar de esta situación al usuario y proveerle sugerencias de qué hacer en este caso. Una sugerencia es aumentar la dimensión de la malla, o revisar si los contornos de control proveen una región plana donde sea posible construir una malla 2D admisible.

Es importante señalar que el usuario nunca proveerá el número de puntos con que cuenta la frontera de la región de estudio, esto lo hará la interfaz que llamará a la rutina para obtener la malla. El usuario solamente proveerá la mínima información necesaria a través de la interfaz gráfica diseñada para tal propósito. Los contornos de control que definen la región de estudio o sub-regiones internas es la información que requiere el sub-módulo. Una adecuada transferencia de esa información al sub-módulo permitirá construir sub-mallas sobre un región irregular, pero perfectamente determinada por los contornos de control, internos o externos.

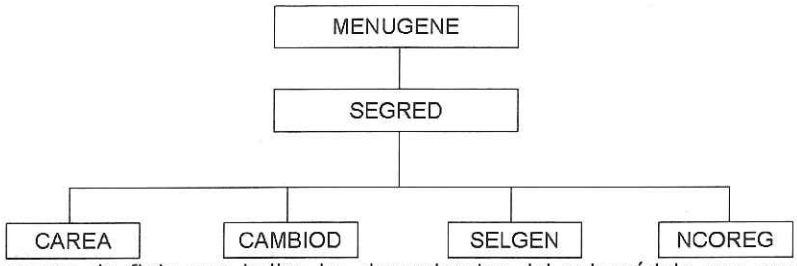


Figura 2.2.1: Diagrama de flujo que indica las dependencias del sub-módulo que genera la malla por interpolación transfinita.

2.2.3 Mallas obtenidas con este sub-módulo

A continuación se muestran algunas mallas obtenidas con este módulo.

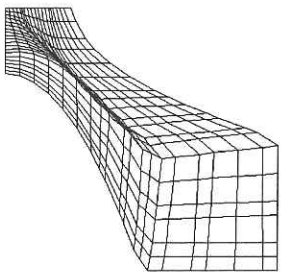


Figura 2.2.2 (A-TFI)

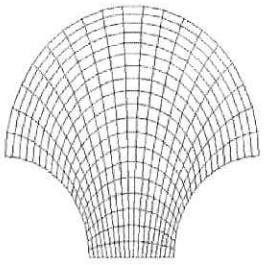


Figura 2.2.2 (B-TFI)

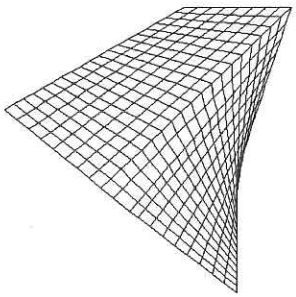


Figura 2.2.2 (C-TFI)

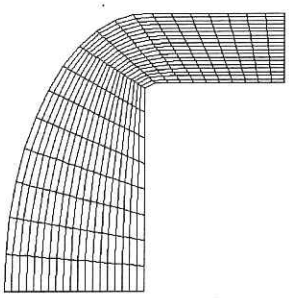


Figura 2.2.2 (D-TFI)

Como puede observarse, no en todos los casos es posible obtener una malla convexa (una malla convexa es aquella donde cada celda es un cuadrilátero convexo) con este procedimiento algebraico, por lo que en la mayoría de las veces, será necesario mejorarla por algún criterio que permita obtener mallas convexas, suaves y cercanas a la ortogonalidad. Los simuladores CMG y ECLIPSE solamente ofrecen esta modalidad. Para este simulador se tiene contemplado un suavizamiento de la malla para mejorarla y obtener mallas con líneas curvilíneas suaves y celdas casi ortogonales.

2.3. Sub-módulo Generador de Mallas Suaves y Casi-Ortogonales (Funcional AO)

2.3.1 Objetivos

A partir de la malla inicial, podemos obtener una malla suave y casi ortogonal a través de un método discreto. En esta primera fase del simulador, se pretende incorporar esta modalidad como una opción que el usuario pueda elegir, esto es, que pueda contar con la malla algebraica obtenida en el paso anterior, o bien obtener la suavidad y casi ortogonalidad de manera directa. Este módulo hace uso del módulo de optimizador de Newton Truncado con Región de Confianza.

2.3.2 Descripción del sub-módulo

Esta rutina recibe una malla sobre una región plana de referencia de dimensión $m \times n$ en formato `RED` y en la salida, devuelve, bajo el mismo arreglo, una malla suave y casi ortogonal. Esto se logra optimizando el funcional discreto de Área-Ortogonalidad (descrito en el Apéndice A) y haciendo uso del optimizador Newton Truncado con Región de Confianza, que describiremos en el siguiente apartado.

```
C     ****************************************************************
C
      subroutine menuop(mn,il,jl,n,idiwa,iww,iwa,ww,red,filini)
C
C     ****************************************************************
C
C     Esta rutina recibe una malla sobre una región plana de dimensión ilxjl
C     y en la salida, devuelve una malla suave y casi ortogonal. Esto se
C     logra usando el funcional de Área-Ortogonalidad descrito en el apéndice
C     y haciendo uso del optimizador Newton Truncado con Región de Confianza.
C
C   * Inputs
C
C     MN      - Total number of grid points.
C               INTEGER
C
C     IL      - Number of vertical grid points.
C               INTEGER
C
C     JL      - Number of horizontal grid points.
C               INTEGER
C
C      N      - Number of interior grid points: 2*(il-1)*(jl-1)
C               INTEGER
C
C     IWW     - Dimernsion of array WW = n*24 + mn + 3*(10*n)
C               INTEGER
C
C     IDIWA   - Dimension of array
C               IWA = 16*n + 4*(10*n) + 2*(il-1)*(jl-1) + 4
C               INTEGER
C
C
C      IWA    - Work array of dimension IDIWA
C               INTEGER
C
C      WW     - Work array of dimension IWA
C
C
C   * Outputs
C
```

```
C     RED      - One dimensional array containing the grid points
C                RED(MN)
C
C     FILINI   - Last initial grid file name.
C                CHARACTER*23
C
C
```

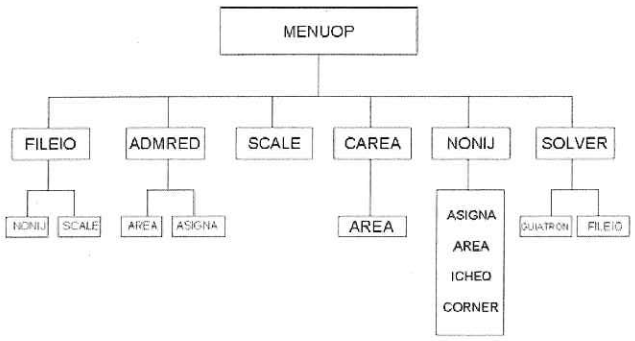


Figura 2.2.3: Diagrama de flujo indica las dependencias de las rutinas que suavizan la malla por el método discreto de Área-Ortogonalida.

2.3.3 Mallas obtenidas con este sub-módulo

A continuación se muestran algunas mallas obtenidas con este módulo.

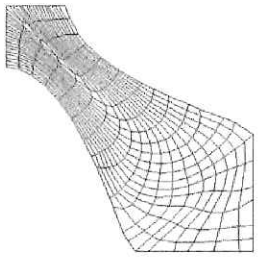


Figura 2.2.4 (A-AO)

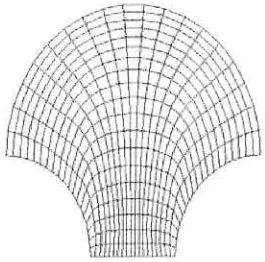


Figura 2.2.4 (B-AO)

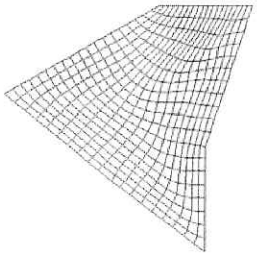


Figura 2.2.4 (C-AO)

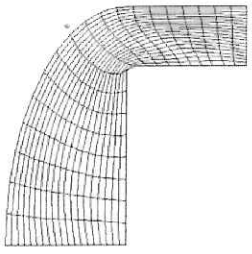


Figura 2.2.4 (D-AO)

Como puede observarse, las Figuras 2.2.4 (A-AO), B-AO y C-AO presentan una mejor distribución de los nodos interiores de la malla plana, logrando con ello observarse la propiedad de casi ortogonalidad y suavidad de las celdas, sin cambiar la región y la distribución de puntos sobre los contornos de control. En el caso de las Figuras 2.2.4(A-AO) y C-AO, las mallas aquí obtenidas, son convexas. Sin embargo en el caso de la Figura 2.2.4 (D-AO), la malla obtenida no es convexa, por lo que será necesario llevar a cabo una redistribución de los puntos sobre los contornos de control o bien integrar mejores suavizadores de la malla que garanticen su convexidad.

Conforme se requieran otros funcionales, este sub-módulo: Suavizamiento de la Malla Plana, se irá modificando para aceptar esas opciones, esto es, el módulo permite el mantenimiento de sus partes, de forma que puedan ser revisadas, sustituidas, actualizadas.
La forma en que se encuentran programados los módulos aquí descritos permite su crecimiento y mantenimiento.

2.4 Sub-módulo Optimizador para el problema de la Generación de Mallas Suaves

2.4.1 Objetivos

La generación de la malla plana por un método discreto variacional, involucra un problema de optimización de gran escala, cuando la malla es de dimensión grande. Contar con un optimizador eficiente que nos conduzca al óptimo es fundamental para obtener la malla con las características geométricas deseables: suave y convexa.
Uno de los Métodos de Optimización que hemos estudiado para construir nuestras mallas por métodos discretos, es el Newton Truncado con Región de Confianza. Este optimizador se ha implementado dentro de la rutina SOLVER para resolver de manera eficiente el problema de generación de mallas. El optimizador será una pieza clave en algunas tareas que se pretende tenga el simulador, como por ejemplo en la solución numérica de ecuaciones no lineales.

2.4.2 Descripción del sub-módulo

Hemos investigado diferentes métodos de optimización confiables y eficaces para la solución de problemas de optimización no lineal con un gran número de variables, uno de ellos es el Método de Newton Truncado. Este método será necesario para poder obtener un óptimo del funcional discreto a emplear, por ejemplo, para obtener una malla suave y cerca a la ortogonalidad minimizando el funcional discreto de Área-Ortogonalidad.

Las rutinas que siguen a continuación, usan este método para obtener una malla suave y convexa.

```
C     ****************************************************************
C
      subroutine solver(mn,ibo,il,jl,n,iww,id3,iprint,sigma,
     >    red,ww,ires,filnam,ifcng,inocon,idiwa,iwa,
     >    alfa_lim,alfaprom,chamet,xmin,xmax,ymin,ymax,
     >    alfapreal,action,alfamin,alfamax)
C
C     ****************************************************************
C
C      This subroutine solves a minimization problem
C      arising from generating two dimensional grids from
C      a Direct Variational formulation.
C      The grids are generated by directly minimizing the Area
c      Orthogonality functional
C
C
C     TRON is a trust region Newton method for the solution of large
C     bound-constrained optimization problems. TRON uses a gradient projection
C     method to generate a Cauchy step, a preconditioned conjugate gradient
C     method with an incomplete Cholesky factorization to generate a direction, and
C     a projected search to compute the step. The use of projected searches, in
C     particular, allows TRON to examine faces of the feasible set by generating a
C     small number of minor iterates, even for problems with a large number of
C     variables. As a result TRON is remarkably efficient at solving large bound-
C     constrained optimization problems.


C
C     * Inputs
C
C       MN      - Total number of grid points
C                 INTEGER
C
C       IBO     - = 2*(2*IL + 2*JL - 4) - total number of boundary points
```

```
C              of the grid
C              INTEGER
C
C     IL       - Number of vertical grid points.
C              INTEGER
C
C     JL       - Number of horizontal grid points.
C              INTEGER
C
C     N        - Total number of coordinates of interior grid points
C              INTEGER
C
C     IWW      - Dimension of vector WW. Must be greater than
C              n*24 + mn + 3*(10*n
C              INTEGER
C
C     ID3      - Dimension of vector IPRINT
C              INTEGER
C
C     IPRINT(1) - Specifies the frecuency of the output:
C                 < 0: No output is generated.
C                 = 0: Output only at first and last iterations.
C                 > 0: Output every iteration.
C     IPRINT(2) - Specifies the type of output to be generated:
C                 = 0: Iteration count, number of function calls,
C                      function value, norm of the gradient, and
C                      number of non convex cells.
C                 = 1: same as 0, plus vector of variables and
C                      gradient vector at the initial point.
C                 = 2: same as 1, plus vector of variables.
C     IPRINT(3) - Specifies the way to see the output:
C                 = 0: Only in the screen.
C                 = 1: Only in an output file.
C                 = 2: Both options: screen and output file.
C                 INTEGER(ID3)
C
C     SIGMA    - Weigtht for the functional.  0 <= SIGMA <= 1
C              double precision
C
C     RED      - Array containing the points of the grid
C              double precision(MN)
C
C    WW     -  Work array of dimension IWW
C              DOUBLE PRECISION
C
C
C    IFCNG    - The functional to be used
C               Area- Ortogonality
C
C               INTEGER
C
C     IDIWA   - Dimension of array
C               IWA = 16*n + 4*(10*n) + 2*(il-1)*(jl-1) + 4
C               INTEGER
C
C
C      IWA    - Work array of dimension IDIWA
C               INTEGER
C
C   ALFA_LIM -  The least value to consider that a cell area is
c               positive, i.e., convex
C
C     * Outputs
C
C     RED      - Solution of the optimization method.
```

```
C               double precision(MN)
C
C     G       - Gradient vector at the solution point.
C               double precision(N)
C
C     IRES    - Flag :
C               = 1 The solution grid was written to a file.
C               = 0 The solution grid was NOT written to a file.
C               INTEGER
C
C     INOCON - Number of nonconvex grid
C               INTEGER
C
```

El optimizador será una pieza clave en varias tareas con las que contará el simulador, como por ejemplo en la solución numérica de ecuaciones no lineales; por lo que será muy importante adaptar las rutinas del algoritmo al problema de optimización no-lineal que se desee abordar.

En el Apéndice K.1 se enlistan el manejo de las instancias de las subrutinas señaladas hasta el momento.

2.5 Sub-módulo para Reconstruir Superficies

2.5.1 Objetivos

Una pieza clave para obtener una malla 3D de simulación, es la reconstrucción de cada una de las capas o superficies de propiedades que nos interesa medir sobre el yacimiento. Usualmente, el geólogo entrega al grupo de especialistas de la construcción de la malla 3D un mapa de contornos de la cima del yacimiento, sobre esa cima debe proyectarse una malla 2D desde el plano de referencia para contar con una malla sobre la superficie de la cima, esto es, cada nodo de la malla plana se encuentra sobre la superficie. Para lograr esto es necesario reconstruir la superficie del mapa de contornos.

Para la reconstrucción de cada capa o superficie, probamos algunos de los métodos clásicos para interpolación de datos dispersos que nos permitieron obtener una buena representación de superficies. Hicimos pruebas con problemas reportados en la literatura como la Función de Franke y con datos provenientes tipo de un yacimiento sin falla. Calibramos los parámetros involucrados en los métodos y ganamos experiencia en el tipo de capas o superficies involucradas en yacimientos de hidrocarburos.

Se estudiaron tres métodos para reconstruir superficies. El primero se basó en obtener por cada curva de nivel del mapa de contornos una representación NURBS (Non Uniform Racional B-Splines) e interpolar entre cada curva así representada. El segundo método es el algoritmo ACM 526 para la interpolación bivariada. El tercer método estudiado es el Suavizamiento de datos por B-spline Multinivel. El primero fue una primera forma de atacar el problema, debido a las dificultades para ordenar o contar con un ordenamiento de los puntos de la curva no resultó ser un método eficiente, sin embargo se sugirieron algunas formas de resolver el problema de interpolación entre curvas de nivel, este método fue desehechado con la finalidad de estudiar otros que no solamente resuelvan el problema para mapas de contorno, sino que abarquen diferentes colecciones de datos dispersos.

2.5.2 Reconstrucción de Contornos de datos a partir de curvas NURBS

La geometría de un yacimiento puede ser descrita a través de una colección de capas o superficies que al ser debidamente unidas conforman la forma tridimensional del yacimiento. La tarea que resuelve este sub-módulo es la reconstrucción del yacimiento a través de sus capas.
Usualmente la información que nos provee un estudio geológico del yacimiento, es a través de una colección de datos que siguen curvas de nivel, estas son los mapas de contornos. En otras situaciones, la información es provista a través de una malla gruesa que abarca la región de estudio.
La reconstrucción de una capa del yacimiento, requiere de métodos de interpolación sofisticados. Debido a que generalmente se proporcionan mapas de contorno, se propuso usar una técnica moderna para resolver este problema como una reconstrucción de mapas de contoros. Se encontró que las superficies NURBS proporcionan una herramienta que tiene varias características para describir de manera eficiente curvas, por lo que se desarrolló un método para la reconstrucción de la geometría del yacimiento usando superficies NURBS implementado en el programa `Reconst-0.0`.

El programa `Reconst-0.0` reconstruye la geometría de un yacimiento simple a partir de un conjunto de curvas de nivel. Para poder usarlo, es necesario tener instalada la biblioteca `NURBS++` la cual se puede obtener libremente en `Internet`. Una vez hecho

esto, el programa requerirá pasarle el archivo de puntos de las curvas de nivel (ver formato en el archivo Leeme.txt) y una vez hecho esto, se le debe proveer el número de renglones y columnas de la malla sobre la superficie reconstruida (se espera que sean valores mayores que cero), con esta información el programa generará un archivo con los respectivos puntos pertenecientes a la superficie. Adicionalmente el programa genera un archivo en formato `vrml` con el que se puede ver la superficie reconstruida.

Las funciones que componen Reconst-0.0 son:

```
/*Función que ordena dos vectores de puntos en el espacio de acuerdo
  a sus distancias mínimas */
void    ordenapuntos(Vector_Point3Df &, Vector_Point3Df &);

/*Funciones para el cálculo de la distancia de puntos en el espacio y
  el espacio homogéneo*/
double  distancia(Point3Df &,Point3Df &);
double  distancia2(HPoint3Df &,HPoint3Df &);

/*Función que calcula la distancia más corta entre dos curva NURBS */
double  distmcorta(double, NurbsCurvef &, NurbsCurvef &);

/*Función que lee dos datos de las isocurvas en un archivo*/
void    cargaisocurvas(std::string,std::vector<Vector_Point3Df> &);

/*Función que imprime los puntos de las isocurvas*/
void    imprimepuntos(std::vector<Vector_Point3Df>);

/*Función que ordena todos los puntos de las isocurvas*/
void    ordenavecpuntos(std::vector<Vector_Point3Df>&);

/* Función que construye la curva de interpolación NURBS*/
void    construyeisocurvas(std::vector<Vector_Point3Df>&,NurbsCurveArrayf &);

/*Función que construye la nuevas curva de interpolación usando los
puntos   con distancias mÃnimas*/
void    refinainterpolacion(NurbsCurveArrayf &);

/*Función que construye la superficie Skinned usando la curva NURBS
  de interpolación calculadas en la anterior función*/
void    construyesuperfice(NurbsCurveArrayf &,PlNurbsSurfacef &);

/*Función que evalua la superficie NURBS reconstruida y genera
  el archivo puntos.txt*/
void    evaluasuperficie(PlNurbsSurfacef &,int,int);
```

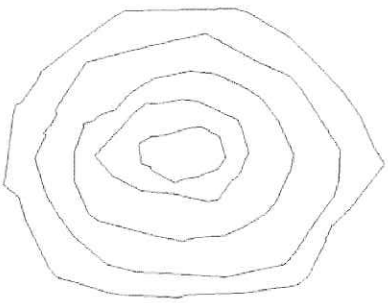


Figura 2.5.1a: Colección de contornos originales

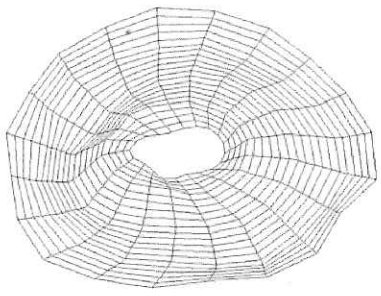


Figura 2.5.1a: Colección de contornos interpolados malla de 20x20

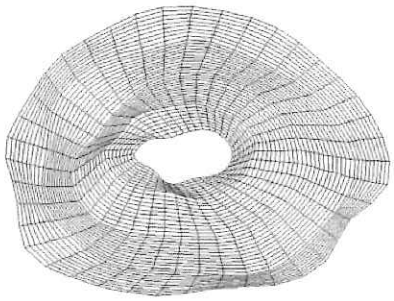


Figura 2.5.1b: Colección de contornos interpolados malla de 40x40

Figura 2.5.1c: Superficie obtenida. Modelada con VRML

Se observa que las superficies así obtenidas presentan muchas rugosidades. Esto se debe fundamentalmente a la cantidad de puntos con que se describe cada curva de nivel. Una idea de resolver este problema es a través del Tratamiento de Contornos que involucra el Suavizamiento de la curva y la Parametrización.

En el apéndice se señala el algoritmo implementado y la técnica de interpolación usada. La fase siguiente es mejorar el algoritmo de interpolación usando el método Inverse distance Weighting o bien C1 Continuos Terrain Reconstruction.

2.5.3 Reconstrucción de Superficies por Interpolación Bivariada

Para la reconstrucción de superficies por el Método de Interpolación Bivariada, se estudió el algoritmo ACM 526 de Hiroshi Akima, el cual es software libre. Este método parte de un conjunto de puntos (x_d, y_d) (para $d = 1,...,n_c$) arbitrariamente distribuidos sobre una región de estudio y obtiene una función suave que interpola a los datos $z_d = F(x_d, y_d)$. Adicionalmente esta técnica permite eliminar excesivas ondulaciones entre los puntos dados de la malla.

Usualmente, la solución a este problema se puede dividir en tres partes:

1. Construir una triangulación para Ω.
2. Estimar las derivadas parciales de F con respecto a x y y en cada nodo y usando la información de los nodos más cercanos.
3. Para un punto (x, y) en Ω, determinar el triángulo que contiene a dicho punto, y calcular un valor interpolado $z = F(x, y)$, usando los valores de los datos y las derivadas parciales de cada vértice del triángulo.

El algoritmo ACM 526 aproxima un polinomio bivariado de grado 5 de la forma

$$F(x, y) = \sum\sum q_{jk} x^j y^k$$

a través de condiciones de continuidad en las derivadas y sobre los lados del triángulo donde se encuentre (x, y) se obtiene de manera unívoca el polinomio.

Para hacer uso de la rutina, se le debe proveer la cantidad de puntos dispersos sobre los cuales se debe interpolar, y la malla rectangular sobre la cual será evaluada la función $F(x,y)$.

Nota: Como se comenta en el Apéndice C, el método consiste en construir una triangulación sobre el casco convexo Ω que contiene a los puntos y aproximar las derivadas parciales de F sobre cada nodo (x_d, y_d). Si se ha evaluado ya $F(x,y)$ sobre una malla rectangular, y se desea evaluar de nuevo sobre otra malla que contiene nodos distintos, no es necesario volver a construir la triangulación sobre Ω y calcular las derivadas sobre los nodos (x_d, y_d), el programa permite la opción *reverse communication*, lo que permite hacer uso de la información guardada en memoria y simplemente evaluar el polinomio bivariado $F(x,y) = \sum\sum q_{jk} x^j y^k$ que le corresponda a (x,y) en Ω. Esto nos permite un ahorro considerable al evitar rehacer cálculos.

A continuación se muestra el encabezado de la rutina principal `idbvip.f90`

```
!/* Esta subrutina lleva a cabo la interpolación bivariada de los datos
!*  irregulares X, Y.
!*
!* Hecho por:
!@author Hiroshi Akima
!@version 08 / 1975
!@version Fue reescrito en 1976
!@version Nueva versión en 08 / 1984
!*
!* Últimas modificaciones por:
!@author Michael Pernice  NCAR's Scientific Computing Division
!@version 02 / 1985
!*
!@version Modificado el 23 / 01 / 2003.
!*
!* History:
!*
!* The original version of BIVAR was written by Hiroshi Akima in
!* August 1975 and rewritten by him in late 1976.  It was incorporated
!* into NCAR's public software libraries in January 1977.  In August
!* 1984 a new version of BIVAR, incorporating changes described in the
!* Rocky Mountain Journal of Mathematics article cited below, was
!* obtained from Dr Akima by Michael Pernice of NCAR's Scientific
!* Computing Division, who evaluated it and made it available in February,
!* 1985.
!*
!* References:
!*
!* Hiroshi Akima,
!* Algorithm 526,
```

```
!* A Method of Bivariate Interpolation and Smooth Surface Fitting
!* for Values Given at Irregularly Distributed Points,
!* ACM Transactions on Mathematical Software,
!* Volume 4, Number 2, June 1978.
!*
!* Hiroshi Akima,
!* On Estimating Partial Derivatives for Bivariate Interpolation
!* of Scattered Data,
!* Rocky Mountain Journal of Mathematics,
!* Volume 14, Number 1, Winter 1984.
!*
!* Method:
!
!* The XY plane is divided into triangular cells, each cell having
!* projections of three data points in the plane as its vertices, and
!* a bivariate quintic polynomial in X and Y is fitted to each
!* triangular cell.
!*
!* The coefficients in the fitted quintic polynomials are determined
!* by continuity requirements and by estimates of partial derivatives
!* at the vertices and along the edges of the triangles.  The method
!* described in the rocky mountain journal reference guarantees that
!* the generated surface depends continuously on the triangulation.
!*
!* The resulting interpolating function is invariant under the following
!* types of linear coordinate transformations:
!* 1) a rotation of the XY coordinate system
!* 2) linear scale transformation of the Z axis
!* 3) tilting of the XY plane, i.e. new coordinates (u,v,w) given by
!*     u = x
!*     v = y
!*     w = z + a*x + b*y
!* where a, b are arbitrary constants.
!*
!* complete details of the method are given in the reference publications.
!*
!* Discussion:
!*
!* The data points must be distinct and their projections in the
!* X-Y plane must not be collinear, otherwise an error return
!* occurs.
!*
!* Purpose:
!*
!* To provide bivariate interpolation and smooth surface fitting for
!* values given at irregularly distributed points.
```

```
!*
!* The resulting interpolating function and its first-order partial
!* derivatives are continuous.
!*
!* The method employed is local, i.e. a change in the data in one area
!* of the plane does not affect the interpolating function except in
!* that local area.  This is advantageous over global interpolation
!* methods.
!*
!* Also, the method gives exact results when all points lie in a plane.
!* This is advantageous over other methods such as two-dimensional
!* Fourier series interpolation.
!*
!* Usage:
!*
!* This package contains two user entries, IDBVIP and IDSFFT, both
!* requiring input data to be given at points
!* ( X(I), Y(I) ), I = 1,...,N.
!*
!* If the user desires the interpolated data to be output at grid
!* points, i.e. at points
!* ( XI(I), YI(J) ), I = 1,...,NXI, J=1,...,NYI,
!* then IDSFFT should be used.  This is useful for generating an
!* interpolating surface.
!*
!* The other user entry point, IDBVIP, will produce interpolated
!* values at scattered points
!* ( XI(I), YI(I) ), i = 1,...,NIP.
!* This is useful for filling in missing data points on a grid.
!*
!@see IDTANG,IDLCTN,IDPDRV,IDPTIP.
!*/
!*
      SUBROUTINE idbvip ( md, ndp, xd, yd, zd, nip, xi, yi, zi )
      IMPLICIT NONE
!*
!* Parameters:
!*
!/* Input, integer MD, mode of computation.   MD must be 1,
!*  2, or 3, else an error return occurs.
!*
!*  1: if this is the first call to this subroutine, or if the
!*  value of NDP has been changed from the previous call, or
!*  if the contents of the XD or YD arrays have been changed
!*  from the previous call.
!*
```

```
!*  2: if the values of NDP and the XD and YD arrays are unchanged
!*  from the previous call, but new values for XI, YI are being
!*  used.  If MD = 2 and NDP has been changed since the previous
!*  call to IDBVIP, an error return occurs.
!*
!*  3: if the values of NDP, NIP, XD, YD, XI, YI are unchanged from
!*  the previous call, i.e. if the only change on input to IDBVIP is
!*  in the ZD array.  If MD = 3 and NDP or NIP has been changed since
!*  the previous call to IDBVIP, an error return occurs.
!*
!*  Between the call with MD = 2 or MD = 3 and the preceding call, the
!*  IWK and WK work arrays should not be disturbed. !*/
!*
!/* Input, integer NDP, the number of data points (must be 4 or
!*  greater, else an error return occurs). !*/
!*
!/* Input, real ( kind = 8 ) XD(NDP), Y(NDP), the X and Y coordinates
!*  of the data points. !*/
!*
!/* Input, real ( kind = 8 ) ZD(NDP), the data values at the data points.
!*/
!*
!/* Input, integer NIP, the number of output points at which
!*  interpolation is to be performed (must be 1 or greater, else an
!*  error return occurs). !*/
!*
!/* Input, real ( kind = 8 ) XI(NIP), YI(NIP), the coordinates of the
!*  points at which interpolation is to be performed. !*/
!*
!/* Output, real ( kind = 8 ) ZI(NIP), the interpolated data values.
!*/
!*
```

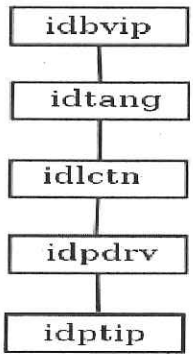


Figura 2.5.2: Diagrama de flujo de la rutina idbvip.f90

Una descripción breve de las tareas de cada subrutina es la siguiente:

- `idtang`, se encarga de realizar la triangulación del casco convexo donde se encuentran los nodos.
- `idlctn`, localiza los puntos con los que se va a realizar la interpolación en el dominio de trabajo.
- `idpdrv`, estima todas las derivadas parciales en los nodos (x_d, y_d).
- `idptip`, se encarga de evaluar el polinomio de interpolación en éstos nodos, regresando valores de la malla rectangular.

A continuación se muestran dos superficies obtenidas con esta rutina sobre una colección de nodos que provienen de un mapa de contornos de prueba.

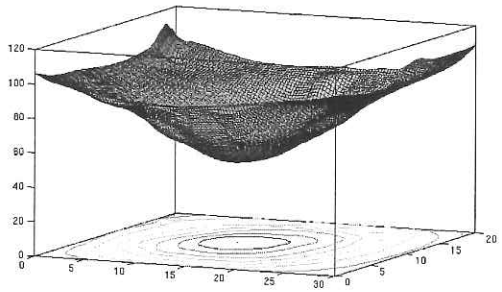


Figura 2.5.3: Superficie obtenida con la rutina idbvip usando los contornos originales.

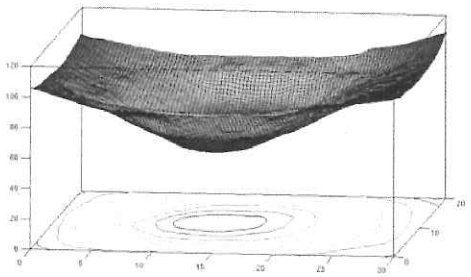


Figura 2.5.4: Superficie obtenida con la rutina idbvip usando los contornos interpolados.

Como se observa, en la segunda figura, la superficie obtenida sobre contornos interpolados, presenta mejores resultados, en ella ya no se observan rugosidades superfluas de consideración. Es por esto que se sugiere darle un tratamiento previo a los mapas de contornos antes de reconstruir la capa o superficie. En el Apéndice C se muestra la teoría detrás del Método de Interpolación Bivariada datos dispersos en 3D.

En el Apéndice K.2. son descritas las instancias de las rutinas empleadas para este método de interpolación bivariada.

2.5.4 Reconstrucción por Suavizamiento B-spline a través de una técnica multinivel

Este método requiere únicamente del ingreso de una colección de puntos $P=\{(x_c,y_c,z_c)\}_{c=1}^{n}$ con los cuales se construye la superficie suave, así que es necesario presentar las condiciones bajo las cuales es posible utilizar este método. Se puede entender por superficie a la gráfica de $(x,y,f(x,y))$ donde $f:\Omega\to\mathbb{R}$, por lo que observamos la condición para la colección de datos de entrada *P*,

$$\text{si } z_i \neq z_j \text{ para } i\neq j \Rightarrow (x_i,y_i)\neq(x_j,y_j),$$

ya que de no ser así, no estaríamos hablando de una función, con esto también se quiere decir que es posible repetir en la colección de datos a una triada completa, sin que esto sea perjudicial para el método.

Otra condición del método, es que Ω sea una región rectangular en el plano que contenga al conjunto $\{(x_c,y_c)\}_{c=1}^{n}$ de las primeras dos entradas de *P*, es decir de la forma $\Omega=\{(x,y)|0\le x<m, 0\le y<n\}$ donde m, n son naturales. Así, bien, la superficie suave resultante de este método será construida únicamente en esta región acotada.

El algoritmo entregable MBA con refinamiento se encuentra programado dentro de la rutina `mba_ref`, la cual se programó en Fortran90. Para hacer uso de esta rutina, se leen los datos de un archivo de texto plano (fichero de entrada) y se deposita la salida en un archivo de texto plano (fichero de salida), se recomienda dar extensión `.dat` por ser ficheros de datos.

El programa `mba_ref` emplea las siguientes funciones y subrutinas:

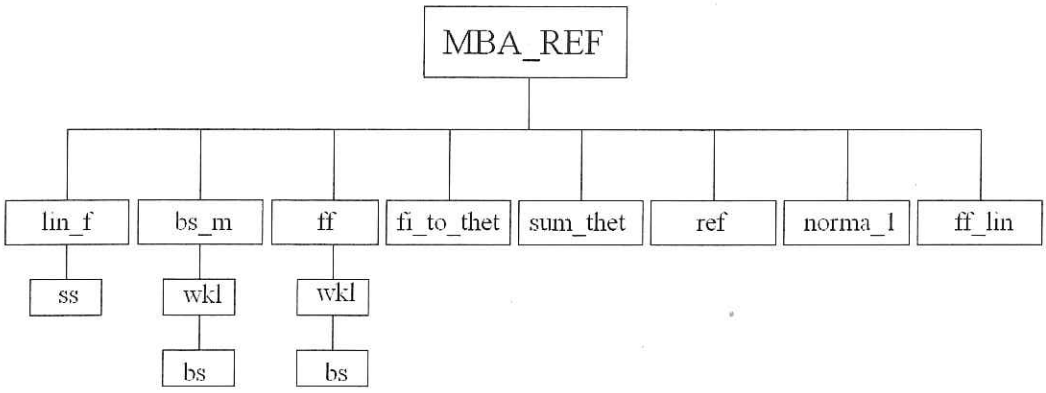


Figura 2.5.6: Diagrama de las rutinas empleadas por est sub-módulo

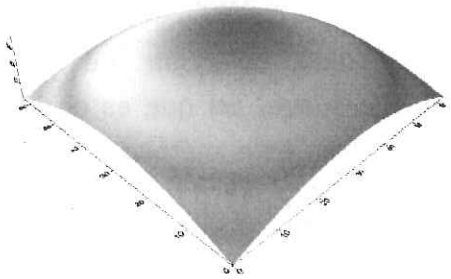


Figura 2.5.7a: Superficie obtenida a partir de una función analítica

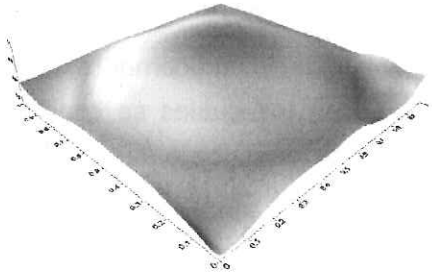


Figura 2.5.7.b: Superficie reconstruida por la la técnica de suavizamiento B-splinen usando 5 niveles y dimensión de escala 1 1.

Y un ejemplo para una superficie, dadas curvas de nivel es el siguiente.

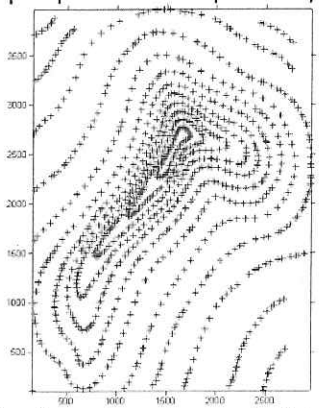


Figura 2.5.8a: Mapa de contornos de datos.

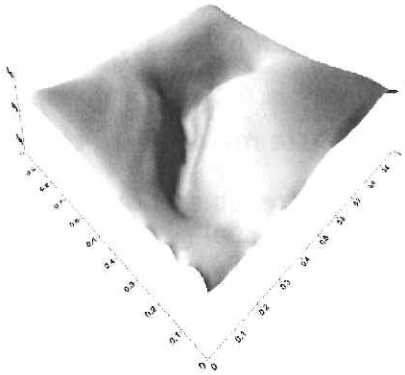


Figura 2.5.8b: Superficie reconstruida.

En el Apéndice K.3 se describen las instancias para las rutinas empleadas para el método de Suavizamiento por B-spline usando una técnica multinivel

2.6 Diseño de la Interfaz Gráfica

2.6.1 Objetivos

A partir de la experiencia obtenida con el módulo GRID de ECLIPSE y el paquete ArgusOne, se hizo una: Propuesta para el diseño de la Interfase Gráfica para el Sistema Simulador de Yacimientos de Hidrocarburos, Módulo: Sistema Generador de la malla de Simulación 3D (sin fracturas), el cual sigue un estilo cercano a la descripción de tareas de Ingeniería de Software. En el se describe de manera general la forma de abordar el problema de la construcción de la malla 3D de simulación a partir de la reconstrucción de las capas o superficies de propiedades. De igual manera se observó que la representación gráfica de la malla 3D es fundamental para el estudio de la simulación que sobre un yacimiento se desea llevar a cabo. Por ello se estudiaron formas de representar gráficamente la malla 3D de simulación para hacer eficiente su visualización tando en la construcción y definición de propiedades así como para observar el comportamiento de una simulación elaborada.

Durante los últimos meses iniciamos una serie de contactos con el Grupo de Interfaz y Visualización Gráfica para definir de manera adecuada cada una de las tareas propuestas para el módulo GRID del simulador. Iniciamos definiendo las tareas globales que el módulo debe satisfacer, y conforme éstas fueron discutiéndose justificando su uso dentro del entorno e incluso cambiamos la naturaleza de alguna de ellas. Esa fue la primera etapa. Durante la segunda, el grupo de Visualización se abocó a la tarea de diseñar un programa prototipo que pudiera definir mapas de contornos y editar éstos sobre una pantalla gráfica es decir: mover puntos, insertar puntos, eliminar contornos. La última fase de ese programa prototipo ha sido el añadir el módulo de Tratamiento de Contornos para probar la viabilidad de integrar poco a poco las tareas que hemos desarrollado a lo largo del presente periodo. Se espera continuar trabajando con el Grupo de Interfaz y Visualización Gráfica de manera cercana durante las próximas semanas para pulir algunas rutinas de intercambio de información y ayudar a definir una estructura eficiente de datos para manipular en memoria la malla 3D. En el Apéndice I se describen las tareas contempladas así como las tareas abordadas hasta este momento y aquellas que se están valorando. Es importante señalar que el Grupo de Interfaz y Visualización Gráfica ha mostrado interés en nuestras sugerencias para la Interfaz y la Visualización de la Malla 3D, un hecho relevante es que el prototipo que actualmente maneja para desplegar la simulación contempla las sugerencias en la visualización de la malla 3D.

2.6.2 Propuesta para el diseño de la Interfaz Gráfica

Se llevó a cabo un esfuerzo por nuestra parte para entender primero la naturaleza del problema, segundo identificar las diferentes tareas que contempla la construcción de la malla 3D y proponer una solución eficiente.

Se estudiaron algunos simuladores como lo son ECLIPSE y CMG, así como un sistema geoestadístico: ArgusOne, con estos sistemas se observó que la parte geoestadística se encuentra obsoleta en el simulador Eclipse y en CMG es pobre, CMG es eficiente para resolver problemas grandes y complejos de simulación, pero no para ofrecer una malla 3D satisfactoria.

Con la experiencia adquirida se planteó que el trabajo se realice a base de proyectos, el proyecto es la malla 3D de simulación. En el Apéndice D: Propuesta para el diseño de la Interfaz Gráfica para el Sistema Simulador de Yacimientos de Hidrocarburos: Módulo

Sistema Generador de la malla de Simulación 3D (sin fallas), se encuentra la descripción detallada de la propuesta.

En este documento se realiza una breve descripción de las tareas involucradas en un proyecto, las restricciones generales para el usuario del sistema; son descritos los requerimientos específicos para su diseño, siguiendo un estilo de Ingeniería de Software para llevar a cabo el Análisis General y describir los Requerimientos Funcionales de Módulo GRID así como de cada una de las tareas que se ha propuesto cuente este módulo.

2.6.3 Algunos aspectos sobre la Visualización de una malla 3D estructurada

La simulación de un yacimiento de hidrocarburos, contempla una serie de tareas elementales para poder describir de manera gráfica el comportamiento de los fluidos a su paso a través la roca. A continuación se enumeran una serie de tareas que debe contemplar la representación numérica y gráfica de un yacimiento.

- Realizar la visualización del modelo 3D de una malla regular que represente la simulación numérica de un yacimiento de hidrocarburos.
- Describir la malla mediante celdas, donde cada celda tiene asignadas algunas propiedades tales como: porosidad, permeabilidad, etc.
- Representar los cambios de valores en las propiedades mediante un degradado de colores. Básicamente se requiere de una interpolación sobre los valores de los nodos.
- Visualizar las celdas internas de la malla.
- Proveer la visualización de cualquiera de las 6 caras de la malla.
- Limitar la visualización sólo a las celdas activas en la simulación.
- Colocar pozos dentro de la malla, éstos pueden variar de simulación en simulación y se puede modificar su localización en tiempo real.

Las propiedades a observar sobre el yacimiento son medidas escalares asociadas a cada elemento de la celda 3D (hexaedro), o sobre la celda 2D de cada corte de la malla. Para poder usar esa información, es conveniente que tanto los nodos de la malla como sus valores asociados sean almacenados en la misma forma. En una primera instancia, tenemos

- m es el número de puntos (el número de nodos de la malla 3D).
- Se requieren tres arreglos de tamaño m, a saber, X,Y, y Z para almacenar las coordenadas de los puntos de la malla.
- Se requiere para cada propiedad: porosidad, permeabilidad, etc., un vector para almacenar la información de cada nodo.

Si consideramos que esos vectores, de posición y propiedades, son arreglos tridimensionales, tendremos una manera directa de obtener o asignar información al ordenar los nodos de la malla 3D de la misma forma en que se tiene un orden en los arreglos tridimensionales. De esta manera, podremos identificar plenamente los nodos vecinos de uno dado y calcular de manera eficiente alguna propiedad geométrica de la malla 3D.

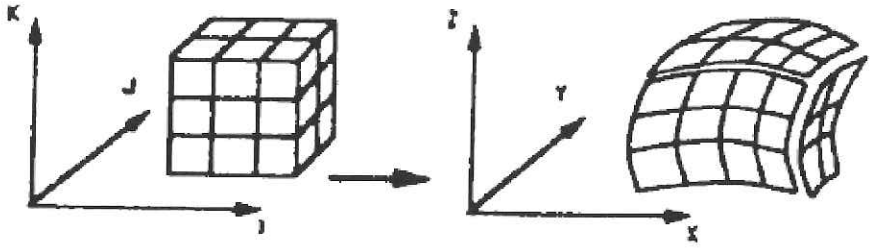


Figura 2.6.1: Malla de referencia y malla de simulación

La malla de referencia es fundamental para la visualización y la edición de las propiedades que se le asignarán para el yacimiento. Es por ello que incorporarla directa o indirectamente agilizará en mucho el pre-proceso y post-proceso de simulación. Un uso inmediato es la asignación manual de celdas activas o inactivas sobre la malla 3D de simulación al contar con esta forma de representar la malla de simulación por la malla 3D de referencia.

2.7 Herramientas de Visualización Gráfica

2.7.1 Objetivos

Las tareas programadas para este año del proyecto fueron ambiciosas. Abarcaron por una parte la reconstrucción de superficies, la que involucró visualizar la superficie reconstruida para calibrar los métodos y proponer que en caso de contar con mapas de contornos como datos dispersos para reconstruir superficies, las curvas poligonales de esos datos fuesen suavizados e interpolados a manera de contar con una mejor y mayor información de su comportamiento. Para cumplir con estas expectativas fue necesario programar y diseñar pequeños programas gráficos que nos permitieran visualizar nuestros resultados antes de incorporar las rutinas al simulador y de manera directa, comparar los resultados obtenidos por el simulador y nuestro desarrollo previo.

Estos programas fueron definidos conforme avance el proyecto y consideremos relevantes para el desarrollo de las sub-tareas contempladas.

2.7.2 Programa para visualizar Mallas 2D

Para observar la malla 2D generada por interpolación o por suavizamiento a través del funcional discreto de área ortogonalidad, se elaboró un programa para graficar mallas 2D de manera que nos permitiera observar aquella que será mapeada sobre cada capa de la superficie. Este programa se encuentra escrito en Fortan Visual 6.6 y en la Figura 2.7.1 se muestra una malla que fue usada a lo largo de la fase de este proyecto.

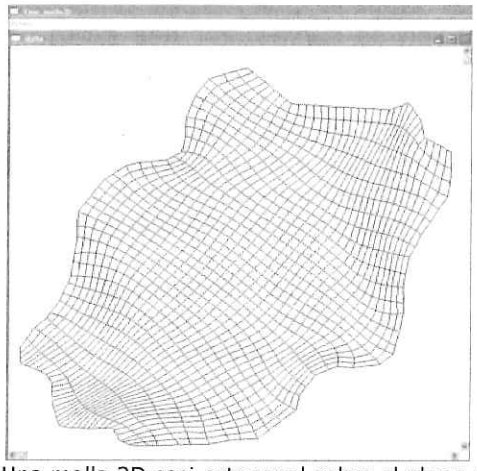


Figura 2.7.1: Una malla 2D casi-ortogonal sobre el plano de referencia.

2.7.3 Visualizador de curvas de nivel por Interpolación Bivariada

La malla 2D de estudio se construye sobre un plano de referencia paralelo a la cima del yacimiento. Para lograr una malla 3D que siga el flujo de los fluidos es necesario observar la forma de la cima, esto es de la superficie reconstruida, con la finalidad que la malla 3D conforme la geometría del yacimiento y siga el flujo esperado del fluido. Para lograrlo es necesario observar la forma de la superficie reconstruida. Por ello es necesario contar un la visualización de las curvas de nivel de la superficie.
Este programa permite a partir de una colección de datos y usando interpolación bivariada graficar una colección de curvas de nivel.
Este programa nos resultó indispensable para calibrar el sub-módulo del tratamiento de contornos así como para observar las superficies obtenidas bajo distintos escenarios de

datos dispersos. Este programa permite guardar la información en archivo lo que a su vez puede ser usado para un uso posterior, como el recuperar información de una malla 3D del simulador.

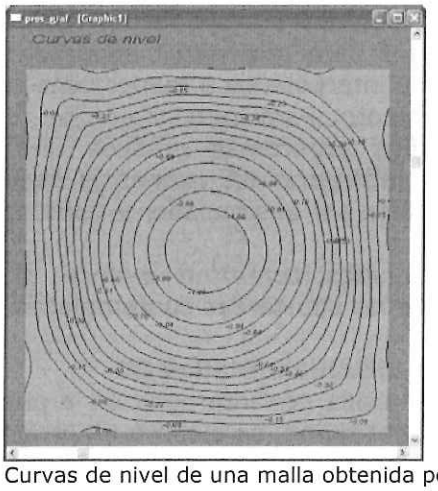


Figura 2.7.2: Curvas de nivel de una malla obtenida por bivar.f90.

2.7.4 Programa para visualizar el Tratamiento de Contornos

El tratamiento de contornos engloba dos tareas primordiales: el suavizamiento de polígonos y la reparametrización de curvas. Por una parte, el suavizamiento de curvas poligonales permite deshacernos de puntos superfluos sobre segmentos de curvas y suavizar los picos de los polígonos, con esto, obteniendo una curva C1. La reparametrización de esta curva a lo largo de la longitud de arco, permite distribuir puntos sobre la curva de manera uniforme o concentrando aquellos de acuerdo a una función monitor. Esto es, una vez reparametrizada la curva, podemos añadir puntos o colocar puntos a lo largo de una sección bajo un espaciamiento particular.

Elaboramos un programa para mostrar las bondades del Tratamiento de Contornos, así como las distintas formas de poder emplearlo. Este programa se presenta en la Figura 2.7.3 la cual representa una ventana del mismo. Con este programa el usuario puede seleccionar una parte de la curva a suavizar y/o controlar el número de puntos a distribuir a lo largo de esa sección. Cabe mencionar que la herramienta computacional para elegir la sección poligonal es rupestre pero nos permite evaluar el potencial para muchas aplicaciones donde podemos suavizar curvas poligonales.

Es importante señalar que este sub-módulo ha sido implantado ya en un prototipo de un programa diseñado por el grupo de Interfaz y Visualización Gráfica para observar la viabilidad de esta tarea.

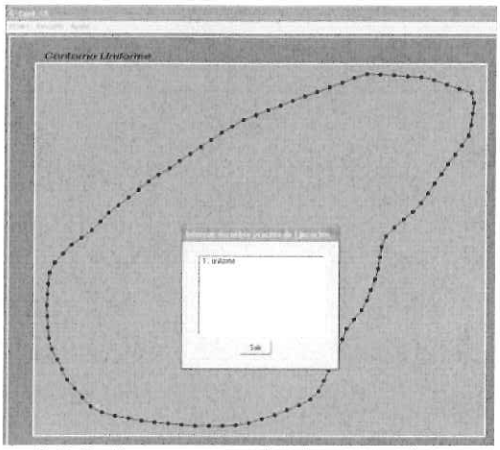


Figura 2.7.3: Programa en donde se suaviza una curva poligonal o parte del ella.

2.8 Sub-módulo para el Tratamiento de Contornos

2.8.1 Objetivos

Como se comentó en la Introducción, las más de las veces se provee información de la complejidad del yacimiento a través de mapas de contornos sobre la cima, la base o sobre una colección de capas del mismo. Esos mapas de contornos son curvas poligonales cerradas o abiertas que describen la forma del yacimiento al representar éstas curvas de nivel de la superficie a estudiar. Se tiene la experiencia de que si no se cuenta con los suficientes puntos o bien, los contornos presentan discontinuidades C1, esto puede llevarnos a obtener una superficie de mala calidad por cualquier método de interpolación o suavizamiento de datos. Es decir, se presentan rugosidades y ondulaciones en secciones donde no se cuente con suficiente información. Es por esto que es necesario proveer al módulo GRID un sub-módulo que permita suavizar e interpolar esas curvas poligonales con el fin de generar una mayor cantidad de puntos y poder reconstruir superficies aceptables.

Cabe señar que por suavizamiento del contorno entenderemos una técnica para eliminar puntos superfluos de la curva poligonal (que no aporten a la geometría de la curva poligonal), así como esquinas del mismo, lo que implica insertar puntos adecuadamente; para esto usaremos un spline cónico de suavizamiento. Por Interpolación, entenderemos una técnica que nos permita reparametrizar C1 la curva poligonal, reparametrizándolo con respecto a la longitud de arco y una técnica de acumulación de puntos en zonas claves para cada contorno. Esto nos permite tener un mejor control de la distribución de puntos a lo largo de la curva.

Esta técnica de tratamiento de contorno cuenta con el valor agregado al usarlo en la tarea de definición de la región de estudio donde se definirá la malla 2D sobre el plano de referencia. Será una pieza útil y reusable dentro del módulo GRID.

2.8.2 Suavizamiento de polígonos usando NURBS

Una propiedad que se desea observar en la curva S es que respete la forma de un polígono P, para lograrlo, se elige a S entre la familia de splines cónicos, con la característica de que uno o varios segmentos sean cónicas singulares. De esta forma, la curva S que suaviza a P se define como:

$$S(u) = s_i(u), \quad u \in [u_i, u_{i+1}], \quad i = 1, \ldots, N.$$
$$s_i(u) : [u_i, u_{i+1}] \rightarrow IR^2 \quad \text{es una curva racional cuadrática.}$$

Cada segmento del spline cónico se representa en su forma de Bézier racional cuadrática estándar.

En vista de que se requiere suavizar a P conservando su forma lo más posible, el objetivo del algoritmo es quitar sólo "los picos". Por cada terna de puntos consecutivos p_{j-1}, p_j, p_{j+1}, se aproxima el vértice p_j usando un arco cónico cuyos extremos se eligen en los lados $p_{j-1}p_j$ y $p_j p_{j+1}$, respectivamente, tratando que entre cada segmento cónico no singular se incluya uno singular; en la Figura 2.81, se muestran dos ejemplos de suavizamiento para contornos cerrados.

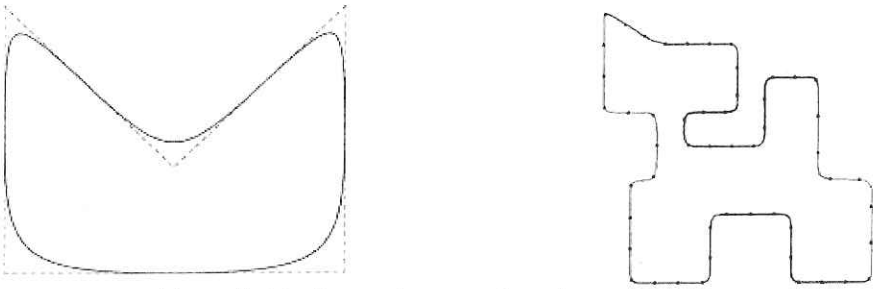


Figura 2.8.1: Dos contornos poligonales suavizados

2.8.3 Parametrización e Interpolación de curvas

En la fase de suavizamiento se genera un spline cónico $S(u):[0,1]\to\mathbb{R}^2$ de clase C^1 que aproxima un contorno poligonal. En la mayoría de los casos la parametrización del spline cónico obtenido no es adecuada para generar una repoblación del contorno, por tanto, resulta indispensable contar con un algoritmo que permita obtener numéricamente la reparametrización, ***S,*** usando la longitud de arco de *S*.

Sobre el programa `Tratamiento_contorno`

La reparametrización de contornos permite distribuir una colección determinada de puntos sobre el contorno a lo largo de la longitud de arco, con esto, se pueden concentrar puntos en zonas específicas donde se requieran, debido a la curvatura de la región poligonal. En esta fase del desarrollo del módulo, se puede distribuir de manera uniforme una colección de puntos a lo largo del contorno, o se puede señalar, manualmente, una sección del contorno donde se desea concentrar determinados puntos.

Se acompaña a las rutinas desarrolladas un programa, `Tratamiento_contorno`, para mostrar una manera de combinar las tareas del módulo a fin de obtener un contorno favorable. En la Figura 2.82, se ilustra esta idea.

Figura 2.8.2: Proceso para el tratamiento de contornos

Las rutinas principales son las siguientes:

- `suaviza_con`, rutina que construye un spline cónico con derivada continua, la cual aproxima el contorno poligonal. Se usa la representación de Bézier racional en su forma estándar.
- `long_arco_spl`, rutina que calcula la longitud de arco de un spline cónico dividiendo el intervalo de definición en n partes.
- `larcoSplcon`, rutina que calcula la longitud de arco del spline cónico sobre un intervalo.
- `SplineRacLin`, rutina que calcula un spline racional lineal que aproxima la función inversa de la longitud de arco del spline cónico.
- `ReparamSplcon`, Rutina que, dada una aproximación del spline racional lineal genera la siguiente.
- `Ratsplval`, Rutina que evalúa el spline racional lineal en los valores guardados en el arreglo U, el resultado se almacena en el arreglo T_DU.

En el Apéndice K.4 se describen instancias de las rutinas de este sub-módulo.

2.9 Sub-módulo para el Optimizador puntual

2.9.1 Objetivos

Cuando se construye la malla 2D, en algunas situaciones, es necesario suavizar una sección de la región de estudio particular, ya sea porque se han introducido algunas modificaciones previas o bien, se desea centrar el estudio en determinada sección. Con este fin, se desarrollará una rutina que optimice la malla de manera puntual.

Esta modalidad de suavizar la malla nos permite generar mallas planas y suaves de dimensiones muy grandes y a un bajo costo ya que únicamente es necesario algunas localidades de memoria para aplicar el procedimiento optimización Newton con Región de Confianza punto a punto.

En el período Abril-Mayo, se entregó un módulo para optimizar mallas usando el Método de Newton con Región de Confianza, los nodos interiores de la malla representan las incógnitas del problema (la malla inicial para el problema de optimización es la malla algebraica obtenida por Transfinite Interpolation). En esta ocasión se presenta una colección de rutinas modificadas que atacan el problema de optimización puntual. En esta entrega se consideró importante añadir la opción de que una colección de puntos representada por arreglos dimensionales puedan ser fijos dentro del proceso de optimización, esto, con la finalidad de atacar el problema de seguir fallas y fijar pozos en la malla 2D plana.

2.9.2 Suavizamiento de mallas de dimensiones grandes

En el período Abril-Mayo, se entregó un módulo para optimizar mallas usando el Método de Newton con Región de Confianza, los nodos interiores de la malla representan las incógnitas del problema (la malla inicial para el problema de optimización es la malla algebraica obtenida por Transfinite Interpolation). Ahora, se entrega un sub-módulo que lleva a cabo la optimización puntual de la malla. Este módulo cuenta con la posibilidad de indicar la colección de puntos que permanecerán fijos a lo largo de la optimización de la malla. Básicamente la idea es obtener para cada punto interior de la malla 2D una submalla de dimensión 3x3 y optimizarla. Esto se lleva a cabo sobre todos los puntos de la malla plana, excepto los fijos. Este procedimiento se repite sobre la malla resultante, hasta obtener un criterio de paro que puede ser el número de iteraciones permitidas o un criterio de convergencia.

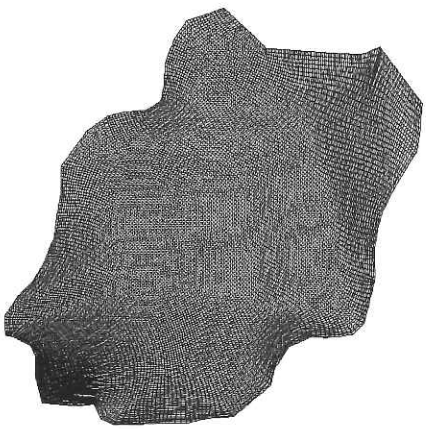


Figura 2.9.1 Una malla sobre una región plana de dimensión 99x99.

En la rutina guiatron_p, parte de la cual se muestra a continuación, viene implementado este procedimiento.

```
!*
!* El metodo de optimizacion usado es TRON: Metodo de Newton con Region de Confianza y la
!* optimización de la malla se hace punto a punto.
!*
!@see DSETSP,CALFACTO,DCOPY,FYG_P,PROJGR_T,ESCRES,PKHPK,
!*    DSPHESD,DTRON,NONIJ.
!*/
      SUBROUTINE guiatron_p(mn,ibo,il,jl,n,iww,idiwa,nnz,id3,iprint, &
       sigma,red,g,s,xc,adiag,bdiag,ldiag,xl,xu,a,ifcng, &
       arow_ind,acol_ind,arow_ptr,acol_ptr,epsmch, &
       listp,ngrp,idsp,y,eta,wa1,ga,w,noi,noj,iwa, &
       alfaprom,epsf,epsg,alfa_lim, &
       iter,nflag,alfapreal,red_ant,nfix,xfix,yfix,alfamin,inocon)
!*
!* * Declarations of DUMMY Arguments
!*
      IMPLICIT NONE
      INTEGER n,id3,mn,il,jl,ibo,iww,nnzex,nfix
      INTEGER iprint(id3),ifcng,k_cycle
      INTEGER noi((il-1)*(jl-1)),noj((il-1)*(jl-1))
        INTEGER xfix(nfix),yfix(nfix)
      DOUBLE PRECISION red(mn),g(n),ga(n),red_ant(mn)
      DOUBLE PRECISION sigma
!*
!* * Declarations for TRON
!*
       INTEGER nnz,idiwa
       INTEGER iwa(idiwa-(2*nnz+5*n+2*(il-1)*(jl-1)+2))
       DOUBLE PRECISION s(n),xc(n),adiag(n),bdiag(n),ldiag(n),xl(n),xu(n)
       DOUBLE PRECISION w(iww-12*n-nnz-mn),a(nnz)
       INTEGER info
!*
!* * Declarations of INTERNAL variables.
!*
      INTEGER mxfun,igra,ifun,iter,ipri,ient,nflag,isel,inocon,i,j
      DOUBLE PRECISION acc,f,gnorm,fant,fgor,epsg,epsf

!*
!* * Declarations of functions
!*
      DOUBLE PRECISION sqrt

!*
!*  * Declarations for TRON
!*
      INTEGER arow_ind(nnz), acol_ind(nnz)
      INTEGER arow_ptr(n+1), acol_ptr(n+1)
        INTEGER itermax
      INTEGER isave(8)
      DOUBLE PRECISION  dsave(10)
      DOUBLE PRECISION epsmch,fatol,fmin,cgtol,delta
      DOUBLE PRECISION  dnrm2
!*
!* * Declarations for the evaluation of the Hessian matrix.
!*
      INTEGER maxgrp,numgrp
      INTEGER listp(n), ngrp(n),idsp(n)
        DOUBLE PRECISION eta(n),y(n),wa1(n),php
```

```
!*
      LOGICAL finish,search,icheq
!*
!* * Declarations for the new functionals
!*
      DOUBLE PRECISION alfamin,alfamax,alfa_lim,alfaprom
        DOUBLE PRECISION factor1,factor2
      DOUBLE PRECISION alfapreal
!*
!* *  Declarations for making the point by point optimization
!*
     INTEGER k,funcion,ih,jh,ini,out,iloop
       DOUBLE PRECISION p(2),temp,tempr,tempra
     LOGICAL loop
!*
!* * Sets ACC to SQRT(macheps)
!*
      acc=SQRT(epsmch)
!*
!* * Initialitations of variables for the calling to TRON
!*  MXFUN - Maximum number of function  and gradient evaluations
!*          permitted
!*  IFUN - Counter for function evaluations
!*  IGRA - Counter for gradient evaluations
!*
!*
      mxfun = 100000
      igra = 0
      ifun = 0
      ient = 0
      ipri = 0
      iter = 0
      nflag = 0
      finish = .false.
        fant = 1.d+3
        isel = 3
!*
!* * Bounds
!*
       DO i = 1,n
          xl(i) = 0.d0
          xu(i) = 1.d0/SQRT(alfapreal)
       ENDDO
!*
!* * Calculate the sparsity pattern.
!*
          k = 0
          DO j = 1,n
             DO i = j,n
               k = k+1
               arow_ind(k) = i
               acol_ind(k) = j
             ENDDO
          ENDDO
      nnzex = n*(n+1)/2
      CALL dsetsp(n,nnzex,arow_ind,acol_ind,acol_ptr,arow_ptr,1, &
                  info,listp,ngrp,maxgrp,iwa)
         IF (info .le. 0) THEN
!*          WRITE (*,*) 'ERROR: INFO IN SUBROUTINE SETSP IS ',info
            stop
         ENDIF
!*
!* * The new factor for the normalization of the functionals are
!*  computed
!*
```

```
      CALL calfacto(il,jl,alfaprom,factor1,factor2)
!*
!* * Set parameters for the TRON method
!*
         itermax = 2
         fatol = 0.d0
         fmin = 0.d0
         cgtol = 0.1d0
!*
         loop = .true.
         iloop = 0
!*
      DO while (loop)
!*
!* * Save the array obtained in a whole 10 OUT cycles, to compare it with
!*   the one to be obtained in the following block of 10 OUT cycles
!*
          CALL dcopy(mn,red,1,red_ant,1)
!*
!* * Begin a block of 10 cycles of point by point optimization
!*
          DO out = 1,20  !*75

       DO i = 2,il-1
           DO j = 2, jl-1
!*
!* * Check if the point is in the list of fix points
!*
          CALL cheqlist(nfix,xfix,yfix,i,j,icheq)
       IF(.not.icheq)THEN
!*
!* * The point can be moved
!*
          igra = 0
          ifun = 0

!*
!* * Bring the coordinate values of points P(i,j)
!*
```

2.9.3 Uso del sub-módulo para modelar Fallas y pozos

Un problema que se desea resolver en el simulador numérico, es fijar puntos que representen a pozos o puntos de control determinados o puntos de control que representen a una falla, y construir una malla donde esos puntos son restricciones del problema. Para esto, se ha diseñado un módulo que permita:

- Optimizar de manera puntual toda la malla.
- Dejar fijos en el proceso de optimización a una colección de puntos (restricciones).

Para poder implementar esta idea en el simulador, será necesario trabajar con el equipo de Visualización e Interfaz Gráfica a fin de diseñar un mecanismo para que el sistema pueda transferir a este módulo la información de los puntos a fijar en la malla 2D.

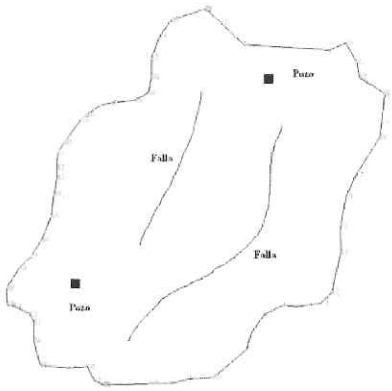


Figura 2.9.2: Una región plana de estudio que representa la vista areal de una sección del yacimiento a estudiar, la cual cuenta con fallas y pozos.

A continuación se muestran algunas mallas suaves y convexas cuyas restricciones son puntos fijos que siguen una forma particular, esta se comenta al pie de cada figura.

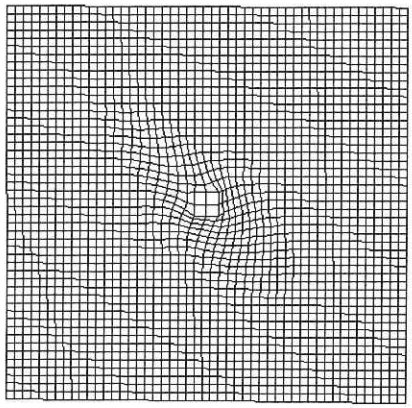


Figura 2.9.3a: Malla con 9 puntos fijos (cuatro cuadrados). Malla de 40x40

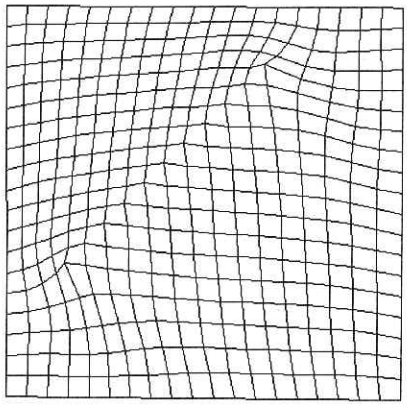


Figura 2.9.3b: Malla con puntos fijos sobre una recta. Malla de 20x20

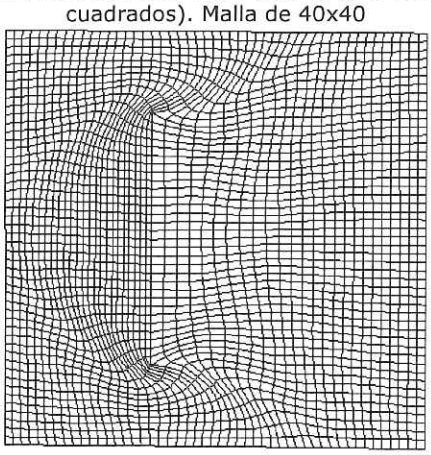


Figura 2.9.3c: Malla con puntos fijos sobre una línea vertical que es parte de una línea curvilínea de la malla.
Malla de 40x040

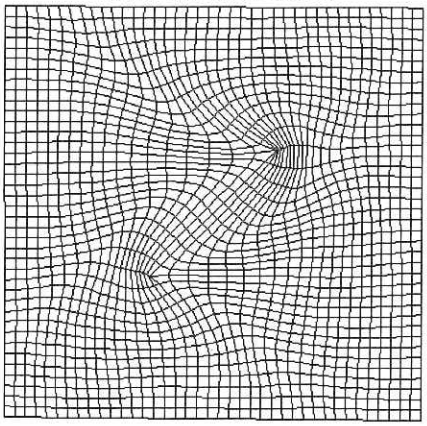


Figura 2.9.3d. Malla con puntos fijos en una línea inclinada que es parte de una línea curvilínea de la malla.
Malla 40x40

Esta subtarea es muy importante en la modelación de yacimientos ya que usada de manera adecuada, podemos construir mallas 3D donde se indique dónde estarán los pozos, ya sea desde una vista areal del yacimiento, así como indicar sobre qué bloques de celdas pasará el pozo horizontal. Si logramos construir las herramientas de cómputo necesarias podemos resolver este importante problema y así la malla 3D de simulación será representativa del tipo de yacimiento y de las condiciones de perforación y explotación de pozos.

En el Apéndice K.5 son descritas las instancias de las rutinas empleadas en este sub-módulo.

2.10 Generador Experimental de mallas 3D

2.10.1 Objetivos

Una vez reconstruida la cima del yacimiento, y habiendo construido una malla plana adecuada sobre el plano de referencia, proyectamos esta sobre la superficie de la cima obteniendo una malla sobre esa superficie y la bajamos verticalmente a un grosor constante con esto obtendremos una primera malla 3D sobre la sección de estudio de nuestro yacimiento.

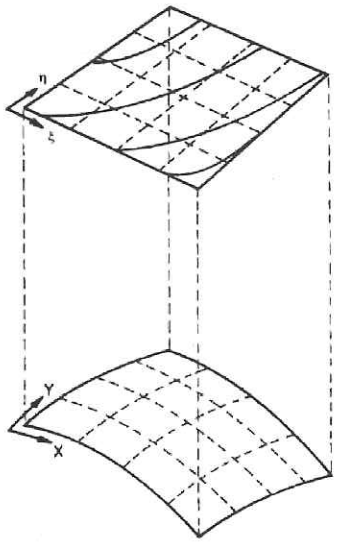


Figura 2.10.1: Procedimiento para obtener una malla 2D sobre la superficie. Tomado de Hirasaki y O'Dell, 1970.

Recordemos que la forma en que contamos con información de la cima es a través de datos dispersos. Este procedimiento que se propone es simple. Reconstruir la superficie a partir de los datos dispersos a través de alguno de los métodos atacados. Luego observar las curvas de nivel de esa superficie sobre un plano de referencia, sobre este plano definir una malla cuasi-ortogonal y proyectar esta sobre la superficie de la cima. La malla 3D de simulación se formará al bajar copias de esta a partir de la cima. Estas copias representan los slides de la malla 3D, slides cuya separación es el grosor entre cada una de las copias.

En la siguiente figura se muestra una descripción gráfica del proceso involucrado.

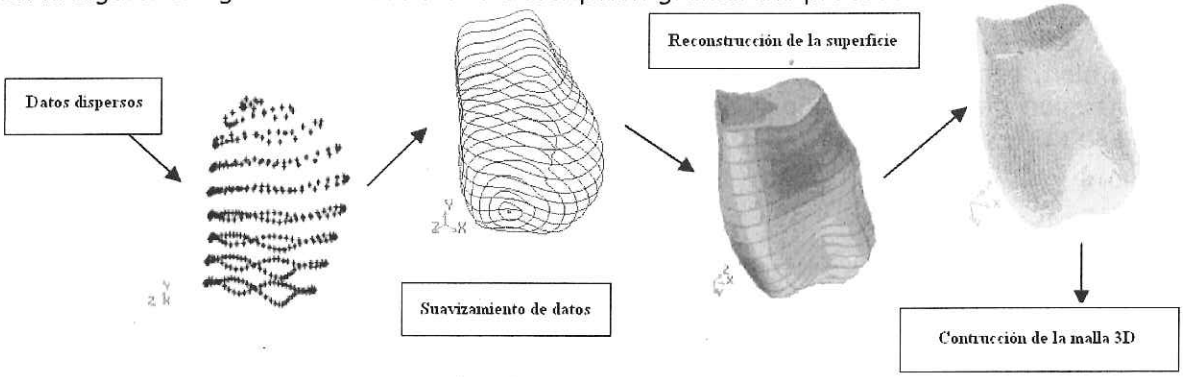


Figura 2.10.2: Ilustración gráfica de la forma de construir una malla 3D.

2.10.2 Mallas 3D con la Geometría Punto de Esquina (Corner-Point)

Esta forma de construir la malla 3D de simulación es lo que se conoce como una malla que sigue la geometría punto esquina, esto es que los vértices del bloque de malla se encuentran sobre una superficie tanto en la cara de arriba como en la de abajo siguiendo con esto la geometría del yacimiento.

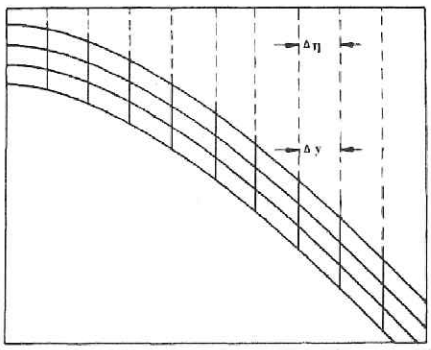


Figura 2.10.3: Forma usual de construir una malla 3D a partir de la cima de la superficie mediante desplazamientos verticales.

El módulo entregado, construye la superfice a partir de una colección de datos dispersos usando el método de Suavizamiento por B-spline Multinivel. En la Figura 2.10.4 se pueden observar las curvas de nivel obtenidas de la superficie reconstruida sobre la cima de estudio.

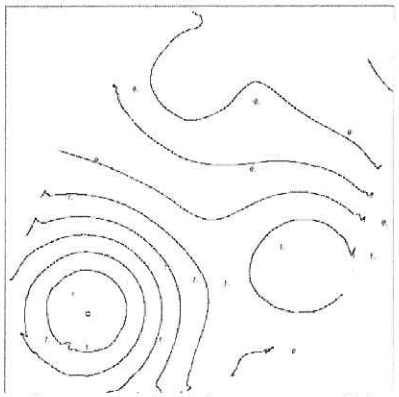


Figura 2.10.4: Curvas de nivel de una superficie reconstruida.

Ahora bien, sobre un plano de referencia se definieron los cuatro segmentos de control de la region plana de estudio y se generó una malla 2D suave y casi-ortogonal. Esta malla se observa en la Figura 2.10.5.

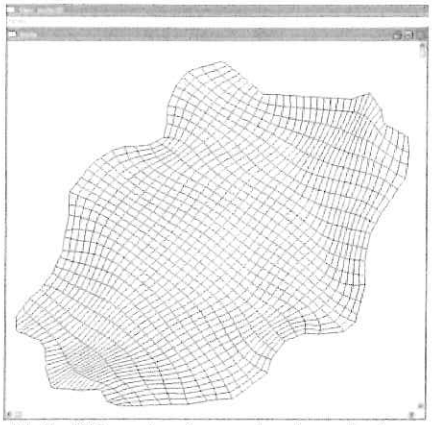


Figura 2.10.5: Malla 2D casi-ortogonal sobre el plano de referencia.

Esta malla 2D sobre el plano de referencia se proyectó sobre la superficie de la cima y copias de esta malla fueron tiradas verticalmente con un grosor variable. El resultado se observa en la Figura 2.10.6.

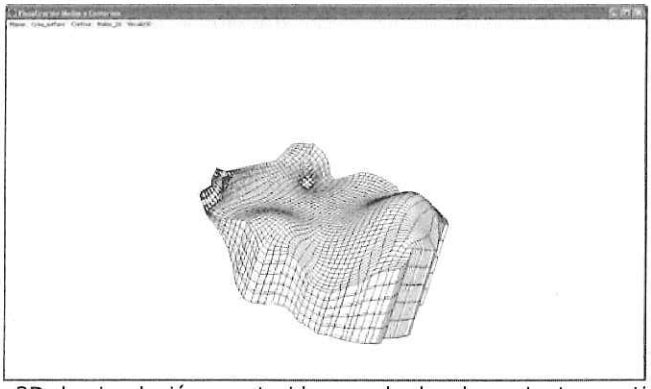


Figura 2.10.6: Malla 3D de simulación construida usando desplazamientos verticales no uniformes.

Esta malla 3D así obtenida puede ser usada para la simulación numérica del yacimiento.

La forma en que hemos descrito cómo construir la malla 3D a partir de la cima, no es única. Como se señaló en la Introducción, podemos considerar que la dirección z en la que se desplaza la malla sea ortogonal a la superficie construyendo con esto una malla 3D cuyas caras "verticales" sean ortogonales a la superficie. Este tipo de mallas son valiosas porque siguen la geometría del yacimiento y muestran ortgonalidad con las caras horizontales, con esto la discretización de las ecuaciones fluido-flujo siguen la propiedad de ser bloques ortogonales.

De igual manera será necesario tomar decisiones al momento de construir la malla ya que dependiendo del método para reconstruir la superficie o de la información que se cuente tanto de la capa, como de la base, puede ocurrir que la malla 3D se cruce en la dirección de z y entonces promediar o bien asignar inactiva esa celda problemática. Este es solo un ejemplo por mencionar del escenario posible.

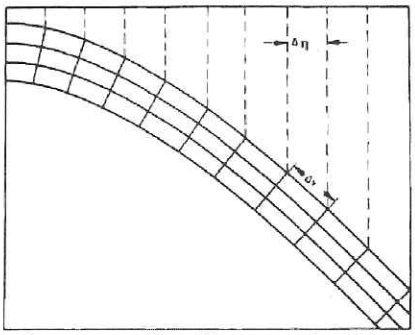


Figura 2.10.7: Otra forma construir una malla 3D a partir de la cima de la superficie mediante desplazamientos normales a la superficie.

Apéndice A: Generación Numérica de Mallas Planas usando Interpolación Transfinita y el funcional Discreto de Área-Ortogonalidad

A.1. Planteamiento del problema

La generación de mallas es un proceso que consiste en partir un dominio físico en subdominios elementales para observar algún fenómeno medible sobre cada subdominio. Una malla estructurada es aquélla bajo la cual podemos identificar de forma rápida los nodos de la malla, interiores y exteriores y cada nodo interior cuenta con el mismo número de elementos adyacentes. Este tipo de mallas son muy socorridas en aplicaciones de EDP, pero presentan dificultades en la aproximación sobre la frontera cuando el dominio no es simple. Existen muchos métodos para obtener este tipo de mallas estructuradas y uno de ellos es el método del mapeo que consiste en obtener un mapeo que transforme el dominio físico en uno más sencillo y entonces lograr de forma directa una malla, ver Figura (A.1)

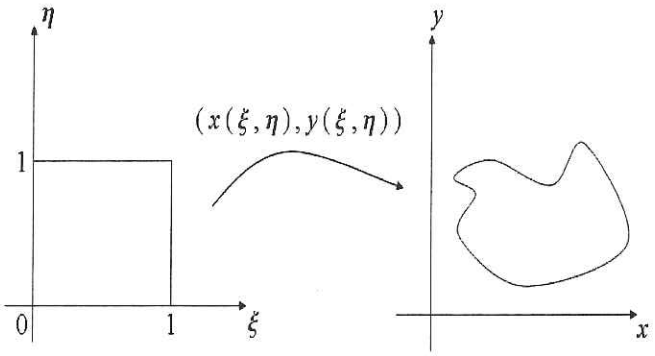


Figura A.1: Mapeo entre regiones

A.2 Método que se usa

Contando con un mapeo continuo entre fronteras nos lleva a considerar la frontera de la región en cuatro segmentos, segmentos curvilíneos o subfronteras. Así, la idea gráfica de la generación de una malla es unir las fronteras 1-2 con 4-3 y 1-4 con 2-3 (ver Figura (A.2)), por medio de segmentos curvilíneos de tal forma que no se intersecten. La idea es "tirar" líneas entre segmentos opuestos. Como en la Figura (A.3).

En muchas aplicaciones la región de trabajo es solución de un problema diferencial o algebraico; en otros es posible contar con la descripción de la región en forma continua a través de una parametrización del contorno. En nuestro caso, disponemos de puntos sobre el contorno obtenidos de alguna observación. A continuación, hablaremos del método que emplea `Driv_Gen_mall` para generar una malla inicial, conocido como de Interpolación Transfinita (TFI).

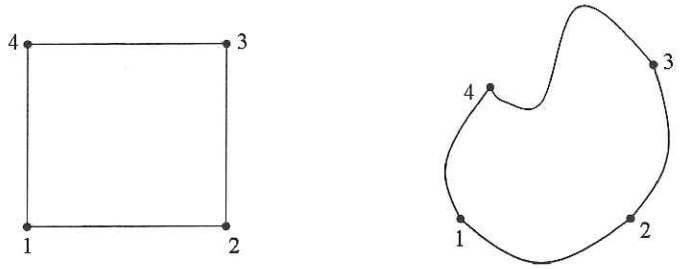


Figura A.2: Fronteras opuestas de una región

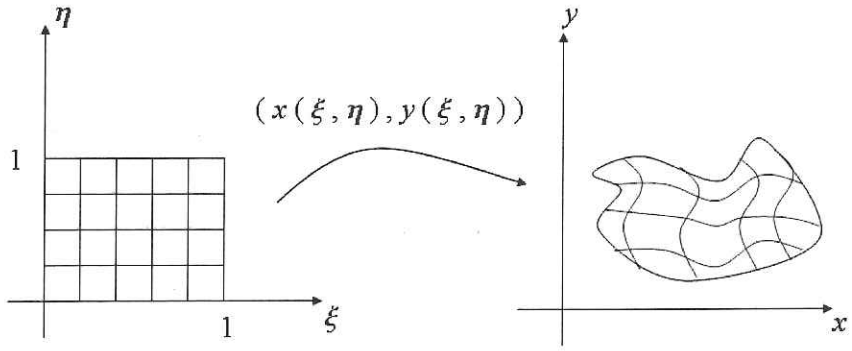


Figura A.3: Una malla a partir de un mapeo entre regiones

A.2.1 Interpolación Transfinita (TFI)

Los métodos más sencillos para generar una malla inicial se basan en que el mapeo $(x(\xi,\eta),y(\xi,\eta))$ está representado por las funciones de frontera y funciones de interpolación. La forma más sencilla de hacerlo es por interpolación transfinita **TFI**, así describiremos este método a continuación.

Para describir el método haremos las siguientes consideraciones. Para trabajar en el caso de dos dimensiones tomaremos como espacio lógico a el cuadrado unitario $B_2=[0,1]\times[0,1]$ de coordenadas ξ y η, y como región física a la región Ω con coordenadas x y y.

Supongamos que contamos con una descripción de la frontera de Ω a través de los cuatro segmentos de frontera, véase la Figura (A.4), todas ellas parametrizadas en la forma

$$\begin{aligned} &(x_b(\xi),y_b(\xi)), && (x_t(\xi),y_t(\xi)), && 0\le\xi\le 1\\ &(x_l(\eta),y_l(\eta)), && (x_r(\eta),y_r(\eta)), && 0\le\eta\le 1 \end{aligned}$$

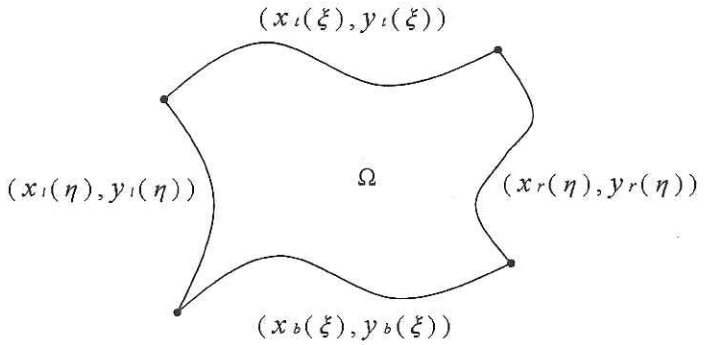


Figura A.4: Segmentos de frontera de $\partial\Omega$

con la propiedad de continuidad del mapeo; es decir, que las parametrizaciones se empaten

$$\begin{aligned} &(x_b(0), y_b(0)) = (x_l(0), y_l(0)) \quad (x_b(1), y_b(1)) = (x_r(0), y_r(0)) \\ &(x_r(1), y_r(1)) = (x_t(1), y_t(1)) \quad (x_t(0), y_t(0)) = (x_l(1), y_l(1)) \end{aligned} \tag{1}$$

El mapeo que proviene de la interpolación transfinita TFI tiene coordenadas

$$\begin{aligned} x(\xi,\eta) &= (1-\eta)x_b(\xi) + \eta x_t(\xi) + (1-\xi)x_l(\eta) + \xi x_r(\eta) \\ &\quad - \{\xi\eta x_t(1) + \xi(1-\eta)x_b(1) + \eta(1-\xi)x_t(0) + (1-\xi)(1-\eta)x_b(0)\} \\ y(\xi,\eta) &= (1-\eta)y_b(\xi) + \eta y_t(\xi) + (1-\xi)y_l(\eta) + \xi y_r(\eta) \\ &\quad - \{\xi\eta y_t(1) + \xi(1-\eta)y_b(1) + \eta(1-\xi)y_t(0) + (1-\xi)(1-\eta)y_b(0)\} \end{aligned} \tag{2}$$

En la práctica, cuando contamos con una región plana $\Omega \subset \mathbb{R}^2$, por lo general únicamente contamos con una discretización de su frontera $\partial\Omega$ a través de una colección de puntos (Figura (A.5)), y sobre esa región cerrada hemos de construir el mapeo TFI.

El paso siguiente es parametrizar los cuatro segmentos de frontera $(x_b(\xi), y_b(\xi))$, $(x_l(\eta), y_l(\eta))$, $(x_t(\xi), y_t(\xi))$ y $(x_r(\eta), y_r(\eta))$.

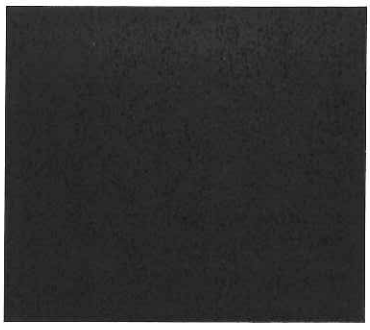


Figura A.5: Puntos que describen la región poligonal

La forma usual de obtener una parametrización es construyendo una interpolación lineal paramétrica. Contando con la parametrización de los segmentos de frontera se procede a obtener una colección de puntos tratando que sean igualmente espaciados pero respetando la forma del contorno.

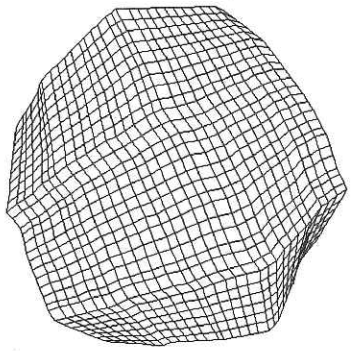


Figura A.6: Malla construida con TFI de cima3040

El mapeo es sencillo ya que esta descrito a partir de las fronteras, sin embargo, es un mapeo que propaga las discontinuidades diferenciales de la frontera hacia el interior de la región, por consiguiente al usarlo sobre regiones irregulares el resultado es una malla muy "doblada". Sin embargo, el mapeo TFI nos es de gran utilidad para generar una malla inicial que puede ser optimizada, ver Figura (A.6).

A.2.2. Funcional de Área Ortogonalidad

Nuestro interés en este sistema es principalmente el de obtener mallas con la propiedad de Área y de Ortogonalidad, es por ello que daremos una breve explicación de la combinación de estas, a través del funcional discreto de Área-Ortogonalidad.

En las siguientes definiciones, se precisará lo que entendemos por una malla estructurada y por un funcional discreto.

Consideremos una región Ω del plano definida por una poligonal γ de vértices $V=\{v_1,v_2,...v_q\}$, cerrada, simple y orientada en sentido positivo.

Definición 1 *Sean m y n números naturales mayores que 2. Decimos que el conjunto de puntos del plano*

$$G=\{P_{i,j}|i=1,...,m;j=1,...,n\}$$

con lados

$$L_1(G)=\{P_{i,1}|i=1,...,m\}$$
$$L_2(G)=\{P_{m,j}|j=1,...,n\}$$
$$L_3(G)=\{P_{i,n}|i=1,...,m\}$$
$$L_4(G)=\{P_{1,j}|j=1,...,n\}$$

es una malla estructurada admisible y discreta de orden $m\times n$ para Ω, si se satisface que

$$V \subseteq \bigcap_{i=1}^{4} L_i(G).$$

Decimos además que la malla G es convexa si cada uno de los $(m-1)(n-1)$ *cuadriláteros (o celdas)* $c_{i,j}$ *de vértices* $P_{i,j}$, $P_{i+1,j}$, $P_{i+1,j+1}$, $P_{i,j+1}$, *con* $1 \le i < m$ *y* $1 \le j < n$, *no es convexo.*

Definición 2 *Un funcional discreto I sobre una malla* $G = \{P_{i,j}\}$ *es una función*

$$I(G) = \sum_{i,j} f(c_{i,j})$$

donde $c_{i,j}$ *es la celda i,j de la malla y*

$$f(c_{i,j}) = f(P_{i,j}, P_{i+1,j}, P_{i+1,j+1}, P_{i,j+1})$$

es una función de sus vértices.

Para escribir la forma que tienen los funcionales discretos, denotemos los triángulos en la celda i, j con vértices $P_{i,j}, Q_{i,j}, R_{i,j}, S_{i,j}$ como

$$\Delta S_{i,j}, P_{i,j}, Q_{i,j} = \Delta^1_{i,j}$$
$$\Delta Q_{i,j}, R_{i,j}, S_{i,j} = \Delta^2_{i,j}$$
$$\Delta P_{i,j}, Q_{i,j}, R_{i,j} = \Delta^3_{i,j}$$
$$\Delta R_{i,j}, S_{i,j}, P_{i,j} = \Delta^4_{i,j}$$

de tal manera que el funcional discreto sobre una malla G sea

$$I(G) = \sum_{i=1}^{m-1} \sum_{j=1}^{n-1} \sum_{k=1}^{4} f(\Delta^k_{i,j})$$

Así, el funcional de suavidad F_S, el funcional de longitud F_L, el funcional de área F_A y el funcional de ortogonalidad F_O están dados como

$$F_S = \sum_{i=1}^{m-1} \sum_{j=1}^{n-1} \sum_{k=1}^{4} \frac{\lambda(\Delta^k_{i,j})}{\alpha(\Delta^k_{i,j})}$$
$$F_L = \sum_{i=1}^{m-1} \sum_{j=1}^{n-1} \sum_{k=1}^{4} \lambda(\Delta^k_{i,j})$$
$$F_S = \sum_{i=1}^{m-1} \sum_{j=1}^{n-1} \sum_{k=1}^{4} (\alpha(\Delta^k_{i,j}))^2$$
$$F_S = \sum_{i=1}^{m-1} \sum_{j=1}^{n-1} \sum_{k=1}^{4} (o(\Delta^k_{i,j}))^2$$

Donde, α, λ y o son los funcionales discretos de área, longitud y ortogonalidad en los triángulos. Véase [5].

Para combinar los efectos de varios funcionales en una malla, es natural considerar combinaciones lineales entre ellos,

$$F(x)=\sigma_1 F_1(x)+\sigma_2 F_2(x)+\sigma_3 F_3(x)$$

donde $\sigma_1,\sigma_2,\sigma_3 \geq 0$ y $\sigma_1+\sigma_2+\sigma_3=1$.

La elección de los ``pesos'' σ_i y los funcionales f_i está motivada por las propiedades que se desea aparezcan en la malla.

Una de las combinaciones más útiles es el funcional de área-ortogonalidad, para el cual $\sigma_1=\sigma_2=\frac{1}{2}$ y $F_1(x)=F_A(x)$, $F_2(x)=F_O(x)$, donde F_A y F_O representan a los funcionales de área y ortogonalidad; esta combinación es

$$F_{AO}=\frac{F_A+F_O}{2}$$

El funcional de Área-Ortogonalidad produce mallas muy suaves y con pocas celdas no convexas, pero no mallas convexas en general.

Para finalizar, hemos de enfatizar que *no necesariamente* debemos obtener el óptimo del problema, lo que se requiere es generar una malla *convexa* que satisfaga la propiedad geométrica de ortogonalidad.

A.3. Tareas principales del programa

El sistema `Driv_Gen_mall` es un procedimiento que genera mallas con la propiedad de Área Ortogonalidad sobre una región Ω plana irregular dada a partir de su frontera, (ver Figura (A.7)). Dependiendo de los datos de entrada, los resultados obtenidos pueden ser los siguientes:

1, **Una red inicial obtenida por TFI**. Dado un conjunto de puntos $P_i=(x_i,y_i)$, vértices de la región poligonal Ω, y el tamaño de la red deseada $il \times jl$, donde il es el número de puntos horizontales y jl el número de puntos verticales de la malla, se puede obtener una malla inicial con TFI.

2. **Una red óptima generada por Área Ortogonalidad**. Dada una malla inicial de una región Ω, la malla solución se obtiene optimizando la malla inicial mediante el método de Área Ortogonalidad.

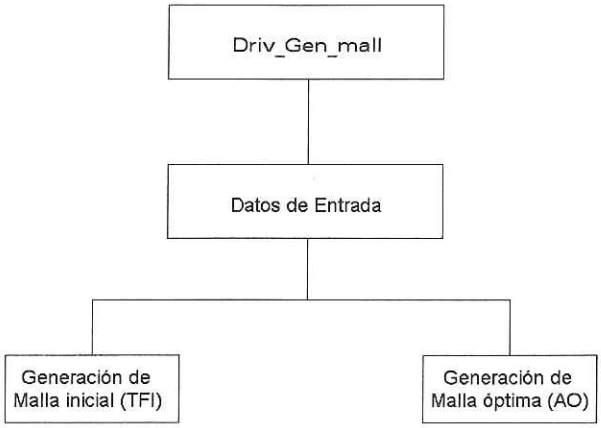


Figura A.7:Diagrama de opciones de `Driv_Gen_mall`

A.4. Descripción breve de los módulos

`Driv_Gen_mall`. Programa principal que presenta los menús y submenús del sistema. Los módulos principales de este programa son dos, ver Figura (A.8).

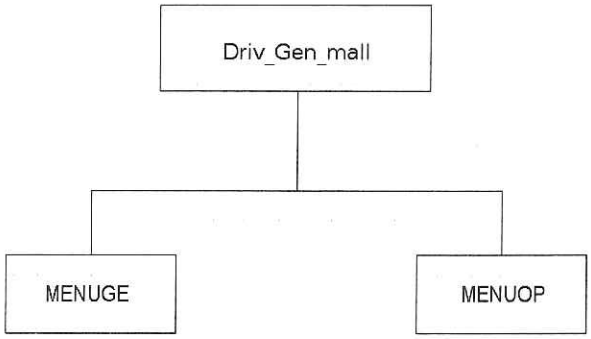


Figura A.8: Diagrama de estructura de `Driv_Gen_mall`

1. Módulo `MENUGENE`. Este módulo coordina la generación automática de una malla inicial, para lo cual se encarga de leer los datos de entrada proporcionados por el usuario, a través del módulo `MENUGE`.

Módulo `MENUGE`. Este módulo genera una malla inicial sobre la región llamando a la rutina `SEGRED`, ver Figura (A.9).

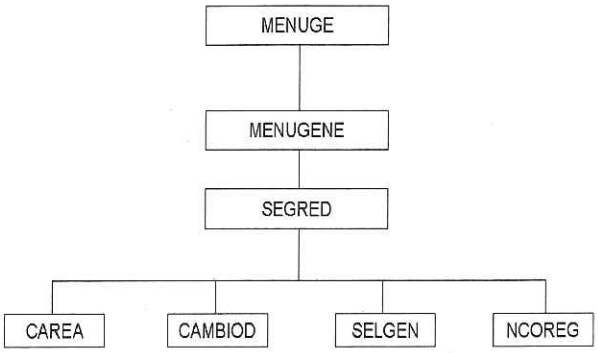


Figura A.9: Diagrama de estructura de MENUGE

Módulo SEGRED. Este módulo calcula el área de la región haciendo un llamado a las rutinas CAREA, CAMBIOD, SELGEN, NCOREG.

Módulo CAREA. Este módulo calcula el área de un triángulo dadas las coordenadas de los puntos A, B y C que lo forman.

Módulo CAMBIOD. Esta rutina cambia la orientación de la frontera cambiando la frontera inferior por la frontera superior.

Módulo SELGEN. Este módulo coordina la generación de una malla sobre una región simplemente conexa, llamando a las rutinas PARLIN, CONVEX, MODULO, RED_INI. Ver Figura (A.10).

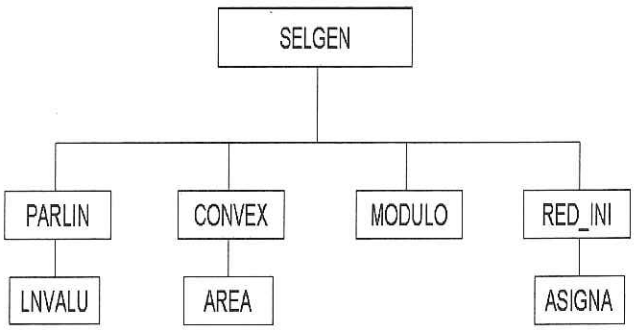


Figura A.10: Diagrama de estructura de SELGEN

Módulo PARLIN. Este módulo coordina la ejecución de la interpolación. Llama al módulo LNVALU para obtener los puntos que formarán la frontera de la malla.

Módulo LNVALU. Llama al módulo INTERV para identificar los intervalos a los cuales pertenecen los puntos donde se va a evaluar la función interpolante y obtiene x y y de los puntos que formarán la frontera de la malla.

Módulo INTERV. Esta rutina recibe de LNVALU los puntos donde se quiere evaluar la función interpolante e identifica el intervalo al cual pertenece, regresando este dato para la evaluación.

Módulo CONVEX. Llama al módulo AREA para calcular el área de los cuatro triángulos de las esquinas de la malla y verificar si ésta es admisible.

Módulo MODULO. Esta subrutina proporciona el orden de los puntos obtenidos por interpolación.

Módulo RED_INI. Este módulo genera la red inicial mediante el método algebraico TFI (Interpolación Transfinita), llamando al módulo ASIGNA.

Módulo ASIGNA. Esta rutina hace una asignación de los puntos en el interior y en la frontera de la malla de manera que ésta sea admisible.

Módulo NCOREG. Este módulo coordina la generación de una malla sobre una región no conexa, llamando a las rutinas PARLIN, CONVEX, MODULO, RED_INI. Todas ellas explicadas en los módulos anteriores.

2. Módulo MENUOP. Este módulo coordina la generación de una malla óptima donde el usuario elige el método de optimización para obtener la red solución, y para ello llama a las rutinas FILEIO, ADMRED, SCALE, CAREA, NONIJ y SOLVER. Ver Figura (A.11).

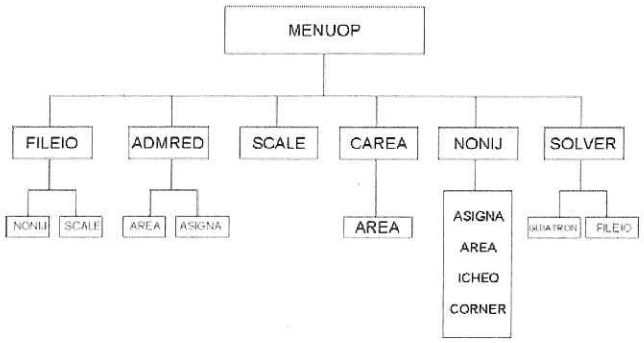


Figura A.11: Diagrama de estructura de MENUOP

Módulo FILEIO. Este módulo hace la asignación del archivo de entrada o salida para leer o escribir los puntos de la malla. Llama a las rutinas NONIJ y SCALE.

Módulo NONIJ. Este módulo verifica la convexidad de una malla a través de ASIGNA, AREA y ICHEQ. Note que los módulos ASIGNA y AREA ya han sido mencionados antes.

Módulo ICHEQ. Esta subrutina verifica la convexidad de las celdas que no están en las esquinas de la región.

Módulo ADMRED. Esta subrutina calcula el área de las cuatro celdas que están en las esquinas de la región para checar su admisibilidad.

Módulo SCALE. Este módulo reestablece los valores de la malla.

Módulo SOLVER. Este módulo resuelve un problema de minimización asignado al problema de generación de mallas variacional. Llama a las rutinas GUIATRON, FILEIO.

Módulo GUIATRON. Este módulo coordina la ejecución del método de optimización Newton Truncado.

A.5. Entrada y salida de datos

La especificación de los datos que emplea el sistema ayuda a conocer como intercambiar la información hacia y desde el sistema con nuestra aplicación. Describamos el formato de datos que maneja el sistema.

Los datos que se manejan son: contorno y malla. Cada uno tiene un formato diferente en formato ASCII.

A.5.1. Formato del archivo de datos de los contornos

Los archivos de un contorno cuentan con un formato muy simple: el número de puntos, una identificación de los segmentos de frontera con que cuenta y las coordenadas del contorno. Enfatizamos que los contornos son cerrados por lo que la descripción debe considerar que el punto de inicio y el punto final sean iguales. Otro punto importante es que el sistema acepta regiones con agujeros siempre que se señale.

El archivo de datos tiene el formato:

```
m ibflag nb1 nb2 nb3 nb4
x_1 y_1
x_2 y_2
x_3 y_3
. .
. .
. .
x_m y_m
```

bajo este formato `x_m=x_1` y `y_m=y_1`. Se explica cada una de las entradas.

`M`	Es el número total de puntos a la frontera. El primero y el último deben repetirse para considerar el contorno cerrado.
`Ibflag`	Indicador del tipo de contorno. =1. Los segmentos de frontera irán definidos. =13. Los segmentos de frontera 1 y 3 son iguales. En una región con agujeros es necesario hacer un corte y entonces dos segmentos de frontera son iguales. =24. Los segmentos de frontera 2 y 4 son iguales. Similar al caso anterior. =0. No hay segmentos de frontera determinados para el mapeo. El sistema considerará una elección automática.
`nb1`	Número de puntos que describen el segmento de frontera 1.
`nb2`	Número de puntos que describen el segmento de frontera 2.
`nb3`	Número de puntos que describen el segmento de frontera 3.
`nb4`	Número de puntos que describen el segmento de frontera 4.
x_i y_i	Coordenadas *x* y *y* del contorno.

Un ejemplo de archivo donde la frontera 1 y la frontera 3 son iguales es el siguiente:

```
13    13    2     7     2     5
 0.0    0.0
 9.0    0.0
12.0    4.0
16.0    4.0
19.0    1.0
15.0   -3.0
11.0   -3.0
 9.0    0.0
 0.0    0.0
20.0  -15.0
```

```
30.0    1.0
12.0   15.0
 0.0    0.0
```

El contorno que representa este archivo de datos se observa en la Figura (A.12).

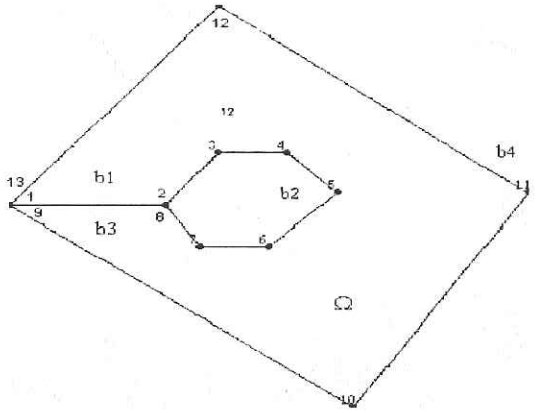


Figura A.12: Contorno de una región con un agujero

A.5.2. Formato de archivo de mallas

El sistema maneja dos tipos de formato para los archivos de mallas, formato XY y formato RED. El primero ha sido empleado en sistemas anteriores y su formato sigue la estructura de una matriz. El segundo es un formato de reciente diseño donde los nodos interiores son fácilmente localizables y ha resultado ser éste último muy práctico en la optimización de los funcionales discretos. Pasemos a describir cada uno de ellos.

Tomando en cuenta los nodos interiores y exteriores de la malla como un gran matriz, numeramos los nodos y su posición por filas y columnas. Fijando los segmentos de frontera "abajo", "derecha", "arriba" e "izquierda", nuestra matriz de nodos partiría de abajo hacia arriba y de izquierda a derecha, ver Figura (A.13).

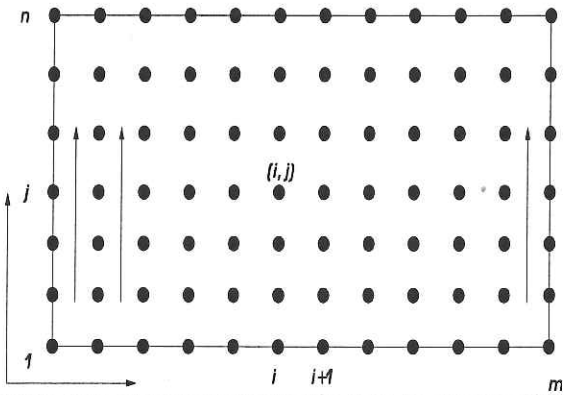


Figura A.13: Matriz de nodos y su orientación para la numeración de los mismos

Siguiendo esta idea, un nodo *r* tendría coordenadas *i* y *j*; es decir, en la columna *i* y fila *j* de nodos. Una celda $c_{i,j}$ tendrá por vértices a los nodos (i,j), $(i+1,j)$, $(i+1,j+1)$,

$(i, j+1)$. Bajo estas suposiciones, veamos cada uno de los formatos dispuestos para la malla.

A.5.2.1. Formato XY

El formato **XY** consiste en imprimir en archivo la matriz de nodos por columnas, iniciando desde la izquierda y hasta la última columna imprimiendo 7 valores por línea de archivo bajo el siguiente esquema

```
m     n
filename
x_11  y_11  x_12  y_12  x_13  y_13  x_14
y_14  x_15  y_15  x_16  y_16  x_17  y_17
.     .     .
.     .     .
.     .     .
x_21  y_21  x_22  y_22  x_23  y_23  x_24
y_24  x_25  y_25  x_26  y_26  x_27  y_27
.     .     .
.     .     .
.     .     .
x_m1  y_m1  x_m2  y_m2  x_m3  y_m3  x_m4
y_m4  x_m5  y_m5  x_m6  y_m6  x_m7  y_m7
.     .     .
.     .     .
.     .     .
.     .     .     x_mn  y_mn
```

En este formato `m` × `n` es la dimensión de la malla. Expliquemos cada una de las entradas

`m`	Número de puntos en cada línea horizontal de la malla. Tengamos en cuenta que la dimensión de la malla es `mn`.
`n`	Número de puntos en cada línea vertical de la malla. Tengamos en cuenta que la dimensión de la malla es `mn`.
`filename`	Nombre del archivo de datos.
$x_{i,j}, y_{i,j}$	Coordenadas de los nodos de la malla. Las coordenadas se imprimen 7 por cada línea de archivo luego viene un salto de línea y se prosigue hasta finalizar todos los nodos.

Este formato como bien hemos señalado es natural en cuanto al orden empleado para nombrar los nodos como elementos de una matriz. El formato de impresión sigue siendo `F11.4` aunque para su lectura basta que los datos se encuentren separados por un espacio siguiendo la distribución de sus elementos.

A.5.2.2. Formato RED

El formato `RED` surge como una necesidad para delimitar los nodos a la frontera y los nodos interiores para de ésta forma, manipular fácilmente los nodos internos de la malla en el módulo de optimización. Los nodos interiores son las incógnitas al problema de optimización. Bajo ésta idea el formato `RED` describe primero los nodos a la frontera y seguidamente los nodos interiores, describamos el formato de los archivos. Un archivo típico de mallas bajo el formato `RED` tiene la forma

```
m     n
filename
x_11  y_11  x_21  y_21  x_31  y_31  x_41
y_41  x_51  y_51  x_61  y_61  x_71  y_71
. . .
. . .
. . .
. . . x_m1 y_m1  x_m2  y_m2
x_m3  y_m3  x_m4  y_m4  x_m5  y_m5  x_m6
. . .
. . .
. . .
. . . x_12 y_12  x_22  y_22
y_23  x_23  y_24  x_24  y_25  x_26  y_26
. . .
. . .
. . .
. . .x_2m-1 y_2m-1 x_32 y_32
x_33  y_33  x_34  y_34  x_35  y_35  x_36
. . .
. . .
. . .
. . .x_mn  y_mn
```

donde

`m`	Número de puntos en cada línea horizontal de la malla. Tengamos en cuenta que la dimensión de la malla es `mn`.
`n`	Número de puntos en cada línea vertical de la malla. Tengamos en cuenta que la dimensión de la malla es `mn`.
`Filename`	Nombre del archivo de datos.
x_r, y_r	Coordenadas de los nodos de la malla. Los datos se imprimen 7 por cada línea de archivo luego viene un salto de línea y se prosigue hasta finalizar todos los nodos. Primero se imprimen los nodos de la frontera 1, luego los que se encuentran en la frontera 2 sin repetir, luego los nodos de la frontera 3 sin repetir y finalmente los de la frontera 4 sin repetir ni uno. Luego vienen los nodos interiores imprimiéndolos por líneas verticales inmediatamente después de la frontera 4 hasta antes de la frontera 2.

Este formato permite contar en un arreglo a todos los nodos de la malla y a partir de un número $ibo = 2(2m+2n-4)$ todos los nodos interiores. El formato de impresión de los datos es `F11.4`, es decir, con cuatro cifras decimales.

A.6. Ejemplos

A continuación, presentamos algunos ejemplos de mallas diseñadas de forma automática con el sistema `Driv_Gen_mall`

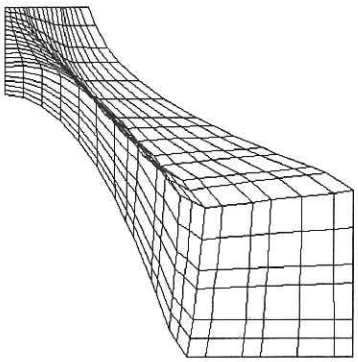


Figura A.14: Malla de la región `TIE` generada con TFI

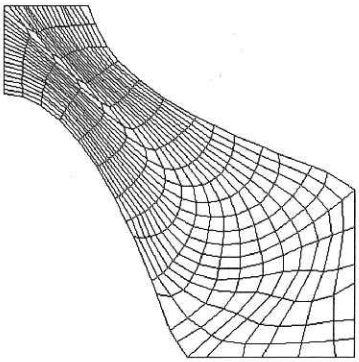


Figura A.15: Malla optimizada con A-O

En el siguiente ejemplo se presenta una región para la cual por la forma en que esta parametrizada su frontera el resultado de generar una malla inicial con TFI es una malla no admisible. Figura (A.16). Por otro lado, en la Figura (A.17) la malla sobre este contorno ya es admisible debido a una reparametrización de la frontera. Así, es posible optimizar la malla con área-ortogonalidad, Figura (A.18).

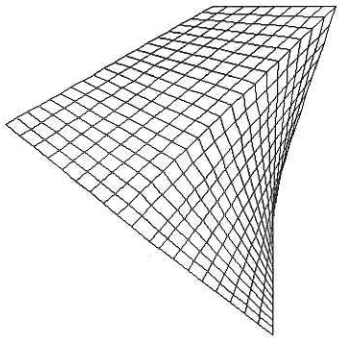


Figura A.16: Malla *no admisible* de la región `M23` generada con TFI

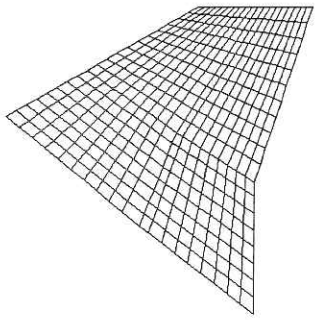


Figura A.17: Malla *admisible* de la región `M23` generada por TFI

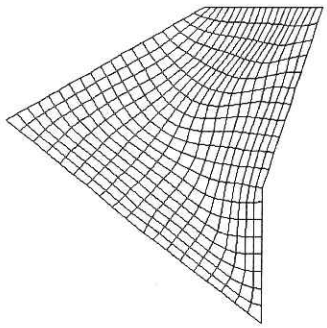


Figura A.18: Malla optimizada con A-O

Por último, presentamos un ejemplo donde se observa que por TFI obtenemos una malla convexa, Figura (A.19), pero cuando optimizamos resulta una malla con una celda no convexa, ver Figura (A.20).

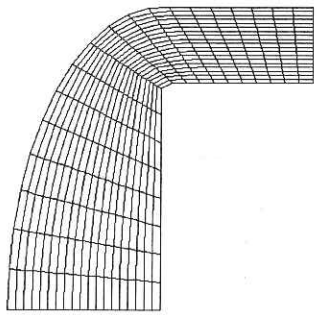


Figura A.19: Malla de la región `PLOW` generada por TFI

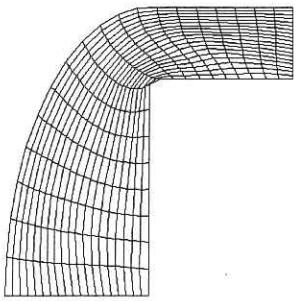


Figura A.20: Malla optimizada con A-O

A.7 Referencias

[1] Barrera, P., Castellanos, L., y Pérez, A. 1994. *Métodos Variacionales Discretos para la Generación de Mallas*, DGAPA-UNAM, México

[2] Barrera, P., González, G., Pérez, A. y Castellanos, L. 1994. *Manual de Usuarios del Sistema UNAMALLA v. 1.0: Generación de Mallas Planas sobre Regiones Irregulares* DGAPA-UNAM, México.

[3] Barrera, P. and Tinoco J.G. 1997. *Smooth and Convex Grid Generation over General Plane regions* in Mathematics and Computer in Simulation.

[4] Barrera, P., García, I. y González, G. 2000. *Manual Operativo del Sistema UNAMALLA v. 2.0 para PC*, Cuadernos de Investigación, Área I, Física-Matemáticas e Ingeniería, Universidad Autónoma de Coahuila, México.

[5] Domínguez-Mota, F.J. 2005. *Sobre la Generación Variacional Discreta de Mallas Casiortogonales en el Plano*, Tesis de Doctorado, UNAM, México.

[6] González Flores, G.F., 1994. *Generación de Mallas en Regiones Planas Irregulares*. Tesis de Licenciatura. Universidad Autónoma de Yucatán, Yucatán, Méx. 1994.

[7] Knupp, P. and Steinberg, S. 1993. *Fundamentals of Grid Generation*. CRC Press, Inc.

[8] Steinberg, S., and Roache, P.J., 1986. *Variational Grid Generation*, Num. Meth. for P.D.E.s., {\bf 2}, 71--96.

[9] Tinoco, J.G. 1997. *Funcionales Discretos para la Generación de Mallas Suaves y Convexas sobre Regiones Planas Irregulares*, Tesis de Doctorado, CIMAT, México.

Apéndice B: Reconstrucción de Superficies usando NURBS

Módulo Reconst-0.0

B.1. Introducción.

Este módulo contiene rutinas en C++ con las cuales es posible reconstruir la geometría de yacimientos simples, a partir de un conjunto de curvas de nivel usando una superficie NURBS, y la biblioteca NURBS++ desarrollada por Philippe Lavoie[1].

La rutina principal `Reconst.cpp` llama al conjunto de funciones (ver archivo `funciones.h`), que forman el algoritmo de reconstrucción que se describe en la siguiente sección. Así, para su funcionamiento es necesario pasarle un archivo en el cual estén los puntos que describen a las curvas de nivel[2] del yacimiento que se desea reconstruir, una vez hecho esto el programa preguntará por el número de renglones y columnas de puntos que se desea tener en la superficie reconstruida. Cuando estos son proporcionados, el programa generará un archivo llamado `puntos.txt` con las coordenadas de estos, adicionalmente se creará un archivo en formato VRML con el que se puede visualizar la superficie reconstruida con algún software libre para tal efecto.

B.2. Descripción del algoritmo.

El algoritmo de reconstrucción de la geometría del yacimiento que se implementa en Reconst-0.0 utiliza técnicas de interpolación NURBS, las cuales requieren por definición que los puntos de las curvas de nivel tengan un cierto orden. Por esta razón los primeros pasos del procedimiento se encargan de ordenar los puntos de cada curva de nivel con respecto a la siguiente curva. Una vez realizado esto, se reconstruye la superficie usando una superficie "Skinned".

Si denotamos como $P^i = \{P_0^i,...,P_{mi}^i\}$ al conjunto de puntos de la i-ésima curva de nivel y $C_i(u)$ la respectiva curva de interpolación NURBS, en donde $u \in R$ y $i = 0,...,k$ para $k > 0$, el proceso de reconstrucción sigue los siguientes pasos.

Algoritmo.

1. Ordenar cada punto del conjunto P_{i+1} de acuerdo a la distancia más pequeña que hay a los puntos de P_i para $i = 0,...,k-1$.

```
for (l=0; l<=n; l++)
  for (j=0; j<=n_{i+1}; j++)
    P_l^{i+1} = distancia_minima{P_l^i, P_j^{i+1}}
```

[1] Ver en http://libnurbs.source.net/
[2] El formato de este archivo se describe en la sección 3

2. Si $\overline{U} = \{\overline{u}_0,...,\overline{u}_n\}$ es un conjunto de valores parametrales equidistantes tales que $\overline{u}_i \in [0,1]$ para $i = 0,...,n$. Entonces para cada punto $P_l^i = C_i(\overline{u}_l)$ encontrar el punto $P_m^{i+1} = C_{i+1}(\overline{u}_m)$ cuya distancia es mínima. Para $i = 0,...,k-1$ y $l,m = 0,...,n$, es decir

```
for (l=0; l<=n; l++)
  for (m=0; m<=n; m++)
```
$$P_l^{i+1} = \texttt{distancia\_minima}\{P_m^{i+1}, P_l^i\}$$

3. Realizar la interpolación de los nuevos puntos P^i y obtener las nuevas curvas de interpolación NURBS $C_i(u)$, para $i = 0,...,k$.
4. Construir la superficie "Skinned" con las curvas $C_i(u)$, para $i = 0,...,k$.

B.3. Modo de uso.

Para poder usar el programa Reconst es necesario tener instaladas las bibliotecas Nurbs++, OpenGL y Glut (o su versión libre Mesa) las cuales se pueden obtener libremente bajo la licencia GNU en las páginas de Internet:

- http://libnurbs.sourceforge.net/download.shtml
- http://www.mesa3d.org/

Así una vez instaladas todas éstas bibliotecas, el funcionamiento de `Reconst-0.0` requiere un archivo de datos en donde el primer valor debe indicar el número de curvas de nivel y a continuación deben estar los puntos de las curvas de nivel en coordenadas R^3, las cuales deberán separarse con

```
1.0000000E+30 1.0000000E+30 1.0000000E+30
```

por ejemplo

```
5
3916.439 4688.086 3025.000
4012.789 4559.529 3025.000
4446.365 4286.343 3025.000
4623.007 4382.761 3025.000
4863.882 4430.970 3025.000
5313.515 4623.808 3025.000
5458.041 4896.993 3025.000
5329.574 5395.155 3025.000
4976.290 5636.201 3025.000
4767.532 5620.131 3025.000
4430.306 5491.574 3025.000
4237.606 5427.294 3025.000
3964.614 5411.225 3025.000
3707.681 5218.388 3025.000
3755.856 4768.435 3025.000
3964.614 4672.017 3025.000
3916.439 4688.086 3025.000
1.0000000E+30 1.0000000E+30 1.0000000E+30
3787.973 6118.294 3235.000
3595.272 5957.596 3235.000
. . .
. . .
. . .
```

se recomienda que los archivos de datos se guarden en la carpeta datos, así para correr en un sistema Unix, se debe teclear

```
[Reconst-0.0]$ ./Reconst datos/nombrearchivo.txt
```

Una vez hecho esto, el programa preguntará el número de puntos que se requieren para los renglones y columnas de la malla sobre la superficie reconstruida (equivalente al número de isocurvas para los dos valores parametrales u,v [3]), una vez que el usuario los proporcione se creará un archivo `puntos.txt` en donde estarán los puntos de la superficie evaluada.

Una corrida típica de `Reconst-0.0` sería

```
[Reconst-0.0]$ ./Reconst datos/cima25.txt
Reconst: Proporciona el número de renglones y
columnas de la malla sobre superficie
40 60
Reconst: El archivo de salida es: "puntos.txt"
```

Adicionalmente, `Reconst-0.0` contiene su Makefile (para ambientes UNIX), el cual se ejecuta con la instrucción:

```
[Reconst-0.0]$ make
```

Como ejemplo del funcionamiento de Reconst-0.0, se presentan en la figura B.1 las curvas de nivel con las que se desea reconstruir las superficies en el archivo `cimas25.txt`. En las figuras B.2 la malla 2D generada por Reconst-0.0 (archivo puntos.txt) tomando 20×20 renglones y columnas, en la figura B.3 tomando 40×40 y finalmente en la figura 4 la superficie reconstruida (archivo `superficie.wrl`).

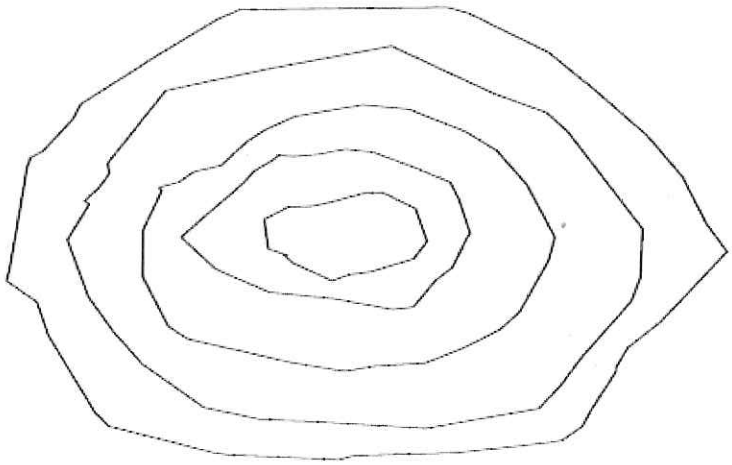


Figura B.1. Curvas de Nivel

[3] Esto es por la definición de la superficie NURBS

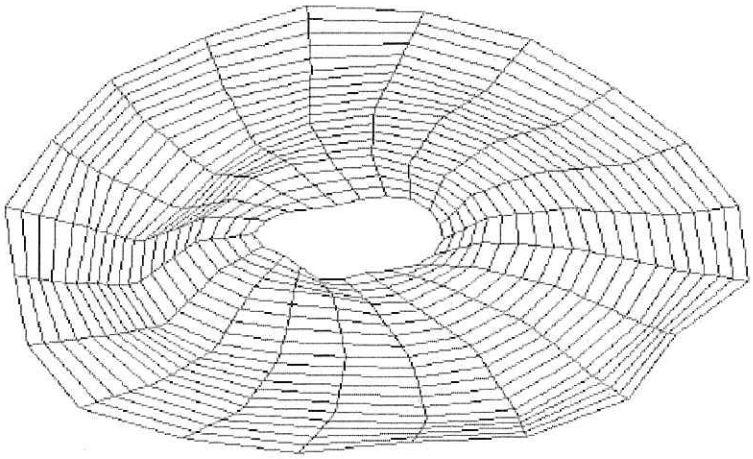


Figura B.2. Malla 2D tomando 20×20

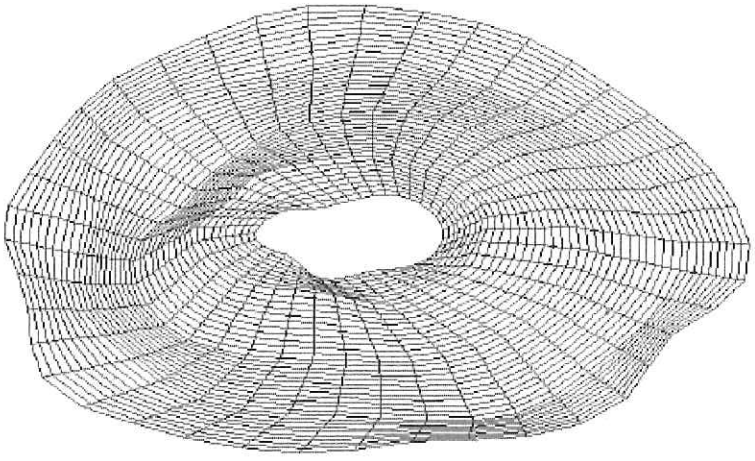


Figura B.3. Malla 3D tomando 30×30

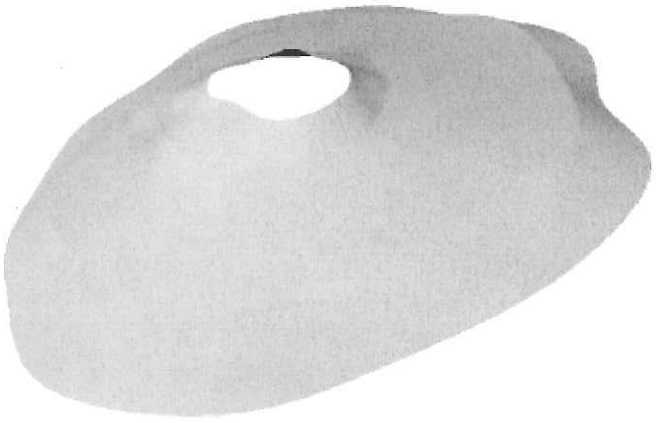


Figura B.4. Superficie resultante

B.4. Referencias

[Blom91] *Bloomenthal M. y Riesenfield R.F.*, Aproximation **of Sweep Surfaces by Tensor Product NURBS**,Curves and Surfaces in Computer Vision and Graphics II, Society of Photo-Optical Instrumetation Enginneers, Vol. 1610 páginas 132-144, 1991.

[Coqu87] *Coquillart S.*, A **Control Point Based Sweeping Technique**, IEEE Comput. Graph and Appl., Vol 7, No.11, páginas 36-45, 1987.

[Cox82] **Practical Spline Aproximation**, *M.G. Cox*, NPL Report Division of Information Technology and Computing, 1982.

[DeBoor78] **A Practical Guide to Splines**, *De Boor, C.*, New York: Springer-Verlag, 1978.

[Far90] **Curves and Surfaces for CAGD, a practical guide**, *Gerald Farin*, Academic Press, Inc. Third Edition, 1990.

[PT97] **The NURBS Book** *Les Piegl*, *Wayne Tiller*, Springer-Verlang 1997.

[Rog01] *David F. Rogers*, **An Introducction to NURBS With Historical Perspective**, Morgan Kaufmann Pubishers 2001.

[tesis] *Cervantes Cabrera Daniel*, **Estudio de la Construcción de Superficies con NURBS**, **Tesis de Maestría**, Posgrado en Computación UNAM, 2005.

Apéndice C: Reconstrucción de Superficies usando el método de Interpolación Bivariada

C.1. Planteamiento del problema

Dada una colección de puntos en el plano $\{(x_i, y_i)\}^n{}_{i=1}$ distribuidos arbitrariamente, y una colección de valores $\{z_i\}^n{}_{i=1}$ asociados con cada punto en el plano, el problema es determinar una función suave $F(x,y)$ sobre la región más pequeña y convexa Ω que contiene a los puntos, de manera tal, que interpole a los datos, es decir

$$F(x_i, y_i) = z_i, \qquad \text{para } i = 1,...,n.$$

Cabe señalar, que la región Ω se le conoce como la "cáscara convexa" de la colección de puntos $\{(x_i, y_i)\}^n{}_{i=1}$ en el plano, esto es, Ω es la región convexa más pequeña que contiene a los puntos $\{(x_i, y_i)\}^n{}_{i=1}$.

Usualmente, la solución a este problema se puede dividir en tres partes: triangulación de Ω, estimación de las derivadas parciales, y construcción del interpolante a evaluar.

1. Construir una triangulación para Ω.
2. Estimar las derivadas parciales de F con respecto a x y y en cada nodo, usando la información de los nodos más cercanos.
3. Para un punto (x,y) en Ω, determinar el triángulo que contiene a dicho punto, y calcular un valor interpolado $z = F(x,y)$, usando los valores de los datos y las derivadas parciales de cada vértice del triángulo.

La forma de resolver este problema de interpolación, puede cambiar radicalmente dependiendo de la triangulación a emplear, y la forma en que se construye el interpolante. En este trabajo, usaremos el interpolante utilizado por Akima dentro del algoritmo ACM 526, para resolver el problema de interpolación que está inmerso en la reconstrucción de las capas. Discutiremos las debilidades del algoritmo, para proponer el estudio de otros algoritmos.

C.2. Descripción del algoritmo ACM 526

El método de interpolación bivariada que emplearemos en esta fase del proyecto, es el algoritmo ACM 526 desarrollado por Hiroshi Akima, el cual realiza una triangulación sobre Ω tomando como vértices las proyecciones al plano de los puntos $\{(x_i, y_i, z_i)\}^n{}_{i=1}$ proporcionados, una vez hecho esto, se construye un polinomio de interpolación bivariado de grado 5 para cada celda formada por un triángulo, considerando para esto, los valores estimados de las derivadas parciales de primer y segundo orden en cada vértice del triángulo.

Para la triangulación, se adopta el criterio llamado "*max-min angle*" sugerido por Lawson [3]; este criterio es simple: primeramente se conecta el par de puntos más cercanos, y una vez hecho esto se construye un triángulo agregando un nuevo punto en orden ascendente a la distancia del punto medio de este par de puntos, lo cual asegura que

los nuevos estén fuera del polígono construido con los anteriores. El proceso continua de manera recursiva sobre los siguientes pares de puntos. Un ejemplo de este proceso se muestra en la Figura C.1.

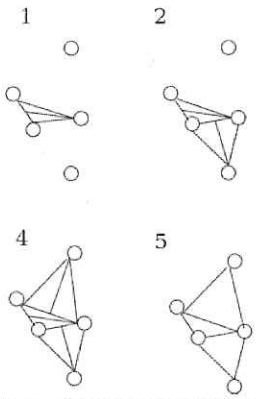


Figura C.1. Proceso de triangulación.

La interpolación de los valores de $F(x,y)$ en cada triángulo está basado en las siguientes tres suposiciones:

a. El valor de la función en el punto $(x,y)\in\Omega$, sobre un triángulo, es interpolado por el polinomio bivariado de grado 5 de la forma

$$F(x,y)=\sum_{j=0}^{5}\sum_{k=0}^{5-j}q_{jk}x^{j}y^{k}$$

Para determinar unívocamente este polinomio, debemos determinar los valores de 21 coeficientes.

b. Los valores de la función y su primer y segunda derivada parcial, es decir $F,F_x,F_y,F_{xx},F_{xy},F_{yy}$ deben ser estimados sobre cada vértice del triángulo, notemos que esto impone un total de 6x3=18 condiciones para la función $F(x,y)$.
c. Las derivadas parciales de la función $F(x,y)$ en dirección normal a cada lado del triángulo es un polinomio de a lo más grado tres.

Ya que el triángulo tiene tres lados, esta suposición implica tres condiciones, por lo que tenemos un total de 18+3=21 condiciones para $F(x,y)$, por lo que el polinomio está completamente determinado.

Para estimar las derivadas parciales de primer orden en un punto P_0 se usan varios puntos P_i, $i=1,...,n_c$ de los datos en el plano cercanos a P_0. Con esto, se consiederan dos puntos P_i y P_j de los n_c puntos y calculamos el vector que resulta del producto cruz de los vectores $\overrightarrow{P_0P_i}$ y $\overrightarrow{P_0P_j}$ cuya magnitud es igual al área del paralelogramo formado por

esos vectores. En este procedimiento se consideran las combinaciones posibles de $\overrightarrow{P_0P_i}$ y $\overrightarrow{P_0P_j}$ con $i \neq j$ y tomados con orientación positiva, y se elige la suma resultante de los vectores del producto cruz. Finalmente las derivadas de primer orden de F_x y F_y, en P_0 son estimadas como las derivas parciales del plano normal al vector de suma resultante.

Para la estimación de la derivadas parciales de segundo orden aplicamos el procedimiento anterior, pero ahora aplicado sobre F_x y F_y, de tal forma, que F_{xx} en P_0 se aproxima sobre los valores F_x de los puntos P_i, $i=1,...,n_c$, más cercanos a P_0, lo propio se realiza para F_{yy}, donde se usan los valores F_y de los puntos P_i, $i=1,...,n_c$, más cercanos a P_0. Para calcular F_{xy} se lleva a cabo un promedio de la aproximación de este procedimiento sobre $(F_x)_y$ y $(F_y)_x$.

C.3. Modo de uso de la rutina `idbvip.f90`

El algoritmo ACM 526 de Akima, se encuentra programado dentro de la rutina `idbvip`, la cual se encuentra escrita en Fortran90. Para hacer uso de esta rutina, se desarrolló un programa `interpbivar.f90` que lee los datos de un archivo de texto plano `datos3d.txt` y deposita la salida en un archivo de texto plano `salida.mesh`, esto para facilitar la interacción con el usuario.

El programa `interpbivar.f90` tiene la estructura modular siguiente:

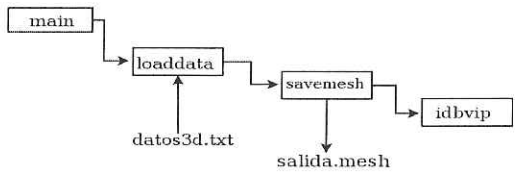


Figura C.2: Estructura modular de la rutina interbivar.f90

En esta se puede observar que la función principal `main` se encarga de llamar a la subrutina `loaddata` que como su nombre lo indica lee los datos de un archivo, cuyo nombre es proporcionado por el usuario.

El formato de éste debe ser:

```
No. de puntos 3d.
Domino de trabajo.
Puntos 3d
```

Un ejemplo de este formato es:

```
783
-2 2 -2 2
0.00000 0.00000 0.00000
-1.06000 2.00000 0.21400
-1.32600 1.80000 0.21400
.....
```

Una vez que la lectura se realiza con éxito, el programa pedirá al usuario que proporcione el nombre del archivo de salida el cual de preferencia se pide esté en formato `mesh` (formato del paquete libre `GeomView`), este proceso se lleva a cabo por medio de la subrutina `savemesh`.

C.4. Ejemplos

A continuación se muestran dos ejemplos que realizaron usando el algoritmo ACM 526. Para el primer ejemplo, se consideró una función de prueba dada por la expresión $F(x,y) = x^2 + 3y^2 \exp\left(1-\left(x^2+y^2\right)\right)$. Se consideraron 45 puntos distribuidos aleatoriamente sobre la región de trabajo $[-2,2]\times[-2,2]$. Los datos fueron graficados sobre una malla uniforme de 100×100 sobre la región de estudio.

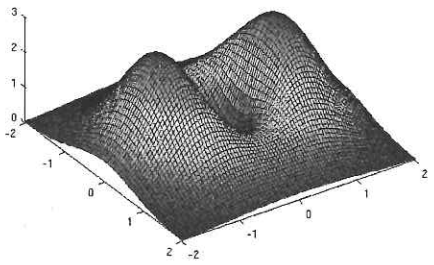


Figura C.3a: Superficie original

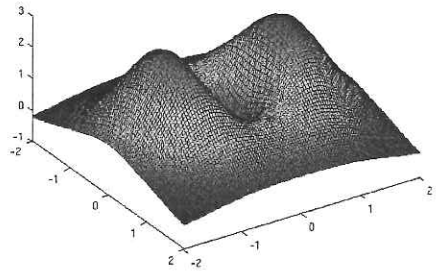


Figura C.3b: Superficie obtenida por el algoritmo ACM 526

Comparando las curvas de nivel de cada superficie, tenemos

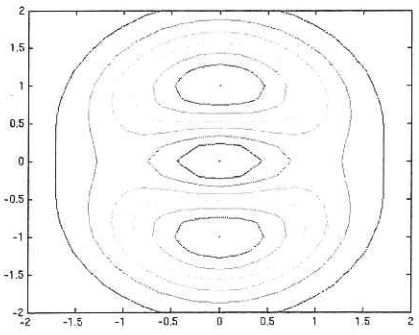


Figura C.4a: 8 curvas de nivel de la superficie original

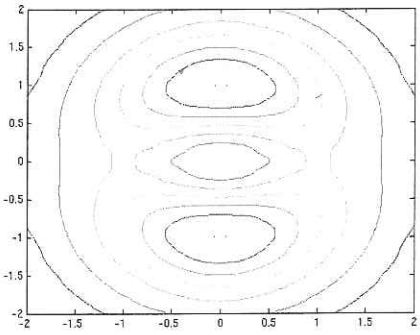


Figura C.4b: 8 curvas de nivel obtenidas de superficie calculada

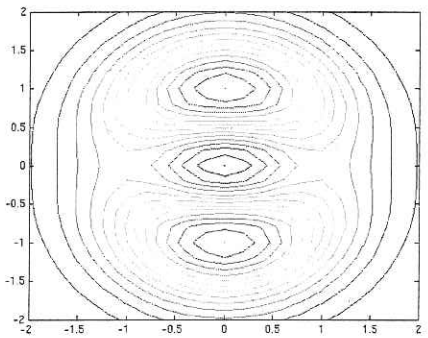


Figura C.4c: 15 curvas de nivel de la superficie original

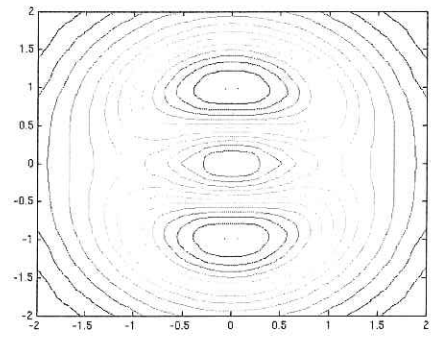


Figura C.4d: 15 curvas de nivel de la superficie calculada

Para el segundo ejemplo, se consideró una colección de datos proporcionados por el paquete GRID de ECLIPSE, estos datos presentan la forma:

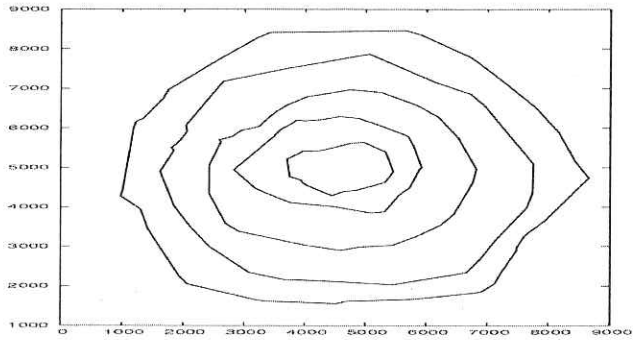


Figura C.5: Mapa de datos "cimascuadradas.dat"

Con el algoritmo de interpolación bivariada 526 se obtuvo la superficie:

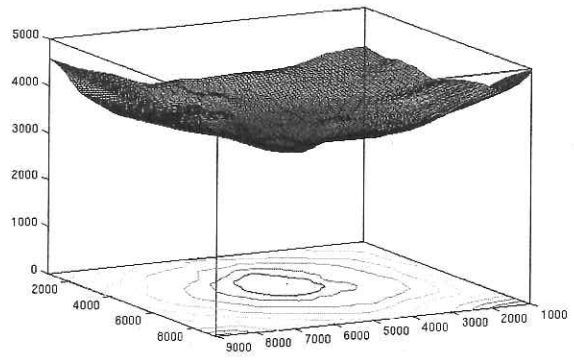


Figura C.6: Superficie reconstruida por el método de interpolación bivariada.

Algunas de sus curvas de nivel presentan la forma:

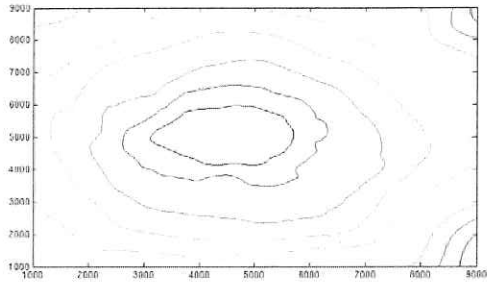


Figura C.7: Curvas de nivel de la superficie reconstruida.

Como se observa, la superficie presenta rugosidades suplerfluas, producto de la mala calidad de la información de los contornos y la mala distribución de los mismos. Como se ha comentado, es esencial que el usuario encargado del sistema conozca las bondades y debilidades de cada método de interpolación o suavizamiento de datos con el fin de que pueda modificar o añadir información, con la finalidad de obtener resultados acordes a lo esperado.

En este caso, se sugiere un procesamiento previo de los contornos, por medio de interpolación y suavizamiento de los mismos.

C.5. Bibliografía.

[1] Akima, H. *Algorithm. A method of bivariate interpolation, and smooth surface fitting based on local procedures*. Comm. ACM 17, 1 (Jan. 1974), 18-20.

[2] Akima, H. *A method of bivariate interpolation and smooth surface fitting for values given at irregularly distributed points*. OT Rep. 75-70, U,S. Govt. Printing Office, Washington, D.C. Aug. 1975.

[3] Lawson, C.L. *Generation of a triangular grid with aplication to contour plotting*. Tech. Memo. 299, Sect, 914, Jet Propulsion Lab., Calif. Inst. Tech., Pasadena, Calif., Feb. 1972.

C.A Instrucciones de compilación y ejecución de `Interpbivar.f90`

Para realizar la compilación de interpbivar.f90 es necesario que antes se realize la compilación de bivar.f90, para esto es recomendable construir la biblioteca usando algún compilador de Fortran 90. En esta sección explicaremos la compilación suponiendo que se está trabajando sobre un sistema Linux y que se utiliza el compilador libre de Fortran 90, g95[4], entonces desde la línea de comandos se debe teclear:

```
$g95 -c bivar.f90
$ar  qc libbivvar.a bivar.o
```

Así una vez construida la biblioteca, para compilar `interpbivar.f90` escribimos:

[4]Para mayor información acerca de este compilador ver en http://www.g95.org

```
$g95 -o interpbivar.o -c interpbivar.f90
$g95 -o interpbivar interpbivar.o -lbivar
```

y así para ejecutar el programa escribimos en la línea de comandos

```
$./interpbivar
```

Seguidamente, debe proporcionar el nombre del archivo de puntos 3d, por ejemplo: `ejemplo1.txt`, para posteriormente, proporcionar el nombre del archivo de salida `salida.mesh`.

C.B Sobre el formato `mesh`

El formato `mesh` es útil para manejo de archivos que describen mallas estructuradas rectangulares sobre alguna superficie, el formato de archivo es el siguiente:

```
          MESH
# Pts. en la dirección x   # Pts. en la dirección y
valor en la direccón x          valor en la dirección y          valor de z
```

Ejemplo:

```
    MESH
    100          100
 0.00000      0.00000      4284.97
 0.00000      90.9091      4287.48
 0.00000      181.818      4290.13
 0.00000      272.727      4292.90
 0.00000      363.636      4295.82
 0.00000      454.545      4298.86
 0.00000      909.091      4258.24
 0.00000      1000.00      4262.23
  .....
```

C.C Los módulos `savecontourns.m` y `removequals.m`

Adicionalmente al módulo `interpbivar.f90`, se presentan dos programas realizados en `Matlab`, los cuales nos han sido útiles para la generación de archivos de contornos de una función analítica bivariada.

El programa `savecontourns.m` genera un archivo de contornos de una función analítica bivariada usando la función `contour.m` de `Matlab`, para usarla es necesario tener definida una función analítica y pasarle ciertos parámetros. El prototipo de esta función es como se indica a continuación

```
savecontourns(function,a,b,c,d,step,no_contours)
function          - Función análitica bivariada.
a,b,c,d           - Rangos parametrales del dominio de estudio,  a<b y c<d.
```

`step` - Delta de partición del dominio de estudio.
`no_contours` - Número de contornos.

Ejemplo de uso:

```
>> f = @(x,y) (x.^2+3.*y.^2).*exp(1-(x.^2+y.^2));
>> savecontorns(f,-2,2,-2,2,0.2,10);
```

El archivo resultante es llamado `contornos.txt`.

Ya que el algoritmo de interpolación bivariada que usa el módulo `interpbivar.f90` requiere que los puntos 3D proporcionados sean distintos, el programa `removequals.m` se encargar de eliminar los puntos iguales, el formato de lectura del archivo que se desea depurar debe de seguir el siguiente formato:

```
Número de puntos.
Puntos 3d.
```

Ejemplo:

```
   819
0.000 0.000 0.000
-0.000 0.000 0.000
0.000 -0.000 0.000
0.000 0.000 0.000
0.000 0.000 0.000
-1.060 2.000 0.214
-1.200 1.909 0.214
-1.326 1.800 0.214
-1.400 1.729 0.214
-1.519 1.600 0.214
-1.600 1.485 0.214
-1.663 1.400 0.214
-1.769 1.200 0.214
-1.800 1.112 0.214
-1.852 1.000 0.214
-1.912 0.800 0.214
-1.948 0.600 0.214
.....
```

El archivo resultante es llamado `contornos2.txt`.

Nota: Todas las rutinas escritas en Matlab, se encuentran disponibles dentro del directorio `interpbivar` del CD de distribución de los productos entregables.

Apéndice D: Propuesta para el diseño de la Interfaz Gráfica para el Sistema Simulador de Yacimientos Multipropósito: Módulo Generador de la Malla 3D de Simulación (sin fallas)

D.1. Antecedentes

En la simulación numérica de yacimientos, una parte muy importante es el proceso de pre-procesamiento y post-procesamiento de la información que se tiene y la que generará el simulador numérico multipropósitos (objetivo del presente proyecto). En la etapa de pre-procesamiento de información se desea que el simulador multipropósitos esté equipado con algoritmos de solución nuevos y actualizados, que le permitan ser una opción que pueda competir con la que ofrecen algunos simuladores comerciales. Entre las primeras fases, se encuentra aquella donde se requiere la construcción de una malla sobre un dominio plano. Esta malla plana se obtendrá por interpolación y posteriormente se obtendrá una mejorada casi ortogonal.

Una herramienta que será muy útil es la reconstrucción y visualización de la geometría del yacimiento a partir de los contornos, o bien, a través de una cuadrícula (mesh) de la región de estudio. Esto será importante para identificar la forma y las posibles dificultades que presente el yacimiento. Contando con la reconstrucción de la forma del yacimiento, el objetivo será construir una malla 3D de simulación.

D.2. Objetivo

El objetivo del presente trabajo es describir las características deseables de un sistema semi-automático con el cual obtener una malla 3D de simulación sobre un yacimiento de hidrocarburos.

D.3. Descripción general del problema

La forma usual de trabajo en simuladores numéricos comerciales, es a través de módulos, cada uno de los cuales se encarga de una tarea particular. Este es el caso del módulo que nos ocupa, cuya tarea es definir la malla de simulación 3D.

En la práctica, se cuenta con información de diversas propiedades que constituyen el yacimiento de hidrocarburos, como pueden ser: profundidad, porosidad, etc. Estas propiedades son mediciones que los Geólogos encargados del estudio de la conformación material del yacimiento proveen para su estudio. Típicamente se entrega al personal encargado de la simulación, datos digitalizados en un sistema numérico, y distribuidos de manera particular por contornos o curvas de nivel o isolíneas, datos dispersos o datos provenientes de una retícula rectangular sobre el área de estudio, que corresponden a la propiedad o propiedades a medir.

El operador del sistema debe introducir esa información dentro del simulador a través de datos digitalizados, para posteriormente y a través de técnicas de interpolación, extrapolación y suavizamiento, contar con una buena representación numérica de las propiedades del yacimiento. Usualmente, quienes realizan ese tipo de mediciones lo llevan a cabo a través de **capas** o **formaciones rocosas** que consideran son importantes para la identificación de las propiedades del yacimiento.

Cada capa o formación rocosa, la constituyen sedimentos de roca acumulados en las diferentes eras de formación de la tierra, es por ellos que de identificarlos nos proveerán

de una información valiosa sobre la posible cantidad y calidad del yacimiento para su posterior proceso de extracción en primera y segunda fase.

La descripción de cada capa se logra a través de mediciones logradas con instrumentos geofísicos, proveyendo información sobre una colección de datos dispersos, o a través de contornos o isolíneas de información.
La representación gráfica de este procedimiento la observamos en la siguiente Figura:

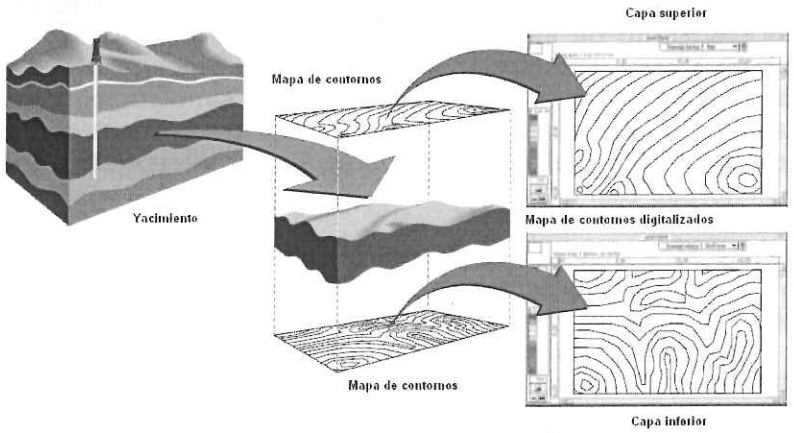


Figura D.1: Mapas de contornos de la base y la cima de un yacimiento.

Sin embargo, no es la única forma de proveer esta información. Esta puede ser provista a través de una colección de datos provenientes de una cuadrícula rectangular sobre el área plana de trabajo (malla gruesa o mesh), o bien, a partir de datos aislados (datos dispersos, *scattered data*) o bien, como se observa en la Figura D.2.

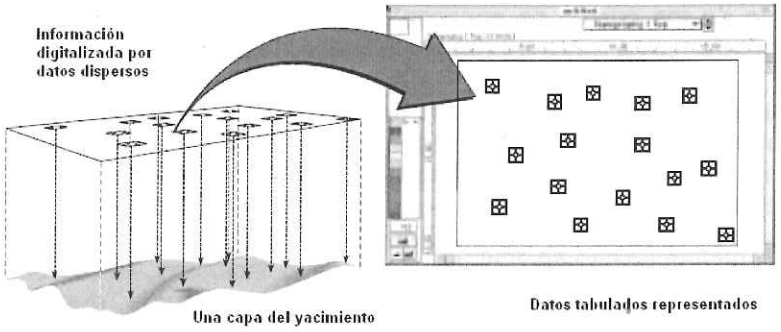


Figura D.2: Datos dispersos de una cima del yacimiento.

Contando con los datos, el sistema debe ser capaz de reconstruir numéricamente y de manera adecuada, la superficie que representa cada capa para entonces, tener una representación eficiente de una sección de estudio del yacimiento. Desde luego, esta representación depende en gran medida del número de capas necesarias y del método numérico de interpolación y suavizamiento usado para reconstruir cada capa o superficie.

Una vez que se cuente con una forma de reconstruir una sección del yacimiento, el siguiente paso es construir una malla 3D de simulación. La manera usual de lograrlo, es considerar una malla 2D suficientemente suave y ortogonal, o cercana a ello, y levantarla

sobre cada capa para que, localmente, se conforme una malla 3D al unir cada malla 2D sobre cada capa, una malla 3D. La idea se muestra en la Figura D.3.

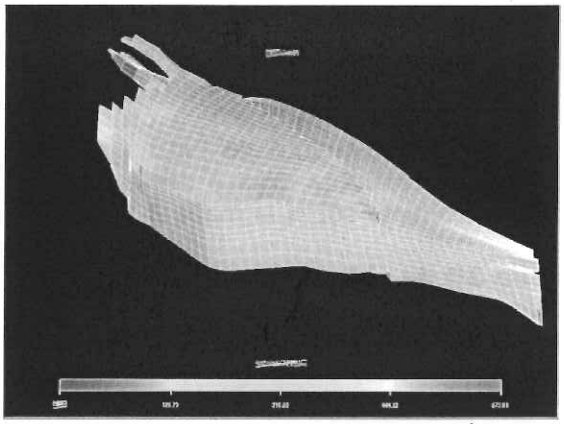


Figura D.3: Una malla 3D de simulación.

Y con esto, llevar a cabo los procesos de simulación del yacimiento de hidrocarburos.

La idea es que el sistema contemple éstas tareas en un sólo módulo integral que construye la malla 3D de simulación a partir de los datos digitalizados. Las principales tareas contempladas para este módulo integral o sistema, son las siguientes:

- **Datos digitalizados de cada capa**
- **Reconstrucción de cada capa**
- **Construcción de una malla plana 2D**
- **Construcción de una malla 3D de simulación**

D.4. Breve descripción de las tareas

Datos digitalizados
Los datos que competen a cada capa, pueden ser introducidos como una colección de datos dispersos, datos sobre una rejilla gruesa sobre un área de estudio, o bien a través de contornos o isolíneas. Esta información puede ser obtenida a través de archivos de trabajo de texto plano (para facilitar su interacción con el usuario y quien provee la información) o bien introducidos directamente en pantalla a través de herramientas de edición de curvas poligonales cerradas y abiertas, así como datos dispersos.

Reconstrucción de cada capa
A partir de una colección de datos digitalizados de cada capa, se debe contar con una representación numérica de la superficie que describen. Para lograrlo, es necesario contar con diferentes técnicas de interpolación sobre datos 2D dispersos, o que siguen la forma de un contorno o curva poligonal. No necesariamente se debe hacer uso de técnicas de interpolación de datos; en muchas aplicaciones es conveniente contar con técnicas de suavizamiento de datos, dado que estos pueden provenir de mediciones con un margen alto de error por medición. La reconstrucción de cada capa es un paso muy delicado, ya que debe poderse obtener una superficie que siga la tendencia de los datos y evite, las más de las veces, rugosidades superfluas.

Construcción de una malla plana 2D
La idea de la construcción de una malla de simulación 3D a partir de las capas que conforma la sección de yacimiento de estudio, se basa en definir una malla plana 2D sobre la región de estudio con las características de ser suave (contar con cambios diferenciales pequeños a lo largo de las curvilíneas) y casi ortogonal para posteriormente mapearla o levantarla sobre cada capa. Es sumamente importante que la malla plana 2D sea adecuada para la simulación, que sea suave y casi ortogonal, para obtener una eficiente simulación numérica del yacimiento de hidrocarburos. En muchas ocasiones, será necesario contar con una herramienta gráfica de "ajuste" manual, que nos permita mover líneas o fijar puntos particulares sobre la malla 2D.

Construcción de una malla 3D de simulación
Contando con la reconstrucción de las capas y con la malla plana 2D sobre la región de estudio, podemos obtener de manera directa, una malla sobre la superficie, y con esto, al unir cada malla sobre las superficies, contar con una malla 3D. Este procedimiento no es el óptimo; sin embargo, es muy simple y económico de implementar. La generación directa o variacional de mallas 3D, no se limita a unir mallas sobre superficies por segmentos rectilíneos; en todo caso, una malla sobre una superficie debe seguir la forma de la superficie, adaptarse a ella, y posteriormente, unir las mallas de manera adecuada. Esta será una tarea pendiente a futuro.

El diseño del sistema debe contemplar la posibilidad de que de manera automática pueda obtenerse la reconstrucción de cada capa, la malla plana, y por consiguiente la malla 3D de simulación. Naturalmente, esto no será suficiente para lograr una buena representación de la malla de simulación, por consiguiente, deberá considerarse una serie de tareas específicas que permitan al usuario modificar los resultados obtenidos, de manera que ya sea a través de procedimientos de ajuste e interpolación, así como de forma manual y directa sobre las líneas y nodos se puedan manipular los resultados.
Esto implica el desarrollo de herramientas gráficas para tareas de carácter específico, para que, dependiendo del trabajo a realizar: datos, superficies, malla plana, una malla 3D, la herramienta se adecue a cada una de ellas.

El sistema deberá estar orientado a plataformas PC con despliegue gráfico Windows. El sistema será automático en el sentido de que con sólo unas pocas pulsaciones de teclas será posible lograr una malla 3D adecuada. Es deseable que se tome en cuenta un esquema modular de interacción con fin de que el crecimiento y mantenimiento del sistema queden plenamente garantizados.

D.5. Características del Usuario

Algo que es importante señalar, es que el sistema presentará los resultados de manera gráfica, y es que la calidad de los resultados obtenidos será determinada por la experiencia del usuario; es decir, será el usuario quien determine si son suficientes los resultados o bien debe proveerse información adicional. Dependiendo del tipo de usuario el sistema integral hará el uso amplio o restringido de las tareas probables a realizar. Por tal motivo se han diseñado dos modos de uso del sistema: automático, y manual.

Los usuarios potenciales del sistema más que profesionales en la construcción y obtención de mallas 3D óptimas, son expertos en las áreas donde desean aplicar la malla de simulación, por lo que solamente desean obtener una discretización razonable de la región de estudio 3D. Para estos usuarios es necesario que el sistema cuente con una modalidad automática, tanto para la reconstrucción de las capas, como la construcción de una malla plana de estudio, y desde luego, la malla 3D de simulación.

Otro de los usuarios tipo para el sistema es el "experto", quien conoce cada paso involucrado en la obtención de la malla 3D de simulación. Por una parte, debe entender las debilidades o deficiencias de cada método de interpolación o suavizamiento involucrados en la reconstrucción de las superficies. Por lo cual se le deberá proveer de las opciones generales de cada método a través de opciones de menús, a fin de que pueda experimentar con los diversos parámetros y obtener aquella superficie que crea se ajusta a los datos. Por supuesto, debe contar con las mismas facilidades quien sea conocedor de los detalles finos involucrados en la obtención de una malla plana sobre la región de estudio, y desde luego lo correspondiente a la malla 3D de simulación.

En cualquiera de los casos el usuario, debe contar con un mínimo de conocimientos en el uso del sistema de ventanas y con herramientas gráficas de edición, para así, poder interactuar con el sistema de manera satisfactoria.

Restricciones generales para el usuario

La restricción principal con la que se topará el usuario del sistema, es que se trabajará con una colección finita de datos que determinan cada objeto geométrico.

Sobre la edición de contornos

Por una parte, cuando se introducen los datos digitalizados de las capas, estos representan una colección de datos dispersos o de una cuadrícula, o conforman polígonos que representan contornos o curvas de nivel de las propiedades, es decir, el usuario trabajará sobre segmentos de línea recta para construir y operar los contornos. Esto limitará en gran medida los resultados obtenidos, ya que dependerá fuertemente la posición de los puntos para poder obtener una representación razonable. Es por esta razón por la que el sistema deberá contar con herramientas gráficas que permitan añadir eliminar y suavizar secciones de cada contorno.

La primera versión del sistema contemplará poder interpolar y/o suavizar cada contorno, pero a final de cuentas, siempre contará con una colección finita de puntos para representarlo.

Sobre la reconstrucción de cada capa

La primera versión del sistema contemplará un par de métodos de interpolación y suavizamiento para la reconstrucción de la superficie. En una primera etapa, esta información será representada como una cuadrícula en la región plana de trabajo, conteniendo en cada nodo, el valor aproximado sobre la superficie. La segunda etapa contará con la representación gráfica de la superficie a través de curvas de nivel, ya que estas nos proveen de una información directa de la forma de la superficie. Es importante señalar que algunos métodos son sensibles a la distribución de puntos (básicamente de

la malla triangular o rectangular a construir sobre de ellos) y por consiguiente la forma de la superficie se verá afectada.

Sobre la construcción de una Malla plana 2D
En la construcción de la malla plana inicial, los puntos sobre la frontera son distribuidos uniformemente buscando conservar la forma de la frontera de la región y son sobre éstos puntos fijos sobre los que se trabaja para obtener la malla. La construcción inicial de la malla dependerá fuertemente de esta distribución, parametrización que influye en los métodos discretos. Acorde con la teoría: si es posible construir una malla plana convexa sobre la región de estudio será entonces posible hallarla con el sistema, a través de los funcionales discretos implantados.
La construcción de la malla 2D es una pieza importante, ya que debe garantizar una adecuada discretización de la región plana de estudio a través de las celdas (cuadriláteros convexos), los métodos discretos para obtener una malla plana suave y casi ortogonal implantados en la primera etapa, garantizan las más de las veces obtener una malla convexa y con líneas suaves. En la segunda etapa del desarrollo del sistema serán incorporados funcionales que nos permitan garantizar mallas suaves, convexas y casi ortogonales.

Sobre la construcción de la Malla 3D de simulación
La malla 3D de simulación será construida a partir de las mallas sobre la superficie de cada capa al unir cada una de ellas por segmentos de líneas. Esta desde luego no es la mejor forma, pero será una muy buena aproximación si no existen grandes variaciones de la forma de cada capa. Otra restricción, es que la malla sobre la superficie de cada capa es obtenida al mapear la malla 2D sobre la región de trabajo sobre la superficie, esto es, se cuenta con una malla sobre la superficie, no una malla que sigue la forma de la superficie o se adapta a ella. En una versión posterior, deberá contemplarse que la malla sobre la superficie se adapte a la forma de la superficie.

D.6. Requerimientos específicos

Requerimiento funcional 1: Ventana principal

- Resumen:
  Se requerirá de un entorno de trabajo provisto por una ventana de Windows, donde serán visualizados los gráficos, la construcción de las curvas de nivel, y de las mallas. Será necesario contar con otra ventana, donde se podrá tener un control directo sobre cada una de las secciones o tareas del sistema. La ventana principal, contará de una colección de menús, desde los cuales se operará con la tarea en curso, y algunas de las opciones del menú variarán, dependiendo de la tarea específica a trabajar. En todo momento se contará con una ayuda en línea capaz de indicarle al usuario, la tarea que lleva a cabo y cómo mejor desarrollarla. El trabajo con el sistema será a través de proyectos de trabajo, los cuales contarán con cada uno de los pasos para obtener una malla 3D de simulación.
- Entrada:
  Para la ejecución del sistema, el usuario no deberá requerir parámetro alguno de entrada. Podrá continuar con un proyecto de trabajo previo, o bien iniciar con uno nuevo.
- Procesamiento:
  Controla y distribuye las tareas principales del sistema.
- Salida:

Todo el trabajo se realiza en curso y sobre el sistema gráfico, en cada paso será posible guardar la información de la tarea en curso y contar con un historial de los pasos desarrollados, por lo que al concluir un proyecto, se deberá contar con una colección de archivos relacionados al proyecto de trabajo.

Requerimiento funcional 2: Construcción de un Mapa de Contornos

- Resumen:
  Es necesario contar con una herramienta que permita la construcción del mapa contornos o isolíneas de propiedades sobre las cuales llevar a cabo el estudio de simulación.
  La herramienta de captura o creación de un contorno es muy útil en áreas de experimentación con datos, por lo que las tareas aquí contempladas será útiles cuando definamos la región de estudio para la malla 2D.
- Entrada:
  No hay elementos de entrada ya que se construirá sobre la región de trabajo.
- Procesamiento:
  Se define los segmentos de contornos introduciendo puntos al ir pulsando botones del mouse. Conforme se introduce el contorno los datos del mismo se van añadiendo a una lista en memoria y estos datos son desplegados en pantalla y enumerados. No se debe permitir que se introduzcan puntos repetidos. Por otra parte, los contornos no se pueden intersectar, ni consigo mismo (curvas dobladas), ni con otros contornos.
- Salida:
  Al finalizar la captura, la información del contorno permanece en memoria y será sobre la que trabajará el usuario. El usuario podrá continuar con este procedimiento hasta que lo considere suficiente.

Requerimiento funcional 3: Edición de un Contorno

- Resumen:
  Debe ser posible editar cada contorno que se defina, ya sea del mapa de contornos o isolíneas de propiedades, o el contorno de la región poligonal de estudio para la malla 2D. Por editar el contorno, entenderemos la acción de modificarlo, reagrupar elementos, eliminar puntos, añadir puntos y suavizar el contorno, esto a través del uso del mouse, o bien, a través de un procedimiento automático diseñado para tal efecto.
- Entrada:
  El contorno en memoria, el cual será activado para edición al pulsar sobre de uno de los segmentos que lo definen con el indicador de mouse.
- Procedimiento:
  Al ser señalado como activo, se podrá proceder a su edición. Por ejemplo añadir puntos de manera manual y sobre el segmento que así se desee. Eligiendo un punto, podrá este cambiar de posición arrastrando el símbolo de punto que le define. De igual manera podrá eliminarse el punto con alguna tecla designada para tal propósito. Por otra parte, podrá optarse a suavizar el contorno por medio de un spline lineal o una curva NURBS (la primera versión del sistema, contempla el uso de spline lineal de interpolación). De igual manera, podrá optarse a eliminar puntos de manera automática del contorno, los llamados puntos superfluos.
- Salida:

Al finalizar este procedimiento, se contará en pantalla y en memoria con un contorno que reúne las características que mejor considere el usuario sean necesarias para definir el contorno.

Requerimiento funcional 4: Lectura de un Mapa de Contornos

- Resumen:
  El sistema debe contemplar mecanismos para tomar la información de un mapa de contornos o isolíneas de propiedades, de un archivo con un formato específico y llevarlo a memoria para su posterior uso.
- Entrada:
  Nombre de archivo en disco.
- Procedimiento:
  Mediante un *browser*, el usuario debe elegir el nombre del archivo y señalar su ubicación.
- Salida:
  El mapa de contornos o isolíneas de propiedades, en memoria.

Requerimiento funcional 5: Propiedades de un contorno

- Resumen:
  Cada contorno tiene propiedades asociadas de interés, como por ejemplo, el perímetro o el área que encierra (cuando se trata de contornos cerrados), el número de puntos, la profundidad asociada y la etiqueta. Estas propiedades serán desplegadas cuando así lo sea requerido por el usuario del sistema.
- Entrada:
  El contorno en memoria.
- Procedimiento:
  Al estar activo el contorno en memoria, se podrá hacer uso de la colección de puntos que describen el contorno y poder entonces llamar un par de rutinas que nos indiquen el perímetro del polígono que describen, así como el área de la región que encierran, en el caso de contornos cerrados. De igual manera se contará con la propiedad asociada al contorno, su valor y la etiqueta asignada.
- Salida:
  En una ventana de despliegue, se presenta esta información.

Requerimiento funcional 6: Construcción de un mapa de datos dispersos

- Resumen:
  Como se ha señalado, otra forma de contar con una discretización de la capa o superficie es a través de una colección de datos dispersos. Es necesario contar con una herramienta de edición que nos permita definir sobre la región de trabajo dicha colección de puntos.
- Entrada:
  La región de trabajo en pantalla.
- Procedimiento:
  A través de elegir la herramienta gráfica para tal propósito y con el uso del *mouse*, deberá introducirse la colección de datos en pantalla, por cada dato en posición (x, y), deberá asignarse el valor de la propiedad o profundidad de dicho punto. Esto se hará a través de una ventana de trabajo para tal efecto, donde será introducida de manera opcional, una etiqueta asociada a cada punto. A través de herramientas de

edición gráfica, podrá modificarse cada punto, esto es, cambiar de posición (arrastrarlo con el mouse a otra posición), eliminarlo, así como asociarle un valor distinto de propiedad o profundidad.

- Salida:
  La colección de puntos dispersos que representan una discretización o digitalización de algunos puntos de la capa o superficie de estudio.

Requerimiento funcional 7: Construcción de un mapa de datos por medio de una malla rectangular o rejilla (*mesh*)

- Resumen:
  Otra forma de contar con una discretización de la capa o superficie es a través de una colección de datos distribuidos a lo largo de una malla rectangular sobre la región de trabajo, o parte de ella. Es necesario contar con una herramienta de edición que nos permita definir sobre pantalla una rejilla o malla rectangular, que nos permita posteriormente editarla, en tamaño, posición y distribución. Una vez definida la rejilla, podemos sobre cada punto asociarle el valor de la de la profundidad o propiedad a estudiar.
- Entrada:
  La región de trabajo en pantalla.
- Procedimiento:
  A través de la herramienta gráfica para tal propósito y el *mouse*, deberá introducirse la rejilla sobre la región de trabajo (o parte de ella) y con las herramientas gráficas para tal efecto, poder eliminar líneas verticales u horizontales, así como poder añadir líneas. Una vez definida la rejilla, deberemos asignarle el valor de propiedad o profundidad asociado a cada nodo de la misma.
- Salida:
  La colección de puntos sobre la rejilla que representan una discretización o digitalización de algunos puntos de la capa o superficie de estudio.

Requerimiento funcional 8: Lectura de un mapa de datos digitalizados

- Resumen:
  Como se mencionó, la información de cada capa a través de datos digitalizados puede lograrse a través de un mapa de contornos, y también a través de un mapa de datos digitalizados; los cuales pueden estar distribuidos bajo algún patrón de esparcimiento (una malla gruesa, o mesh) o bien a través de una colección de datos dispersos.
- Entrada:
  Un archivo de datos en formato convenido.
- Procedimiento:
  El archivo de datos puede presentar la colección de nodos a través de una colección de datos dispersos, o bien siguiendo un patrón de esparcimiento a partir de una malla rectangular o rejilla sobre la región de trabajo.
- Salida:
  La colección de datos en pantalla. Sin son dispersos, estos se despliegan aislados, si provienen de una malla gruesa, es conveniente que sea desplegada esta en pantalla.

Requerimiento funcional 9: Reconstrucción de una capa o superficie a partir de un mapa de contornos o de una colección de datos

- Resumen:
  Contando con una colección de datos provenientes de un mapa de contornos o datos dispersos o sobre una rejilla, la tarea es reconstruir la capa o superficie de la cual provienen los datos.
- Entrada:
  El mapa de contornos o colección de datos sobre la región de estudio.
- Procedimiento:
  Existen varias técnicas para reconstruir una superficie a partir de una colección de datos. La primera que aquí experimentaremos se trata de la interpolación bivariada, y la cual, construye un interpolante que de manera local suaviza los datos. En esta primera fase se hará uso del algoritmo 526 de la colección ACM.
- Salida:
  En pantalla se podrá contar con una rejilla fina o gruesa sobre la región de trabajo de los datos interpolados. Se deberá tener la opción de desplegar una colección de curvas de nivel o isolíneas de propiedades ya que estas nos proveen de una mejor información de la forma de la capa o superficie.

Requerimiento funcional 10: Construcción de la región plana de estudio

- Resumen:
  La malla 3D de simulación se construye a partir de una malla plana 2D sobre la región de trabajo y esta es mapeada o levantada sobre cada capa o superficie. La región plana estará conformada por una región poligonal, por lo que la herramienta de edición de contorno y aquella donde se define el mapa de contornos es útil en esta situación.
- Entrada:
  Se cuenta con la región de trabajo en pantalla, sobre la cual será definida la región plana de estudio.
- Procesamiento:
  La región plana de estudio estará conformada por una región poligonal, la cual es introducida por medio del mouse y sobre el espacio de trabajo.
- Salida:
  Al finalizar se tendrá una región plana en pantalla conformada por un polígono.

Requerimiento funcional 11: Definición de los segmentos de frontera para construir una malla sobre la región plana de estudio

- Resumen:
  La malla plana sobre la región de estudio será una malla estructurada conformada por cuadriláteros. Esta malla puede obtenerse a través de la técnica del mapeo, donde es necesario definir los cuatro segmentos de frontera: abajo, derecha, arriba, e izquierda. Con esto, la malla estructurada se obtiene uniendo de manera adecuada, segmentos de frontera opuestos.
- Entrada:
  Un contorno previamente definido, sea en pantalla o por fichero. Si las fronteras se encuentran ya asignadas, podemos modificarlas.
- Procesamiento:
  El usuario determinará los cuatro segmentos de frontera definiendo los cuatro vértices que dará origen a la intersección de fronteras consecutivas. El usuario

seguirá contando con herramientas de edición del contorno, de tal manera que pueda modificar la región plana de estudio.

- Salida:
  Al finalizar se contará con una región poligonal donde no haya pico hacia adentro en las esquinas y en principio pueda obtenerse una malla convexa; es decir, una malla sin líneas curvilíneas dobladas.

Requerimiento funcional 12: Guardar información del contorno de la región plana de estudio

- Resumen:
  Una vez creado el contorno de la región plana de estudio, es útil contar con la información en disco bajo formato convenido.
- Entrada:
  La información del contorno en memoria y el nombre del archivo donde se guardarán los datos.
- Procedimiento:
  Al elegir la opción de salvar la información debe ser desplegada en pantalla una caja de diálogo, un *browser*, donde se podrá señalar la ubicación donde quedará el archivo y su nombre.
- Salida:
  El archivo del contorno en disco.

Requerimiento funcional 13: Generación de la malla plana inicial sobre la región de estudio

- Resumen:
  Una vez que se cuenta con un contorno aceptable, podemos generar una malla por interpolación que nos permita usarla como punto de partida o inicial para el procedimiento de optimización involucrado en la generación de la malla o los métodos discretos.
  El método TFI de interpolación que comúnmente usamos en mallas planas, conserva la forma de la frontera de la región poligonal y propaga las singularidades de la frontera hacia el interior de la región.
- Entrada:
  Un contorno aceptable en la asignación de frontera. La dimensión de la malla.
- Procesamiento:
  A través de campo de entrada se introducirá la dimensión de la malla y al pulsar la opción diseñada se construirá la malla por el método de Interpolación Transfinita TFI, que básicamente une por segmentos de línea lados opuestos.
- Salida:
  La malla plana inicial, siempre que la elección de dimensión de la malla es "adecuada'" para lograr una malla que conserve la forma de la región poligonal sin que el contorno deje de ser aceptable. En caso contrario no podremos trabajar con el resultado. El concepto de aceptabilidad de un contorno viene definido en el entregable 1 de la primera fase del reporte de actividades del GGNM.

Requerimiento funcional 14: Lectura de la malla plana sobre la región de estudio

- Resumen:
  Necesitamos trabajar con mallas previamente generadas o bien continuar el trabajo que realizamos sobre ellas. Para esto, es indispensable poder llevar a la memoria una malla en disco.
- Entrada:
  Un archivo de mallas en formato convenido.
- Procedimiento:
  A través de una caja de diálogo el usuario proveerá el nombre del archivo o bien mediante el cursor y las opciones del *browser*, navegará por los directorios del sistema hasta señalar el archivo deseado.
- Salida:
  Al elegir el archivo de mallas los datos se cargaran en memoria y serán desplegados en pantalla.

Requerimiento funcional 15: Herramientas de despliegue gráfico de la malla plana de estudio

- Resumen:
  Una herramienta que ha hecho avanzar mucho el estudio posterior de los métodos discretos y la obtención de los resultados, ha sido el desplegar la malla conforme se va obteniendo en pantalla, observarla en puntos discretos y observar cómo va esta evolucionando hasta su convergencia, así como para el análisis de los resultados.
- Entrada:
  Una malla en memoria.
- Procedimiento:
  Con la malla en memoria mediante el uso de controles deslizadores y botones de opción se cambiará la caja de despliegue para poder hacer un *zoom* hacia adentro o hacia afuera y de un lado a otro. De igual forma la caja de visualización podrá mover con el fin de observar parte de la malla.
- Salida:
  Despliegue en pantalla.

Requerimiento funcional 16: Optimización de la malla plana de estudio

- Resumen:
  La optimización de una malla va en el sentido de lograr propiedades óptimas acorde a los funcionales discretos, esto es, que la malla obtenida esté cercana a las propiedades del funcional. Aquí se deberá optimizar el funcional de área-ortogonalidad para la obtención de mallas suaves y convexas.
- Entrada:
  Una malla en memoria.
- Procedimiento:
  A través de una serie de opciones y menús el usuario podrá elegir la opción que desee o bien usar las predeterminadas. Al ejecutar la opción de realizar el procedimiento deberá aparecer en pantalla una ventana donde se muestre el desarrollo de las iteraciones así como en pantalla podrá observarse gráficamente cómo evoluciona la malla hacia el óptimo.
- Salida:

Una malla optimizada.

Requerimiento funcional 17: Guardar información de la malla plana de estudio a disco

- Resumen:
  Una vez obtenida la malla es necesario guardar la información en disco para su posterior uso.
- Entrada:
  Una malla en memoria y el nombre de archivo donde se guardarán los datos.
- Procedimiento:
  Al igual que en el procedimiento de lectura podemos a través de un *browser*, depositar la información de la malla en el directorio deseado así como indicar el tipo de formato en que será guardada la información.
- Salida:
  La malla en archivo de disco.

Requerimiento funcional 18: Refinamiento uniforme

- Resumen:
  El refinamiento de la malla plana de estudio, es una herramienta útil en muchas aplicaciones, y en la construcción de la malla 3D de simulación, será importante que la malla plana sobre la región de estudio 2D pueda ser refinada localmente.
  Contamos con una malla sobre la región en algunas aplicaciones deseamos hacer algún cálculo con precisión en subregiones o partes de la región plana de estudio.
- Entrada:
  Una malla plana convexa en pantalla.
- Procedimiento:
  Por refinamiento uniforme de la malla plana de estudio, entenderemos una interpolación entre las líneas originales. En la opción de refinamiento uniforme deberemos elegir el número de líneas que serán interpoladas entre dos ya existentes y realizar entonces la interpolación.
  De manera alternativa, la interpolación uniforme puede ser llevada a cabo de manera manual, al indicar a través de herramientas gráficas de edición las líneas verticales a añadir así como las líneas horizontales, y esta tarea repetirse el número de veces necesarias.
- Salida:
  La malla en pantalla con el refinamiento uniforme deseado.

D.7. Requerimientos de Interfase externa

User Interfase

La interfase con el usuario debe ser amigable, debe usar recursos de interfase comunes como mouse, teclado y contar con la posibilidad de usar alguno de ellos cuando el otro no exista. La idea es contar con un sistema totalmente gráfico por lo que la interfase debe ser "natural" al usuario en el manejo de las opciones. Conectar algunas tareas con otras, de manera que el proceso sobre el mapa de contornos, la reconstrucción de las capas o superficies y la malla 2D sobre la región de estudio se encuentren plenamente identificadas, para así poder obtener de manera directa la malla 3D de simulación.

Hardware Interface

El sistema no deberá depender de ningún hardware. No deberá estar orientado a plataforma de sistemas específicos, por el contrario ser lo más amplio posible. No dependerá del despliegue gráfico de los *pixels* de la pantalla gráfica o de especificaciones de tarjeta gráfica alguna, en todo caso, deberá hacerse uso de bibliotecas libres para el despliegue gráfico.

Software Interface

Para el despliegue gráfico se usarán primitivas basadas en OpenGL de manera que sea lo más independiente de cualquier otra biblioteca gráfica. Sólo deberán emplearse rutinas de distribución libre y en su caso bibliotecas gráficas soportadas en casi cualquier sistema.

Requerimientos no funcionales

El sistema debe tener la capacidad de ser llamado desde una PC con Windows 2000 o superior y sin límite de usuarios, excepto quizá las limitaciones de recursos en la memoria que el sistema permita.
El sistema debe ser rápido en ejecución y todo en pantalla. Una buena colección de mensajes desplegados apropiadamente ayudará a que los usuarios puedan saber las razones por las que no es posible realizar una operación o bien si ésta fue llevada a cabo de manera satisfactoria.

El sistema debe contar con los siguientes atributos:

- Disponibilidad
  En todo momento el usuario podrá ejecutar el sistema siempre que se encuentre trabajando en el sistema de ventanas de la computadora.
- Mantenimiento
  El mantenimiento del sistema jugará un papel importante en futuros desarrollos. Contemplar que el sistema cuente con una infraestructura que permita actualizar, sustituir o eliminar sus partes, permitirá el crecimiento del mismo con un costo mínimo. Si la construcción el sistema se encuentra muy bien documentada en sus partes permitirá, que un mayor número de personas puedan participar en futuros desarrollos o aplicaciones del mismo.

Apéndice E: Generación Numérica de Mallas Planas usando el Funcional Discreto de Área-Ortogonalidad usando Optimización Puntual

E.1. Planteamiento del Problema.

La generación de mallas es un proceso que consiste en partir un dominio físico en subdominios elementales para observar algún fenómeno medible sobre cada subdominio. Una malla estructurada es aquélla bajo la cual podemos identificar de forma rápida los nodos de la malla, interiores y exteriores, y cada nodo interior cuenta con el mismo número de elementos adyacentes. Este tipo de mallas son muy socorridas en aplicaciones de EDP, pero presentan dificultades en la aproximación sobre la frontera cuando el dominio no es simple.

Existen muchos métodos para obtener este tipo de mallas estructuradas y uno de ellos es el método del mapeo, que consiste en obtener un mapeo que transforme el dominio físico en uno más sencillo y entonces lograr de forma directa una malla, ver Figura E.1.

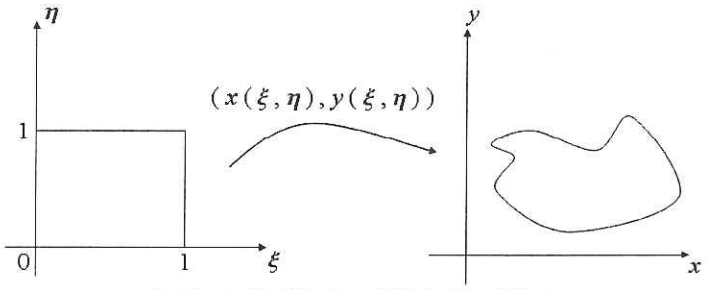


Figura E.1: Mapeo entre dos regiones.

E.2. Método que se usa

Contando con un mapeo continuo entre fronteras nos lleva a considerar la frontera de la región en cuatro segmentos curvilíneos o subfronteras. De tal modo que la idea gráfica de la generación de una malla es unir las fronteras 1-2 con la 4-3 y la 1-4 con la 2-3 (ver Figura (E.2), por medio de segmentos curvilíneos de tal forma que no se intersecten. La idea es trazar líneas entre segmentos opuestos, como se observa en la Figura (E.3).

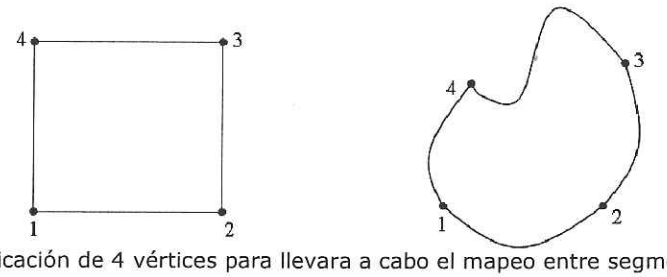


Figura E.2: Identificación de 4 vértices para llevara a cabo el mapeo entre segmentos de frontera.

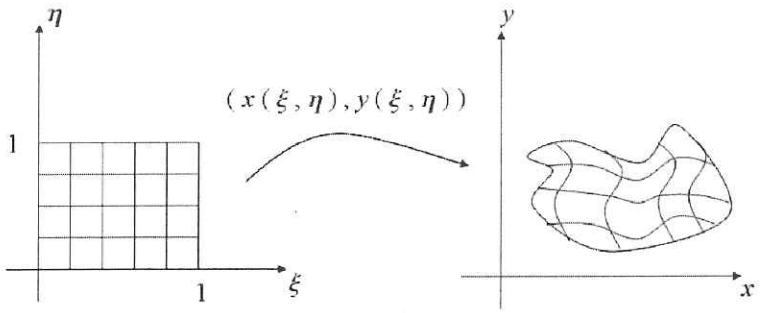


Figura E.3: Una malla a partir de un mapeo entre regiones.

En muchas aplicaciones la región de trabajo es solución de un problema diferencial o algebraico; en otras es posible contar con la descripción de la región en forma continua a través de una parametrización del contorno. En nuestro caso, disponemos de puntos sobre el contorno obtenidos de alguna observación.

E.3. Funcional de Área-Ortogonalidad.

Nuestro interés en el sistema `Driv_Gen_malla_p` es principalmente obtener mallas con la propiedad de Área y de Ortogonalidad, es por ello que daremos una breve explicación de la combinación de estas, a través del funcional discreto de Área-Ortogonalidad.

En las siguientes definiciones, se precisará lo que entendemos por una malla estructurada y por un funcional discreto.

Consideremos una región Ω del plano definida por una poligonal γ con vértices $V = \{v\_1, v\_2, ..., v\_q\}$, cerrada, simple y orientada en sentido positivo.

Definición 1. *Sean m y n números naturales mayores que* 2. *Decimos que el conjunto de puntos del plano*

$$G = \{P_{i,j} \mid i = 1,2,...,m;\ j = 1,2,...,n\}$$

de lados

$$L_1(G) = \{P_{i,1} \mid i = 1,...,m\}$$
$$L_2(G) = \{P_{m,j} \mid j = 1,...,n\}$$
$$L_3(G) = \{P_{i,n} \mid i = 1,...,m\}$$
$$L_4(G) = \{P_{1,j} \mid j = 1,...,n\}$$

es una malla estructurada admisible y discreta de orden $m \times n$ para Ω, *si se satisface que*

$$V \subseteq \cup_{i=1}^{4} L_i(G)$$

Decimos además que la malla G es convexa si cada uno de los $(m-1)(n-1)$ *cuadriláteros (o celdas)* $c_{i,j}$ *de vértices* $P_{i,j}, P_{i+1,j}, P_{i+1,j+1}, P_{i,j+1}$, *con* $i < m$ *y* $1 \le j < n$ *no es convexo.*

Definición 2. *Un funcional discreto I sobre una malla* $G = P_{i,j}$ *es una función*

$$I(G) = \Sigma_{i,j} f(c_{i,j})$$

donde $c_{i,j}$ *es la celda i,j de la malla y*

$$f(c_{i,j}) = f(P_{i,j}, P_{i+1,j}, P_{i+1,j+1}, P_{i,j+1})$$

es una función de sus vértices.

Para escribir la forma que tienen los funcionales discretos, denotemos los triángulos en la celda i, j con vértices $P_{i,j}, Q_{i,j}, R_{i,j}, S_{i,j}$ como

$$\begin{aligned}
\Delta S_{i,j}, P_{i,j}, Q_{i,j} &= \Delta^1_{i,j} \\
\Delta Q_{i,j}, R_{i,j}, S_{i,j} &= \Delta^2_{i,j} \\
\Delta P_{i,j}, Q_{i,j}, R_{i,j} &= \Delta^3_{i,j} \\
\Delta R_{i,j}, S_{i,j}, P_{i,j} &= \Delta^4_{i,j}
\end{aligned}$$

de tal manera que el funcional discreto sobre una malla *G* sea

$$I(G) = \Sigma_{i=1}^{m-1} \Sigma_{j=1}^{n-1} \Sigma_{k=1}^{4} f(\Delta^k_{i,j})$$

Así, el funcional de suavidad F_S , el funcional de longitud F_L , el funcional de área F_A y el funcional de ortogonalidad F_O están dados como

$$\begin{aligned}
F_S &= \sum_{i=1}^{m-1} \sum_{j=1}^{n-1} \sum_{k=1}^{4} \frac{\lambda(\Delta^k_{i,j})}{\alpha(\Delta^k_{i,j})} \\
F_L &= \sum_{i=1}^{m-1} \sum_{j=1}^{n-1} \sum_{k=1}^{4} \lambda(\Delta^k_{i,j}) \\
F_A &= \sum_{i=1}^{m-1} \sum_{j=1}^{n-1} \sum_{k=1}^{4} (\alpha(\Delta^k_{i,j}))^2 \\
F_O &= \sum_{i=1}^{m-1} \sum_{j=1}^{n-1} \sum_{k=1}^{4} (o(\Delta^k_{i,j}))^2
\end{aligned} \qquad \text{(E.1.)}$$

Donde α , λ y *o* son los funcionales discretos de área, longitud y ortogonalidad en los triángulos.

Para combinar los efectos de varios funcionales en una malla, es natural considerar combinaciones lineales entre ellos,

$$F(x) = \sigma_1 F_1(x) + \sigma_2 F_2(x) + \sigma_3 F_3(x)$$

donde $\sigma_1, \sigma_2, \sigma_3 \geq 0$ y $\sigma_1 + \sigma_2 + \sigma_3 = 1$.

La elección de los "pesos" σ_i y los funcionales f_i está motivada por las propiedades que se desea aparezcan en la malla.

Una de las combinaciones más útiles es el funcional de área-ortogonalidad, para el cual $\sigma_1 = \sigma_2 = \frac{1}{2}$ y $F_1(x) = F_A(x), F_2(x) = F_O(x)$, donde $F_A(x)$ y $F_O(x)$ representan a los funcionales de área y ortogonalidad. Esta combinación es

$$F_{AO} = \frac{F_A + F_O}{2} \qquad \text{(E.2.)}$$

El funcional de Área-Ortogonalidad produce mallas muy suaves y con pocas celdas no convexas, pero no mallas convexas en general.

Para finalizar, hemos de enfatizar que *no necesariamente* debemos obtener el óptimo del problema, lo que se requiere es generar una malla *convexa* que satisfaga la propiedad geométrica de ortogonalidad.

E.4. Tarea del programa

El sistema `Driv_Gen_mall_p` es un procedimiento que genera mallas con la propiedad de Área Ortogonalidad sobre una región Ω plana irregular, dada a partir de su frontera. A partir de una malla inicial de una región Ω, la malla solución se obtiene optimizando puntualmente la malla inicial usando un método de Newton con Región de Confianza, que opera sobre un nodo de la malla cada vez. Esto da la posibilidad de poder mantener puntos fijos en determinados lugares de la región, pues no se hace el proceso de optimización sobre ellos, lo que resulta útil en la práctica para las zonas donde existen pozos o fallas. La optimización del funcional de Area-ortogonalidad se dice entonces que es punto a punto, exceptuando los que previamente se marcaron como fijos.

E.5. Descripción breve de los módulos

`Driv_Gen_mall_p`. Programa principal que lee las dimensiones de la malla y prepara las dimensiones totales requeridas por el procedimiento de obtención de una malla suave y convexa. También se lee el nombre del fichero que contiene los puntos que se van a mantener fijos, en el siguiente formato:

nfix - Cantidad de puntos fijos
i,j,x,y – Indices i,j del punto fijo y sus coordenadas en la región física, es decir,
P(i,j) =(x,y)

Módulo `MENUOP_P`. Este módulo coordina la generación de una malla óptima. Para ello llama a las rutinas `FILEIO_PS`, `ADMRED`, `SCALE`, `CAREA`, `NONIJ` y `SOLVER_P`. Ver Figura (F.4).

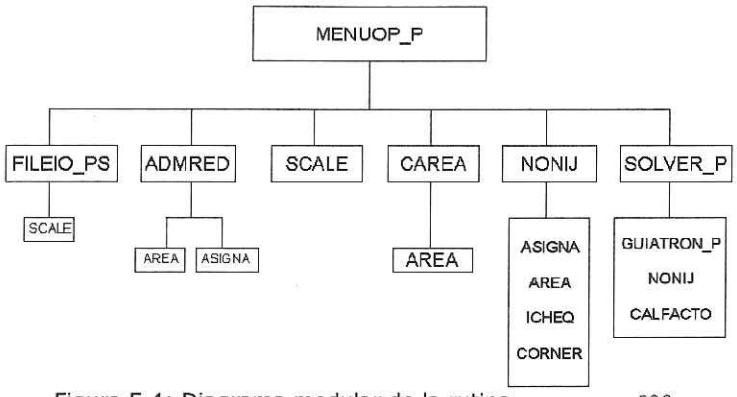


Figura E.4: Diagrama modular de la rutina `menuop_p.f90`.

Módulo `FILEIO_PS`. Lee desde un archivo los puntos de la malla inicial o escribe a un archivo los puntos de la malla final, de acuerdo a una bandera de entrada. Llama a la rutina `SCALE`.

Módulo `NONIJ`. Verifica la convexidad de una malla a través de `ASIGNA`, `AREA` y `ICHEQ`.

Módulo `ASIGNA`. Dados los índices i,j de un punto P(i,j), este modulo devuelve el apuntador a la posición de dicho punto en el arreglo unidimensional RED, así como las coordenadas (x,y) del punto en P(1)=x y P(2)=y

Módulo `AREA`. Función que calcula el área de un triángulo teniendo en cuenta la orientación positiva (en contra de las manecillas del reloj).

Módulo `ICHEQ`. Verifica la convexidad de las celdas que no están en las esquinas de la región.

Módulo `ADMRED`. Calcula el área de las cuatro celdas que están en las esquinas de la región para checar su admisibilidad.

Módulo `SCALE`. Transforma los valores de la malla inicial a la región [0,1]x[0,1] y/o los reestablece a su escala física.

Módulo `SOLVER_P`. Prepara la entrada a la rutina de optimización que resuelve el problema de minimización proveniente de la generación de mallas variacional. Asigna las tolerancias para los diferentes criterios de parada y llama a la rutina `GUIATRON_P`.

Módulo `GUIATRON_P`. Este módulo coordina la ejecución del método de Newton con Región de Confianza, la que se hace punto por punto, es decir, se busca el óptimo del funcional sobre toda la malla, pero sólo se considera variable P(i,j) y el resto de los puntos no se mueven, con lo que la dimensión del proceso es de orden 2, lo que implica un ahorro considerable de memoria y la posibilidad de poder obtener mallas muy finas. Se continúa optimizando de esta forma todos los puntos de la malla, excepto aquellos que el usuario asumió que estarían siempre fijos en el valor de las coordenadas que se dieron de entrada.

Módulo CALFACTO. Función que calcula el factor de normalización del funcional de Area-ortogonalidad.

E.6. Bibliografía

[1] Barrera1994, Barrera, P., Castellanos, L., y Pérez, A. 1994. *Métodos Variacionales Discretos para la Generación de Mallas*, DGAPA-UNAM, México.

[2] Barrera1994b, Barrera, P., González, G., Pérez, A. y Castellanos, L. 1994. *Manual de Usuarios del Sistema UNAMALLA v. 1.0: Generación de Mallas Planas sobre Regiones Irregulares* DGAPA-UNAM, México.

[3] Barrera1997, Barrera, P. and Tinoco J.G. 1997. *Smooth and Convex Grid Generation over General Plane regions* in Mathematics and Computer in Simulation.

[4] Barrera2000, Barrera, P., García, I. y González, G. 2000. *Manual Operativo del Sistema UNAMALLA v. 2.0 para PC*, Cuadernos de Investigación, **21** Área I, Física-Matemáticas e Ingeniería, Universidad Autónoma de Coahuila, México.

[5] Mota, Domínguez-Mota, F.J. 2005. *Sobre la Generación Variacional Discreta de Mallas Casiortogonales en el Plano*, Tesis de Doctorado, UNAM, México.

[6] González1994, González Flores, G.F., 1994. *Generación de Mallas en Regiones Planas Irregulares*. Tesis de Licenciatura. Universidad Autónoma de Yucatán, Yucatán, Méx. 1994.

[7] Knupp1993, Knupp, P. and Steinberg, S. 1993. *Fundamentals of Grid Generation*. CRC Press, Inc.

[8] Steinberg1986, Steinberg, S., and Roache, P.J., 1986. *Variational Grid Generation, Num. Meth. for P.D.E.s.*, **2**, 71--96.

[9] Tinoco1997, Tinoco, J.G. 1997. *Funcionales Discretos para la Generación de Mallas Suaves y Convexas sobre Regiones Planas Irregulares*, Tesis de Doctorado, CIMAT, México.

Apéndice F: Algunos aspectos sobre la Visualización de una Malla 3D Estructurada para la Simulación de un Yacimiento de Hidrocarburos

F.1. Introducción

Los objetivos principales de la ingeniería petrolera son, básicamente, la búsqueda de yacimientos, su explotación, la transformación de hidrocarburos en productos derivados y el estudio de esos productos tales como la gasolina, en motores.

El caso que nos compete es la simulación de la explotación de un yacimiento de hidrocarburos, el cual se define como una formación de roca porosa que contiene fundamentalmente, agua, aceite y gas. Existen varias fases dentro de la explotación de un yacimiento de hidrocarburos. En la primera un pozo o más son perforados extrayendo el aceite hasta que la presión disminuya y el agua comience a aparecer. En la segunda fase, se inyecta agua o gas para desplazar a los fluidos, ésta es la que se conoce como recuperación secundaria o técnica mejorada de recuperación. En ocasiones se considera una tercera fase, donde se hace uso de procesos químicos o térmicos. La inyección de químicos y el incremento de la temperatura puede repercutir en un importante cambio de las propiedades del fluido.

Lo importante en estas dos últimas fases, es que la calidad del aceite recuperado depende de las propiedades de la formación rocosa, tales como porosidad y permeabilidad, las cuales miden la capacidad del fluido de atravesar los espacios porosos. En este sentido, la producción de aceite de los yacimientos, conlleva al cálculo de parámetros escalares tales como el tipo de roca, la presión, la temperatura, la saturación del aceite, la saturación del agua, etc.

La forma tradicional de representar un yacimiento es a través de una malla 3D que proporciona una discretización del mismo, la cual permite llevar a cabo algunas mediciones y simulaciones numéricas sobre la calidad y el volumen de aceite a producir a lo largo del tiempo.

F.2. Representación Numérica y Gráfica del Yacimiento

La simulación de un yacimiento de hidrocarburos, contempla una serie de tareas elementales para poder describir de manera gráfica el comportamiento de los fluidos a su paso a través la roca. A continuación se enumeran una serie de tareas que debe contemplar la representación numérica y gráfica de un yacimiento.

- Realizar la visualización del modelo 3D de una malla regular que represente la simulación numérica de un yacimiento de hidrocarburos.
- Describir la malla mediante celdas, donde cada celda tiene asignadas algunas propiedades tales como: porosidad, permeabilidad, etc.
- Representar los cambios de valores en las propiedades mediante un degradado de colores. Básicamente se requiere de una interpolación sobre los valores de los nodos.
- Visualizar las celdas internas de la malla.
- Proveer la visualización de cualquiera de las 6 caras de la malla.

- Limitar la visualización sólo a las celdas activas en la simulación.
- Colocar pozos dentro de la malla, éstos pueden variar de simulación en simulación y se puede modificar su localización en tiempo real.

F.3. Visualización de la malla 3D por medio de Superficies

Las propiedades a observar sobre el yacimiento son medidas escalares asociadas a cada elemento de la celda 3D (hexaedro), o sobre la celda 2D de cada corte de la malla. Para poder usar esa información, es conveniente que tanto los nodos de la malla como sus valores asociados sean almacenados en la misma forma. En una primera instancia, tenemos

- m es el número de puntos (el número de nodos de la malla 3D).
- Se requieren tres arreglos de tamaño m, a saber, X,Y, y Z para almacenar las coordenadas de los puntos de la malla.
- Se requiere para cada propiedad: porosidad, permeabilidad, etc., un vector para almacenar la información de cada nodo.

Si consideramos que esos vectores, de posición y propiedades, son arreglos tridimensionales, tendremos una manera directa de obtener o asignar información al ordenar los nodos de la malla 3D de la misma forma en que se tiene un orden en los arreglos tridimensionales. De esta manera, podremos identificar plenamente los nodos vecinos de uno dado y calcular de manera eficiente alguna propiedad geométrica de la malla 3D.

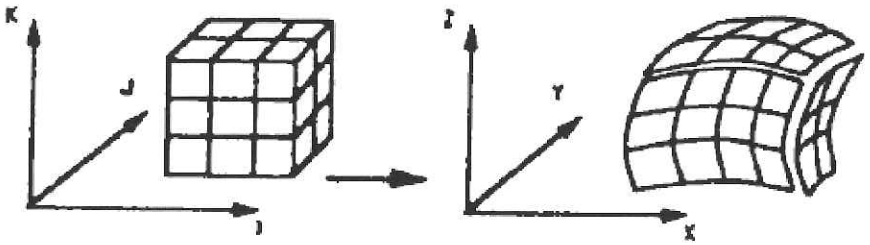


Figura F.1: Malla de referencia y malla de simulación

Siguiendo esta idea tendremos un mapeo directo de la malla 3D de referencia a la malla 3D de simulación dada por

$$\begin{pmatrix} i \\ j \\ k \end{pmatrix} \mapsto \begin{pmatrix} x(i,j,k) \\ y(i,j,k) \\ z(i,j,k) \end{pmatrix}$$

Esto nos permite movernos de un punto P_1 sobre la malla 3D de simulación a un punto P_2 de la misma en tres formas distintas, siguiendo los tres índices i,j,k. Más aún, sobre cualquiera de esas direcciones podemos llevar a cabo una interpolación de los valores

asociados a los nodos y con ello trazar planos u objetos que nos permitan entender el efecto de la simulación sobre el yacimiento. Los puntos pueden representar pozos extractores o inyectores.

Si sobre la malla de referencia, fijamos dos de los índices, digamos j,k, moverse sobre el índice i corresponde a describir una curva sobre la malla 3D de simulación, y al fijar un solo índice, se puede entonces recorrer la malla 3D a través de cortes o superficies.

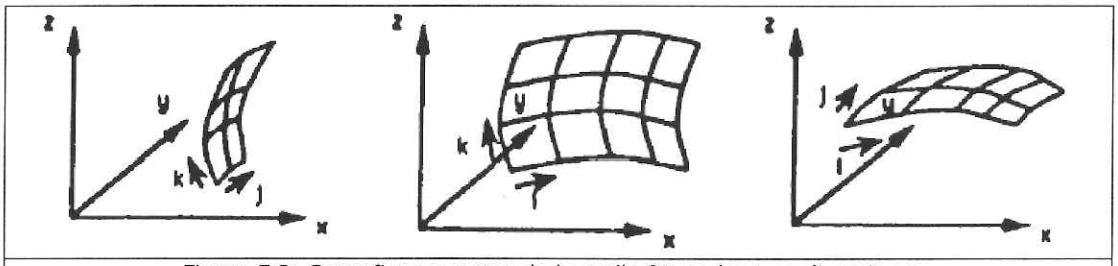


Figura F.2: Superfices o cortes de la malla 3D en las tres direcciones.

Por consiguiente, se podrían considerar 6 rutinas distintas para trazar elementos de la malla 3D. Con las primeras 3 se describiría una curva fijando dos índices y el otro se haría variando en un intervalo apropiado. Las otras 3 rutinas combinarían dos de las ya señaladas y con ello, se trazarían superficies o cortes de la malla al fijar un índice y variar los otros.

Por otra parte, el dominio a visualizar puede ser restringido al definir un subvolumen de la malla 3D original, mediante el uso de subíndices de la malla 3D de referencia como n_{i0},n_{i1}, para el índice i, y lo mismo usando n_{j0},n_{j1} para j y n_{k0},n_{k1} para k.

Las superficies o cortes de la malla, pueden ser graficadas de diferentes formas. Dado que cada curva está compuesta por puntos (curva poligonal), se pueden visualizar como una colección de ellos. Sin embargo por sí solos no ofrecen mucha información, por lo que unir dichos puntos por segmentos poligonales, proporcionaría una idea más clara de su comportamiento dentro de la malla 3D de simulación.

Haciendo uso de estas rutinas, de manera apropiada, se dispondrá de distintas formas de observar la malla 3D de simulación:

- Observarla como una colección de superficies, donde se grafican las líneas curvilíneas al variar apropiadamente dos índices y variar el tercero dependiendo de la colección de superficies o cortes a graficar (Figura F.3).

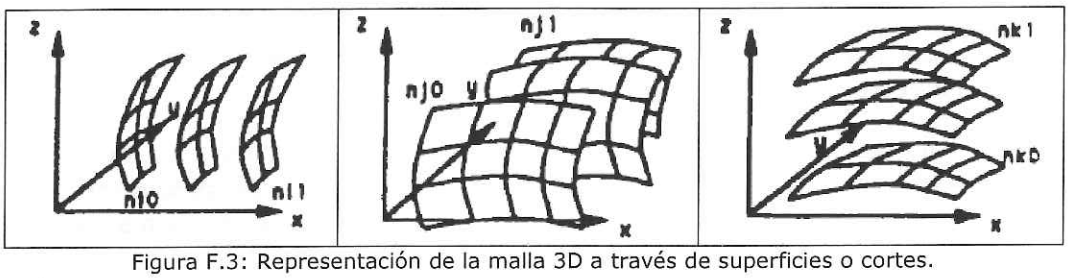


Figura F.3: Representación de la malla 3D a través de superficies o cortes.

- Observar la malla como una colección de tres superficies obtenidas al usar las tres rutinas que grafican una superficie por cada una de las direcciones i, j, k. Las superficies se cruzan Figura F.4).

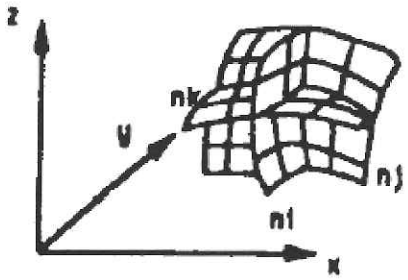


Figura F.4: Representación de la malla 3D a través de superficies o cortes cruzados.

- O bien, observar la malla como una colección de seis superficies, obtenidas al graficar las fronteras del dominio (Figura F.5).

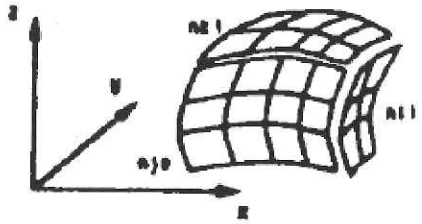


Figura F.5: Representación de la malla 3D a través de la frontera del dominio.

Modificando iterativamente los parámetros $n_{i0}, n_{i1}, n_{j0}, n_{j1}, n_{k0}$ y n_{k1}, el usuario podrá observar cortes de las mallas hacia el interior de esta, podrá seleccionar secciones volumétricas, separar las mallas por cortes cruzados y, en general, podrá seleccionar observar la simulación en forma volumétrica o sobre alguna sección transversal, al interpolar sobre esta sección las propiedades a medir. Naturalmente, este procedimiento puede mezclarse para que según transcurra el tiempo el tiempo y se lleven a cabo transformaciones rígidas sobre la malla 3D, se obtenga este tipo de visualización.

F.4. Uso de la malla 3D por medio de la malla de referencia

La malla 3D de referencia permite localizar los puntos sobre la malla 3D de simulación de manera directa, ya que es mucho más sencillo elegir una colección de puntos o líneas sobre la malla al estar esta definida sobre un espacio de índices.

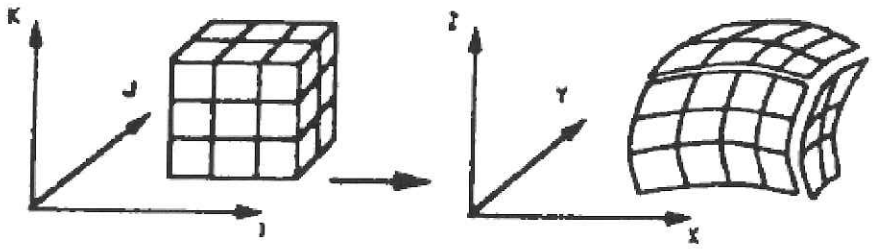


Figura F.6: Malla de referencia y malla de simulación

A partir de esta idea, se pueden asignar propiedades a la malla 3D de simulación de forma directa al trabajar con cada celda de la malla de referencia. Esto resulta útil para el usuario al poder editar las propiedades de la roca, como porosidad o permeabilidad, directamente sobre la malla de referencia y entonces proceder a llevar a cabo una simulación.
Por otra parte, se puede definir un pozo (extractor o inyector) sobre la malla 3D de simulación, al señalar sobre la malla 3D de referencia la posición en índices i, j, k del pozo. En la Figura F.7, se puede observar esta idea.

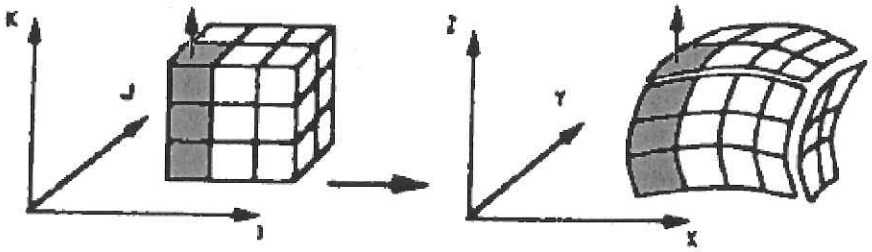


Figura F.7: Asignación de pozos mediante la malla de referencia.

De igual manera es posible asignar celdas activas o inactivas para el proceso de simulación, e irlas activando o desactivando según lo que se desee simular.

Otra utilidad de la malla de referencia está al definir sobre esta, y entonces sobre la malla 3D de simulación, una sección sobre la cual se desee llevar a cabo un refinamiento local o global de la malla para lograr una mejor aproximación del modelo numérico (Figura F.8).

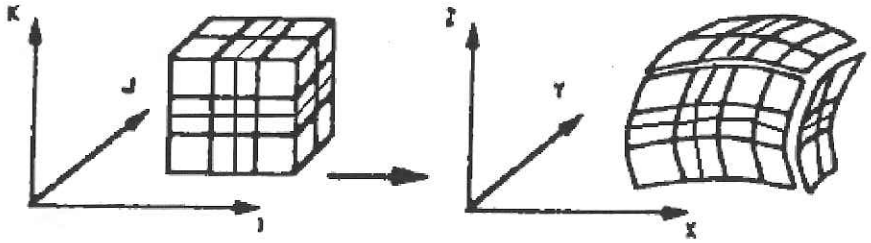


Figura F.8: Refinamiento local de la malla 3D de simulación.

En suma, la malla 3D de referencia nos permite editar de manera eficiente la malla 3D de simulación, nos permite visualizar elementos que la componen, señalar los pozos y refinar local y globalmente la malla al identificar una sección de estudio.

Apéndice G: Graficación de Mapa de Contornos por medio de Soluciones Sucesivas de Ecuaciones Polinomiales de grado 5

G.1. Resumen.

En este reporte describimos la idea básica del algoritmo de Albrecth Preusser [3] para el cálculo de contornos a partir del interpolante de Hiroshi Akima [1], así como el uso del módulo `CONTOUR_GRAF.f90` el cual usa este algoritmo para graficar un conjunto deseado de contornos para alguna superficie proporcionada.

G.2. Introducción.

Es importante en las aplicaciones relacionadas con la reconstrucción de superficies que representan yacimientos petroleros, poder recuperar un conjunto de contornos de una o varias capas tanto para su visualización como para su estudio. Así el objetivo de este módulo es la implementación de una subrutina que permita de forma simple obtener una visualización de un conjunto de contornos dada una superficie.

G.3. Descripción del método.

Muchos de los métodos de interpolación de puntos dispersos en el espacio desarrollados actualmente, están relacionados con la teoría del elemento finito, uno de éstos, es el esquema de interpolación basado en polinomios bivariados de grado 5 definido en cada triángulo cuyos vértices están formados por las proyecciones de los puntos en el plano.

El método desarrollado por Hiroshi Akima[1] pertenece a este tipo de técnicas y es uno de los procedimientos que hemos utilizado en este proyecto para la reconstrucción de yacimientos petroleros dado un conjunto de contornos.

La idea general para el cálculo de contornos basado en este tipo de interpolantes se puede resumir en los siguientes pasos:

1. Formar triángulos en el plano *xy* usando las proyecciones de los puntos 3d irregularmente distribuidos.
2. Ordenar una malla de puntos con respecto a los triángulos que los contiene.
3. Estimar las derivadas parciales sobre todos los puntos usando los valores de sus vecinos.
4. Cálculo de los coeficientes de un polinomio bivariado para cada triángulo.
5. Evaluación del polinomio en la malla en puntos dentro del triángulo.
6. Interpolación lineal entre los puntos de la malla y formación del contorno.

La técnica de recuperación de contornos propuesto por Preusser realiza los pasos 4-6 utilizando el interpolante de Akima, por lo que el paso 2 ya no es necesario y los pasos 1 y 3 se resuelven usando el algoritmo 526 [2] también de Akima.

Así el algoritmo en cada triángulo realiza los siguientes pasos:

1. Cálculo de los coeficientes del polinomio de orden cinco de la función de interpolación a través de los tres lados del triángulo. Es decir los valores q_{jk} del polinomio:

$$z(x,y) = \Sigma^{5}_{j=0} \Sigma^{5-j}_{k=0} q_{jk} x^j y^j$$

2. Cálculo del valor en los tres lados del triángulo en donde la primera derivada de los respectivos polinomios es igual a cero.

3. Cálculo de los ceros de funciones las cuales son definidas como la diferencia entre los niveles de los contornos y los polinomios. Los puntos encontrados en el inciso 2 son usados como puntos de inicio para las iteraciones. Es decir si c es el nivel del contorno. Entonces para un punto (x,y) en el contorno debe suceder que :

$$f(x,y) - c = 0 \qquad \text{(G.1)}$$

4. Si existen los ceros de las funciones la representación del polinomio de Akima dentro del triángulo es calculado.

5. Sucesivos cálculos de los puntos de un contorno inician y finalizan con los puntos del paso 3. Los cuales como se mencionó deben cumplir la igualdad (G.1), sin embargo la forma analítica de esta función no es determinada, en vez de eso los valores son tomados del polinomio bivariado del inciso 4.

G.4. Resultados Numéricos

El módulo `CONTOUR_GRAF.f90` permite graficar los puntos de un número de contornos deseados sobre una superficie obtenida usando el método descrito anteriormente (el cual está implementado en la subrutina `626.f90` [5]), al recibir un archivo en formato "Mesh" generado por el módulo `Interpbivar.f90`, el que es una interfase del método de interpolación de Akima. Adicionalmente `CONTOUR_GRAF.f90` guarda los puntos de los contornos que graficó en un archivo con nombre proporcionado por el usuario.

Así en términos esquemáticos `CONTOUR_GRAF.f90`, junto con `interpbivar.f90` se comunican de la siguiente manera.

[5] Que se puede obtener libremente en la página de Internet www.netlib.org.

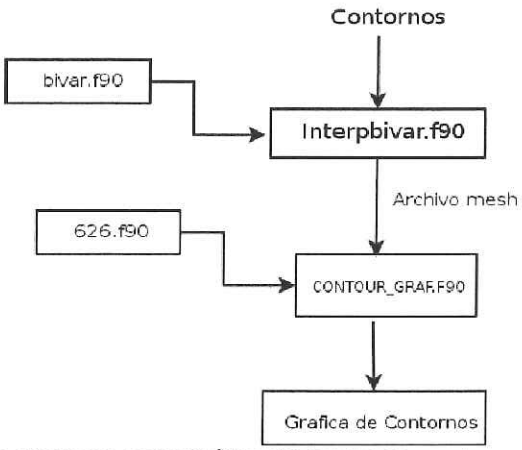


Figura G.1: Diagrama esquemático del programa `contour_graf.f90`.

El modo de funcionamiento de `CONTOUR_GRAF.f90` es simple, a continuación mostramos un ejemplo de su ejecución

```
Nombre del archivo de contornos
cimascuadradas.mesh
Numero de contornos deseado
20
2938.902
2991.944
3044.986
3098.028
3151.070
3204.112
3257.154
3310.196
3363.238
3416.280
3469.322
3522.364
3575.406
3628.448
3681.490
3734.532
3787.574
3840.616
3893.658
3946.700
Proporciona el nombre del archivo de salida
contornos.txt
```

En este ejemplo el programa `CONTOUR_GRAF.f90` pide un archivo en formato "mesh", que representa la malla de una superficie reconstruida con el método de interpolación bivariada de Akima (ver figuras G.2b y G.2c) , a partir de un conjunto de contornos proporcionados (ver Figura G.2a) , una vez leído, se pedirá que se proporcione el número de niveles que se desea recuperar de esta superficie, en ese momento el

programa imprimirá los niveles que calculará de acuerdo a los mínimos y máximos de los datos y pedirá que se proporcione el nombre de un archivo de salida en donde se guardarán los puntos de los contornos y finalmente se realiza una impresión en la pantalla de éstos (ver Figura G.2d).

El archivo de salida tiene el siguiente formato

Coordenadas X	Coordenadas Y	Nivel
.	.	.
.	.	.
.	.	.

Por ejemplo el archivo `contornos.txt` para el ejemplo anterior se ve así

```
7.02274         913.091         3787.57
6.89031         913.241         3787.57
6.62468         913.540         3787.57
6.62468         913.540         3787.57
6.62468         913.540         3787.57
6.62468         913.540         3787.57
58.7775         861.879         3787.57
58.6611         862.041         3787.57
58.4249         862.364         3787.57
58.1868         862.686         3787.57
57.7019         863.322         3787.57
57.2055         863.949         3787.57
56.6979         864.568         3787.57
56.1813         865.179         3787.57
55.6563         865.782         3787.57
55.1220         866.378         3787.57
54.5797         866.966         3787.57
54.0292         867.546         3787.57
53.4732         868.122         3787.57
52.9094         868.689         3787.57
52.3412         869.252         3787.57
51.1869         870.361         3787.57
50.0150         871.450         3787.57
48.8273         872.522         3787.57
47.6281         873.581         3787.57
46.4201         874.630         3787.57
   .               .               .
   .               .               .
   .               .               .
   .               .               .
```

A continuación se presentan las gráficas de los resultados obtenidos en ejemplo.

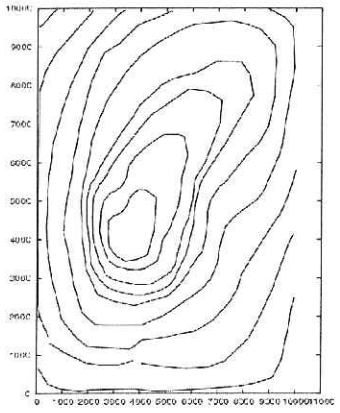


Figura G.2a Contornos originales.

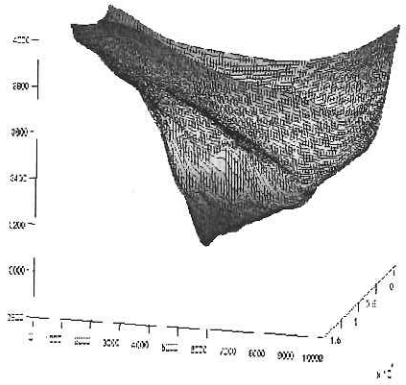


Figura G.2b: Superficie reconstruida por el algoritmo de Akima.

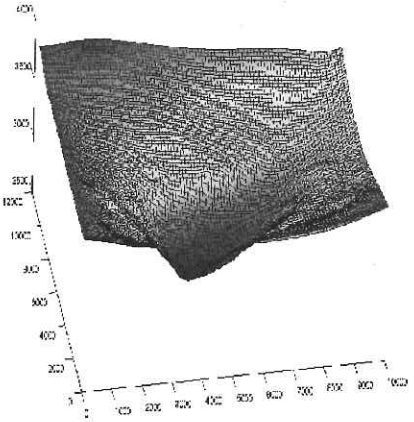


Figura G.2c: Superficie reconstruida por el algorimo de Akima.

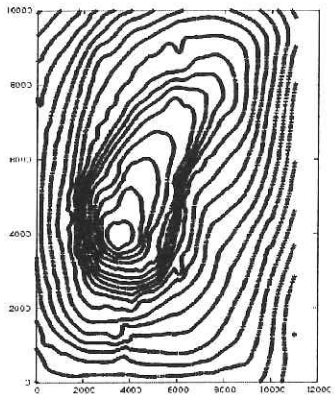


Figura G.2d: Contornos recuperados.

G.5. Conclusión.

A manera de conclusión podemos decir que el módulo `CONTOUR_GRAF.f90`, es una implementación que permite una interfase gráfica conveniente del algoritmo 626 de Albrecht Preusser que facilita la visualización del número de contornos deseado para una superficie proporcionada. Además puede ser fácilmente conectada al módulo *interpbivar.f90*, lo que permite la recuperación de un número deseado de contornos a partir un número fijo de éstos.

G.6. Referencias.

[1] Akima, H. A method of bivariate interpolation and smooth surface fitting for values given at irregularly distributed points. *OT Rep.* 75-70, U.S. Government Printing Office, Washington D.C., Aug. 1975.

[2] Akima, H. A method of bivariate interpolation and smooth surface fitting for irregularly distributed data points. *ACM Trans. Math Softw.* 4, 2, (June 1978), 148-159.

[3] Preusser A. Computing contours by successive solutions of quintic polynomial equations. *ACM Trans. Math. Softw.* 10, 4 (Dec. 1984), 463-472.

Apéndice H: Suavizamiento de Superficies a través de B-Splines usando una técnica multinivel

H.1. Resumen.

El propósito de este trabajo consiste en presentar un método de solución y el marco teórico del problema de suavizamiento de superficies. Este método de solución se le conoce como *B-spline multinivel*, ya que este método presenta suavizamiento tipo spline cúbico bilineal, que construye aproximaciones sucesivas sobre mallas cada vez más finas.

H.2. Introducción

Este método requiere únicamente del ingreso una colección de puntos $P=\{(x_c,y_c,z_c)\}_{c=1}^{n}$ con los cuales se construye la superficie suave, así que es necesario presentar las condiciones bajo las cuales es posible utilizar este método. Se puede entender por superficie a la gráfica de $(x,y,f(x,y))$ donde $f:\Omega\to\mathbb{R}$, por lo que observamos la condición para la colección de datos de entrada *P*,

$$\text{si } z_i\neq z_j \ para\ i\neq j \Rightarrow (x_i,y_i)\neq(x_j,y_j),$$

ya que de no ser así, no estaríamos hablando de una función, con esto también se quiere decir que es posible repetir en la colección de datos a una triada completa, sin que esto sea perjudicial para el método.

Otra condición del método, es que Ω sea una región rectangular en el plano que contenga al conjunto $\{(x_c,y_c)\}_{c=1}^{n}$ de las primeras dos entradas de *P*, es decir de la forma $\Omega=\{(x,y)|0\le x<m,0\le y<n\}$ donde m, n son naturales. Así, la superficie suave resultante de este método será construida únicamente en esta región acotada.

La superficie solución no exige que $z_c=f(x_c,y_c)$, pero por ser un método multinivel se puede decir que existe un nivel N para el cual $|f(x_c,y_c)-z_c|<\varepsilon\ \forall\varepsilon$, es decir, que puntualmente la función converge a la colección de datos. Esto en vez de ser un problema, puede ser una ventaja, ya que en el caso de querer levantar superficies de datos obtenidos en el campo, que en su mayoría cuentan con errores de medición, dichos errores son comparables con el obtenido en la aproximación

El objetivo principal se puede plantear formalmente de la siguiente manera:

Dada una colección de puntos en el espacio $P=\{(x_c,y_c,z_c)\}_{c=1}^{n}$, *donde la pareja ordenada* $\{(x_c,y_c)\}_{c=1}^{n}$ *está en un dominio rectangular* $\Omega=\{(x,y)|0\le x<m,0\le y<n\}$ *distribuidos arbitrariamente. Ajustar una superficie suave* (x,y,f) *con* $f=f(x,y)\in C^1$ *donde* $f:\Omega\to\mathbb{R}$ *sea tal que* $z_c=f(x_c,y_c)$.

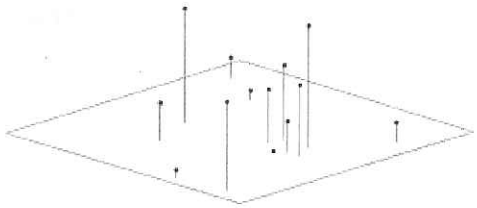


Figura H.1: Una colección de datos dispersos.

Dado que conocemos las condiciones mínimas de este problema, nos resta resaltar las ventajas del planteamiento. Cuando se dice 'distribuidos arbitrariamente', se esta diciendo que para cualquier distribución que se le de a los puntos $\{(x_c, y_c)\}_{c=1}^{n}$ el método presenta una solución, puede en este caso ser aleatoria, o es posible utilizar distribución uniforme, puntos sobre curvas de nivel o cualquier otra distribución deseada. Aunque la solución no es independiente de la distribución, así que para la reconstrucción de superficies, es también posible plantear puntos 'clave', o distribuciones 'clave' para obtener una solución más exacta.

El método B-*spline multinivel* tiene como primer objetivo, realizar una iteración del método el cual en la literatura se le conoce como algoritmo BA, al que comúnmente nombramos 'nivel', y así posteriormente concentrarnos en como realizar las iteraciones; por cada nivel encontramos una reparametrización que se realiza en la región Ω, que constituye la segunda fase del algoritmo MBA. Al final presentaremos las libertades que dicho método presenta.

H.3. Descripción del método

La función $f:\Omega\to\mathbb{R}$ que se presenta como solución b-spline al problema, está dada en términos de espacio bilineal spline, de la forma

$$f(x,y)=\sum_{k=0}^{3}\sum_{l=0}^{3}B_k(s)B_l(t)\Phi_{(i+k)(j+l)}$$

con

$$s=x-[x], t=y-[y] \quad y \quad i=[x]-1, j=[y]-1,$$

donde los coeficientes bilineales son planteados a través de un latice Φ, y la base del espacio spline empleada es conocida por base de Shepard, y es la siguiente:

$$\begin{aligned}
B_0(t)&=(1-t)^3/6,\\
B_1(t)&=(3t^3-6t^2+4)/6,\\
B_2(t)&=(-3t^3+3t^2+3t+1)/6,\\
B_3(t)&=t^3/6
\end{aligned}$$

definida para $t\in[0,1]$.

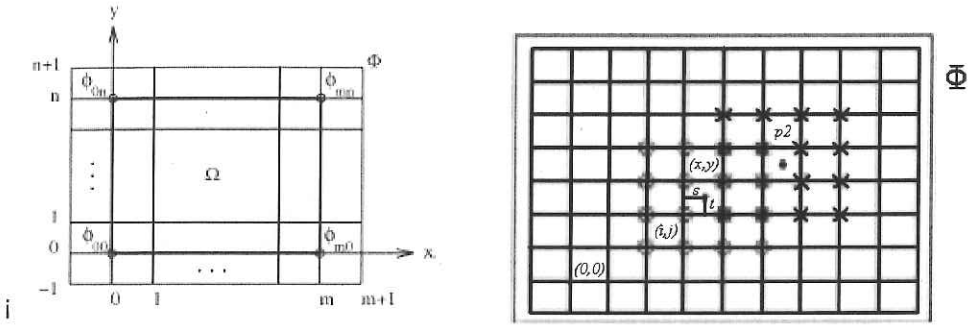
La dificultad de esto se encuentra en calcular el latice de control Φ, adecuado a la colección de datos P. Es importante mencionar que para que la función tenga valor en la frontera, se ocupa un latice, de dimensión $(m+3,n+3)$, ya que un spline cúbico utiliza 4

valores dados por pedazos aun en la frontera, es decir, se cubre la región Ω con una región rectangular $\{(x,y)|-1<x<m+1,-1<y<n+1\}$ y así se corresponde $\Phi(-1:m+1,-1:n+1)$, ver figura 2a.

Para obtener dicho latice en función de los datos se ocupa un criterio de **inverso de las distancias** entre cada dato y los 16 nodos mas cercanos de la malla entera, ver Figura H.2b. criterio que se obtiene a través de la seudo inversa sobre las distancias de cada dato con sus 16 vecinos en la malla entera

$$\phi_{kl}=\frac{w_{kl}z_c}{\sum_{a=0}^{3}\sum_{b=0}^{3}w_{ab}^2}$$

donde $w_{kl}=B_k(s)B_l(t)$ y $s=x_c-1, t=y_c-1$.



(a) Latice de control definido

(b) Inverso de la distancia entre el dato y sus 16 vecinos de la malla entera

Figura H.2: Latice empleado en un paso.

De esta manera se obtiene una función inicial f_0 que representa a la superficie inicial, como primer objetivo a atacar (ver introducción), conocido por algoritmo BA, donde nombraremos Φ_0 al latice de control. Dicha función inicial f_0 es en la mayoría de los casos una mala aproximación, es por esto que se procede con un método iterativo que por niveles se aproxime a la solución.

Para realizar el algoritmo multinivel MBA es necesario calcular f_0 por el algoritmo BA y calcular los errores en aproximación $\Delta z_c^{i+1}=z_c^i-f_i(x_c^i,y_c^i)$, y también es necesario reparametrizar Ω de la siguiente manera, sea $\Omega_0=\Omega$, entonces $\Omega_{i+1}=2\Omega_i$ y para el siguiente nivel, los errores serán ahora las alturas asociadas a la reparametrización cada dato, es decir, $P=\{(x_c^{i+1},y_c^{i+1},\Delta z_c^{i+1})\}_{c=1}^{n}$, donde $(x_c^{i+1},y_c^{i+1})=(2x_c^i,2y_c^i)$. Con estos nuevos datos es posible calcular f_{i+1} con el algoritmo BA a través del latice Φ_{i+1}. Una vez calculadas $f_0,f_1,..,f_n$ y haciendo uso de las propiedades vectoriales del espacio bilineal, se puede decir que la solución $f=f_0+f_1+...+f_n$ es una función que se aproxima con n niveles a los datos iniciales. Este proceso iterativo puede repetirse hasta cierto ε, tal que $|f(x_c,y_c)-z_c|<\varepsilon$. Ver Figura H.3.

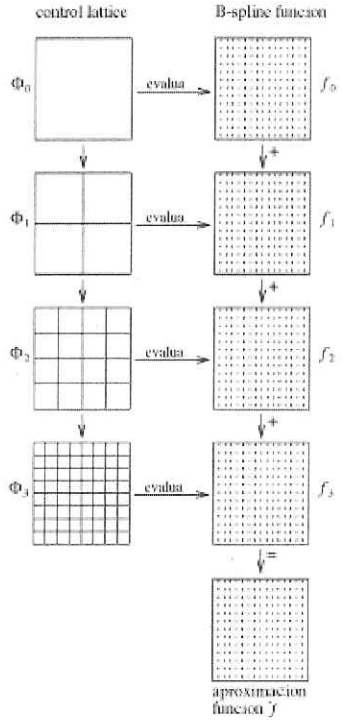


Figura H.3: Proceso iterativo dentro del latice de referencia.

H.4. Algoritmo Básico Multinivel AB.

4. Calcular Φ_0 utilizando la colección de puntos $P=\{(x_c,y_c,z_c)\}_{c=1}^{n}$
5. Se evalúa la función f_0 sobre los puntos de $\Omega_0=\Omega$ cuyas coordenadas sean enteras utilizando el latice Φ_0
6. Para cada una de las iteraciones deseadas, se realiza:
   - Se reparametriza la región Ω_i a $\Omega_{i+1}=2\Omega_i$ y se calcula el error de aproximación para z_c^i como $\Delta z_c^{i+1}=z_c^i-f_i\left(x_c^i,y_c^i\right)$.
   - Calcular Φ_{i+1} utilizando la colección de puntos $P=\left\{\left(x_c^{i+1},y_c^{i+1},\Delta z_c^{i+1}\right)\right\}_{c=1}^{n}$.
   - Se evalúa la función f_{i+1} sobre los puntos de Ω_{i+1} cuyas coordenadas sean enteras utilizando el latice Φ_{i+1}.
7. Se suman las iteraciones de la función $f=\sum_i f_i$.

Este algoritmo tiene ciertas desventajas a atacar, el hecho de calcular en cada iteración la función b-spline f_i requiere demasiado tiempo de cómputo.

Así que existe un método alternativo en el cual no es necesario calcular f_i por cada iteración, este método es conocido como algoritmo multinivel con refinamiento, y consiste en refinar el latice de control de cada iteración con un criterio razonable para ser llevado al siguiente nivel sin tener que calcular cada vez la función f_i. El criterio de refinamiento es el siguiente:

$$\phi'_{2i,2j} = \frac{1}{64}\begin{bmatrix} \phi_{i-1,j-1} + \phi_{i-1,j+1} + \phi_{i+1,j-1} + \phi_{i+1,j+1} \\ +6\left(\phi_{i-1,j} + \phi_{i,j-1} + \phi_{i,j+1} + \phi_{i+1,j}\right) + 36\phi_{ij} \end{bmatrix},$$

$$\phi'_{2i,2j+1} = \frac{1}{16}\begin{bmatrix} \phi_{i-1,j} + \phi_{i-1,j+1} + \phi_{i+1,j} + \phi_{i+1,j+1} \\ +6\left(\phi_{i,j} + \phi_{i,j+1}\right) \end{bmatrix},$$

$$\phi'_{2i+1,2j} = \frac{1}{16}\begin{bmatrix} \phi_{i,j-1} + \phi_{i,j+1} + \phi_{i+1,j-1} + \phi_{i+1,j+1} \\ +6\left(\phi_{i,j} + \phi_{i+1,j}\right) \end{bmatrix},$$

$$\phi'_{2i+1,2j+1} = \frac{1}{4}\left[\phi_{i,j} + \phi_{i,j+1} + \phi_{i+1,j} + \phi_{i+1,j+1}\right].$$

Este criterio de refinamiento se entiende como, dado el latice $\Phi_0 = \Psi_0$, construir el latice refinado $\Psi_0^{'}$, simultáneamente constrúyase el latice del primer nivel Φ_1 por el algoritmo MBA y posteriormente se considera $\Psi_1 = \Phi_1 + \Psi_0^{'}$, para el resto de las iteraciones, considérese el latice Φ_i obtenido por el algoritmo MBA en el i-ésimo nivel y el latice $\Psi_{i-1}^{'}$ refinado de Ψ_{i-1}, y constrúyase $\Psi_i = \Phi_i + \Psi_{i-1}^{'}$, al final de los n niveles deseados evalúe la función $f(x,y) = \sum_{k=0}^{3}\sum_{l=0}^{3} B_k(s)B_l(t)\Psi_n\left((i+k),(j+l)\right)$.

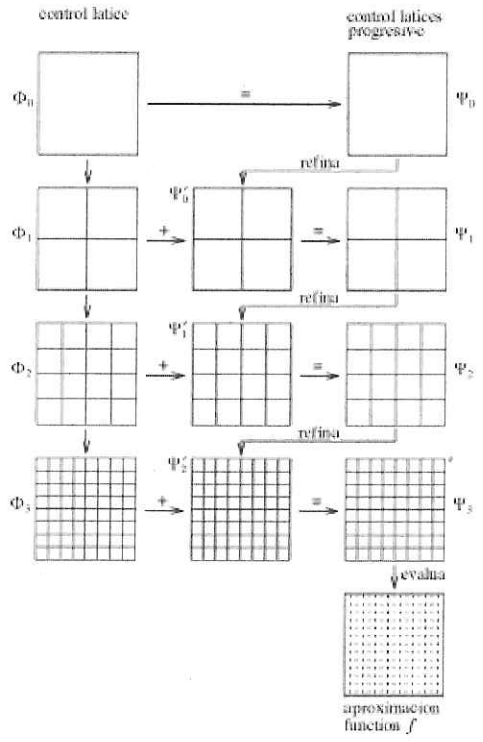


Figura H.4: Proceso adoptivo dentro del latice de referencia.

H.5. Algoritmo multinivel b-spline por refinamiento.

1. Calcular Φ_0 utilizando la colección de puntos $P=\{(x_c,y_c,z_c)\}_{c=1}^{n}$
2. Sea $\Psi'_0=0$
3. Para las iteraciones deseadas.
   - Se reparametriza la región Ω_i a $\Omega_{i+1}=2\Omega_i$ y se calcula el error de aproximación para z_c^i como $\Delta z_c^{i+1}=z_c^i-f_i\left(x_c^i,y_c^i\right)$.
   - Se calcula $\Psi_{i+1}=\Psi'_i+\Phi_i$
   - Calcular Φ_{i+1} utilizando la colección de puntos $P=\left\{\left(x_c^{i+1},y_c^{i+1},\Delta z_c^{i+1}\right)\right\}_{c=1}^{n}$.
   - Se refina $\Psi'_{i+2}=R\left(\Psi_{i+1}\right)$
4. Se evalúa la función f sobre los puntos de Ω_{i+1} cuyas coordenadas sean enteras utilizando el latice Φ_{i+1}.

H.6. Modo de uso de las rutinas y ejemplos

El algoritmo entregable MBA con refinamiento se encuentra programado dentro de la rutina `mba_ref`, la cual se programó en Fortran90. Para hacer uso de esta rutina, se leen los datos de un archivo de texto plano (fichero de entrada) y se deposita la salida en un archivo de texto plano (fichero de salida), se recomienda dar extensión `.dat` por ser ficheros de datos.

El programa `mba_ref` emplea las siguientes funciones y subrutinas:

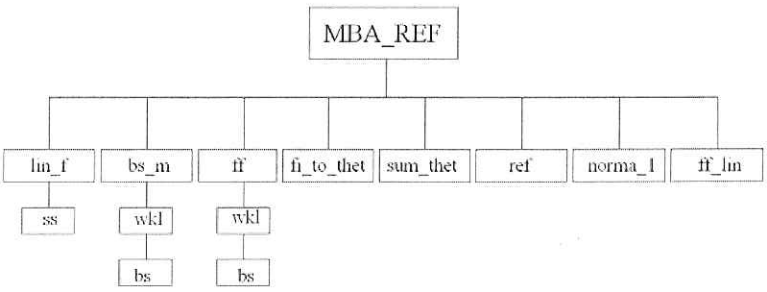


Figura H.5: Diagrama esquemático del programa `mba_ref`.

`BS`, evalúa x en los polinomios del la base.
`WKL`, evalúa $B_k(s)B_l(t)$
`BS_M`, calcula el latice de control Φ
`FF`, evalúa $f(x,y)=\sum_{k=0}^{3}\sum_{l=0}^{3}B_k(s)B_l(t)\Phi_{(i+k)(j+l)}$ usando Φ
`FI_TO_THET`, nombra $\Psi_0=\Phi_0$
`SUM_THET`, calcula Ψ_i
`REF`, refina $\Psi_{i-1}'=R\left(\Psi_{i-1}\right)$
`NORMA_1`, criterio de paro

Para poder correr los 3 programas es necesario tener un fichero con formato fijo, en primer instancia se debe de tener los bordes de la región Ω (no es necesario maneja $\Omega = \{(x,y) | 0 \le x < m, 0 \le y < n\}$, ya que el programa traslada los mínimos al (0,0), pero si es necesario que Ω esté contenido en el primer cuadrante de $\mathbb{R}^2$, es decir que $x_c \ge 0, y_c \ge 0 \; \forall c$) en el siguiente orden, $\min_x \Omega \min_y \Omega \max_x \Omega \max_y \Omega$, posteriormente hay que tener el número de puntos a interpolar es decir $\#P = nc$, posteriormente con la colección de puntos en el espacio, colocados en 3 columnas, $X|Y|Z$. Como se muestra en el siguiente ejemplo:

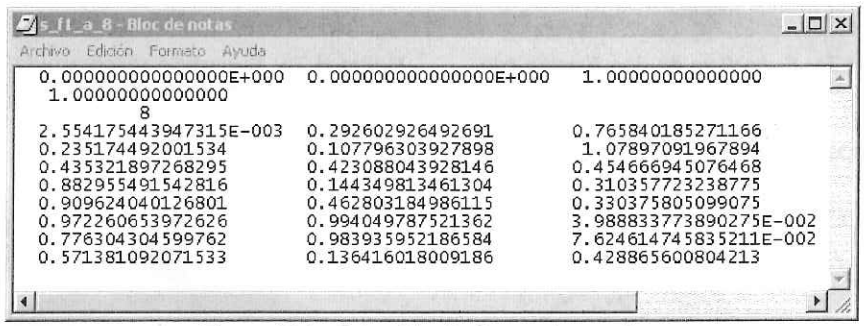


Figura H.6: Colección de Datos de prueba.

Una vez que la lectura se realiza con éxito, el programa pedirá al usuario que proporcione el nombre del archivo de salida de preferencia con extensión `.dat`, posteriormente te pedirá:

- El número de nivel máximo
- Dimensión de escala mínima
- Número de niveles deseados

Para poder ejecutar exitosamente el programa la siguiente tabla relaciona dimensiones de escala con niveles.

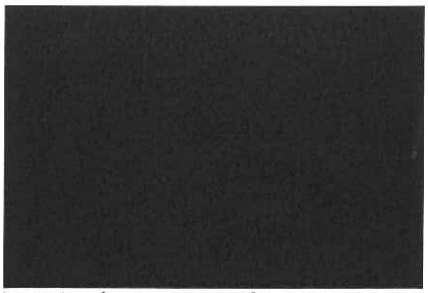


Tabla H.1: Relación de dimensión de escala con niveles.

Así pues, la siguiente desgualdad es el criterio para tener una salida favorable.

 # nivel máximo ≥ # niveles deseados + nivel asociado a dimensión de escala

Un ejemplo con expresión analítica $f:\Omega\to\mathbb{R}$ en la región $\Omega=[0,1)\times[0,1)$, dada por

$$f(x,y)=e^{-(x-\frac{1}{2})^2+(y-\frac{1}{2})^2}$$

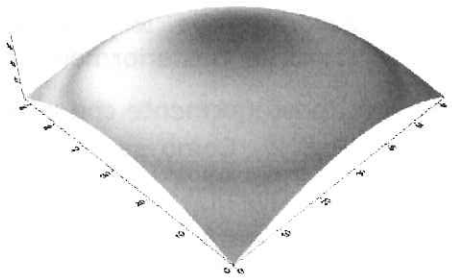


Figura H.7a: Superficie de una función cde prueba.

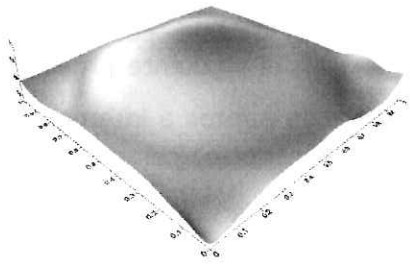


Figura H.7b: Superficie reconstruida con 5 niveles y dimensión de escala 1 1

Y un ejemplo para una superficie, dadas curvas de nivel es el siguiente.

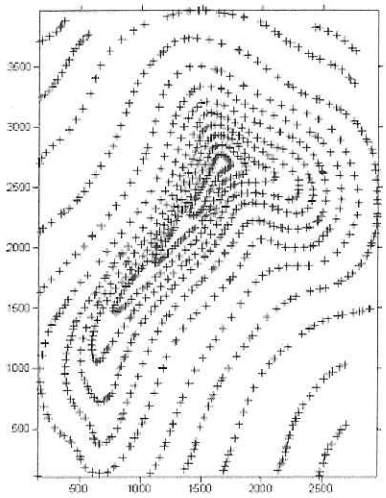


Figura H.8a: Mapa de datos de profundidad de un yacimiento.

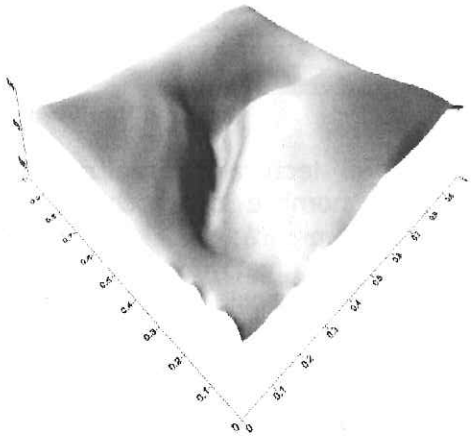


Figura H.8b: Superficie reconstruida con 7 niveles y dimensión de escala 1 1

H.7. Conclusión

En términos generales este método de suavizamiento es muy eficiente para datos dispersos, pero sólo funciona para aproximar superficies de clase 1, C^1, además es necesario tener definida la superficie en toda la región de estudio Ω. Otra de las discusiones que se dieron en este trabajo fue como hacer para que el algoritmo intente construir evaluaciones sobre la parte superior y derecha de la frontera de Ω, ya que no debemos olvidar que $\Omega=\{(x,y)|0\le x<m, 0\le y<n\}$. Así que una alternativa fue construir una malla rectangular un poco más pequeña que Ω. Una cuestión interesante resulta de observar que en ningún momento es necesario empezar el algoritmo BA a partir de Ω, en todo caso, lo importante radica en conservar las proporciones de distancia de cada punto, así que a este algoritmo previamente se le puede implementar una

reparametrización al cuadrado unitario, y al terminar el proceso, regresar el resultado a sus dimensiones originales.

En este trabajo se presentó como construir superficies suaves en regiones rectangulares, pero es posible extender este resultado a regiones convexas haciendo uso de el latice de control, remplazando la evaluación en la malla rectangular por la malla convexa, por supuesto contenida en Ω. La finalidad de esto es poder hacer uso de este recurso para construir la malla 3D, definida entre 2 superficies.

Una última discusión se da en la distribución arbitraria de los datos $\{(x_c, y_c)\}_{c=1}^{n}$. De que manera es conveniente dar entrada a puntos 'clave' de la superficie. Para eso es necesario estudiar la reconstrucción de superficies como planteamiento analítico a través de distintas distribuciones, para saber si existe alguna influencia directa sobre la solución.

H.8. Referencias

[1] S. Lee, K.-Y. Chwa, S. Y. Shin, and G. Wolberg, "Image Metamorphosis Using Snakes and Free-Form Deformation", *Computer Grafics* (Proc. SIGRAPH'95) .

[2] S. Lee, K.-Y. Chwa, S. Y. Shin, and G. Wolberg, "Image Metamorphosis with Scattered Feature Constants", *IEEE Trans. Visualization and Computer Graphics*, vol. 2, no. 4, pp. 439-448, 1995

[3] T. Lyche and K. Morken, "Making the Oslo Algorithm More Efficient", SIAM J. Numerical Analysis, vol. 23, no. 3, pp 663-675, 1986

Apéndice I: Tratamiento de contornos

I.1. Planteamiento del problema

Dado un conjunto de puntos en el plano real $P=\{p_i\}_{i=1}^{n}$ que describen un contorno poligonal, el problema es incrementar o disminuir el número de puntos sobre éste, es decir, repoblar el contorno.

Este problema tiene dos etapas básicas: suavizamiento e interpolación del contorno. En la primera etapa se obtiene una curva C^1, S, que aproxima a P; en la segunda, se construye una parametrización adecuada de S que permita repoblar fácilmente el contorno.

I.2. Suavizamiento de contornos

Una propiedad que se desea observar en la curva S, es que respete la forma de P. Para lograrlo, se elige a S entre la familia de splines cónicos, con la característica de que uno o varios segmentos sean cónicas singulares. De esta forma, la curva S que suaviza a P se define como:

$$S(u)=s_i(u), \quad u\in[u_i,u_{i+1}], \quad i=1,\ldots,N.$$
$$\text{donde } s_i(u):[u_i,u_{i+1}]\to\mathbb{R}^2 \quad \text{es una curva racional cuadrática.}$$

Cada segmento del spline cónico se representa en su forma de Bézier racional cuadrática estándar.

I.2.1. Cónicas en su forma de Bézier racional cuadrática

Una función racional $c:\mathbb{R}\to\mathbb{R}^2$ dada por

$$c(t)=\frac{w_0(1-t)^2 b_0+w_1 2t(1-t)b_1+w_2 t^2 b_2}{w_0(1-t)^2+w_1 2t(1-t)+w_2 t^2},$$

es llamada una curva de Bézier racional cuadrática, los coeficientes b_i son llamados puntos de control, mientras que al coeficiente w_i se le conoce como peso asociado al punto de control b_i, ver figura 1.

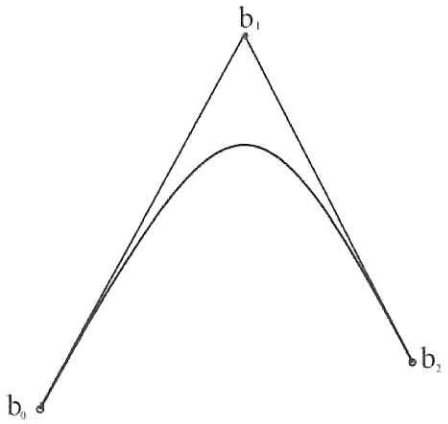


Figura 1. Curva de Bézier racional cuadrá tica.

Propiedades

Una curva de Bézier racional cuadrática con puntos de control b_0, b_1, b_2 y pesos w_0, w_1, w_2 tiene las siguientes propiedades:

1. El primer y último punto de la curva coincide con el primer y último punto del polígono de control : $c(0) = b_0, \quad c(1) = b_1$.
2. El vector tangente en los extremos del arco tiene la misma dirección que los lados del polígono de control: $c'(0) = \frac{2w_1}{w_0}(b_1 - b_0), \; c'(1) = \frac{2w_1}{w_2}(b_2 - b_1)$.
3. Si todos los pesos son positivos y $t \in [0,1]$, entonces c(t) cae en la cubierta convexa de los puntos de control.

I.2.2. Forma de Bézier racional cuadrática estándar de una cónica

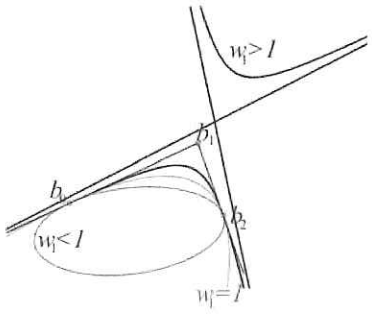
En la práctica, cuando se requiere obtener la representación de Bézier racional de una cónica, es conveniente reducir el número de parámetros que la describen. Se puede mostrar que si todos los pesos de la cónica son positivos, ésta se puede reparametrizar para llevarla a su forma estándar

$$b(t) = \frac{(1-t)^2 b_0 + w_1 2t(1-t) b_1 + t^2 b_2}{(1-t)^2 + w_1 2t(1-t) + t^2}$$

Clasificación

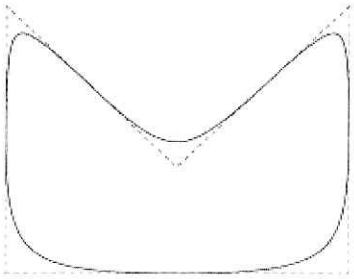
El tipo de cónica depende de sus singularidades, las cuales corresponden a los ceros de la función de peso $w(t) = -2(w_1 - 1)t^2 + 2(w_1 - 1)t + 1$. Se puede ver fácilmente que las singularidades caen fuera del intervalo unitario y que el tipo de cónica en función de w_1 es la siguiente:

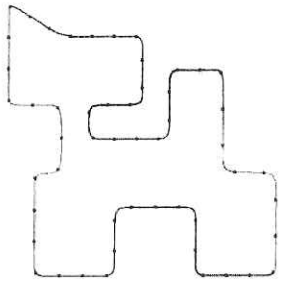
$$\text{tipo de cónica}\begin{cases}\text{elipse} & \text{si } w_1 < 1\\ \text{parábola} & \text{si } w_1 = 1\\ \text{hipérbola} & \text{si } w_1 > 1\end{cases}$$



I.2.3. Algoritmo para el suavizamiento de contornos

En vista de que se requiere suavizar a *P* conservando su forma lo más posible, el objetivo del algoritmo es quitar sólo "los picos". Así, por cada terna de puntos consecutivos p_{j-1}, p_j, p_{j+1}, se aproxima el vértice p_j usando un arco cónico cuyos extremos se eligen en los lados $p_{j-1}p_j$ y $p_j p_{j+1}$, respectivamente, tratando de que entre cada segmento cónico no singular se incluya uno singular. En la siguiente figura se muestran dos ejemplos de suavizamiento para contornos cerrados.





La construcción del spline cónico S con continuidad C^1 consta de dos etapas. En la primera, se construye un spline G^1 obteniéndose el polígono de control del spline $b_0,\ldots,b_{2N}$ y los pesos $w_1,\ldots w_N$; en la segunda, se determinan los nodos $u_0,\ldots,u_N$ que hacen del spline S uno de clase C^1.

I.3. Reparametrización de curvas

En la fase de suavizamiento se genera un spline cónico $S(u):[0,1]\to\mathbb{R}^2$ de clase C^1 que aproxima un contorno poligonal. En la mayoría de los casos la parametrización del spline cónico obtenido no es adecuada para determinar una población del contorno, por tanto, resulta indispensable contar con un algoritmo que permita obtener numéricamente la reparametrización , ***S,*** mediante la longitud de arco de *S*.

Sean $s(u)=\int_0^u \|S'(\tau)\|_2 d\tau$, $la=\int_0^1 \|S'(\tau)\|_2 d\tau$ la longitud de arco de S y $s^{-1}(l)=h(l):[0,la]\to[0,1]$ entonces, la composición de funciones $\mathbf{S}(l)=S(h(l)):[0,la]\to\mathbb{R}^2$ es la reparametrización $\boldsymbol{S}$ de S mediante la longitud de arco.

De esta forma, el problema numérico consiste en determinar un spline α tal que

1. $\alpha(l_i)=h(l_i)$, $l_i=la/n$, $i=0,\ldots,n.$
2. $\alpha(l)$ monótono y con derivada continua.
3. $\alpha(l)\approx h(l)$

$\alpha(l)$ se determina construyendo una sucesión de funciones $\alpha_k(l)$ tal que $\alpha_k(l)\xrightarrow[k\to\infty]{}\alpha(l)$.

I.3.1. Construcción de aproximaciones

Restricciones para generar la aproximación inicial α_0

1. $u_i^0=i/n$
2. $l_i^0=s(u_i^0)$
3. $\alpha'_i(l_i^0)=1/\|S'(u_i^0)\|_2$
4. α_0 función monótona y con derivada continua.

Dada α_k, se calculan las condiciones de interpolación : $u_i^{k+1}=\alpha_k(l_i)$ $i=0,\ldots,n$ y $l_i^{k+1}=s(u_i^{k+1})$, entonces la siguiente aproximación debe satisfacer las restricciones:

1. $\alpha_{k+1}(l_i^{k+1})=u_i^{k+1}$
2. $\alpha'_{k+1}(l_i^{k+1})=1/\|S'(u_i^{k+1})\|_2$
3. α_{k+1} función monótona y con derivada continua.

En la construcción de cada spline aproximante, α_k, es importante que se respeten las condiciones de interpolación y, puesto que la solución se usará para reparametrizar una curva S, es conveniente que el spline α_k sea de grado uno. De este modo, cada aproximación se construye usando un esquema de interpolación racional lineal.

I.3.2. Construcción del Spline

Para determinar el spline α_{k+1}, por cada dos nodos consecutivos l_i^{k+1}, l_{i+1}^{k+1} se usa un nodo auxiliar, $l_{i+1/2}^{k+1}$, para costruir un spline racional lineal formado por dos curvas de Bézier racional lineal. Entonces, el spline está dado por

$$\alpha_{k+1}(l)=\begin{cases} b_{il}^1(l), & l_i^{k+1}\le l\le l_{i+1/2}^{k+1} \\ b_{ir}^1(l), & l_{i+1/2}^{k+1}\le l\le l_{i+1}^{k+1} \end{cases}$$

donde

$$b_{il}^1(l)=\frac{w_i u_i^{k+1}\left(l_{i+1/2}^{k+1}-l\right)+w_{i+1/2}u_{i+1/2}^{k+1}\left(l-l_i^{k+1}\right)}{w_i\left(l_{i+1/2}^{k+1}-l\right)+w_{i+1/2}\left(l-l_i^{k+1}\right)},$$

$$b_{ir}^1(l)=\frac{w_{i+1/2}u_{i+1/2}^{k+1}\left(l_{i+1}^{k+1}-l\right)+w_{i+1}u_{i+1}^{k+1}\left(l-l_{i+1/2}^{k+1}\right)}{w_{i+1/2}\left(l_{i+1}^{k+1}-l\right)+w_{i+1}\left(l-l_{i+1/2}^{k+1}\right)},$$

$$w_0=1$$

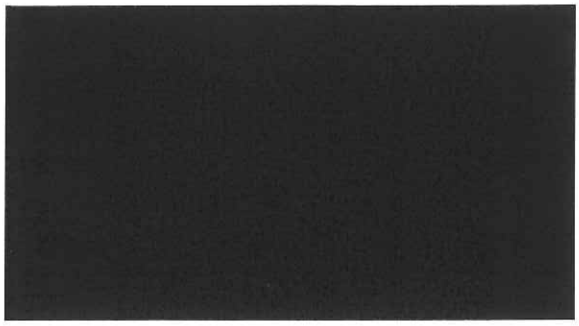
$$w_{i+1}=\left(\alpha'_{k+1}\left(l_i^{k+1}\right)\Big/\alpha'_{k+1}\left(l_{i+1}^{k+1}\right)\right)^{1/2}w_i$$

$$u_{i+1/2}^{k+1}=\frac{w_i u_i^{k+1}+w_{i+1}u_{i+1}^{k+1}}{w_i+w_{i+1}}$$

$$w_{i+1/2}=\left(\frac{w_i\alpha'_{k+1}\left(l_i^{k+1}\right)+w_{i+1}\alpha'_{k+1}\left(l_{i+1}^{k+1}\right)}{2}\right)\left(\frac{l_{i+1}^{k+1}-l_i^{k+1}}{u_{i+1}^{k+1}-u_i^{k+1}}\right)$$

I.4. Descripción del programa

El tratamiento de un contorno involucra dos etapas: el suavizamiento de una poligonal plana *P* y la reparametrización mediante la longitud de arco, de la curva generada en la etapa previa; ambos algoritmos se implementaron en Fortran 90 y para su experimentación se implementó la rutina principal: **Tratamiento_contorno.f90**.



Entrada:

M Número de puntos sobre el contorno
XB XB(i) abscisa del i-ésimo punto sobre el contorno
YB YB(i) ordenada del i-ésimo punto sobre el contorno

Salida

num_puntos Cantidad de puntos del contorno repobaldo
X_SC_R X_SC_R(i) abscisa del i-ésimo punto sobre el contorno repoblado
Y_SC_R Y_SC_R(i) ordenada del i-ésimo punto sobre el contorno repoblado

I.4.1. Tratamiento de contornos

La rutina principal **Tratamiento_contorno.f90** se divide en: lectura de datos, suavizamiento del contorno, cálculo de la longitud de arco del contorno suavizado, cálculo de la inversa de la función longitud de arco, poblar el contorno y repoblación.

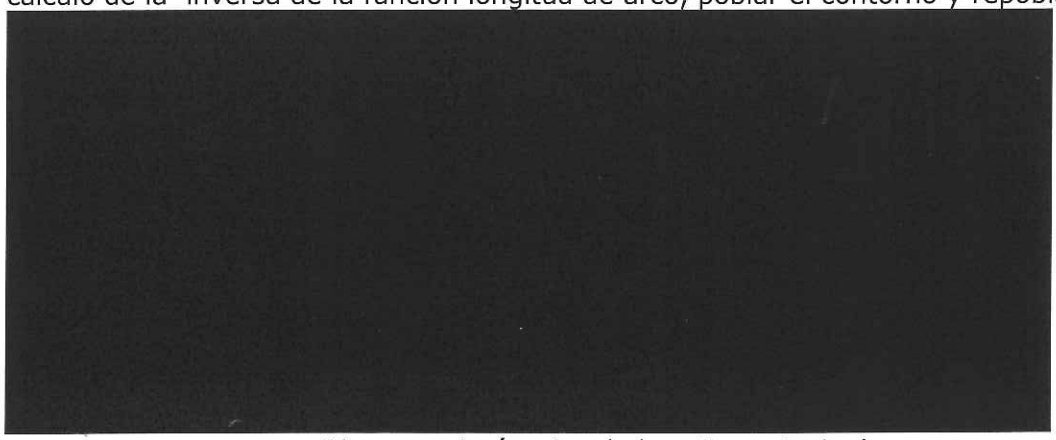


Diagrama jerárquico de la rutina principal

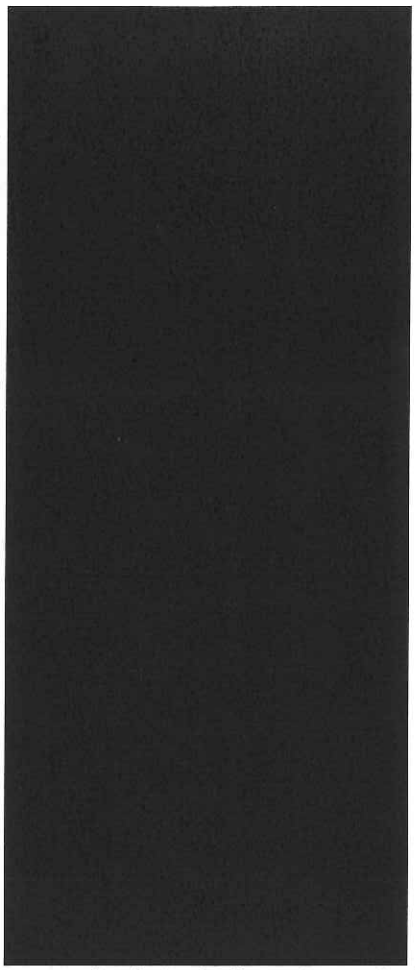


Diagrama de flujo de la rutina principal

I.4.2. Lectura de datos

La entrada de los datos a suavizar se realiza a través de la lectura de un archivo con extensión .con.

Formato para los archivos de contorno:

$m\ ibflag\ nb1\ nb2\ nb3\ nb4$

$x_1 \quad y_1$
$\vdots \quad \vdots$
$x_m \quad y_m$

M Número de puntos

Ibflag Bandera que indica el tipo de contorno

$$ibflag \begin{cases} -1 & \textbf{contorno abierto} \\ 0 & \textbf{contorno cerrado} \\ 1 & \textbf{contorno cerrado} \\ & \textbf{consubfronteras definidas} \end{cases}$$

nbi Número de puntos de la i-ésima subfrontera

La lectura del archivo se hace en dos etapas: Lectura de parámetros y lectura de las coordenadas. En la primera, se efectúa la lectura de la primera línea del archivo, una vez que se tiene el valor de *m*, en tiempo de ejecución, se reserva el espacio de memoria suficiente para almacenar las coordenadas de los puntos para posteriormente realizar la lectura de las coordenadas.

I.4.3. Rutinas usadas

`Lee_parametros(FNAME,M,IBFLAG,NB1, NB2,NB3,NB4)`

Función que dado el nombre del archivo de un contorno lee el encabezado del archivo.

Parámetro de entrada:
`FNAME` – nombre del archivo de contorno.

Parámetros de salida:
`M,IBFLAG,NB1,NB2,NB3,NB4`

Dato de regreso
`0`- error al leer el archivo,
`1` - lectura exitosa.

`Lee_Coordenadas(FNAME,XB,YB,M)`

Parámetro de entrada:
`FNAME` – nombre del archivo de contorno.
`M` – número de puntos.

Parámetros de salida:
`XB( i )`- abscisa del i-ésimo punto

YB(i)- ordenada del i-ésimo punto

Dato de retorno
0- error al leer el archivo,
1 - lectura exitosa.

Suavizar contorno

```
Suaviza_con(XB,YB,M,IBFLAG,CONTROL_P,MAX_POINTS,MAX_SECTIONS,
                    SECTIONS,WEIGHTS,KNOTS,SINGULAR_CONIC)
```

Función que, dado el número de puntos del contorno y sus coordenadas, genera un spline cónico con derivada continua que aproxima el contorno poligonal . Se usa la representación de Bézier racional en su forma estándar.

Parámetros de entrada:

M	Número de puntos
XB,YB	Coordenadas de los puntos
IBFLAG	Tipo de contorno
MAX_POINTS	Número máximo de puntos de control del spline cónico.
MAX_SECTIONS	Número máximo de secciones del spline cónico.

Parámetros de salida:

CONTROL_P(MAX_POINTS,2)	Polígono de control del spline cónico.
SECTIONS	Número de secciones del spline.
WEIGHTS(MAX_SECTIONS)	Pesos de cada sección cónica.
KNOTS(MAX_SECTIONS-1)	Nodos del spline
SINGULAR_CONIC(MAX_SECTIONS)	SINGULAR_CONIC(i) =1 si la i-ésima sección cónica del spline es singular, en otro caso 0.

Rutinas o funciones usadas

DOUBLE PRECISION FUNCTION `BXY(A,B,t)`

Función que obtiene la interpolación lineal determinada por los nodos **A, B** y parámetro **t.**

DOUBLE PRECISION FUNCTION `pesow1(b0x, b0y, b1x, b1y, b2x, b2y)`

Función que asigna el peso central de la cónica con puntos de control:
`bi =( bix,biy).`

`SUBROUTINE KNOTS(CONTROL_P,MAX_POINTS,WEIGHTS,`
`MAX_SECTIONS,SECTIONS,KNOT)`

Rutina que, dado el polígono de control y los pesos de un spline cónico G^1 determina los nodos para generar un spline cónico C^1.

Calcula longitud de arco

DOUBLE PRECISION FUNCTION
`long_arco_spl(CONTROL_P,MAX_POINTS,WEIGHTS,MAX_SECTIONS,KNOTS,`
`SINGULAR_CONIC,SECTIONS,n)`

Función que calcula la longitud de arco de un spline cónico dividiendo el intervalo de definición en n partes .

Parámetros de entrada:

`CONTROL_P`	Polígono de control del spline cónico.
`MAX_POINTS`	Número máximo de puntos de control del spline cónico.

	MAX_SECTIONS	Número máximo de secciones del spline cónico.
	WEIGHTS	Pesos de cada sección cónica.
	KNOTS	Nodos del spline
	SINGULAR_CONIC	SINGULAR_CONIC(i) = 1 si la i-ésima sección cónica del spline es singular, en otro caso, toma el valor de 0
	SECTIONS	Número de secciones del spline.
	N	Tamaño de la partición uniforme del intervalo unitario para calcular la longitud de arco del spline.
Dato de regreso	**Longitud_de_arco**	Longitud de arco del spline cónico.

Rutinas o funciones usadas:

DOUBLE PRECISION FUNCTION
`larcoSplcon(CONTROL_P,MAX_POINTS,WEIGHTS,MAX_SECTIONS,KNOTS,SINGULAR_CONIC,Sections,l_ini,l_fin)`

Función que calcula la longitud de arco del spline cónico en el intervalo $[\mathbf{l\_ini}, \mathbf{l\_fin}]$.

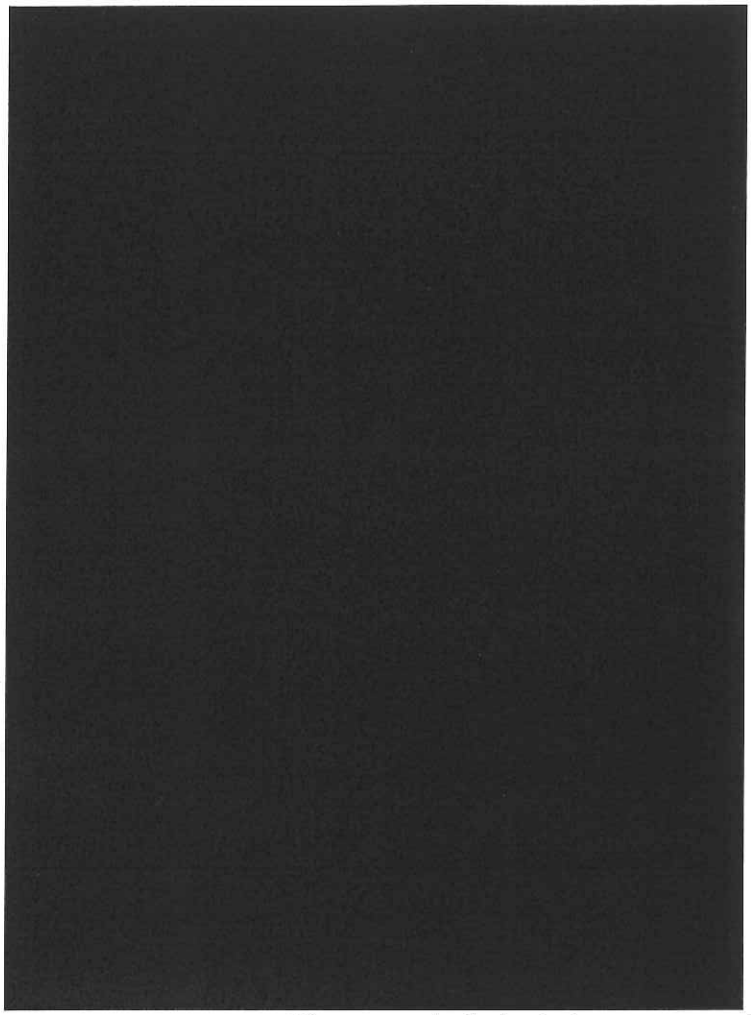
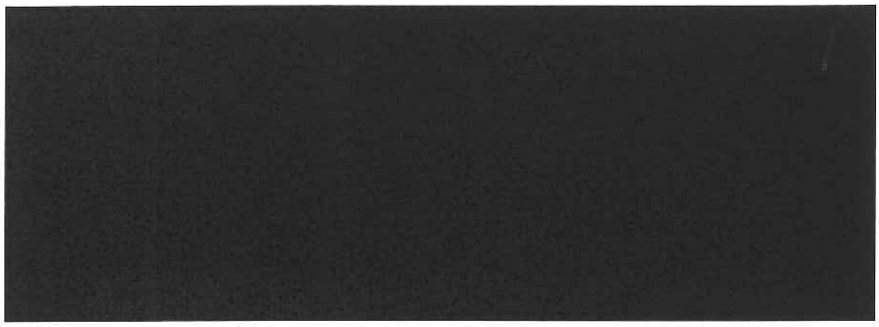


Diagrama de flujo de long_arco_spl

Calcula inversa de la longitud de arco



SUBROUTINE `SplineRacLin(Control_P,MAX_POINTS,Weights,MAX_SECTIONS, KNOTS, Singular_conic,Sections,Long_arco_spl,num_nodos,knot_srl,coef_srl)`

Esta rutina calcula un spline racional lineal que aproxima la función inversa de la longitud de arco del spline cónico.

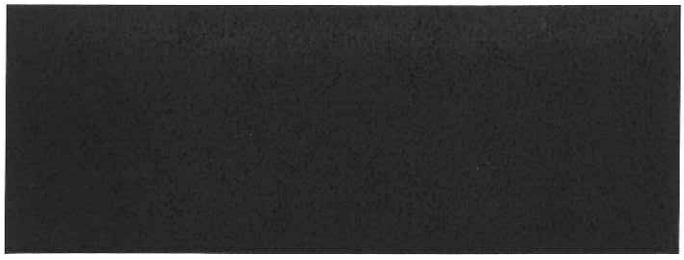
Parámetros de entrada

`CONTROL_P(MAX_POINTS,2)`	Polígono de control
`MAX_POINTS`	Máximo número de puntos de control del spline.
`Weights(MAX_SECTIONS)`	Pesos de cada sección del spline.
`MAX_SECTIONS`	Máximo número de secciones del spline.
`KNOTS(MAX_SECTIONS+1)`	Nodos del spline cónico
`SINGULAR_CONIC(MAX_SECTIONS)`	SINGULAR_CONIC(i) = 1 si la i-ésima sección cónica del spline es singular, en otro caso, toma el valor de 0.
`Long_arco_spl`	Longitud de arco del spline.
`num_nodos`	Cantidad de nodos para generar el spline racional lineal.

Parámetros de salida

`Knots_srl(2num_nodos-1)`	Nodos del spline racional lineal
`Coef_srl(2(num_nodos-1),4)`	Coeficientes del spline racional lineal.

Rutinas o funciones usadas



SUBROUTINE reparamSplcon(Control_P,MAX_POINTS,WEIGHTS,MAX_SECTIONS,KNOTS,Singular_conic,Sections,t_k,deltau,n,t_k_sig,knots_srl,coef_srl)

Rutina que, dada una aproximación del spline racional lineal genera la siguiente.

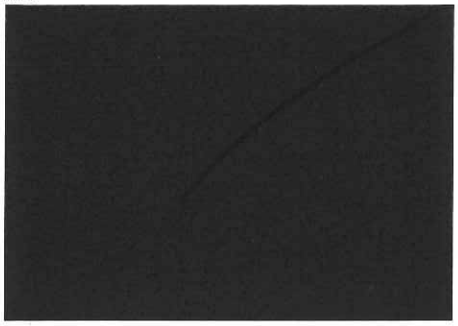
SUBROUTINE CONIC_SPLINE_VAL(Control_P,MAX_POINTS,WEIGHTS,MAX_SECTIONS, KNOTS, Singular_conic,Sections,u, X_SP_t,Y_SP_t)

Esta rutina obtiene el punto del spline cónico correspondiente al valor del parámetro u.

Poblar Contorno

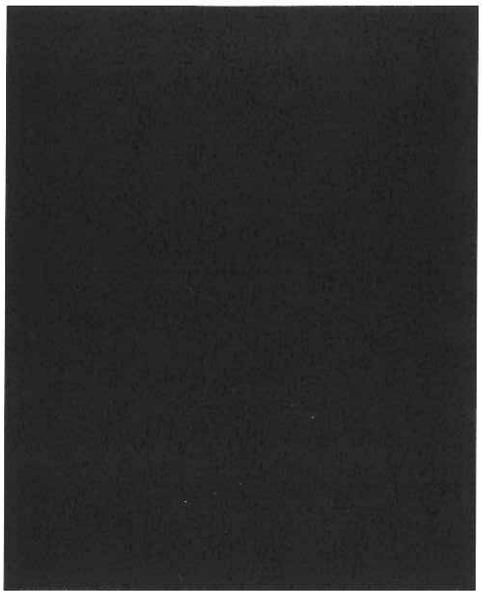
En esta etapa se evalúa la reparametrización del spline cónico mediante la longitud de arco en tantos valores como lo indica la variable Tam_población.

Rutinas o funciones usadas



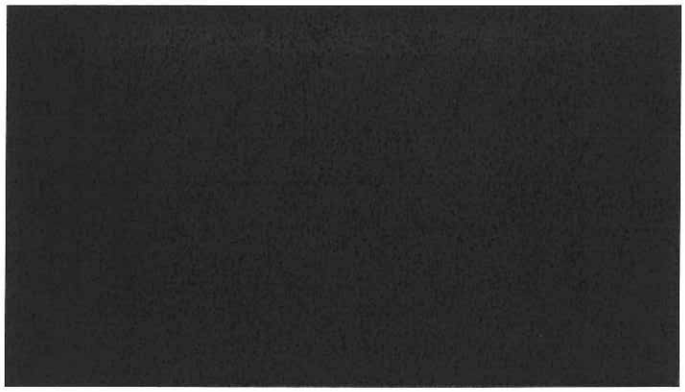
SUBROUTINE ratsplval(coef_srL,knots_srl,dim,U,Tam_poblacion,T_DU)
Rutina que evalúa el spline racional lineal en los valores guardados en el arreglo U, el resultado se almacena en el arreglo T_DU.

SUBROUTINE CONIC_SPLINE_VAL(Control_P,MAX_POINTS,WEIGHTS,MAX_SECTIONS, KNOTS, Singular_conic,Sections,u, X_SP_t,Y_SP_t)



Repoblar contorno

Esta fase permite repoblar el contorno determinado por la población generada en la etapa previa. La sección del contorno a repoblar se indica a través del índice asociado a los puntos extremos que lo determinan. De esta forma, dados el índice inicial, el índice final y la cantidad de puntos a agregar, se determina la repoblación del contorno.



Apéndice J: Tareas para el módulo GRID

J.1. Antecedentes

Una pieza clave para obtener una malla 3D de simulación, es la reconstrucción de cada una de las capas o superficies de propiedades que interesa medir sobre el yacimiento. La forma tradicional de construir una malla 3D de simulación usando esta información es, primero, construir una malla plana 2D de estudio sobre la cima de la sección del yacimiento y, segundo, bajar esta sobre cada capa. Este será el punto de partida para el presente trabajo.

A lo largo del presente año, hemos desarrollado herramientas que nos permiten reconstruir superficies sobre datos dispersos. De igual manera, contamos con aquellas que nos permiten construir una malla plana 2D. Por lo cual, estamos en la dirección correcta para atacar el problema de la construcción de la malla 3D sobre yacimientos homogéneos; es decir, sin fracturas.

J.2. Tareas del módulo GRID

Nos reunimos con el Grupo de Visualización e Interfaz Gráfica en diferentes sesiones, para describir el conjunto de tareas que contará el Módulo GRID, generador de mallas 3D del Simulador Numérico. Se discutió sobre la filosofía de trabajo y el tipo de usuario que utilizará el sistema. En general, se discutió el documento "Propuesta para el diseño de la Interfase Gráfica para el Sistema Simulador de Yacimientos de Hidrocarburos, Módulo: Sistema Generador de la malla de Simulación 3D (sin fracturas)" que formó parte del primer reporte sobre la interfaz gráfica.

En la segunda entrega sobre la interfaz gráfica con el usuario, se enumeraron las tareas que se desea cumpla el módulo para visualizar la malla 3D de simulación.

En el reporte: "Algunos aspectos sobre la Visualización de una Malla 3D Estructurada para la Simulación de un Yacimiento de Hidrocarburos", se discutió una forma muy interesante de visualizar la malla 3D de simulación. En ese documento se discute la importancia de contar con esa malla de referencia, con el fin de poder observar cortes transversales de la malla 3D así como asignar propiedades de la roca a cada una de las celdas e indicar las celdas activas o inactivas y los pozos, extractores o inyectores.

Las tareas que involucra el módulo GRID son:

Tareas

1. Datos de la cima
    a. Mapa de contornos de datos
        i. Leer los datos desde un archivo
            1. Definirlos (Los contornos no se intersecan)
                a. Contornos abiertos (botón)
                b. Contornos cerrados (botón)
            2. Eliminar contornos previamente definidos
            3. Editar los contornos (Mueves puntos, añades, quitas)
            4. Suavizarlos (suavizar e interpolar vía reparametrización)
            5. Editar propiedades: profundidad, densidad, permeabilidad, etc.

Tareas

6. Mostrar el contorno mediante colores dependiendo de su propiedad
7. Mostrar información (Área, perímetro, etiqueta, cerrado, abierto)

ii. Crearlos en pantalla (cíclico)
1. Definirlos (Los contornos no se intersecan)
a. Contornos abiertos (botón)
b. Contornos cerrados (botón)
2. Eliminar contornos previamente definidos
3. Editar los contornos (Mueves puntos, añades, quitas)
4. Suavizarlos (suavizar e interpolar vía reparametrización)
5. Editar propiedades: profundidad, densidad, permeabilidad, etc.
6. Mostrar el contorno mediante colores dependiendo de su propiedad
7. Mostrar información (Área, perímetro, etiqueta, cerrado, abierto)

b. Datos dispersos
i. Leer los datos desde un archivo
1. Definirlos
2. Editar los puntos (Mueves puntos, añades, quitas)
3. Editar propiedades: profundidad, densidad, permeabilidad, etc.
4. Mostrar el punto mediante colores dependiendo de su propiedad
5. Mostrar información (etiqueta y propiedades)

ii. Crearlos en pantalla (cíclico)
1. Definirlos
2. Editar los puntos (Mueves puntos, añades, quitas)
3. Editar propiedades: profundidad, densidad, permeabilidad, etc.
4. Mostrar el punto mediante colores dependiendo de su propiedad
5. Mostrar información (etiqueta y propiedades)

2. Reconstruir la superficie de la cima (Proceso iterativo)
a. Elegir el método de reconstrucción de superficie
i. Elegir la dimensión de la retícula
ii. Interpolación bivariada
iii. Suavizamiento por B-splines multinivel (el usuario define el número de niveles))
b. Mapa de contornos de la superficie (iterativo)
i. El usuario asigna el número de curvas de nivel
c. Mostrar la superficie (sabana 3D)*
d. Validar superficie (Se puede validad a partir de 2 cimas)
3. Construir la región de estudio
a. Definir contornos de control
i. Se pueden definir curvi-líneas de control horizontales
ii. Se pueden definir curvi-líneas de control verticales
iii. (Observación) Las curvi-líneas de una misma clase no se pueden cortar(intersecar)
iv. (Observación) Siempre existen dos sub-fronteras opuestas más externas
v. (Nota mental) Ya definidas dos líneas de control de una clase, opuestas más externas, el usuario podrá definir las otras dos líneas de control de la clase faltante, iniciando solamente por los vértices de las líneas

Tareas

iniciales.
   vi. (Observación) Las líneas de control llevan un orden de recorrido en sentido contrario a las manecillas del reloj
   vii. Todas las líneas de control internas deben tener sus vértices extremos externos en las líneas de control más externas(contorno de control)
b. Editar los contornos de control
   i. Mueves puntos, añades puntos, quitas puntos
   ii. Quitas contornos
c. Suavizar los contornos de control (suavizar e interpolar vía tratamiento de contornos)
   i. (Idea) Definir los puntos extremos del contorno de control y el numero de puntos internos
d. Definir sub-regiones
   i. Definir el numero de puntos sobre cada sub-frontera de cada subregión

4. Construir la malla 2D
   a. Construir la malla de interpolación sobre cada subregión
      i. Se cuenta con la distribución de puntos sobre cada sub-frontera de cada subregión
      ii. Obtener la dimensión de la subregión
      iii. Pegar la malla de cada subregión para obtener la malla 2D(Si los contornos de control son tipo fallas)
      iv. (Opción) Se construye de manera adecuada las 4 fronteras externas de la región de estudio y se construye la malla de interpolación(solo si los contornos de control son para refinar)
   b. Calidad de la malla (¿que tan buena es la malla? Cíclico)
      i. Verificar que la malla sea conexa (Rutina nonij.f90)
      ii. Verificar que tan uniforme es en área
      iii. Verificar que tan uniforme es en ortogonalidad
   c. Obtener una malla suave y convexa sobre la región de estudio
      i. Entregar cada sub-malla al modulo de suavizamiento de mallas
      ii. (Opción) Entregar la malla 2D al modulo de suavizamiento de mallas
   d. Refinamiento uniforme de la malla 2D
   e. Asignar propiedades a la malla (ej. Nodos de control, líneas de control, se pueden colocar restricciones)
   f. Editar los nodos de la malla (ej. Mover nodos, asignar pozos)
   g. Volver a suavizar (con posibilidad de restricciones)
   h. NOTAS: Se Puede usar una malla de referencia para:
      i. Elegir una sub-malla y
         1. Hacer un refinamiento local
         2. Asignar celdas activas e inactivas
         3. Asignar propiedades(pozos)
      ii. Editar líneas(eliminar líneas)
5. Construir la malla 3D
   a. Se cuenta con una malla 2D
   b. Se levanta la malla 2D sobre cada una de las cimas
      i. Se cuenta con la cima k reconstruida por z = f_k(x,y) (interpolación o suavizamiento)
      ii. Los datos de la malla 2D siguen la estructura vectorial xij, yij

Tareas
iii. Para cada k se construye un nodo P(i, j, k) de la malla 3D, de forma p(i, j, k)=(xij, yij, f_k(xij, yij)) c. Se visualiza la malla 3D d. Calidad de la malla 3D (convexidad, ortogonalidad, -¿que tan buena es?-) Aplicar criterios geométricos que miden la calidad de la malla, similares al caso de la malla 2D

Al cierre de la presente fase se cuenta con un programa prototipo, el cual nos permite editar mapas de contornos de datos y suavizar dichos contornos por el módulo: Tratamiento de Contornos entregado durante la fase anterior.

Actualmente continuamos trabajando con el Grupo de Interfaz y Visualización para discutir las tareas que tienen que ver con las rutinas de enlace de información entre el Simulador Numérico y nuestros módulos. Durante las próximas semanas se espera que el trabajo se intensifique con el fin de obtener un sistema con las herramientas esenciales descritas a lo largo del proyecto.

Apéndice K: Instancias del manejo de las rutinas utilizadas

K.1 Instancias de las rutinas que generan las mallas planas por Interpolación Transfinita y las suavizan por el método discreto AO

MÉTODO	PARÁMETROS	VALOR DEVUELTO	DESCRIPCIÓN
MANEJO DE INSTANCIAS			
SUBROUTINE menuop(mn,il,jl,n,idiwa,iww, iwa,ww,red,filini);	MN -total de puntos en la malla. IL -número de puntos en la vertical de la malla. JL -número de puntos en la horizontal de la malla. N -número de puntos interiores de la malla. IDIWA -dimensión del arreglo IWA. IWW -dimensión del arreglo WW. IWA -arreglo de dimensión IDIWA. WW -arreglo de dimension IWA.	RED – contiene los puntos de la malla. FILINI – nombre de la última malla inicial.	Este módulo coordina la generación de una malla óptima.
SUBROUTINE fileio (mn,ibo,il,jl,noi,noj, index,iflag,filnam, red,inocon,xmin,xmax, ymin,ymax,alfapreal, action,alfaprom, alfamin,alfamax, alfa_lim)	IBO -total de puntos en la frontera. FILNAM -nombre de archivo. INDEX -Lee y escribe el archive red de entrada y salida.	RED, IFLAG – bandera de salida para verificar si hay error.	Este módulo hace la asignación del archivo de entrada o salida para leer o escribir los puntos de la malla.
SUBROUTINE nonij(mn,ibo,il,jl,red,noi, noj,inocon,alfamin, alfamax,alfaprom, alfa_lim)	INOCON –número celdas no convexas. IBO,MN,INI	INOCON	Esta subrutina calcula cuantas celdas no convexas contiene la malla.
SUBROUTINE asigna(mn,ibo,il,jl,i,j, red,p,ini)	MN,IBO,IL,JL, I –subíndice que indica la absisa de la celda. J –subíndice que indica la ordenada de la celda. RED –arreglo que contiene los puntos de la malla.	P,INI	Dados los índices i,j de un punto, este modulo devuelve el apuntador a la posición de dicho punto en el arreglo unidimensional RED, asi como las coordenadas del punto en P(1) y P(2)

MÉTODO	PARÁMETROS	VALOR DEVUELTO	DESCRIPCIÓN
`DOUBLE PRECISION FUNCTION area(c,b,a)`	`A` –vector de coordenadas en el punto A. `B` –vector de coordenadas en el punto B. `C` –vector de coordenadas en el punto C.	Valor de: `area(c,b,a)`	Esta función calcula el área de un triángulo teniendo en cuenta la orientación positiva (en contra de las manecillas del reloj).
`SUBROUTINE icheq(a1,a2,a3,a4,iband, alfa_lim)`	`A1` –área del primer triángulo de la malla. `A2` –área del segundo triángulo de la malla. `A3` –área del tercer triángulo de la malla. `A4` –área del cuarto triángulo de la malla.	`IBAND` – bandera para verificar la convexidad.	Esta subrutina verifica la convexidad de las celdas que no están en las esquinas de la región.
`SUBROUTINE admred(mn,ibo,il,jl,inocon, red,ssarea,scarea)`	`MN,IBO,IL,JL, RED`	`INOCON, SSAREA` - suma de las áreas en las esquinas. `SCAREA` - suma del cuadrado de las áreas en las esquinas.	Esta subrutina calcula el área de las cuatro celdas que están en las esquinas de la región para chequear su admisibilidad.
`SUBROUTINE scale(mn,red,xmin,xmax,ymin, ymax,action,alfapreal)`	`MN,RED,` `XMIN` –valor minimo de X. `XMAX` –valor maximo de X. `YMIN` –valor minimo de Y. `YMAX` –valor maximo de Y.	`RED`	Este módulo escala los valores de la malla inicial a la región [0,1]x[0,1] y/o los reestablece a su escala física.

`SUBROUTINE solver(mn,ibo,il,jl,n,iww, id3,iprint,sigma,red, ww,ires,filename, ifcng,inocon,idiwa, iwa,alfa_lim,alfaprom, chamet,xmin,xmax,ymin, alfapreal,action, alfamin,alfamax)`	`MN,IBO,IL,JL,N,IWW,`
	`ID3` –dimensión del vector iprint.
	`IPRINT` –especifica la frecuencia de salida.
	`SIGMA` –peso del funcional.
	`RED,WW`
	`IFCNG` –parámetro para elegir funcional.
	`IDIWA,`
	`ALFA_LIM` –valor minimo para el cual se considera que el área de una celda es positiva.
		`RED,G,`
		`IRES` – bandera que da la forma de salida.
		`INOCON, FILNAM`
			Este módulo prepara la entrada a la rutina de optimización que resuelve el problema de minimización proveniente de la generación de mallas variacional.

MÉTODO	PARÁMETROS	VALOR DEVUELTO	DESCRIPCIÓN
SUBROUTINE guiatron(mn,ibo,il,jl,n,iww, idiwa,nnz,id3, iprint,sigma,red, g,xc,adiag,bdiag, ldiag,xl,xu,a,ifcng, arow_ind,acol_ind, arow_ptr,acol_ptr, epsmch,listp,ngrp, idsp,y,eta,wa1,ga, w,noi,noj,iwa,lim_k, alfaprom,epsf,epsg, alfa_lim,ismooth, kglobal,f,gnorm, iter,nflag,kfunc, alfapreal,k_cycle, alfamin,alfamax)	MN,IBO,IL,JL,N,ID3, IPRINT,SIGMA,RED, W –arreglo de trabajo. FACTA –escalar para la normalización del funcional de área o para el de Ivanenko. FACTL –escalar para la normalización del funcional de longitud o para el de AO.	G,RED	Este módulo coordina la ejecución del método de Newton con Región de Confianza, hecha punto por punto, es decir, se busca el óptimo del funcional sobre toda la malla, pero sólo se considera variable P(i,j) y el resto de los puntos fijos, y así se sigue sobre toda la malla, excepto para aquellos puntos que el usuario asumió que estarían siempre fijos en el valor de la malla inicial.
SUBROUTINE calfacto(ifcng,il,jl, alfaprom,factor1, factor2)	OP, IL,JL, ALFAPROM –alfa promedio de la red.	FACTOR1 - Factor de normalizaci on para el primer funcional de la combinació n. FACTOR2 - Factor de normalizaci on para el segundo funcional de la combinació n.	Este módulo consta de una función que calcula el factor de normalización del funcional de Area-ortogonalidad.
SUBROUTINE caltim(hri,hrf,mii,mif, sei,sef,hdi,hdf)	HRI –hora inicial. MII –minuto inicial. SEI –segundo inicial. HDI –cien segundos inciales.		Subrutina que calcula el tiempo de ejecución de un programa.

MÉTODO	PARÁMETROS	VALOR DEVUELTO	DESCRIPCIÓN
`SUBROUTINE cambiod(M,XB,YB,NB1)`	`M` –número de puntos en el contorno. `XB` –absisas del contorno en el sentido original. `YB` –Ordenadas del contorno en el sentido original. `NB1` –número de puntos en la primera frontera.	`XB,YB`	Subrutina que cambia la orientación de la frontera cambiando la distribución de puntos.
`SUBROUTINE carea(ibo,red,areag)`	`IBO,RED`	`AREAG` –área de la región.	Esta subrutina calcula el área total de una región.
`DOUBLE PRECISION FUNCTION cheqex(x)`	`X` –contiene el exponente que se va a verificar.	Contiene la exp(x)	Función que verifica los valores permitidos de la función exponencial.
`SUBROUTINE convex(ND1,IL,JL,NP,INOCON, WXF,WYF,SAREA,SCAREA)`	`ND1` –dimensión de los arreglos WXF,WYF. `IL,JL,INOCON,` `WXF` –coordenadas X's sobre los puntos de la frontera. `WYF` –coordenadas Y's sobre los puntos de la frontera.		Subrutina que verifica la convexidad de la malla.
`SUBROUTINE corner(il,jl,i,j,a1,a2,a3, a4,iband,alfaprom)`	`IL,JL,I,J,A1,A2,A3,A4`	`IBAND`	Esta subrutina verifica la convexidad de las cuatro celdas de las esquinas.
`SUBROUTINE daxpy(n,da,dx,incx,dy,incy)`	`N` –variable entera. `DA` –variable de doble precisión. `DX` –variable de doble precisión. `INCX` –variable entera. `DY` –variable de doble presición. `INCY` –variable entera.		Constant times a vector plus a vector.

MÉTODO	PARÁMETROS	VALOR DEVUELTO	DESCRIPCIÓN
SUBROUTINE dbreakpt(n,x,xl,xu,w,nbrpt, brptmin,brptmax)	N –número de variables. X –vector de dimensión n. XL –vector de cotas inferiores. XU –vector de cotas superiores. W –vector de dimension n. NBRPT –número de puntos malos. BRPTMIN –mínimo de puntos malos. BRPTMAX –máximo de puntos malos.		Esta subrutina calcula el número de puntos malos y el mínimo y máximo de puntos malos de la proyección x + alpha*w sobre el intervalo n-dimensional [x1,xu].
SUBROUTINE dcauchy(n,x,xl,xu,a,diag, col_ptr,row_ind, g,delta,alpha,s,wa)	N,X, XL –vector de cotas inferiors. XU –vector de cotas superiores. A –arreglo que contiene los elementos de una matriz triangular inferior. DIAG –contiene los elementos de la diagonal de A. COL_PTR –arreglo de dimension n+1, contiene los elementos de las columnas de A. ROW_IND –arreglo de dimension NNZ, contiene los elementos de la matriz triangular inferior. G –arreglo de dimensión N, contiene al gradiente. DELTA –tamaño de la region de confianza. ALPHA –estimación actual del paso de Cauchy. S –paso de Cauchy. WA –arreglo de dimensión N.		Esta subrutina calcula un paso de Cauchy que satisface una región de confianza y una condición suficientemente decreciente.
SUBROUTINE dcopy(n,dx,incx,dy,incy)	N,INCX,INCY		Esta subrutina copia un vector x a un vector y.
DOUBLE PRECISION FUNCTION ddot(n,dx,incx, dy,incy)	N,INCX,INCY	Nos devuelve el producto de dos vectores en ddot	Esta función efectúa el producto interno de dos vectores.

MÉTODO	PARÁMETROS	VALOR DEVUELTO	DESCRIPCIÓN
SUBROUTINE degr(n,indrow,jpntr,indcol, ipntr,ndeg,iwa)	N, INDROW –arreglo con las filas no cero de la matriz A. JPNTR –especifica la localización de los indices de las filas en INDROW. INDCOL –arreglo con las columnas no cero de A. IPNTR – especifica la localización de los indices de las filas en INDCOL. NDEG –especifica el grado de la sucesión. IWA		Dada una matriz A sparse de m por n, esta subrutina determina el grado de la sucesión de la intersección gráfica de las columnas de A.
SUBROUTINE dgpstep(n,x,xl,xu,alpha,w,s)	N,X,XL,XU,ALPHA,W,S		Esta subrutina calcula el paso de la proyección gradiente.
SUBROUTINE dicf(n,nnz,a,diag,col_ptr, row_ind,p,info,indr, indf,list,w)	N, NNZ, A,DIAG,COL_PTR, ROW_IND, P –memoria disponible para la factorización incompleta de Cholesky. INFO –nos da información sobre la factorización. INDR –arreglo de trabajo de dimension N. INDF –arreglo de trabajo de dimension N. LIST –arreglo de trabajo de dimension N. W		Dada una matriz A simétrica sparse en una fila, esta subrutina calcula la factorización incompleta de Cholesky.

MÉTODO	PARÁMETROS	VALOR DEVUELTO	DESCRIPCIÓN
SUBROUTINE dicfs(n,nnz,a,adiag,acol_ptr, arow_ind,l,ldiag, lcol_ptr,lrow_ind, p,alpha,iwa,wa1,wa2)	N –orden de la matriz A. NNZ –número de entradas no cero de la matriz triangular inferior. A, ADIAG –contiene los elementos de la diagonal A. ACOL_PTR –contiene los elementos de las columnas de A. ARROW_IND –contiene los elementos de las filas de A. L –contiene los elementos de la parte triangular inferior. LDIAG –contiene los elementos de la diagonal de L. LCOL_PTR –contiene los puntos de las columnas de L. LROW_IND –contiene las filas de la parte triangular inferior de L. P –memoria disponible para la factorización de Cholesky. ALPHA –desplazamiento. IWA, WA1 –arreglo de trabajo de doble presición. WA2 –arreglo de trabajo de doble presición.		Dada una matriz A simétrica sparse en una columna, esta subrutina calcula la factorización incompleta de Cholesky.
SUBROUTINE dmid(n,x,xl,xu)	N,X,XL,XU		Esta subrutina calcula la proyección de x sobre el intervalo n-dimensional [xl,xu].
DOUBLE PRECISION FUNCTION DNRM2 (N, X, INCX)	N,X,INCX	DNRM2 := sqrt(x'*x)	Esta función nos da la norma euclidiana de un vector.
DOUBLE PRECISION FUNCTION dpmeps()		dpmeps = a	Esta función calcula el parámetro de presición de la máquina.

MÉTODO	PARÁMETROS	VALOR DEVUELTO	DESCRIPCIÓN
SUBROUTINE dpofa(a,lda,n,info)	A –matriz simétrica factorizada. LDA –dimensión principal del arreglo A. N	A,INFO	Esta subrutina calcula un factor de doble precision de una matriz simétrica positiva definida.
SUBROUTINE dprsrch(n,x,xl,xu,a,diag, col_ptr,row_ind,g,w, wa1,wa2)	N,X,XL,XU,A,DIAG, COL_PTR,ROW_IND,G,W, WA1,WA2		Esta subrutina usa una búsqueda proyectada para calcular un paso que satisface una condición suficientemente decreciente para la cuadrática q(s) = 0.5*s'*A*s + g'*s.
PROGRAM Driv_Gen_Mall	No hay parámetro de entrada	Como salida nos devuelve una malla optimizada con AO	Este programa coordina la optimización de una malla inicial dada por el usuario. Aqui solo se lee el tamaño de la malla.
SUBROUTINE dscal(n,da,dx,incx)	N,DA,DX,INCX		Esta subrutina escala un vector por una constante.
SUBROUTINE dsel2(n,x,keys,k)	N,X, KEYS –arreglo de dimension n. K –es el k-ésimo elemento más grande.		Dado un arreglo x, esta subrutina permuta los elementos del arreglo.
SUBROUTINE dsetsp(n,nnz,row_ind,col_ind, col_ptr,row_ptr,method, info,listp,ngrp,maxgrp, iwa)	N,NNZ,ROW_IND,COL_IND, COL_PTR,ROW_PTR, METHOD –parámetro para determinar el método a usar. INFO LISTP –especifica la permutación de la matriz. NGRP –especifa la partición de las columnas de A. MAXGRP –número de grupos en la partición de las columnas de A. IWA		Dado los elementos no cero de una matriz simétrica A en formato de coordenadas, esta subrutina calcula la información para determinar A triangular inferior por el método de sustitución.

MÉTODO	PARÁMETROS	VALOR DEVUELTO	DESCRIPCIÓN
SUBROUTINE dspcg(n,x,xl,xu,a,adiag, acol_ptr,arow_ind,g, delta,rtol,s,nv,itermax, iters,info,b,bdiag, bcol_ptr,brow_ind,l, ldiag,lcol_ptr,lrow_ind, indfree,gfree,w,wa,iwa)	N,X,XL,XU,A,ADIAG, ACOL_PTR,AROW_IND,G, DELTA -tamaño de la region de confianza. RTOL -especifica la precisión del mínimo final. S -paso de Cauchy NV -memoria de la factorización incompleta de Cholesky. ITERMAX -limite sobre el número de iteraciones del gradiente conjugado. ITERS -conjunto del número de iteraciones del gradiente conjugado. INFO, B -parte triangular inferior de B. BDIAG -contiene los elementos de la diagonal de B. BCOL_PTR -contiene puntos de las columnas de B. BROW_IND -contiene las filas de la parte triangular inferior de B. L,LDIAG,LCOL_PTR LROW_IND -contiene puntos de las filas de la parte triangular inferior de L. INDFREE -Arreglo entero de trabajo de dimensión N. GFREE -Arreglo de trabajo de doble precision de dimensión N. W,WA,IWA		Esta subrutina genera una sucesión de aproximación al mínimo para el subproblema min { q(x) : xl <= x <= xu }.
SUBROUTINE dsphesd(n,row_ind,col_ind, row_ptr,col_ptr, listp,ngrp,maxgrp, numgrp,eta,fhesd, fhes,diag,iwa)	N,ROW_IND,COL_IND, ROW_PTR,COL_PTR, LISTP,NGRP,MAXGRP, NUMGRP -conjunto de un número de grupos. ETA -diferencia del vector parámetro. FHESD -arreglo de doble precisión de longitud N. FHES -Arreglo de doble precisión de longitud NNZ. DIAG,IWA		Esta subrutina calcula una aproximación a la matriz Hessiana (simétrica) de una función por un método de substitución.

MÉTODO	PARÁMETROS	VALOR DEVUELTO	DESCRIPCIÓN
`SUBROUTINE dsscfg(nx,ny,x,f,fgrad, task,lambda)`	`NX,NY,X,` `F` -conjunto de funciones evaluadas en X. `FGRAD` -contiene el gradiente evaluado en X. `TASK` -especifica la acción de la subrutina.		Esta subrutina calcula la función y el gradiente del problema de combustión en equilibrio.
`SUBROUTINE dssm(n,npairs,indrow,indcol, method,listp,ngrp, maxgrp,mingrp,info,ipntr, jpntr,iwa,liwa)`	`N,` `NPAIRS,` `INDROW,` `INDCOL,` `METHOD,LISTP,` `NGRP,MAXGRP,MINGRP,` `INFP,IPNTR,JPNTR,IWA,` `LIWA` -Entero positivo, que no es menor que 6*n.		Dado el modelo sparse de una matriz simétrica A de orden n, esta subrutina determina una permutación simétrica de A y una partición de las columnas de A que consiste de determinar A como una matriz triangular inferior por el método de substitución.
`SUBROUTINE dssyax(n,a,adiag,jptr, indr,x,y)`	`N,A,ADIAG,JPTR,` `INDR,X,Y`		Esta subrutina calcula el producto matriz-vector y = A*x.
`SUBROUTINE dstrsol(n,l,ldiag,jptr, indr,r,task)`	`N,L,LDIAG,JPTR,` `INDR,R,TASK`		Esta subrutina resuelve el sistema triangular L*x = r o L'*x = r.

MÉTODO	PARÁMETROS	VALOR DEVUELTO	DESCRIPCIÓN
SUBROUTINE dtron(n,x,xl,xu,f,g,a,adiag, acol_ptr,arow_ind, frtol,fatol,fmin,cgtol, itermax,delta,isel, b,bdiag,bcol_ptr, brow_ind,l,ldiag, lcol_ptr,lrow_ind, xc,s,indfree,isave, dsave,wa,iwa,bandtu, cond,ilcu,epsmch)	N,X,XL,XU,F,G,A,ADIAG, ACOL_PTR,AROW_IND, FRTOL –especifica el error relatiVo de la función. FATOL –especifica el error de la función. FMIN –especifica una cota inferior de la función. CGTOL –especifica el criterio de convergencia para el método de gradiente conjugado. ITERMAX –especifica el limite del número de iteraciones de gradiente conjugado. DELTA,ISEL,B,BDIAG, BCOL_PTR,BROW_IND, L,LDIAG,LCOL_PTR, LROW_IND, XC –arreglo de trabajo de doble precisión, de dimensión N. S,INDFREE, ISAVE –arreglo entero de trabajo de dimensión 3. DSAVE –arreglo de trabajo de doble precisión, de dimensión 3. WA,IWA, BANDTU –variable lógica. COND –variable caracter de longitud 1. ILCU –variable caracter de longitud 1. EPSMCH –variable doble precisión.		Esta subrutina implementa un método de Newton de región de confianza para la solución de problemas de optimización acotados.
SUBROUTINE dtrpcg(n,a,adiag,acol_ptr, arow_ind,g,delta,l, ldiag,lcol_ptr, lrow_ind,tol,stol, itermax,w,iters,info, p,q,r,t,z)	N,A,ADIAG,ACOL_PTR, AROW_IND,G,DELTA,L, LDIAG,LCOL_PTR, LROW_IND, TOL, STOL, ITERMAX,W,ITERS,INFO, P,Q,R,T,Z	INFO	Dada una matriz A sparse simétrica en una columna, esta subrutina usa un método de gradiente conjugado precondicionado para encontrar una paroximación al mínimo del subproblema de región de confianza min { q(s) : || L'*s || <= delta }.

MÉTODO	PARÁMETROS	VALOR DEVUELTO	DESCRIPCIÓN
SUBROUTINE dtrqsol(n,x,p,delta,sigma)	N,X,P,DELTA,SIGMA		Esta subrutina calcula la solución más grande (no-negativa) de la ecuación de la región de confianza: ||x + sigma*p|| = delta.
SUBROUTINE escres(n,id3,iprint,iter,ifun, igra,inocon,gnorm,epsg, epsf,sigma,tao,x,f,g, finish,ipri,ifcng)	N,ID3,IPRINT,ITER, IFUN -número de veces que se evalua la función. IGRA -número de veces que se evalua el gradiente. EPSG -tolerancia del criterio del gradiente. EPSF -tolerancia del criterio de la función. SIGMA,X,G		
SUBROUTINE fao(mn,ibo,il,jl,n,f,g, red,isel,factl)	MN,IBO,IL,JL,N,RED, FACTL	F,G,INOCON	Esta subrutina calcula el valor del funcional de A-O en un vector x de puntos interiores de una malla.
SUBROUTINE fhfv(p,q,r,s,qp,rq,sr,ps, fh,fv)	P,Q,R,S	FH - evaluación de la parte horizontal del funcional de área ortogonalidad para la celda dada. FV - evaluación de la parte vertical del funcional de área ortogonalidad para la celda dada. QP,RQSR,PS	Esta subrutina evalua la parte horizontal y vertical del funcional de área-ortogonalidad para la celda de vértices PQRS.

MÉTODO	PARÁMETROS	VALOR DEVUELTO	DESCRIPCIÓN
SUBROUTINE fyg(mn,ibo,il,jl,inocon,sigma, red,g,f,n,tao,ifcng,epsa, k1,isel,facta,factl,ga, alfaprom,alfa_lim)	MN,IBO,IL,JL,INOCON, SIGMA,RED,N, TAO -valor del parámetro del funcional Barrera-Perez. IFCNG -parámetro para elegir el funcional. EPSA -valor usado para el funcional de k-Suavidad. KL -variable de doble precisión. ISEL,FACTA, FACTL -escalar para la normalización del segundo funcional. GA -variable de doble precisón para un espacio de memoria. ALFAPROM,ALFA_LIM	INOCON,F,G	Esta subrutina calcula la función escalar f sobre la malla actual y, si es necesario, su vector gradiente.
SUBROUTINE gao(qp,rq,sr,ps,g1,g2, g3,g4,fh,fv)	P,Q,R,S	G1,G2,G3, G4,FH,FV	Esta subrutina calcula el valor del funcional A-O de una celda PQRS y sus derivadas correspondientes con respecto a p, q, r, s.
SUBROUTINE gpabtr(p,q,r,g1,g2,g3,f, pqr,isel)	P,Q,R,PQR		Esta subrutina calcula las derivadas respecto a cada tres puntos que determinan el triángulo PQR.
SUBROUTINE INTERV(XT, LXT, X, LEFT, MFLAG)	XT -sucesión real de dimension LXT. LXT -número de terminos en la sucesión XT. X -es el punto localizado respecto a la sucesión XT.		Esta subrutine calcula: LEFT = MAX(i;1 .le. i .le. LXT .and. XT(i) .le. X).
DOUBLE PRECISION FUNCTION LNVALU(XYB,TF, I,M,L, TFI,H, HXY)	TF -variable de doble precisón con los puntos parametrizados. TFI -punto en el cual se evaluará.	LNVALU, de F en X	Esta subrutina llama INTERV y calcula el valor en X.

MÉTODO	PARÁMETROS	VALOR DEVUELTO	DESCRIPCIÓN
SUBROUTINE MENUGE(MN,IL,JL,idiwa,RED, iwa,FNAMEI)	IL,JL,FNAMEI	RED	Esta subrutina genera una malla inicial, para regiones en dos dimensiones.
SUBROUTINE menugene(MN,IL,JL,RED,M, IBFLAG,NB1,NB2,NB3, NB4,XB, YB, INFO,WTB, WXF,WYF,DY,XFT,YFT)	IL,JL,RED,M,NB1,NB2, NB3,NB4	RED	Esta subrutina genera una malla inicial, para regiones en dos dimensiones.
SUBROUTINE MODULO(M,MARCA,ZB,ZBT)	M, MARCA –punto inicial. ZB – vector con los puntos en la posición anterior. ZBT – vector con los puntos en la nueva posición.		
SUBROUTINE NCOREG(MN,M,MP,NB1,NB2,NB3, NB4,IL,JL,ND1,ND4, IBFLAG,NINFR,ND5,ND6, XF,YF,WTB,XB,YB,RED, IBFAIL)	MN,M,NB1,NB2,NB3, NB4,IL,JL,ND1,ND4, NINFR,ND5,ND6	RED,WTB, XF,YF,XB, YB	Esta subrutina genera la malla inicial para una región no conexa.
SUBROUTINE pabg(il,jl,n,g,fj,u,v,factor)	IL,JL,N,G, FJ -gradiente respecto al punto P(u,v). U –índice i del punto respecto al cual se deriva. V –índice j del punto respecto al cual se deriva. FACTOR –escalar constante por el que se multiplica cada componente del gradiente.	G	Esta rutina coloca en la posición correcta el gradiente con respecto a el punto (u,v).
SUBROUTINE PARLIN(ND1,M,NPB,INIC1,INIC2, FIN,NPOINT,J3,XB,YB,TF, XF,YF)	ND1,M, XB –coordenadas Xi's de los puntos de la malla. YB –coordenadas Yi's de los puntos de la malla. NPOINT –número de puntos. TF –vector con los puntos parametrizados.	XF,YF	

MÉTODO	PARÁMETROS	VALOR DEVUELTO	DESCRIPCIÓN
SUBROUTINE pkhpk(n,p,g,x,hpk,xk1,acc, pnorm,php,ient)	N,P,G,X, ACC –precisión de la máquina. PNORM – cuadrado de la norma del vector P. PHP – valor de la cuadrática pk'Hkpk. IENT –bandera.	HPK,PHP, IENT,X,XKL	Esta subrutina calcula la forma cuadrática pk'Hpk.
SUBROUTINE posder2(i,j,col,il,jl,nnz, arow_ind,acol_ind, sigma,jj)	I,J,COL,IL,JL,NNZ, AROW_IND,ACOL_IND, SIGMA,JJ		Esta rutina da el índice de elementos no cero del Hessiano.
SUBROUTINE posder(n,il,jl,nnz,arow_ind, acol_ind,sigma,nzeros)	N,IL,JL,NNZ,AROW_IND, ACOL_IND,SIGMA,NZEROS		Hace un llamado a posder2.
SUBROUTINE red_ini(mn, marca, npoint, il, jl, nd1, nd4, nd5,mp, ninfr, xft, yft, red)	MN,IL,JL,ND1,ND4,ND5, NINFR –vector con la posición de los puntos de las esquinas.	XFT,YFT, RED	Programa para generar la red inicial mediante el metodo algebraico TFI (Interpolacion Transfinita).
SUBROUTINE SEGRED(MN,IL,JL,ND1,ND2,ND4, ND6,RED,WTB,WXF,WYF,DY, MP,XFT,YFT,XB,YB,M, IBFLAG,NB1,NB2, NB3,NB4,BBFLAG)	MDIM –mayor número de puntos en la vertical. IL,JL,M,ND1,ND2, ND4,FNAME	RED,WTB, WXF,WFY,DY	
SUBROUTINE SELGEN(MN,MP,M,N1,IL,JL,ND1, ND2,ND4,ND5,ND6,NINFR, NP,XFT,YFT,WXF,WYF,WTB, RED,DY,XB,YB,IBFAIL)	MN, MP, M,N1,IL,JL,ND1, ND2,ND4,ND5,ND6,NP, WTB –arreglo de trabajo unidimensional. WXF –arreglo de trabajo unidimensional par alas coordenadas Xi's. WYF –arreglo de trabajo unidimensional par alas coordenadas Yi's. DY –arreglo de trabajo unidimensional. NP –número de puntos para la interpolación. IA –bandera para cambiar o no la dirección del contorno. RED	RED	Esta subrutina coordina la generación de mallas simplemente conexas.

K.2 Instancias del manejo de rutinas para el sub-módulo de reconstrucción de superficies empleando el método de Interpolación Bivariada

MÉTODO	PARÁMETROS	VALOR DEVUELTO	DESCRIPCIÓN
MANEJO DE INSTANCIAS			
SUBROUTINE idbvip (md, ndp, xd, yd, zd, nip, xi, yi, zi)	MD –modo de computación. Toma valor 1,2 o 3, en otro caso hay error. NDP –número de datos. XD,YD –coordenadas de los puntos de la malla. ZD –valores de los puntos de la malla. NIP –número de puntos que salen de la interpolación. XI,YI –coordenadas de los puntos en los cuales se interpola.	ZI –valores interpolados	Esta subrutina lleva a cabo la interpolación vibariada de los datos irregulares X, Y.
SUBROUTINE idgrid (xd, yd, nt, ipt, nl, ipl, nxi, nyi, xi, yi, ngp, igp)	XD,YD NT –número de triángulos IPT –índices de los vértices de los triángulos NXI,NYI –número de puntos en las coordinas X y Y XI,YI	NGP – número de de puntos de la malla que pertenecen a cada triángulo IGP – número de puntos de la malla que se guardan en orden ascendiente	Esta subrutina organiza los puntos de la malla adecuadamente en la superficie.
SUBROUTINE idlctn (ndp, xd, yd, nt, ipt, nl, ipl, xii, yii, iti)	NDP,XD,YD,IPT,NL,IPL XII,YII –coordenadas del puntos localizado	ITI – número del triángulo	Esta subrutina encuentra el triángulo que contiene un punto.
SUBROUTINE idpdrv (ndp, xd, yd, zd, nt, ipt, pd, wk)	NDP,XD,YD,ZD,IPT	PD	Esta subrutina estima la primera y segunda derivada parcial en puntos respectivos.

MÉTODO	PARÁMETROS	VALOR DEVUELTO	DESCRIPCIÓN
SUBROUTINE idptip (ndp,xd, yd, zd, nt, ipt, nl, ipl, pdd, iti, xii, yii, zii)	NDP,XD,YD,ZD,NT,IPT, NL,IPL PDD –derivadas parciales de los puntos. ITI,XII,YII	ZII – valores interpolados	Esta subrutina lleva a cabo la interpolación, determinando un valor de Z dando X y Y.
SUBROUTINE idsfft (md, ndp, xd, yd, zd, nxi, nyi, nzi, xi, yi, zi)	MD,NDP,XD,YD,ZD,NXI, NYI,NZI,XI,YI	ZI	Esta subrutina genera una superficie suave Z(X,Y) dado los datos irregulares (X,Y,Z).
SUBROUTINE idtang (ndp, xd, yd, nt, ipt, nl, ipl, iwl, iwp, wk)	NDP,XD,YD	NT,IPT,NL, IPL	Esta subrutina lleva a cabo la triangulación de datos irregulares (X,Y,Z).
FUNCTION idxchg (x, y, i1, i2, i3, i4)	X,Y, I1,I2,I3,I4 –número de puntos de los cuatro puntos P1,P2,P3,P4 que forman un cuarilátero con P3 y P4 conectados por una diagonal.	IDXCHG – reporta donde los triángulos deben cambiarse	Esta subrutina determina uno de los dos triángulos que debe cambiarse.
PROGRAM main			Programa que genera una malla dado un conjuto de puntos 3d, usando el algoritmo de interpolación bivariada de Hiroshi Akima.
SUBROUTINE loaddata			Función que pide el nombre del archivo de datos 3d, en donde es necesario que en el primer renglón esté el número de puntos y en el segundo el dominio de trabajo para x y y.

MÉTODO	PARÁMETROS	VALOR DEVUELTO	DESCRIPCIÓN
`SUBROUTINE savemesh(f_name,nd,xd,yd,zd, a,b,c,d)`	`ND,XD,YD,ZD, A,B,C,D` –valores del dominio de trabajo	`F_NAME` – nombre del archive donde se guardará la malla	Esta función se encarga calcular los puntos 2d de la malla (xi,yi) de acuerdo a los parámetros a,b,c,d y llama a la función idbvip que obtiene la interpolación bivariada f(x,y) y los puntos zi tales que f(xi,yi)=zi para toda i. Una vez hecho esto, graba los puntos (xi,yi,zi) en un archivo cuyo nombre es proporcionado por el usuario. De preferencia éste debe tener extensión mesh.
`SUBROUTINE quita_iguales(xd,yd,zd,nd)`	`XD` - arreglo de nd reales que representan los valores de x. `YD` – arreglo de nd reales que representan los valores de y. `ZD` – arreglo de nd reales que representan los valores de z. `ND` – número de reales.		Subrutina que quita puntos xd,yd,zd iguales.

K.3 Instancias del manejo de rutinas para el sub-módulo de reconstrucción de superficies empleando el suavizamiento de datos a través de B-splines usando una técnica multinivel

MÉTODO	PARÁMETROS	VALOR DEVUELTO	DESCRIPCIÓN
MANEJO DE INSTANCIAS			
mba_ref	Fichero de entrada con extensión DAT	Fichero de salida con extensión DAT	Programa Multilevel B-Spline Aproximation ref. Basado en el algoritmo descrito en: "Scattered Data Interpolation with Multilevel B-Spline"
SUBROUTINE norma_1(nc,deltzc,norm_zc)	NC –número de datos de entrada DELTZC –diferencia entre la aproximación y el valor de ZC	NORM_ZC – norma del vector DELTZC	Subrutina para definir criterios de paro.
SUBROUTINE FI_to_thet(md,nd,thet,fi)	MD –dimensió de X de la malla ND –dimensión Y de la malla FI –latice de control	THET – latice de control global	Renombra el latice de control
SUBROUTINE sum_thet(md,nd,thet,fipri,fi)	MD,ND FIPRI –latice de control local de 4x4 FI	THET – latice de control global	Construye puntualmente el latice global
SUBROUTINE ref(md,nd,FI,fipri)	MD,ND FI	FIPRI	Construye el refinamiento del latice del nivel anterior
FUNCTION ff_lin(x,y,a)	X,Y,A	FF_LIN	Subrutina que dado el vector de coeficientes lineales y la posición X,Y devuelve la entrada Z sobre un plano
FUNCTION ff(x,y,fi,md,nd)	X,Y,FI,MD,ND	FF –función evaluada	Evalua la función sobre la malla rectangular
SUBROUTINE bs_m(md,nd,nc,xc,yc,zc,FI)	MD,ND,NC XC,YC,ZC –datos de entrada FI		Subrutina que construye el latice de control
FUNCTION BS(ind,x)	IND – X –		Evalua en la base Shepard

MÉTODO	PARÁMETROS	VALOR DEVUELTO	DESCRIPCIÓN
`FUNCTION wkl(k,l,s,t)`	`K,L,S,T`		Efectua el producto bilineal de la evaluación de la base
`SUBROUTINE Lin_f(nc,xc,yc,zc,x)`	`NC,XC,YC,ZC` `X`		
`FUNCTION SS(a,b,n)`			

K.4 Instancias del manejo de rutinas para el sub-módulo: Tratamiento de Contornos

MÉTODO	PARÁMETROS	VALOR DEVUELTO	DESCRIPCIÓN
INT FUNC SPLCON_C1(X,Y,M,AUTO, CONTROL_P,MAX_CP, MAX_SECTIONS,SECTIONS, WEIGHT,KNOT,SINGULAR)	X –No de puntos X,Y – Coordenadas de los puntos AUTO –0 para contornos cerrados, 1 para contornos abiertos MAX_CP - Máximo de puntos de control del spline cónico MAX_SECTIONS –No máximo de secciones del spline cónico	CONTROL_P – Polígono de control del spline cónico SECTIONS –No de secciones del spline WEIGHTS –Pesos de cada seccion cónica KNOTS –Nodos del spline SINGULAR –1 si la i-ésima sección cónica del spline es singular, en otro caso, toma el valor de 0	Función que, dado el número de puntos del contorno y sus coordenadas, genera un spline cónico con derivada continua que aproxima el contorno poligonal . Se usa la representación de Bézier racional en su forma estándar.
Double FUNC BXY(Coord,Vertice,parametro)	Coord – Coordenadas de una seccion de B V – Coordenadas de A Parámetro -t	BXY - Coordenadas de interpolación lineal	Función que obtiene la interpolación lineal determinada por los nodos **A, B** y parámetro **t.**
Double FUNC pesow1(x0,y0,x1,y1,x2,y2)	x0,y0 - coordenada b0 x1,y1 – coordenada b1 x2,y2 - coordenada b2	pesow1 -peso asociado al vertice b1	Función que asigna el peso central de la cónica con puntos de control: **bi=(xi,yi).**
Sub KNOTS(CONTROL_POLY,D_CP, WEIGHT,D_W,SECTIONS,KNOTS_C1)	CONTROL_POLY - Polígono de control del spline cónico D_CP -Máximo de puntos de control del spline cónico D_W -No máximo de secciones del spline cónico SECTIONS –No de secciones del spline	KNOTS_C1 – Nodos	Rutina que, dado el polígono de control y los pesos de un spline cónico G^1 determina los nodos para generar un spline cónico C^1.

MÉTODO	PARÁMETROS	VALOR DEVUELTO	DESCRIPCIÓN
`Double FUNC long_arco_spl(CP,D_CP,W,D_W,K,S, sections,num)`	`CP` – CONTROL_P `W` –WEIGHTS `K` –KNOTS_C1 `S` - 1 si la i-ésima sección cónica del spline es singular, en otro caso, toma el valor de 0 `sections` - Número de secciones del spline `num` - Tamaño de la partición uniforme del intervalo unitario para calcular la longitud de arco del spline	`long_arco_spl` - Longitud de arco del spline cónico	Función que calcula la longitud de arco de un spline cónico dividiendo el intervalo de definición en n partes.
`Double FUNC larcoSplcon(Control_P,D_CP,W, D_W,K,S,sections,uini,ufin)`	Control_P - Polígono de control del spline cónico	`uini,ufin` - valores inicial y final del parametro que definen el segmento de curva	Función que calcula la longitud de arco del spline cónico en el intervalo **[l_ini,l_fin]**.
`subroutine SplineLinRac(CP,D_CP,W,D_W,K,S, nsec,Long_arco_spl,n, knot,coef)`	`CP` -Polígono de control del spline cónico `nsec` -No de secciones del spline `Long_arco_spl` -Longitud de arco del spline cónico `n` -cantidad de datos que se usará para obtener el spline racional lineal	`knot` –nodos del spline racional lineal `coef` - coeficientes del spline racional lineal	Esta rutina calcula un spline racional lineal que aproxima la función inversa de la longitud de arco del spline cónico.

MÉTODO	PARÁMETROS	VALOR DEVUELTO	DESCRIPCIÓN
subroutine reparamSplcon(PC,D_CP,W,D_W,K,S, sections,t_k,deltau,n,t_k_sig, knots,coef)	t_k -k-esima aproximación de los valores del parametro t_i^*; se espera que los puntos ß(t_i^*) estén uniformemente distribuidos sobre la curva ß. Deltau - vector guarda la longitud de arco de ß entre ß(t(i)) y ß(t(i+1)). n- cantidad de puntos de la muestra que se desea obtener con distribución uniforme sobre ß(t)	t_k_sig - k+1 aproximación de los valores del parametro t_i^*. knots -Nodos del spline racional coef - coeficientes del spline racional	Rutina que, dada una aproximación del spline racional lineal genera la siguiente.
SUBROUTINE CONIC_SPLINE_VAL(C_P,D_CP,W,D_W, K,S,SECTIONS,U,CS_XU,CS_YU)	U -valor del parametro	CS_XU - coordenada x en el spline cónico CS_YU - coordenada y en el spline cónico	Esta rutina obtiene el punto del spline cónico correspondiente al valor del parámetro u.
subroutine ratsplval(coef,knots,nk,ss,nn,uval)	Chef_srL - coeficientes del spline racional Knots_srL - Nodos del spline racional nk -cantidad de nodos del spline ss - vector que contiene los puntos donde se desea evaluar el spline nn - cantidad de puntos donde se evaluara el spline	uval -vector de los valores del spline en cada punto del arreglo ss	Rutina que evalúa el spline racional lineal en los valores guardados en el arreglo U, el resultado se almacena en el arreglo T_DU. OJO - se esta suponiendo que los ss estan organizados en orden creciente

K.5 Instancias del manejo de subrutinas para el sub-módulo que optimiza puntualmente la malla 2D

MÉTODO	PARÁMETROS	VALOR DEVUELTO	DESCRIPCIÓN
MANEJO DE INSTANCIAS			
SUBROUTINE menuop_op(mn,il,jl,n,idiwa, iww,iwa,ww,red, filini,red_ant, nfix,xfix,yfix, xx,yy);	MN -total de puntos en la malla. IL -número de puntos en la vertical de la malla. JL -número de puntos en la horizontal de la malla. N -número de puntos interiores de la malla. IDIWA -dimensión del arreglo IWA. IWW -dimensión del arreglo WW. NFIX – Cantidad de puntos fijos durante el proceso de optimizacion XFIX – Arreglo que contiene el indice I del punto P(I,J) que debe estar fijo YFIX – Arreglo que contiene el indice J del punto P(I,J) que debe estar fijo XX – Arreglo que contiene la abcisa del punto P(I,J) que debe estar fijo YY - Arreglo que contiene la ordenada del punto P(I,J) que debe estar fijo	RED – contiene los puntos de la malla. FILINI – nombre de la última malla inicial. RED_ANT – Arreglo de trabajo para almacenar la malla con que inicia la sucesiòn de opt.puntual IWA – arreglo de dimensión IDIWA. WW –arreglo de dimension IWA.	Este módulo coordina la generación de una malla óptima.

MÉTODO	PARÁMETROS	VALOR DEVUELTO	DESCRIPCIÓN
SUBROUTINE fileio_ps (mn,ibo,il,jl,noi, noj,index,iflag, filnam,red,inocon, xmin,xmax,ymin, ymax,alfapreal, action,nfix,xfix, yfix,xx,yy)	MN,IL,JL IBO –total de puntos en la frontera. FILNAM –nombre de archivo. INDEX –Lee y escribe el archive red de entrada y salida.	RED, IFLAG – bandera de salida para verificar si hay error XMIN –valor mínimo de X XMAX –valor máximo de Y YMIN –valor mínimo de Y YMAX –valor máximo de Y INOCON – número de celdas no convexas ACTION - Indica si la malla está en [0,1]X[0,1]	Este módulo hace la asignación del archivo de entrada o salida para leer o escribir los puntos de la malla.
SUBROUTINE nonij(mn,ibo,il,jl,red,noi, noj,inocon,alfamin, alfamax,alfaprom, alfa_lim)	INOCON –número celdas no convexas. IBO,MN,INI	INOCON	Esta subrutina calcula cuantas celdas no convexas contiene la malla.
SUBROUTINE asigna(mn,ibo,il,jl,i,j, red,p,ini)	MN,IBO,IL,JL, I –subíndice que indica la absisa de la celda. J –subíndice que indica la ordenada de la celda. RED –arreglo que contiene los puntos de la malla.	P,INI	Dados los índices i,j de un punto, este modulo devuelve el apuntador a la posición de dicho punto en el arreglo unidimensional RED, asi como las coordenadas del punto en P(1) y P(2)
DOUBLE PRECISION FUNCTION area(c,b,a)	A –vector de coordenadas en el punto A. B –vector de coordenadas en el punto B. C –vector de coordenadas en el punto C.	Valor de: area(c,b,a)	Esta función calcula el área de un triángulo teniendo en cuenta la orientación positiva (en contra de las manecillas del reloj).

MÉTODO	PARÁMETROS	VALOR DEVUELTO	DESCRIPCIÓN
SUBROUTINE icheq(a1,a2,a3,a4,iband, alfa_lim)	A1 –área del primer triángulo de la malla. A2 –área del segundo triángulo de la malla. A3 –área del tercer triángulo de la malla. A4 –área del cuarto triángulo de la malla.	IBAND – bandera para verificar la convexidad.	Esta subrutina verifica la convexidad de las celdas que no están en las esquinas de la región.
SUBROUTINE admred(mn,ibo,il,jl,inocon, red,ssarea,scarea)	MN,IBO,IL,JL, RED	INOCON, SSAREA - suma de las áreas en las esquinas. SCAREA - suma del cuadrado de las áreas en las esquinas.	Esta subrutina calcula el área de las cuatro celdas que están en las esquinas de la región para chequear su admisibilidad.
SUBROUTINE scale(mn,red,xmin,xmax,ymin, ymax,action,alfapreal)	MN,RED, XMIN –valor minimo de X. XMAX –valor maximo de X. YMIN –valor minimo de Y. YMAX –valor maximo de Y.	RED	Este módulo escala los valores de la malla inicial a la región [0,1]x[0,1] y/o los reestablece a su escala física.
SUBROUTINE solver_p(mn,ibo,il,jl,n,iww, id3,iprint,sigma,red, ww,ifcng,inocon, idiwa,iwa,alfa_lim, alfaprom,chamet, alfapreal,red_ant,nfix, xfix,yfix,alfamin)	MN,IBO,IL,JL,N,IWW, ID3 –dimensión del vector iprint. IPRINT –especifica la frecuencia de salida. SIGMA –peso del funcional. RED,WW IFCNG –parámetro para elegir funcional. IDIWA, ALFA_LIM –valor minimo para el cual se considera que el área de una celda es positiva.	RED,G, IRES – bandera que da la forma de salida. INOCON, FILNAM WW, ALFA_MIN, RED_ANT	Este módulo prepara la entrada a la rutina de optimización que resuelve el problema de minimización proveniente de la generación de mallas variacional.

MÉTODO	PARÁMETROS	VALOR DEVUELTO	DESCRIPCIÓN
SUBROUTINE guiatron_p(mn,ibo,il,jl,n,iww, idiwa,nnz,id3, iprint,sigma,red, g,s,xc,adiag,bdiag, ldiag,xl,xu,a,ifcng, arow_ind,acol_ind, arow_ptr,acol_ptr, epsmch,listp,ngrp, idsp,y,eta,wa1,ga, w,noi,noj,iwa,lim_k, alfaprom,epsf,epsg, alfa_lim,iter,nflag, alfapreal,red_ant, nfix,xfix,yfix, alfamin,inocon)	MN,IBO,IL,JL,N,ID3, IPRINT,SIGMA,RED, W –arreglo de trabajo. FACTA –escalar para la normalización del funcional de área o para el de Ivanenko. FACTL –escalar para la normalización del funcional de longitud o para el de AO.	G,RED,WW, INOCON, Y –arreglo de trabajo RED_ANT S XC ADIAG LDIAG XL XU A AROW_IND – arreglo de trabajo ACOL_IND – arreglo de trabajo AROW_PTR - arreglo de trabajo AROW_PTR – arreglo de trabajo	Este módulo coordina la ejecución del método de Newton con Región de Confianza, hecha punto por punto, es decir, se busca el óptimo del funcional sobre toda la malla, pero sólo se considera variable P(i,j) y el resto de los puntos fijos, y así se sigue sobre toda la malla, excepto para aquellos puntos que el usuario asumió que estarían siempre fijos en el valor de la malla inicial.
SUBROUTINE calfacto(ifcng,il,jl, alfaprom,factor1, factor2)	OP, IL,JL, ALFAPROM –alfa promedio de la red.	FACTOR1 - Factor de normalizaci on para el primer funcional de la combinació n. FACTOR2 - Factor de normalizaci on para el segundo funcional de la combinació n.	Este módulo consta de una función que calcula el factor de normalización del funcional de Area-ortogonalidad.
SUBROUTINE caltim(hri,hrf,mii,mif, sei,sef,hdi,hdf)	HRI –hora inicial. MII –minuto inicial. SEI –segundo inicial. HDI –cien segundos inciales.		Subrutina que calcula el tiempo de ejecución de un programa.

MÉTODO	PARÁMETROS	VALOR DEVUELTO	DESCRIPCIÓN
SUBROUTINE cambiod(M,XB,YB,NB1)	M –número de puntos en el contorno. XB –absisas del contorno en el sentido original. YB –Ordenadas del contorno en el sentido original. NB1 –número de puntos en la primera frontera.	XB,YB	Subrutina que cambia la orientación de la frontera cambiando la distribución de puntos.
SUBROUTINE carea(ibo,red,areag)	IBO,RED	AREAG –área de la región.	Esta subrutina calcula el área total de una región.
DOUBLE PRECISION FUNCTION cheqex(x)	X –contiene el exponente que se va a verificar.	Contiene la exp(x)	Función que verifica los valores permitidos de la función exponencial.
SUBROUTINE convex(ND1,IL,JL,NP,INOCON, WXF,WYF,SAREA,SCAREA)	ND1 –dimensión de los arreglos WXF,WYF. IL,JL,INOCON, WXF –coordenadas X's sobre los puntos de la frontera. WYF –coordenadas Y's sobre los puntos de la frontera.		Subrutina que verifica la convexidad de la malla.
SUBROUTINE corner(il,jl,i,j,a1,a2,a3, a4,iband,alfaprom)	IL,JL,I,J,A1,A2,A3,A4	IBAND	Esta subrutina verifica la convexidad de las cuatro celdas de las esquinas.
SUBROUTINE daxpy(n,da,dx,incx,dy,incy)	N –variable entera. DA –variable de doble precisión. DX –variable de doble precisión. INCX –variable entera. DY –variable de doble presición. INCY –variable entera.		Constant times a vector plus a vector.

MÉTODO	PARÁMETROS	VALOR DEVUELTO	DESCRIPCIÓN
SUBROUTINE dbreakpt(n,x,xl,xu,w,nbrpt, brptmin,brptmax)	N –número de variables. X –vector de dimensión n. XL –vector de cotas inferiores. XU –vector de cotas superiores. W –vector de dimension n. NBRPT –número de puntos malos. BRPTMIN –mínimo de puntos malos. BRPTMAX –máximo de puntos malos.		Esta subrutina calcula el número de puntos malos y el mínimo y máximo de puntos malos de la proyección x + alpha*w sobre el intervalo n-dimensional [x1,xu].
SUBROUTINE dcauchy(n,x,xl,xu,a,diag, col_ptr,row_ind, g,delta,alpha,s,wa)	N,X, XL –vector de cotas inferiors. XU –vector de cotas superiores. A –arreglo que contiene los elementos de una matriz triangular inferior. DIAG –contiene los elementos de la diagonal de A. COL_PTR –arreglo de dimension n+1, contiene los elementos de las columnas de A. ROW_IND –arreglo de dimension NNZ, contiene los elementos de la matriz triangular inferior. G –arreglo de dimensión N, contiene al gradiente. DELTA –tamaño de la region de confianza. ALPHA –estimación actual del paso de Cauchy. S –paso de Cauchy. WA –arreglo de dimensión N.		Esta subrutina calcula un paso de Cauchy que satisface una región de confianza y una condición suficientemente decreciente.
SUBROUTINE dcopy(n,dx,incx,dy,incy)	N,INCX,INCY		Esta subrutina copia un vector x a un vector y.
DOUBLE PRECISION FUNCTION ddot(n,dx,incx, dy,incy)	N,INCX,INCY	Nos devuelve el producto de dos vectores en ddot	Esta función efectúa el producto interno de dos vectores.

MÉTODO	PARÁMETROS	VALOR DEVUELTO	DESCRIPCIÓN
SUBROUTINE degr(n,indrow,jpntr,indcol, ipntr,ndeg,iwa)	N, INDROW –arreglo con las filas no cero de la matriz A. JPNTR –especifica la localización de los indices de las filas en INDROW. INDCOL –arreglo con las columnas no cero de A. IPNTR – especifica la localización de los indices de las filas en INDCOL. NDEG –especifica el grado de la sucesión. IWA		Dada una matriz A sparse de m por n, esta subrutina determina el grado de la sucesión de la intersección gráfica de las columnas de A.
SUBROUTINE dgpstep(n,x,xl,xu,alpha,w,s)	N,X,XL,XU,ALPHA,W,S		Esta subrutina calcula el paso de la proyección gradiente.
SUBROUTINE dicf(n,nnz,a,diag,col_ptr, row_ind,p,info,indr, indf,list,w)	N, NNZ, A,DIAG,COL_PTR, ROW_IND, P –memoria disponible para la factorización incompleta de Cholesky. INFO –nos da información sobre la factorización. INDR –arreglo de trabajo de dimension N. INDF –arreglo de trabajo de dimension N. LIST –arreglo de trabajo de dimension N. W		Dada una matriz A simétrica sparse en una fila, esta subrutina calcula la factorización incompleta de Cholesky.

MÉTODO	PARÁMETROS	VALOR DEVUELTO	DESCRIPCIÓN
SUBROUTINE dicfs(n,nnz,a,adiag,acol_ptr, arow_ind,l,ldiag, lcol_ptr,lrow_ind, p,alpha,iwa,wa1,wa2)	N –orden de la matriz A. NNZ –número de entradas no cero de la matriz triangular inferior. A, ADIAG –contiene los elementos de la diagonal A. ACOL_PTR –contiene los elementos de las columnas de A. ARROW_IND –contiene los elementos de las filas de A. L –contiene los elementos de la parte triangular inferior. LDIAG –contiene los elementos de la diagonal de L. LCOL_PTR –contiene los puntos de las columnas de L. LROW_IND –contiene las filas de la parte triangular inferior de L. P –memoria disponible para la factorización de Cholesky. ALPHA –desplazamiento. IWA, WA1 –arreglo de trabajo de doble presición. WA2 –arreglo de trabajo de doble presición.		Dada una matriz A simétrica sparse en una columna, esta subrutina calcula la factorización incompleta de Cholesky.
SUBROUTINE dmid(n,x,xl,xu)	N,X,XL,XU		Esta subrutina calcula la proyección de x sobre el intervalo n-dimensional [xl,xu].
DOUBLE PRECISION FUNCTION DNRM2 (N, X, INCX)	N,X,INCX	DNRM2 := sqrt(x'*x)	Esta función nos da la norma euclidiana de un vector.
DOUBLE PRECISION FUNCTION dpmeps()		dpmeps = a	Esta función calcula el parámetro de presición de la máquina.

MÉTODO	PARÁMETROS	VALOR DEVUELTO	DESCRIPCIÓN
SUBROUTINE dpofa(a,lda,n,info)	A –matriz simétrica factorizada. LDA –dimensión principal del arreglo A. N	A,INFO	Esta subrutina calcula un factor de doble precision de una matriz simétrica positiva definida.
SUBROUTINE dprsrch(n,x,xl,xu,a,diag, col_ptr,row_ind,g,w, wa1,wa2)	N,X,XL,XU,A,DIAG, COL_PTR,ROW_IND,G,W, WA1,WA2		Esta subrutina usa una búsqueda proyectada para calcular un paso que satisface una condición suficientemente decreciente para la cuadrática q(s) = 0.5*s'*A*s + g'*s.
PROGRAM Driv_Gen_Mall_p	No hay parámetro de entrada	Como salida nos devuelve una malla optimizada con AO	Este programa coordina la optimización de una malla inicial dada por el usuario. Aqui solo se lee el tamaño de la malla.
SUBROUTINE dscal(n,da,dx,incx)	N,DA,DX,INCX		Esta subrutina escala un vector por una constante.
SUBROUTINE dsel2(n,x,keys,k)	N,X, KEYS –arreglo de dimension n. K –es el k-ésimo elemento más grande.		Dado un arreglo x, esta subrutina permuta los elementos del arreglo.
SUBROUTINE dsetsp(n,nnz,row_ind,col_ind, col_ptr,row_ptr,method, info,listp,ngrp,maxgrp, iwa)	N,NNZ,ROW_IND,COL_IND, COL_PTR,ROW_PTR, METHOD –parámetro para determinar el método a usar. INFO LISTP –especifica la permutación de la matriz. NGRP –especifa la partición de las columnas de A. MAXGRP –número de grupos en la partición de las columnas de A. IWA		Dado los elementos no cero de una matriz simétrica A en formato de coordenadas, esta subrutina calcula la información para determinar A triangular inferior por el método de sustitución.

MÉTODO	PARÁMETROS	VALOR DEVUELTO	DESCRIPCIÓN
SUBROUTINE dspcg(n,x,xl,xu,a,adiag, acol_ptr,arow_ind,g, delta,rtol,s,nv,itermax, iters,info,b,bdiag, bcol_ptr,brow_ind,l, ldiag,lcol_ptr,lrow_ind, indfree,gfree,w,wa,iwa)	N,X,XL,XU,A,ADIAG, ACOL_PTR,AROW_IND,G, DELTA -tamaño de la region de confianza. RTOL -especifica la precisión del mínimo final. S -paso de Cauchy NV -memoria de la factorización incompleta de Cholesky. ITERMAX -limite sobre el número de iteraciones del gradiente conjugado. ITERS -conjunto del número de iteraciones del gradiente conjugado. INFO, B -parte triangular inferior de B. BDIAG -contiene los elementos de la diagonal de B. BCOL_PTR -contiene puntos de las columnas de B. BROW_IND -contiene las filas de la parte triangular inferior de B. L,LDIAG,LCOL_PTR LROW_IND -contiene puntos de las filas de la parte triangular inferior de L. INDFREE -Arreglo entero de trabajo de dimensión N. GFREE -Arreglo de trabajo de doble precision de dimensión N. W,WA,IWA		Esta subrutina genera una sucesión de aproximación al mínimo para el subproblema min { q(x) : xl <= x <= xu }.
SUBROUTINE dsphesd(n,row_ind,col_ind, row_ptr,col_ptr, listp,ngrp,maxgrp, numgrp,eta,fhesd, fhes,diag,iwa)	N,ROW_IND,COL_IND, ROW_PTR,COL_PTR, LISTP,NGRP,MAXGRP, NUMGRP -conjunto de un número de grupos. ETA -diferencia del vector parámetro. FHESD -arreglo de doble precisión de longitud N. FHES -Arreglo de doble precisión de longitud NNZ. DIAG,IWA		Esta subrutina calcula una aproximación a la matriz Hessiana (simétrica) de una función por un método de substitución.

MÉTODO	PARÁMETROS	VALOR DEVUELTO	DESCRIPCIÓN
SUBROUTINE dsscfg(nx,ny,x,f,fgrad, task,lambda)	NX,NY,X, F -conjunto de funciones evaluadas en X. FGRAD -contiene el gradiente evaluado en X. TASK -especifica la acción de la subrutina.		Esta subrutina calcula la función y el gradiente del problema de combustión en equilibrio.
SUBROUTINE dssm(n,npairs,indrow,indcol, method,listp,ngrp, maxgrp,mingrp,info,ipntr, jpntr,iwa,liwa)	N, NPAIRS, INDROW, INDCOL, METHOD,LISTP, NGRP,MAXGRP,MINGRP, INFP,IPNTR,JPNTR,IWA, LIWA -Entero positivo, que no es menor que 6*n.		Dado el modelo sparse de una matriz simétrica A de orden n, esta subrutina determina una permutación simétrica de A y una partición de las columnas de A que consiste de determinar A como una matriz triangular inferior por el método de substitución.
SUBROUTINE dssyax(n,a,adiag,jptr, indr,x,y)	N,A,ADIAG,JPTR, INDR,X,Y		Esta subrutina calcula el producto matriz-vector y = A*x.
SUBROUTINE dstrsol(n,l,ldiag,jptr, indr,r,task)	N,L,LDIAG,JPTR, INDR,R,TASK		Esta subrutina resuelve el sistema triangular L*x = r o L'*x = r.

MÉTODO	PARÁMETROS	VALOR DEVUELTO	DESCRIPCIÓN
SUBROUTINE dtron(n,x,xl,xu,f,g,a,adiag, acol_ptr,arow_ind, frtol,fatol,fmin,cgtol, itermax,delta,isel, b,bdiag,bcol_ptr, brow_ind,l,ldiag, lcol_ptr,lrow_ind, xc,s,indfree,isave, dsave,wa,iwa,bandtu, cond,ilcu,epsmch)	N,X,XL,XU,F,G,A,ADIAG, ACOL_PTR,AROW_IND, FRTOL -especifica el error relatiVo de la función. FATOL -especifica el error de la función. FMIN -especifica una cota inferior de la función. CGTOL -especifica el criterio de convergencia para el método de gradiente conjugado. ITERMAX -especifica el limite del número de iteraciones de gradiente conjugado. DELTA,ISEL,B,BDIAG, BCOL_PTR,BROW_IND, L,LDIAG,LCOL_PTR, LROW_IND, XC -arreglo de trabajo de doble precisión, de dimensión N. S,INDFREE, ISAVE -arreglo entero de trabajo de dimensión 3. DSAVE -arreglo de trabajo de doble precisión, de dimensión 3. WA,IWA, BANDTU -variable lógica. COND -variable caracter de longitud 1. ILCU -variable caracter de longitud 1. EPSMCH -variable doble precisión.		Esta subrutina implementa un método de Newton de región de confianza para la solución de problemas de optimización acotados.
SUBROUTINE dtrpcg(n,a,adiag,acol_ptr, arow_ind,g,delta,l, ldiag,lcol_ptr, lrow_ind,tol,stol, itermax,w,iters,info, p,q,r,t,z)	N,A,ADIAG,ACOL_PTR, AROW_IND,G,DELTA,L, LDIAG,LCOL_PTR, LROW_IND, TOL, STOL, ITERMAX,W,ITERS,INFO, P,Q,R,T,Z	INFO	Dada una matriz A sparse simétrica en una columna, esta subrutina usa un método de gradiente conjugado precondicionado para encontrar una paroximación al mínimo del subproblema de región de confianza min { q(s) : || L'*s || <= delta }.

MÉTODO	PARÁMETROS	VALOR DEVUELTO	DESCRIPCIÓN
SUBROUTINE dtrqsol(n,x,p,delta,sigma)	N,X,P,DELTA,SIGMA		Esta subrutina calcula la solución más grande (no-negativa) de la ecuación de la región de confianza: ||x + sigma*p|| = delta.
SUBROUTINE escres(n,id3,iprint,iter,ifun, igra,inocon,gnorm,epsg, epsf,sigma,tao,x,f,g, finish,ipri,ifcng)	N,ID3,IPRINT,ITER, IFUN -número de veces que se evalua la función. IGRA -número de veces que se evalua el gradiente. EPSG -tolerancia del criterio del gradiente. EPSF -tolerancia del criterio de la función. SIGMA,X,G		
SUBROUTINE fao_p(mn,ibo,il,jl,iind, jind,n,f,g,red,isel, factl)	MN,IBO,IL,JL,N,RED, FACTL IIND -índice I del punto que se esta optimizando JIND -índice J del punto que se está optimizando	F,G,INOCON	Esta subrutina calcula el valor del funcional de A-O en un vector x de puntos interiores de una malla.
SUBROUTINE fhfv(p,q,r,s,qp,rq,sr,ps, fh,fv)	P,Q,R,S	FH - evaluación de la parte horizontal del funcional de área ortogonalidad para la celda dada. FV - evaluación de la parte vertical del funcional de área ortogonalidad para la celda dada. QP,RQSR,PS	Esta subrutina evalua la parte horizontal y vertical del funcional de área-ortogonalidad para la celda de vértices PQRS.

MÉTODO	PARÁMETROS	VALOR DEVUELTO	DESCRIPCIÓN
SUBROUTINE fyg_p(mn,ibo,il,jl,iind,jind, inocon,red,g,f,n,ifcng, isel,facta,factl,ga, alfaprom,alfa_lim)	MN,IBO,IL,JL,INOCON, SIGMA,RED,N,IIND,JIND, IFCNG –parámetro para elegir el funcional. ISEL,FACTA, FACTL –escalar para la normalización del segundo funcional. GA –variable de doble precisón para un espacio de memoria. ALFAPROM,ALFA_LIM	INOCON,F,G GA,DFRED	Esta subrutina calcula la función escalar f sobre la malla actual y, si es necesario, su vector gradiente.
SUBROUTINE gao(qp,rq,sr,ps,g1,g2, g3,g4,fh,fv)	P,Q,R,S	G1,G2,G3, G4,FH,FV	Esta subrutina calcula el valor del funcional A-O de una celda PQRS y sus derivadas correspondientes con respecto a p, q, r, s.
SUBROUTINE gpabtr(p,q,r,g1,g2,g3,f, pqr,isel)	P,Q,R,PQR		Esta subrutina calcula las derivadas respecto a cada tres puntos que determinan el triángulo PQR.
SUBROUTINE INTERV(XT, LXT, X, LEFT, MFLAG)	XT –sucesión real de dimension LXT. LXT –número de terminos en la sucesión XT. X –es el punto localizado respecto a la sucesión XT.		Esta subrutine calcula: LEFT = MAX(i;1 .le. i .le. LXT .and. XT(i) .le. X).
DOUBLE PRECISION FUNCTION LNVALU(XYB,TF, I,M,L, TFI,H, HXY)	TF –variable de doble precisón con los puntos parametrizados. TFI –punto en el cual se evaluará.	LNVALU, de F en X	Esta subrutina llama INTERV y calcula el valor en X.
SUBROUTINE MENUGE(MN,IL,JL,idiwa,RED, iwa,FNAMEI)	IL,JL,FNAMEI	RED	Esta subrutina genera una malla inicial, para regiones en dos dimensiones.
SUBROUTINE menugene(MN,IL,JL,RED,M, IBFLAG,NB1,NB2,NB3, NB4,XB, YB, INFO,WTB, WXF,WYF,DY,XFT,YFT)	IL,JL,RED,M,NB1,NB2, NB3,NB4	RED	Esta subrutina genera una malla inicial, para regiones en dos dimensiones.

MÉTODO	PARÁMETROS	VALOR DEVUELTO	DESCRIPCIÓN
SUBROUTINE MODULO(M,MARCA,ZB,ZBT)	M, MARCA -punto inicial. ZB - vector con los puntos en la posición anterior. ZBT - vector con los puntos en la nueva posición.		
SUBROUTINE NCOREG(MN,M,MP,NB1,NB2,NB3, NB4,IL,JL,ND1,ND4, IBFLAG,NINFR,ND5,ND6, XF,YF,WTB,XB,YB,RED, IBFAIL)	MN,M,NB1,NB2,NB3, NB4,IL,JL,ND1,ND4, NINFR,ND5,ND6	RED,WTB, XF,YF,XB, YB	Esta subrutina genera la malla inicial para una región no conexa.
SUBROUTINE pabg(il,jl,n,g,fj,u,v,factor)	IL,JL,N,G, FJ -gradiente respecto al punto P(u,v). U -índice i del punto respecto al cual se deriva. V -índice j del punto respecto al cual se deriva. FACTOR -escalar constante por el que se multiplica cada componente del gradiente.	G	Esta rutina coloca en la posición correcta el gradiente con respecto a el punto (u,v).
SUBROUTINE PARLIN(ND1,M,NPB,INIC1,INIC2, FIN,NPOINT,J3,XB,YB,TF, XF,YF)	ND1,M, XB -coordenadas Xi's de los puntos de la malla. YB -coordenadas Yi's de los puntos de la malla. NPOINT -número de puntos. TF -vector con los puntos parametrizados.	XF,YF	
SUBROUTINE pkhpk(n,p,g,x,hpk,xk1,acc, pnorm,php,ient)	N,P,G,X, ACC -precisión de la máquina. PNORM - cuadrado de la norma del vector P. PHP - valor de la cuadrática pk'Hkpk. IENT -bandera.	HPK,PHP, IENT,X,XKL	Esta subrutina calcula la forma cuadrática pk'Hpk.
SUBROUTINE posder2(i,j,col,il,jl,nnz, arow_ind,acol_ind, sigma,jj)	I,J,COL,IL,JL,NNZ, AROW_IND,ACOL_IND, SIGMA,JJ		Esta rutina da el índice de elementos no cero del Hessiano.
SUBROUTINE posder(n,il,jl,nnz,arow_ind, acol_ind,sigma,nzeros)	N,IL,JL,NNZ,AROW_IND, ACOL_IND,SIGMA,NZEROS		Hace un llamado a posder2.

MÉTODO	PARÁMETROS	VALOR DEVUELTO	DESCRIPCIÓN
SUBROUTINE red_ini(mn, marca, npoint, il, jl, nd1, nd4, nd5,mp, ninfr, xft, yft, red)	MN,IL,JL,ND1,ND4,ND5, NINFR -vector con la posición de los puntos de las esquinas.	XFT,YFT, RED	Programa para generar la red inicial mediante el metodo algebraico TFI (Interpolacion Transfinita).
SUBROUTINE SEGRED(MN,IL,JL,ND1,ND2,ND4, ND6,RED,WTB,WXF,WYF,DY, MP,XFT,YFT,XB,YB,M, IBFLAG,NB1,NB2, NB3,NB4,BBFLAG)	MDIM -mayor número de puntos en la vertical. IL,JL,M,ND1,ND2, ND4,FNAME	RED,WTB, WXF,WFY,DY	
SUBROUTINE SELGEN(MN,MP,M,N1,IL,JL,ND1, ND2,ND4,ND5,ND6,NINFR, NP,XFT,YFT,WXF,WYF,WTB, RED,DY,XB,YB,IBFAIL)	MN, MP, M,N1,IL,JL,ND1, ND2,ND4,ND5,ND6,NP, WTB -arreglo de trabajo unidimensional. WXF -arreglo de trabajo unidimensional par alas coordenadas Xi's. WYF -arreglo de trabajo unidimensional par alas coordenadas Yi's. DY -arreglo de trabajo unidimensional. NP -número de puntos para la interpolación. IA -bandera para cambiar o no la dirección del contorno. RED	RED	Esta subrutina coordina la generación de mallas simplemente conexas.