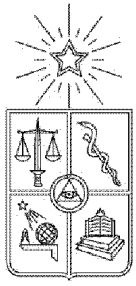

UNIVERSIDAD DE CHILE

Facultad de Ciencias Físicas y Matemáticas

Departamento de Ingeniería Matemática

DISEÑO ÓPTIMO DE EXPERIMENTOS PARA ESTIMAR EL CAMPO DE ESFUERZOS EN EL MACIZO ROCOSO EN TORNO AL FRENTE DE AVANCE DE UNA CAVIDAD MINERA

LUIS ALBERTO SAAVEDRA GODOY.

Profesor Guía	:	Sr. Felipe Álvarez Daziano.
Profesor Co-Guía	:	Sr. Juan Dávila Bonczos.
Profesor Integrante	:	Sr. Alejandro Jofré Cáceres.
Profesor Invitado	:	Sr. Julián Ortiz Cabrera.

MEMORIA PARA OPTAR AL TÍTULO DE INGENIERO CIVIL MATEMÁTICO.

SANTIAGO DE CHILE, 11 DE JULIO DE 2007

Resumen

El fenómeno de explosión de rocas y el colapso de galerías es de gran interés en el proceso de extracción del mineral. Debido a su importancia se han realizado estudios, tanto teóricos como experimentales, con el objetivo de predecir el comportamiento del macizo rocoso. En este marco se encuentra el proyecto "Explosión de Rocas en El Teniente", que ha dado como fruto un modelo de *Superposición de Campos Básicos* con el cual es posible estimar el estado tensional del macizo rocoso, y a través de esta estimación lograr inferir que configuración tensional es propensa a colapsos o explosión de roca.

El objetivo de este trabajo es realizar un estudio de los modelos de *Superposición de Campos Básicos*, a fin de lograr una comprensión que permita obtener un diseño de la red de estaciones de medición de manera "óptima". Para ello nos hemos impuesto el objetivo de justificar la elección de un modelo de *Superposición de Campos Básicos*, además nos hemos propuesto el presentar la Teoría de Diseño Óptimo de Experimentos como una herramienta que permite abordar el problema del diseño de la red de estaciones de medición, por este motivo fue necesario un objetivo más, el de implementar un Código de Diseño Óptimo, a fin de mostrar con pruebas numéricas la utilidad de esta teoría en la práctica. Las pruebas numéricas dan cuenta lo positivo de los resultados, en particular hemos resuelto el problema de añadir estaciones a una red de estaciones dada, y somos capases de "medir" la importancia de una estación para un diseño determinado, además hemos "comparado" el estimador encontrado con un estimador alternativo, propuesto en el proyecto, mostrando lo ventajoso de nuestro estimador. El criterio con el cual hemos medido y comparado es el criterio clásicamente utilizado en el diseño óptimo de experimentos, denominado D-Criterio.

Este trabajo es el punto de partida para el estudio de modelos de *Superposición de Campos Básicos* que incorporen ciertos fenómenos, como plasticidad o no homogeneidad del macizo, así como también modelos tridimensionales, ya que dentro de los desarrollos no fue necesaria la suposición de un modelo bidimensional.

Agradecimientos

Esta memoria pudo ser realizada gracias al apoyo de tres instituciones: el *Centro de Modelamiento Matemático* y el *Departamento de Planificación de El Teniente* quienes me motivaron y acogieron en su proyecto conjunto *Explosión de Rocas en El Teniente*. Asimismo, al *Departamento de Ingeniería Matemática* de la *Universidad de Chile*, que forjó en mi persona la disciplina y moral característica de una de las instituciones más nobles de nuestro país, la *Universidad de Chile*.

Mi más amplio agradecimiento para el Dr. Felipe Álvarez D., mi Profesor Guía, cuyo invaluable y generoso apoyo, motivación e interés hicieron posible la realización de esta memoria, así como por su confianza en mi trabajo.

También quisiera hacer patente mi agradecimiento al Dr. Juan Diego Dávila, mi profesor Co-Guía, cuya invaluable comprensión y apoyo fue decisivo a la hora de integrar la teoría de Diseño Óptimo de Experimentos a esta memoria, muchas gracias.

Así también, quiero agradecer al Dr. Alejandro, quien tuvo la gentileza de aceptar formar parte de la comisión. Así como también al Dr. Julián Ortiz, a quien además le debo un *sinnúmero* de correcciones que han contribuido al perfeccionamiento de mi trabajo.

Al Dr. Sergio Gaete B., quien ha motivado parte de mi trabajo y presentado un sinnúmero de aplicaciones de la Ingeniería Matemática, además de haberme acogido en el *Departamento de Planificación de El Teniente Codelco-Chile*, donde realicé mi última práctica profesional.

También quisiera expresar mi agradecimiento a aquellos maestros que me han servido de ejemplo, apoyo y fuente de energía con su actitud, comportamiento y consejo, un especial saludo al Dr. Roberto Cominetti C., quien ha sido ejemplo del júbilo de esta disciplina, al Dr. Carlos Conca R.,

quien ha sido ejemplo de rectitud y ha estado abierto en todo momento a la discusión de nuevas ideas, al Dr. Axel Osses A., quien ha sido un ejemplo de investigación con sus exhaustivos análisis bibliográficos realizados en cátedra, y es el culpable de mi creencia en que estos deberían ser obligatorios, al Dr. Marcos Kiwi K., quien ha mostrado constantemente aplicaciones en cátedra, haciendo de ella un lugar sorprendente, y en especial a todos aquellos profesores del *Departamento de Ingeniería Matemática* a quienes les agradezco por hacer de éste, un verdadero templo del saber, la disciplina y la moral.

Finalmente, quiero agradecer a aquellos compañeros que hicieron que esta etapa sea más agradable, en especial a: Mosquera, Lara, Pedro, Castel, Poyo, Edu, Ale Puente, Leo, la Caro y su Felipe, Krilin, etc; del DIM, a : Felipe Gonzáles, Francisco Jara, Felipe Macias, Francisco Silva, Alonso Silva, Diego Morán, Juan Campos, Aitor Aldunate, Felipe Olmos, Claudio Telha, Bolivar Díaz, Gonzalo Cisternas, Oscar Vasquez, Miguel Carrasco, David Gómez, y algún otro más que seguramente he olvidado.

Desde luego, llego al final de este proyecto gracias al apoyo que me otorgan y al cariño que me inspiran mis padres, al apoyo recibido de infancia por parte de mi hermano mayor Daniel, quien ha sido como un padre en parte de mi vida, al cariño que me inspiran mis hermanos menores Victor y Felipe, en especial este último, quien ha mostrado un creciente interés por las ciencias. También tengo siempre presentes a todos mis amigos, a mis maestros y a quienes siempre me han enseñado algo, quienes por cierto son muchos y no podría enumerar suficientemente.

A todos mi mayor reconocimiento y gratitud.

Índice general

Capítulo 1

Sobre la Memoria

1.1. Introducción

Desde hace muchos años la principal fuente de riqueza de Chile es el cobre. De ahí la importancia de que esta industria sea lo más productiva posible. Para lograr este objetivo, es necesario estudiar acuciosamente cada proceso que se realiza, desde la extracción del mineral, hasta su conversión en cátodos, a modo de optimizar cada uno de ellos reduciendo los costos de operación, la contaminación y la pérdida de materiales nobles. Este trabajo se orienta, en particular, a mejorar una parte del proceso de extracción que se realiza en la mina El Teniente.

En términos precisos, se trata de un aporte teórico y práctico a los modelos de *Superposición de Campos Básicos* utilizados en la estimación del tensor de esfuerzos del macizo rocoso, el cual está complementado tanto con pruebas como con un código escrito en PYTHON. La importancia de estimar los esfuerzos radica en la existencia, al menos en teoría, de configuraciones tensionales más propensas a sufrir colapsos y estallidos de roca, los cuales causan serios daños operacionales y ponen en riesgo al personal, provocando el cierre parcial o total del sector afectado.

1.2. Objetivos y organización del texto

Esta memoria tiene tres objetivos principales.

El primero de ellos es presentar una justificación matemática de la adopción de una *Superposición de Campos Básicos* para la estimación del estado tensional del *macizo rocoso*. De esto nos encargamos en el capítulo 3, hay que advertir, que durante toda la memoria hemos supuesto al *macizo rocoso* como un medio perfectamente elástico, homogéneo e isótropo, lo que corresponde a una primera aproximación del fenómeno.

El segundo objetivo es presentar la teoría de *Diseño Óptimo de Experimentos* (ODE), como una herramienta para la elaboración del diseño de la red de estaciones de medición del estado tensional del *macizo rocoso*. De esto nos encargamos en el capítulo 4. La principal dificultad, en este objetivo, es lograr "visualizar" el problema de diseño óptimo de la red de estaciones como un problema de diseño óptimo de experimentos, el cual, depende en forma directa de los *Campos Básicos* utilizados, y de forma indirecta del *macizo rocoso*, es decir, un cambio en los *Campos Básicos* utilizados puede cambiar completamente la configuración del diseño, mientras que un cambio en el estado tensional del *macizo rocoso* no cambia al diseño, pero podría afectar la elección de los *Campos Básicos* utilizados.

El último objetivo, es la implementación computacional de un *código de diseño óptimo* (ODC), con el cual poder realizar pruebas numéricas, para evaluar la utilidad de esta herramienta. A lo largo de toda la memoria hacemos referencia al sector *Esmeralda* de la mina *El Teniente*. De este sector extraemos tanto la geometría del *frente de avance* como la localización de las *estaciones*, ejemplo con el cual ejecutamos cada uno de los objetivos. En el capítulo 5, nos encargamos de explicar la implementación computacional junto con dar la API del *código de diseño óptimo* (ODC). En el capítulo 6, nos encargamos de realizar las pruebas numéricas que apuntan a mostrar la utilidad de la herramienta.

En el capítulo 2 presentamos el problema práctico y en el capítulo 7 las conclusiones y recomendaciones.

Capítulo 2

Presentación del problema

2.1. Introducción

El Teniente es la mina subterránea de cobre más grande del mundo.

Se encuentra ubicada aproximadamente a 44km al noreste de Rancagua en la Sexta Región de Chile y enclavada cerca de 2.500m sobre el nivel del mar, en plena cordillera de los Andes, alrededor de la latitud 70°20'W y longitud 34°04'S (ver figura 2.2).

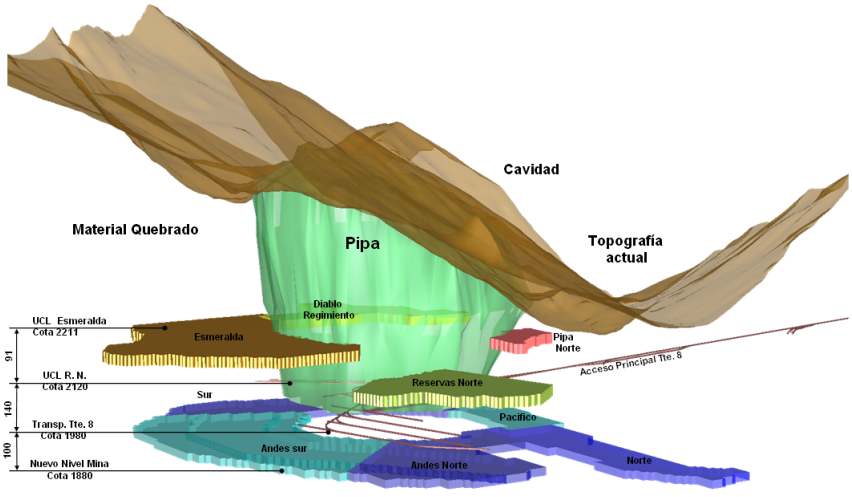

Figura 2.1: Descripción física de las distintas zonas de El Teniente.

Esta mina es una verdadera ciudad bajo la tierra. Dentro de ella, los casi 1.000 trabajadores

pueden encontrar casinos para comer, oficinas con computadores, teléfonos, atención médica, ascensores, bodegas, redes de luz eléctrica, agua potable, etc.

Como se puede imaginar, uno de los mayores problemas en una ciudad sumergida en la cordillera y destinada a la minería, es la actividad sísmica producida por la actividad minera, sobre todo si parte de ésta consiste en la *explotación por hundimiento* o *block caving*.

La *explotación por hundimiento* se basa en que tanto la roca mineralizada como la roca encajadora esté fracturada bajo condiciones más o menos controladas. La extracción del mineral crea una zona de hundimiento sobre la superficie, por encima del yacimiento. En consecuencia es muy importante el establecer un proceso de fracturación continuo y completo, ya que las cavidades subterráneas no soportadas presentan un riesgo elevado de desplomes repentinos, así como de eventos sísmicos importantes denominados *estallido de rocas* debido al daño que provocan, estos originan graves efectos a posterioridad en el funcionamiento de la explotación.

Las siguientes secciones están basadas en [19].

Figura 2.2: Ubicación de La Mina El Teniente.

2.2. Reseña histórica

De acuerdo a antecedentes históricos, este yacimiento de cobre fue descubierto por un oficial español fugitivo en los años 1800, y tiene sus primeros registros de explotación en el año 1819, iniciándose desde entonces la explotación del yacimiento en forma sistemática. El mineral de mayor ley era escogido a mano y transportado en animales. La zona más importante de explotación estaba en un sector denominado Fortuna.

Posteriormente, alrededor del año 1900, agotándose las reservas de mineral de alta ley, para continuar la producción, los propietarios contrataron los servicios del Ingeniero de Minas italiano Marcos Chiaponni, quien inspeccionó el yacimiento y recomendó la instalación de una planta concentradora. Debido a la falta de recursos de los propietarios, se buscó financiamiento en Europa pero no hubo una acogida favorable. En el año 1904 se interesó W. Braden de EE.UU. y conjuntamente con E. W. Nash formaron la primera compañía propietaria de El Teniente denominada "Braden Copper Company". En un principio se construyó un camino para carretas y una planta concentradora. El ferrocarril entre Rancagua y el campamento minero de Sewell fue construido entre 1906 y 1911, periodo en el cual los concentrados de cobre fueron enviados en carretas al pueblo de Graneros.

En 1908 el "Grupo Guggenheinn" tomó el control de la propiedad y aumentó la capacidad de la planta concentradora. En 1915 la "Kennecott Copper Corporation" adquirió los derechos. Cuando el único Concentrador estaba en Sewell se llegó a producir hasta 36.000 toneladas diarias de mineral proveniente de la mina. En abril de 1967, el Estado chileno adquiere a la "Braden Copper Company" el 51 % de la propiedad del yacimiento constituyéndose la "Sociedad Minera El Teniente". Bajo este convenio, a contar del año 1970 se materializó una gran expansión de la mina en conjunto con la construcción de una nueva planta concentradora en Colón aumentando la producción total a 63.000 toneladas de mineral por día. Según una Reforma Constitucional, el 11

de Julio de 1971 la Mina El Teniente pasa a ser propiedad de todos los chilenos. Posteriormente, el año 1976, se forma la "Corporación Nacional del Cobre de Chile" (CODELCO), de la cual pasa a conformar la División El Teniente. Hoy la producción de la mina supera las 100.000 toneladas de mineral por día, y el actual plan de expansión considera aumentar hasta 126.000 toneladas por día. Hasta el año 1975, la mina había extraído 500 millones de toneladas de mineral, y entre ese año y 1995 se extrajeron otras 500 millones de toneladas. El plan minero de los próximos 25 años estima extraer 1.400 millones de toneladas, siendo las reservas de cobre reconocidas de este yacimiento las mayores en el mundo.

2.3. Fundamentos de los Métodos de Explotación por Hundimiento

Dado que los métodos de explotación utilizados actualmente en la Mina El Teniente corresponden a técnicas de hundimiento gravitacional masivo de bloques o paneles, se explican a continuación los conceptos en que se basan dichas técnicas. El método de explotación de Hundimiento de Bloques o Paneles, en su forma más sencilla, puede definirse como el conjunto de operaciones mineras destinadas a cortar la base de sostenimiento de un bloque o panel de mineral, asegurándose que no queden puntos de apoyo, de tal forma que la base inferior de dicho bloque o panel se comporte como una viga (simplemente apoyada o empotrada en sus extremos), y la acción de las fuerzas externas, principalmente la gravitacional, produzcan una primera socavación y posteriormente el desplome completo del bloque o panel, de tal manera que los fragmentos de mineral generados debido al progreso del hundimiento en altura puedan ser manejados y transportados de acuerdo al diseño minero del sector productivo en cuestión.

2.3.1. Evolución de los Métodos de Explotación en la Mina El Teniente

La Mina El Teniente inició sus operaciones en forma industrial en el año 1906 y, desde entonces, se han empleado diferentes métodos de explotación, los que van desde el "Realce sobre Mineral" combinado con "Hundimiento de Pilares" (Shrinkage Stopping & Pillar Caving) hasta el "Hundimiento de Bloques" (Block Caving) actual, todos ellos utilizados en sectores productivos emplazados en mineral secundario[1]. Posteriormente, como consecuencia del cambio de las propiedades físico-mecánicas de la roca y de la profundización de los sectores productivos, la explotación de las reservas de mineral primario[2] ha significado mecanizar las operaciones mineras. Esta situación llevó a que el método de "Hundimiento de Bloques" utilizado en mineral secundario[3], cuya característica principal era el traspaso manual o semi-mecanizado, evolucionara hacia el "Hundimiento de Paneles" (Panel Caving) con traspaso mecanizado e incorporación continua de área hundida a la producción (frente de hundimiento dinámico).

2.3.2. Evolución Tecnológica de los Métodos de Hundimiento por Paneles en Mina El Teniente

En el método de "Hundimiento de Paneles", la alternativa mecanizada más ampliamente utilizada en la Mina es aquella con traspaso vía equipo "LHD" (Load-Haul-Dump) que se ha usado desde 1982 en el primer sector productivo que ha explotado mineral primario en El Teniente (caso del Teniente-4 Sur). La secuencia operacional de explotación aplicada en este sector corresponde

[1]Mineral secundario: zona secundaria. Corresponde a la parte que se ubica inmediatamente sobre la primaria, en que los minerales han sido alterados por efecto de la circulación de aguas de origen superficial, lo cual produce disolución de algunos minerales (por ejemplo, anhidrita) y enriquecimiento de los sulfuros, lo cual consiste en el aumento del contenido de cobre, pasando a constituir otro mineral (por ejemplo, transformación de calcopirita, con un $35\,\%$ de cobre, a calcosina, con un $80\,\%$ de cobre). Generalmente constituyen las zonas de mejores leyes en sulfuros de un yacimiento.

[2] Mineral primario: zona primaria. Corresponde a la parte profunda de un yacimiento en que se han preservado las características de su formación original, con minerales formados a grandes presiones y temperaturas, por lo que las rocas son en general duras e impermeables. En yacimientos de cobre, los minerales de mena característicos son los sulfuros bornita, calcopirita y pirita.

[3]Ver nota anterior.

a:

1) Desarrollo y Construcción de las Galerías del Nivel de Producción,

2) Socavación del Nivel de Hundimiento, y

3) Extracción del Mineral.

Este método es el denominado “Panel Caving Convencional”.

Sin embargo, basados en el conocimiento adquirido de la explotación de roca primaria con “Panel Caving Convencional” (21 años de aplicación en Mina El Teniente), se ha observado que el proceso de avance del frente de hundimiento es la principal fuente generadora de daños en las galerías de los niveles inferiores. De acuerdo a esta experiencia, se definió una variante al método de explotación “Panel Caving” denominada “Hundimiento Previo” (Pre-Undercut), la cual consiste básicamente en avanzar adelantado con el frente de socavación-hundimiento (ver figura 2.3) y llevar desfasada gran parte o toda la preparación en los niveles inferiores (detrás del frente y bajo el área socavada-hundida), reduciendo de esta manera el grado de daño en las galerías ubicadas bajo el Nivel de Socavación-Hundimiento y la ocurrencia de estallido de roca asociado al paso del frente. La aplicación a nivel industrial se realiza en el Sector Esmeralda de La Mina El Teniente. La secuencia operacional de explotación es:

1) Socavación del Nivel de Hundimiento,

2) Desarrollo y Construcción de las Galerías del Nivel de Producción, y

3) Extracción del Mineral.

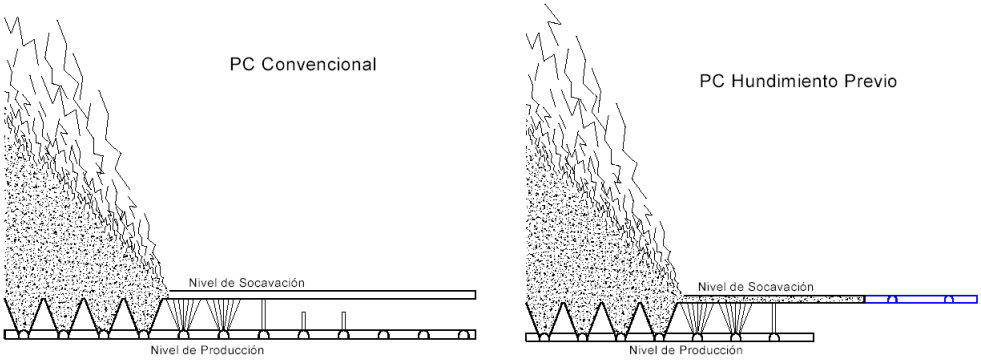

Figura 2.3: PC convencional v/s PC Hundimiento Previo.

2.3.3. Las principales inestabilidades geomecánicas de las variantes

Cada variante lleva consigo distancias permisibles, que permiten reducir el nivel de riesgo asociado a las operaciones unitarias requeridas para poner en producción el sector. Este riesgo es la resultante de un estado tensional variable en magnitud y dirección de los esfuerzos (zona de transición), y del avance del frente de hundimiento.

Las principales inestabilidades geomecánicas de las variantes son:

Colapso : Descenso paulatino de un área, generalmente en el nivel de producción, con daño observado en los pilares corona (Crown Pillar) y calle/Zanja, cuya expresión máxima es el cierre total de las galerías afectadas, reduciendo el área productiva debido a la pérdida de los accesos a ella.

Estallido de rocas : Inestabilidad en que se produce una pérdida de la continuidad del proceso productivo de la operación minera, provocado por la ruptura (evento sísmico) y proyección instantánea del macizo rocoso.

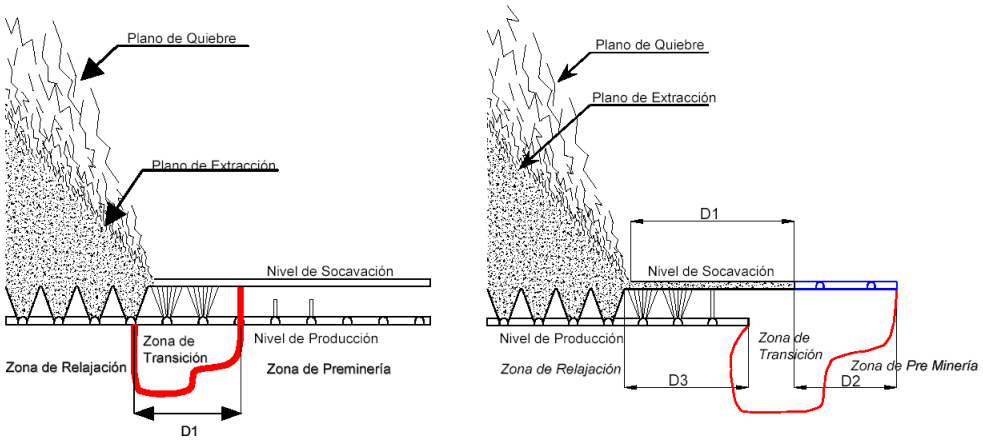

Figura 2.4: PC convencional v/s PC Hundimiento Previo.

2.3.4. Consecuencias Económicas de las Inestabilidades

Los efectos económicos originados por *las principales inestabilidades geomecánicas de las variantes* se pueden clasificar en dos grupos, los costos destinados a reparar las maquinarias, galerías afectadas, así como también indemnizaciones por daños a obreros, y los costos originados por la paralización, retrasos y modificaciones de los planes de extracción.

Es debido a los efectos económicos involucrados que se han invertido grandes sumas en investigar métodos que permitan un mejor control.

Actualmente, se utilizan varios procedimientos para determinar y predecir si se producirá un daño en un bloque dadas ciertas condiciones de extracción. Estos no logran determinar de manera efectiva *las inestabilidades* y, además, aprovechan sólo en parte la información disponible, tanto de los distintos eventos de *inestabilidades* como de las condiciones asociadas a estos eventos, la que han sido recopilada en bases de datos durante años de extracción. Es así como ha nacido el proyecto *"Estudio del comportamiento del macizo rocoso frente a variaciones geométricas de la cavidad y daño inducido en la roca"* cuyo objetivo general se puede resumir en *generar modelos que mejoren el conocimiento de los estallidos de rocas y eventualmente su predicción a mediano y corto plazo*,

el cual se desarrolla en forma conjunta por el **CMM** y el Departamento de Planificación de El Teniente.

2.4. Contexto de la memoria

Un punto de vista de estos modelos es el originado por el análisis de la mecánica del problema. Con esto, se ha logrado dividir el problema en dos partes, la primera es generar líneas de investigación sobre la configuración del estado tensional del macizo que origina *las inestabilidades* y fracturas de roca, y la segunda es la de generar líneas de investigación sobre métodos que permitan determinar el estado tensional del macizo.

Esta división está fundamentada en base a la hipótesis de que, conociendo el estado tensional del macizo se puede predecir cuando ocurrirán las inestabilidades.

Es precisamente en la etapa de investigación sobre métodos que permitan determinar el estado tensional donde se ubica esta memoria, ya que no solo se ha logrado la creación del grupo de investigación en ésta parte del proyecto, sino que también se han generado nuevos modelos matemáticos para la estimación del estado tensional del macizo, uno de ellos denominado de *Superposición de Campos Básicos y Estimación de Factores de Intensidad*, se ve potenciado con el trabajo presentado en esta memoria.

En la determinación del estado tensional del macizo, se invierten grandes sumas por concepto de la colocación de estaciones de monitoreo. La localización de estas estaciones generalmente se realiza en base a las necesidades inmediatas, sin ningún tipo de guía en la toma de decisión más que la experiencia e intuición. El trabajo presentado en esta memoria contribuye en gran medida a dar una guía sobre el diseño de localización de las estaciones de monitoreo, así como también a un mejor provecho de las que actualmente se encuentran.

Capítulo 3

Superposición de Campos Básicos y Factores de Intensidad

En el siguiente capítulo se presentan los fundamentos de la adopción de un modelo de *Superposición de Campos Básicos* para estimar el estado tensional en un dominio Ω. Luego se muestra una forma de aplicar esta superposición a un problema práctico, en una zona de la *División El Teniente*. Este ejemplo lo usaremos en adelante para visualizar los resultados de nuestro desarrollo.

3.1. Campos Básicos y Factores de Intensidad

3.1.1. El modelo de *Superposición de Campos Básicos*

En el modelo del macizo rocoso, que llamaremos Ω, se hacen los siguientes supuestos:

Geométrico : El primer gran supuesto, es la determinación de la geometría del dominio Ω, o bien, una simplificación de ésta.

Elasticidad lineal : El segundo supuesto, es que el dominio Ω, es un sistema elástico que cumple la ley de Hooke.

Constitutivo : El tercer supuesto es sobre el conocimiento del coeficiente de elasticidad $c_{jkml}(x)$ involucrado en la ley de Hooke, en cada punto del dominio Ω.

La justificación del supuesto *Geométrico*, se puede plantear en términos prácticos; es imposible describir la geometría, ya sea por la abundancia de detalles, por la falta de información, o por que no existe un límite establecido entre las distintas partes (cavidad, macizo y frente) sino más bien un paso gradual. Otro argumento en favor de una simplificación geométrica es el mayor interés en ciertas partes, y no en la geometría completa.

La justificación de haber escogido el supuesto de *Elasticidad Lineal*, como del supuesto *Constitutivo*, se puede plantear en términos académicos; en una primera etapa de la investigación se debe recurrir al modelo más simple, para luego ir avanzando hacia un modelo más elaborado.

Bajo estos tres supuestos el problema se puede plantear en términos generales como el de encontrar w tal que

$$\begin{array}{lll} A(x)w = f & \text{en} & \Omega \\ B(x)w = g & \text{sobre} & \partial\Omega \end{array} \tag{3.1}$$

donde se busca $w \in H^0$, A es un operador lineal de H^0 en H^1, y B es un operador lineal de H^0 en H^2, con H^0, H^1 y H^2, espacios de Hilbert adecuados.

Además se supone conocida la función f, generalmente proveniente de la gravedad y constante.

Con estos supuestos, para resolver totalmente el problema sólo basta conocer las condiciones en los límites g. Para ello encontramos primero una solución particular u_0 del problema (3.1) con $g = 0$, con el fin de reducirnos al problema homogéneo de encontrar u tal que

$$\begin{array}{lll} Au = 0 & \text{en} & \Omega \\ Bu = g & \text{sobre} & \partial\Omega \end{array} \tag{3.2}$$

Así, toda solución w del problema (3.1), se puede escribir como

$$w = u + u_0.$$

Una vez hecho esto, para determinar g, podemos tomar una base del espacio de Hilbert H^2, supongamos $g_1, g_2, g_3 \ldots$, y escribimos

$$g = \sum_{i=1}^{\infty} \alpha_i g_i.$$

Luego si llamamos u_i a la solución de (3.2), con g reemplazado por g_i, obtenemos que

$$u = \sum_{i=1}^{\infty} \alpha_i u_i.$$

Asimismo, si C es otro operador lineal de H^0 en otro espacio de Hilbert H^3, se obtiene que

$$C(x)u = \textstyle\sum_{i=1}^{\infty} \alpha_i C(x) u_i \quad \text{en} \quad \Omega.$$

En adelante llamaremos $\sigma = Cu$ y $\sigma_i = Cu_i$, con lo cual obtenemos el modelo lineal para σ:

$$\sigma = \sum_{i=1}^{\infty} \alpha_i \sigma_i. \tag{3.3}$$

Claramente lo anterior no depende de la condición de ortogonalidad de los g_i y sólo basta que generen el espacio H^2. En la práctica es así, y sólo se escoge un número finito p de ellos con lo cual se obtiene el modelo de *Superposición de Campos Básicos*:

$$\sigma = \sum_{i=1}^{p} \alpha_i \sigma_i + \varepsilon \tag{3.4}$$

donde ε es el error debido al hecho de que los g_i escogidos no generan todo el espacio H^2.

3.1.2. Los *Campos Básicos*

Llamamos *Campos Básicos*, a los $\sigma_1, \ldots, \sigma_p$ escogidos en *el modelo del macizo rocoso* (3.4). La elección de éstos se realiza de modo que los escogidos sean los "términos dominantes" del modelo (3.3) en el área de interés. Esto produce que el error ε, en el modelo (3.4) sea mínimo, de lo contrario habría que incluir nuevos términos σ_i al modelo (3.4).

3.1.3. Los Factores de Intensidad

Llamamos *factores de intensidad*, a los $\alpha_1, \ldots, \alpha_p$ en *el modelo del macizo rocoso* (3.4). La elección de éstos se realiza de modo que se minimice el error ε, en el modelo (3.4).

3.1.4. Modelo del error

Es de esperar que el error ε, en el modelo (3.4), sea pequeño en el area de interés debido a una buena elección de los $\sigma_1, \ldots, \sigma_p$, es por esto que se modela como una variable aleatoria de esperanza 0 y varianza ςI, recordando que σ es un tensor de nueve componentes en el modelo (3.4).

3.2. El ejemplo del “Frente de Avance”

Un ejemplo práctico donde se puede ocupar lo desarrollado en la sección anterior es cuando se quiere estimar el campo de esfuerzos en el macizo rocoso en torno al frente de avance de una cavidad minera. Este ejemplo fue desarrollado en el proyecto *“Estudio del comportamiento del macizo rocoso frente a variaciones geométricas de la cavidad y daño inducido en la roca”*, y a continuación lo describimos en forma muy breve, sin ahondar en detalles.

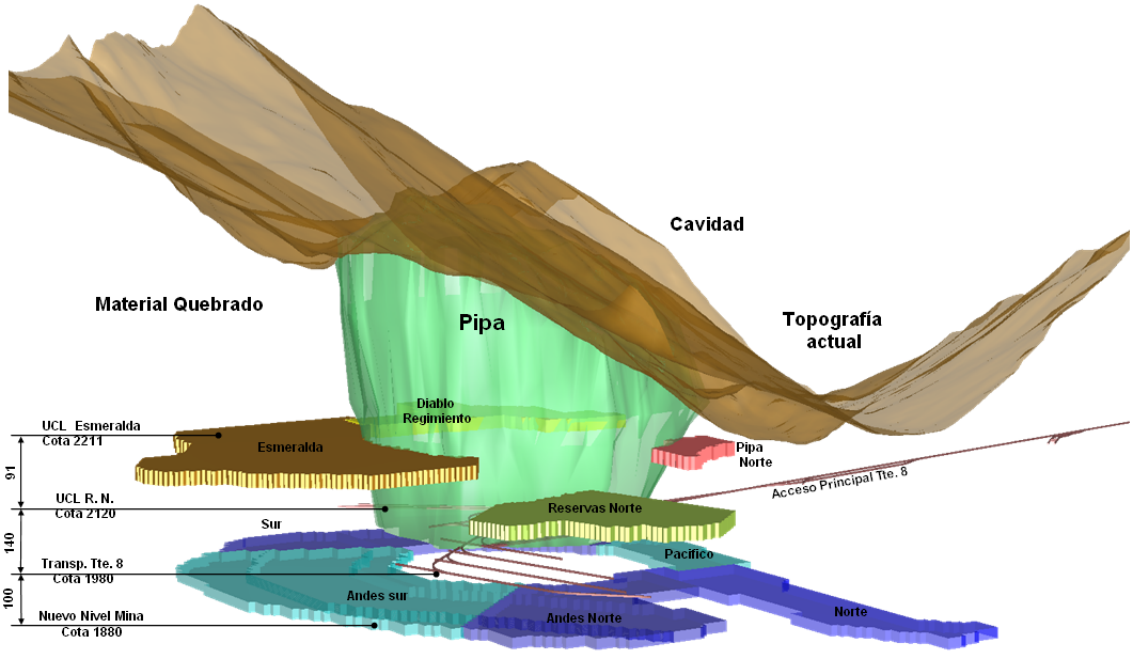

Figura 3.1: Imagen Virtual de La Mina El Teniente

3.2.1. Primera simplificación de la geometría: reducción 2D

Bajo ciertas condiciones, existe un plano de referencia con respecto al cual es posible describir el estado tensional al que se encuentra sometido un cuerpo, dando lugar a un modelo elástico bidimensional.

Para nuestro ejemplo escogeremos el yacimiento Esmeralda. Si tomamos una vista superior de éste, como la mostrada en la figura 3.2, nos podremos percatar que para una región suficientemente

cercana al *frente de avance*, no existe mucha diferencia entre el *frente de avance* real y uno que tenga un ancho

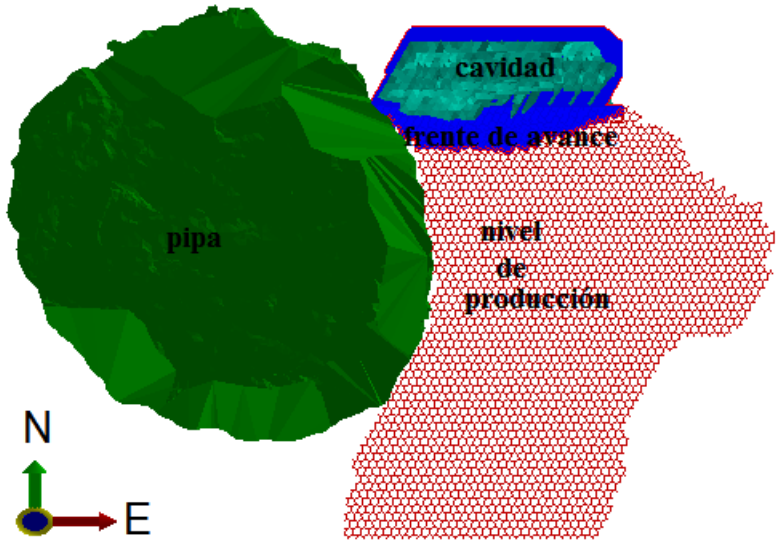

Figura 3.2: Vista superior del yacimiento Esmeralda tradicional.

infinito, esto debido a la gran diferencia que existe entre el ancho y alto del *frente de avance*. Se puede apreciar el alto del *frente de avance* en una vista lateral del yacimiento Esmeralda, tal como la mostrada en la figura 3.3.

Figura 3.3: Vista lateral del yacimiento Esmeralda tradicional.

Éstas son las razones que hacen suponer que el plano de referencia con respecto al cual es po-

sible describir el estado tensional en torno al *frente de avance* es un plano vertical y perpendicular al *frente de avance*. Esta simplificación es simplemente académica, se podría haber optado por considerar una banda, es decir un plano "con ancho", o alguna otra geometría tridimensional que represente mejor la situación real, pero se optó por la más sencilla.

3.2.2. Segunda simplificación de la geometría: localización

Ahora bien, una vez reducidos a una geometría bidimensional, se realiza un análisis de los posibles escenarios de esta geometría, donde se ha mostrado la relevancia de la geometría singular del *frente de avance* por sobre la geometría de la *cavidad minera* (en términos de esfuerzos). Podemos ver algunas de estas pruebas para el *esfuerzo de corte* en la figura 3.4, de donde, al sim-

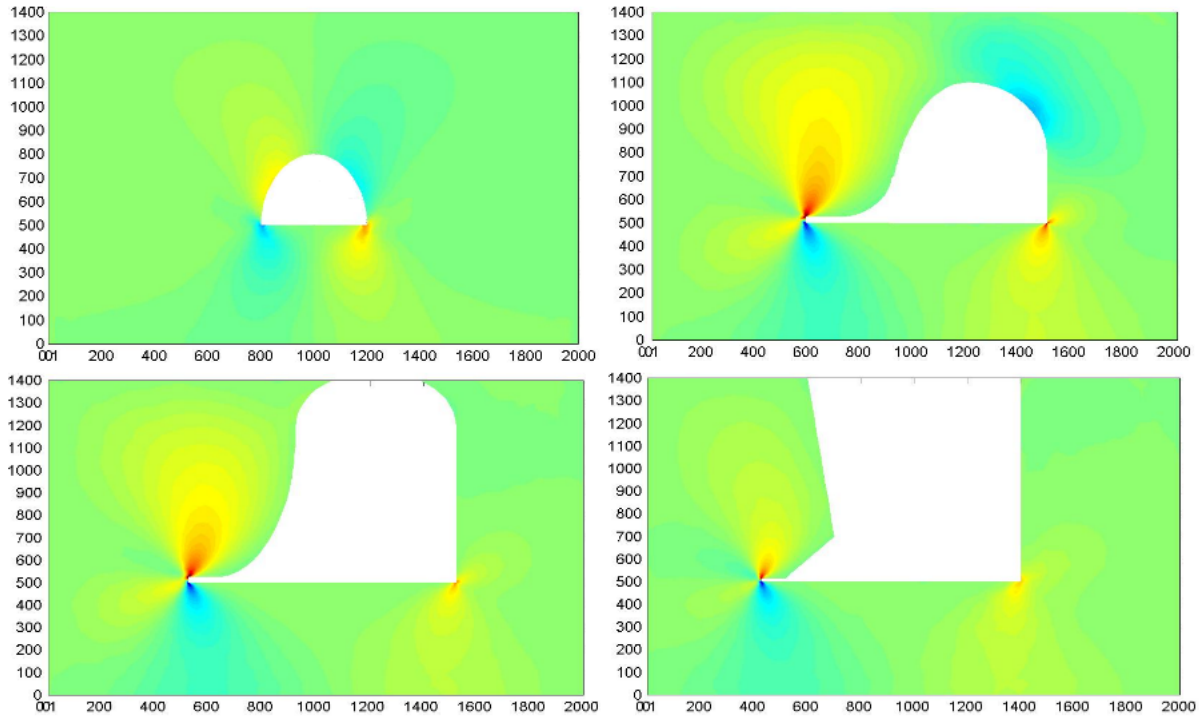

Figura 3.4: Pruebas numéricas, donde se muestra la relevancia cualitativa del *frente de avance* por sobre la *cavidad minera* (en términos de esfuerzos).

plificar la *cavidad minera*, la geometría, para un plano vertical que corte al *frente de avance*, resulta ser un dominio como el mostrado en la figura 3.5.

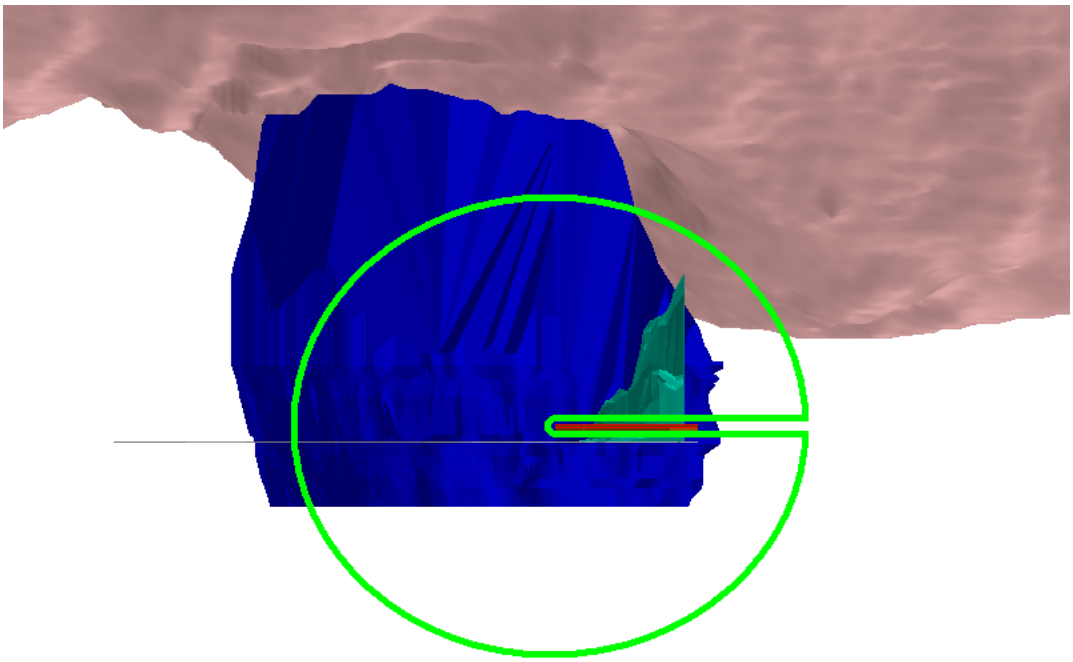

Figura 3.5: Geometría Simplificada.

Aunque se podría ocupar una geometría más representativa de la situación actual, tomando en cuenta por ejemplo la *cavidad minera*, en realidad lo que se busca es una geometría que sirva para distintas situaciones. Estas distintas situaciones para la *cavidad minera* tienen en común, cerca del *frente de avance*, la geometría simplificada que escogimos. Recordemos, por último, que el lugar en torno al *frente de avance* es donde se producen las mayores tensiones del macizo.

3.2.3. El origen de los Campos Básicos

En este ejemplo, para encontrar los Campos Básicos $\sigma_1, \ldots, \sigma_p$ del modelo (3.4), se estudió las características de la singularidad producida por el *frente de avance*. En ese estudio se encontraron tres soluciones $\mathbb{S}1$, $\mathbb{S}2$ y $\mathbb{S}3$, que llamamos *soluciones singulares*, y luego se añadieron otras seis soluciones $\mathbb{C}xx$, $\mathbb{C}xy$, $\mathbb{C}xz$, $\mathbb{C}yy$, $\mathbb{C}yz$ y $\mathbb{C}zz$, que llamaremos *soluciones adicionales*, y que en cierta medida reproducen el comportamiento constante lejos del *frente de avance*.

Construcción de los campos básicos

Recordemos que la construcción de los *campos básicos* se realiza a través de la resolución del problema siguiente:

$$\begin{array}{lll} Au = 0 & \text{en} & \Omega \\ Bu = g & \text{sobre} & \partial\Omega = \Gamma \end{array} \tag{3.5}$$

donde $[Au]_j = c_{jklm}\frac{\partial}{\partial x_k}\frac{\partial}{\partial x_m}u_l$ (suma sobre índices repetidos), con c_{jklm} los **coeficientes de elasticidad**,

$$[Bu]_j = \begin{cases} c_{jklm}\frac{\partial}{\partial x_m}u_l\eta_k & \text{en} \quad \Gamma_0 \cup \Gamma_1 \\ u_j & \text{sobre} \quad \Gamma_2 \end{cases}$$

con η vector normal exterior a Ω, y Γ_0, Γ_1 y Γ_2 como los mostrados en la figura 3.6

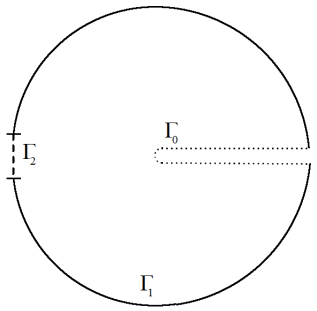

Figura 3.6: Descripción de la frontera de la geometría simplificada (dominio Ω).

Luego, para determinar un *campo básico*, lo único que falta es determinar el g correspondiente, el cual está dado de la siguiente forma

$$g = \begin{cases} 0 & \Gamma_0 \\ \sigma_g \cdot \eta & \Gamma_1 \\ 0 & \Gamma_2 \end{cases}$$

donde la condición en Γ_0 es debida a los supuestos de tracción nula dentro del *frente de avance*, y

la condición de Γ_2 es debida a los supuestos de desplazamientos nulos lejos del *frente de avance* (esta condición es necesaria para la unicidad y la medida de Γ_0 se puede escoger arbitrariamente pequeña pero no nula). Por lo tanto, para determinar la solución lo único que falta es definir el tensor σ_g. Luego, con la solución u definimos un *campo básico* σ en Ω de la siguiente forma:

$$[\sigma]_{jk} = c_{jklm}\frac{\partial}{\partial x_m}u_l$$

Soluciones singulares

Las *soluciones singulares* se originan del análisis asintótico de la ruptura de una fisura como las rupturas mostradas en la figura 3.7

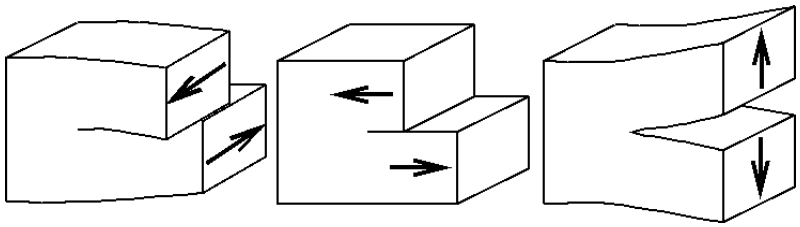

Figura 3.7: Modos de ruptura de una fisura.

donde cada una representa un "modo de ruptura" distinto, $\mathbb{S}1$ para el primero, $\mathbb{S}2$ para el segundo y $\mathbb{S}3$ para el último. Cada "modo de ruptura" define un campo de esfuerzo el cual es ocupado como σ_g. Para ver cómo se definen estos campos y los "modos de ruptura", remitimos al lector a [1] pág. 9-10.

Soluciones adicionales

Las *soluciones adicionales* se originan de la incorporación de los comportamientos constantes del *tensor de esfuerzos* a una gran distancia del *frente de avance*, una solución para cada uno de los posibles comportamientos constantes. Es así como podemos asociar

$$\mathbb{C}xx \text{ para } \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \quad \mathbb{C}xy \text{ para } \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \quad \mathbb{C}xz \text{ para } \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix}$$

$$\mathbb{C}yy \text{ para } \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \quad \mathbb{C}yz \text{ para } \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} \text{ y } \quad \mathbb{C}zz \text{ para } \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

donde las constantes asociadas son las que definen al tensor σ_g para determinar cada *campo básico*.

Capítulo 4

Diseño óptimo de experimentos

En el siguiente capítulo se presenta una introducción al diseño óptimo de experimentos (OED), la cual restringiremos sólo al caso clásico de modelos lineales estadísticos de la forma:

$$\mathbb{Y}(x) = \mathbb{X}(x) \cdot \alpha + \varepsilon(x) \tag{4.1}$$

donde $\mathbb{Y}(x)$ es la función objetivo n-dimensional, $\mathbb{X}(x)$ es una función vectorial $n \times p$-dimensional, α es el vector de parámetros (constante) p-dimensional, y $\varepsilon(x)$ es una variable aleatoria n-dimensional, de esperanza nula y varianza $\varsigma^2 I$.

En el ejemplo del capítulo anterior, vimos como una *Superposición de Campos Básicos* aproximaba al *tensor de tensiones*. En este capítulo, continuamos este ejemplo, y mostramos cómo el modelo de *Superposición de Campos Básicos* resulta ser de la misma forma que el modelo (4.1). Se presenta un método que permite dar, bajo cierto criterio, un estimador óptimo del vector de parámetros α, la mejor localización de las estaciones de medición de la función objetivo $\mathbb{Y}$, junto con la importancia relativa que tiene la información adquirida por cada una de ellas en el diseño, así como también una idea de la cantidad suficiente de mediciones a realizar.

Para una exposición de la teoría de *diseño óptimo de experimentos* en modelos lineales reco-

mendamos [4], en modelos no-lineales recomendamos [17].

4.1. Estimador de mínima varianza y estimador óptimo

Supongamos que realizamos k mediciones en los puntos[1] $x_1 \dots x_k$. Podremos entonces escribir el siguiente sistema:

$$\begin{pmatrix} \mathbb{Y}(x_1) \\ \vdots \\ \mathbb{Y}(x_k) \end{pmatrix} = \begin{pmatrix} \mathbb{X}(x_1) \\ \vdots \\ \mathbb{X}(x_k) \end{pmatrix} \cdot \alpha + \begin{pmatrix} \varepsilon(x_1) \\ \vdots \\ \varepsilon(x_k) \end{pmatrix} \tag{4.2}$$

Luego, si añadimos las hipótesis sobre los momentos de $\varepsilon(x)$

$$\mathbb{E}(\varepsilon(x)) = 0 \qquad \mathbb{V}(\varepsilon(x)) = \varsigma^2 I \tag{4.3}$$

sabemos, del teorema de Gauss-Markov, ver [4] (1.21), que un *estimador de mínima varianza* $\widehat{\alpha}(x_1, \dots, x_k)$ para α con la muestra $x_1, \dots, x_k$ está dado por

$$\widehat{\alpha}(x_1,\dots,x_k) = \left[\begin{pmatrix} \mathbb{X}(x_1) \\ \vdots \\ \mathbb{X}(x_k) \end{pmatrix}^t \begin{pmatrix} \mathbb{X}(x_1) \\ \vdots \\ \mathbb{X}(x_k) \end{pmatrix} \right]^{-1} \begin{pmatrix} \mathbb{X}(x_1) \\ \vdots \\ \mathbb{X}(x_k) \end{pmatrix}^t \begin{pmatrix} \mathbb{Y}(x_1) \\ \vdots \\ \mathbb{Y}(x_k) \end{pmatrix} \tag{4.4}$$

además, sabemos que la varianza de $\widehat{\alpha}(x_1, \dots, x_k)$ está dada por:

$$\mathbb{V}(\widehat{\alpha}(x_1, \dots, x_k)) = \varsigma^2 \left[\begin{pmatrix} \mathbb{X}(x_1) \\ \vdots \\ \mathbb{X}(x_k) \end{pmatrix}^t \begin{pmatrix} \mathbb{X}(x_1) \\ \vdots \\ \mathbb{X}(x_k) \end{pmatrix} \right]^{-1} \tag{4.5}$$

[1] No necesariamente distintos.

Como podemos observar[2], este *estimador de mínima varianza*, depende de los puntos $x_1, \ldots, x_k$ y resulta ser idéntico al encontrado cuando se minimiza el cuadrado de la norma de $(\varepsilon(x_1)^t, \ldots, \varepsilon(x_k)^t)^t$.

Recordemos que la igualdad (4.5) se puede obtener directamente de la igualdad (4.4) dado que la varianza de $({}_{\mathbb{Y}(x_1)}{}^t, \ldots, {}_{\mathbb{Y}(x_k)}{}^t)^t$ está dada por $\varsigma^2 I$, lo que se obtiene de la segunda igualdad de (4.3) y (4.2). Llamando A al término que multiplica a $({}_{\mathbb{Y}(x_1)}{}^t, \ldots, {}_{\mathbb{Y}(x_k)}{}^t)^t$ en la igualdad (4.4), la igualdad (4.5), se obtiene con la fórmula general

$$\mathbb{V}(\widehat{\alpha}(x_1, \ldots, x_k)) = A \cdot \mathbb{V}(({}_{\mathbb{Y}(x_1)}{}^t, \ldots, {}_{\mathbb{Y}(x_k)}{}^t)^t) \cdot A^t. \tag{4.6}$$

Por otra parte, también debemos recordar, que el orden considerado en el espacio de las matrices simétricas **Sym** y definidas no negativas **NND** (donde se encuentran las varianzas), **es parcial**, comúnmente llamado el *ordenamiento de Loewner*, y definido para $A, B \in$ **Sym** por

$$A \geq B \;\; \Leftrightarrow \;\; A - B \geq 0 \;\; \Leftrightarrow \;\; A - B \in \textbf{NND}.$$

Con esto podemos definir:

Definición 1 (Estimador Óptimo) *Llamamos* estimador óptimo*, a un estimador de mínima varianza* $\widehat{\alpha_{opt}}(x_1, \ldots, x_{k_{opt}})$*, tal que si* $\widehat{\alpha}(x_1, \ldots, x_k)$ *es otro estimador de mínima varianza, entonces,*

$$\mathbb{V}(\widehat{\alpha_{opt}}) \geq \mathbb{V}(\widehat{\alpha}).$$

Observación 1 .

[2]Para resolver las dudas con respecto a nociones básicas de estadística, como por ejemplo, la noción de *estimador*, la noción de *varianza de un estimador*, o la noción de *estimador de mínima varianza*, recomendamos al lector ocupar la enciclopedia virtual google, existen una infinidad de "sitios" donde puede encontrar esas definiciones.

- *Hay que recordar que el orden considerado es parcial.*

- k_{opt} *no tiene porqué ser igual a* k*.*

- *Un resultado sorprendente es la* ***no*** *existencia de un* estimador óptimo*, una demostración de esto se puede encontrar en [4](cap. 4.7.), donde se muestra como obtener a partir de un estimador, que pretende ser óptimo, otro con varianza estrictamente menor.*

Si bien, el hecho de que el *ordenamiento de Loewner* sea parcial, suponía cierta dificultad en la tarea de obtener un *estimador óptimo*, el hecho de la **no** existencia de un *estimador óptimo* destruye por completo las pretensiones de obtener uno. La dificultad esencial radica en el hecho de que el *ordenamiento de Loewner* sea parcial, por lo que nuestra tarea en adelante será la de proponer un **orden total** "compatible" con el *ordenamiento de Loewner*, y con el cual podamos obtener la optimalidad de nuestro estimador para este nuevo orden.

4.2. Diseño de experimentos

Primero empecemos por hacer una buena definición de los objetos que deseamos ordenar. Para ello, notemos que si en la muestra $x_1, \ldots, x_k$ sólo se encuentran m puntos distintos $p_1, \ldots, p_m$, obtenemos que tanto la expresión del estimador (4.4) como la de su varianza (4.5) se reducen a las siguientes

$$\widehat{\alpha}(p_1, \ldots, p_m, k_1, \ldots, k_m) = [\sum_{i=1}^{m} k_i \mathbb{X}(p_i)^t \cdot \mathbb{X}(p_i)]^{-1} \cdot [\textstyle\sum_{i=1}^{m} k_i \mathbb{X}(p_i)^t \cdot \mathbb{Y}(p_i)] \tag{4.7}$$

$$\mathbb{V}(\widehat{\alpha}(p_1, \ldots, p_m, k_1, \ldots, k_m)) = \varsigma^2 [\sum_{i=1}^{m} k_i \mathbb{X}(p_i)^t \cdot \mathbb{X}(p_i)]^{-1} \tag{4.8}$$

donde $k_1, \ldots, k_m \in \mathbb{N}$, son tales que $\sum k_i = k$ y representan el número de réplicas del punto p_i en la muestra $x_1, \ldots, x_k$.

De esta forma, resulta natural introducir

$$\mathcal{X}(p_1, \ldots, p_m) = \begin{pmatrix} \mathbb{X}(p_1) \\ \vdots \\ \mathbb{X}(p_m) \end{pmatrix} \tag{4.9}$$

$$\mathcal{Y}(p_1, \ldots, p_m) = \begin{pmatrix} \mathbb{Y}(p_1) \\ \vdots \\ \mathbb{Y}(p_m) \end{pmatrix} \tag{4.10}$$

$$\mathcal{W}(w_1, \ldots, w_m) = \begin{pmatrix} W_1 & 0 & \cdots & 0 \\ 0 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & W_m \end{pmatrix} \tag{4.11}$$

donde $w_i = k_i/k$ y $W_i = w_i I_n$ (con I_n la identidad en $\mathbb{R}^n$ y n la dimensión de $\mathbb{Y}$). Con los cuales podemos escribir

$$\widehat{\alpha}(p_1, \ldots, p_m, w_1, \ldots, w_m) = (\mathcal{X}^t \mathcal{W} \mathcal{X})^{-1} \cdot (\mathcal{X}^t \mathcal{W} \mathcal{Y}) \tag{4.12}$$

$$\mathbb{V}(\widehat{\alpha}(p_1, \ldots, p_m, w_1, \ldots, w_m)) = k^{-1} \varsigma^2 (\mathcal{X}^t \mathcal{W} \mathcal{X})^{-1} \tag{4.13}$$

Es interesante, una vez escritas las ecuaciones (4.12) y (4.13) obtenidas de forma independiente desde (4.4) y (4.5) respectivamente, verificar que (4.13) se puede deducir de (4.12) a través de la fórmula general (4.6) (convenientemente modificada). Para hacer tal verificación sería necesario saber la varianza $\mathcal{V}$ de[3] $\mathcal{Y}$, que justamente es lo que no sabemos, pero como tal verificación debe

[3]Esta variable depende explícitamente de $p_1, \ldots, p_m$ e implícitamente de $k_1, \ldots, k_m$.

ser verdadera, podemos escribir

$$k^{-1}\varsigma^2(\mathcal{X}^t\mathcal{W}\mathcal{X})^{-1} = (\mathcal{X}^t\mathcal{W}\mathcal{X})^{-1}(\mathcal{X}^t\mathcal{W})\mathcal{V}(\mathcal{W}\mathcal{X})(\mathcal{X}^t\mathcal{W}\mathcal{X})^{-1}$$

lo que con la sola hipótesis de que $\mathcal{X}$ tenga rango completo[4], nos conduce a la igualdad

$$\mathcal{V} = k^{-1}\varsigma^2\mathcal{W}^{-1}.$$

que es una matriz diagonal, con términos de la forma

$$\frac{\varsigma^2}{k_i}$$

en su diagonal. La confusión que esto podría causar, es el razonamiento erróneo; "como la coordenada i-ésima de $\mathcal{Y}$ es $\mathbb{Y}_{(p_i)}$, entonces la varianza de $\mathbb{Y}_{(p_i)}$ tendría que ser $\frac{\varsigma^2}{k_i}$", lo que es obviamente falso, ya que sabemos que esta última varianza es ς^2. El error del razonamiento es pensar que la coordenada i-ésima de $\mathcal{Y}$ es $\mathbb{Y}_{(p_i)}$ ¡a secas!, la verdad es que la coordenada i-ésima de $\mathcal{Y}$ se podría interpretar mejor como

$$\mathbb{Y}_{(p_i)} \text{ "preguntada" } k_i \text{ veces.}$$

lo malo de esta interpretación es que nos entrega k_i valores, y no un valor único como $\mathbb{Y}_{(p_i)}$. Una interpretación con un único valor, de estos k_i valores, es su promedio, que como sabemos tiene justamente varianza $\frac{\varsigma^2}{k_i}$, lo que concuerda con nuestro resultado. La interpretación correcta para k_i sería "el número de veces que hay que realizar la muestra de $\mathbb{Y}_{(p_i)}$", o en su defecto $w_i = k_i/k$ "el porcentaje de importancia, que tiene la muestra $\mathbb{Y}_{(p_i)}$, en el estimador $\widehat{\alpha}$". Esta última interpretación

[4]Lo que se supone desde que se considera $\mathcal{X}^t\mathcal{W}\mathcal{X}$ invertible.

es independiente del número de muestras k, lo que nos conduce al caso ideal en el que podamos realizar una muestra infinita, con la anulación de la varianza (4.13) del estimador. Sin embargo, como quiera que se anule la varianza, habrán formas más rápidas que otras, recordar esto nos servirá más adelante.

Como podemos darnos cuenta, no es fácil obtener una interpretación de la variable $\mathcal{Y}$ ya que ella depende implícitamente de $k_1, \ldots, k_m$. Esta dificultad contrasta con la claridad de la igualdad (4.13), ya que en la variable $\mathcal{X}$ no existe tal ambigüedad; efectivamente la coordenada i-ésima de $\mathcal{X}$ es $\mathbb{X}(p_i)$.

A continuación haremos algunas definiciones que nos ayudarán a formalizar las ideas antes explicadas.

Definición 2 (Diseño de tamaño k) *Decimos que*

$$\tau = \begin{pmatrix} p_1 & w_1 \\ \vdots & \vdots \\ p_m & w_m \end{pmatrix} \tag{4.14}$$

*es un **diseño de tamaño** k, si $p_1, \ldots, p_m$ son puntos distintos, con $m \leq k$, existen $k_1, \ldots, k_m \in \mathbb{N}$, tales que $k = \sum k_i$ y $w_i = k_i/k$. Al conjunto de todos los diseños de tamaño k lo denotamos $\mathcal{T}_k$.*

Lo primero que nos percatamos es que en la varianza dada por (4.13) de todos los diseños de tamaño k siempre es un múltiplo de $k^{-1}\varsigma^2$, con lo cual, resulta natural introducir, la *matriz de información* del diseño τ, dada por

$$\mathcal{M}(\tau) = \mathcal{X}^t \mathcal{W} \mathcal{X} \tag{4.15}$$

Con esto se tiene la siguiente equivalencia para dos diseños τ_1, τ_2 en $\mathcal{T}_k$:

$$\mathbb{V}(\widehat{\alpha}(\tau_1)) \geq \mathbb{V}(\widehat{\alpha}(\tau_2)) \Leftrightarrow \mathcal{M}(\tau_1) \leq \mathcal{M}(\tau_2) \tag{4.16}$$

lo que además de comparar sus valores, compara la "velocidad" con la que se anula la varianza al crecer la muestra $k \to \infty$, en términos del tamaño de $\mathcal{M}(\tau)$.

Ahora bien, si $\tau_1 \in \mathcal{T}_{k_1}$ y $\tau_2 \in \mathcal{T}_{k_2}$ con $k_1 \neq k_2$, nos gustaría poder compararlos a través de sus respectivas *matriz de información* (si tales matrices son comparables). Para ello nos damos cuenta que, para todo par k_1, k_2, siempre existe un k_G suficientemente grande tal que $\mathcal{T}_{k_i} \subset \mathcal{T}_{k_G}$ para $i = 1, 2$; en efecto, basta escoger $k_G = k_1 \cdot k_2$, con lo cual, $\tau_i \in \mathcal{T}_{k_G}$ para $i = 1, 2$, resulta natural entonces la siguiente definición, que formaliza el caso ideal en que podemos tomar una muestra infinita $k \to \infty$

Definición 3 (Diseño de experimentos) *Decimos que*

$$\tau = \begin{pmatrix} p_1 & w_1 \\ \vdots & \vdots \\ p_m & w_m \end{pmatrix} \tag{4.17}$$

*es un **diseño de experimentos**, si $m \in \mathbb{N}$, $p_1, \ldots, p_m$ son distintos, $w_i \geq 0$ y $\sum w_i = 1$. Al conjunto de todos los diseños lo denotamos $\mathcal{T}$.*

Observamos que para todo k, $\mathcal{T}_k \subset \mathcal{T}$, y que la comparación de diseños se hace en base a su **matriz de información**. Así, un diseño τ_1 será mejor que otro τ_2 si, $\mathcal{M}(\tau_1) \geq \mathcal{M}(\tau_2)$. Nuevamente podríamos ser tentados a buscar un "diseño óptimo" para el *ordenamiento de Loewner*, lamentablemente tendríamos los mismos problemas que tuvimos al buscar un *estimador óptimo*. Por lo cual, en la siguiente sección presentaremos el *orden total* "clásicamente utilizado" compatible con el *ordenamiento de Loewner*.

4.3. Diseño D-óptimo

Debido a las dificultades que se presentan a la hora de buscar óptimos en un orden parcial, digamos $\preceq$, es necesario introducir algún orden total, digamos $\leq$, que sea compatible con el orden parcial $\preceq$, en el sentido de que, por una parte

$$\text{si } A \preceq B \text{ entonces } A \leq B$$

y por otra parte

$$\text{si } A \leq B \text{ y } A, B \text{ son comparables con } \preceq \text{, entonces } A \preceq B.$$

Esta propiedad, es llamada *la propiedad de monotonicidad*. Asimismo, una **función**[5] ϕ en un subconjunto $S \subset \mathbb{R}^{n \times n}$, es *monotónica* si el orden que induce $\leq_\phi$ es total y tiene la propiedad de monotonicidad. A los diseños τ_{opt} tales que $\tau \leq_\phi \tau_{opt}$ para todo $\tau \in \mathcal{T}$ los llamamos diseños ϕ-óptimos.

Observemos que cualquier función monotónica podría servir para optimizar nuestro diseño, y que el decidirnos por una u otra puede conducir a diferentes resultados que podrían ser importantes (desde el punto de de vista de la interpretación del valor de la función ϕ).

La función "clásicamente utilizada" para optimizar el diseño es el determinante de la matriz de información. A estos diseños (eventualmente único) τ_D los llamamos *diseños D-óptimos*:

$$\tau_D \in \arg \max_{\tau \in \mathcal{T}} \det(\mathcal{M}(\tau))$$

[5]Recordemos que las funciones son a valores reales.

Con esto se logra que el determinante de la matriz de varianza para el estimador $\widehat{\alpha}(\tau)$ sea mínimo, y como el determinante es monotónico, se logra minimizar la varianza del estimador.

4.4. El problema a resolver

El problema a resolver se puede plantear como el siguiente:

$$\begin{array}{ll} \text{máx} & \mathrm{JM}(\tau) \\ s.a. & p_i \in \Lambda_i \\ & w_i \geq 0 \\ & \sum_i w_i = 1 \end{array} \tag{4.18}$$

donde se tiene que $\Lambda_i \subset \Omega$ y JM es el criterio a utilizar, que para nosotros resulta ser el determinante de la matriz de información (criterio clásico):

$$\mathrm{JM}(\tau) = \det(\mathcal{M}(\tau))$$

recordemos además la forma de un diseño

$$\tau = \begin{pmatrix} p_1 & w_1 \\ \vdots & \vdots \\ p_m & w_m \end{pmatrix}$$

Un problema práctico es encontrar un m suficientemente pequeño con el cual empezar la búsqueda en nuestro problema, ya que, resulta mucho más simple ir quitando puntos (reduciendo m en τ). Como se puede esperar, en algoritmos de esta naturaleza, es muy probable que un diseño con $m = 2$ tenga un criterio superior al encontrado partiendo con $m = 1$, y así sucesivamente, sin embargo, un resultado sorprendente es que se sabe que, en el caso escalar $n = 1$, siempre existe un D-óptimo con a lo más $\lceil p(p+1)/2 \rceil$ puntos (p es el número de parámetros), es decir que solo necesitamos $\lceil p(p+1)/2 \rceil$ puntos para comenzar la búsqueda, y que al partir con un m

mayor no lograremos sobrepasar al encontrado. Y en el caso vectorial $n > 1$, se puede comenzar la búsqueda con m igual a $\lceil p_m(p_m+1)/2 \rceil$, donde p_m es la dimensión del espacio generado por $\{\mathbb{X}(x); x \in \cup_i \Lambda_i\}$.

Vamos a examinar ahora el ejemplo que hemos ido desarrollando en los capítulos anteriores.

4.5. El ejemplo del "Frente de Avance"

Primero, la función objetivo $\mathbb{Y}$ está dada por

$$\mathbb{Y}(p_i) = \begin{pmatrix} \sigma_{xx} \\ \sigma_{xy} \\ \sigma_{xz} \\ \sigma_{yy} \\ \sigma_{yz} \\ \sigma_{zz} \end{pmatrix} \tag{4.19}$$

donde $\sigma_{xx}, \ldots, \sigma_{zz}$ representa el tensor de esfuerzos del macizo rocoso σ escrito en notación columna, que representamos con el modelo lineal (3.4). Ahora bien, cada σ_i en el modelo lineal (3.4), la representamos como una columna de la función $\mathbb{X}$, que está dada por

$$\mathbb{X}(p_i) = \begin{pmatrix} \mathbb{S}1_{xx} & \mathbb{S}2_{xx} & \mathbb{S}3_{xx} & \mathbb{C}xx_{xx} & \mathbb{C}xy_{xx} & \mathbb{C}xz_{xx} & \mathbb{C}yy_{xx} & \mathbb{C}yz_{xx} & \mathbb{C}zz_{xx} \\ \mathbb{S}1_{xy} & \mathbb{S}2_{xy} & \mathbb{S}3_{xy} & \mathbb{C}xx_{xy} & \mathbb{C}xy_{xy} & \mathbb{C}xz_{xy} & \mathbb{C}yy_{xy} & \mathbb{C}yz_{xy} & \mathbb{C}zz_{xy} \\ \mathbb{S}1_{xz} & \mathbb{S}2_{xz} & \mathbb{S}3_{xz} & \mathbb{C}xx_{xz} & \mathbb{C}xy_{xz} & \mathbb{C}xz_{xz} & \mathbb{C}yy_{xz} & \mathbb{C}yz_{xz} & \mathbb{C}zz_{xz} \\ \mathbb{S}1_{yy} & \mathbb{S}2_{yy} & \mathbb{S}3_{yy} & \mathbb{C}xx_{yy} & \mathbb{C}xy_{yy} & \mathbb{C}xz_{yy} & \mathbb{C}yy_{yy} & \mathbb{C}yz_{yy} & \mathbb{C}zz_{yy} \\ \mathbb{S}1_{yz} & \mathbb{S}2_{yz} & \mathbb{S}3_{yz} & \mathbb{C}xx_{yz} & \mathbb{C}xy_{yz} & \mathbb{C}xz_{yz} & \mathbb{C}yy_{yz} & \mathbb{C}yz_{yz} & \mathbb{C}zz_{yz} \\ \mathbb{S}1_{zz} & \mathbb{S}2_{zz} & \mathbb{S}3_{zz} & \mathbb{C}xx_{zz} & \mathbb{C}xy_{zz} & \mathbb{C}xz_{zz} & \mathbb{C}yy_{zz} & \mathbb{C}yz_{zz} & \mathbb{C}zz_{zz} \end{pmatrix} \tag{4.20}$$

donde se puede apreciar que $\mathbb{S}1$, $\mathbb{S}2$, $\mathbb{S}3$, $\mathbb{C}xx$, $\mathbb{C}xy$, $\mathbb{C}xz$, $\mathbb{C}yy$, $\mathbb{C}yz$ y $\mathbb{C}zz$ están representados como columnas, al igual que σ. Bien entendida esta notación, se puede escribir:

$$\begin{aligned} \mathbb{Y}(p_i) &= \sigma(p_i) \\ \mathbb{X}(p_i) &= \begin{pmatrix} \mathbb{S}1 & \mathbb{S}2 & \mathbb{S}3 & \mathbb{C}xx & \mathbb{C}xy & \mathbb{C}xz & \mathbb{C}yy & \mathbb{C}yz & \mathbb{C}zz \end{pmatrix} \end{aligned} \tag{4.21}$$

Por último, el vector de parámetros α resulta estar compuesto por los *factores de intensidad de esfuerzos*, con lo cual sólo nos queda resolver el problema (4.18).

Capítulo 5

Implementación computacional

En una primera parte se hace la distinción entre dos clases importantes de problemas en la resolución de (4.18). Estas diferencias concluyen en la adopción de distintos tipos de algoritmos para cada uno. Para finalizar, se explica la implementación computacional del código de diseño óptimo **ODC**, dando su correspondiente API.

5.1. Dos Clases de Problemas

Recordemos la forma general (4.18) del problema que queremos resolver,

$$\begin{array}{ll} \text{máx} & \mathrm{JM}(\tau) \\ s.a. & p_i \in \Lambda_i \\ & w_i \geq 0 \\ & \sum_{i=1}^{m} w_i = 1 \end{array} \tag{5.1}$$

Vamos ahora a examinar los distintos comportamientos de JM con respecto a cada tipo de variable; tenemos por una parte las variables espaciales p_i, y por otra, los pesos w_i. Los diferentes comportamientos provienen de la fórmula de JM,

$$\mathrm{JM}(\tau) = \det\left(\sum_{i=1}^{m} w_i \mathbb{X}(p_i)^t \mathbb{X}(p_i)\right), \tag{5.2}$$

donde se puede apreciar que las propiedades de diferenciabilidad con respecto a los p_i dependen de la función $\mathbb{x}$ y del dominio Λ_i, mientras que las propiedades de diferenciabilidad con respecto a los pesos w_i está asegurada.

A continuación, no haremos hipótesis de regularidad sobre las funciones $\mathbb{x}$, que bien podrían ser discontinuas, y nos concentraremos en explotar las propiedades de regularidad de JM con respecto a los pesos w_i.

Con este objetivo, distinguiremos entre los siguientes casos:

Problema de los Pesos Óptimos : En este problema, Λ_i (para cada i) se han reducido al punto p_i, de modo que las posiciones de las estaciones están fijas.

Problema de Localizar Estaciones : En este problema, tenemos un conjunto de índices I cuyos Λ_i se han reducido al punto p_i para cada $i \in I$, y otro conjunto de índices J cuyos Λ_j son idénticos y los llamaremos Λ para todo $j \in J$.

Es claro que en el **Problema de Localizar Estaciones**, si el conjunto de índices J es vacío, se reduce al **Problema de los Pesos Óptimos**. También permitiremos que el conjunto I pueda ser vacío.

5.2. El Problema de los Pesos Óptimos

En este caso, el problema general (5.1) se reduce al siguiente:

$$\begin{array}{ll} \text{máx} & \det\left(\sum_{i=1}^{m} w_i M_i\right) \\ s.a. & w = (w_1, \ldots, w_m) \in K_m \end{array} \tag{5.3}$$

donde $K_m := \{w = (w_1, \ldots, w_m); \sum_{i=1}^{m} w_i = 1, w_i \geq 0\}$ y las matrices constantes M_i están dadas por,

$$M_i = \mathbb{X}_{(p_i)}{}^t \mathbb{X}_{(p_i)}.$$

Notemos primeramente, que si para todo $w \in K_m$ se tiene que

$$\det\left(\sum_{i=1}^{m} w_i M_i\right) = 0$$

entonces, claramente el problema (5.3) resulta absurdo, por lo que descartaremos este caso.

Transformemos ahora este problema a un problema con existencia y unicidad. Para ello, recordamos que la función $\log : [0, \infty) \to [-\infty, \infty)$, definida en 0 por $\log(0) = -\infty$, es una función estrictamente creciente, por lo tanto preservadora del orden. Ahora bien, dado que las matrices M_i son semidefinidas positivas, nuestra función objetivo es positiva, por lo que podemos reemplazarla por $\log(\det\left(\sum_{i=1}^{m} w_i M_i\right))$ sin alterar el problema. Escribimos entonces el siguiente problema equivalente:

$$\begin{array}{ll} \text{mín} & -\log(\det\left(\sum_{i=1}^{m} w_i M_i\right)) \\ s.a. & w \in K_m \end{array} \tag{5.4}$$

Ahora bien, se puede demostrar (ver [20] pag.81) que la función $M \to -\log(\det(M))$ resulta ser convexa en el cono de las matrices simétricas definidas no negativas y estrictamente convexa en el cono de las matrices definidas positivas. Además su gradiente está dado por

$$\nabla(-\log(\det))(M) = -M^{-1}, \tag{5.5}$$

es decir, $D(-\log(\det))(M)[N] = \nabla(-\log(\det))(M) \cdot N = \text{traza}\,(-M^{-1}N)$, asimismo, para el hessiano tenemos que

$$H(-\log(\det))(M)[N][L] = -\text{traza}\left(M^{-1}NM^{-1}L\right), \tag{5.6}$$

de donde se puede obtener la convexidad estricta de la función en el cono abierto de las matrices definidas positivas.

Con esto, se puede asegurar la existencia y unicidad del problema, ya que estamos suponiendo que al menos para algún $\widetilde{w} \in K_m$, se tiene que $\det\left(\sum_{i=1}^{m} \widetilde{w}_i M_i\right) > 0$, de donde nos podemos restringir al convexo compacto $\{w \in K_m; -\log(\det\left(\sum_{i=1}^{m} w_i M_i\right)) <= -\log(\det\left(\sum_{i=1}^{m} \widetilde{w}_i M_i\right))\}$ donde la función objetivo es estrictamente convexa.

Una vez que el problema de existencia y unicidad está justificado podemos aplicarle toda la maquinaria de programación convexa, para lo cual consultaremos [21], libro de distribución gratuita que se encuentra en la dirección http://www.stanford.edu/ boyd/cvxbook/.

En nuestra implementación ocupamos el **open-source software** para matemáticas, ciencia, e ingeniería, llamado SCIPY (pronunciado "Sigh Pie"), que junto a la librería NUMPY nos ayudaron a la programación en PYTHON. Nuestro programa utiliza además el paquete CVOPT desarrollado por los autores de [21], el cual utiliza un método Primal-Dual de Punto-Interior, cuya documentación teórica podemos encontrarla completamente en [21], libro en línea, al cual deberá acudir el lector interesado. El paquete es distribuido bajo los términos de una *GNU General Public License*.

5.3. El Problema de Localizar Estaciones

Observemos que hemos construido una función que, dados los puntos distintos $p_1, \ldots, p_m$, nos entrega la solución $w \in K_m$ del problema (5.4), con la cual podemos evaluar la función objetivo y así construir una nueva función $\mathcal{J}$ dependiente únicamente de los puntos $p_1, \ldots, p_m$,

$$\begin{array}{rcl} \mathcal{J} : \Lambda^m & \mapsto & \mathbb{R} \cup \{\infty\} \\ (p_1, \ldots, p_m) & \mapsto & \text{mín}_{w \in K_m} - \log(\det\left(\sum_{i=1}^{m} w_i \mathbb{X}(p_1,\ldots,p_m)^t \mathbb{X}(p_1,\ldots,p_m)\right)) \end{array} \tag{5.7}$$

Tenemos entonces que resolver el siguiente problema:

$$\begin{array}{ll} \text{mín} & \mathcal{J}(p) \\ s.a. & p \in \Lambda^m \end{array} \tag{5.8}$$

Ahora bien, aunque se podrían hacer hipótesis sobre la regularidad de los $\mathbb{x}$ con respecto a los $(p_1, \ldots, p_m)$, y así aprovechar de alguna forma nuevamente la regularidad de la función $-\log(\det(M))$, la irregularidad práctica que nunca vamos a poder salvar es la del dominio de definición Λ. Para tener una idea de esta irregularidad, debemos recordar que las estaciones se localizan en torno a una cavidad minera, donde es lógico que existan zonas donde es factible colocarlas y que estén separadas topológicamente, razón por la cual es necesario suponer Λ como un dominio disconexo, lo que conduce a que $\mathcal{J}$ pueda ser incluso discontinua.

En la práctica, para trabajar con el dominio irregular, lo que haremos será traspasar su irregularidad a la función objetivo. Esto lo haremos con una transformación que deje al dominio conexo como se muestra en la siguiente figura 5.1.

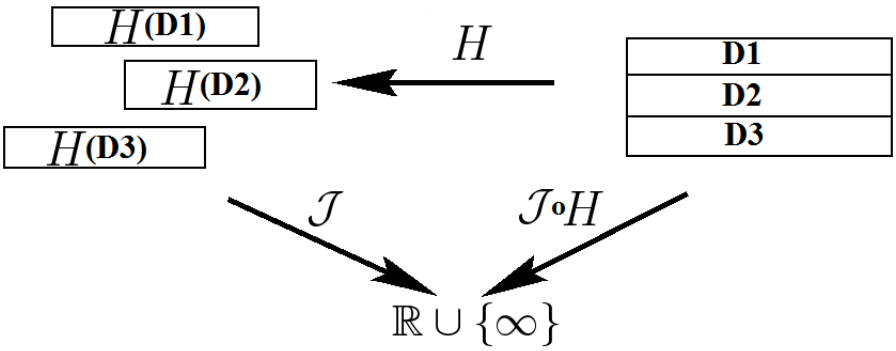

Figura 5.1: Regularización del Dominio.

Despreciaremos los problemas en los bordes de cada componente conexa debidos a la dificultad de alcanzarlos completamente con la función H, y privilegiaremos la inyectividad y simplicidad de H.

Tenemos entonces que $\Lambda = H(D1) \cup H(D2) \cup H(D3)$, definimos $\Xi = D1 \cup D2 \cup D3$ y $\Upsilon = \mathcal{J} \circ H$, el problema se transforma en

$$\begin{array}{ll} \text{mín} & \Upsilon(x_1, \ldots, x_m) \\ s.a. & x_i \in \Xi. \end{array} \tag{5.9}$$

Un ejemplo de la gráfica de Υ al variar sólo un punto en Ξ podemos apreciarla en la figura 5.2, donde hemos preferido mostrar $-\Upsilon$, en lugar de Υ, ya que de esta forma se pueden apreciar mejor sus discontinuidades.

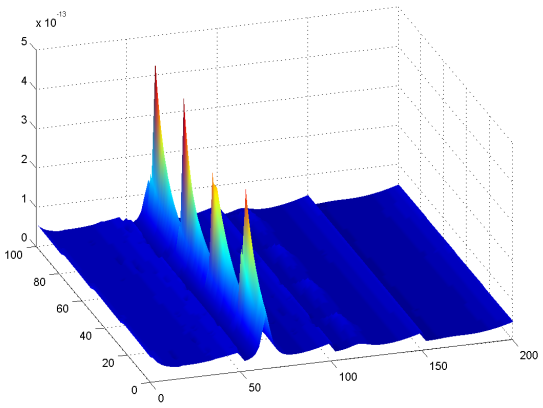

Figura 5.2: Ejemplo de la gráfica de $-\Upsilon$.

Para este ejemplo Ξ resulta ser un cuadrado de ancho 200 y alto 100, y Λ resulta estar compuesta por los niveles mostrados en la figura 5.3, la transformación H, divide a Ξ horizontalmente en partes iguales, tantas veces como niveles tenga Λ, y cada una de estas divisiones le aplica una transformación afín al nivel correspondiente, respetando el orden dado por la altura.

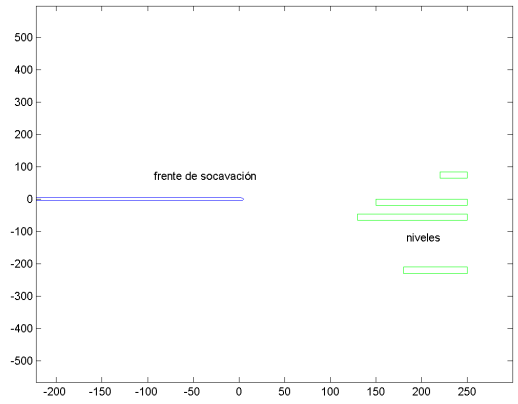

Figura 5.3: Λ elegido para el ejemplo de la gráfica de $-\Upsilon$.

Debido a la gran cantidad de mínimos locales, así como a las discontinuidades presentes de la función Υ, se han propuesto distintos algoritmos para la búsqueda de diseños óptimos (ver [18]). La estructura general de estos algoritmos de búsqueda sería la siguiente

- Dado un diseño inicial.
- Construir un nuevo diseño a partir del anterior siguiendo una operación de refresco.
- Evaluar el nuevo diseño y decidir si se continua con la operación de refresco.

donde la operación de refresco es crítica ya que de ella depende la calidad del diseño final.

El problema de determinar cuál método es el mejor para la búsqueda de diseño óptimo no lo abordamos en esta memoria. Las razones que nos mueven a decidirnos por el método de *Simulated Annealing* son simplemente prácticas; es uno de los algoritmos más utilizados en la búsqueda de diseño óptimo debido a su rapidez y calidad del diseño obtenido, una implementación de este algoritmo para MATLAB se puede encontrar en [17], en el cual nos hemos basado a la hora de implementarlo en PYTHON.

5.4. El algoritmo utilizado (Simulated Annealing)

5.4.1. Descripción del Algoritmo

La implementación del algoritmo usado en esta memoria está basada en Corana et al. (1987), (ver [13]), y T.H. Waterhouse (2005), (ver [17]). En adelante daremos una pequeña descripción de su procedimiento.

Consideremos un *diseño de localizaciones* ξ como una matriz de $m \times d$, tal como la siguiente

$$\xi = \begin{bmatrix} x_1 \\ \vdots \\ x_m \end{bmatrix} \tag{5.10}$$

con $\xi_{i,j}$ cada una de sus componentes y $x_i \in \mathbb{R}^d$ un vector fila. Recordemos que el dominio de búsqueda Ξ consta de los *diseños de localizaciones* ξ, tales que cada vector fila x_i pertenezca

a un cubo d-dimensional. Escribimos las matrices que definen estos cubos por, **L** para las cotas inferiores y **U** para las cotas superiores.

En la k-ésima iteración del algoritmo, una coordenada de ξ es perturbada aleatoriamente para obtener $\xi_{i,j}^k$. La *amplitud de búsqueda* está definida por la matriz **Vstep** $= \{v_{i,j}\}$, cuyas componentes $v_{i,j}$ son ajustadas cada cierto número de iteraciones. El valor $\xi_{i,j}^k$ es obtenido con una distribución uniforme en el intervalo $\xi_{i,j}^{k-1} \pm v_{i,j}$.

Para cada nuevo diseño ξ^k, se calcula su criterio $C(\xi^k) = -\Upsilon(\xi^k)$ y se compara con el criterio para el "mejor" diseño hasta entonces ξ^*. Los diseños con un mejor valor para el criterio siempre son aceptados y pasan a ser ξ, los diseños con un valor más bajo para el criterio son aceptados o rechazados de acuerdo con el criterio de Metropolis, i.e. la probabilidad de aceptarlos es $\exp[(C(\xi^k) - C(\xi^*)/\mathbf{temp}]$, donde **temp** es la temperatura actual del algoritmo. La temperatura baja cada cierto número de iteraciones, con un enfriamiento geométrico,

$$\mathbf{temp} \rightarrow \mathbf{cool} \cdot \mathbf{temp}$$

El valor inicial de la temperatura $\mathbf{temp}_0$ se calcula después de un cierto número de iteraciones iniciales, con $\mathbf{temp}_0$ escogido como la temperatura para la cual la probabilidad inicial de aceptar "malos" diseños sea cercana a **initprob** (por defecto $95\,\%$). Los diseños fuera del rango definido por **L** y **U** siempre son rechazados.

El algoritmo se detiene cuando se sobrepasa el número máximo de iteraciones **maxit** o la *amplitud de búsqueda promedio* es menor que cierto valor, i.e. $\frac{1}{m \cdot d} \sum_{i,j} v_{i,j} <$ **vmin**, donde **vmin** es un valor de tolerancia pre-establecido.

5.4.2. El algoritmo

Paso 1 : Inicializar los parámetros.

Paso 2 : Calcular la temperatura inicial $\mathbf{temp}_0$.

Paso 3 : Iterar lo siguiente mientras el número de iteraciones no sobrepase **maxit** y la amplitud de búsqueda promedio sea menor que **vmin** :

SubPaso1 : Hacer **ncyct** veces lo siguiente:

SubSubPaso1 : Hacer **ncycs** veces un **ciclo de búsqueda**.

SubSubPaso2 : Ajustar cada componente de la amplitud de búsqueda **Vstep** (esto depende tanto de la temperatura como del número de rechazos).

SubPaso2 : Ajustar la temperatura (enfriar el algoritmo).

5.4.3. Inicialización de los parámetros

Vamos a ver a continuación una descripción de los parámetros con el fin de inicializarlos de forma adecuada.

initprob Es la probabilidad inicial de aceptar que el algoritmo escape de un máximo local. En otras palabras, es la probabilidad inicial de que el algoritmo tome un camino de descenso (recordemos que estamos maximizando el criterio) (por defecto es igual a $0{,}95$).

Vstep Es la *amplitud de búsqueda inicial* para el algoritmo. En otras palabras, son las coordenadas del cubo donde se concentra la búsqueda (por defecto es igual a $\mathbf{UB} - \mathbf{UL}$).

cool Es la taza con la cual baja la temperatura del algoritmo, es decir, con la que se realiza el *ajuste de la temperatura*

$$\textbf{temp} \rightarrow \textbf{cool} \cdot \textbf{temp}$$

La temperatura **temp**, es la energía cinética de la búsqueda. Ésta afecta directamente, tanto a la probabilidad de aceptar que el algoritmo escape de un máximo local **initprob** (a mayor energía cinética, mayor esta probabilidad), como a la *amplitud de búsqueda* **Vstep** (a mayor energía cinética, mayor la amplitud de búsqueda) (por defecto es igual a $0{,}9$).

ncycs Es el número de **ciclos de búsqueda** entre cada ajuste de la *amplitud de búsqueda* **Vstep** (por defecto es igual a 10).

ncyct Es el número de ajustes de la *amplitud de búsqueda* **Vstep** entre cada *ajuste de temperatura* **temp** (por defecto es igual a 20).

maxit Es el máximo número de iteraciones (por defecto es igual a $1e7$).

vmin Es la mínima *amplitud relativa de búsqueda*, es decir, si luego de un *cambio de temperatura*, se tiene que $\frac{\textbf{Vstep}}{\textbf{UB}-\textbf{UL}} < \textbf{vmin}$ (para cada coordenada), entonces el algoritmo se detiene (por defecto es igual a $0{,}001$).

Los parámetros anteriores son de carácter general, es decir, la mayoría de los algoritmos "de templamiento" (annealing) tienen incorporados estos parámetros, sin embargo, nuestro algoritmo, que es una modificación del utilizado por T.H. Waterhouse (2005), tiene incorporada una etapa previa que calcula la temperatura inicial del algoritmo, para ello, además de utilizar **initprob** utiliza el siguiente parámetro:

ncyci Es el número de **ciclos de búsqueda** para calcular la temperatura inicial (por defecto es igual a 100).

Por último, también se tiene el parámetro siguiente:

dispint Es el número de *ajustes de temperatura* entre cada aparición de los resultados en la pantalla (por defecto es igual a 5).

Estos parámetros se pueden ingresar, si así se desea, parcial o totalmente, en un arreglo de entrada **options**.

5.4.4. Ciclo de búsqueda

Éste está compuesto por el siguiente bloque:

Para cada coordenada $\xi_{j,k}$ de ξ se realiza lo siguiente

- El contador de iteraciones se aumenta en 1.
- Se define:

$$\begin{aligned} \mathbf{tmp}_\xi &= \xi \\ \mathbf{tmp}_\xi &= \xi_{j,k} + (2\mathbf{rand} - 1)v_{j,k} \end{aligned}$$

  donde **rand** es una variable uniformemente distribuida en $[0, 1]$.

- - Si $\mathbf{tmp}_\xi \notin \Xi$, se define

$$d = \infty$$

  - Sino, se define

$$d = C(\mathbf{tmp}_\xi)$$

    donde C es el criterio a maximizar.

- - Si $d >$ **opt**, se redefinen

$$\begin{aligned} \xi &= \mathbf{tmp}_{\xi} \\ \mathbf{opt} &= d \end{aligned}$$

- Sino,
  - Si $\exp(\frac{d-\mathbf{opt}}{\mathbf{temp}}) > \mathbf{rand}$, se redefinen:

$$\begin{aligned} \xi &= \mathbf{tmp}\xi \\ \mathbf{opt} &= d \end{aligned}$$

  - Sino, se redefine

$$\mathbf{Rech}_{j,k} = \mathbf{Rech}_{j,k} + 1$$

donde **opt** es una variable cuyo valor inicial es $-\infty$, y **Rech** una variable cuyo valor inicial es 0.

- Si **opt** $>$ **best**, se redefinen

$$\begin{aligned} \mathbf{best}_{\xi} &= \xi \\ \mathbf{best} &= d \end{aligned}$$

donde **best** es una variable cuyo valor inicial es $-\infty$.

best y $\mathbf{best}_{\xi}$ son las variables donde se almacena el valor máximo y el "mejor" diseño respectivamente. La variable **Rech** es la que almacena el número de rechazos (necesaria para ajustar la amplitud de búsqueda).

5.5. API del Código de Diseño Óptimo (ODC) implementado

A continuación se describe la API del código de diseño óptimo implementado en PYTHON, las palabras escritas en negrita corresponden a términos propios de la programación en PYTHON, la *codificación* de **arreglos** y *matrices* está dada al final se la sección.

5.5.1. Funciones públicas

mFichero(nombre)

- **nombre** es una **cadena** con el nombre de un **fichero** que contenga un **arreglo** *codificado* de dimensiones (N, n, m).

Retorna el **arreglo** *codificado* en el **fichero** nombrado **nombre**.

mFicheroW(A, nombre)

- **A** es un **arreglo** de dimensiones (N, n, m).
- **nombre** es una **cadena** que representa al nombre de un **fichero** donde se *codificará* el **arreglo A**.

No retorna nada, sólo *codifica* el **arreglo A** y lo guarda en el **fichero** nombrado **nombre**.

put_puntos(r,L,H)

- **r** es un **flotante** que representa la densidad de puntos en la malla del dominio Ξ.
- **L** es un **flotante** que representa el largo del rectángulo utilizado para representar al dominio Ξ.
- **H** es un **flotante** que representa el alto del rectángulo utilizado para representar al dominio Ξ.

No retorna nada, sólo *codifica* un **arreglo** de dimensiones $(1, N, 2)$, que representa una malla de puntos en Ξ, y esta *codificación* la guarda en el **fichero** nombrado **puntos** en el directorio donde se realice la llamada a la función.

put_puntosL(geometria, puntos)

- **geometria** es un **arreglo** de dimensiones $(K, 4)$, donde cada fila es de la forma $[x_i, x_f, y_i, y_f]$, y define un rectángulo en el dominio Ω, como se muestra en la figura 5.4

- **puntos** es un **arreglo** de dimensiones $(N, 2)$ que representa la malla de puntos en Ξ.

No retorna nada, sólo *codifica* un **arreglo** de dimensiones $(1, N, 2)$, que representa la imagen en Ω de malla de puntos **puntos** de Ξ, y esta *codificación* la guarda en un **fichero** nombrado **puntosL** en el directorio donde se realice la llamada a la función.

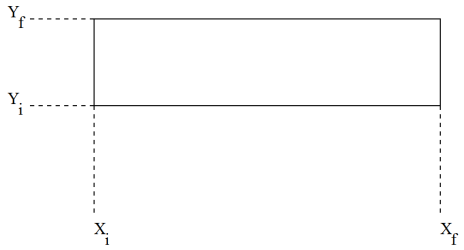

Figura 5.4: Parametrización de un rectángulo en Ω.

put_pesos(X)

- **X** es un **arreglo** de dimensiones (m, n, p) que representa las m matrices obtenidas al evaluar la función $\mathbb{x}$ en cada punto $p_1, \ldots, p_m$. Recordemos que cada una de estas matrices son de dimensiones (n, p).

No retorna nada, sólo *codifica* un **arreglo** de dimensiones $(1, 1, m)$, que representa los pesos de cada una de las m estaciones $p_1, \ldots, p_m$, y guarda esta *codificación* en un **fichero** nombrado **pesos** en el directorio donde se realice la llamada a la función.

put_disegno(m,puntos,X,XF,TpuntosL)

- **m** es un **entero** que representa el número de estaciones que se quieren añadir al diseño.
- **puntos** es un **arreglo** de dimensiones $(N, 2)$ que representa la malla de puntos en Ξ.
- **X** es un **arreglo** de dimensiones (N, n, p) que representa a la función $\mathbb{x}$ evaluada en cada una de las N imágenes en Ω de los puntos de la malla **puntos**.
- **XF** es un **arreglo** de dimensiones (NF, n, p) que representa a la función $\mathbb{x}$ evaluada en cada una de las NF *estaciones* elegidas como fijas (esas son las *estaciones* que ya

están instaladas).

- **TpuntosL** es un **arreglo** de dimensiones $(N + NF, n, p)$ que representa la totalidad de puntos en Ω, es decir, los puntos imágenes de la malla de puntos **puntos** y al final y los puntos correspondientes a **XF**.

No retorna nada, sólo *codifica* un **arreglo** de dimensiones $(1, m + NF, 3)$, que representa al *diseño óptimo* τ obtenido en el dominio Λ y solución aproximada del problema (5.1), y esta *codificación* la guarda en un **fichero** nombrado **disegno** en el directorio donde se realice la llamada a la función.

5.5.2. Codificación de arreglos y *matrices*

La codificación de un **arreglo** A de dimensiones (N, n, m) está dada a continuación

```
N n m
 Fila 1 de la matriz 1
      .
      .
      .
 Fila n de la matriz 1
           .
           .
           .
 Fila 1 de la matriz N
      .
      .
      .
```

```
Fila n de la matriz N
```

donde "`Fila k de la matriz j`" denota a la fila dada por los elementos $[A]_{jk1} \ldots [A]_{jkm}$, escritos sin paréntesis y separados por al menos un espacio.

La codificación de un **arreglo** o *matriz* B de dimensiones (n, m) se realiza como si fuera un **arreglo** A de dimensiones $(1, n, m)$, cuyos elementos están dados por:

$$[A]_{1jk} = [B]_{jk}.$$

Capítulo 6

Protocolos, resultados, comparación y discusión

Se entiende, durante todo este capítulo, que nos referimos con *directorio de trabajo*, al directorio donde se ubica el fichero nombrado **ODC.py**.

6.1. Preparación de los protocolos utilizados

En esta sección se muestra la forma de preparar los protocolos que deberán automatizarse para concluir en los protocolos de la siguiente sección.

6.1.1. Preparación del protocolo para obtener los pesos óptimos.

Recordemos el problema (5.4) que debemos resolver:

$$\begin{array}{ll} \text{mín} & -\log(\det\left(\sum_{i=1}^{m} w_i M_i\right)) \\ s.a. & \sum_{i=1}^{m} w_i = 1 \\ & w_i \geq 0. \end{array} \tag{6.1}$$

para resolverlo debemos definir las matrices M_i, las cuales están dadas por

$$M_i = \mathbb{X}_{(p_i)}{}^t\mathbb{X}_{(p_i)}.$$

donde $p_1, \ldots, p_m$ son las ubicaciones de las estaciones consideradas. Procedemos entonces a *codificar* un arreglo, digamos A, de dimensiones (m, n, p) cuyos elementos están dados por

$$[A]_{ijk} = [\mathbb{X}_{(p_i)}]_{jk}$$

y guardamos esta *codificación* en un fichero nombrado **baseF** ubicado en el directorio de trabajo.

6.1.2. Preparación del protocolo para añadir estaciones.

Recordemos el problema (5.8) que debemos resolver:

$$\begin{array}{ll} \text{mín} & \mathcal{J}(p_1, \ldots, p_m) \\ s.a. & p_i \in \Lambda \end{array} \tag{6.2}$$

donde $\mathcal{J}$ está dado en (5.7). Para resolverlo es necesario definir el dominio Λ, el cual lo definiremos como un conjunto formado por cuatro rectangulos, como los mostrados en la siguiente figura:

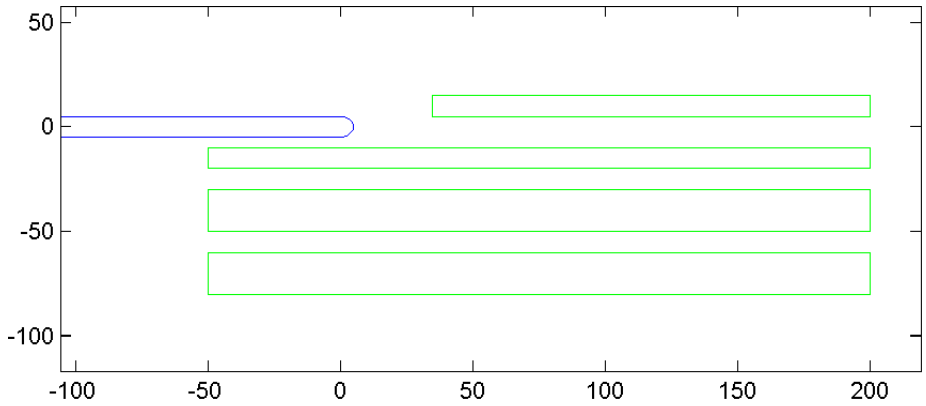

Figura 6.1: El dominio de búsqueda Λ que ocuparemos.

codifico estos rectángulos de la siguiente forma:

```
1 4 4
35 200 5 15
-50 200 -20 -10
-50 200 -50 -30
-50 200 -80 -60
```

que no es ni más ni menos que la *codificación* de una matriz de dimensiones $(K, 4)$, donde K representa el número de rectángulos y cada fila $[x_i, x_f, y_i, y_f]$ representa a un rectángulo como el mostrado en la figura 5.4. Guardamos esta codificación en un fichero ubicado en el directorio de trabajo y nombrado **geometria**. Una vez definida la geometría de Λ, pasamos a definir una malla en Λ, para ello definimos el dominio Ξ y su correspondiente malla con la *codificación* siguiente:

```
1 1 3
1 100 50
```

que corresponde a la *codificación* de una matriz con una única fila `1 100 50`, el primer elemento de esta fila representa la densidad de puntos de la malla (en nuestro caso `1`), el segundo elemento representa el ancho de Ξ (en nuestro caso `100`) y el último elemento representa el alto (en nuestro caso `50`). El dominio Ξ será entonces, un cuadrado de ancho `100` y alto `50`. Guardamos esta codificación en un fichero ubicado en el directorio de trabajo y nombrado **malla**.

Luego ejecuto las siguientes líneas en PYTHON:

```
>>from ODC import *
>>malla = mFichero('malla')[0][0]
```

```
>>geometria = mFichero('geometria')[0]
>>put_puntos(malla[0],malla[1],malla[2])
>>puntos = mFichero('puntos')[0]
>>put_puntosL(geometria,puntos)
```

con las cuales se obtienen los ficheros nombrados **puntos** y **puntosL**, quienes contienen una *codificación* de una matriz de dimensiones $(N, 2)$ que representa a la malla de puntos de Ξ y Λ respectivamente. Podemos ver estas mallas en la figura 6.2

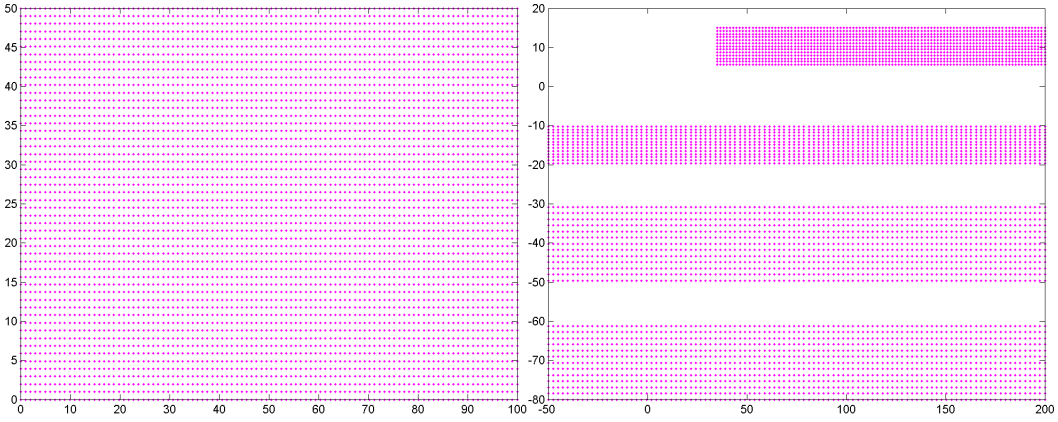

Figura 6.2: A la izquierda la malla del dominio Ξ, a la derecha su correspondiente malla del dominio Λ.

Las diferentes densidades que se ven en el dominio Λ son producidas por las distintas dimensiones de sus rectángulos, pues cada uno tendrá la misma cantidad de puntos, esto hay que tenerlo en cuenta a la hora de definir la geometría *codificada* en el fichero nombrado **geometria**.

El fichero nombrado **puntos** es de "privado". El fichero nombrado **puntosL** lo ocuparemos para evaluar la función $\mathbb{x}$ en cada uno de los puntos que definen la malla *codificada* en **puntosL**, tendremos entonces $\mathbb{x}(pL_1), \ldots, \mathbb{x}(pL_N)$, los cuales *codificaremos* en un arreglo, digamos A, de dimensiones (N, n, p) cuyos elementos estarán dados por

$$[A]_{ijk} = \left[\mathbb{x}(pL_i)\right]_{jk}$$

este arreglo lo guardaremos en un fichero nombrado **base** en el directorio de trabajo.

A continuación, debemos definir la localización de las "estaciones fijas" $pLF_1, \ldots, pLF_{NF}$, para lo cual *codificamos* una matriz, digamos B, de dimensiones $(NF, 2)$ cuyos elementos están dados por

$$[B]_{jk} = [pLF_j]_k$$

lógicamente, en el caso de no ocupar "estaciones fijas" $NF = 0$, y el arreglo resultará estar vacío, en cuyo caso debemos ocupar el siguiente *código* para la *matriz* B:

```
1 0 2
```

Independientemente de la *codificación* utilizada para B, ésta debemos guardarla en un fichero nombrado **puntosLF** ubicado en el directorio de trabajo.

Procedemos a *codificar*, análogamente a como lo hicimos en unas líneas más arriba, los valores de la función $\mathbb{x}$ en cada uno de los puntos $pLF_1, \ldots, pLF_{NF}$, que definen en la *codificación* del fichero nombrado **puntosLF**, y ésta la guardamos, análogamente, en un fichero nombrado **baseF**. Si estamos en el caso de que $NF = 0$, entonces la *codificación* resulta ser la siguiente:

```
0 6 9
```

Para finalizar debemos indicar la cantidad m de estaciones que queremos añadir, número que *codificamos* en una matriz de dimensiones $(1, 1)$ cuyo único elemento será m. Esta codificación la guardamos en un fichero nombrado **m** ubicado en el directorio de trabajo.

6.2. Protocolos para obtener el Diseño Óptimo

En esta sección se explica el uso de la API para obtener tanto los pesos óptimos como el diseño óptimo. La definición de los ficheros nombrados **baseF**, **malla**, **geometria**, etc., se puede encontrar en la sección anterior, donde se muestra la preparación de estos protocolos.

6.2.1. Protocolo para obtener los pesos óptimos

1. Escribo el fichero **baseF**.

2. Ejecuto el las siguientes líneas en PYTHON.

```
>>from ODC import *
>>baseF = mFichero('baseF')
>>put_pesos(baseF)
```

En el fichero nombrado **pesos** se encontrará la codificación de un arreglo con los pesos óptimos correspondientes, exactamente en el mismo orden que las matrices *codificadas* en el fichero nombrado **baseF**.

6.2.2. Protocolo para obtener el diseño óptimo

1. Escribo los ficheros **malla** y **geometria**.

2. Ejecuto las siguientes líneas en PYTHON.

```
>>from ODC import *
>>malla     = mFichero('malla')[0][0]
>>geometria = mFichero('geometria')[0]
>>put_puntos(malla[0],malla[1],malla[2])
```

```
>>puntos = mFichero('puntos')[0]
>>put_puntosL(geometria,puntos)
```

con las líneas anteriores se obtiene un fichero nombrado **puntosL**, el cual contiene la *codificación* de un arreglo que representa la malla de puntos en Ω, donde tendremos que evaluar $\mathbb{X}$.

3. Escribo los ficheros **baseF**, **puntosLF**, **base m**.

4. Ejecuto las siguientes líneas en PYTHON.

```
>>from ODC import *
>>m        = mFichero('m')[0][0]
>>baseF    = mFichero('baseF')
>>puntosLF = mFichero('puntosLF')[0]
>>base     = mFichero('base')
>>puntos   = mFichero('puntos')[0]
>>puntosL  = mFichero('puntosL')[0]
>>Lp       = concatenate((puntosLF,puntosL), axis=0)
>>put_disegno(m,puntos,base,baseF,Lp)
```

En el fichero nombrado **disegno** se encontrará la *codificación* de un arreglo con el diseño óptimo τ.

6.3. Diseño Óptimo

Supongamos que hasta el momento, no hemos colocado ninguna estación de monitoreo, que no contamos con una preminería, y que las estaciones están restringidas a pertenecer al Λ mostrado

en la figura 6.1, comenzando con **cuatro** puntos escogidos aleatoriamente, obtenemos el diseño mostrado en la figura 6.3, donde el número que acompaña al punto es su respectivo peso, y podemos percatarnos que sólo dos de los cuatro puntos son relevantes.

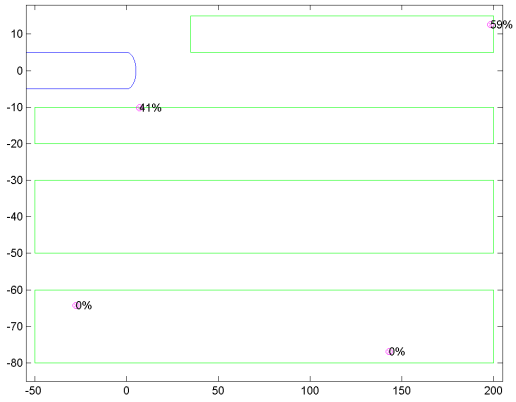

Figura 6.3: Diseño obtenido al variar **cuatro** puntos.

Asimismo, los diseños obtenidos partiendo de una menor cantidad de puntos, se muestran en la figura 6.4.

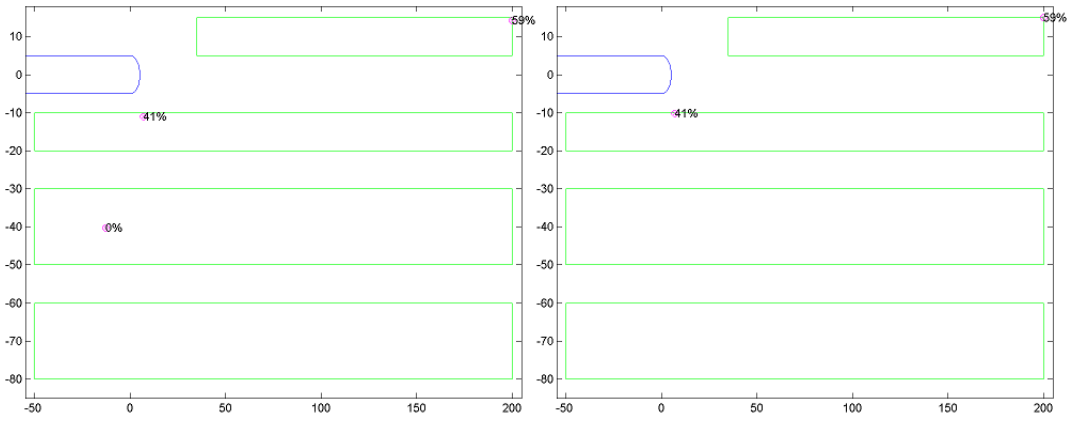

Figura 6.4: Diseños obtenidos al variar: a la izquierda **tres** puntos, y a la derecha **dos**.

Como se puede esperar, los puntos con pesos muy cercanos a 0 resultan ser muy inestables y su efecto se ve reflejado en el mayor tiempo de calculo que toma nuestro algoritmo de búsqueda.

6.4. Análisis de la distancia al *frente de avance* de la preminería

Para estas pruebas Λ se compone de un único rectángulo, tal como el mostrado en la figura 6.5, cuya codificación es la siguiente:

```
1 1 4
300 1000 -20 20
```

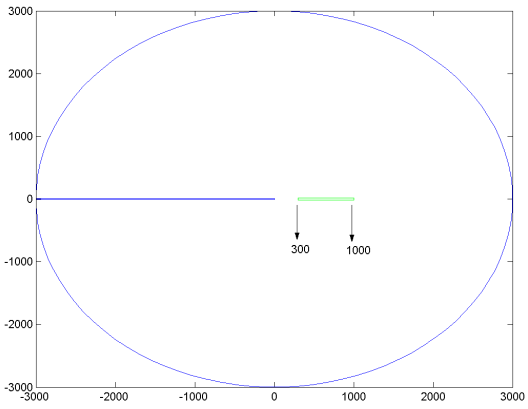

Figura 6.5: El dominio de búsqueda Λ para las pruebas de esta sección.

La malla que utilizamos para discretizar el dominio de búsqueda Λ podemos apreciarla en la figura 6.6.

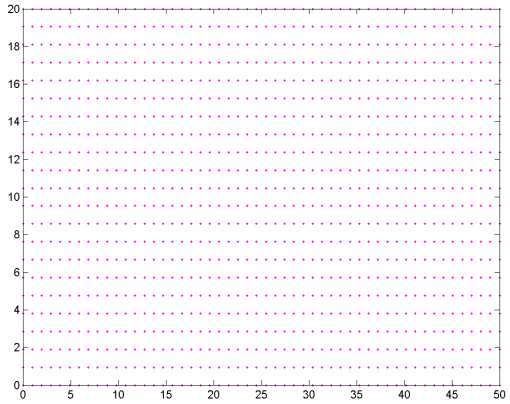

Figura 6.6: Discretización del dominio de búsqueda Λ.

Los planos elegidos corresponden a los generados por los puntos ubicados a una distancia $2R$ del "centro de gravedad" del *frente de avance* (R es el "radio" del frente de avance), estos además están ubicados entre los ángulos -60 y 60 con respecto a la ordenada, tal como se muestra en la figura 6.7

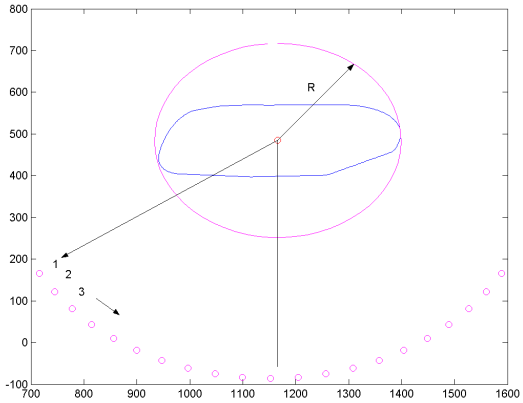

Figura 6.7: Puntos elegidos para generar los planos.

Observemos que cada ángulo define un punto con el cual se define un plano perpendicular al *frente de avance*, el ángulo que forma este plano con la ordenada no es igual al ángulo con el que se define al punto. En adelante cuando hablemos de "ángulos" estaremos refiriéndonos a los ángulos con los que definimos estos puntos, los cuales nos servirán para parametrizar nuestras pruebas.

Para estas pruebas debemos resolver tanto el problema de encontrar los pesos óptimos, cuando hablemos de "punto fijo a cierta distancia del *frente de avance*", y tendremos que resolver el problema de añadir una estación de medición.

Para el primer resultado, supongamos que queremos colocar una estación en Λ que reemplace a la preminería[1], la pregunta que nos formulamos es : ¿A qué distancia del frente debemos colocarla? La respuesta se muestra en la figura 6.8, en ella se ve la distancia al frente óptima (dentro de los límites de Λ) en función del ángulo con el que definimos los puntos de la figura 6.7, donde debemos

[1]La preminería se toma como el valor de una estación a una distancia "lejana" del frente.

recordar que la distancia de 1000 mt. es la máxima distancia dentro de los límites de Λ.

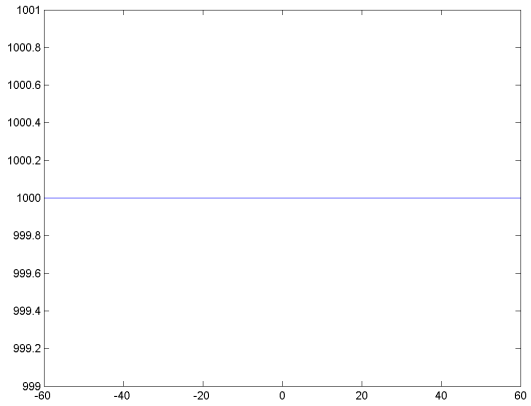

Figura 6.8: Distancia a la cual es óptimo colocar una estación que reemplace a la preminería.

También podemos apreciar un gráfico del peso que se le otorga a esta estación en el diseño óptimo, (recordemos que también estamos considerando las estaciones ubicadas en el sector Esmeralda), esto se puede apreciar en la figura 6.9, en ella se puede apreciar el peso óptimo en función del ángulo con el que definimos los puntos de la figura 6.7.

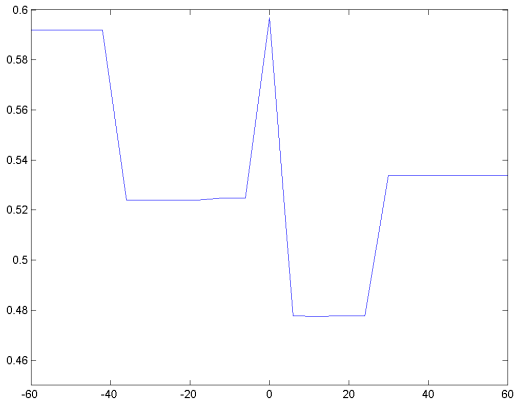

Figura 6.9: Pesos en el diseño óptimo, para la estación que reemplace a la preminería.

La pregunta siguiente es ¿cuanto perdemos, en términos relativos, con nuestro criterio, al escoger otra distancia al frente para los puntos?

La respuesta se puede apreciar en la figura 6.10, donde se muestra para cada ángulo, y por tanto, para cada plano,

$$\frac{\mathrm{JM}_{\text{punto a distancia dada}}}{\mathrm{JM}_{\text{punto movil en lambda}}}$$

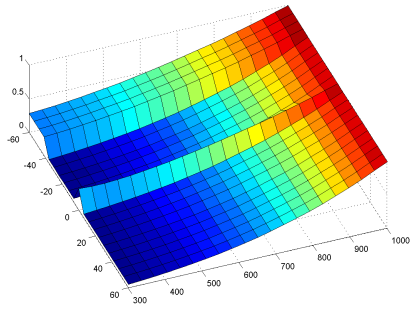

Figura 6.10: Perdida en términos relativos, con nuestro criterio, al escoger otra distancia al *frente de avance*.

donde $\mathrm{JM}_{\text{punto a distancia dada}}$ es el conseguido al fijar el punto a una distancia determinada del frente (en el plano definido por el ángulo) y buscar con él y las estaciones los pesos óptimos, y en ese diseño encontrado evaluar el criterio. Si queremos apreciar con mejor detalle lo que perdemos al ubicar el punto fijo a una distancia determinada del frente lo podemos ver en la figura 6.11, donde se encuentran superpuestas las imágenes para cada ángulo con el cual definimos los planos.

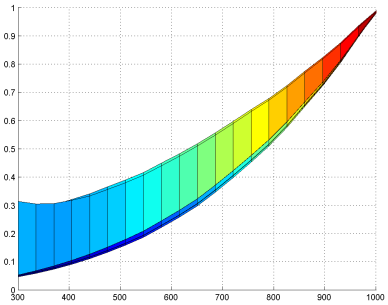

Figura 6.11: Superposición para distintos planos de la pérdida en términos relativos, al escoger otra distancia al *frente de avance*.

Con respecto a los "$\mathrm{JM}_{\text{punto a distancia dada}}$", podemos agregar la gráfica de sus pesos en el diseño, como se muestra en la figura 6.12, donde se puede apreciar el cambio producto del cambio de ángulos de definición del plano.

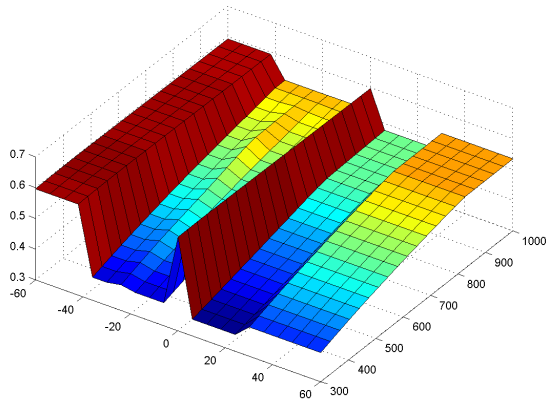

Figura 6.12: Pesos en el diseño para los puntos en el plano definido por un ángulo dado y a una distancia dada del *frente de avance*.

Esto seguramente se produce debido a que estamos considerando sólo las estaciones ubicadas a 100 metros del plano perpendicular al *frente de avance*, y por tanto, al variar el ángulo con el que definimos estos planos, pasa discretamente de considerar una estación a no considerarla, lo que influye en la configuración del diseño y por ende en los pesos. Para apreciar mejor la dependencia con el ángulo que define los planos podemos ver la figura 6.13, donde se encuentran superpuestas las imágenes para las distintas distancia al *frente de avance*, donde se pueden distinguir 4 zonas.

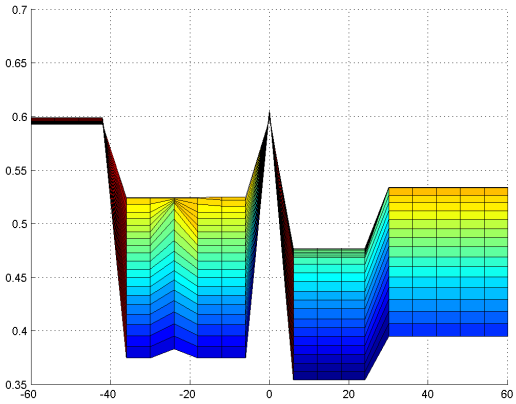

Figura 6.13: Superposición para distintas distancias al *frente de avance*, de los pesos en el diseño para los puntos ubicados en un plano definido por un ángulo dado.

Podríamos también apreciar la dependencia de estos pesos con respecto a la distancia al *frente de avance*, como se muestra en la figura 6.14, donde se puede ver que para distancias mayores de 700 metros, independientemente del ángulo con el que se defina al plano, los pesos de estos puntos se encuentran entre un $40\,\%$ y $60\,\%$.

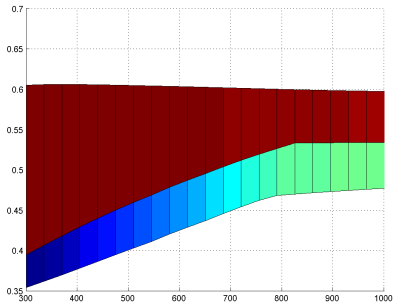

Figura 6.14: Superposición para distintos ángulos que definen los planos, de los pesos en el diseño para los puntos ubicados a una distancia dada del *frente de avance*.

6.5. Comparación con un estimador alternativo

Consideremos el caso en el que buscamos los **pesos óptimos** y comparemos el estimador obtenido, con el otro obtenido al considerar **cuatro estaciones** y asignar como pesos los dados por

$\max\{0{,}01, \frac{1}{\sqrt{1+dist}}\}$, donde $dist$ es la distancia de la estación con el plano considerado. Es claro que esto último define un diseño y entonces podemos evaluar nuestro criterio en éste.

La figura (6.5) se muestra $\mathrm{JM}(\tau_a)/\mathrm{JM}(\tau_{opt})$, donde τ_a es el diseño obtenido al asignar los pesos, mientras que τ_{opt} es el obtenido al buscar los pesos óptimos.

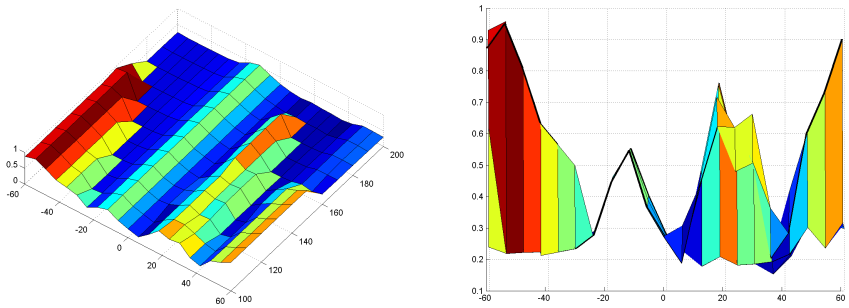

Figura 6.15: Comparación relativa $\mathrm{JM}(\tau_a)/\mathrm{JM}(\tau_{opt})$ con las mejores cuatro estaciones.

donde la elección de las **cuatro estaciones** es la que nos dio el mejor resultado.

En la figura (6.5) se muestra lo mismo, salvo que esta vez escogimos las **cuatro estaciones** que nos dieron el peor resultado.

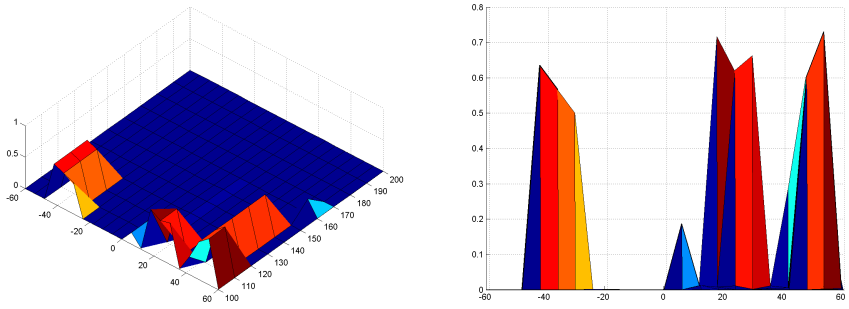

Figura 6.16: Comparación relativa $\mathrm{JM}(\tau_a)/\mathrm{JM}(\tau_{opt})$ con las peores cuatro estaciones.

Para finalizar, en la figura (6.17) comparamos con el promedio de $\mathrm{JM}(\tau_a)$ sobre las elecciones de las **cuatro estaciones**.

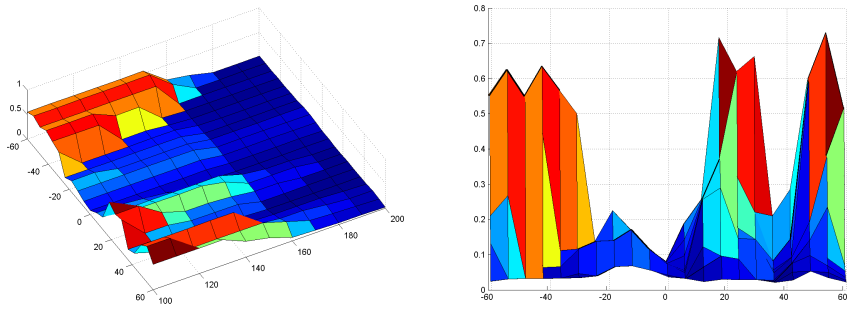

Figura 6.17: Comparación relativa $\mathrm{JM}(\tau_a)/\mathrm{JM}(\tau_{opt})$ con el promedio de elegir cuatro estaciones.

Los puntos considerados para definir los planos de incidencia con el frente son los mismos que utilizamos en la sección anterior y los podemos recordar con la siguiente gráfica:

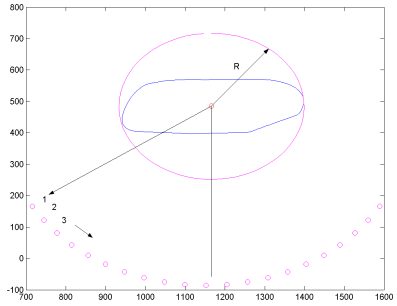

Figura 6.18: Puntos elegidos para generar los planos.

Además en los gráficos se muestra la variación con respecto a la distancia máxima permitida para las estaciones. Esto es, para cada valor V entre 100 y 200, el conjunto de estaciones consideradas, para obtener tanto τ_a como τ_{opt}, es el formado por las estaciones que se encuentran a no más de V metros de distancia del plano de incidencia definido por el punto.

Capítulo 7

Conclusiones y recomendaciones

En este capítulo haremos una mezcla tanto conclusiones como recomendaciones para futuras pruebas numéricas y trabajos.

Las primeras recomendaciones son sobre la elección de los *Campos Básicos* utilizados.

Debido al número reducido de estaciones utilizadas en los diseños óptimos que obtuvimos, se sugiere investigar el aumento del número de *Campos Básicos* (pasar de nueve a doce), esto con el fin de conseguir una mayor precisión del estado tensional del *macizo rocoso*.

También, debido a los bajos valores que se obtienen con nuestro criterio, se sugiere hacer una revisión de los *Campos Básicos* utilizados, ya que un bajo valor de nuestro criterio indica un alto "parecido" de los *Campos Básicos* utilizados, en otras palabras, no estarían siendo muy ortogonales. Recordemos que el origen de la mayor parte de los *Campos Básicos* es debido a un análisis asintótico bidimensional, lo que se recomienda es explotar los métodos estadísticos existentes para la determinación de un mejor conjunto de *Campos Básicos*.

Una observación notable, es que en ninguna parte fue necesario que los *Campos Básicos* fueran a dominio bidimensional, por lo que se recomienda la elección de *Campos Básicos* con dominios tridimensionales, los cuales se adaptan mucho mejor al fenómeno, ya que éste es tridimensional,

y se concluye que este aumento dimensional no supone una revisión del estudio hecho en esta memoria.

La elección de los *Campos Básicos* podría integrar la no homogeneidad del *macizo rocoso*, u otros que no hemos considerado en esta memoria y que mantienen la linealidad del problema.

Pasamos ahora a hacer recomendaciones sobre las modificaciones del *código de diseño óptimo* implementado en PYTHON.

Cuando queremos añadir estaciones de medición, el algoritmo de *Simulated Annealing* que estamos ocupando, cada vez que evalúa la función objetivo tiene que resolver un problema de optimización convexa, el cual hemos resuelto con un algoritmo genérico para optimización convexa, por lo que se recomienda la implementación de un algoritmo más ad-hoc con el problema que estamos resolviendo, esto debido a que el número de evaluaciones de la función objetivo es considerable, y por tanto, también el número de resoluciones de un problema de optimización convexa.

También se recomienda hacer comparaciones en tiempo, del algoritmo de *Simulated Annealing* que estamos utilizando, con respecto a un algoritmo de *Simulated Annealing* que **no** resuelva en cada evaluación de su función objetivo un problema de optimización convexa, sino que tome los pesos como variables adicionales las cuales incluya en cada iteración de *Simulated Annealing*, tal como lo realiza el algoritmo implementado por T.H. Waterhouse en [17].

Pasamos ahora a hacer recomendaciones y conclusiones sobre los resultados obtenidos en las pruebas numéricas.

Como ya hemos dicho, llama la atención el número reducido de mediciones para un diseño óptimo, lo que alienta a tomar un mayor número de *Campos Básicos*. Como sabemos, si los *Campos Básicos* son suficientemente ortogonales, el número suficiente de estaciones de medición, por un tema algebraico, es dos, en efecto la matriz $\mathcal{X}$ obtenida con dos mediciones, es una matriz 12×9-

dimensional a rango completo (si los campos son suficientemente ortogonales en cada punto), por lo que puede ser perfectamente un diseño óptimo.

También, observemos que diseños con puntos de peso nulo son equivalentes a diseños a los cuales se suprimen esos puntos, y como podemos observar, en las pruebas de diseño óptimo, cuando hemos buscado diseños con un número mayor, lo único que hemos añadido son puntos de peso nulo.

Las diferencias considerables en nuestro criterio, para una preminería colocada a una distancia dada del *frente de avance*, con respecto a una preminería colocada a la distancia óptima (como se puede apreciar nuevamente en la figura 7.1, donde se puede ver además la dependencia exponencial de esta razón), muestra la potencia de la herramienta implementada a la hora de buscar patrones en el diseño de las estaciones de medición, en particular, se muestra que a medida que colocamos más y más lejos del *frente de avance* la posición ficticia de la preminería, esta toma cada vez mayor importancia en el diseño óptimo (mayor peso), por lo que se sugiere evaluar cuál es la importancia real que se le quiere dar a la preminería, a fin de colocarla a una distancia adecuada. Recordemos que a partir de una distancia de 700 mt., el peso de la preminería en el diseño supera el $40\,\%$.

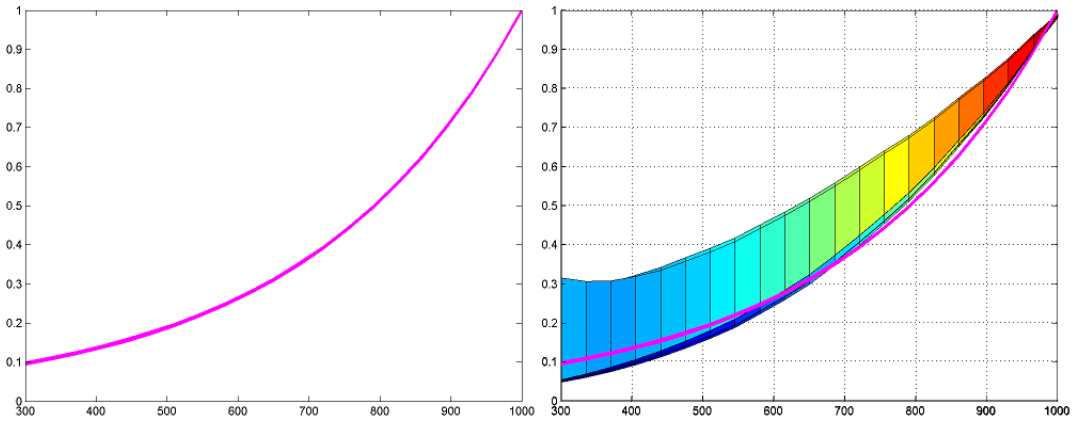

Figura 7.1: A la izquierda la función $e^{\frac{x-1000}{300}}$, a la derecha la superposición de ésta con la figura 6.11.

Para finalizar, las diferencias considerables en nuestro criterio, entre elegir los pesos de ma-

nera óptima y elegir los pesos con un estimador alternativo, también confirma la utilidad de la herramienta de elección de los pesos óptimos.

Bibliografía

[1] H. D. Bui(1978) : *Mécanique de la Rupture Fragile*.

[2] L. D. Landau and E. M. Lifshitz(1969) : *Teoría de la elasticidad*.

[3] N. I. Muskhelishvili (1954) : *Some Basic Problems of the Mathematical Theory of Elasticity*.

[4] F. Pukelsheim (1965) : *Optimal design of experiments*.

[5] A. C. Atkinson and A. N. Donev (1992): *Optimum Experimental Designs*.

[6] S. D. Silvey (1980) : *Optimal Design*.

[7] V. V. Fedorov (1972) : *Theory of Optimal Experiment*.

[8] J. Kiefer and J. Wolfowitz (1960) : *The equivalence of two extremum problems*.

[9] J. Kiefer and J. Wolfowitz (1959) : *Optimum designs in regression problems*.

[10] D. Uciński and B. Bogacka (2002) : *Construction of T-optimum designs for multiresponse dynamic model*.

[11] A. C. Atkinson and V. V. Fedorov (1975a) : *The design of experimental for discriminating between two rival models*.

[12] A. C. Atkinson and V. V. Fedorov (1975b) : *Optimal design: experiments for discriminating between several models*.

[13] A. Corana, M. Marchesi, C. Martini, and S. Ridella (1987) : *Minimizing multimodal functions of continuous variables with the "simulated annealing" algorithm*‘.

[14] Z. Botev and D. P. Kroese (2004) : *Global likelihood optimization via the cross-entropy method with an aplication to mixture models*.

[15] D. P. Kroese, S. Porotsky, and R. Rubinstein (2005) : *The cross-entropy method for continuous multi-extremal optimization*.

[16] R. Y. Rubinstein and D. P. Kroese (2004) : *The Cross-Entropy Method: A Unified Approach to Combinatorial Optimization, Monte-Carlo Simulation, and Machine Learning*, Springer-Verlag.

[17] T.H. Waterhouse : *Optimal Experimental Design for Nonlinear and Generalised Linear Models*

[18] R. Jin., W. Chen., A. Sudjianto., (2003): *An Efficient Algorithm for constructing optimial design of Experiments*, Proceedings of DETC'03 ASME 2003 Design Engineering Technical Conferences and Computers and Information in Engineering Conferenc, Chicago, Illinois USA, September 2-6.

[19] R. Molina, E. Rojas : *Development of the Variant Previous Collapse in the Method for Panels in Mines The Lieutenant*, Codelco Chile.

[20] A. Pazman (1986): *Foundations of Optimum Experimental Design*. D. Reidel publishing company. Dordrecht.

[21] S. Boyd and L. Vandenberghe (2006): *Convex Optimization* Cambridge University Press, http://www.stanford.edu/ boyd/cvxbook/.

Apéndice A

Código de Diseño Óptimo (ODC)

En este apéndice damos el *Código de Diseño Óptimo* (ODC) implementado en PYTHON.

A.1. El fichero ODC.py

```
from scipy import array,matrix,zeros,ceil, \
 sqrt,argmin,compress,concatenate,transpose,asarray
from RandomArray import permutation
from anneal import anneal
from pesos import pesos,func_pesos
from ficheros import mFicheroW


#privado
def matrices(base):
 N,n,m = base.shape
 M = zeros([N,m,m])


 for i in range(N):
```

```
  b    = matrix(base[i])
  M[i] = transpose(b)*b

 return M

#privado def magnetic(_x,_p):
 x = array(_x)
 p = array(_p)

 nx,mx = x.shape
 np,mp = p.shape

 if not(mx==mp) or not(nx<=np):
  print "Error dimensional \
   en las variables de entrada (magnetic)."
  return None

 In=zeros(nx,int)
 for i in range(nx):
  In[i]=argmin( sum(transpose((p-x[i,:])**2)) )
  for j in range(mx):
   _x[i,j] = p[In[i],j]
 return In.astype(int)

#privado def func_obj(x,p,M,Mf):
 In = magnetic(x,p)
```

```
 Mi = M[In]
 MM = concatenate((Mi,Mf),axis=0)
 ww = pesos(MM)
 return -func_pesos(ww,MM)

#privado def disegno(_m,puntos,M,MF):
 m = int(_m)

 # Se escoge un x0 aleatoriamente
 N  = puntos.shape[0]
 I  = permutation(N)[0:m]
 x0 = puntos[I]
 #------------------------------

 #Se obtienen las cotas de x0
 L = max(puntos[:,0])
 H = max(puntos[:,1])
 LB = 0*x0
 UB = LB+H
 UB[:,0] = L
 #---------------------------

 #Se obtiene el disegno optimo
 (x,F) = anneal(func_obj,x0,LB,UB,args=(puntos,M,MF))
 #---------------------------
```

```
 #Se obtienen los pesos del disegno
 I  = magnetic(x,puntos)
 MM = concatenate((M[I],MF),axis=0)
 ww = transpose(pesos(MM))[0]
 NF = MF.shape[0]
 II = concatenate((I,range(N,N+NF)),axis=1)
 #--------------------------------


 disegno = concatenate(([II],[ww]),axis=0)
 return disegno,F


#publico def put_puntos(r,L,H):
 n = int(ceil(sqrt(r)*L))
 m = int(ceil(sqrt(r)*H))
 rL = (L+0.0)/(n+1)
 rH = (H+0.0)/(m+1)
 N = (n+2)*(m+2)


 fichero = open('puntos','w')
 fichero.write(str(1)+' '+str(N)+' '+str(2)+'\n')
 for i in range(-1,n+1):
  for j in range(-1,m+1):
   fichero.write(str((i+1)*rL)+' '+str((j+1)*rH)+'\n')
 fichero.close()


#publico def put_puntosL(geometria,puntos):
```

```
 put_puntosLW(geometria,puntos,'puntosL')


#privado def put_puntosLW(geometria,puntos,salidaLP):
 L = max(puntos[:,0])+0.0
 H = max(puntos[:,1])+0.0


 puntosL = zeros(puntos.shape)
 N = puntos.shape[0]
 K = geometria.shape[0]
 for i in range(K):
  In = compress(( (H-(i+1)*H/K)<puntos[:,1] \
   )*( puntos[:,1]<=(H-i*H/K) ),range(N))
  if i==(K-1):
   In = compress(( 0<=puntos[:,1] )* \
    ( puntos[:,1]<=(H/K) ),range(N))


  n = len(In)
  for j in range(n):
   puntosL[In[j],0] = geometria[i,0]+ \
    puntos[In[j],0]*(geometria[i,1]-geometria[i,0])/L
   puntosL[In[j],1] = geometria[i,2]+((i+1-K)+ \
    puntos[In[j],1]*K/H)*(geometria[i,3]-geometria[i,2])


 fichero = open(salidaLP,'w')
 fichero.write(str(1)+' '+str(N)+' '+str(2)+'\n')
 for i in range(N):
```

```
  fichero.write(str(puntosL[i,0])+' '+str(puntosL[i,1])+'\n')

#publico def put_pesos(base):
 put_pesosW(base,'pesos')

#privado def put_pesosW(base,nombre):
 M = matrices(base)
 w = pesos(M)
 N = len(w)
 fichero = open(nombre,'w')
 fichero.write(str(1)+' '+str(N)+' '+str(1)+'\n')
 for i in range(N):
  fichero.write(str(w[i])+'\n')

#publico def put_disegno(m,puntos,base,baseF,Lp):
 put_disegnoW(m,puntos,base,baseF,Lp,'disegno')

#privado def put_disegnoW(m,puntos,base,baseF,Lp,salidaD):
 M  = matrices(base)
 MF = matrices(baseF)

 (Xi,F) = disegno(m,puntos,M,MF)

 In = Xi[0,:]
 In = In.astype(int)
 Di = Lp[In,:]
```

```
 DD = transpose(concatenate((transpose(Di),[Xi[1,:]]),axis=0))
 mFicheroW(asarray([DD]),salidaD)
```

A.2. El fichero pesos.py

```
from cvxopt import solvers
from cvxopt.base import matrix
from cvxopt.lapack import sysv

from scipy import
array,ones,zeros,eye,rand,transpose,trace,prod,sqrt,log from
scipy.linalg import det,norm

pesos_func_max = 1e100

def cholesky(Mo):
 M = matrix(Mo)

 (nM,mM) = M.size

 if not(nM==mM):
  print "Error en la variable de entrada (cholesky)."
  return None

 M = 0.5*(M+transpose(M))
```

```
 L = zeros([nM,nM],complex)

 for i in range(nM):
  for j in range(i+1):
   L[i,j] = M[i,j];
   for k in range(j):
    L[i,j] = L[i,j]-L[i,k]*L[j,k];
   if i==j:
    L[i,j] = sqrt(L[i,j]);
   else:
    L[i,j] = L[i,j]/L[j,j];
 return array(L.real)

def determinante(Mo):
 M = matrix(Mo)

 (nM,mM) = M.size

 if not(nM==mM):
  print "Error en la variable de entrada (determinante)."
  return None

 L = cholesky(M)
 I = range(nM)

 return prod(L[I,I])**2
```

```
 def func_pesos_completa(wo,Mo):
  M = array(Mo)
  w = matrix(array(wo))

  (N,nM,mM) = M.shape
  (nw,mw)   = w.size

  if not(nw==N) or not(nM==mM) or not(mw==1):
   print "Error dimensional en las \
variables de entrada (func_pesos_completa)."
   return None

  SwM = 0*M[0]
  for i in range(N):
   SwM = SwM+w[i,0]*M[i]
  a = determinante(SwM)

  if a>0:
   f = -log(a)
  else:
   f = pesos_func_max
  return f,matrix(SwM)

 def func_pesos(w,M):
  f,SwM = func_pesos_completa(w,M)
```

```
 return f


def pesos(Mo):
 M = array(Mo)


 (N,n,m) = M.shape


 def F(x=None, z=None):
  if x is None:
   return 0, (1.0/N)*matrix(ones([N,1]))


  f,SwM = func_pesos_completa(x,M)


  if f==pesos_func_max:
   Df = transpose(x) - (1.0/N)*matrix(ones([1,N]))
   H  = 1000*norm(Df)*matrix(eye(N))
  else:
   Df = matrix(zeros([1,N]))
   H  = matrix(zeros([N,N]))


   SwM = matrix(SwM)
   PM  = rand(N,9,9)*0


   for j in range(N):
    PMj = matrix(M[j])
    sysv(SwM,PMj)
```

```
   PM[j] = PMj


  for j in range(N):
   Df[j] = -trace(PM[j])
   PMj   = matrix(PM[j])
   for k in range(N):
    PMjMk  = PMj*matrix(PM[k])
    H[j,k] = trace(PMjMk)


 if z is None: return f, Df
 return f, Df, z*H

G = -matrix(eye(N))
h = matrix(zeros([N,1]))

A = matrix(ones([1,N]))
b = matrix(ones([1,1]))

solvers.options['show_progress']=False
w = solvers.cp(F,G=G,h=h,A=A, b=b)['x']

if w is None:
 print "M : "
 print M
return w
```

A.3. El fichero anneal.py

```
from registro import registro
from MLab import min,max,mean
from scipy import array,matrix,zeros,rand,inf, \
 log,exp,maximum,minimum,any

class Options(registro):
 initprob   = 0.95  #0.95
 cool       = 0.9   #0.9
 ncycs      = 10    #10
 ncyct      = 20    #20
 maxit      = 1e7   #1e7
 vmin       = 0.5   #1e-3
 ncyci      = 100   #100
 dispint    = 10    #inf
 tempVstep  = 50.0  #10.0 use private
 alpha      = 1.0   #0.33 use private

def anneal(func,xo,LBo,UBo,args=(),options=Options()):
 """ANNEAL Finds the constrained maximum of a function of
 several variables by simulated annealing.

 ANNEAL solves problems of the form:
   max F(X) subject to: LB <= X <= UB
```

(X,FVAL) = ANNEAL(func,x0,LB,UB) starts at x0 and finds a maximum X to the function func, subject to the lower and upper bounds LB and UB. func accepts input X and returns a scalar function value F evaluated at X. x0 may be a scalar, vector, or matrix.

(X,FVAL) = ANNEAL(func,x0,LB,UB,args) maximises with the args extra parameters to function.

(X,FVAL) = ANNEAL(func,x0,LB0,UB0,args,OPTIONS) maximises with the default optimisation parameters replaced by values in the structure OPTIONS. The list of options available:

initprob	initial probability of accepting downhill steps (default = 0.95)
Vstep	initial step size (default = UB-LB)
cool	cooling rate: temp = cool*temp (default = 0.9)
ncycs	number of cycles between changes in step size (default = 10)
ncyct	number of cycles between changes in temperature (default = 20)
maxit	maximum number of iterations (default = 1e7)
vmin	minimum change in step size (in space) (default = 1e-3)

```
  ncyci       number of initial cycles (to calculate temperature)
              (default = 100)
  dispint     number of temperature changes between displaying
              interim results (default = 5)
"""

X  = array(matrix(xo))
LB = array(matrix(LBo))
UB = array(matrix(UBo))

(m, n) = X.shape
Vstep  = UB-LB
b  = -inf
cr = 0
RJCT   = zeros(X.shape)

bestX  = array(X)
d  = func(X,*args)
best   = d
opt    = d

cnt    = 0

umax   = 0     # maximo salto de la funcion
omin   = inf       # minimo alcanzado
omax   = best      # maximo alcanzado
```

```
# Calculamos la temperatura inicial:
for i in range(options.ncyci):
 cnt = cnt+1
 for j in range(m):
  for k in range(n):
   tmpX      = array(X)
   tmpX[j,k] = min([ max([ tmpX[j,k]+(2*rand(1)[0]-1)* \
    Vstep[j,k],LB[j,k] ]), UB[j,k] ])

   d1 = func(tmpX,*args)

   if d1 == -inf:
    d1 = d

   if abs(d - d1) > umax:
    umax = abs(d - d1)

   if d1 > omax:
    omax  = d1
    bestX = array(tmpX)

   if d1 < omin:
    omin = d1

 if (cnt%options.dispint)==0:
```

```
  print "iteracion: ",cnt, " valor alcanzado: ", omax

if umax == 0 or umax >= 100:
 temp = 1e3
else:
 temp = -umax/log(options.initprob)
# se termino de calcular la temperatura inicial

if options.dispint<inf:
 print "umax: ",umax," omax: ",omax," omin: ",omin, \
  " omax-omin ",(omax-omin)

best = omax
opt  = omax

cnt = 0

while max( max( Vstep/(UB-LB) ) )>options.vmin and \
 cr<options.maxit:

 X = array(bestX)

 for cyct in range(options.ncyct):
  for cycs in range(options.ncycs):
   for j in range(m):
    for k in range(n):
```

```
cr   = cr + 1

tmpX      = array(X)
tmpX[j,k] = X[j,k] + (2*rand(1)[0]-1)*Vstep[j,k]

if tmpX[j,k] < LB[j,k] or tmpX[j,k] > UB[j,k]:
 d = -inf
else:
 d = func(tmpX,*args)

if d > opt:
 X   = array(tmpX)
 opt = d
else:
 if exp((d-opt)/temp)>rand(1)[0]:
  if X[j,k]==tmpX[j,k]:
   RJCT[j,k] = RJCT[j,k] + 1
  else:
   X   = array(tmpX)
   opt = d
 else:
  RJCT[j,k] = RJCT[j,k] + 1

if opt > best:
 bestX    = X
 best = opt
```

```
  # porcentaje de rechazos
  FRJCT = maximum(RJCT/options.ncycs,0.01)
  c_sa  = min([temp, options.tempVstep])
  Vstep = minimum( (1-FRJCT*options.alpha)* \
   Vstep+c_sa,Vstep) #modificacion Luis Saavedra
  RJCT  = zeros((m,n))

 temp  = temp*options.cool
 cnt   = cnt+1

 if (cnt%options.dispint)==0:
  print ""
  print "temp: ", temp
  print "Vstep :"
  print Vstep
  print "bestX: "
  print bestX
  print "best: ", best

# termino del while

if not(options.dispint==inf):
 if cr >= options.maxit:
  print "Maximum number of iterations reached."
 else:
```

```
  print "Cooling finished."
 print "Number of temperature changes: ", cnt

#print "bestX: ",bestX, " best: ", best
X  = bestX
FVAL   = best
return X,FVAL
```

A.4. Los ficheros adicionales

A.4.1. El fichero para el manejo de archivos de datos ficheros.py

```
from scipy import array, zeros

#private def mFila(s):
 return array(map(float,s.split()))

#public def mFichero(nombre):
 try:
  fichero = open(nombre,'r')
 except IOError:
  print 'Falta el fichero: ', nombre
  return

 dim = mFila(fichero.readline()).astype(int)
 if len(dim)!=3:
```

```
  print 'El archivo ',nombre, \
   ' esta corrupto en las dimensiones'
  return None

 M = zeros([dim[0],dim[1],dim[2]])+0.0
 for i in range(dim[0]):
  for j in range(dim[1]):
   aux = mFila(fichero.readline())
   if len(aux)==dim[2]:
    M[i,j] = aux
   else:
    print 'El archivo ',nombre, \
     ' esta corrupto en las filas ',len(aux),'=',dim[2],'?'
    print 'La matriz ',i,' fallo en la fila ',j
    return

 fichero.close()
 return M

#public def mFicheroW(A,nombre):
 (N,n,m) = A.shape

 fichero = open(nombre,'w')
 fichero.write(str(N)+' '+str(n)+' '+str(m)+'\n')
 for i in range(N):
  for j in range(n):
```

```
    for k in range(m):
     fichero.write(str(A[i][j][k])+' ')
    fichero.write('\n')
 fichero.close()
```

A.4.2. El fichero para el manejo de registros registro.py

```
import warnings

class metaMetaBunch(type):
  def __new__(cls, classname, bases, classdict):
    def __init__(self, **kw):
      for k in self.__dflts__: setattr(self, k, self.__dflts__[k])
      for k in kw: setattr(self, k, kw[k])

    def __repr__(self):
      rep = [ '%s=%r' % (k, getattr(self, k)) \
       for k in self.__dflts__
              if getattr(self, k) != self.__dflts__[k]]
      return '%s(%s)' % (classname, ', '.join(rep))

    # Build the newdict that we'll use as
    # class-dict for the new class
    newdict = {'__slots__':[], '__dflts__':{}, \
     '__init__':__init__, '__repr__':__repr__}
```

```
    for k in classdict:
      if k.startswith('__'):
        # Special methods: copy to \emph{newdict},
        # warn about conflicts.
        if k in newdict:
          warnings.warn("Can't set attr %r in bunch-class %r" \
           % (k, classname))
        else:
          newdict[k] = classdict[k]
      else:
        # Class variables, store name in
        # \texttt{__}\emph{slots}\texttt{__} and name and value
        # as an item in \texttt{__}\emph{dflts}\texttt{__}.
        newdict['__slots__'].append(k)
        newdict['__dflts__'][k] = classdict[k]
    # Finally delegate the rest of the work to
    # \emph{type}.\texttt{__}\emph{new}\texttt{__}
    return type.__new__(cls, classname, bases, newdict)


class registro(object):
  __metaclass__ = metaMetaBunch
```