

INSTITUTO POLITÉCNICO NACIONAL

CENTRO DE INVESTIGACIÓN EN COMPUTACIÓN

TESIS DE MAESTRÍA EN CIENCIAS

LA APLICACIÓN DE MÉTODOS DE LA PROGRAMACIÓN
MATEMÁTICA PARA LA IDENTIFICACIÓN
DE FUENTES SIMPLES DE CONTAMINACIÓN DEL AIRE

QUE PARA OBTENER EL GRADO DE:
MAESTRO EN CIENCIAS EN COMPUTACIÓN

P R E S E N T A:

ROBERTO JURADO JIMÉNEZ

Director: Dr. Mikhail Alexandrov
Codirector: Dr. Alexander Gelbukh



MEXICO D.F.

2002

Indice

Tema	Pag.
Resumen	4
Abstract	4
Introducción	5
I.1 Justificación del tema	5
I.2 Objetivo y metas	6
I.3 Aplicaciones	6
I.4 Divulgación del trabajo	7
I.5 Agradecimientos	7
Capitulo I. Estado del arte y planteamiento del problema	8
1.1 Evaluación de fuentes de contaminación	8
1.2 Modelos de contaminación del aire	13
1.3 Experiencia en Geofísica de prospección	15
1.4 Métodos de optimización	18
1.5 Resumen	27
Capitulo II. Propuesta de solución	28
2.1 Función de criterio	28
2.2 Ajuste de los parámetros del método NM	37
2.3 Aplicación de una función de preferencia	39
2.4 Combinación de métodos	46
2.5 Resumen	53
Capitulo III. Diseño y programación del sistema	54
3.1 Consideraciones generales	54
3.2 Componentes del sistema	55
3.3 Funcionalidad del sistema	58
3.4 Resumen	64
Conclusiones	65
C.1 Novedad científica del trabajo	65
C.2 Importancia práctica	65
C.3 Trabajo a futuro	65

Bibliografía	66
Glosario de términos	68
Índice de figuras	70
Índice de tablas	71
Anexos	72
A.1 Análisis de rendimiento de los métodos de optimización	72
A.2 Métodos de construcción de funciones de prueba	74
A.3 Ejemplos de funciones de criterio	78
A.4 Ejemplos del registro de resultados	80
A.5 Códigos del módulo principal	82
A.6 Códigos de biblioteca de optimización	94

Resumen

Se considera el problema de la búsqueda de fuentes locales de contaminación por medio de la técnica de programación matemática. Este se presenta en forma de un problema de minimización de una función de criterio que refleja la diferencia entre los datos de modelación y los experimentales. Se determinaron los parámetros óptimos del método Nelder-Mead para algunas funciones de criterio. Se demuestra como usar efectivamente la función de multa y las combinaciones de métodos para buscar un mínimo global entre varios mínimos locales. Se desarrolló el sistema MAC con una biblioteca de optimización y una interfaz gráfica por medio de la cual el usuario puede manejar el proceso de búsqueda. El sistema se programó en un ambiente de programación de C++.

Abstract

The problem of search of local sources of contamination by means of mathematical programming technique is considered. This problem is presented in the form of a minimization problem for a criteria function reflecting the difference between data of modeling and experimental data. The optimum parameters of Nelder-Mead method were determined for some criteria functions. It is demonstrated how to use effectually the penalty function and the combination of methods for searching a global minimum between several local minimums. It was developed the system MAC with one optimization library and graphic interface. With it a user can manage the search process. The system was programmed in C++.

INTRODUCCIÓN

I.1 Justificación del tema

Probablemente la primera experiencia peligrosa que tuvo el hombre con la contaminación del aire fue cuando produjo el fuego en sus cavernas pobremente ventiladas. Desde entonces hemos continuado con la contaminación sobre la superficie de la tierra, hasta que recientemente los problemas de contaminación ambiental han sido peligrosos a la vida debido a la industrialización de la sociedad, la introducción de vehículos motorizados y la explosión demográfica. Este problema no se disminuye a pesar de la propia habilidad de la Tierra de absorber y purificar cantidades pequeñas de contaminantes.

El control de la contaminación del aire no es trivial ya que hay muchos intereses de por medio que no permiten la aplicación rigurosa de políticas ambientales en beneficio de la gente. A diferencia de la contaminación del agua y del suelo que pueden tratarse en forma aislada, con la contaminación del aire no se puede, hacer esto, razón por la cual los problemas sobre la contaminación atmosférica requieren especial atención [World, 1992; Collinz, 1993].

Es necesario hacer notar que prácticamente todas las ciudades tienen su propio control ecológico el cual es especialmente fuerte en las grandes urbes. El significado del término “control” en el presente trabajo es el de determinar la contribución de varias fuentes a la contaminación del aire, la cual debe evidentemente verse reflejada en el modelo. Un usuario puede cambiar los parámetros del modelo y observar el resultado de estos cambios. Esta tecnología es utilizada con el programa ISC (Industrial Sources Complex) el cual utiliza dos tipos de modelos de dispersión uno para periodos cortos y otro para largos, para resolver un problema directo de modelación [Technical manual, 1995a, 1995b].

La determinación de los parámetros del modelo es un “problema inverso”, definiendo como tal, encontrar la causa de un problema a partir del conocimiento de los efectos de esta. La búsqueda manual de estos parámetros es una tarea difícil utilizando los modelos multivariados, ya que en cada paso de la búsqueda es necesario evaluar la exactitud de la solución tomando en cuenta la desviación entre los datos observados y los datos de modelación en el espacio de los parámetros. Por otro lado los métodos automáticos tienen posibilidades limitadas debido a la organización formal de la búsqueda, en muchos casos se detienen cerca de cualquier extremo local y en otros encuentran soluciones no existentes, por esta razón, el usuario debe controlar el trabajo del método automático en base a la interpretación de los resultados que se van obteniendo, por lo que resulta necesario combinar ambos tipos de búsqueda (manual y automática) de los parámetros óptimos del modelo, y además es necesario ajustar los métodos automáticos a las propiedades de la función de criterio que se usa.

Esta técnica es muy utilizada en Geofísica de prospección desde hace ya varios años y se aplicó en varios sistemas de programas interactivos para encontrar la solución de problemas inversos en modo gráfico [Smith, 1972; Alexandrov, 1982; Yudin, 1983]. Pudiera ser efectiva para la solución de algunos problemas inversos del medio ambiente.

Para comprobar esta técnica se trabajó con el modelo ISCST3. El modelo utiliza una ecuación Gaussiana de estado estable para modelar la pluma lineal contaminante [Technical manual, 1995a; Moussiopoulos, 1996].

I.2 Objetivo y metas

Desarrollar una técnica de búsqueda de simples fuentes locales de contaminación atmosférica mediante la aplicación de métodos de programación matemática.

Para alcanzar el objetivo mencionado es necesario cumplir las siguientes metas:

- Dar un panorama de algunos métodos de optimización y desarrollar la biblioteca de optimización con métodos cuyos parámetros de control sean variables.
- Investigar el comportamiento del método NM para modificarlo y obtener los parámetros óptimos.
- Evaluar la posibilidad de usar información a priori acerca de la ubicación del mínimo global de la función de criterio usando una función de preferencia.
- Encontrar la estrategia de búsqueda óptima del mínimo global mediante la combinación de métodos de optimización.
- Desarrollar procedimientos de diálogo interactivo para el control de la búsqueda de solución y realizar la programación del sistema.

I.3 Aplicaciones

Esta investigación forma parte de los proyectos siguientes:

1. La predicción del desarrollo del campo de contaminación de la atmósfera de México D.F.: software para modelación matemática (CONACyT, No. 27928-a, 1999-2001).
2. Determinación de fuentes contaminantes locales y de área mediante técnicas de modelación matemática (CGEPI – IMP, No. 990080, 1999-2000).

En el marco de los proyectos mencionados el autor desarrolló la biblioteca de optimización y realizó experimentos numéricos de búsqueda de fuentes locales de contaminación en un área definida.

I.4 Divulgación del trabajo

Los resultados del trabajo fueron publicados en:

1. Jurado, R., Alexandrov, M. (2000): *Software for the search of simple sources of air pollution using some methods of optimization*, In: "Acta Academia", Publ. House of Intern. Inform. Academy under UN (branch of Moldova), NY-Chisinau etc., annual issue, pp. 381-391.
2. Alexandrov, M., Jurado, R. (2000): *Determinación de fuentes contaminantes locales y de área mediante técnicas de modelación matemática*, Reporte técnico final, CIC-IPN.

Se tiene un acta de aplicación del sistema en la Academia Estatal de Prospección Geológica de Moscú.

I.5 Agradecimientos

La idea del tema de esta tesis surgió de mi trabajo práctico y desde hace dos años que se la expuse a mis asesores no tenía idea de los obstáculos con los que me tenía que enfrentar para la realización de un trabajo científico. Un primer obstáculo que se presentó fue que el enfoque que se utilizó en este trabajo es conocido pero no se había aplicado en sistemas reales para el control de la contaminación del aire. Un segundo obstáculo fue que mis conocimientos en el campo de la programación matemática no eran muy extensos. Por esto quiero agradecer a mis asesores el Dr. Mikhail Alexandrov y el Dr. Alexander Gelbukh por la gran ayuda que me dieron para el planteamiento de este problema y la elaboración de la tesis.

Quiero agradecerles también al Dr. Zvi Retchciman y al Dr. Mikhail Yudin por sus comentarios muy útiles. Mis agradecimientos al Prof. Jesús Figueroa, Doctor en física teórica de la Universidad de Edimburgo, por su atención, crítica y sugerencias.

CAPITULO I.

Estado del arte y planteamiento del problema

Las fuentes simples de contaminación atmosférica son aquellas que pueden considerarse como locales en un área considerada. En este capítulo se da el planteamiento del problema de la búsqueda de los parámetros de fuentes locales y se destacan las dificultades de esta usando el modo automático. Se muestra una experiencia de solución de problemas inversos que se presentan en Geofísica de prospección cuyo enfoque podría usarse en problemas de contaminación del aire. La selección de los métodos de optimización los cuales deben ser usados para la búsqueda de los parámetros del modelo, requiere de un análisis de las características de los métodos. El resultado breve de este análisis se presenta en este capítulo.

1.1 Evaluación de fuentes de contaminación del aire

1. Fuentes de contaminación en zona urbana

Generalmente las fuentes de contaminación están bien determinadas y el problema consiste en pronosticar el cambio del nivel de contaminación cuando cambian la velocidad de emisión de las fuentes. Sin embargo, a veces es necesario revisar la intensidad real de las fuentes dadas o detectar algunas fuentes desconocidas. Tal control reduce el problema a sólo poder identificar el modelo. Debe ponerse especial atención a la zona urbana donde la situación con la contaminación del aire es peligrosa para la salud de sus habitantes.

Las fuentes principales de contaminación atmosférica en las zonas urbanas son los vehículos, siendo estas fuentes distribuidas por calles y avenidas. Sin embargo las calles y avenidas (o partes de ellos) junto con los vehículos pueden considerarse como fuentes locales en la escala de una ciudad grande. Tal aproximación es correcta para México D.F., Beijing y muchas otras megalópolis con extensiones de miles kilómetros cuadrados. En este caso las fuentes distribuidas se sustituyen por una fuente local equivalente que se ubica en el centro geométrico de estas. La intensidad de la fuente local debe ser igual a la intensidad integral de estas fuentes distribuidas.

Además en muchas cosas las fuentes locales como son las industrias también tienen una contribución significativa sobre la contaminación atmosférica. Claro que en este caso no se requiere ninguna sustitución.

Los principales contaminantes encontrados en muchas áreas urbanas son: monóxido de carbono, óxidos nitrosos, óxidos sulfurosos, hidrocarburos y materia particulada (sólida y líquida). Estos contaminantes están dispersos en la atmósfera del mundo, en concentraciones suficientemente altas como para causar serios problemas de salud, los cuales pueden ocurrir rápidamente cuando los contaminantes del aire se concentran, por ejemplo cuando ocurre una inversión térmica, donde la ciudad queda atrapada en una bolsa de aire llena de contaminantes que no pueden dispersarse por convección natural.

El software aquí desarrollado puede aplicarse para cualquiera de los contaminantes mencionados. Es necesario solamente reflejar las propiedades de estos contaminantes en el modelo aceptado para simular la dispersión de contaminantes. Por ejemplo, en este trabajo se utilizó el modelo de dispersión Gaussiana del paquete ISC el cual es de estado estacionario (independiente del tiempo) y local (ver párrafo 1.2). Si los contaminantes a considerar no pueden dispersarse o son dinámicos y distribuidos, entonces es necesario elegir otro modelo, pero la filosofía de búsqueda será la misma.

Los datos sobre la contaminación del aire en el Valle de México se consideran en la monografía de Collinz [Collinz, 1993]. Un análisis generalizado de la situación de la contaminación del aire en megalópolis del mundo está escrito en un reporte [World, 1992].

2. Planteamiento del problema en forma estocástica

Se requiere determinar los parámetros del modelo empleado para simular la dispersión de contaminantes atmosféricos a partir del conocimiento de una nube dispersa de estos en la atmósfera sobre una cierta región.

La información inicial necesaria para determinar los parámetros del modelo mencionado son los datos experimentales de las concentraciones del contaminante registrados en los sitios de monitoreo.

Sean:

D = dominio de simulación
 D_k = puntos de observación, $k=1,2,\dots,n$
 \mathbf{X} = parámetros del modelo dado, $\mathbf{X} = \{x_i\} i=1,2,\dots,m$
 $F_o(D_k)$ = datos de observación
 $F_m(\mathbf{X}, D_k)$ = resultados de la modelación calculados en los puntos de observación.

El planteamiento en forma estocástica del problema supone que:

Dados:

- 1) $P(x_i, a_i, b_i, \dots) =$ leyes de distribución de los parámetros $\{x_i\}$ del modelo. Aquí a_i, b_i, \dots son parámetros de las leyes, por ejemplo, para la ley normal representan el valor medio y la desviación estándar.
- 2) $P(F_k | \mathbf{X}) =$ probabilidades condicionales de los valores de observación para los parámetros dados del modelo.

Con base a esta información es posible usar las estimaciones Bayesianas de los parámetros del modelo teniendo en cuenta los datos de observación $F_o(D_k)$. Sin embargo todas estas probabilidades son desconocidas y por eso el enfoque Bayesiano no es aplicable.

Se puede simplificar el problema de la manera siguiente:

- Se acepta que cada parámetro del modelo x_i tiene un ley normal con los parámetros desconocidos $m_i =$ valor medio y $\sigma_i^2 =$ desviación.
- Se acepta que el modelo dado refleja correctamente los procesos de contaminación del aire y se omiten los errores de medición. En este caso los valores $(F_k | \mathbf{X})$ son resultados de modelación $F_m(\mathbf{X}, D_k)$ calculados en los puntos de observación. De acuerdo a Cramér [Cramér, 1946] se puede considerar cada valor F_k como un valor normal y determinar sus parámetros en el marco de la aproximación lineal, es decir,

a) el valor medio $M_{F_k} = F_m(m_1, m_2, \dots, D_k)$ (1.1-1)

b) la desviación $\sigma_{F_k}^2 = \sum_{i=1}^m \left[\frac{\partial}{\partial x_i} F_m(x_1, x_2, \dots, D_k) \Big|_{x_1=m_1, x_2=m_2, \dots} \right]^2 \sigma_i^2$ (1.1-2)

En base a esta información es posible usar el método clásico de máxima verosimilitud teniendo en cuenta los datos de observación $F_o(D_k)$. En este caso el problema se reduce al problema de minimización acerca de las evaluaciones de los valores medios m_i :

$$f(m_1, m_2, \dots) = \sum_{k=1}^n 1/\sigma_{F_k}^2 (M_{F_k} - F_o(D_k))^2 \rightarrow \min. \quad (1.1-3)$$

Teniendo una información a priori acerca de los intervalos posibles de variación de los parámetros del modelo se pueden evaluar previamente los valores m_i y σ_i^2 y después calcular $\sigma_{F_k}^2$ en cada punto por la formula (1.1-2). Substituyendo (1.1-1) en (1.1-3) y haciendo $\alpha_k = 1/\sigma_{F_k}^2$ se obtiene:

$$f(m_1, m_2, \dots) = \sum_{k=1}^n \alpha_k (F_m(m_1, m_2, \dots, D_k) - F_o(D_k))^2 \rightarrow \min. \quad (1.1-4)$$

Como se ve, tal formalización requiere muchas suposiciones y definiciones por eso la solución se vuelve muy sensible a ellos. Las funciones del tipo (1.1-4) son no lineales y dependen en forma compleja de sus parámetros, y esta circunstancia origina los problemas nuevos que se consideran en la sección siguiente.

3. Planteamiento del problema en forma determinística

Aquí se usarán las mismas definiciones utilizadas en la sección el punto anterior.

Desde el punto de vista determinístico hay dos enfoques conocidos para determinar los parámetros del modelo:

- Enfoque directo, que esta basado en transformaciones de los datos experimentales.
- Enfoque indirecto, que esta basado en la comparación de los datos de modelación y los de experimentación.

El primer enfoque se aplica en casos especiales y simples. Los métodos que aquí se emplean crean una función de transformación Ψ , que permite obtener evaluaciones de los parámetros que se buscan:

$$\Psi(F_o(D)) \rightarrow X \quad (1.1-5)$$

Generalmente la función Ψ se construye en base a ecuaciones de balance de materia en el marco de los modelos de receptores.

Se demostrará este enfoque sobre un ejemplo. Sea S un dominio definido que incluye n chimeneas con diferente producción de emisiones. Considérese la hipótesis de que todo el contaminante se dispersa y/o precipita en el dominio mencionado. La intensidad integral x del conjunto de chimeneas puede evaluarse por medio de la formula:

$$x = [\int_S G(D) dS] / T \quad (1.1-6)$$

Donde:

$G(D)$ = función de distribución de la contaminación del área en el dominio dado, es decir una función de coordenadas. Esta distribución se determina en base a procedimientos de interpolación y aproximación usado los datos experimentales, es decir, $G(D) = \theta(F_o(D_k))$ donde θ es un método de interpolación.

T = una constante de tiempo que refleja el tiempo promedio de disminución del nivel de contaminación en caso de ausencia de las fuentes. Este constante es el valor inverso del coeficiente de depositación σ , o sea $T = 1/\sigma$.

La función de transformación (1.1-5) para este caso es:

$$\Psi(F_o(D)) = [\int_S \theta(F_o(D_k)) dS] / T$$

Desde este enfoque no se necesita ningún modelo que describa los procesos de dispersión de contaminantes, pero la exactitud no es muy buena por lo que solo es posible determinar los parámetros integrales del tipo (1.1-6).

El segundo enfoque es general cuando es posible determinar cualquiera de los parámetros de las fuentes en el marco del modelo aceptado.

Sean $f(x_1, x_2, \dots, x_m)$ = una función de criterio que refléjale éxito de la búsqueda de las fuentes de contaminación. El éxito será evaluado por la diferencia entre los datos de la modelación y los datos experimentales, es decir:

$$f(x_1, x_2, \dots, x_m) = f[F_m(X, D), F_o(D)]$$

Los parámetros a buscar son un resultado de solución del problema de optimización:

$$f(x_1, x_2, \dots, x_m) \rightarrow \min.,$$

Es decir:

$$X = \operatorname{argmin} [f(x_1, x_2, \dots, x_m)]$$

La forma concreta de la función $f(\cdot)$ se define en base al interés del usuario y por los requisitos de precisión y sensibilidad de la solución. Por ejemplo en geofísica, a menudo esta función se presenta en la forma:

$$f(x_1, x_2, \dots, x_m) = \sum_{k=1}^n |F_m(x_1, x_2, \dots, x_m, D_k) - F_o(D_k)|^2 \quad (1.1-7)$$

Este enfoque se usa ampliamente en las ciencias naturales y en ingeniería para identificar sistemas complejos. En este trabajo será utilizado.

Es fácil ver que la forma de la función de criterio (1.1-7) se parece a la de la función (1.1-4) introducida en la sección 2. Pero ahora:

- no es necesario hacer algunas suposiciones acerca de la ley de distribución de cada parámetro del modelo.
- el resultado de la búsqueda no es un valor exacto sino los parámetros de algunas leyes de distribución

Cabe mencionar que siempre que haya la posibilidad de usar las formas determinísticas de formalización de un problema hay que usarlas.

El problema principal es que generalmente la función de criterio $f(\cdot)$ tiene muchos extremos locales, y además esta función es poco sensible a los cambios de valor de algunas de sus variables, esto significa que el problema (1.1-7) es un problema matemáticamente incorrecto. Esta situación es muy típica en problemas inversos. La experiencia existente para la solución de problemas inversos tanto en Geofísica como en otras áreas puede ser útil para el campo ambiental.

1.2 Modelos de contaminación atmosférica

1. *Software para modelación de procesos contaminantes del aire*

Se presenta un panorama de los modelos de contaminación ambiental en un reporte de la comisión Europea [Moussiopoulos, 1996]. Tanto los métodos numéricos como los modelos atmosféricos se consideran en el libro [Jerald, 1996].

Los modelos que simulan el transporte y dispersión de los contaminantes en el aire pueden distinguirse en muchos terrenos, por ejemplo: sobre la escala espacial, temporal, sobre el tratamiento de las ecuaciones de transporte, sobre el tratamiento de varios procesos y sobre la complejidad de la aproximación.

Pueden distinguirse los siguientes modelos:

- *Modelos de elevación de la pluma.* Estos modelos calculan el desplazamiento vertical y el comportamiento general de la pluma en la etapa inicial de dispersión.
- *Modelos Gaussianos.* Son los mas comunes de los modelos utilizados, se basan en la hipótesis de que la concentración de la pluma viento abajo tiene distribuciones gaussianas independientes tanto en la horizontal como en la vertical.
- *Modelos Semi-empíricos.* Ejemplos de estos modelos son los de caja y varios modelos paramétricos. Son muy simplificados.
- *Modelos Eulerianos.* Estos modelos resuelven numéricamente la ecuación de difusión atmosférica. Están integrados generalmente a modelos meteorológicos de pronóstico.
- *Modelos Lagrangianos.* Describen elementos de fluido que siguen el flujo instantáneo, como puffs, partículas, etc..
- *Modelos Químicos.* Varios modelos de contaminación de aire incluyen módulos para el cálculo de transformaciones químicas.

Algunos de estos tipos de modelos se utilizan en los paquetes de software:

- RAMS (Walko and Tremback, 1991; Pielke et al., 1992)
- HYPACT (Tremback et al., 1993)
- CIT (McRae et al., 1982; McRae and Seinfeld, 1983, Russell et al., 1988)
- UAM (Ames et al., 1985; Chico and Lester, 1992)
et al, etc.
- ISCST3 (EPA = Environment Protection Agency of USA, 1995). Este modelo de la EPA es usado en varias ciudades europeas para propósitos de control

A pesar de esta variedad de modelos y paquetes ninguno se utiliza para resolver problemas inversos como sería la determinación de posibles fuentes contaminantes a partir de los datos de dispersión de estos sobre un área definida.

2. Modelos atmosféricos aplicados a la ciudad de México

Para la ciudad de México se han aplicado varios modelos como son los meteorológicos, los de dispersión, los químicos etc. Su aplicación ha dependido del tipo de estudio requerido y de los contaminantes de interés.

Los modelos HOTMAC y RAPTAD desarrollados en los Laboratorios Nacionales de Los Álamos, Nuevo México, se aplicaron para conocer la distribución de las partículas suspendidas en la atmósfera de la zona metropolitana del valle de México. Se usó también el modelo tridimensional CIT utilizado en los Estados Unidos de Norte América, para simular la dispersión de los contaminantes químicos atmosféricos sobre el Distrito Federal.

El modelo ISCST3 que significa Industrial Source Complex Short Term se usó en el Instituto Mexicano del Petróleo [Technical manual, 1995a, 1995b]. La base del modelo es la ecuación de la pluma gaussiana de estado estable y lineal que se usa con algunas modificaciones para modelar emisiones provenientes de chimeneas.

Este modelo acepta datos meteorológicos horarios para definir las condiciones de elevación de la pluma, transporte, difusión y deposición. El modelo estima la concentración o el valor de deposición para cada combinación de fuente y receptor en forma horaria y calcula los promedios a corto plazo. El modelo tiene opciones para modelar emisiones provenientes de un amplio rango de fuentes que puedan estar presentes en un complejo industrial.

3. Modelo matemático utilizado en este trabajo

El modelo usa la ecuación de estado estable de pluma gaussiana para una fuente continua elevada. Para cada fuente y cada hora, el origen del sistema coordenado de las fuentes se coloca a nivel de la superficie en la base de la chimenea. El eje X es positivo en la dirección viento abajo, el eje Y está normal al eje X y el eje Z se extiende verticalmente. Las ubicaciones fijas de los receptores son convertidas para cada sistema coordenado de la fuente, para cada cálculo horario de concentración.

Las concentraciones horarias calculadas para cada fuente en cada receptor son sumadas para obtener la concentración total producida en cada receptor por la combinación de las emisiones de las fuentes.

Para una pluma Gaussiana de estado estable, la concentración horaria χ a una distancia X viento abajo (metros) y a una distancia Y contra el viento (en metros) está dada por:

$$\chi = \frac{QKV D}{2u_s \sigma_y \sigma_z \pi} \exp \left[-0.5 \left(\frac{y}{\sigma_y} \right)^2 \right]$$

Donde:

Q = tasa de emisión contaminante (masa por unidad de tiempo)

K = un coeficiente de escala para convertir las concentraciones calculadas a las unidades deseadas (valor por emisión = 10^6 para Q en g/s y la concentración en $\mu\text{g}/\text{m}^3$)

V = término vertical

D = término de decaimiento

σ_y, σ_z = desviación estándar de la distribución lateral y vertical de la concentración

u_s = velocidad promedio del viento (m/s) a la altura de la liberación de emisión.

y = la distancia viento cruzado hacia el receptor desde el centro de la pluma

Este modelo está programado en el paquete ISCST3. Se usó en los experimentos numéricos para resolver el problema directo, es decir, para el cálculo del nivel de contaminación del aire en los puntos de observación dentro de una región.

1.3 Experiencia en Geofísica de prospección

1. Problemática de solución del problema inverso

La idea principal en la que está basada la presente tesis está fundamentada en la metodología de solución de problemas inversos en Geofísica de prospección, cuyo enfoque principal, es la utilización de métodos de optimización aplicados a una función de criterio.

En Geofísica de prospección se buscan yacimientos de recursos naturales, el descubrimiento y la evaluación de los yacimientos son problemas inversos que se resuelven en base a datos experimentales de varios campos geofísicos como son los gravimétricos, eléctricos, magnéticos, etc. En el problema inverso los parámetros del yacimiento se consideran como los parámetros del modelo a determinar.

Como ya se mencionó en la sección 3 del párrafo 1.1 hay dos enfoques conocidos para determinar los parámetros del modelo:

- Enfoque directo, que esta basado en transformaciones de los datos experimentales y
- Enfoque indirecto, que se basa en la comparación de los datos de modelación y datos experimentales.

El enfoque directo permite evaluar solamente algunos parámetros integrales del yacimiento como su masa, localización del centro de masa etc. Para esto se usan varios métodos de transformación de los campos geofísicos. Este enfoque y su aplicación a varios problemas prácticos de otras áreas está descrito detalladamente en los libros de ingeniería [Hensel, 1991; Trujillo, 1997]. Los procedimientos de transformación de los datos experimentales que se usan bajo este enfoque se reducen a problemas incorrectos de física matemática. Aquí hay que usar los así llamados métodos de regularización [Heinz, 1996] que son el desarrollo de la teoría clásica de regularización de Tikhonov. Pero la regularización requiere de información a priori acerca de los parámetros probables del modelo. A menudo se carece de esta información.

El enfoque principal es el enfoque indirecto que permite evaluar los parámetros del modelo mucho más detalladamente. Aquí el problema inverso se reduce a la solución de un problema de optimización donde la función de criterio depende de la cercanía entre los datos experimentales y los datos de modelación. La forma concreta de esta función se define en base al interés del usuario y a los requisitos de precisión de la solución. Como se mencionaba anteriormente a menudo se usa la siguiente función cuadrática:

$$f(X) = 1/n \sum_{k=1}^n |F_m(X, D_k) - F_o(D_k)|^2$$

Donde:

$X = \{x_1, x_2, \dots, x_m\}$ = parámetros del modelo dado.

D_k = puntos de observación.

$F_m(X, D_k)$ = resultados de la modelación.

$F_o(D_k)$ = datos experimentales.

Desde luego se pueden proponer diversas formas para la función de criterio que satisfagan requisitos especiales.

La primera experiencia de la aplicación de los métodos de optimización para la interpretación de los datos geofísicos fue descrita en [Bulakh, 1972]. En este libro se consideró el modo automático cuando un usuario no toma parte en el control de búsqueda. Pero el modo automático encuentra muchas dificultades en caso de los modelos complejos. Estas dificultades se muestran en la figura 1.3-1, donde hay varios extremos relativos de la función y existe una zona de insensibilidad al cambio de valor en uno de los parámetros. Como consecuencia de esto, los métodos automáticos encuentran un mínimo local lejano del mínimo global y usan demasiados pasos de iteración. En los problemas reales cada paso del método requiere de muchos recursos de cómputo ya que hay que resolver el problema directo varias veces.

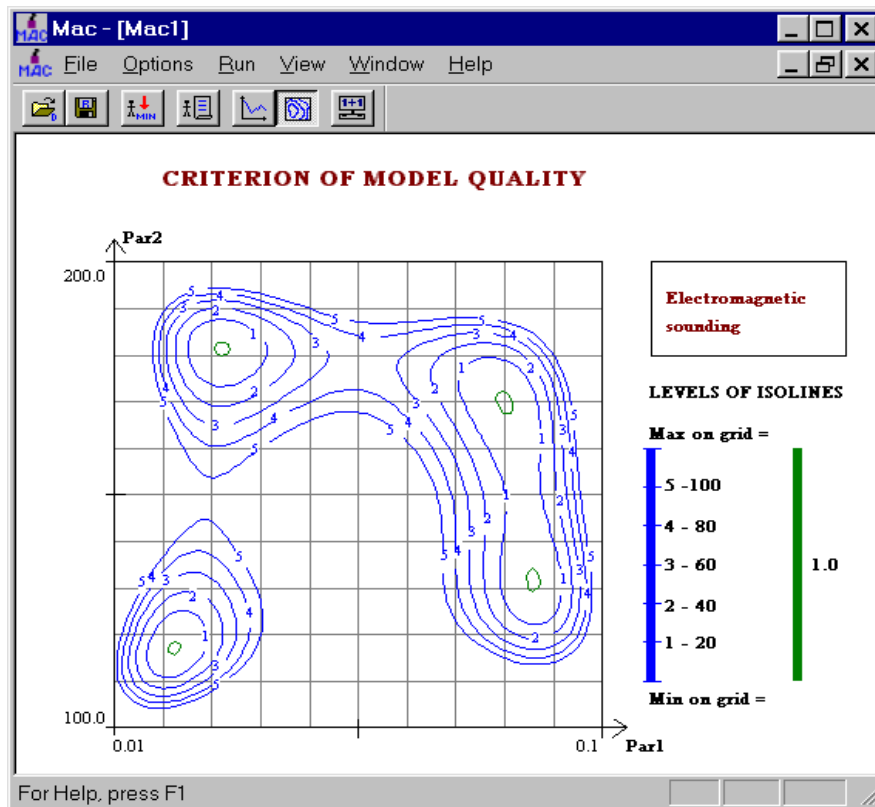


Fig. 1.3-1. Superficie típica ajustada de una función de criterio que se encuentra en problemas de Geofísica de prospección

Es necesario destacar que las dificultades descritas son consecuencia de la incorrectitud matemática del problema inverso, independientemente del enfoque aceptado (directo o indirecto). Por eso no hay que pensar que el enfoque de optimización puede ayudar a evadir las dificultades relacionadas con esta incorrectitud.

2. Experiencia sobre el problema inverso

Para disminuir las dificultades descritas los geofísicos empezaron la aplicación del modo interactivo gráfico. Este modo se usa en varias formas. Al principio los geofísicos rechazaron los métodos de optimización y en este caso en cada paso de la búsqueda, el usuario cambiaba los parámetros del modelo observando simultáneamente los datos de observación y los datos de modelación en una forma gráfica [Smith, 1972]. El cómputo se usa solamente para calcular el campo del modelo, pues se emplea el modo opcional cuando el sistema de búsqueda cambia los parámetros del modelo y el usuario evalúa los resultados de la modelación en cada paso de la búsqueda [Alexandrov, 1982; Yudin, 1983]. Este modo resulta útil en caso de gran incertidumbre acerca de la estructura del modelo que causa las dificultades en la formalización de la función de criterio.

Los geofísicos usaron y a la fecha continúan usando el modo semi-automático en donde el usuario sigue el proceso de búsqueda automática observando la dinámica de cambio de la función de criterio. Generalmente la búsqueda automática se realiza por medio de los métodos de programación no lineal o programación estocástica [Alexandrov, 1982, Tumarkin, 1989]. El usuario termina el trabajo del método cuando ve que la función no cambia y su valor está bastante lejos de un mínimo global. En este caso el usuario asigna un nuevo punto inicial o un nuevo dominio de búsqueda, o cambia de método de optimización. No se investigaron detalladamente como aprovechar combinaciones de los métodos de optimización ya que en Geofísica de prospección casi siempre las funciones de criterio son multi-extremo y cualquier sustitución del método de optimización no conduce a algún éxito. En la tesis esta cuestión se considera especialmente aplicada al problema de contaminación del aire. Los primeros resultados están publicados en la revista [Jurado, 2000] y en el reporte técnico [Alexandrov, 2000].

Nota.

Los modelos diferenciales que se usan en la modelación de contaminación del aire se refieren a la clase de modelos de transporte de masas y transporte de calor. Los problemas inversos para esta clase de modelos se consideran en [Alifanov, 1994, 1995; Kurpysz, 1995]. Pero todos estos libros están dedicados ante todo a cuestiones de matemática y no consideran especialmente los problemas de búsqueda de fuentes por medio de métodos de programación matemática.

1.4 Métodos de optimización

1. Clasificación de los métodos usados en este trabajo

Las funciones típicas de criterio que se usan en los problemas inversos de Geofísica son funciones no lineales y estas mismas se usan en el problema de determinación de los parámetros de fuentes locales. Para solucionar este problema se usan los métodos de programación no lineal y programación estocástica.

1. Métodos de programación no lineal

Este grupo de métodos de optimización es el más desarrollado y actualmente los esfuerzos principales se dirigen a la determinación de las combinaciones óptimas de ellos, que sean más adecuadas a las propiedades de la función de criterio.

Hay muchas variantes de clasificación de métodos no lineales.

Según la dimensión de problema:

- Métodos univariables
- Métodos multivariables

Muchos métodos multivariantes pueden trabajar en casos de una variable, pero algunos de ellos principalmente están orientados a trabajar únicamente en espacio de muchas variables como, por ejemplo, el método de Nelder-Mead.

Según la manera de búsqueda:

- Métodos de corte
- Métodos de paso a paso

Los métodos de corte no buscan un punto mínimo pero disminuyen el área de incertidumbre que contiene este punto mínimo. El ejemplo clásico es el método de la sección dorada. Los métodos de paso a paso buscan un punto mínimo. Todos estos métodos buscan un mínimo local ya que ellos estiman el comportamiento de la función en una vecindad del punto actual de la búsqueda. El ejemplo clásico es el método unidimensional de paso a paso. Estos métodos se conocen también como métodos de búsqueda dirigida.

Según la información sobre la función de criterio, la cual se usa para la búsqueda:

- Métodos de orden 0
- Métodos de orden 1
- Métodos de orden 2

Los métodos de orden 0 usan solamente los valores de la función de criterio. Un ejemplo es el método de Nelder-Mead. Los métodos de orden 1 usan tanto valores de la función como valores de sus derivadas. Un ejemplo es el método de descenso más rápido. Los métodos de orden 2 usan tanto valores de la función como valores de sus primeras y segundas derivadas. Un ejemplo es el llamado B-método.

2. Métodos de programación estocástica

Este grupo de métodos de optimización se divide según la manera de búsqueda:

- Métodos de aproximación estocástica
- Métodos de búsqueda aleatoria
- Métodos informativos estadísticos

Los métodos de aproximación estocástica suponen que los valores de la función no son conocidos exactamente, es decir, sus valores se dan con un error. Los métodos de búsqueda aleatoria trabajan con funciones cuyos valores son exactos, pero en el proceso de búsqueda se introduce artificialmente una aleatoriedad. Los métodos informativos estadísticos trabajan con funciones determinísticas, pero cada función se considera como el resultado de la elección aleatoria de un conjunto de funciones con propiedades comunes.

Ya que en el proceso de búsqueda la elección del siguiente punto tiene carácter aleatorio, entonces ningún mínimo local puede atraer al método de programación estocástica. Por eso todos los métodos de este grupo buscan un mínimo global.

Estas clasificaciones son la generalización del material presentado en [Himmelblau, 1992; Rastrigin, 1975].

2. Algunos resultados teóricos sobre la eficiencia de los métodos

Nemirovski y colaboradores [Nemirovski, 1989], plantearon el problema siguiente: evaluar la fuerza potencial de varias clases de métodos en varias clases de funciones. Ellos consideraron las siguientes clases de métodos: de gradientes (1er. orden), de paso a paso (orden 0), de segundo orden y aún, los métodos de búsqueda aleatoria. Las clases de funciones usadas fueron: convexas, suaves, y suaves de orden k. Sus resultados fueron los siguientes:

- Los métodos aleatorios no tienen ninguna ventaja sobre los métodos determinísticos, aunque hay quienes piensan que estos métodos son más poderosos en caso de los problemas de dimensión grande.
- Todas las evaluaciones no dependen del número de limitaciones del problema. Razón por la cual se pueden usar muchas limitaciones. La ventaja de las limitaciones es que disminuyen el área de incertidumbre.
- Para una dimensión grande del problema, las evaluaciones son muy pesimistas, pero es posible organizar la búsqueda en subespacios. Tal búsqueda puede ser programada en el marco del enfoque orientado a objetos. La tecnología correspondiente está descrita brevemente en el párrafo 2.5

Las evaluaciones concretas para algunas clases de funciones están presentadas en el Anexo 1.

3. Los métodos utilizados en este trabajo

La mejor manera de realizar la búsqueda del mínimo de una función de criterio es utilizando los métodos que mejor se ajusten a la superficie de la función. Para el caso de este trabajo existen muchas formas diferentes de funciones de criterio debido a la naturaleza misma de las variables. En una situación es necesario buscar los parámetros geográficos como serían las coordenadas de las chimeneas, en otra, se buscan los parámetros físicos como la velocidad de emisión, etc. La misma función de criterio en espacios diferentes tiene superficies diferentes, por eso es adecuado tener alguna biblioteca de métodos de optimización de donde un usuario pueda elegir los métodos adecuados a la función considerada. Este enfoque es bueno para los usuarios que trabajan en el área de optimización. Como el sistema desarrollado en este trabajo está dirigido a usuarios no especializados en esta área del conocimiento, se decidió no disponer de una amplia variedad de métodos de optimización.

Se presentan a continuación dos métodos de programación matemática, el primero perteneciente a los métodos de programación no lineal para buscar un mínimo local y el segundo perteneciente a los métodos de programación estocástica para buscar una sub-área del área total de búsqueda, que podría contener un mínimo global. Tal selección de los métodos refleja el enfoque del problema de la estrategia simple de búsqueda interactiva descrita en el párrafo 2.4.

Teniendo en cuenta la diversa variedad de superficies que se pueden generar con las funciones de criterio y el carácter complejo de las derivadas de estas funciones, se eligió un método de orden 0, concretamente el método de Nelder-Mead, el cual es muy común en aplicaciones relacionadas con problemas de ciencias naturales. De entre los métodos de programación estocástica se seleccionó el más sencillo, el método directo de búsqueda aleatoria. Ambos métodos están incluidos en la biblioteca de métodos de optimización de dominio público para cualquier programador. Esta biblioteca fue desarrollada en conjunto por el autor de este trabajo y por el Dr. Alexandrov, profesor del CIC-IPN y se presenta como un módulo externo en C++.

1. Método de Nelder-Mead

El método de Nelder-Mead (NM) es un método de programación no lineal, pertenece al grupo de métodos de orden 0, también se le conoce como el método del poliedro deformado o el método simplex no lineal [Himmelblau, 1992].

El método propuesto por Nelder-Mead en 1964 es un desarrollo del método de búsqueda mediante los simplex regulares [Himmelblau, 1992]. El principio fundamental del método de simplex regulares consiste en el análisis de los valores de la función de criterio en cada paso en todos los puntos del simplex dado, y la construcción del simplex nuevo sin el peor punto. Pero el uso de los simplex regulares tiene dos defectos: la ausencia de la aceleración en caso de que los pasos sean exitosos y la inadaptación al contorno del valle. El método de Nelder-Mead no tiene estos defectos y es especialmente efectivo para ser hacer búsquedas en los valles de las superficies.

A continuación se describe el algoritmo de este método para el caso de dos variables, visualizándolo en la figura 1.4-1 considerando un simplex en el área de búsqueda. Este simplex tiene $(n+1)$ nodos, donde n es la dimensión del espacio, en este caso $n=2$ (espacio de dos parámetros). El método evalúa la función en cada punto del simplex.

0. Se inicia con $n+1$ nodos $(x^{(1)}, x^{(2)}, x^{(3)})$ evaluando la función en cada uno.
1. Reflexión. Se toma el punto peor punto y el centroide del simplex. Estos son los puntos $x^{(1)}$ y $x^{(4)}$ respectivamente. Después se ejecuta la reflexión desde el peor punto a través del centroide obteniéndose el punto $x^{(5)}$. Aquí se usa el coeficiente de reflexión $\alpha \geq 1$.
2. Distensión. Aquí se consideran el punto reflejado y el mejor punto de todos los puntos del simplex. Sea el mejor punto $x^{(3)}$. En caso de éxito, es decir, si el punto $x^{(5)}$ es mejor que el punto $x^{(3)}$, entonces se ejecuta la distensión con el coeficiente de distensión $\gamma > 1$. Se obtiene el punto $x^{(6)}$. Y ahora si el punto $x^{(6)}$ es mejor que $x^{(3)}$ entonces el peor punto $x^{(1)}$ se sustituye por $x^{(6)}$ y se regresa al paso 1 del algoritmo. Pero si el punto $x^{(6)}$ no es mejor que $x^{(3)}$ entonces el peor punto $x^{(1)}$ se sustituye por $x^{(5)}$ y se regresa al paso 1 del algoritmo.
3. Compresión. En caso de no tener éxito, es decir, si el punto $x^{(5)}$ es peor que el mejor punto $x^{(3)}$ pero es mejor que el peor punto $x^{(1)}$ entonces se ejecuta la compresión del intervalo $x^{(1)}$ y $x^{(4)}$ con un valor del coeficiente de compresión en el intervalo $0 < \beta < 1$ sobre $x^{(1)}$. Se obtiene el punto $x^{(7)}$. Después se descarta el punto $x^{(1)}$ y se regresa al paso 1.
4. Reducción. Si el punto reflejado $x^{(5)}$ es peor que el peor punto $x^{(1)}$ entonces todos los vectores del poliedro se reducen en un factor de 0.5 sobre el mejor punto $x^{(3)}$ y se regresa al paso 1.

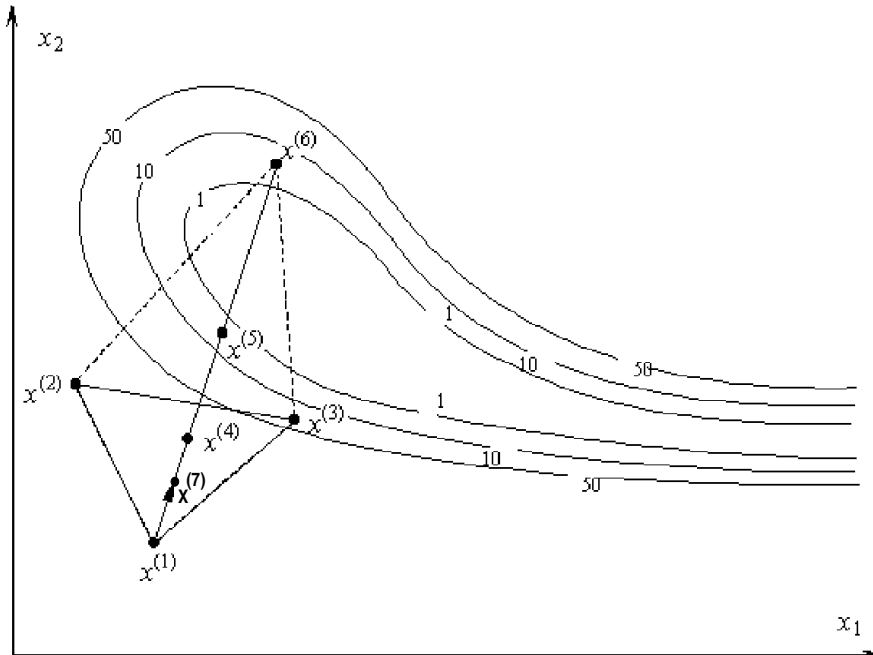


Fig. 1.4-1. Método de Nelder-Mead

En este trabajo se usa la versión del método Nelder-Mead descrita en el texto de [Himmelblau, 1992]. Se cambia la descripción del paso 3, ahora en el proceso de búsqueda toma parte el segundo peor punto, es decir un punto que es peor que todos los otros excepto el peor de los puntos.

0. Se inicia con $n+1$ nodos ($x^{(1)}, x^{(2)}, x^{(3)}$) evaluando la función en cada uno.
1. Reflexión. Se toman el peor punto y el punto del centro de gravedad del simplex. Estos son los puntos $x^{(1)}$ y $x^{(4)}$ respectivamente. Después se ejecuta la reflexión desde el peor punto por el centro de gravedad obteniéndose el punto $x^{(5)}$. Aquí se usa el coeficiente de reflexión $\alpha \geq 1$.
2. Distensión. Aquí se consideran el punto reflejado y el mejor punto entre todos los puntos del simplex. Sea el mejor punto $x^{(3)}$. En caso de éxito, es decir, si el punto $x^{(5)}$ es mejor que el punto $x^{(3)}$, entonces se ejecuta la distensión con el coeficiente de distensión $\gamma > 1$. Se obtiene el punto $x^{(6)}$. Y ahora si el punto $x^{(6)}$ es mejor que $x^{(3)}$ entonces el peor punto $x^{(1)}$ se sustituye por $x^{(6)}$ y se regresa al paso 1 del algoritmo. Pero si el punto $x^{(6)}$ no es mejor que $x^{(3)}$ entonces el peor punto $x^{(1)}$ se sustituye por $x^{(5)}$ y se regresa al paso 1 del algoritmo.
3. Análisis. En caso de no tener éxito, es decir, si el punto reflejado $x^{(5)}$ es peor que el mejor punto $x^{(3)}$ entonces se busca el segundo peor punto. Sea el punto $x^{(2)}$. Después se ejecuta el análisis:
 - Si el punto reflejado $x^{(5)}$ es mejor que $x^{(2)}$, entonces el peor punto $x^{(1)}$ se sustituye por $x^{(5)}$ y se regresa al paso 1 del algoritmo.

- Si el punto reflejado $x^{(5)}$ es peor que $x^{(2)}$ pero es mejor que el peor punto $x^{(1)}$ entonces el peor punto $x^{(1)}$ se sustituye por $x^{(5)}$ y se va al paso 4 del algoritmo.
 - Si el punto reflejado $x^{(5)}$ es peor que el peor punto $x^{(1)}$ se va al paso 4 del algoritmo.
4. Compresión. Se ejecuta la compresión del intervalo $x^{(1)}$ y $x^{(4)}$ con un valor del coeficiente de compresión en el intervalo $0 < \beta < 1$ sobre $x^{(1)}$. Se obtiene el punto $x^{(7)}$. Si $x^{(7)}$ es mejor que $x^{(2)}$ entonces el peor punto $x^{(1)}$ se sustituye por $x^{(7)}$ y se regresa al paso 1 del algoritmo. Al contrario se ejecuta el último paso 5.
 5. Reducción. Todos los vectores del poliedro se reducen en un factor de 0.5 sobre el mejor punto $x^{(3)}$ y se regresa al paso 1.

Todos estos pasos del algoritmo constituyen un ciclo de búsqueda.

¿Cuáles son los criterios para finalizar la búsqueda?, hay dos: el número máximo de pasos y el error definido. El error se evalúa como la distancia máxima entre el centro de gravedad del poliedro y sus nodos.

El método de Nelder-Mead se controla por 3 parámetros que son los coeficientes mencionados α , β , y γ . A veces se usa un cuarto coeficiente δ cuya variación es $0 < \delta < 1$ y que define el tamaño del simplex con relación al tamaño del área total al principio de la búsqueda. Los valores típicos para estos parámetros son: $\alpha = 1$, $\beta = 0.5$, $\gamma = 2$, $\delta = 0.1$.

2. Método de búsqueda aleatoria no dirigida

Este método es un método de programación estocástica, pertenece al grupo de métodos de búsqueda aleatoria. En él los puntos de prueba se eligen en forma aleatoria dentro del área de interés. No hay ninguna preferencia en cuanto a algunas direcciones o sub-áreas y por eso todos los puntos se disponen uniformemente sobre el área.

En este método la posición del punto siguiente no depende de las posiciones de todos los puntos seleccionados anteriormente, y por eso la cantidad de puntos aleatorios puede calcularse de antemano sobre la base de las condiciones de exactitud y probabilidad de la localización del mínimo global. La diferencia principal entre los métodos no dirigidos y los dirigidos de búsqueda aleatoria, radica en que los últimos cambian su estrategia de generación de puntos, en función de los valores que adquiere la función de criterio en los pasos anteriores. Tales métodos también se conocen como métodos de búsqueda aleatoria con preferencia.

El método no dirigido de búsqueda aleatoria generalmente utiliza un conjunto de generadores de números aleatorios que vienen a ser las coordenadas de los puntos aleatorios. Por eso la cantidad de generadores es igual a la dimensión del

problema. En este trabajo se considera el problema de optimización en un espacio de dos variables por lo que se utilizan solo dos generadores.

Ya que no existe preferencia alguna sobre una sub-área que pudiera contener al mínimo global, se deberán usar generadores de números aleatorios con una distribución uniforme. Por ejemplo, si los valores de la primer variable oscilan dentro del intervalo $[a_1, b_1]$ y los de la segunda variable oscilan dentro del intervalo $[a_2, b_2]$, entonces los generadores mencionados deberán tener distribución uniforme en $[a_1, b_1]$ y $[a_2, b_2]$ respectivamente, por lo contrario, si el usuario conoce alguna sub-área que pueda contener al mínimo, entonces los generadores deberán tener una distribución que refleje esta preferencia. Por ejemplo, la figura 1.4-2 demuestra el trabajo del método en este caso. Aquí la sub-área $[\tilde{a}_1, \tilde{b}_1] \times [\tilde{a}_2, \tilde{b}_2]$ es una sub-área de preferencia del usuario.

La relación entre el número puntos aleatorios N , el tamaño relativo de la vecindad del punto mínimo alcanzado que puede contener al punto mínimo global, el error de precisión ε y la probabilidad p , que refleja la certeza de que el mínimo global realmente se encuentra en esta vecindad en una distribución uniforme de puntos aleatorios [Rastrigin, 1975] es la siguiente:

$$N = [LN(1.-p) / LN(1.-\varepsilon)] + 1 \quad (1.4-1)$$

Aquí $p < 1$ y $\varepsilon > 0$. N crece cuando crece p y disminuye ε . Justamente esta fórmula se usa en el método incluido en la biblioteca desarrollada para este trabajo.

El método realiza exactamente N cálculos, si $N \leq N_{lim}$, donde N_{lim} = el número máximo de iteraciones asignado por el usuario. En caso contrario se toma $N = N_{lim}$ y se recalcula el nivel del error alcanzado usando la siguiente fórmula:

$$\varepsilon = 1. - (1.-p)^{1/N_{lim}} \quad (1.4-2)$$

La fórmula anterior se deriva de la fórmula (1.4-1). Hay que aclarar que significa ε . En el caso del problema unidimensional este valor equivale a la longitud del intervalo sobre el punto mínimo alcanzado, para el caso bidimensional equivale al cuadrado del área sobre el punto mínimo, etc.

El método no dirigido de búsqueda aleatoria se controla con un parámetro de probabilidad p . Este parámetro define la cantidad de experimentos. Generalmente se toma $p = 0.95$.

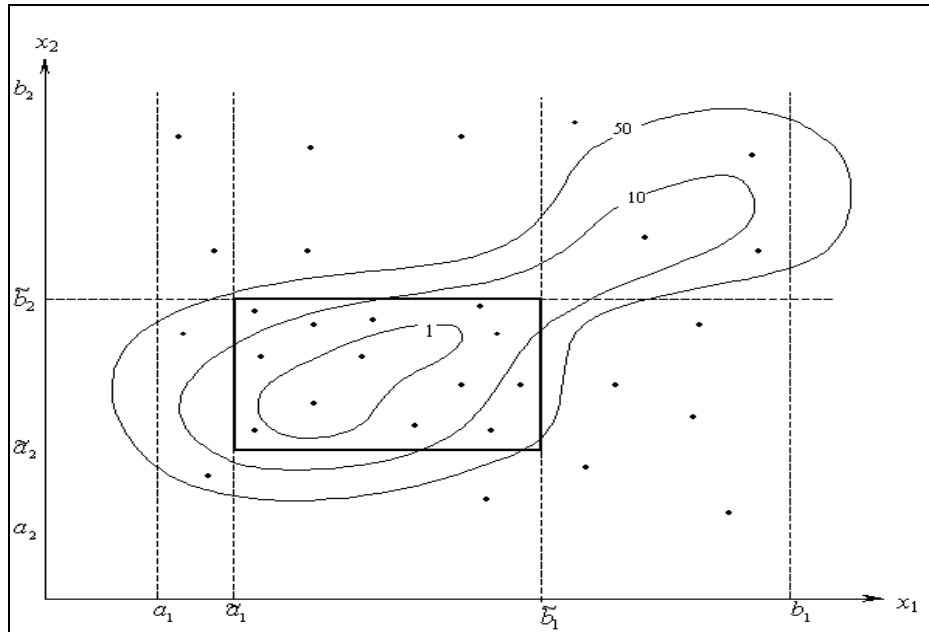


Fig. 1.4-2 Método no dirigido de búsqueda aleatoria

3. Contenido de la biblioteca

La biblioteca de métodos de optimización del sistema desarrollado contiene 2 métodos univariados y 2 métodos multivariados:

- Método tradicional de paso a paso. Es un método unidimensional, de búsqueda dirigida, de orden cero.
- Método de sección dorada. Es un método unidimensional, de corte, de orden cero.
- Método de Nelder-Mead. Es un método multidimensional, de búsqueda dirigida, de orden cero. Este método fue modificado como se describe en el párrafo 2.1
- Método no dirigido de búsqueda aleatoria. Es un método tanto unidimensional como multidimensional de búsqueda aleatoria.

Los métodos univariados se incluyeron para usarse en otras aplicaciones. Cabe mencionar que los problemas reales son problemas multivariados.

1.5 Resumen

Se mencionan las causas principales de la contaminación ambiental así como algunos de los modelos numéricos y paquetes de software que actualmente se utilizan para la modelación de los procesos de dispersión y transporte de contaminantes atmosféricos. Se dice que las calles y avenidas (o partes de ellos) con los vehículos pueden considerarse como fuentes locales en la escala de una ciudad grande.

Se formaliza el problema de búsqueda de fuentes de contaminación como problema de optimización para realizar un modo de búsqueda automática. Se considera la experiencia acumulada en geofísica. Las dificultades principales son: multi-extremidad e insensibilidad de la función criterio. Para disminuir estas dificultades es necesaria la participación del usuario en el proceso de búsqueda automática. Se presentan algunos resultados teóricos de la efectividad de algunos métodos y la descripción de los métodos multivariantes incluidos en la biblioteca desarrollada.

CAPITULO II.

Propuesta de solución

Este capítulo es el capítulo principal del trabajo. Ante todo desde el punto de vista de la optimización se consideran los requisitos para una función de criterio que permiten determinar los parámetros de las fuentes contaminantes. Se lleva a cabo una modificación al método NM propuesta en la literatura de optimización. Se investiga el comportamiento de este variando sus parámetros. Se propone una forma de utilizar información a priori en base a la función de preferencia. Para localizar en forma más eficiente el mínimo global se realiza la combinación de los métodos de búsqueda aleatoria y NM. Al final del capítulo se propone otra forma de búsqueda en un espacio de dimensión grande.

2.1 Función de criterio

1. Requisitos para una función de criterio

Como ya se mencionó en el capítulo anterior, el enfoque sobre la búsqueda de parámetros de las fuentes se reduce a la búsqueda de un mínimo global de una función de criterio que refleja la diferencia entre los datos de observación y los datos de la modelación. La selección del criterio es un procedimiento subjetivo, depende de la experiencia del usuario, de la precisión de los datos de observación, etc., pero hay unos requisitos generales los cuales deben satisfacer todas estas funciones. La referencia sobre un ejemplo de aplicación a la Geofísica en donde se describen estos requisitos, se da en [Alexandrov, 1987].

Como se definió en el párrafo 1.1, una función de criterio se presenta en la forma siguiente:

$$f(x_1, x_2, \dots, x_n) = f[F_m(X, D), F_o(D)]$$

Donde:

f = valor de la función de criterio,

F_m = resultados de la modelación,

F_o = datos de observación,

$X = \{x_1, x_2, \dots, x_n\}$ = parámetros del modelo aceptado,

D = dominio de simulación.

En caso de que el modelo aceptado describa correctamente los procesos que se observan, entonces es posible encontrar un punto $(x_1^*, x_2^*, \dots, x_m^*)$ en el espacio de los parámetros donde $F_m(x_1^*, x_2^*, \dots, x_m^*, D) = F_o(D)$. Por eso todas las funciones del tipo:

$$f(x_1, x_2, \dots, x_m) = f[F_m(x_1, x_2, \dots, x_m, D) - F_o(D)]$$

satisfacen la condición $f[\cdot]=0$. Además es mas cómodo tener $f[\cdot] \geq 0$ para analizar la distancia entre el valor corriente de la función de criterio y su mínimo global.

Por ejemplo, se puede tomar:

$$f(x_1, x_2, \dots, x_m) = 1/n \sum_{k=1}^n |F_m(x_1, x_2, \dots, x_m, D_k) - F_o(D_k)|^p$$

donde $p=1, 2, 4, \dots$

Para recalcar las diferencias grandes entonces p debe ser un número grande, en caso contrario se puede elegir $p=1$. Se pueden usar diversos tipos de funciones, por ejemplo,

$$f(x_1, x_2, \dots, x_m) = \exp \left(\sum_{k=1}^n |F_m(x_1, x_2, \dots, x_m, D_k) - F_o(D_k)|^2 \right) \text{ etc.}$$

Pero en cualquier caso, lo ideal es utilizar la función más simple para simplificar la interpretación de los resultados finales. Resulta cómodo cuando la función de criterio puede ser interpretada fácilmente por el usuario. Por ejemplo, la función puede ser homogénea dimensionalmente para los datos de observación, y en este caso la dimensión es $[\mu\text{g}/\text{cm}^3]$. Con este punto de vista el último criterio no es cómodo (no tiene ningún sentido), y el criterio anterior debe corregirse:

$$f(x_1, x_2, \dots, x_m) = \left(\sum_{k=1}^n |F_m(x_1, x_2, \dots, x_m, D_k) - F_o(D_k)|^p \right)^{1/p}$$

Es recomendable que la función de criterio:

- no dependa de la cantidad de datos de observación y que
- dependa de la precisión de medición.

En este caso se puede usar, por ejemplo, la función:

$$f(x_1, x_2, \dots, x_m) = \left(\left[\sum_{k=1}^n \alpha_k |F_m(x_1, x_2, \dots, x_m, D_k) - F_o(D_k)|^p \right] / n \right)^{1/p} \quad (2.1-1)$$

donde α_k = coeficiente que refleja la precisión de los datos de observación en el punto k .

Obviamente que en el punto mínimo global el valor de función $f(\cdot)$ debe ser cercano al error promedio de medida. Este hecho se usa para evaluar si el mínimo alcanzado es un mínimo local o global. Cabe mencionar nuevamente que la mayoría de las funciones de criterio que se usan en aplicaciones reales tienen muchos extremos locales.

Generalmente los parámetros del modelo (x_1, x_2, \dots, x_n) tienen naturaleza distinta y por eso sus valores se diferencian en algunos ordenes. Bajo esta condición, los

métodos de optimización a menudo no trabajan correctamente perdiendo la precisión de cálculo en cada paso. Por eso en el caso de la optimización multidimensional hay que usar los parámetros normalizados, es decir:

$$\tilde{x} = (x - x_{min}) / (x_{max} - x_{min}) \quad (2.1-2)$$

2. Construcción de una función específica de criterio

Las funciones de criterio utilizadas en aplicaciones sobre problemas naturales, son superficies complejas con muchos extremos locales, por ejemplo, una vista típica suavizada de las funciones que se encuentran en Geofísica subterránea se muestra en la Fig. 1.3-1. Para las investigaciones realizadas en este trabajo se construyeron las funciones de criterio de la manera siguiente:

- 1) Se utilizaron algunos datos de las concentraciones de los contaminantes NO_x medidas en 11 sitios de monitoreo situados dentro de un área aproximada de 15 km^2 alrededor de la refinería Miguel Hidalgo ubicada en la ciudad de Tula, Hgo. y de la refinería de Cadreyta. También se usó información referente a los parámetros físicos de las fuentes emisoras, tales como la altura de las chimeneas, la velocidad de las emisiones, la temperatura de estas, las coordenadas geográficas, etc. Estos datos se designan como $F_o(D_k)$, donde D_k = puntos de observación.
- 2) Se seleccionaron dos parámetros que reflejaran características específicas de estas chimeneas, por ejemplo, la velocidad de emisión, la temperatura de los gases en la salida de chimeneas, sus coordenadas, etc. Estos parámetros se denominaron parámetros del modelo. Los parámetros seleccionados fueron x_1 y x_2 y se fijaron sus intervalos de cambio (x_{1min} , x_{1max}) y (x_{2min} , x_{2max}). Los parámetros reales x_1^* , x_2^* de las chimeneas deben pertenecer a estos intervalos, es decir: $x_1^* \in (x_{1min}, x_{1max})$ y $x_2^* \in (x_{2min}, x_{2max})$.

Dentro del área construida se tomó una malla regular por cada eje y como resultado se obtuvo una matriz ($x_{1,i}$, $x_{2,j}$).

x_{2max}					
...					
$x_{2,j}$					
...					
x_{2min}					
	x_{1min}	...	$x_{1,j}$...	x_{1max}

Tabla 2.1-1. Matriz de puntos para el cálculo de la función de criterio

- 1) Para cada combinación de los parámetros $(x_{1,i}, x_{2,j})$ se calcularon valores de concentración de CO_2 en todos los puntos de observación, es decir $F_m(x_{1,i}, x_{2,j}, D_k)$. La modelación se llevó a cabo usando el modelo ISCST3 que ha sido usado en estudios ambientales en el Instituto Mexicano del Petróleo [Technical manual, 1995a, 1995b].
- 2) Usando la formula (2.1-1) se determinó la matriz de valores de la función de criterio. En los experimentos realizados en este trabajo se usó la formula siguiente:

$$f(x_{1,i}, x_{2,j}) = \left(\left[\sum_{k=1}^n |F_m(x_{1,i}, x_{2,j}, D_k) - F_o(D_k)|^2 \right] / n \right)^{1/2} \quad (2.1-3)$$

Los valores de esta función forman una superficie en el espacio de parámetros del modelo (x_1, x_2) . En estos experimentos se consideraron las superficies obtenidas usando una matriz cuadrada de orden 5. Los valores de la función de criterio en todos puntos intermedios de la matriz se calcularon mediante interpolación cuadrática. Los ejemplos de varias matrices se presentan en el apéndice de este trabajo.

Los procedimientos para el cálculo de la matriz de la función de criterio y de la interpolación de sus valores no se aplican con las funciones de prueba incluidas en el sistema desarrollado. En este caso el valor de la función se calcula directamente en cada punto de búsqueda.

3. Funciones de criterio ambientales

A continuación se muestran cinco gráficas con las isolíneas de las funciones de criterio generadas con los datos experimentales mencionados y con los resultados de las simulaciones realizadas con el modelo ISCST3. Estas funciones se utilizaron para probar la técnica de búsqueda del mínimo global propuesta en esta tesis.

Función No.1.

Se define al parámetro 1 como la velocidad de emisión y al parámetro 2 como la temperatura de salida de los gases. Los intervalos de valores para estos parámetros son $[0.1, 8]$ m/sec. y $[300, 480]$ C° respectivamente. Los valores de la función de criterio están en el archivo Exp1_mac.txt.

La función de criterio se presenta en forma de isolíneas numeradas de acuerdo al porcentaje que representan los datos con relación a los valores mínimo y máximo del conjunto total de ellos, como lo indica la escala azul a la derecha de la gráfica, (esto es, # de isolínea = valor mínimo + % (valor máximo – valor mínimo). Las isolíneas verdes indican los valores que están un 5% cerca del mínimo global de la función de criterio (relación análoga). En el rectángulo superior derecho se indican

el método de búsqueda utilizado y la dimensión del problema. Los ejes corresponden al espacio de parámetros.

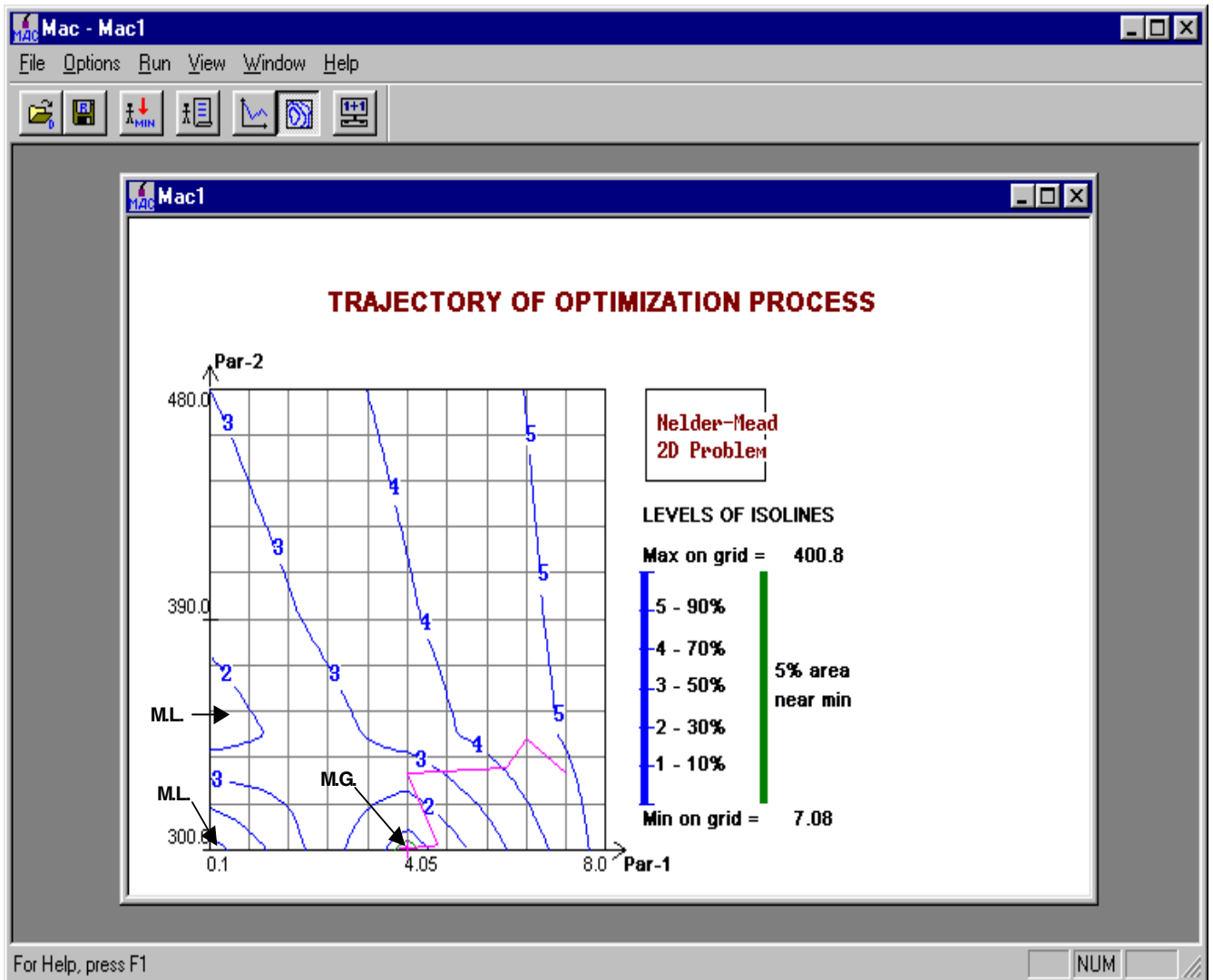


Fig. 2.1-1. Función de criterio No.1.

La superficie de criterio que se muestra en la Fig. 2.1-1 presenta dos mínimos locales (M.L.) y un mínimo global (M.G.). La línea y los puntos rojos en la figura reflejan el proceso de búsqueda de un óptimo por el método NM. Los valores de las funciones de criterio están en el apéndice en los archivos Exp1_mac.txt a Exp4_mac.txt

Función No.2.

Se definen los parámetros igual que en el ejemplo anterior con los mismos intervalos de variación. En este caso la función de criterio presenta un mínimo local y un mínimo global.

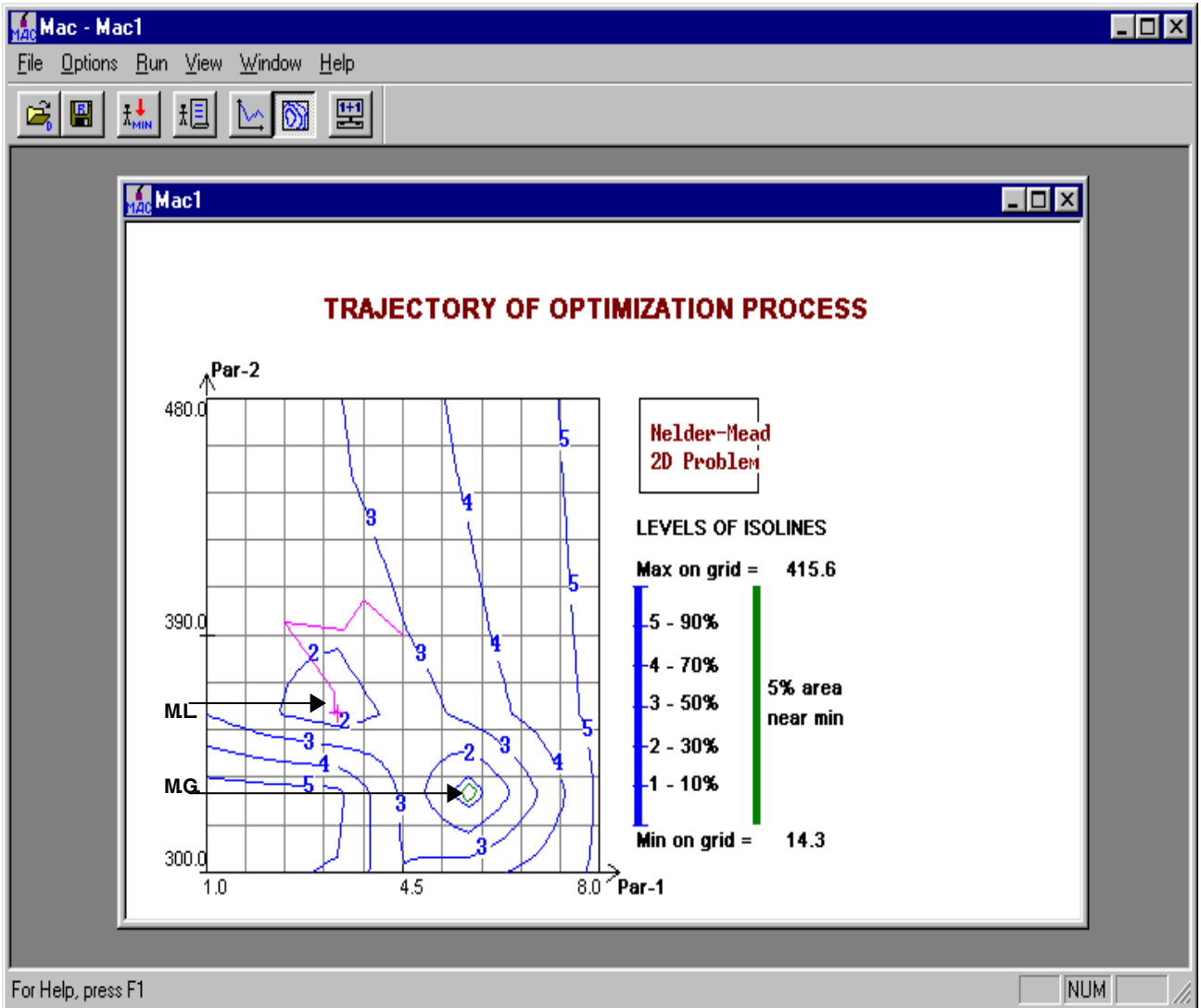


Fig. 2.1-2. Función de criterio No.2.

Función No.3.

Se definen los parámetros como las coordenadas georeferenciadas de dos chimeneas. Donde el rango de variación para el parámetro 1 es $[-50,-10]$ y el del parámetro 2 de $[-70,-30]$. Esta función de criterio tiene un mínimo local (M.L.) y un mínimo global (M.G.) como se indica en la figura.2.1-3.

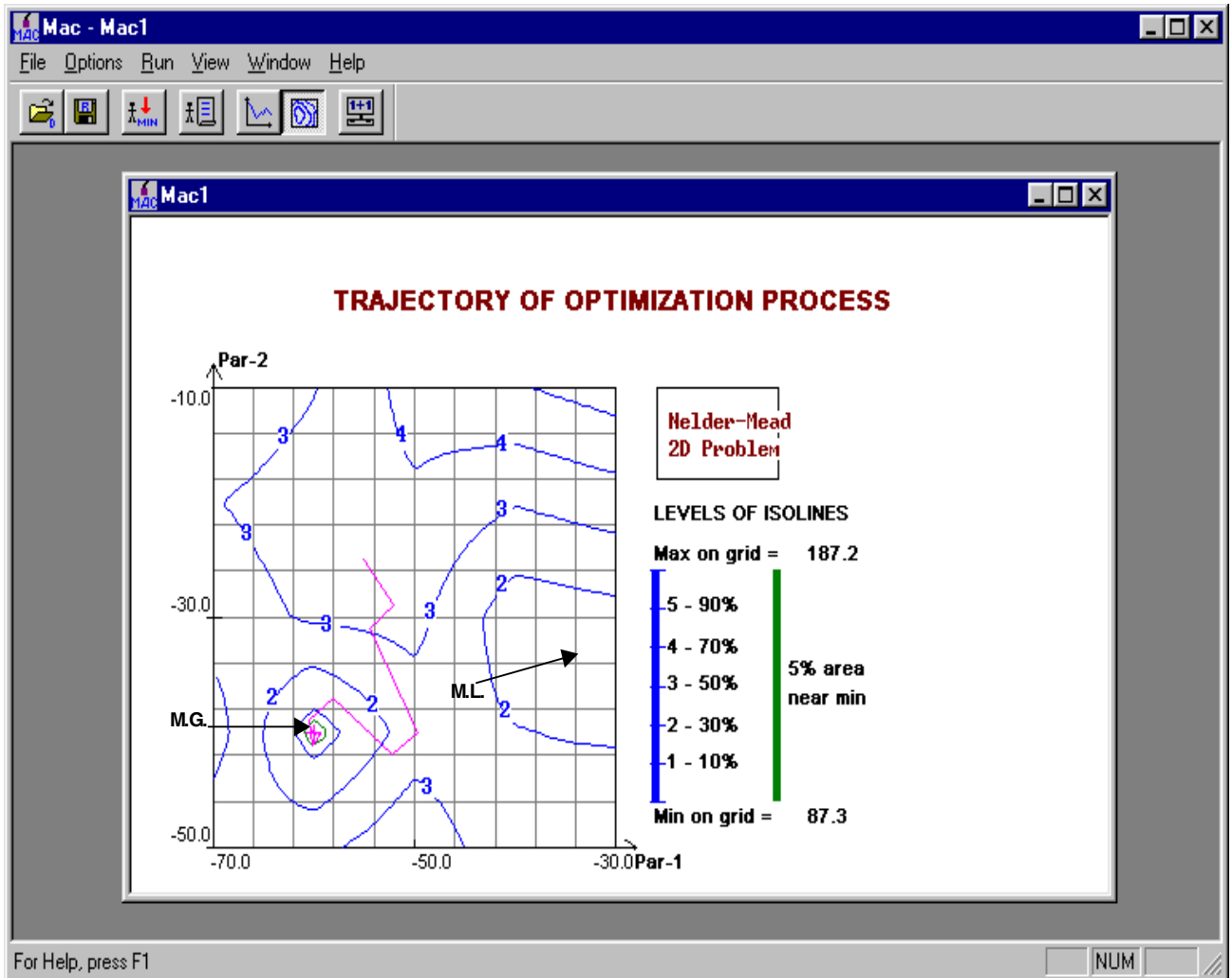


Fig. 2.1-3. Función de criterio No.3.

Función No.4

Se definen los parámetros como las velocidades de emisión Qs_1 y Qs_2 de los contaminantes emitidos por las chimeneas, cuyos rangos de variación son $[4.0,8.0]$. La función de criterio presenta un mínimo global y dos locales como se muestra en la figura 2.1-4.

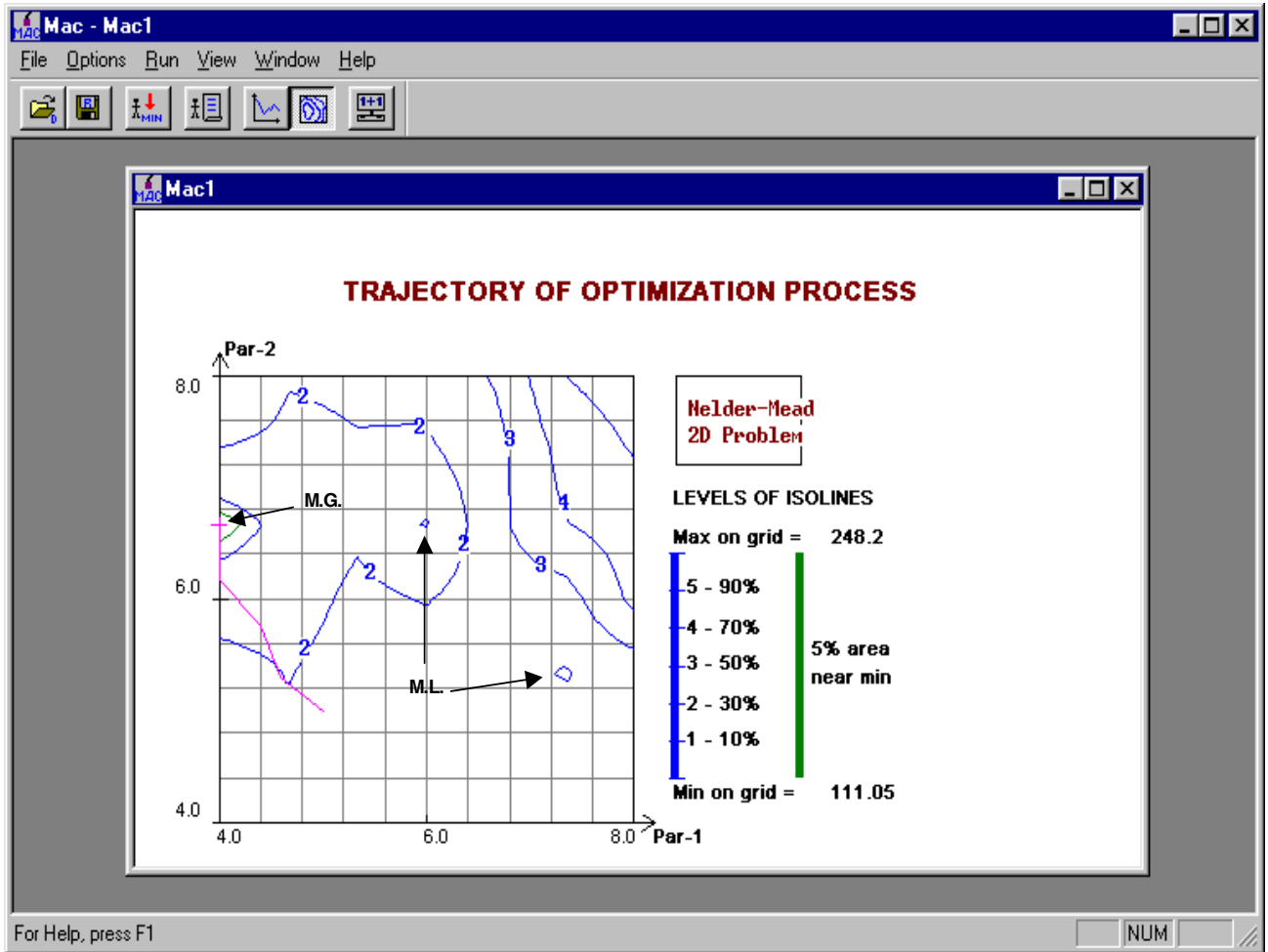


Fig. 2.1-4. Función de criterio No.4.

Función No.5.

Se define el parámetro 1 como la coordenada geográfica longitudinal de la chimenea y el parámetro 2 como la tasa de emisión del contaminante. Los respectivos rangos de variación son $[0.1,0.5]$ y $[-30.0,-10.0]$. La función de criterio presenta un solo extremo global a la vez como se muestra en la figura 2.1-5.

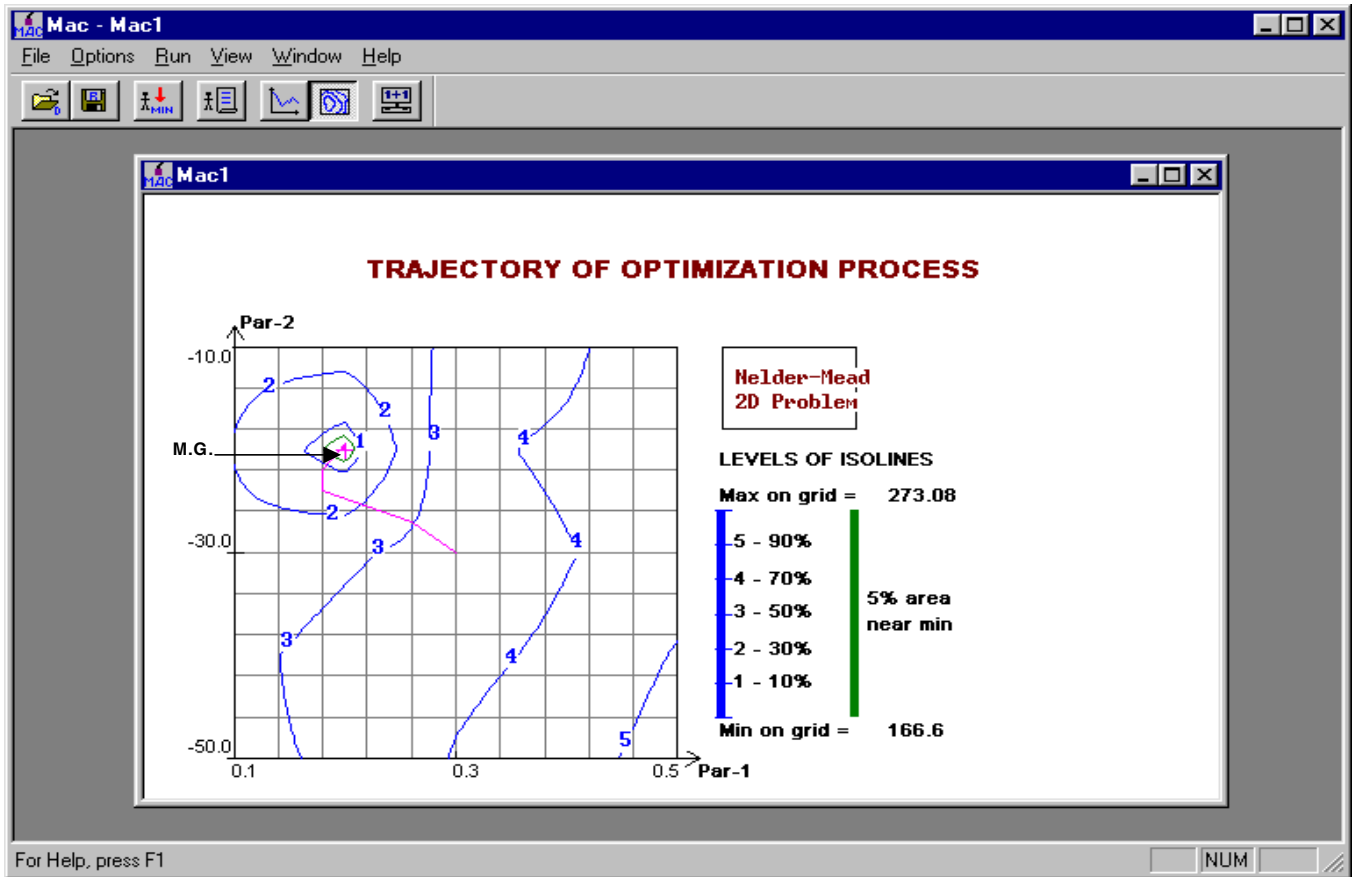


Fig. 2.1-5. Función de criterio No.5.

Este ejemplo muestra que no siempre se tienen funciones de criterio multiextremo por lo que la técnica propuesta en esta tesis no es necesaria para la búsqueda de óptimo global.

2.2 Ajuste de los parámetros del método NM

1. Fórmula general para comparar las variantes de un método

Como ya se mencionó en el párrafo 1.4, el método NM tiene algunos parámetros que definen su comportamiento en un procedimiento de búsqueda. El problema es encontrar tales parámetros que proveen la convergencia más rápida con la clase funciones ambientales. Para esto es necesario comparar las variantes del método NM usando combinaciones diferentes de sus parámetros.

Para realizar esta competencia hay que evaluar los esfuerzos de cada variante del método para alcanzar un óptimo en una clase dada de funciones desde un punto aleatorio inicial. Desde luego la dimensión de problema y el error final deben fijarse.

Los esfuerzos de búsqueda generalmente se consideran como una suma de esfuerzos para el cálculo de la función $f(x^{(k)})$ [o sus derivadas parciales $f_i''(x^{(k)})$ en caso de métodos de primero o segundo orden] y para la determinación del siguiente punto $x^{(k)} \rightarrow x^{(k+1)}$. Comúnmente la segunda parte de esta suma requiere de menos esfuerzo que la primera. La fórmula general para calcular este esfuerzo es [Batichev, 1982]:

$$\chi(k, \varepsilon) = \frac{1}{mn} \sum_{i=1}^m \sum_{j=1}^n \chi_{i,j}(k, \varepsilon) \quad (2.2-1)$$

Donde:

$\chi_{i,j}(k, \varepsilon)$ = esfuerzos para la búsqueda del punto inicial i usando la función j ,

k = dimensión del problema, ε = error de búsqueda, m = el número de puntos iniciales de la búsqueda, n = número de funciones de prueba. Se supone desde luego que todas estas funciones pertenecen a la misma clase.

Nota.

Obviamente que la formula (2.2-1) puede usarse no solo para comparar métodos diferentes sino las variantes de un método. Se utilizará para encontrar los parámetros óptimos del método NM.

2. Parámetros óptimos del método NM

El método NM tiene 4 parámetros: α es el coeficiente de reflexión, β el coeficiente de compresión, γ el coeficiente de extensión y adicionalmente se introduce el coeficiente δ que refleja los tamaños iniciales relativos del poliedro. ¿Cuales son los valores óptimos para asignar a estos parámetros?, esta pregunta se consideró

detalladamente en la investigación de Paviani. Las referencias a este trabajo están en el texto de Himmelblau [Himmelblau, 1992].

Para este trabajo se trataron de encontrar los valores óptimos de dichos parámetros que hicieran que el método utilizara el mínimo promedio de pasos de búsqueda, es decir, que dieran los menores valores de la ecuación (2.4-1). Esta minimización se realizó por simple observación de los resultados numéricos.

Analizando el parámetro α . Para éste se considera la situación cuando el poliedro deformado está calibrado de acuerdo con la forma del valle de manera apropiada, su forma debe mantenerse constante hasta que los cambios en la topología de la función de criterio requieran una nueva forma del poliedro. Esto es posible realizarlo solamente para $\alpha=1$. Además Nelder y Mead mostraron prácticamente que con este valor se requieren menos cálculos de la función que con $\alpha<1$. Por otro lado α no debe ser mayor que 1 esencialmente ya que:

- a) el poliedro se adaptó más fácilmente a la topología de la función con valores de α menores que 1, especialmente cuando fue necesario cambiar la dirección de búsqueda rápidamente.
- b) en la vecindad del mínimo local los tamaños del poliedro deben reducirse y bajo estas condiciones si $\alpha >1$, entonces esta circunstancia alenta la convergencia, por esta razón se eligió $\alpha=1$.

Nelder, Mead y Paviani resolvieron un conjunto de problemas con varias combinaciones de β y γ . Nelder y Mead recomiendan como valores satisfactorios $\beta=0.5$ y $\gamma=2$. Paviani recomienda $0.4\leq\beta\leq 0.6$ y $2.8\leq\gamma\leq 3$. El tamaño y la orientación inicial no influyeron significativamente en el tiempo de solución pero α, β, γ si lo hicieron.

Se realizó el mismo experimento con datos experimentales. Se consideraron 3 funciones de criterio, se utilizaron 3 puntos iniciales para la función y 9 combinaciones de parámetros. El error establecido fue de 10^{-3} y el número máximo de iteraciones fue de 50. Los resultados del experimento se presentan en la Tabla 2.2-1. Aquí: F.1, F.2, F.3 y F.4 designan el número de la función de criterio y P.1, P.2, P.3 designan el número de punto de búsqueda.

Experi- mento	Beta	Gamma	F.1 P.1	F.1 P.2	F.1 P.3	F.2 P.1	F.2 P.2	F.2 P.3	F.3 P.1	F.3 P.2	F.3 P.3	Media χ
1	0.2	1.5	9	9	10	9	16	12	27	22	23	15.2
2	0.2	2	7	7	8	7	15	10	21	22	24	13.4
3	0.2	2.5	6	7	7	6	9	16	22	22	23	13.1
4	0.5	1.5	9	9	10	9	20	12	24	23	23	15.4
5	0.5	2	7	7	8	7	17	10	21	20	19	12.9
6	0.5	2.5	6	7	7	6	9	16	25	20	25	13.4
7	0.7	1.5	9	9	10	9	31	12	25	23	22	16.6
8	0.7	2	7	7	8	7	22	10	20	18	19	13.1
9	0.7	2.5	6	7	7	6	9	16	23	20	22	12.9

Tabla 2.2-1 Serie de experimentos para evaluar los parámetros del método Nelder-Mead.

Calculando promedios χ para cada combinación se obtuvo que $0.5 \leq \beta \leq 0.7$ y $2 \leq \gamma \leq 2.5$ son los rangos de valores preferibles para los parámetros. Usando la formula (2.4-1) se puede calcular que para la clase de funciones de criterio ambientales utilizadas y para el método de Nelder-Mead el esfuerzo $\chi(2, 10^{-3})=14$.

En cuanto a δ (el tamaño relativo del poliedro) el valor aceptado fue 0.1 cuando se está en la vecindad del extremo (desde el punto de vista del usuario) ó 0.2 cuando se está lejos del extremo (de acuerdo a la experiencia en geofísica).

La elección de los parámetros óptimos del método de Nelder-Mead se hizo en base a funciones reales de criterio en espacio de dos variables.

2.3 Aplicación de una función de preferencia

1. Evaluaciones teóricas

Cuando se dice que se tiene una información a priori entonces se sobreentiende que se conoce una sub-área que contiene a los extremos que se buscan, en ese caso simple, ésta sub-área se representa en la forma de desigualdades:

$$\bar{a}_i \leq x_i \leq \bar{b}_i \quad (2.3-1)$$

donde $i=1,2... n$. Hay que mencionar que todos los métodos considerados anteriormente trabajan en un marco de limitaciones parecidas:

$$a_i \leq x_i \leq b_i \quad (2.3-2).$$

Y entonces, ¿para que usar otras limitaciones complementarias? La diferencia consiste en que las condiciones (2.3-1) reflejan una preferencia, es decir, se permite buscar un extremo fuera de este intervalo; por lo contrario, las condiciones (2.3-2) prohíben cualquier salida del área.

Aquí se consideran los métodos de programación no lineal. Todos estos métodos empiezan su movimiento en dirección del punto mínimo local. Generalmente las funciones de criterio con las cuales se trata tienen muchos extremos y por lo tanto hay que tomar algunos puntos aleatorios en la sub-área (2.3-1) y después realizar la búsqueda desde cada punto como punto inicial. Enseguida es necesario tomar el punto mínimo con el menor valor de la función, pero si se tiene una información más exacta sobre el comportamiento de la función de criterio, entonces se puede usar otra manera de búsqueda que sea más eficaz. Esta es la aplicación de la así llamada función de preferencia.

La idea consiste en lo siguiente: una función original se cambia por otra para que no tenga el mínimo local en una vecindad del mínimo absoluto. La función de preferencia se toma en la forma:

$$C\|x - x^{(p)}\| \quad (2.3-3)$$

donde $x^{(p)} = (x_1^{(p)}, x_2^{(p)}, \dots, x_n^{(p)})$ es el punto de preferencia y C es llamado el coeficiente de multa. Este miembro se añade a la función de criterio obteniéndose una función suma de dos funciones, claro que si más se aleja del punto $x^{(p)}$, más se sanciona. Este enfoque es análogo a los métodos de funciones de multa descritos en los trabajos clásicos [Fiacco, 1968] y en la literatura [Himmelblau, 1992].

¿Cómo hay que asignar este coeficiente de multa C ?

Primeramente se considera una función unidimensional $f(x)$. En este caso (2.3-3) se presenta como una diferencia absoluta $C|x - x^{(p)}|$. Considérese que esta función $f(x)$ satisface la condición de Lipschitz, es decir,

$$|f(x_1) - f(x_2)| \leq L|x_1 - x_2|, \text{ para } \forall x_1, x_2 \in (\tilde{a}, \tilde{b}) \quad (2.3-4)$$

donde (\tilde{a}, \tilde{b}) es un intervalo de preferencia y L es la constante de Lipschitz. A propósito si la función es suave dentro de (\tilde{a}, \tilde{b}) entonces se puede evaluar L . Realmente la desigualdad se presentaría en la siguiente forma conocida:

$$|f(x_1) - f(x_2)| \leq \max |f'(x)| |x_1 - x_2|$$

donde $x \in (\tilde{a}, \tilde{b})$. Por eso se puede decir que $L = \max |f'(x)|$, donde $x \in (\tilde{a}, \tilde{b})$. Este máximo existe, ya que la función $f(x)$ es suave y su sub-área está limitada.

Considérese que el punto $x^{(a)}$ es un mínimo absoluto y $x^{(l)}$ es un mínimo local. Acéptese también que se conoce el punto $x^{(a)}$, y por eso se puede asignar que $x^{(p)}=x^{(a)}$. Entonces se tiene una nueva función:

$$F(x) = f(x) + C|x - x^{(a)}|$$

que debe excluir la posibilidad para que el método alcance el punto $x^{(l)}$ en vez de $x^{(a)}$ en la vecindad de $x^{(a)}$.

El punto x^* es el punto de mayor valor posible de la función $f(x)$ en el intervalo $[x^{(l)}, x^{(a)}]$. Este punto puede determinarse fácilmente a partir de la igualdad:

$$f^{(l)} + K(x^* - x^{(l)}) = f^{(a)} + K(x^{(a)} - x^*)$$

La justificación de esta fórmula sigue de la Fig.2.3-1. Aquí

$$Tg\alpha = L, f(x^{(l)}) = f^{(l)}, f(x^{(a)}) = f^{(a)}, f(x^*) = f^{(*)}.$$

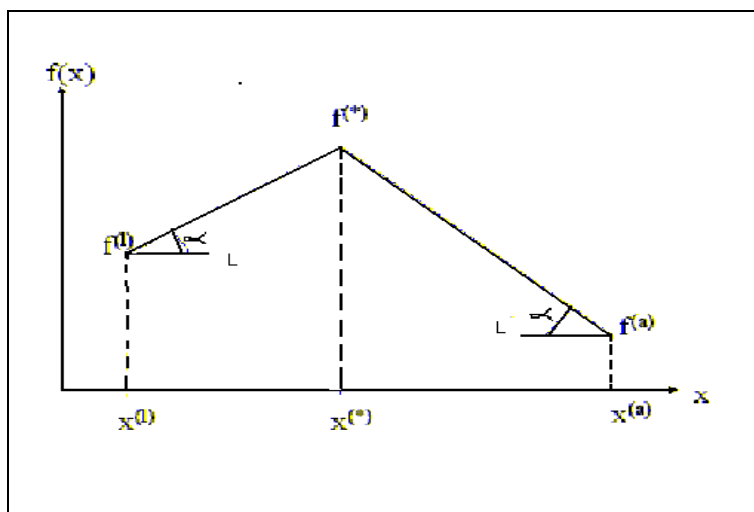


Fig. 2.3-1 Justificación de la asignación del coeficiente de multa

El punto $x^{(l)}$ no debe 'atraer' al método, entonces es necesario proveer:

$$F(x^{(l)}) \gg F(x^{(*)}) \text{ ó } f^{(l)} + C|x^{(l)} - x^{(a)}| \geq f^{(*)} + C|x^{(*)} - x^{(a)}|$$

Entonces debe ser:

$$\frac{f^{(*)} - f^{(l)}}{x^{(*)} - x^{(l)}} \leq C \quad (2.3-5)$$

Por otro lado, según (2.3-4):

$$|f^{(*)} - f^{(l)}| \leq L|x^{(*)} - x^{(l)}| \quad \text{ó} \quad \frac{f^{(*)} - f^{(l)}}{x^{(*)} - x^{(l)}} \leq L \quad (2.3-6)$$

De (2.3-5) y (2.3-6) obviamente se sigue que debemos concluir que $C \geq L$, ya que:

$$C \geq \frac{f^{(*)} - f^{(l)}}{x^{(*)} - x^{(l)}} \quad \text{para } \forall x^{(*)}, f^{(*)}, \text{ entonces } C \geq \max \frac{f^{(*)} - f^{(l)}}{x^{(*)} - x^{(l)}} = L$$

Análogamente se puede probar esta regla en el caso de funciones multidimensionales.

Lamentablemente esta evaluación es teórica, ya que trabajando con datos experimentales se desconoce el punto mínimo absoluto, y la constante de Lipschitz, por lo que se puede solamente esperar que el punto de preferencia $x^{(p)}$ esté junto al punto mínimo absoluto $x^{(a)}$, y si se conoce aproximadamente el valor máximo de $|f'|$ donde $x \in [a, b]$, entonces la función de preferencia podrá ayudar a dirigir el método por la ruta correcta.

En el sistema MAC la función de preferencia (2.3-3) se usa de la manera siguiente:

- a) En el caso unidimensional esta función es simplemente una diferencia absoluta que se considera arriba.
- b) En el caso multidimensional como ya se ha notado en el párrafo 2.1 todas las variables se normalizan a sus intervalos de cambio. Por eso (2.3-3 en caso de dos variables) se presenta en el sistema por lo siguiente:

$$C \left[\sqrt{(\tilde{x}_1 - x_1^{(p)})^2 + (\tilde{x}_2 - x_2^{(p)})^2} \right]$$

donde $\tilde{x} = (x - x_{\min}) / (x_{\max} - x_{\min})$ y $x^{(p)} = (x_1^{(p)}, x_2^{(p)}, \dots, x_n^{(p)})$ es el punto de preferencia

Una aproximación del cálculo de la constante de Lipschitz es

$$L \sim \left| \frac{f_{\max} - f_{\min}}{\delta_{\max, \min}} \right| \quad (2.3-7)$$

donde f_{\max} =valor máximo de la función dentro del intervalo de interés.

f_{\min} =valor mínimo de la función dentro del intervalo de interés.

$\delta_{\max, \min}$ = distancia entre los dos puntos.

En caso de trabajar en un espacio multivariable $\delta_{\max, \min}$ es adimensional que varía dentro del intervalo $[0, \sqrt{2}]$.

2. Experimentos numéricos

Para justificar las ventajas de usar la función de preferencia se usó el ejemplo considerado en el párrafo 1.1. En este ejemplo se considera la función de criterio presentada en la Fig. 2.1-1. Esta función de criterio tiene un mínimo local en el punto (0.1, 345.) y un mínimo global en el punto (4., 300.).

En base a la fórmula 2.3-7 y a la figura 2.1-1, se puede encontrar una aproximación inicial del coeficiente de multa: $C \cong 500$, ya que $f_{\max}=400$ $f_{\min}=7$ y $\delta_{\max,\min}=0.8$ (distancia relativa). Entonces se puede esperar que en muchos casos se logre evadir alcanzar un mínimo local asignando el coeficiente de multa $C \geq 500$ y tomando el punto de preferencia no demasiado lejos del punto mínimo global.

Para cada experimento se utilizaron distintos puntos de preferencia, ya que el punto de preferencia refleja información a priori, estos puntos deben ser ubicados en una vecindad del mínimo global ó entre mínimos locales y el global. En los experimentos que se realizaron se tomaron varios puntos de preferencia como punto de preferencia las coordenadas del centro del espacio de parámetros. Hay que decir que sin función de preferencia el método NM no encuentra el mínimo global desde este punto.

Ya que el usuario no sabe donde se ubica el mínimo global se considerarán varios puntos iniciales distribuidos uniformemente en el área de búsqueda. Aquí se utilizaron 10 puntos.

Se realizaron 4 series de experimentos con 3 valores de la función de multa el primero con un valor de $C=100, 200, 500$, el segundo con $C=100, 200$ y 600 , el tercero con $C=10, 50$ y 100 y el cuarto con los mismos valores que el tercero. Los valores finales de C se obtuvieron mediante la aproximación de la constante de Lipschitz usando la fórmula 2.3-7. Los resultados de estos experimentos se muestran en las tablas 2.3-1 a 2.3-4. Las letras L y G significan que el método alcanzó el mínimo local o global respectivamente, los números significan la cantidad de pasos y SE sin éxito alguno.

C = 0					C = 100				
26 L	34 L	37 L	34 L	29 L	9 L	20 L	30 L	19 L	20 L
39 L	43 L	48 G	32 L	45 L	19 L	26 L	20 G	19 L	26 L
C = 200					C = 500				
10 SE	15 G	22 G	23 L	20 L	10 SE	15 G	38 SE	23 SE	28 SE
16 SE	26 L	19 G	25 L	27 L	28 SE	32 SE	26 SE	21 SE	34 SE

Tabla 2.3-1. Convergencia del método NM con la función de criterio No.1
(punto de preferencia = (2,345))

En la Fig. 2.1-1 se muestra el efecto de la influencia de una función de preferencia. Esa figura muestra el relieve de la función de criterio sin función de preferencia. La Fig. 2.4-2 muestra el relieve de la suma de la función de criterio y la función de preferencia con el valor $C=500$.

C = 0					C = 100				
15 L	18 L	31 L	19 G	? L	23 L	23 G	35 L	19 G	16 SE
26 L	32 L	20 G	20 L	31 L	28 L	33 G	19 G	23 L	33 L
C = 200					C = 600				
27 L	22 G	24 G	17 G	22 SE	27 G	24 G	18 G	21 G	25 G
27 G	30 G	20 G	19 L	23 L	20 G	28 G	20 G	29 G	25 SE

Tabla 2.3-2. Convergencia del método NM con la función de criterio No.2
(punto de preferencia = (6,375))

C = 0					C = 10				
18 G	6 L	12 L	14 L	4 L	16 G	6 SE	23 L	18 L	4 SE
5 L	17 L	29 L	17 G	7 L	5 SE	19 L	27 L	17 G	7 L
C = 50					C = 100				
8 G	8 SE	22 L	28 L	16 SE	10 G	30 G	34 G	21 G	12 SE
5 SE	23 L	12 SE	18 G	10 L	29 G	38 G	23 G	12 SE	9 L

Tabla 2.3-3. Convergencia del método NM con la función de criterio No.3
(punto de preferencia = (-60,-20))

C = 0					C = 10				
13 G	18 G	24 L	16 L	43 G	13 G	16 G	26 L	21 L	26 G
19 G	22 G	22 L	17 G	20 L	23 G	27 G	24 SE	17 G	23 L
C = 50					C = 100				
19 G	17 G	19 G	23 L	25 G	25 G	9 SE	19 G	33 SE	27 G
23 G	36 L	20 G	18 G	16 L	27 SE	28 G	19 G	19 G	15 SE

Tabla 2.3-4. Convergencia del método NM con la función de criterio No.4
(punto de preferencia = (5,7))

La tabla 2.3-5 muestra los mejores resultados de éxitos obtenidos con el mejor coeficiente de multa de cada serie de experimentos.

Experimentos	Búsqueda sin preferencia	Búsqueda con preferencia
Experimento 1	1/10	3/10
Experimento 2	2/10	9/10
Experimento 3	2/10	7/10
Experimento 4	6/10	7/10

Tabla 2.3-5 Resultados generalizados de los experimentos

Análisis de resultados:

1. En 3 de 4 casos el método NM alcanzó el mínimo global empezando su búsqueda en un punto aleatorio. Se obtuvo de 3 a 4 veces mas casos de éxito utilizando la función de preferencia que sin ella.
2. En uno de los experimentos no hubo influencia de la función de preferencia para alcanzar el mínimo global.
3. Si se utiliza un coeficiente de multa de valor grande no adecuado al relieve de la función de criterio, el método pierde el mínimo global y alcanza otro extremo generado por la adición de las funciones de criterio y de preferencia.

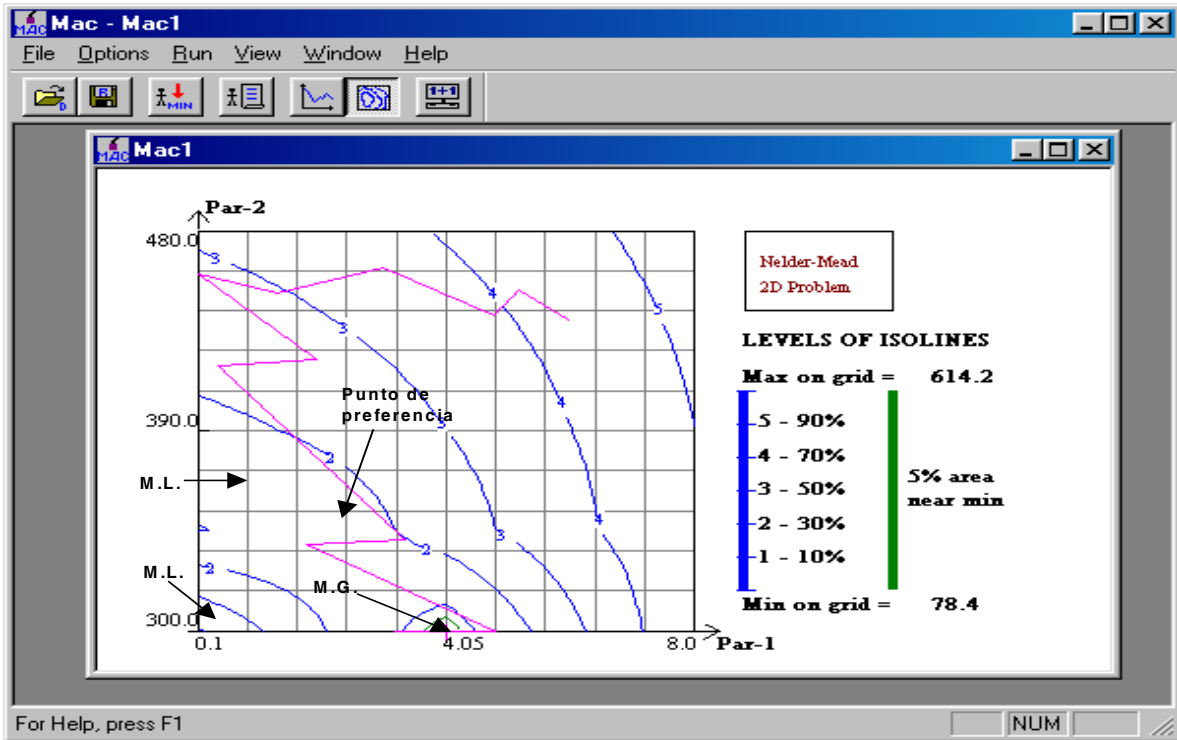


Fig. 2.3-2 Relieve de la suma de la función de criterio No.1 y la de multa.

Nota.

Es necesario enfatizar que en realidad el usuario desconoce la ubicación del mínimo global y que el coeficiente de multa se asigna en base a analogías con otros casos. Como un comentario importante debe decirse que no hay que excederse en el uso de los coeficientes de multa debido a que se puede desviar la búsqueda del método hacia un punto mínimo ficticio que sería igual al punto de preferencia del usuario, el cual no es igual al punto mínimo global.

2.4 Combinación de métodos

1. Estrategia de búsqueda

La idea de combinar métodos no es nueva [Rastrigin, 1981] pero en la mayoría de los trabajos se consideraron los problemas de búsqueda óptima por métodos de programación no lineal que logran alcanzar un mínimo local. Los investigadores se han esforzado en formular unas reglas para el cambio de un método a otro teniendo en cuenta el relieve de la función de criterio. Por ejemplo, considerando la Fig. 1.3-1 es posible seleccionar las siguientes tres etapas típicas [Vasiliev, 1988]:

1. Descenso al valle. Aquí los valores de la función disminuyen rápidamente
2. Recorrido por el valle. Aquí cambia bruscamente la dirección de la búsqueda y continúa el movimiento a lo largo del profundo y estrecho valle donde los valores de la función cambian lentamente
3. Búsqueda en la vecindad del extremo, donde cualquier función suave es casi una función cuadrática

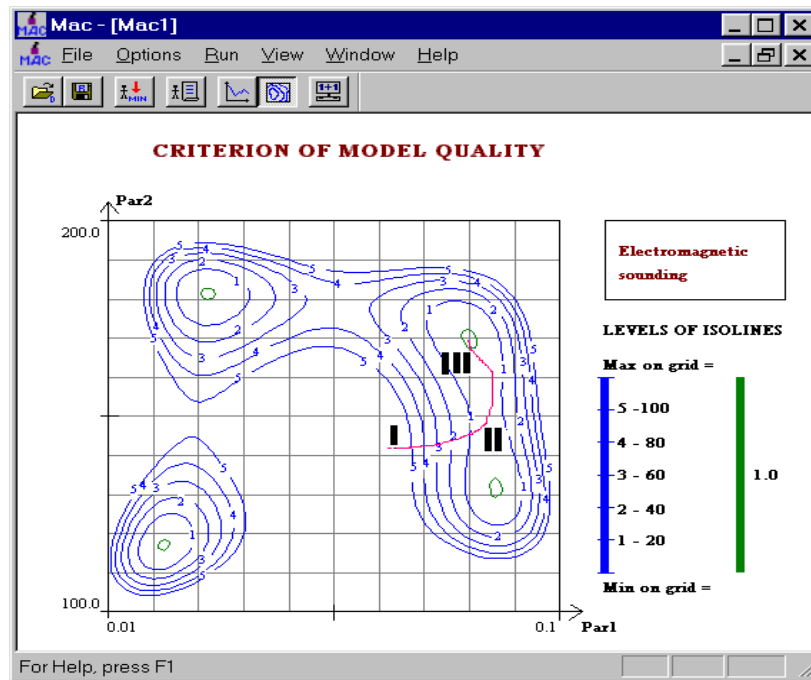


Fig. 2.4-1. Etapas típicas de búsqueda de un óptimo

Como se conocen bien los métodos que trabajan mejor en cada etapa es necesario organizar el cambio de modos, interactivo o automático, siendo importante resolver el problema de la búsqueda de un mínimo global entre varios mínimos locales. Para resolver este problema se utiliza:

- algún método de programación estocástica.
- algún método de programación no lineal que realice la búsqueda desde varios puntos iniciales.

Ambos enfoques tienen sus desventajas:

- Los métodos de programación estocástica buscan un mínimo global. Pero el resultado de la búsqueda tiene carácter estadístico, es decir, la estimación del mínimo global se da con una probabilidad. Alta probabilidad requiere muchos puntos de prueba.
- Los métodos no lineales buscan exactamente un mínimo de acuerdo con un error dado. Pero el mínimo alcanzado es un mínimo local y es necesario investigar toda el área de búsqueda para asegurarse que el mínimo alcanzado es realmente un mínimo global. Todo esto también requiere de muchos puntos de prueba.

Se propone combinar métodos estocásticos y no lineales para disminuir el número de puntos de prueba. Concretamente se propone la estrategia siguiente:

- I. Iniciar con el método no dirigido de búsqueda aleatoria, el parámetro de confianza se toma entre 0.95 y 0.99, la vecindad de confianza (la precisión relativa) se toma aproximadamente 20%-30% con respecto al área total. Bajo estos valores el método no usa muchos puntos aleatorios.
- II. Después se toma una sub-área sobre un punto mínimo encontrado en el paso anterior y se trabaja ahora con el método de Nelder-Mead. Los tamaños de esta sub-área deben ser iguales a los mencionados, 20%-30% del área que se usa en el paso I. La precisión relativa en este paso debe ser bastante alta, por ejemplo 10^{-3} , 10^{-4} .
- III. Recibiendo el resultado de la búsqueda anterior (paso II, el usuario decide terminar o continuar la búsqueda. Si se continúa, se repiten los pasos I y II, pero ahora sobre otra sub-área.

La Fig. 2.4-2 muestra la búsqueda en la etapa I. La Fig. 2.4-3 demuestra la búsqueda en la etapa II. Observando la dinámica de búsqueda en la etapa II Fig. 2.4-4, el usuario toma la decisión de que el método alcanzó un mínimo global.

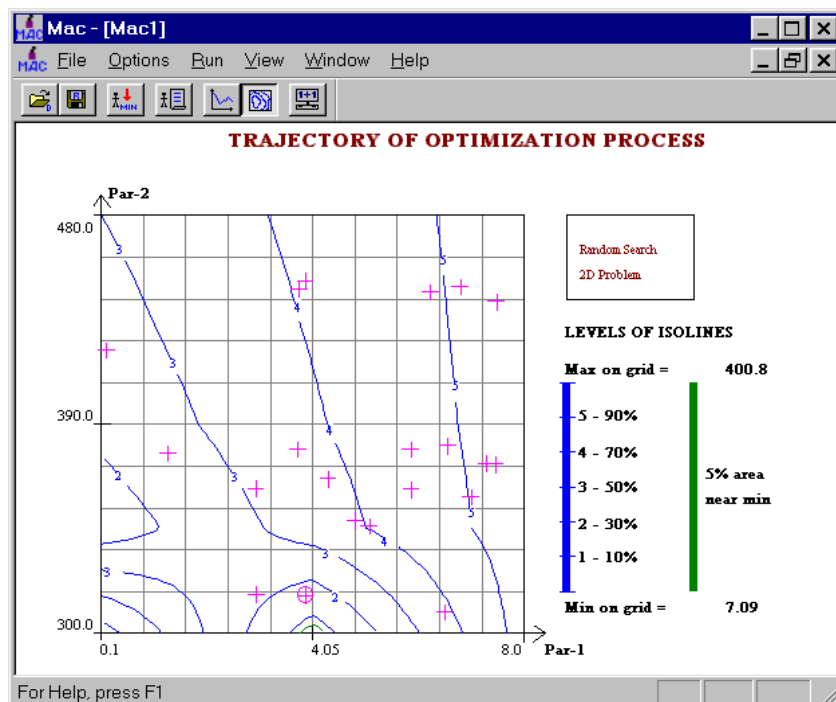


Figura 2.4-2. Trayectoria de búsqueda en la etapa I ($\epsilon=0.2$, $p=0.99$)

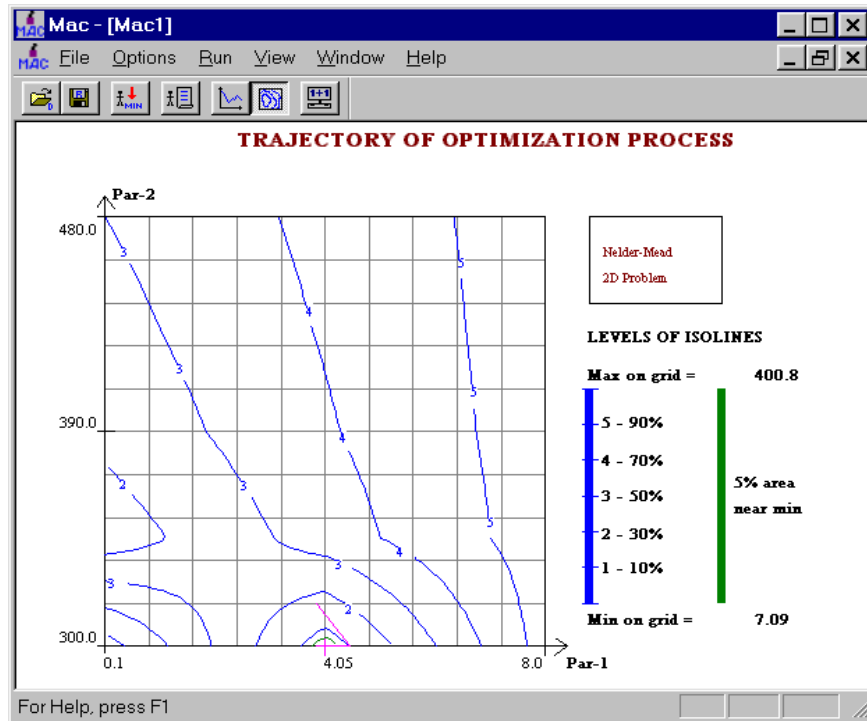


Fig. 2.4-3. Trayectoria de búsqueda en la etapa II ($\epsilon=0.001$)

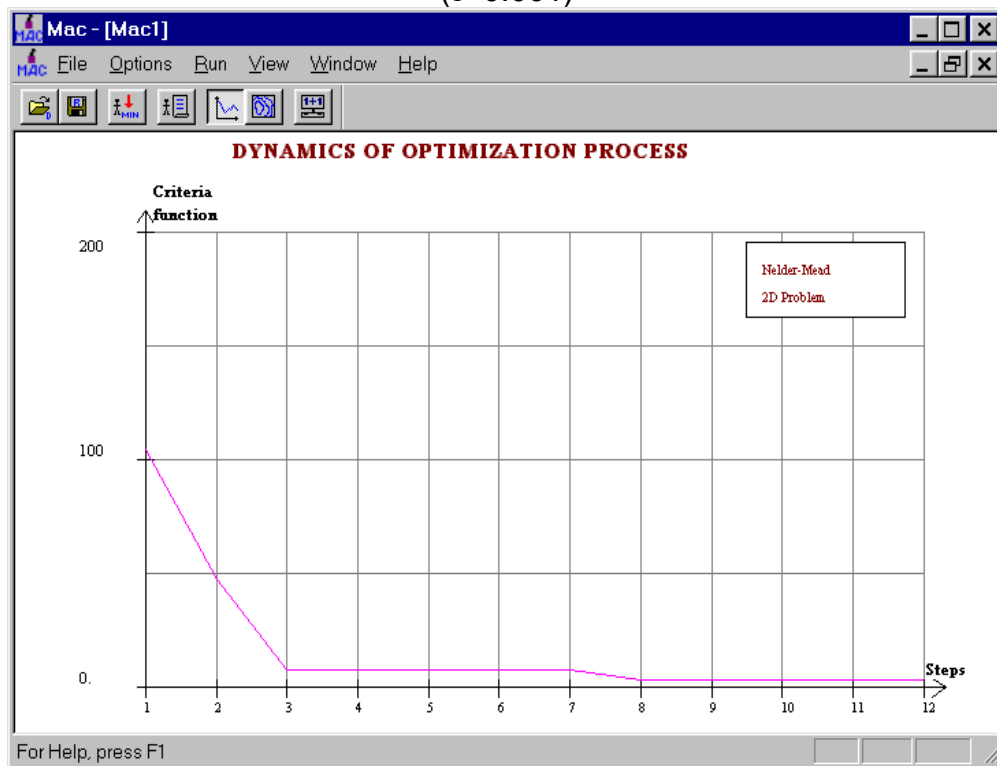


Fig. 2.4-4. Dinámica de búsqueda de la etapa II

2. Experimentos numéricos

Estos experimentos deben probar que la aplicación de la combinación del método no dirigido de búsqueda aleatoria con el método NM provee gran ahorro de puntos de búsqueda en comparación con la aplicación de cada método separadamente. En los experimentos se usó el ejemplo descrito en el párrafo 2.1.

Se realizaron 3 series de los experimentos:

- En la primera serie se usó el método NM desde 10 puntos iniciales que fueron distribuidos uniformemente en el área de búsqueda. El error establecido fue 0.001
- En la segunda serie se usó 10 veces el método no dirigido de búsqueda aleatoria. El error definido fue 0.001 y el nivel de confianza de 0.95
- En la tercera serie se usó 10 veces la combinación de los métodos. En la primera etapa el error fue de 0.2 y el nivel de confianza de 0.99. En la segunda etapa el error fue de 0.001.

Los resultados de los métodos utilizando las cuatro funciones de criterio ambientales se muestran en las tablas 2.4-1 a la 2.4-4. Las letras L y G significan que el método alcanzó el mínimo local o global respectivamente. Los números significan la cantidad de pasos.

Método NM $\epsilon=0.001$					Método de búsqueda aleatoria $\epsilon=0.001$ $p=0.95$ $n=3000$					Combinación de métodos				
16 L	19 L	20 L	21 L	18 L	G	G	G	G	G	21+11=32 G	21+14=35 L	21+13=34 G	21+14=35 L	21+17=38 G
19 L	21 L	20 G	18 L	24 L	G	L	L	G	L	21+14=35 G	21+11=32 G	21+17=38 L	21+15=36 G	21+12=33 G

Tabla 2.4-1 Convergencia bajo estrategias diferentes de búsqueda usando la función de criterio No.1

Método NM $\epsilon=0.001$					Método de búsqueda aleatoria $\epsilon=0.001$ $p=0.95$ $n=3000$					Combinación de métodos				
15 L	18 L	31 L	19 G	17 L	G	L	L	G	G	21+15=36 G	21+22=43 L	21+17=38 L	21+14=35 G	21+16=37 G
26 L	32 L	20 G	20 L	31 L	G	L	G	G	L	21+22=43 L	21+21=42 L	21+19=40 L	21+18=39 L	21+14=37 G

Tabla 2.4-2 Convergencia bajo estrategias diferentes de búsqueda usando la función de criterio No.2

Método NM $\epsilon=0.001$					Método de búsqueda aleatoria $\epsilon=0.001$ $p=0.95$ $n=3000$					Combinación de métodos				
18 G	6 L	12 L	14 L	4 L	G	G	G	L	G	21+16=37 G	21+18=39 L	21+9=30 L	21+20=41 L	21+19=40 L
5 L	17 L	29 L	17 G	7 L	G	L	G	G	G	21+17=38 G	21+13=34 G	21+18=39 G	21+20=41 L	21+14=35 G

Tabla 2.4-3 Convergencia bajo estrategias diferentes de búsqueda usando la función de criterio No.3

Método NM $\epsilon=0.001$					Método de búsqueda aleatoria $\epsilon=0.001$ $p=0.95$ $n=3000$					Combinación de métodos				
13 G	18 G	24 L	16 L	43 G	SE	G	G	G	G	21+17=38 L	21+15=36 G	21+15=36 G	21+12=33 G	21+12=33 G
19 G	22 G	22 L	17 G	20 L	G	G	SE	G	SE	21+15=36 L	21+13=34 G	21+18=39 L	21+15=36 L	21+21=42 G

Tabla 2.4-4 Convergencia bajo estrategias diferentes de búsqueda usando la función de criterio No.4

	Nelder - Mead	Búsqueda aleatoria	Combinación de métodos
Experimento 1	20 (1)	3000 (7)	35 (7)
Experimento 2	23 (2)	3000 (6)	39 (4)
Experimento 3	13 (2)	3000 (8)	37 (5)
Experimento 4	24 (6)	3000 (7)	36 (6)

Tabla 2.4-5 Resultados generalizados de los experimentos.

La tabla 2.4-5 muestra los promedios de números de pasos y número de éxitos para cada uno de los experimentos.

Análisis de resultados:

En este análisis se tienen en cuenta dos circunstancias: números de pasos y número de éxitos.

1. En todos los experimentos el número de pasos en la combinación de métodos es mayor 2 veces más en promedio, pero estos números son casi 100 veces menor que el número de pasos de la búsqueda aleatoria.
2. En 3 experimentos de 4 la combinación de métodos alcanza el éxito en promedio 3 veces más que en NM y un 30% menos que la búsqueda aleatoria.
3. Solamente en un experimento los tres métodos logran casi el mismo número de éxitos y la combinación de métodos solamente en un 50% es mayor que el número de pasos de NM.

Todo esto prueba la ventaja de la estrategia propuesta de búsqueda.

3. Propuesta de trabajo a futuro

En el caso de un gran número de parámetros a ser determinados, la función de criterio tendrá muchos óptimos locales y la aplicación ordinaria de los métodos de optimización no tendrá éxito. Realmente cuando un método comienza a trabajar, inmediatamente cae en el hoyo más cercano, esto es, en el óptimo local más cercano.

En vez de buscar un óptimo en el espacio de todos los parámetros se propone el procedimiento de búsqueda de paso a paso en los espacios de menor dimensión. Se espera que en espacios de menores dimensiones, el número de óptimos sea menor en comparación con el número de óptimos en el espacio total.

Para esto el espacio total de parámetros se divide en varios subespacios tomando en cuenta la posible conexión mutua entre los parámetros. La dimensión de los subespacios no debe ser mayor de 5 a 7. Estas cifras se obtuvieron de las experiencias de geofísica (esta división de los subespacios es un procedimiento aparte). Entonces cualquier método de optimización es asignado para trabajar en tal subespacio (esta selección es también otra tarea aparte). Estos métodos empiezan a competir tratando de encontrar un extremo simultáneamente.

Una interacción entre los métodos mencionados puede realizarse desde el enfoque orientado a objetos, donde cada objeto es responsable de buscar en un cierto subespacio. El contenido del algoritmo es la competencia entre estos objetos por el derecho a buscar un extremo. La tecnología de interacción de objetos esta bien descrita en [Shlaer, 1992]. Este enfoque es un propósito que debe ser probado, por lo que puede considerarse como un trabajo a futuro.

2.5 Resumen

Este capítulo considera los diversos aspectos de la organización de la búsqueda de extremos de funciones de criterio, que se encuentran en el problema de identificación de fuentes de contaminación.

Ante todo basándose en los requisitos generales de las funciones de criterio, se construyeron las funciones concretas para los ejemplos usados en este trabajo.

Fueron encontrados los parámetros óptimos del método NM para las funciones ambientales usadas en este trabajo.

Se investigaron las posibilidades de usar la función de preferencia y la combinación de ciertos métodos de optimización para buscar un mínimo global en la presencia de mínimos locales. Se realizaron experimentos numéricos que aprueban las ventajas de estas técnicas propuestas.

CAPITULO III.

Diseño y programación del sistema

Este capítulo describe la estructura y las funciones del sistema MAC, de acuerdo a la programación orientada a objetos. MAC trabaja solamente con funciones de una y dos variables, esto está relacionado con la simplicidad de su representación gráfica, sin embargo, los métodos multidimensionales de optimización incluidos en la librería, permiten trabajar con funciones de un número arbitrario de variables.

3.1 Consideraciones generales

1. Desarrollo del sistema

El sistema fue desarrollado y programado dentro del ambiente de Microsoft Visual C++ 6.0. Se usó la programación orientados a objetos solamente para la interfaz gráfica y para la organización del programa. Se utilizaron las siguientes clases:

- Herederos de clases generales CMacDoc, CMacView, CMacFrame etc. Para la organización de las interacciones entre todos los elementos del programa incluyendo los procesos de optimización, visualización de resultados y presentación de ventanas.
- Clases de interacción con el usuario CFileDialog, COptimDlg, CMethodDlg, CPenaltyDlg, CProtocolDlg para la recepción de los nombres de archivos, definición de condiciones de búsqueda, asignación de parámetros para los métodos de optimización, asignación de parámetros para la función de preferencia y la elección del contenido del registro de resultados.

Clases generales y clases de interfaz se construyeron usando la librería MFC (Microsoft Foundation Class) que es parte del sistema de programación Visual C++ [Kruglinski, 1999].

2. Ayuda para el usuario

Para que un programa sea una herramienta útil para el usuario, resulta conveniente que el programa mismo cuente con ejemplos de aplicaciones ya resueltas y que el usuario pueda manejarlas fácilmente, de otro modo, podría rechazar su uso. En el sistema MAC se incluyeron algunas funciones de criterio estándar con puntos extremos conocidos. Estos ejemplos preparados de antemano pueden ayudar al usuario para simplificar un proceso de construcción

de una función de criterio y para evaluar la convergencia de un método de búsqueda.

El programa usa las siguientes funciones estándar:

- En caso de una dimensión, el polinomio de orden cuatro $F(x)=(x-1)^2 (x+1)^2$
- En caso de dos dimensiones, la Banana de Rosenbroke $F(x)=100(x_2-x_1^2) + (1-x_1)^2$

La primera función tiene dos puntos mínimos en (-1) y (+1). Este tipo de función es conveniente para el análisis de convergencia de búsquedas de funciones multiextremos. La segunda función tiene un mínimo en el punto (1,1). Esta función es usada muy extensamente en la literatura sobre programación no lineal para comparar varios métodos de optimización [Himmelblau, 1992].

3.2 Componentes del sistema

En este párrafo se da la descripción formal del sistema MAC.

1. Diagrama de las clases y módulos

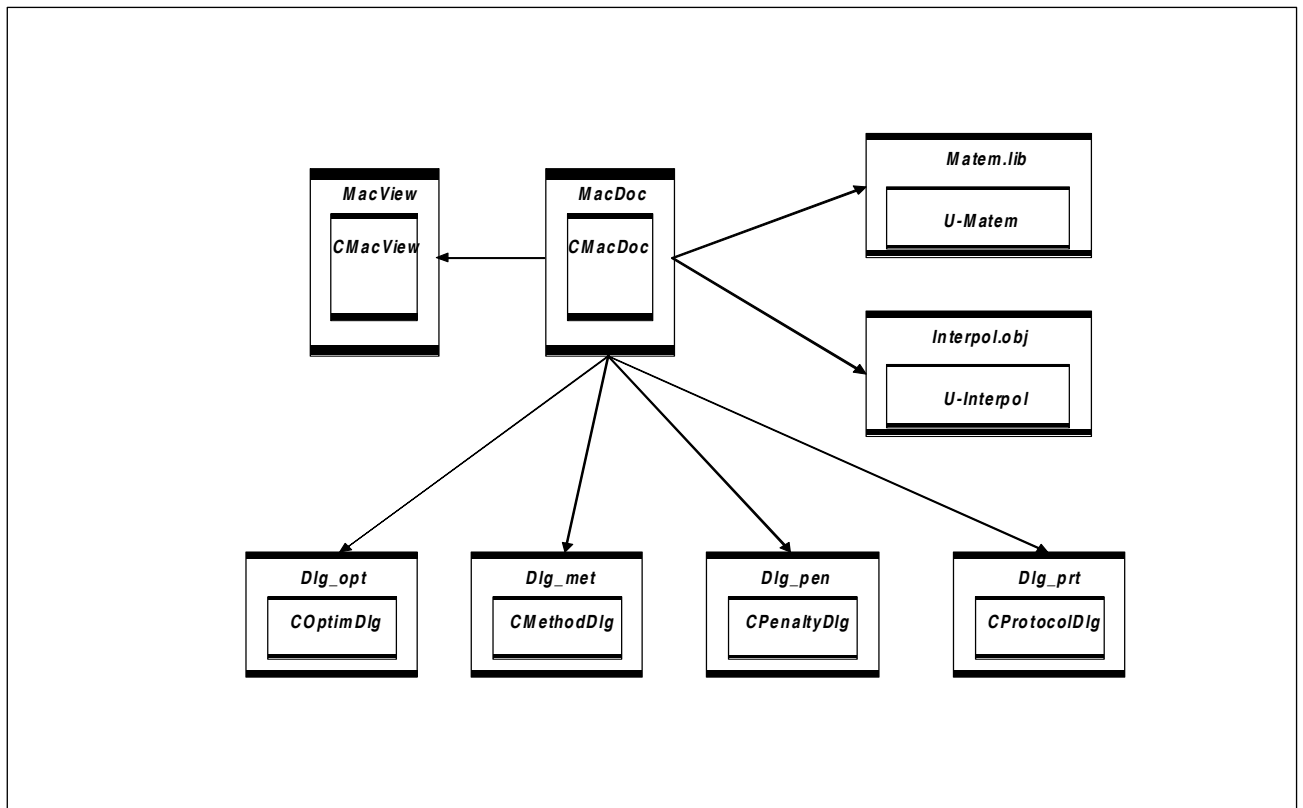


Fig. 3.2-1 Diagrama de clases y módulos del sistema MAC

En este sistema cada clase está incluida en un módulo separado de la siguiente forma:

- Clase CMacDoc (en el Módulo MacDoc) es la clase principal que incluye objetos de todas las demás clases y hace referencia a los elementos de las bibliotecas de optimización e interpolación. Dentro de esta clase se organiza el proceso de búsqueda y la interacción con el usuario mediante las clases de interfaz. La clase CMacDoc contiene también funciones de I/O.
- Clase CMacView (en el módulo MacView) contiene las rutinas de visualización de los procesos de búsqueda. Las figuras 2.5-2, 2.5-3 y 2.5-4 son ejemplos del trabajo de esta clase.
- Clase COptimDlg (en el módulo Dlg_opt) es responsable de la interacción con el usuario mediante una ventana de diálogo como se muestra en la figura 3.2-6 en la cual se establecen las condiciones de la búsqueda.
- Clase CMethodDlg (en el módulo Dlg_met) es responsable de la interacción con el usuario mediante una ventana de diálogo como se muestra en la figura 3.2-7 en la cual se establecen los parámetros de los métodos.
- Clase CPenaltyDlg (en el módulo Dlg_pen) es responsable de la interacción con el usuario mediante una ventana de diálogo como se muestra en la figura 3.2-8 en la cual se establecen la función de multa.
- Clase CProtocolDlg (en el módulo Dlg_prt) es responsable de la interacción con el usuario mediante una ventana de diálogo como se muestra en la figura 3.2-9 donde el usuario puede seleccionar el tipo de protocolo.
- Utilería U-Matem (en el módulo Matem.lib) contiene los métodos de optimización en forma de procedimientos de C++.
- Utilería U-Interpol (en el módulo Interpol.obj) contiene los procedimientos de interpolación, y aproximación de una y dos variables.

El diagrama anterior contiene solamente las clases que se modificaron.

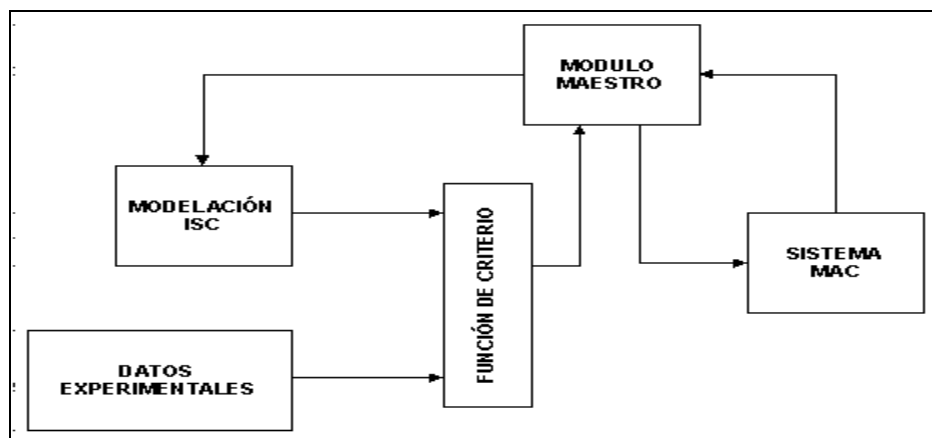


Fig. 3.2-2 Diagrama de interacción del sistema MAC en un problema real usando un módulo maestro

El sistema MAC podría interactuar con un GIS (Sistema de información geográfica) solamente proporcionando información de las coordenadas geográficas de las fuentes contaminantes para ser ubicadas en un mapa y la ubicación de la nube contaminante, la cual podría también localizarse sobre un mapa geográfico.

2. Diagrama de los procesos

Este diagrama refleja el funcionamiento del sistema como se describe en el párrafo 3.3.

Al comienzo de la búsqueda, el usuario debe seleccionar una función ejemplo u obtener una función de criterio real, enseguida por medio de las ventanas de diálogo se definen las condiciones de búsqueda, se asignan los parámetros de los métodos o se define la función de multa. Después el usuario debe seleccionar el modo de búsqueda de optimización de una etapa o varias etapas.

El usuario puede observar la trayectoria de búsqueda o la dinámica de búsqueda cambiando el modo de visualización en cualquier momento. Al final de la búsqueda se puede definir el tipo de protocolo si se desea y salvar los resultados en un archivo de texto.

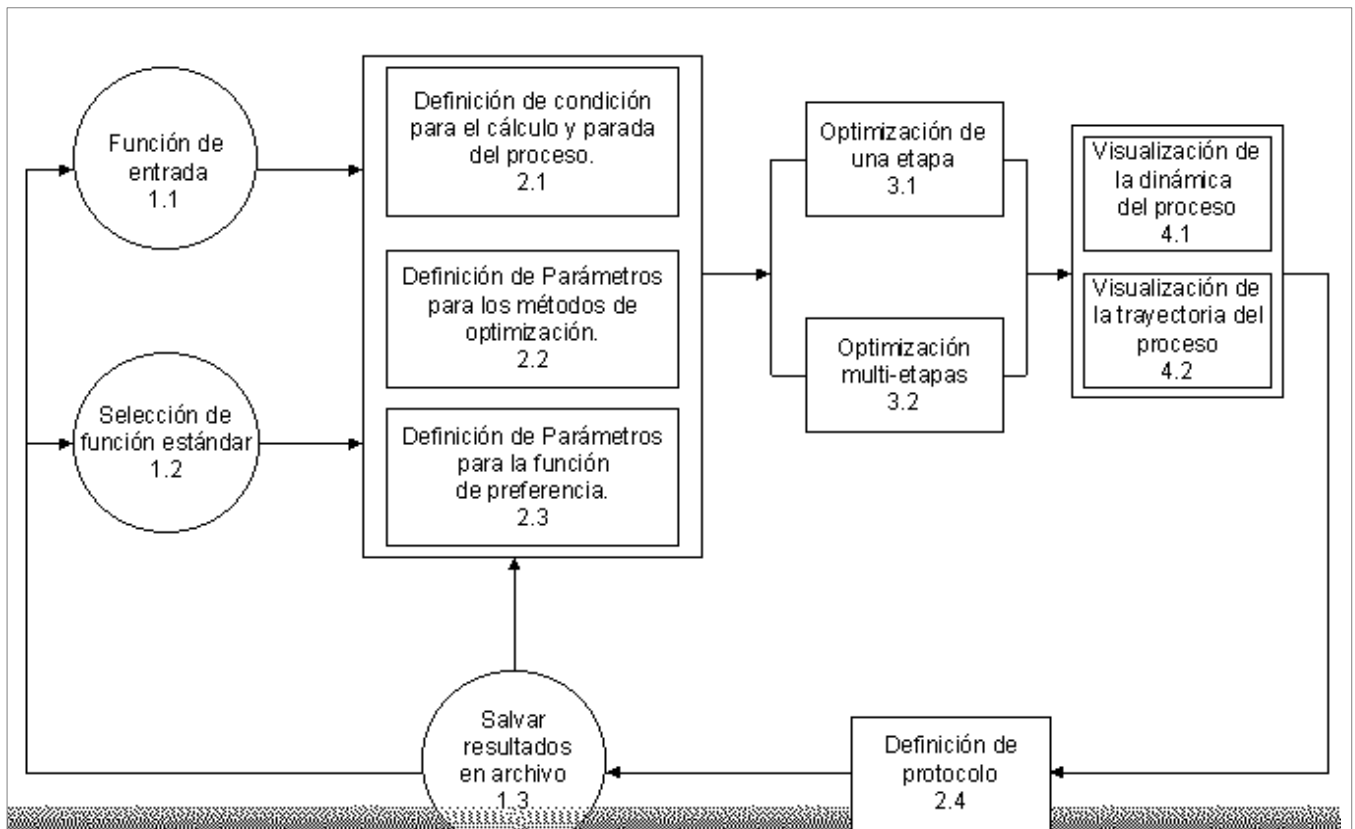


Fig. 3.2-3 Diagrama de procesos

En un sistema real el diagrama anterior tendría dos modificaciones, la primera sería una conexión de la “Función de entrada” con un módulo maestro, el cual proporcionaría al sistema la función de criterio a evaluar y la segunda sería la omisión del bloque “Visualización de la trayectoria del proceso”, ya que el usuario desconoce la forma del relieve de la función de criterio por lo que no podría ser posible visualizar la trayectoria de búsqueda sobre las isolíneas de la función.

3. Comentarios sobre el sistema

La primera versión del sistema se finalizó en el año 2000 y se aprobó en la Academia Estatal de Geología de Moscú y en el C.I.C. Actualmente se continúa trabajando en el sistema con el objeto de mejorar la interfaz gráfica e incluir nuevos métodos de optimización.

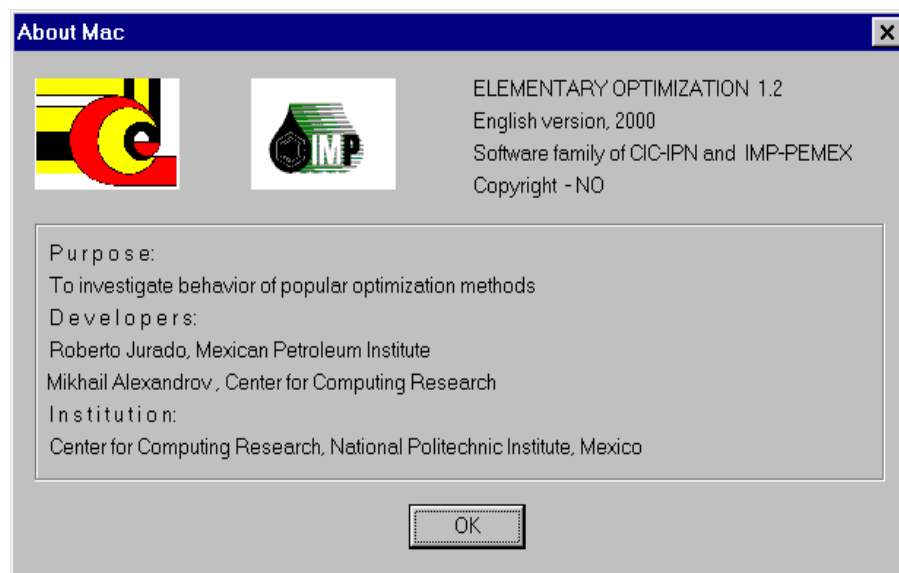


Fig. 3.2-4 Información de MAC

3.3 Funcionalidad del sistema

Resulta lógico comenzar la descripción del sistema con una presentación de sus componentes.

1. Datos fuentes de entrada

La entrada del sistema MAC es una matriz de valores de dos parámetros que representan una función de criterio dada sobre una malla como se describió en la sección 3 del párrafo 2.1. Las matrices resultantes al calcular las funciones de criterio se muestran en el Anexo 3.

MAC permite empezar a trabajar sin ninguna fuente de datos. En este caso el usuario puede tomar algunas funciones estándar que pueden ser consideradas como funciones de criterio. Como se mencionó anteriormente, MAC ofrece el polinomio de grado cuarto $F(x)=(x-1)^2 (x+1)^2$ como una función de una variable y la Banana de Rosenbroke $F(x)=100(x_2-x_1^2) + (1-x_1)^2$ como una función de dos variables.

2. Usando el sistema

Un usuario puede interactuar con el sistema MAC por medio del menú y de las cajas de diálogo similares a las de las aplicaciones de Windows. Algunos de los botones del menú que son usados frecuentemente, están puestos sobre la barra de herramientas como se muestra en la Fig. 3.3-1.

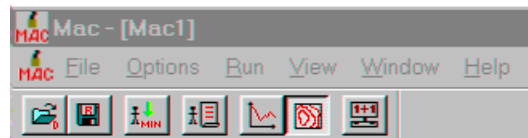


Fig. 3.3-1 Menú principal del sistema y su barra de herramientas.

El menú principal consta de los siguientes encabezados:

- **File**
Selección de la función de criterio y del protocolo de salida.
- **Options**
Definición de las condiciones para los cálculos, ajustes de métodos, función de preferencia y protocolo.
- **Run**
Ejecuta el proceso una o varias veces.
- **View**
Elección de la vista de la dinámica de la búsqueda o gráfico de la función criterio.

File – entrada/salida (Fig. 3.3-2)

Teniendo presionado este botón, el usuario obtiene el siguiente menú.

- **Open Data**
Elección del archivo que contiene la función de criterio.
- **Open Standard 1D-Example**
Elección del polinomio de cuarto orden como función de criterio.
- **Open Standard 2D-Example**
Elección de la Banana de Rosenbroke como función de criterio.
- **Save Result**
Escribe un protocolo en un archivo definido por el usuario anteriormente.

- **Save Result As**
Escribe un protocolo en un archivo con un nombre nuevo.

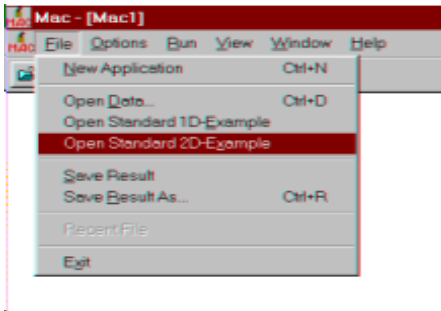


Fig. 3.3-2 Submenú de entrada/salida

Options – varias opciones (Fig. 3.3-3)

Habiendo presionado este botón, el usuario tiene el siguiente submenú:

- **Parameters of Process and Final Results**
La ventana de diálogo se muestra en la Fig. 3.3-6. Se elige un método de optimización, se asignan las restricciones de los parámetros, se define la exactitud y el número de iteraciones. Adicionalmente, se define el nivel de confianza en las vecindades del mínimo para el método de búsqueda aleatoria. Después de terminado el cálculo, la exactitud del resultado y el número completo de iteraciones son visualizados en la parte inferior de la ventana de diálogo.
- **Parameters of Optimization Methods**
La ventana de diálogo se muestra en la Fig. 3.3-7. En ella se asignan los parámetros del método de búsqueda de paso a paso o de Nelder-Mead. Los otros dos métodos no tienen parámetros de ajuste.
- **Parameters of Preference Function**
La ventana de diálogo se muestra en la Fig. 3.3-8. Aquí el usuario define los parámetros más adecuados del modelo y el coeficiente de penalidad. El sistema MAC usa la función de preferencia en la forma de $C|...|$, donde C - coeficiente de multa, $|...|$ - valor absoluto de la diferencia entre la combinación del parámetro actual y los parámetros preferenciales es decir los puntos de intento. Este valor es normalizado sobre la longitud del intervalo para cada coordenada. MAC usa una norma cuadrática.
- **Contents of protocol**
Usando la ventana de diálogo presentada en la Fig. 3.3-9 el usuario selecciona la forma breve o completa del protocolo. En el último caso, el archivo apropiado contiene no solo todas las condiciones de la búsqueda y los resultados finales, sino también toda la trayectoria de búsqueda.

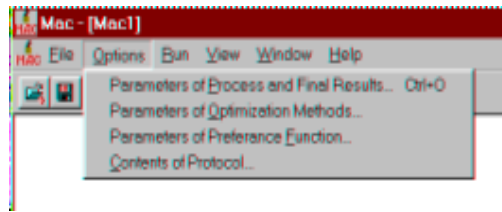


Fig. 3.3-3 Submenú de opciones

Run – ejecutando (Fig. 3.3-4)

Habiendo presionado este botón se llama al submenú y los cálculos se inician.

- ***One-time optimization***

La búsqueda del mínimo es llevada a cabo usando un método de optimización dado. El número de iteraciones no puede exceder al número definido en la ventana de diálogo presentada en la Fig. 3.3-6. En todos los casos, un ciclo de iteraciones de la n -ésima búsqueda no puede contener más de 100 iteraciones.

- ***Multi-time optimization***

En este caso, si después del número máximo de iteraciones no se alcanza la exactitud deseada, entonces el ciclo de búsqueda se repite de nuevo. El número de ciclos de búsqueda no puede exceder de 100.

Independientemente del tipo de búsqueda que se esté ejecutando (one time o multi time), MAC memoriza la información de las últimas 100 iteraciones. Sólo estas iteraciones son visualizadas en pantalla y sólo este número de iteraciones son incluidas en el protocolo completo.



Fig. 3.3-4 Submenú de la ejecución del proceso.

View – visualización del cálculo (Fig. 3.3-5)

Usando este botón el usuario puede definir el contenido de la ventana.

- ***Dinámica de Búsqueda***

Esta representación es la misma tanto para una función de una variable como para la de dos. Un diagrama de la función de criterio en cada punto de la búsqueda es mostrado en la pantalla (Fig. 2.5-4). En caso de que el valor de la función de criterio esté fuera del intervalo 0.001-1000, ésta será representada en escala logarítmica.

- **Trayectoria de búsqueda.**

La representación de una trayectoria de búsqueda es diferente para una función de una variable que para una de dos variables. En el caso de una función de una variable se representan tanto la curva de la función y la línea que representa la búsqueda. En el caso de una función de dos variables, se presentan tanto la función misma por las isólineas y la trayectoria de la búsqueda (Fig. 2.5-3). Para el método de búsqueda aleatoria es imposible hablar de alguna trayectoria, en este caso, todos los puntos calculados son visualizados como cruces y el mejor punto aparece marcado por una cruz dentro de un círculo (Fig. 2.5-2).

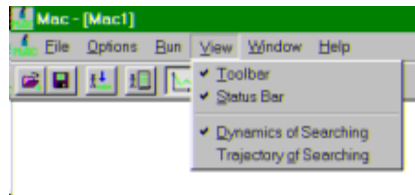


Fig. 3.3-5 Submenú para la selección de la vista.

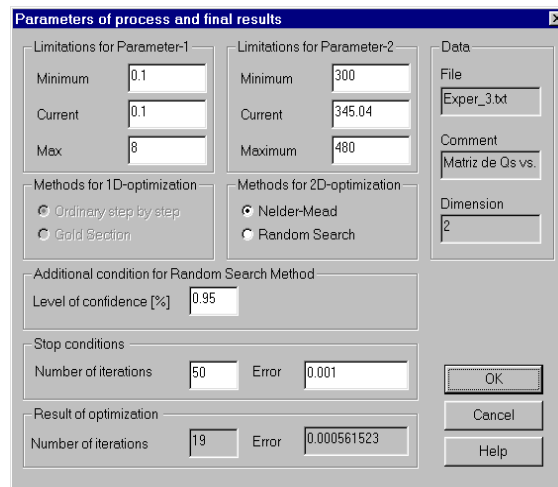


Fig. 3.3-6 Definición de las condiciones de búsqueda

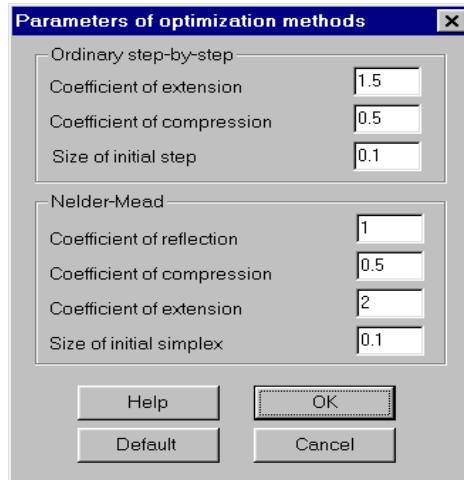


Fig. 3.3-7 Ajuste de los métodos de optimización.

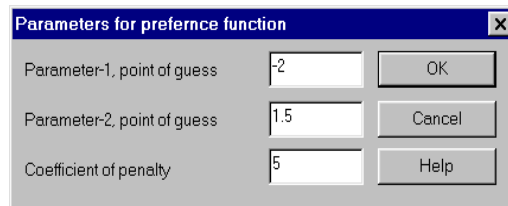


Fig. 3.3-8 Definición de una función de preferencia



Fig. 3.3-9 Selección del tipo de registro.

3. Registro de resultados

El usuario puede guardar los resultados del cálculo en un archivo de texto, habiendo seleccionado previamente la forma del registro mediante el menú (Fig. 3.3-9). Las formas breve y completa del registro de búsqueda se presentan en Anexo 4.

3.4 Resumen

En este capítulo se presenta la descripción del funcionamiento del sistema MAC y de su interfaz gráfica. El sistema cuenta con ejemplos para mostrarle al usuario los modos de trabajo como una forma de facilitar el manejo del sistema.

Se presentan las descripciones de las clases y de los módulos que constituyen al sistema y su interrelación dentro de este. Esta descripción permite la modificación del sistema para nuevas aplicaciones futuras.

En el apéndice 5 se listan los códigos fuentes del módulo principal y de la biblioteca de optimización.

CONCLUSIONES

C.1 Novedad científica del trabajo

1. Se encontraron los valores más adecuados de los parámetros del método Nelder-Mead para algunas funciones de criterio ambientales. Estos valores son diferentes de los típicos que se manejan en la literatura clásica.
2. Se comprobó la propuesta de utilizar información a priori acerca de la localización del mínimo global de una función de criterio. Se da una recomendación para la asignación del coeficiente de multa para la función de preferencia. Este enfoque tiene ventajas sobre la aplicación de los métodos no lineales puros siempre y cuando la selección del punto de preferencia esté dentro de una vecindad alrededor del mínimo global.
3. Se comprobó una técnica propuesta para la localización del mínimo global de una función de criterio utilizando una combinación de métodos de búsqueda aleatoria no dirigida y el método de Nelder-Mead. Esta técnica es más eficiente comparada con la aplicación del método NM y búsqueda aleatoria usados separadamente.

C.2 Importancia practica

1. Se desarrolló una biblioteca abierta de algunos métodos de optimización fáciles de incluir en cualquier aplicación de un usuario, con la flexibilidad de poder ajustar los parámetros.
2. Se desarrolló un sistema de optimización que podría ser útil para resolver problemas inversos ambientales. El sistema trabaja con los resultados numéricos de la modelación realizada con el modelo de dispersión gaussiana ISCST3.
3. El sistema es de fácil uso para la enseñanza de métodos de optimización en el campo de la programación matemática. Actualmente el sistema MAC se utiliza en la Facultad de Geofísica de la Academia Estatal de Geología de Moscú.

C.3 Trabajo a futuro

- Desarrollar estrategias de búsqueda del mínimo global en funciones de criterio en el marco del problema inverso ambiental, en un espacio multidimensional muy grande ($n > 10$). Se propuso un enfoque nuevo para resolver este problema basado en el cambio secuencial de sub-espacios de búsqueda.
- Acoplar este sistema a otros sistemas abiertos para la modelación de procesos ambientales, mediante el desarrollo de un módulo maestro que administre el flujo de datos entre los sistemas.
- Acoplar el módulo de localización de las coordenadas de las fuentes contaminantes a un GIS de forma que se puedan visualizar sobre un mapa.

Bibliografía

1. Alexandrov, M. (1982): *Interpretation of geological-geophysical data by means of dialog graphical system on mini-computers*. Industrial Journ. "Mathematical methods of research in geology". N_2. Moscow, pp. 1-12 (rus.)
2. Alexandrov, M. (1987): *Guide to the course "Geophysical data interpretation" (part 2)*. Publ. House of Moscow State Geological Academy, Moscow (rus.)
3. Alexandrov, M. and Jurado, R.(2000): *Determinación de fuentes contaminantes locales y de área mediante técnicas de modelación matemática*, Reporte técnico final, CIC-IPN.
4. Alifanov, O. (1994): *Inverse Heat Transfer Problems*, Springer Verlag
5. Alifanov, O. et al. (1995): *Extreme Methods for Solving Ill-Posed Problems with Applications to Inverse Heat Transfer Problems*. Begell House Inc, New York
6. Batichev, D. (1982): *Optimization methods*. Publ. House of GGU, Gorkyi (rus.)
7. Booch, G. (1991): *Object-oriented design with applications*. The Benjamin/Cummings Publ. Comp.
8. Bulakh, E. (1973): *Automatic sistem for gravitational anomaly interpretation (method of minimization)*. Publ. House "Scientific idea", Kiev, Ukraina.
9. Collins, C. and Scott, S. (1993): *Air Pollution in the valley of Mexico*. pp.1-15
10. Cramer, H. (1946): *Mathematical methods of statistics*. Cambridge.
11. Fiacco, A., McCormick, G. (1968): *Nonlinear programming*. Wiley, N.Y.
12. Heinz, W., Hanke, M., Neubauer, A. (1996): *Regularization of Inverse Problems*. Kluwer, Dordrecht
13. Hensel, E. (1991): *Inverse Theory and Applications for Engineers*. Prentice-Hall
14. Himmelblau, D. (1972, 1992): *Applied Nonlinear Programming*. McGraw-Hill, NY-London-Toronto.
15. Jerald, L. and Schoor, J. L. (1996): *Environmental Modeling*. Publ. House "John Wiley & Sons Inc.", New-York.
16. Jurado, R. and Alexandrov, M. (2000): *Software for the search of simple sources of air pollution using some methods of optimization*. In: "Acta Academia", Publ. House of Intern. Inform. Academy under UN (branch of Moldova), NY-Chisinau etc., annual issue 2000, pp. 381-391.
17. Kruglinski (1999): *Programación avanzada en Visual C++*. Prentice Hall
18. Kurpysz, K. and Nowak, A. (1995): *Inverse Thermal Problems*. Computational Mechanics Publishers, Boston
19. Moussiopoulos N. et al. (1996): *Ambient air quality, pollutant dispersion and transport models*. Publ. House "European Topic Centre on Air Quality", Kopenhagen.
20. Nemirovski, A., Yudinn, D. (1989): *Complexity and efficiency of optimization algorithms*. Publ. House "Nauka", Moscow (rus., transl. to engl.)
21. Rastrigin, L. (1975): *Methods of random search*. Publ. House 'FML' , Moscow (rus.).
22. Rastrigin, L., Erenshtein, P. (1981): *Methods of collective recognition*. Moscow, Publ. House 'Energoizdat' (rus.).
23. Shlaer, S., Mellor S. (1992): *Modeling the World in States*. Prentice-Hall.

24. Smith, R. et al [1972]: *Computer graphics in geophysics*. Intern. Journ. "Geophysics", v. 37, N_5, pp. 825-838
25. Technical manual (1995a): *User's guide for the industrial source complex (ISC3) dispersion models*. Vol. 1: Description of model's algorithms, Publ. House "U.S. Environmental Protection Agency", North California.
26. Technical manual (1995b): *User's guide for the industrial source complex (ISC3) dispersion models*. Vol. 2: User instructions, Publ. House "U.S. Environmental Protection Agency", , North California.
27. Trujillo, D. and Busby, H. (1997): *Practical Inverse Analysis in Engineering*, CRC Press
28. Tumarkin, G., et.al. (1989): *Imitative modeling for the interpretation of geological-geophysical data*. In: "History of geology, geological education and mathematical geology". Publ. House "Nauka", Moscow, pp. 189-194.
29. Vasiliev, F. (1988): *Numerical methods for solution of extreme problems*, Moscow, Publ. House "Nauka" (rus.)
30. World Health Organisation and United Nations Environment Programme. (1992): *Urban Air Pollution in Megacities of the World*. Blackwell.
31. Yudin, M. and Alexandrov, M. (1983): *Graphical dialog on minicomputers for data interpretation of magnito-telluric sounding (2-dim models)*. In: "Problems of electromagnetic field investigation in a sea", Institute of Earth Electromagnetics of USSR Acad. of Science, Moscow, pp. 156-161 (rus.)

Glosario de términos

Complejidad de clase de métodos. Una característica integral del conjunto de métodos que significa efectividad promedio de los métodos dados sobre una clase de funciones.

Enfoque optimizado. Formalización de un problema de toma de decisión como un problema de búsqueda del óptimo de una función.

Error de búsqueda. Es la diferencia entre la ubicación del extremo real y el resultado de búsqueda de un método de optimización.

Función de preferencia. Conocida también como función de penalización que introduce una penalización a la función de criterio cuando se violan las restricciones del problema.

Función de criterio. Es una función que refleja matemáticamente la calidad de la solución de un problema dado. En las tareas de identificación de un modelo la función de criterio generalmente es una evaluación de la diferencia entre los datos experimentales y los datos de modelación.

Métodos de optimización. Son los métodos que buscan óptimos de una función escalar de una o varias variables sin o con restricciones.

Método directo de búsqueda aleatoria. Un método del grupo de los métodos de programación estocástica que busca el óptimo global a base de la evaluación de una función de criterio en un conjunto de puntos aleatorios distribuidos uniformemente.

Método de Nelder-Mead. Un método del grupo de los métodos no lineales de orden 0 que buscan el óptimo local. Éste método usa un simplex irregular con operaciones de expansión y contracción de paso y también de cambio de su dirección según la superficie de la función dada de criterio.

Método de sección dorada. Un método de una variable del grupo de los métodos no lineales de orden 0 que buscan el óptimo de una función unimodal. El algoritmo se basa en el concepto de que al realizar una evaluación de la función, el intervalo de incertidumbre se reduce en un factor constante $k=0.618034$.

Método tradicional de paso a paso. Un método de una variable del grupo de los métodos no lineales de orden 0 que buscan el óptimo local. Éste método usa expansiones y contracciones de paso.

Mínimo local. Valor mínimo de la función en un punto estacionario x que es el de menor valor de la función en comparación con otros valores en una vecindad del punto x .

Mínimo global. Valor mínimo de la función en un dominio dado de la función

Multi-time optimization. Es una opción de búsqueda automática disponible cuando la búsqueda no tuvo éxito dentro del número de iteraciones establecido y por lo que se repite varias veces

Nivel de confianza. Es una característica que se usa de los métodos de programación estocástica. Es el grado de certeza de que el mínimo real pertenece a la vecindad del punto alcanzado

One-time optimization. Es una opción de una única búsqueda automática

Programación estocástica. Es una rama de los métodos de optimización. Trata con situaciones donde algunos o todos los parámetros de la función de criterio y/o parámetros de búsqueda están descritos por variables aleatorias.

Programación no lineal. Es una rama de los métodos de optimización. Se resuelve el problema general de minimización con un objetivo no lineal y con varias restricciones de la función.

Programación orientada a objetos. Es una disciplina de diseño y programación de sistemas a desarrollar que consiste en la utilización de objetos y sus interacciones entre ellos, entendiéndose como objeto una instancia de una clase de un lenguaje de programación .

Indice de figuras

- Figura 1.3-1. Superficie típica de una función de criterio de un problema de Geofísica de prospección
- Figura 1.4-1. Método de Nelder-Mead
- Figura 1.4-2. Método de búsqueda aleatoria
- Figura 2.1-1. Función de criterio No.1
- Figura 2.1-2. Función de criterio No.2
- Figura 2.1-3. Función de criterio No.3
- Figura 2.1-4. Función de criterio No.4
- Figura 2.1-5. Función de criterio No.5
- Figura 2.3-1. Justificación de la asignación del coeficiente de multa
- Figura 2.3-2. El relieve de suma de función de criterio y función de preferencia
- Figura 2.4-1. Etapas típicas de búsqueda un óptimo
- Figura 2.4-2. Trayectoria de búsqueda en la etapa I
- Figura 2.4-3. Trayectoria de búsqueda en la etapa II
- Figura 2.4-4. Dinámica de búsqueda de la etapa II
- Figura 3.2-1. Diagrama de clases y módulos del sistema MAC
- Figura 3.2-2. Diagrama de interacción del sistema MAC en un problema real usando un módulo maestro
- Figura 3.2-3. Diagrama de procesos
- Figura 3.2-4. Información de MAC
- Figura 3.3-1. Menú principal del sistema y su barra de herramientas.
- Figura 3.3-2. Submenú de entrada/salida
- Figura 3.3-3. Submenú de opciones
- Figura 3.3-4. Submenú de la ejecución del proceso
- Figura 3.3-5. Submenú para la selección de la vista
- Figura 3.3-6. Definición de las condiciones de búsqueda
- Figura 3.3-7. Ajuste de los métodos de optimización
- Figura 3.3-8. Definición de una función de preferencia
- Figura 3.3-9. Selección del tipo de registro
- Figura A.2-1. Función que muestra un valle en forma de banana
- Figura A.2-2. Función que muestra un valle asimétrico
- Figura A.2-3. Función que muestra un valle contorsionado

Indice de tablas

- Tabla 2.1-1. Matriz de puntos para el cálculo de la función de criterio
- Tabla 2.2-1. Serie de experimentos para evaluar los parámetros del método NM
- Tabla 2.3-1. Convergencia del método NM con la función de criterio No.1
- Tabla 2.3-2. Convergencia del método NM con la función de criterio No.2
- Tabla 2.3-3. Convergencia del método NM con la función de criterio No.3
- Tabla 2.3-4. Convergencia del método NM con la función de criterio No.4
- Tabla 2.3-5 Resultados generalizados de los experimentos.
- Tabla 2.4-1. Convergencia bajo estrategias diferentes de búsqueda usando la función de criterio No.1
- Tabla 2.4-2. Convergencia bajo estrategias diferentes de búsqueda usando la función de criterio No.2
- Tabla 2.4-3. Convergencia bajo estrategias diferentes de búsqueda usando la función de criterio No.3
- Tabla 2.4-4. Convergencia bajo estrategias diferentes de búsqueda usando la función de criterio No.4
- Tabla 2.4-5 Resultados generalizados de los experimentos.

ANEXOS

A.1 Análisis de rendimiento de los métodos de optimización

Se presentan algunos resultados obtenidos por Nemirovski y su colaboradores [Nemirovski, 1989],

Definieron:

N – complejidad (dificultad) de clase de métodos, es decir:

a) Número de cálculo de la función (métodos de orden cero).

b) Número de cálculo de sus derivadas (métodos de 1er. orden)

n – dimensión del problema

ε - error de cálculo

χ - dominio de búsqueda del extremo.

Adicionalmente, introdujeron el número de peculiaridad del problema. Para una función de dos variables $f(x_1, x_2)$ este número Q se define por la siguiente forma.

Para la matriz:

$$A = \begin{pmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} \\ \frac{\partial^2 f}{\partial x_1 \partial x_2} & \frac{\partial^2 f}{\partial x_2^2} \end{pmatrix}$$

tenemos λ_1 -máximo número propio, λ_2 -mínimo número propio. Entonces,

$Q = \left| \frac{\lambda_1}{\lambda_2} \right| \geq 1$. Si $Q \rightarrow \infty$ el problema es mal condicionado, si $Q \rightarrow 1$ el problema está

bien condicionado.

Caso I.

Clase de funciones: suaves de orden-k.

Clase de métodos: todos, incluyendo determinísticos y aleatorios.

$$N(\varepsilon) \geq c(k, \chi) \left(\frac{1}{\varepsilon} \right)^{\frac{n}{k}} \quad (\text{A.1-1})$$

donde $c(k, \chi)$ es alguna constante. Si $n \rightarrow \infty$ se tiene un crecimiento característico de N. Por eso es absurdo plantear una cuestión de solución de todos los problemas de cualquier dimensión notable.

Caso II.

Clase de función: todos convexos en un conjunto convexo.

Clase de método: todos los que calculan valores de la función y sus derivadas.

$$C_1 \left\langle \frac{N(\varepsilon)}{1 + n \ln\left(\frac{1}{\varepsilon}\right)} \right\rangle C_2 \quad (\text{A.1-2})$$

donde C_1, C_2 son constantes. La asíntota depende de la geometría de χ :

Si χ es paralelepípedo $N(\varepsilon) \sim n \ln\left(\frac{1}{\varepsilon}\right)$, crece linealmente con n .

Si χ es elipsoide $N(\varepsilon) \sim \frac{1}{\varepsilon^2}$, no depende de n .

Caso III.

Clase de funciones: todos convexos en un conjunto convexo.

Clase de métodos: todos de orden 0.

$$N(\varepsilon) \sim P(n) \ln\left(\frac{n}{\varepsilon}\right) \quad (\text{A.1-3})$$

donde $P(n)$ es algún polinomio de orden n .

Caso IV.

Clase de funciones: suaves y convexos fuerte.

Clase de métodos: de gradiente.

$$N(\varepsilon) \sim Q \ln\left(\frac{1}{\varepsilon}\right) \quad (\text{A.1-4})$$

Para los otros métodos de primer orden (descenso más rápido, etc.) se tiene una baja evaluación:

$$N(\varepsilon) \geq c\sqrt{Q} \ln\left(\frac{1}{\varepsilon}\right) \quad (\text{A.1-5})$$

Conclusiones:

- Los métodos aleatorios no tiene ninguna ventaja sobre los métodos determinísticos, aunque hay quienes piensan que estos métodos son más poderosos en caso de los problemas de dimensión muy grande.
- Todas las evaluaciones no dependen del número de limitaciones del problema. Realmente las modificaciones de los métodos relacionados con limitaciones no incrementan su complejidad, por eso no hay que temer a usar muchas limitaciones. La ventaja de las limitaciones es la disminución del área de incertidumbre.
- Para una dimensión grande del problema, las evaluaciones son muy pesimistas según las fórmulas (A.1-1) y (A.1-2). Pero es posible organizar la búsqueda en subespacios.

Bibliografía

1. Nemirovski, A., Yudin, D. (1989): *Complexity and efficiency of optimization algorithms*. Publ. House "Nauka", Moscow (rus., transl. to engl.)

A.2 Métodos de construcción de funciones de prueba

Se pueden crear funciones de la clase “valle” mediante la técnica propuesta por [Batichev,1982]. Su enfoque se basa en la fórmula de la integración numérica.

$$\int_0^1 f(x)dx = \sum_0^n A_i f(x_i) \quad (\text{A.2-1})$$

donde $x_i \in [0, 1]$, $i = 0, \dots, n$, $A_i =$ coeficientes. Usando el método del trapezoide cuya fórmula es:

$$h \left[\frac{f(x_0)}{2} + f(x_1) + \dots + f(x_{n-1}) + \frac{f(x_n)}{2} \right]$$

O el método parabólico o de Simpson cuya fórmula es:

$$\frac{h}{3} [f(x_0) + 4f(x_1) + 2f(x_2) + \dots + f(x_n)]$$

Gauss determinó tanto A_i^* como x_i^* para la fórmula (A.2-1) la cual sería exacta para todos los polinomios de orden s :

$$f(x) = a_0 + a_1x + \dots + a_sx^s$$

aquí $i=0, \dots, n$, $n=(s+1)/2$. Pero si se toman las otras x entonces la fórmula (A.2-1) no será exacta, por consiguiente la fórmula:

$$F(x_1, x_2, \dots, x_n) = \left[\int_0^1 f(x)dx - \sum_0^n A_i^* f(x_k) \right]^2 \quad (\text{A.2-2})$$

tiene un mínimo en los puntos de Gauss. $x_0=x_0^*$, $x_1=x_1^*$, \dots , $x_n=x_n^*$. y esto es justamente lo que se quiere. Se tiene ahora:

$$\int_0^1 f(x)dx = \int_0^1 \sum_{i=0}^s a_i x^i = \sum_{i=0}^s a_i \int_0^1 x^i dx = \sum_{i=0}^s \frac{a_i}{i+1}$$

$$f(x_k) = \sum_{i=0}^s a_i x_k^i$$

Incluyendo estas formulas en (A.2-2) se tiene:

$$F(x_1, x_2, \dots, x_n) = \left[\sum_{i=0}^s \frac{a_i}{i+1} - \sum_{k=1}^n A_k^* \sum_{i=0}^s a_i x_k^i \right]^2 \quad (\text{A.2-3})$$

Aquí: a_0, a_1, \dots, a_s se pueden obtener de un generador de números aleatorios.

Por ejemplo [Batichev,1982]: si se consideran los límites de integración de $[-1, 1]$ en vez de $[0, 1]$ para $s=3$, se tiene que $n = (s+1)/2 = 2$.

1. Tomando un conjunto de coeficientes a_0, a_1, a_2, a_3 , se puede obtener:

$$F(x_1, x_2) = \frac{\pi^2}{4} \left[-21 - 32(x_1^3 + x_2^3 + 21(x_1^2 + x_2^2) + 13(x_1 + x_2)) \right]^2$$

Este es un valle en forma de banana. La figura A.5-1 refleja los resultados de esta modelación.

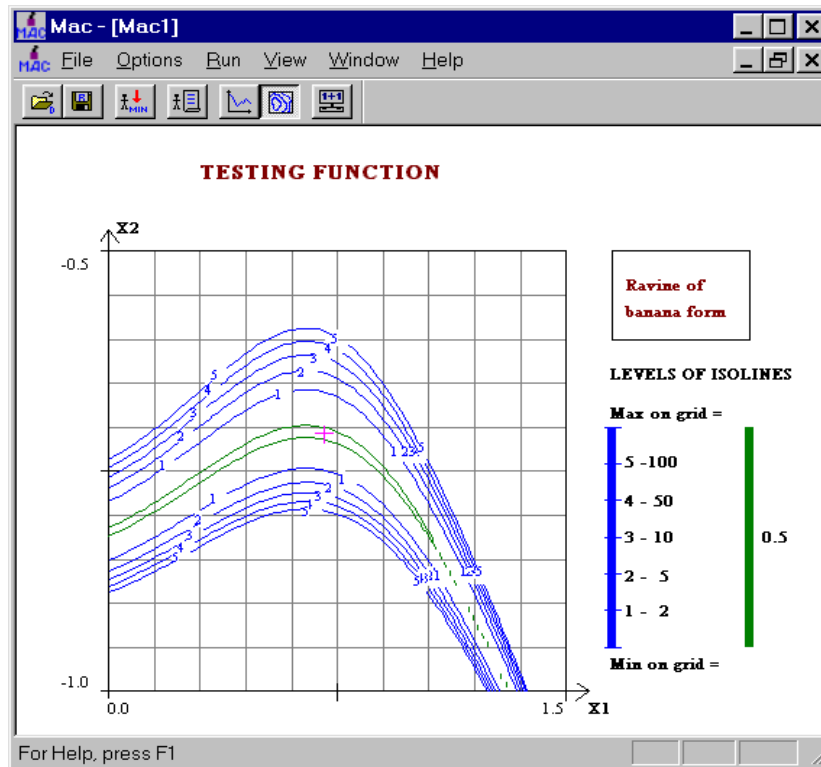


Fig. A.2-1. Función que muestra un valle en forma de banana

Aquí $x_1 \in [0., 1.5]$, $x_2 \in [-1., 0.]$

2. Tomando otro conjunto de coeficientes a_0, a_1, a_2, a_3 se puede obtener:

$$F(x_1, x_2) = \frac{\pi^2}{4} \left[1 - (x_1^3 + x_2^3) - (x_1^2 + x_2^2) - (x_1 + x_2) \right]^2$$

Este es un valle asimétrico. Se calculó la función F usando esta fórmula y los resultados se presentan en la figura A.2-2.

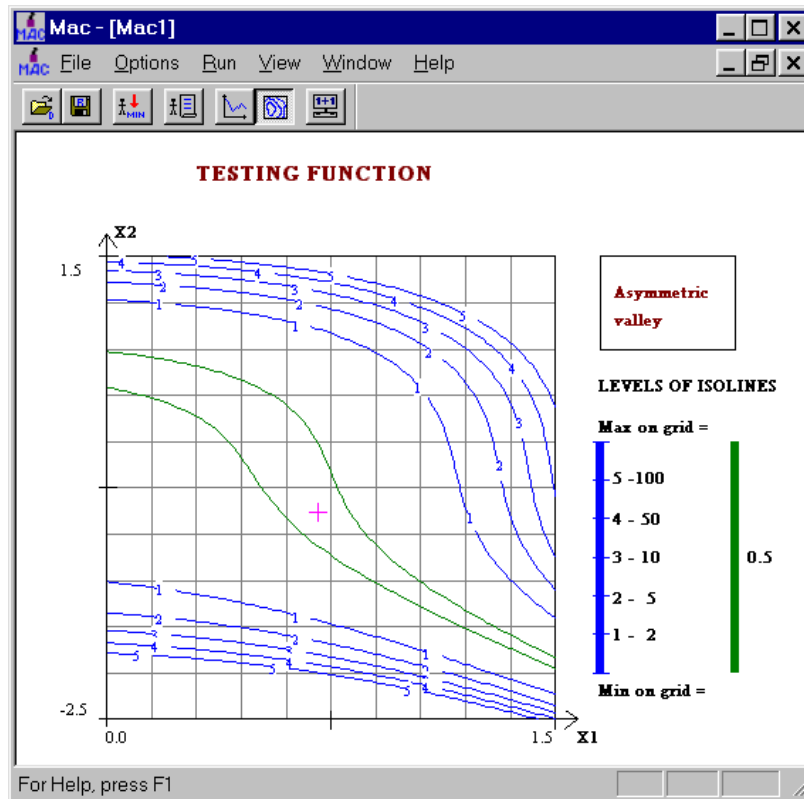


Fig. A.2-2. Función que muestra un valle asimétrico

Aquí $x_1 \in [0, 1.5]$, $x_2 \in [-2.5, 1.5]$

3. Tomando un tercer conjunto de coeficientes a_0, a_1, a_2, a_3 se obtiene:

$$F(x_1, x_2) = \frac{\pi^2}{4} \left[10 + 10(x_1^3 + x_2^3) - 10(x_1^2 + x_2^2) + 5(x_1 + x_2) \right]^2$$

Este es un valle contorsionado. Se calculó la función F usando esta fórmula. Los resultados se presentan en la figura A.2-3:

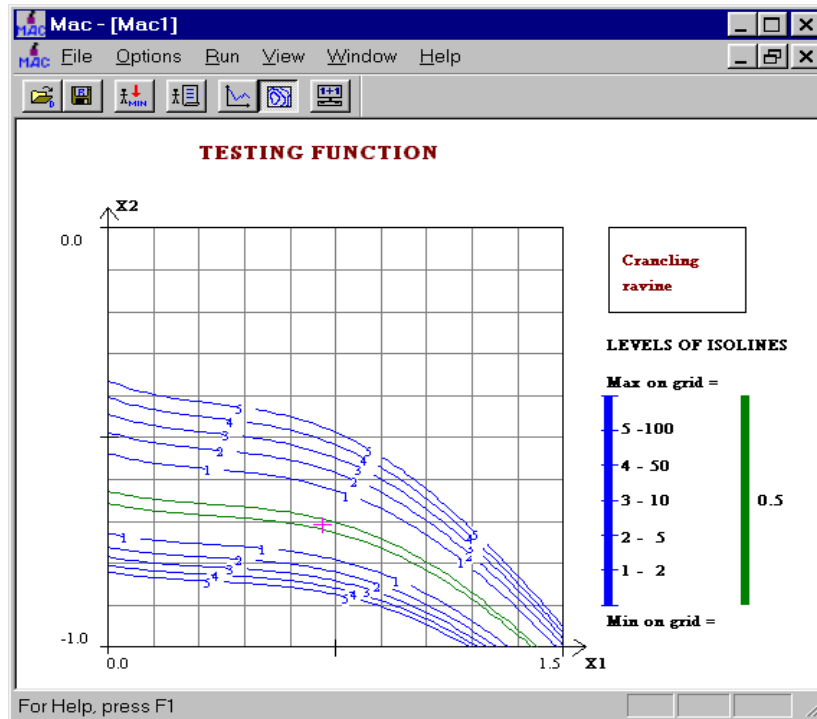


Fig. A.2-3. Función que muestra un valle contorsionado

Estas tres funciones tienen el mismo punto mínimo $\left(\frac{\sqrt{2}}{2}, -\frac{\sqrt{2}}{2}\right)$, punto conocido de las fórmulas integrales de Gauss. Por eso probando los métodos de optimización se espera encontrar un mínimo justamente en este punto.

Tomando $s=5,7,9\dots$ en la fórmula (A.2-3) se puede usar la misma técnica para construir las funciones de prueba en un espacio de 3,4,5... variables.

Bibliografía

1. Batichev, D. (1982): *Optimization methods*. Publ. House of GGU, Gorkyi (rus.)

A.3 Ejemplos de funciones de criterio

1. El archivo *Exp1_mac.txt*

Los parámetros son la tasa de emisión de contaminantes y la temperatura de los gases en la salida de una chimenea.

Matriz de Qs vs. Temp. [Región: Tula]

0.1 8.0 300 480

5 5

396.90	79.56	144.53	176.54	203.60
196.86	160.47	209.21	233.22	253.51
3.17	241.41	273.89	289.90	303.43
203.21	322.33	338.57	346.58	353.34
393.25	399.20	400.02	400.42	400.76

2. El archivo *Exp2_mac.txt*

Los parámetros son la tasa de emisión de contaminantes y la temperatura de los gases de salida de una chimenea.

Matriz de Qs vs. Temp. [Región: Tula]

1.0 8.0 300 480

7 7

395.87	416.17	214.56	161.94	177.34	178.89	186.57
383.67	406.12	140.14	148.56	165.29	170.82	178.14
370.34	396.91	89.56	144.54	176.55	203.61	211.47
216.78	196.87	160.49	209.22	233.23	253.52	257.87
264.26	7.17	241.41	273.90	289.90	303.43	316.41
318.45	203.21	322.33	338.58	346.58	353.34	359.03
390.13	393.25	399.20	400.02	402.42	403.76	403.90

3. El archivo *Exp3_mac.txt*

Los parámetros son las coordenadas georeferenciadas longitudinales de dos chimeneas.

Matriz de X1 vs X2 (Coordenadas) [Región: Cadereyta]

0 100 -50 -10

5 5

122.56	136.08	128.26	136.37	122.73
131.82	95.16	149.95	150.35	136.26
163.07	154.25	151.64	154.34	164.79
121.05	117.92	104.88	138.39	175.66
142.31	115.00	110.57	147.38	187.19

4. El archivo Exp4_mac.txt

Los parámetros son las tasas de emisión de contaminantes de dos chimeneas.

Matriz de Qs1 vs Qs2 [Región: Cadereyta]

4.0 8.0 4.0 8.0

7 7

172.32	170.76	163.56	139.13	111.05	151.34	180.16
161.50	155.44	151.27	140.91	133.11	146.52	153.67
174.12	166.78	164.32	158.17	147.73	150.63	155.84
176.55	160.16	163.89	150.86	122.52	148.35	159.49
163.81	158.74	162.76	166.53	172.96	170.42	182.71
166.92	165.25	150.43	167.84	206.10	217.43	234.61
178.31	171.28	163.61	214.30	221.61	236.17	248.19

A.4 Ejemplos del registro de resultados

1. Registro corto

=== START OF PROTOCOL ===

GLOBAL INFORMATION:

Data= Exp3_mac.txt
Comment= Matriz de Qs vs. Temp.
Dimension= 2 (5x5 points)
Method= Nelder-Mead
Parameters: A=1.0, B=0.5, G=2.0, D=0.1

OPTIONS OF CALCULATIONS:

Preference function isn't used
Limitations for Parameter-1: 0.100 8.0
Limitations for Parameter-2: 300.0 480.0
Max number of iterations= 50
Max accessible error= 0.001

RESULTS OF CALCULATIONS:

Final point for Parameter-1: 0.100
Final point for Parameter-2: 345.004
Final number of iterations= 20
Final error= 0.0007

=== END OF PROTOCOL ===

2. Registro completo

=== START OF PROTOCOL ===

GLOBAL INFORMATION:

Data= Exp3_mac.txt
Comment= Matriz de Qs vs. Temp.
Dimension= 2 (5x5 points)
Method= Nelder-Mead
Parameters: A=1.0, B=0.5, G=2.0, D=0.1

OPTIONS OF CALCULATIONS:

Preference function isn't used
Limitations for Parameter-1: 0.100 8.0
Limitations for Parameter-2: 300.0 480.0
Max number of iterations= 50
Max accessible error= 0.001

RESULTS OF CALCULATIONS:

Final point for Parameter-1: 0.100
Final point for Parameter-2: 345.004
Final number of iterations= 20
Final error= 0.0007

OPTIMIZATION PROCESS:

- 1). Par1= 4.050 Par2= 390.0 Function= 273.897
- 2). Par1= 3.260 Par2= 403.500 Function= 253.787
- 3). Par1= 2.865 Par2= 392.250 Function= 236.129
- 4). Par1= 1.087 Par2= 413.625 Function= 191.582
- 5). Par1= 0.100 Par2= 401.812 Function= 152.938
- 6). Par1= 0.100 Par2= 401.812 Function= 152.938
- 7). Par1= 0.100 Par2= 401.812 Function= 152.938
- 8). Par1= 0.100 Par2= 373.406 Function= 120.576
- 9). Par1= 0.100 Par2= 363.843 Function= 106.770
- 10). Par1= 0.100 Par2= 352.031 Function= 89.714
- 11). Par1= 0.100 Par2= 352.031 Function= 89.714
- 12). Par1= 0.100 Par2= 352.031 Function= 89.714
- 13). Par1= 0.100 Par2= 348.128 Function= 84.080
- 14). Par1= 0.100 Par2= 344.613 Function= 82.289
- 15). Par1= 0.100 Par2= 344.613 Function= 82.289
- 16). Par1= 0.100 Par2= 346.371 Function= 81.542
- 17). Par1= 0.100 Par2= 345.492 Function= 80.273
- 18). Par1= 0.100 Par2= 345.004 Function= 79.569
- 19). Par1= 0.100 Par2= 345.004 Function= 79.569
- 20). Par1= 0.100 Par2= 345.004 Function= 79.569

=== END OF PROTOCOL ===

A.5 Códigos del módulo principal

1. Archivo cabecera

```
// MacDoc.h: header file for the CMacDoc class
//
// Version: 1.2
// Data: August, 2000
//
// Programmer: Roberto Jurado I.M.P.-(MEXICO)
// Programmer and System Analyst: Mikhail Alexandrov C.I.C.-I.P.N.(MEXICO)
//
///////////////////////////////////////////////////////////////////

#ifndef AFX_MACDOC_H_3757526F_C8AE_11D2_8602_444553540000__INCLUDED_
#define AFX_MACDOC_H_3757526F_C8AE_11D2_8602_444553540000__INCLUDED_

#if _MSC_VER >= 1000
#pragma once
#endif // _MSC_VER >= 1000

class CMacDoc : public CDocument
{
protected: // create from serialization only
    CMacDoc();
    DECLARE_DYNCREATE(CMacDoc)

// Attributes

public:

// A1. File operations

CStdioFile m_AscFile;           // CStdioFile object for text data

int m_iTracer;                 // Indicator of tracing input
                                // IDNO - without tracing
                                // IDYES - with tracing

CString m_pszDataName;         // Current text name (*.txt) <= txt-format
CString m_pszFullDataName;     // Current full path to text

CString m_pszResultName;       // Current Protocol (*.txt) <= txt-format
CString m_pszFullResultName;   // Current full path to protocol

// A2. Matrix and Vectors

CString m_pszDataComment;      // Comment about data

static float m_fMatrix[10000]; // Experimental matrix
static float m_fPar1[100];     // Coordinates 1
static float m_fPar2[100];     // Coordinates 2

static int m_iNPar1;           // Dimension 1 (number of rows)
static int m_iNPar2;           // Dimension 2 (number of columns)

// A3. Calculation

// A3.1 Parameters of optimization

float m_fFixedError;           // Admissible error
int m_iFixedIter;              // Max number of iteration
float m_fFixedProbability;     // Given probability for RS-method
float m_fCurrentError;         // Reached error
int m_iCurrentIter;            // Executed number of iteration

static float m_fPar1Low;       // Min for Par1
```

```

static float m_fPar1Current; // Current value for Par 1
static float m_fPar1High; // Max for Par1
static float m_fPar2Low; // Min for Par2
static float m_fPar2Current; // Current value for Par 2
static float m_fPar2High; // Max for Par2

static float m_fPenaltyCoef; // Penalty coefficient of preference function
// Preference function has view C*[...]

static float m_fPar1Guess; // Preferable value for Par 1
static float m_fPar2Guess; // Preferable value for Par 2

static int m_iDimension; // Dimension of problem
// =0 - No any data
// =1 - 1D problem (input data = vector)
// =2 - 2D problem (input data = matrix)

static int m_iStandard; // Indicator of standard function
// =0 - no any standard function
// =1 - standard function is established:
// Parabola (if Dimension = 1)
// Banana of Rozenbrock (if Dimension = 2)

int m_iMethod1D; // Indicator of 1D-method
// =0 - Step-By-Step method
// =1 - Gold Section method

int m_iMethod2D; // Indicator of 2D-method
// =0 - Nelder-Mead method
// =1 - Random Search method

int m_iProtocol; // Indicator of result registration:
// 0 - brief protocol of optimization process
// 1 - full protocol of optimization process

// A3.2 Parameters of methods

float m_fNelderAlpha; // Parameter of Nelder-Mead method
float m_fNelderBeta; // Parameter of Nelder-Mead method
float m_fNelderDelta; // Parameter of Nelder-Mead method
float m_fNelderGamma; // Parameter of Nelder-Mead method

float m_fStepAlpha; // Parameter of step-by-step method
float m_fStepBeta; // Parameter of step-by-step method
float m_fStepDelta; // Parameter of step-by-step method

// A4. Visualization

// A4.1 Readiness of results

BOOL m_bProcessReady; // Indicator of result readiness

// A4.2 View

BOOL m_bViewOfDynamics; // Indicator of View: Dynamics of criteria change
BOOL m_bViewOfTrajectory; // Indicator of View: Trajectory of searching

// A4.3 Trajectory

static float m_fFunTrajectory[100]; // Function =F(number of step)
static float m_fPar1Trajectory[100]; // First coordinate =Par1(number of step)
static float m_fPar2Trajectory[100]; // Second coordinate =Par2(number of step)
static int m_ilterTrajectory; // Size of array

// A5. Current results

int m_iCurrentResult; // Current result of modelling :
// # 3 - reserve
// # 2 - reserve
// # 1 - reserve

```

```

        // # 0 - no errors
        // # -1 - no memory
        // # -2 - invalid data
        // # -3 - errors in input/output system

BOOL      m_bIsResultSaved;    // Indicator: were results saved
                                   // after last changes and calculation
                                   // TRUE - YES
                                   // FALSE - NO

// Implementation

// I1. Standard functions

// I1.1 Standard functions: destructor and debugging

public:
    virtual ~CMacDoc();
#ifdef _DEBUG
    virtual void AssertValid() const;
    virtual void Dump(CDumpContext& dc) const;
#endif

// I1.2 Standard functions: input/output

public:
// Overrides
    // ClassWizard generated virtual function overrides
    //{AFX_VIRTUAL(CRcgDoc)
    public:
    virtual BOOL OnNewDocument();
    virtual void Serialize(CArchive& ar);
    //}AFX_VIRTUAL

    virtual BOOL SaveModified();

protected:

// I2. Data Processing

static long Fun (int iN, float fPar [], float &fF); // Calculation of criteria function

static long View(int iN, float fPar [], float fF); // Vizualization of optimization process

// I3. Input/Output

// I3.1 Data files

void      FileOpenData(); // Open file with matrix (vector),
                                   // read it and close file.
                                   // File name is set in OnOpenData()

// I3.2 Result files

void      FileResultSaveAs(); // Open result file, write it and close file.
                                   // File name is set in OnResultSaveAs()

// I4. Generated message map functions

// Generated message map functions
protected:
    //{AFX_MSG(CMacDoc)
    afx_msg void OnOpenData();           // Menu 1.2 Define file for reading matrix
    afx_msg void OnOpen1DExample();      // Menu 1.3 Define parabola as data
    afx_msg void OnOpen2DExample();      // Menu 1.4 Define "Banana of Rozenbroke" as data
    afx_msg void OnSaveResult();         // Menu 1.5 Write result to file
    afx_msg void OnUpdateSaveResult(CCmdUI* pCmdUI); // Menu 1.5 Update indicator
    afx_msg void OnSaveResultAs();       // Menu 1.6 Define file for writing result
    afx_msg void OnUpdateSaveResultAs(CCmdUI* pCmdUI); // Menu 1.6 Update indicator
    afx_msg void OnOptionOptim();        // Menu 2.1 Set parameterts of optimization
    //}AFX_MSG

```

```

afx_msg void OnUpdateOptionOptim(CCmdUI* pCmdUI); // Menu 2.1 Update indicator
afx_msg void OnOptionMethods(); // Menu 2.2 Set parameters of methods
afx_msg void OnUpdateOptionMethods(CCmdUI* pCmdUI); // Menu 2.2 Update indicator
afx_msg void OnOptionPenalty(); // Menu 2.3 Set parameters of penalty function
afx_msg void OnUpdateOptionPenalty(CCmdUI* pCmdUI); // Menu 2.3 Update indicator
afx_msg void OnOptionRegistration(); // Menu 2.4 Choice a contents of protocol
afx_msg void OnUpdateOptionRegistration(CCmdUI* pCmdUI); // Menu 2.4 Update indicator
afx_msg void OnRunOne(); // Menu 3.1 Run optimization process 1 time
afx_msg void OnUpdateRunOne(CCmdUI* pCmdUI); // Menu 3.1 Update indicator
afx_msg void OnRunMulti(); // Menu 3.1 Run optimization process many times
afx_msg void OnUpdateRunMulti(CCmdUI* pCmdUI); // Menu 3.2 Update indicator
afx_msg void OnViewDynamics(); // Menu 4.3 Select view: dynamics of criteria change
afx_msg void OnUpdateViewDynamics(CCmdUI* pCmdUI); // Menu 4.3 Update indicator
afx_msg void OnViewTrajectory(); // Menu 4.4 Select view: trajectory of searching
afx_msg void OnUpdateViewTrajectory(CCmdUI* pCmdUI); // Menu 4.4 Update indicator
//}}AFX_MSG
DECLARE_MESSAGE_MAP()

};

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

//{{AFX_INSERT_LOCATION}}
// Microsoft Developer Studio will insert additional declarations immediately before the previous line.

#endif // !defined(AFX_MACDOC_H__3757526F_C8AE_11D2_8602_444553540000__INCLUDED_)

```

2. Parte del calculo del módulo principal

```

////////////////////////////////////
// CMacDoc commands
//
// PART I. P R O C E E D I N G
//
// OnRunOne      - Run optimization process one
// OnUpdateRunOne - Update indicator
// OnRunMulti    - Run optimization process many times
// OnUpdateRunMulti - Update indicator
// Fun          - Calculating criteria function
// View         - Accumulating results on every step
//
////////////////////////////////////

void CMacDoc::OnRunOne()
// Run optimization process one
{
float fPCurrent[2],fPHigh[2],fPLow[2];
int iN,iKey;
float fPar1LowNew, fPar1HighNew;

long ind;
float fWork[100];

// 0. Clean the screen

m_iCurrentResult=100; // This result is mean only
// the view initialization (100 #[-3:3], see up).
UpdateAllViews(NULL);

if(m_iDimension==0) return;

// 1. Initial data

m_iCurrentIter = 0;
m_fCurrentError = 0.0f;

```

```

m_ilterTrajectory=0;

// 2. Optimization process

// 2.1 StepByStep method

if((m_iDimension==1)&&(m_iMethod1D==0))
{
    ind = StepByStep(m_fPar1Low,m_fPar1High,m_fPar1Current,
                    m_fFixedError,m_iFixedIter, m_fCurrentError, m_iCurrentIter,
                    Fun, View,
                    m_fStepAlpha, m_fStepBeta, m_fStepDelta);
    goto Finish;
}

// 2.2 Gold Section method

else
if((m_iDimension==1)&&(m_iMethod1D==1))
{
    ind=GoldSection(m_fPar1Low,m_fPar1High,m_fFixedError,m_iFixedIter,
                   fPar1LowNew, fPar1HighNew,m_fCurrentError, m_iCurrentIter,
                   Fun,View);

    if(ind==0)
    {
        m_fPar1Current = (fPar1HighNew + fPar1LowNew)/2.f;
    }

    goto Finish;
}

// 2.3 Nelder-Mead method

else
if((m_iDimension==2)&&(m_iMethod2D==0))
{
    iN=2;
    iKey=0;

    fPLow[0] = m_fPar1Low;
    fPHigh[0] = m_fPar1High;
    fPCurrent[0] = m_fPar1Current;
    fPLow[1] = m_fPar2Low;
    fPHigh[1] = m_fPar2High;
    fPCurrent[1] = m_fPar2Current;

    ind = NonLinearSimplex
        (iN, iKey,
         fPLow,fPHigh, fPCurrent,
         m_fFixedError,m_iFixedIter, m_fCurrentError, m_iCurrentIter,
         fWork,
         Fun,View,
         m_fNelderAlpha,m_fNelderBeta,m_fNelderGamma,m_fNelderDelta);

    if(ind==0)
    {
        m_fPar1Current = fPCurrent[0];
        m_fPar2Current = fPCurrent[1];
    }
    goto Finish;
}

// 2.4 Random Searching method

else
if((m_iDimension==2)&&(m_iMethod2D==1))
{
    iN=2;
    fPLow[0] = m_fPar1Low;
    fPHigh[0] = m_fPar1High;

```

```

fPLow[1] = m_fPar2Low;
fPHigh[1] = m_fPar2High;

ind = RandomSearch(iN, fPLow, fPHigh, fPCurrent,
                  m_fFixedProbability, m_fFixedError, m_iFixedIter,
                  m_fCurrentError, m_iCurrentIter, Fun, View);

if(ind==0)
{
    m_fPar1Current = fPCurrent[0];
    m_fPar2Current = fPCurrent[1];
}

goto Finish;
}

// 3. Finish

Finish: ;

if(ind==0)
{
    m_bProcessReady = TRUE;
}

if(ind==-1)
{
    AfxMessageBox("Error of method: no memory",MB_ICONEXCLAMATION|MB_OK);
    return;
}

if(ind==-2)
{
    AfxMessageBox("Error of method: invalid parameters",MB_ICONEXCLAMATION|MB_OK);
    return;
}

if(ind==-3)
{
    AfxMessageBox("Error of method: invalid calculation of criteria function",MB_ICONEXCLAMATION|MB_OK);
    return;
}

// 4. Vizualization

m_bIsResultSaved = FALSE;

m_iCurrentResult = 0;
UpdateAllViews(NULL);

return;
}

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

void CMacDoc::OnUpdateRunOne(CCmdUI* pCmdUI)
// Update indicator
{
    pCmdUI->Enable( m_iDimension >0 );
}

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

void CMacDoc::OnRunMulti()
// Run optimization process many times
{
#define MAXCYCLES 100
float fPCurrent[2],fPHigh[2],fPLow[2];
int iN,iKey;
float fPar1LowNew, fPar1HighNew;
float w_fPar1Low, w_fPar1High;

```

```

long ind;
float fWork[100];

float fMin1Current;
float fMin2Current;

int iIterTotal;
int iCycles;

// 0. Clean the screen

m_iCurrentResult=100; // This result is mean only
// the view initialization (100 #[-3:3], see up).
UpdateAllViews(NULL);

if(m_iDimension==0) return;

// 1. Initial data

m_iCurrentIter = 0;
m_fCurrentError = 0.0f;

m_iIterTrajectory=0;

// 2. Optimization process

// 2.1 StepByStep method

if((m_iDimension==1)&&(m_iMethodID==0))
{
    iIterTotal = 0;
    iCycles = 0;

L_SBS0: ;
    ind = StepByStep(m_fPar1Low,m_fPar1High,m_fPar1Current,
                    m_fFixedError,m_iFixedIter, m_fCurrentError, m_iCurrentIter,
                    Fun, View,
                    m_fStepAlpha, m_fStepBeta, m_fStepDelta);

    iCycles = iCycles + 1;
    if(iCycles>=MAXCYCLES)goto L_SBS1;

    if((m_fCurrentError > m_fFixedError)&&(ind==0))
    {
        iIterTotal = iIterTotal+ m_iCurrentIter;
        goto L_SBS0;
    }

L_SBS1: ;
    m_iCurrentIter = iIterTotal + m_iCurrentIter;
    goto Finish;
}

// 2.2 Gold Section method

else
if((m_iDimension==1)&&(m_iMethodID==1))
{
    iIterTotal = 0;
    iCycles = 0;

    w_fPar1Low = m_fPar1Low;
    w_fPar1High = m_fPar1High;

L_GS0: ;
    ind=GoldSection(m_fPar1Low,m_fPar1High,m_fFixedError,m_iFixedIter,
                    fPar1LowNew, fPar1HighNew,m_fCurrentError, m_iCurrentIter,
                    Fun,View);

```



```

iCycles = iCycles + 1;
if(iCycles>=MAXCYCLES)goto L_GS1;

if((m_fCurrentError > m_fFixedError)&&(ind==0))
{
  ilterTotal = ilterTotal+ m_iCurrentIter;
  m_fPar1Low = fPar1LowNew;
  m_fPar1High = fPar1HighNew;
  goto L_GS0;
}

L_GS1: ;
m_fPar1Low = w_fPar1Low;
m_fPar1High = w_fPar1High;

if(ind==0)
{
  m_fPar1Current = (fPar1HighNew + fPar1LowNew)/2.f;
}

m_iCurrentIter = ilterTotal + m_iCurrentIter;
goto Finish;
}

// 2.3 Nelder-Mead method

else
if((m_iDimension==2)&&(m_iMethod2D==0))
{
  iN = 2;
  iKey = 0;

  fPLow[0] = m_fPar1Low;
  fPHigh[0] = m_fPar1High;
  fPCurrent[0] = m_fPar1Current;
  fPLow[1] = m_fPar2Low;
  fPHigh[1] = m_fPar2High;
  fPCurrent[1] = m_fPar2Current;

  ilterTotal = 0;
  iCycles = 0;

L_NM0: ;
ind = NonLinearSimplex
  (iN, iKey,
  fPLow, fPHigh, fPCurrent,
  m_fFixedError, m_iFixedIter, m_fCurrentError, m_iCurrentIter,
  fWork,
  CMacDoc::Fun, CMacDoc::View,
  m_fNelderAlpha, m_fNelderBeta, m_fNelderGamma, m_fNelderDelta);

iCycles = iCycles + 1;
if(iCycles>=MAXCYCLES)goto L_NM1;

if((m_fCurrentError > m_fFixedError)&&(ind==0))
{
  ilterTotal = ilterTotal+ m_iCurrentIter;
  iKey=1;
  goto L_NM0;
}

L_NM1: ;
if(ind==0)
{
  m_fPar1Current = fPCurrent[0];
  m_fPar2Current = fPCurrent[1];
}

m_iCurrentIter = ilterTotal + m_iCurrentIter;

```

```

    goto Finish;
}

// 2.4 Random Searching method

else
if((m_iDimension==2)&&(m_iMethod2D==1))
{
    iN=2;
    fPLow[0] = m_fPar1Low;
    fPHigh[0] = m_fPar1High;
    fPLow[1] = m_fPar2Low;
    fPHigh[1] = m_fPar2High;

    iIterTotal = 0;
    iCycles = 0;
    fMin1Current = 1.E32f;
    fMin2Current = 1.E32f;

L_RS0: ;
    ind = RandomSearch(iN, fPLow, fPHigh, fPCurrent,
        m_fFixedProbability, m_fFixedError, m_iFixedIter,
        m_fCurrentError, m_iCurrentIter,
        Fun, View);

    if(ind!=0) goto L_RS1;
    iIterTotal = iIterTotal+ m_iCurrentIter;
    m_fCurrentError = 1.f - float( exp(log(1.-m_fFixedProbability)/double(iIterTotal)));

    iCycles = iCycles + 1;
    if(iCycles>=MAXCYCLES)goto L_RS1;

    if(m_fCurrentError > m_fFixedError)
    {
        fMin1Current = __min(fMin1Current,fPCurrent[0]);
        fMin2Current = __min(fMin2Current,fPCurrent[1]);
        goto L_RS0;
    }

L_RS1: ;
    if(ind==0)
    {
        fMin1Current = __min(fMin1Current,fPCurrent[0]);
        fMin2Current = __min(fMin2Current,fPCurrent[1]);
        m_fPar1Current = fMin1Current;
        m_fPar2Current = fMin2Current;
    }

    m_iCurrentIter = iIterTotal;
    goto Finish;
}

// 3. Finish

Finish: ;

if(ind==0)
{
    m_bProcessReady = TRUE;
}

if(ind==-1)
{
    AfxMessageBox("Error of method : no memory",MB_ICONEXCLAMATION|MB_OK);
    return;
}

if(ind==-2)
{
    AfxMessageBox("Error of method: invalid parameters",MB_ICONEXCLAMATION|MB_OK);
}

```

```

    return;
}

if(ind==-3)
{
    AfxMessageBox("Error of method: input/output",MB_ICONEXCLAMATION|MB_OK);
    return;
}

// 4. Vizualization

m_blsResultSaved = FALSE;

m_iCurrentResult = 0;
UpdateAllViews(NULL);

return;
}

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

void CMacDoc::OnUpdateRunMulti(CCmdUI* pCmdUI)
{
    pCmdUI->Enable( m_iDimension >0 );
}

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

long CMacDoc::Fun(int iN, float fPar [], float &fF)
// Calculating function according values of parameters
{
    float fFMatrix;
    float fP1,fP2;
    float fPenaltyFun;
    float fWork,fWork1,fWork2;
    int ind;

    fP1=fPar[0];
    fP2=fPar[1];

    if(m_iDimension==0) return -2;

// 1. Using vector and matrix

    if(m_iStandard==0)
    {
        if(m_iDimension==1) // Case: 1D
        {
            ind = Interpol_1D_1(1,m_iNPar1,m_fPar1,m_fMatrix,1,&fP1,&fFMatrix);

            if(m_fPar1Low!=m_fPar1High)
                fPenaltyFun = m_fPenaltyCoef*ABS(fP1-m_fPar1Guess)/(m_fPar1High-m_fPar1Low);
            else
                fPenaltyFun = m_fPenaltyCoef*ABS(fP1-m_fPar1Guess);

            fF = fFMatrix + fPenaltyFun;
            fF = __max(fF,0.f);

            return 0;
        }
        else
            if(m_iDimension==2) // Case: 2D
            {
                ind = Interpol_2D_1(m_iNPar1,m_iNPar2,m_fPar1,m_fPar2,m_fMatrix,
                    1,1,&fP1,&fP2,&fFMatrix);

                fWork1=0.f;
                fWork2=0.f;
                if(m_fPar1Low!=m_fPar1High)
                    {

```

```

        fWork1 = (fP1-m_fPar1Guess)/(m_fPar1High-m_fPar1Low);
        fWork1 = fWork1*fWork1;
    }
    if(m_fPar2Low!=m_fPar2High)
    {
        fWork2 = (fP2-m_fPar2Guess)/(m_fPar2High-m_fPar2Low);
        fWork2 = fWork2*fWork2;
    }

    fWork = float(sqrt(double(fWork1+fWork2)));

    fPenaltyFun = m_fPenaltyCoef*fWork;

    fF      = fFMatrix + fPenaltyFun;
    fF      = __max(fF,0.f);

    return 0;
}
}

// 2. Using standard function
if(m_iStandard==1)
{
    if(m_iDimension==1)        // Case: 1D
    {
        fFMatrix=(fP1-1.f)*(fP1-1.f)*(fP1+1.f)*(fP1+1.f);

        if(m_fPar1Low!=m_fPar1High)
            fPenaltyFun = m_fPenaltyCoef*ABS(fP1-m_fPar1Guess)/(m_fPar1High-m_fPar1Low);
        else
            fPenaltyFun = m_fPenaltyCoef*ABS(fP1-m_fPar1Guess);

        fPenaltyFun = m_fPenaltyCoef*ABS(fP1-m_fPar1Guess);

        fF      = fFMatrix + fPenaltyFun;
        fF      = __max(fF,0.f);

        return 0;
    }
    else
    if(m_iDimension==2)        // Case: 2D
    {
        fFMatrix=100.f*(fP2-fP1*fP1)*(fP2-fP1*fP1) + (1.f-fP1)*(1.f-fP1);
        fWork1=0.f;
        fWork2=0.f;
        if(m_fPar1Low!=m_fPar1High)
        {
            fWork1 = (fP1-m_fPar1Guess)/(m_fPar1High-m_fPar1Low);
            fWork1 = fWork1*fWork1;
        }
        if(m_fPar2Low!=m_fPar2High)
        {
            fWork2 = (fP2-m_fPar2Guess)/(m_fPar2High-m_fPar2Low);
            fWork2 = fWork2*fWork2;
        }

        fWork = float(sqrt(double(fWork1+fWork2)));

        fPenaltyFun = m_fPenaltyCoef*fWork;

        fF      = fFMatrix + fPenaltyFun;
        fF      = __max(fF,0.f);

        return 0;
    }
}
}

return 0;

```

```

}

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

long CMacDoc::View(int iN, float fPar[], float fF)
// Accumulating results on every step (no more last 100 steps)
{
int i;

if(m_ilterTrajectory<100)
{
m_ilterTrajectory = m_ilterTrajectory + 1;
m_fPar1Trajectory[m_ilterTrajectory-1] = fPar[0];
m_fPar2Trajectory[m_ilterTrajectory-1] = fPar[1];
m_fFunTrajectory[m_ilterTrajectory-1] = fF;
}
else
{
for(i=1; i<=99; i++)
{
m_fPar1Trajectory[i-1] = m_fPar1Trajectory[i];
m_fPar2Trajectory[i-1] = m_fPar2Trajectory[i];
m_fFunTrajectory[i-1] = m_fFunTrajectory[i];
}
m_fPar1Trajectory[99] = fPar[0];
m_fPar2Trajectory[99] = fPar[1];
m_fFunTrajectory[99] = fF;
}

return 0;
}

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

```

A.6 Códigos de la biblioteca de optimización

1. Archivo cabecera

```

// 'optim.h' - header file for the library of
// mathematical programming
//
// Version : 2.0
// Data : January, 2000
//
// Programmers:
// Mikhail Alexandrov, CIC-IPN (MEXICO)
// Roberto Jurado, IMP-(MEXICO)
//
////////////////////////////////////

/*
    UTILITIES OF CLASSES

    CONTENTS OF LIBRARY

NonLinearSimplex  Nelder-Mead method (non-linear simplex-method)
RandomSearch     Method of random search
GoldSection      Gold section method
StepByStep       Ordinary step by step method
Random           Get standard random number (0-1)
RunRandom        Define series of random values

=====
*/

long NonLinearSimplex(int iN, int iKey,
                    float fXl[], float fXh[], float fX[],

```



```

#include <math.h>
#include <time.h>
#include "optim.h"

long NonLinearSimplex(int iN, int iKey,
                    float fXl[], float fXh[], float fX[],
                    float fEpsMax, int ilterMax, float& fEps, int& ilter,
                    float fWork[],
                    long (*Fun) (int ,float[] ,float& ),
                    long (*View) (int ,float[] ,float ),
                    float fALPHA, float fBETA,
                    float fGAMMA, float fDELTA)

```

```

////////////////////////////////////
//
// Subroutine for function minimization by non-linear
// simplex method (method of destorbed polyhedron)
// with limits in the form of non-equalities.
// Category: multidimensional step by step method.
// Possibility: local minimum
//
//
// LITERATURE: M. Himmelblau 'Appled nonlinear programming'
//             McGraw-Hill Book Company, 1972.
//             This program was adapted by M.Alexandrov,
//             Moscow Control Problem Institute
//
// PROGRAMS, THAT USE THIS ONE:
// PROGRAMS, USED IN THIS ONE: Fun, View
//
// INPUT DATA:
// iN      - dimension of problem
// iKey    - indicator of repeated input
//          0 - first input
//          1 - repeated input
// fXl,fXh - lower and higher restriction,
//            they are vectors by dimension 'n'
// fX      - initial approximation;
//            this is the vector by dimension 'n'
// ilterMax - the limit number of iteration
// fEpsMax - the limit relative size of polyhedron
//
// OUTPUT DATA:
// fX      - value of argument, obtained on the last iteration
// ilter   - obtained number of iteration
// fEps    - obtained relative size of polyhedron
//
// WORK ARRAYS:
// fWork   - array for conservation simplex and function values
//            of previous application, dimension '(n+1)*(n+4)'
//
// RETURN DATA:
// 0       - All is OK!
// -1      - no memory
// -2      - invalid parameters
// -3      - error of input/output (other subprograms)
//
//-----
//
// Parameters , being appointed
//
// ALPFA   coefficient of reflection
// BETA    coefficient of pressure
// GAMMA   coefficient of dilation
// DELTA   coefficient of relative size of polyhedron
//            at the begining of calculation
//
// Standard values of method's parameters:

```

```

// ALPFA=1, BETA=0.5, GAMMA=2, DELTA=0.1
//
//-----
//
// Description of subprogram
//
// Fun -name of subprogram, which determines criteria function;
// ---
// long Fun (int iN, float fX[], float &fF)
// input parameters:
//   iN - dimension of problem
//   fX - vector of arguments
// output parameters:
//   fF - value of criterion function
//
// View -name of virtual subprogram, providing visualization
// ---- of arguments and function after each step of searching
//
// long View(int iN, float fX[], float fF)
// input parameters:
//   iN - dimension of problem
//   fX - vector of arguments
//   fF - value of criterion function
//
//
///////////////////////////////////////////////////////////////////
{
int i,j,k,jgood,jbad;
int iIndex,iIndex2,iIndex3,iIndex4,iIndexBad,iIndexGood;
int ind;
int n,n1,n2 ,n3,n4;
float fF, fFmax, fFmin, fFmax2, fDx, fW;
float* fXd;
float* fFmind;
float* fStep;
float* fXw;
float* fSave=fWork;
float ALPHA,BETA,GAMMA,DELTA;

// Beginning of subprogram

if((iN<=1)|| (fEpsMax<0.f)|| (iIterMax<=1)) return -2;

if((fALPHA<=0.f)|| (fBETA<=0.f)|| (fBETA>=1.f)||
(fGAMMA<=1.f)|| (fDELTA<=0.f)|| (fDELTA>=1.f)) return -2;

n=iN;
n1=n+1; // Number of nodes
n2=n+2; // Number of point - center of gravity
n3=n+3; // Number of point - reflection/shooting
n4=n+4; // Number of point - distance/on success

ALPHA = fALPHA;
BETA = fBETA;
GAMMA = fGAMMA;
DELTA = fDELTA;

if(!fXl) return -2;
if(!fXh) return -2;
if(!fX) return -2;
if(!fWork) return -2;

ind =0;
for(i=1; i<=iN; i++)
{
if((fX[i-1]<fXl[i-1])||(fX[i-1]>fXh[i-1])) return -2;
if(fXl[i-1]>fXh[i-1]) return -2;
if(fXl[i-1]==fXh[i-1]) ind=ind+1;
}
}

```



```

}
if(ind==iN) return -2;

fXd = new float [n*(n+4)];
if( !fXd ) return -1;

fFmind = new float [n4];
if( !fFmind ) return -1;

fStep = new float [n1];
if( !fStep ) return -1;

fXw = new float [n];
if( !fXw ) return -1;

ilter=0;          // Counter of iterations

////////////////////////////////////
//
// Generation of initial simplex
// (n-vectors, located round initial point)
//

if (iKey<=0)      // First input
{
// Forming initial simplex

for(i=1;i<=n; i++)
    fStep[i-1]=(fXh[i-1]-fXl[i-1])*DELTA;

for(i=1;i<=n; i++)
    fX[i-1]=MAX(MIN(fX[i-1],fXh[i-1]-fStep[i-1]),fXl[i-1]);

for(j=1; j<=n1; j++)
for(i=1; i<=n; i++)
{
    iIndex = COMINDEX(n,i,j);
    fXd[iIndex-1]=fX[i-1];
}

// Thin moment ! All unit vectors must be
// located inside area

for(j=2;j<=n1; j++)
{
    iIndex=COMINDEX(n,j-1,j);
    fXd[iIndex-1]=MIN(fXd[iIndex-1]+fStep[j-2],fXh[j-2]);
}

for(j=1; j<=n1; j++)
for(i=1; i<=n; i++)
{
    iIndex=COMINDEX(n,i,j);
    fX[i-1]=fXd[iIndex-1];
}

ind= Fun(iN,fX,fF);
if(ind!=0) goto ErrFun;
fFmind[j-1] = fF;
}
else          // Repeated input
{

// Recovering current simplex

k=0;
for(j=1; j<=n4; j++)

```

```

for(i=1; i<=n; i++)
{
k=k+1;
iIndex=COMINDEX(n,i,j);
fXd[iIndex-1] = fSave[k-1];
}

for(j=1; j<=n4; j++)
{
k=k+1;
ffmind[j-1] = fSave[k-1];
}
}

////////////////////////////////////
//
// The main cycle of movement
//

L_10: ;

// Determination of the best (jgood)
// and the worst (jbad) points of simplex

ffmax=ffmind[0];
jbad=1;
for(i=2; i<=n1; i++)
if(ffmind[i-1]>ffmax)
{
ffmax=ffmind[i-1];
jbad=i;
}

ffmin=ffmind[0];
jgood=1;
for(i=2; i<=n1; i++)
if(ffmind[i-1] < ffmin)
{
ffmin=ffmind[i-1];
jgood=i;
}

// Check - whether we are on the plane?
// If yes, then starting nodes will be considered on circle 1->n1.

if(ffmax == ffmin) jbad=int(fmod(double(iIter), double(n1)))+1;

// Reflection: shooting of the worst point over
// center of gravity - n2

for(i=1; i<=n; i++)
{
fW=0.f;
for(j=1; j<=n1; j++)
{
iIndex=COMINDEX(n,i,j);
fW += fXd[iIndex-1];
}

iIndexBad = COMINDEX(n,i,jbad);
iIndex2 = COMINDEX(n,i,n2);
iIndex3 = COMINDEX(n,i,n3);

fXd[iIndex2-1] = (fW-fXd[iIndexBad-1])/n;
fW = fXd[iIndex2-1] + ALPHA*(fXd[iIndex2-1]-fXd[iIndexBad-1]);
fXd[iIndex3-1]=MIN(MAX(fXl[i-1],fW),fXh[i-1]);
fX[i-1]=fXd[iIndex3-1];

```

```

}

ind= Fun(iN,fX,ff);
if(ind!=0) goto ErrFun;
ffmind[n3-1] = ff;

if(ffmind[n3-1]>ffmin) goto L_50 ;

// n3 - the best point of symplex.
// Extension: we continue the movement to the direction
// the worst--> center of gravity

for(i=1; i<=n; i++)
{
  iIndex2 = COMINDEX(n,i,n2);
  iIndex3 = COMINDEX(n,i,n3);
  iIndex4 = COMINDEX(n,i,n4);

  fW = fXd[iIndex2-1] + GAMMA*(fXd[iIndex3-1]-fXd[iIndex2-1]);
  fXd[iIndex4-1] = MIN(MAX(fXl[i-1],fW),fXh[i-1]);
  fX[i-1] = fXd[iIndex4-1];
}

ind= Fun(iN,fX,ff);
if(ind!=0) goto ErrFun;
ffmind[n4-1] = ff;

if(ffmind[n4-1]<=ffmind[n3-1]) goto L_110;
goto L_100;

// n3 -is not the best point of simplex. It is necessary
// to change the direction of movement.

// In order to do this it is necessary to change the worst step
// on the next step.

// Let's find the second max value. This value is
// the worst on the next step

L_50:

ffmax2=ffmin;
for(i=1; i<=n1; i++)
{
  if(i!=jbad) ffmax2=MAX(ffmax2,ffmind[i-1]);
}

if(ffmind[n3-1] < ffmax2) goto L_100;
if(ffmind[n3-1] >= ffmax) goto L_60;

for(i=1;i<=n; i++)
{
  iIndexBad = COMINDEX(n,i,jbad);
  iIndex3 = COMINDEX(n,i,n3);

  fXd[iIndexBad-1]=fXd[iIndex3-1];
}

ffmind[jbad-1]=ffmind[n3-1];

// n3 - is worse than, the second one on its value.
// It is necessary to get rid of concurrent!
// Condec: interpolation n3 between center of gravity
// and the best from 2 points: (jbad,n3)

L_60:

for(i=1; i<=n; i++)
{
  iIndexBad = COMINDEX(n,i,jbad);

```

```

iIndex2 = COMINDEX(n,i,n2);
iIndex3 = COMINDEX(n,i,n3);

fW=fXd[iIndex2-1] + BETA*(fXd[iIndexBad-1]-fXd[iIndex2-1]);
fXd[iIndex3-1] = MIN(MAX(fXl[i-1],fW),fXh[i-1]);
fX[i-1]=fXd[iIndex3-1];
}

ind= Fun(iN,fX,fF);
if(ind!=0) goto ErrFun;
fFmind[n3-1] = fF;

if(fFmind[n3-1]<fFmax2) goto L_100;

// Changing the worst point for correction of movement
// direction was successful.
// Reduction: condence of all symplex to the best point
// in 2 times. "We can't do anything more..."

for(j=1; j<=n1; j++)
{
for(i=1; i<=n; i++)
{
iIndexGood= COMINDEX(n,i,jgood);
iIndex = COMINDEX(n,i,j);

fW = fXd[iIndexGood-1]+0.5f*(fXd[iIndex-1]-fXd[iIndexGood-1]);
fXd[iIndex-1]=MIN(MAX(fXl[i-1],fW),fXh[i-1]);
fX[i-1]=fXd[iIndex-1];
}

ind= Fun(iN,fX,fF);
if(ind!=0) goto ErrFun;
fFmind[j-1] = fF;
}

goto L_130;

// Replacement of the worst point (jbad) on not the worst one - n3
L_100:

for(i=1; i<=n; i++)
{
iIndexBad = COMINDEX(n,i,jbad);
iIndex3 = COMINDEX(n,i,n3);

fXd[iIndexBad-1] = fXd[iIndex3-1];
}

fFmind[jbad-1]=fFmind[n3-1];
if(fFmind[n3-1]<=fFmin) jgood=n3;
goto L_130;

// Replacement of the worst point (jbad)
// on the best one(after extention)-n4
L_110:

for(i=1; i<=n; i++)
{
iIndexBad = COMINDEX(n,i,jbad);
iIndex4 = COMINDEX(n,i,n4);

fXd[iIndexBad-1]=fXd[iIndex4-1];
}

fFmind[jbad-1]=fFmind[n4-1];
jgood=n4;

```

```

// Finish of main cycle of movement !!!

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

// Check of finishing movement

L_130:

    for(i=1; i<=n; i++)
    {
        iIndexGood = COMINDEX(n,i,jgood);
        fXw[i-1] = fXd[iIndexGood-1];
    }

    fF = fFmind[jgood-1];

ind=View(iN,fXw,fF); // Registration of results
if(ind!=0) goto ErrFun;

ilter=ilter+1;

fEps=0.f;
for(j=1; j<=n1; j++)
{
    fW=0.f;
    for(i=1; i<=n; i++)
    {
        iIndex = COMINDEX(n,i,j);
        iIndex2 = COMINDEX(n,i,n2);
        if(fXh[i-1]==fXl[i-1]) continue;

        fDx= _max(0.000001f,fXh[j-1]-fXl[i-1]);
        fDx=(fXd[iIndex-1]-fXd[iIndex2-1])/fDx;
        fW=fW+fDx*fDx;
    }

    fW=(float)sqrt(double(fW));
    fEps=MAX(fEps,fW);
}

if((ilter<ilterMax)&&(fEps>fEpsMax)) goto L_10;

for(i=1; i<=n;i++)
{
    iIndexGood = COMINDEX(n,i,jgood);
    fX[i-1]=fXd[iIndexGood-1];
}

k=0;
for(j=1; j<=n4; j++)
for(i=1; i<=n; i++)
{
    k=k+1;
    iIndex=COMINDEX(n,i,j);
    fSave[k-1] = fXd[iIndex-1];
}

for(j=1; j<=n4; j++)
{
    k=k+1;
    fSave[k-1]=fFmind[j-1];
}

delete [] fXd;
delete [] fFmind;
delete [] fStep;
delete [] fXw;

return 0;

```

```

// Error in 'Fun' or 'View' subprograms

ErrFun: ;
delete [] fXd;
delete [] fFmind;
delete [] fStep;
delete [] fXw;

return -3;
}

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

long RandomSearch (int iN, float fXI[], float fXh[], float fX[],
                  float fProbMin,
                  float fEpsMax, int ilterMax, float& fEps, int& ilter,
                  long (*Fun) (int ,float[] ,float& ),
                  long (*View) (int ,float[] ,float ))

////////////////////////////////////
//
// Subroutine for function minimization by
// standard method of direct random search
// Category: multidimensional stochastic method
// Possibility: probable absolute minimum
//
//
// LITERATURE:
//
// PROGRAMS, THAT USE THIS ONE:
// PROGRAMS, USED IN THIS ONE: Fun, View, RunRandom, Random
//
// INPUT DATA:
// iN - dimension of problem
// fXI,fXh - lower and higher restriction,
// they are vectors by dimension 'n'
// iProbMin - the accesible level of probability for
// definition of extremum point neighbourhood
// fEpsMax - the limit for relative size of
// extremum point neighbourhood
// ilterMax - the limit number of points
//
// OUTPUT DATA:
// fX - arguments for minimum value of criteria function
// fEps - obtained relative size of
// extremum point neighbourhood
// ilter - calculated number of iteration
//
// RETURN DATA:
// 0 - All is OK!
// -1 - no memory
// -2 - invalid parameters
// -3 - error of input/output (other subprograms)
//
//-----
//
// Description of subprogram
//
// Fun -name of subprogram, which determines criteria function;
// ---
// long Fun (int iN, float fX[], float &fF)
// input parameters:
// iN - dimension of problem
// fX - vector of arguments
// output parameters:
// fF - value of criterion function
//
// View -name of virtual subprogram, providing visualization
// ---- of arguments and function after each step of searching

```

```

//
// long View(int iN, float fX[], float fF)
// input parameters:
//   iN - dimension of problem
//   fX - vector of arguments
//   fF - value of criterion function
//
//
//////////////////////////////////////////////////////////////////
{
int i,ind,iter;
float fF,fFmin;
float* fXw;

RunRandom(); // Run random series of random values

// Beginning of subprogram

if((iN<=0)||((fEpsMax<=0.f)||((fEpsMax>=1.0f)||((iIterMax<=1)||
(fProbMin<=0.f)||((fProbMin>=1.0f)) return -2;

if(!fXl) return -2;
if(!fXh) return -2;
if(!fX) return -2;

ind =0;
for(i=1; i<=iN; i++)
{
if(fXl[i-1]>fXh[i-1]) return -2;
if(fXl[i-1]==fXh[i-1]) ind=ind+1;
}
if(ind==iN) return -2;

fXw = new float [iN];
if( !fXw ) return -1;

for(i=1; i<=iN; i++)
fX[i-1] = fXl[i-1] + (fXh[i-1]-fXl[i-1])*Random();

ind=Fun(iN,fX,fF);
if(ind!=0) goto ErrFun;
ind=View(iN,fX,fF);
if(ind!=0) goto ErrFun;

fFmin=fF;

iIter = int(log(1.-fProbMin)/log(1.-fEpsMax)) + 1;
iIter = MIN(MAX(2,iIter),iIterMax);

fEps = 1.f - float( exp(log(1.-fProbMin)/double(iIter)) );

iter = 1;
if(iter >= iIter)goto Finish;

//-----

// Main cycle of calculation

for(iter=2; iter<=iIter; iter++)
{
for(i=1; i<=iN; i++)
fXw[i-1] = fXl[i-1] + (fXh[i-1]-fXl[i-1])*Random();

ind=Fun(iN,fXw,fF);
if(ind!=0) goto ErrFun;
ind=View(iN,fXw,fF);
if(ind!=0) goto ErrFun;
}
}

```

```

    if( fF<fFmin )
    {
        fFmin=fF;
        for(i=1; i<=iN; i++)
            fX[i-1] = fXw[i-1];
    }
}

//-----

// Final operations

Finish: ;

    delete [] fXw;
    return 0;

ErrFun: ;

    delete [] fXw;
    return -3;

}

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

long GoldSection (float fA0, float fB0, float fEpsMax, int ilterMax,
                 float& fA, float& fB, float& fEps, int& ilter,
                 long (*Fun) (int ,float[] ,float& ),
                 long (*View) (int ,float[] ,float ))

////////////////////////////////////
//
// Subroutine for function minimization by
// gold section method
// Category: one-dimensional method of cutting
// Possibility: absolute minimum of unimodal function
//
//
// LITERATURE:
//
// PROGRAMS, THAT USE THIS ONE:
// PROGRAMS, USED IN THIS ONE: Fun, View
//
// INPUT DATA:
// fA0 - initial left boundary
// fB0 - initial right boundary
// iltermax - the limit for number of iteration
// fEpsmax - the limit for interval of indefinision
//
// OUTPUT DATA:
// fA - final left boundary
// fB - final right boundary
// ilter - obtained number of iteration
// fEps - obtained interval of indefinision (fB-fA)
//
// RETURN DATA:
// 0 - All is OK!
// -1 - no memory
// -2 - invalid parameters
// -3 - error of input/output (other subprograms)
//
//-----
//
// Description of subprogram
//
// Fun -name of subprogram, which determines criteria function;
// ---
// long Fun (int iN, float fX[], float &fF)
// input parameters:

```



```

//      iN - dimension of problem
//      fX - vector of arguments
// output parameters:
//      fF - value of criterion function
//
// View -name of virtual subprogram, providing visualization
// ---- of arguments and function after each step of searching
//
// long View(int iN, float fX[], float fF)
// input parameters:
//      iN - dimension of problem
//      fX - vector of arguments
//      fF - value of criterion function
//
// Nota: here iN=1 !
//
///////////////////////////////////////////////////////////////////

{
int  ind,iter;
float fA1,fB1,fSba,fFA1,fFB1,fR;

// Control and initial conditions

if((fA0 >= fB0) || (fEpsMax < 0.f) || (iIterMax <= 2))
    return -2;

iter=0;
fR=float(sqrt(5.)) - 2.f;

fA=fA0;
fB=fB0;
fSba=fB-fA;

fA1 = fB-(float(sqrt(5.))-1.f)/2.f*fSba;
fB1 = fA+fB-fA1;

ind = Fun(1,&fA1,fFA1);
if(ind!=0) goto ErrFun;
ind = View(1,&fA1,fFA1);
if(ind!=0) goto ErrFun;

iter=iter+1;
if(iter>=iIterMax) goto L_10;

//-----

// Main cycle of calculation

L_3: ;
ind = Fun(1,&fB1,fFB1);
if(ind!=0) goto ErrFun;
ind = View(1,&fB1,fFB1);
if(ind!=0) goto ErrFun;

L_4: ;
iter=iter+1;
if(iter>=iIterMax) goto L_10;

// Comparison of internal points

if(fFA1 > fFB1) goto L_7;

// Segment is closer to point A

fB = fB1;
fB1 = fA1;
fFB1 = fFA1;
fSba = fB-fA;

```

```

if(fSba <= fEpsMax) goto L_10;

fA1 = fB1 - fR*fSba;

ind = Fun(1,&fA1,fA1);
if(ind!=0) goto ErrFun;
ind = View(1,&fA1,fA1);
if(ind!=0) goto ErrFun;
goto L_4;

// Segment is closer to point B

L_7: ;
fA = fA1;
fA1 = fB1;
fFA1 = fFB1;
fSba = fB-fA;
if(fSba <= fEpsMax) goto L_10;

fB1 = fA1 + fR*fSba;
goto L_3;

//-----

// Final operations

L_10: ;
fEps=fSba;
ilter=iter;
return 0;

ErrFun: ;
fEps=fSba;
ilter=iter;
return -3;
}

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

long StepByStep (float fXl, float fXh, float& fX,
                float fEpsMax, int ilterMax, float& fEps, int& ilter,
                long (*Fun) (int ,float[] ,float& ),
                long (*View) (int ,float[] ,float ),
                float fALPHA, float fBETA, float fDELTA)

////////////////////////////////////
//
// Subroutine for function minimization by
// ordinary step-by-step method
// Category: one-dimensional step-by-step method
// Possibility: local minimum
//
//
// LITERATURE: F.P.
//
// PROGRAMS, THAT USE THIS ONE:
// PROGRAMS, USED IN THIS ONE: Fun, View
//
// INPUT DATA:
// fXl - left boundary
// fXh - right boundary
// fX - initial approximation;
// iltermax - the limit for number of iteration
// fEpsmax - the limit for error
//
// OUTPUT DATA:
// fX - obtained result
// ilter - obtained number of iteration
// fEps - obtained error
//

```

```

// RETURN DATA:
// 0 - All is OK!
// -1 - no memory
// -2 - invalid parameters
// -3 - error of input/output (other subprograms)
//
//-----
//
// Parameters , being appointed
//
// ALPFA coefficient of extension for successful step
// BETA coefficient of compression for unseccessful step
// DELTA coefficient of relative size of step
// at the begining of calculation
//
// Standard values of method's parameters:
// ALPFA=1.5, BETA=0.5, DELTA=0.1
//
//-----
//
// Description of subprogram
//
// Fun -name of subprogram, which determines criteria function;
// ---
// long Fun (int iN, float fX[], float &fF)
// input parameters:
// iN - dimension of problem
// fX - vector of arguments
// output parameters:
// fF - value of criterian function
//
// View -name of virtual subprogram, providing visualization
// ---- of arguments and function after each step of searching
//
// long View(int iN, float fX[], float fF)
// input parameters:
// iN - dimension of problem
// fX - vector of arguments
// fF - value of criterian function
//
// Nota: here iN=1 !
//
///////////////////////////////////////////////////////////////////

{
int ind,iter;
float fX0,fX1,fF0,fF1,fStep0;
float ALPHA,BETA,DELTA;
int iLine,iSuccess;

// Control and initial conditions

if((fX1 >= fXh)|| (fEpsMax<0.f)|| (iIterMax<=1)) return -2;
if((fX<fX1)|| (fX>fXh)) return -2;

if((fALPHA<1.f)|| (fBETA<=0.f)|| (fBETA>=1.f)||
(fDELTA<=0.f)|| (fDELTA>=1.f)) return -2;

iter = 0;
iSuccess = 1;
iLine = 1;

ALPHA = fALPHA;
BETA = fBETA;
DELTA = fDELTA;

fStep0=(fXh-fX1)*DELTA;
fX0=fX;

ind = Fun(1,&fX0,fF0);

```

```

if(ind!=0) goto ErrFun;
ind = View(1,&fX0,ff0);
if(ind!=0) goto ErrFun;

iter=iter+1;

//----- Main cycle -----

L_10: ;

fX1 = fX0+fStep0*iLine;
fX1 = MIN(MAX(fX1,fX1),fXh);

ind = Fun(1,&fX1,ff1);
if(ind!=0) goto ErrFun;
ind = View(1,&fX1,ff1);
if(ind!=0) goto ErrFun;

// Check of stopping

iter=iter+1;

ilter = iter;
fEps = fStep0;
fX = fX1;
if(ilter >= ilterMax) return 0;
if(fEps <= fEpsMax) return 0;

// Comparison of last points

if(ff1 >= ff0) goto L_20; // Unsuccessful step

// Successful step

if(iSuccess==1)
{
fX0=fX1;
ff0=ff1;
fStep0=ALPHA*fStep0;
}
else
{
iSuccess=1;
fX0=fX1;
ff0=ff1;
fStep0=ALPHA*fStep0;
}
goto L_10;

// Unsuccessful step

L_20: ;

if(iSuccess==1)
{
iSuccess=0;
fStep0=BETA*fStep0;
}
else
{
iLine=-iLine;
fStep0=BETA*fStep0;
}
goto L_10;

// Final operation

ErrFun: ;
fEps=fStep0;
ilter=iter;

```

```
return -3;
}
////////////////////////////////////

void RunRandom()
// Initial random number for generator
{
// static UINT itime; <= 16 bit computer
static int itime;
time_t ltime;
struct tm* gmt;

time( &ltime ); // Standard time in sec [1,1,1970]
gmt = gmtime( &ltime );
itime = gmt->tm_min*29 + gmt->tm_sec*1081;
srand(itime); // Initial value for rand number generator;
}
```

