UNIVERSIDAD POLITÉCNICA DE MADRID

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA Y DISEÑO INDUSTRIAL

Grado en Ingeniería Eléctrica

TRABAJO FIN DE GRADO

Diseño optimizado de sistemas de puesta a tierra en subestaciones eléctricas

Bogurad Barañski Barañska

Tutor: Gabriel Asensio Madrid
Departamento: matemática aplicada a la ingeniería industrial

Madrid, Junio, 2025

UNIVERSIDAD POLITÉCNICA DE MADRID

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA Y DISEÑO INDUSTRIAL

Grado en Ingeniería Eléctrica

TRABAJO FIN DE GRADO

Diseño optimizado de sistemas de puesta a tierra en subestaciones eléctricas

Firma Autor

Firma Tutor

TÌtulo: Diseño optimizado de sistemas de puesta a tierra en subestaciones eléctricas
Autor: Bogurad Barañski Barañska
Tutor: Gabriel Asensio Madrid

EL TRIBUNAL

Presidente:

Vocal:

Secretario:

Realizado el acto de defensa y lectura del Trabajo Fin de Grado el día ..... de ..... de ... en ....., en la Escuela Técnica Superior de Ingeniería y Diseño Industrial de la Universidad Politécnica de Madrid, acuerda otorgarle la CALIFICACIÓN de:

VOCAL

SECRETARIO PRESIDENTE

Agradecimientos

Agradezco en primer lugar a mi tutor, Gabriel Asensio, por haberme brindado la oportunidad de realizar este trabajo, que me ha permitido profundizar en dos áreas que siempre he deseado explorar con mayor detalle. La primera es la optimización matemática, un campo que me apasiona profundamente dentro de las matemáticas aplicadas; y la segunda, el cálculo mediante elementos finitos aplicado al ámbito eléctrico, donde he podido comprender cómo se implementa esta metodología y cómo permite abordar y resolver problemas complejos.

También quiero dar las gracias a mis compañeros, por estar ahí en los momentos más duros, compartir conmigo el "sufrimiento" de esta etapa y por animarnos mutuamente a seguir adelante. Su compañía y apoyo han hecho todo este proceso mucho más llevadero.

Resumen

Este Trabajo Fin de Grado propone un enfoque integral para el diseño optimizado de sistemas de puesta a tierra en subestaciones eléctricas, orientado a maximizar la seguridad y la eficiencia en el cálculo de potenciales y resistencias. En MATLAB se han desarrollado rutinas de geometría computacional capaces de procesar y refinar mallas de electrodos con gran robustez, junto con un módulo de cálculo basado en el método de simulación de cargas (CSM) que permite obtener tanto la resistencia del electrodo como el potencial eléctrico en cualquier punto del dominio. Para calcular la tensión de contacto se implementa un algoritmo híbrido, que combina un algoritmo genético con un refinamiento posterior mediante gradiente conjugado, consiguiendo reducir significativamente los tiempos de calculo. Finalmente, toda esta funcionalidad se agrupa en una interfaz gráfica de usuario que agiliza el flujo de trabajo desde la definición de la malla hasta el análisis de resultados y la optimización final.

Palabras clave: Puestas a Tierra, Tensión de Contacto, Método de Simulación de Cargas (CSM), Geometría Computacional, Optimización híbrida, Algoritmo Genético, Gradiente Conjugado, MATLAB.

Abstract

This Bachelor's Thesis proposes a comprehensive approach for the optimized design of grounding systems in electrical substations, aimed at maximizing safety and efficiency in the calculation of potentials and resistances. Computational geometry routines have been developed in MATLAB, capable of processing and refining electrode meshes with high robustness, along with a calculation module based on the Charge Simulation Method (CSM) that enables the determination of both the electrode resistance and the electric potential at any point in the domain. To calculate the touch voltage, a hybrid algorithm is implemented, combining a genetic algorithm with subsequent refinement using the conjugate gradient method, significantly reducing computation times. Finally, all this functionality is integrated into a graphical user interface that streamlines the workflow from mesh definition to result analysis and final optimization.

Keywords: Grounding Systems, Touch Voltage, Charge Simulation Method (CSM), Computational Geometry, Hybrid Optimization, Genetic Algorithm, Conjugate Gradient, MATLAB.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

1.1. Motivación del proyecto

El potencial eléctrico que adquiere el suelo en el entorno de una subestación eléctrica cuando se inyecta una intensidad debido a una falta o a la caída de un rayo depende en gran medida de la geometría de la malla del sistema de puesta a tierra. Por ello, para garantizar la seguridad de las personas ante contactos indirectos, es necesario tener un diseño adecuado de las puestas a tierra.

Este potencial se calcula mediante el método de simulación de cargas (CSM), un método numérico tipo diferencias finitas para resolver, de forma aproximada, las ecuaciones de Maxwell en régimen estacionario. Los potenciales generados presentan múltiples y rápidas oscilaciones, lo que hace que su optimización sea todo un reto.

Como resultado de los motivos mencionados previamente, este trabajo de fin de grado tiene como objetivo agilizar el cálculo de los puntos críticos, donde la derivada de la función potencial es nula, y de los puntos singulares, donde la derivada de la función potencial no está definida. Por lo tanto, se busca desarrollar métodos numéricos de rápida convergencia (pocas iteraciones) para lograr un diseño de la puesta a tierra más eficiente.

1.2. Objetivos

Para realizar el proyecto, se proponen los siguientes objetivos:

- Desarrollar herramientas de geometría computacional para el preproceso de los datos.
- Calcular la resistencia de puesta a tierra y el potencial eléctrico en un punto del suelo mediante el método de simulación de cargas (CSM).
- Optimizar el potencial mediante un algoritmo mixto genético-gradiente conjugado y calcular la tensión de contacto y de paso de la puesta a tierra.
- Obtener una herramienta informática con interfaz gráfica de usuario (GUI) que implemente lo expuesto anteriormente.

1.3. Herramientas utilizadas

1.3.1. MATLAB [1]

Dado que gran parte de los desarrollos previos se han realizado en MATLAB, se ha decidido continuar utilizando esta potente herramienta para el cálculo numérico en este proyecto. Además, MATLAB ofrece una amplia cantidad de librerías para la optimización de funciones, lo que facilita el desarrollo.

1.3.2. LaTex [2]

Se ha preferido el uso de LaTeX debido a la facilidad que ofrece para el maquetado de textos, superando a otras herramientas de elaboración de documentos. Además, LaTeX permite crear figuras vectorizadas, representar correctamente ecuaciones y ubicar adecuadamente figuras, tablas y bibliografía.

1.3.3. TikzMaker [3]

Esta herramienta permite crear figuras vectorizadas de LaTeX mediante el paquete de circuitikz. Su principal ventaja radica en la interfaz gráfica que proporciona y en la facilidad para elaborar figuras.

1.3.4. matlab2tikz [4]

Este repositorio de GitHub permite exportar las figuras hechas en MATLAB directamente a LaTeX de manera que no haya pérdidas en la calidad de la figura. Para ello transforma la figura en código de la librería Tikz de LaTeX.

1.4. Estructura del documento

El documento se estructura en varios capítulos donde se describen los distintos pasos realizados en este estudio. No obstante, antes de detallar el contenido de cada capítulo, es necesario aclarar que el trabajo se relaciona con tres disciplinas bien diferenciadas: el cálculo y diseño de puestas a tierra, el desarrollo de herramientas de geometría computacional y la implementación de métodos de optimización.

A continuación y para facilitar la lectura del documento, se detalla el contenido de cada capítulo:

- En el capítulo 1 se realiza una introducción.
- En el capítulo 2 se hace un repaso de desarrollos anteriores teniendo en cuenta las distintas disciplinas involucradas en el desarrollo del proyecto.
- En el capítulo 3 se desarrollan los fundamentos matemáticos del proyecto.
- En el capítulo 4 se describe la implementación de los algoritmos.
- En el capítulo 5 se exponen y analizan los resultados obtenidos en el capítulo anterior.
- En el capítulo 6 se realiza la conclusión.

Capítulo 2

Estado del arte

En este capítulo, se hace un resumen de la bibliografía que ha servido como base para la elaboración del proyecto.

2.1. Cálculo y diseño de puestas a tierra

2.1.1. Instrucciones Técnicas Complementarias [5]

2.1.1.1. Introducción

El reglamento de alta tensión, junto con las instrucciones técnicas complementarias, es la referencia para los requisitos de calidad y seguridad en las instalaciones eléctricas de alta tensión (frecuencia de servicio inferior a 100 Hz, cuya tensión nominal eficaz entre fases sea superior a 1 kV). En particular, para este trabajo, son pertinentes las Instrucciones Técnicas Complementarias ITC-RAT 1 e ITC-RAT 13, que desarrollan las definiciones básicas y las instalaciones de puesta a tierra, respectivamente.

2.1.1.2. ITC-RAT 01. Terminología

En esta Instrucción Técnica Complementaria se definen los conceptos técnicos necesarios para comprender el reglamento. En particular, para este trabajo, algunas definiciones relevantes, se sitúan en el *Apéndice A*.

2.1.1.3. ITC-RAT 13. Instalaciones de puesta a tierra

En esta Instrucción Técnica Complementaria se detallan los requisitos que debe cumplir una puesta a tierra para ser considerada segura desde el punto de vista de las tensiones de paso y de contacto. Aunque el objetivo principal de este trabajo de fin de grado no es enfocar en la seguridad, sino en la optimización del potencial en la zona de estudio, es fundamental comprender la importancia de dicho potencial en relación con la seguridad de las personas.

Para ello, el reglamento define la siguiente gráfica de la tensión de contacto a la que puede estar sometido el cuerpo humano entre los pies y las manos en función del tiempo que dure la corriente de falla (para la tensión de paso, el reglamento considera una tensión 10 veces mayor). La curva se determina bajo los siguientes supuestos:

- La corriente circula entre pies y manos.
- Solo se considera como impedancia el cuerpo humano, sin protecciones adicionales.
- La probabilidad de fibrilación ventricular es del 5 %.
- La impedancia del cuerpo humano tiene una probabilidad del 50 % de ser menor en valor al considerado.

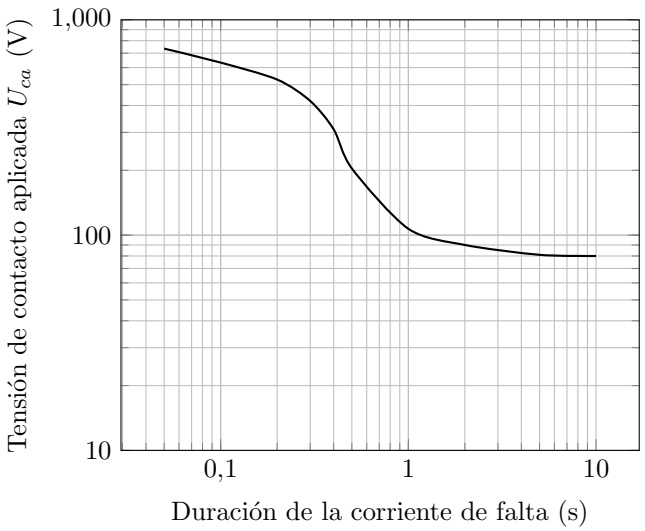


Figura 2.1: Tensión de contacto aplicada admisible en función de la duración de la corriente de falta.

Por lo tanto, a partir del reglamento se obtiene un criterio para determinar si una puesta a tierra tiene un diseño adecuado. Las tensiones aplicadas durante una falla o defecto a tierra deben cumplir con lo expuesto en la gráfica anterior. Es importante destacar que esta evaluación depende en gran medida de los óptimos de la función potencial

No obstante, las puestas a tierra descritas en el reglamento son muy simples y no contemplan electrodos con geometrías más complejas, los cuales son necesarios en subestaciones para cumplir con las tensiones reglamentarias. De ahí surge la importancia de desarrollar la herramienta propuesta.

2.1.2. Sistemas de puestas a tierra en instalaciones de alta tensión. Diseño, cálculo y verificación [6]

2.1.2.1. Introducción

En ausencia de una fuente de referencia actualizada a la normativa para el cálculo y proyección de sistemas a puestas a tierra, se publicó este libro en 2015 para actualizar el anterior documento de referencia UNESA de 1989. A lo largo de sus capítulos, se desarrollan los pasos para el correcto cálculo, diseño y verificación de una instalación de puesta a tierra. En particular, para este proyecto, los capítulos 1 y 2 son de especial relevancia, pues en ellos se desarrolla la base física del cálculo y se discute la validez de los resultados en función del método de cálculo.

2.1.2.2. Capítulo 1. Fundamentos físicos de los sistemas de puesta a tierra

En los cálculos se abordan exclusivamente situaciones en régimen estacionario (válidos también a frecuencia de la red). Aunque se realice una introducción al cálculo en régimen transitorio en el capítulo, no es objeto del trabajo realizado. Por tanto, el planteamiento general del problema es el de un electrodo metálico de geometría genérica (ver figura 2.2), al cual se inyecta una corriente, I, que se libera en el medio.

- Se considera un modelo de **suelo homogéneo** con resistividad constante, ρ y con electrodos ideales.

Para otros modelos más realistas"se pueden consultar:

- Para un modelo de suelo inhomogéneo multicapa, el método CSM se sustituye por ESCD (distribuciones equivalentes de carga superficial) [11].
- Si el modelo de suelo inhomogéneo multicapa incluye capas no planas, se utiliza BEM (método de elementos de contorno) en lugar de CSM [12].
- Si se considera la corrosión de los electrodos, se utiliza un método basado en el siguiente artículo [13].
- Si se tiene en cuenta la resistencia interna y térmica de los electrodos (efecto Joule), se utiliza un método basado en el siguiente artículo [14].

En cuanto al potencial, se considera que en cualquier punto del electrodo, V_e, es constante y que el potencial en un punto alejado una distancia considerable, V_∞, es nulo. A continuación, el libro establece las premisas que se asumen para realizar el cálculo:

1. La conducción queda descrita por un campo vectorial, $\vec{J}(\vec{r})$ y una densidad de corriente (A/m^2) con las siguientes propiedades:
   a) El campo vectorial, $\vec{J}(\vec{r})$, satisface la ecuación de continuidad.
   b) En flujos estacionarios:
   $$\vec{\nabla} \cdot \vec{J} = 0 \qquad (2.1)$$
   c) La intensidad liberada por electrodo activo es (si se cuentan todas las contribuciones, la integral da como resultado cero):
   $$I = \oiint_S \vec{J} \cdot d\vec{S} \qquad (2.2)$$
2. La corriente se establece por la diferencia de potencial entre el electrodo y el terreno. Donde el campo eléctrico asociado, $\vec{E}$, y la densidad de corriente, $\vec{J}$, están relacionadas entre sí para medios homogéneos mediante la conductividad, σ.

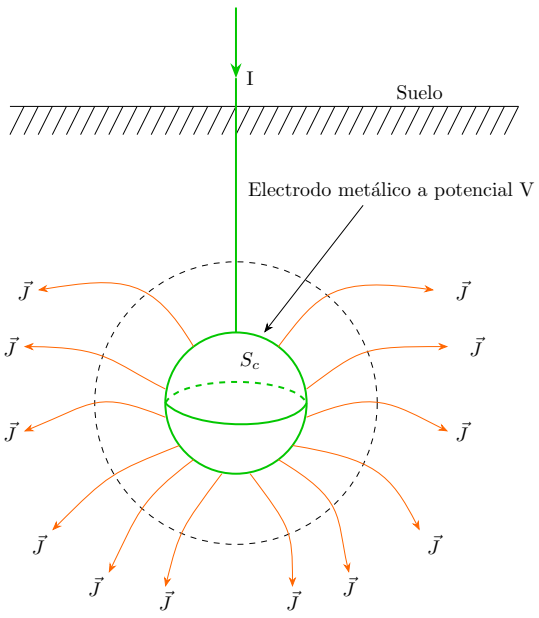


Figura 2.2: Electrodo enterrado en un medio semiinfinito.

Por tanto, a partir del modelo, se obtiene la ecuación de Laplace:

$$\left.\begin{array}{c}\vec{J}=\sigma\vec{E}\\ \vec{\nabla}\cdot\vec{J}=0\\ \vec{E}=-\vec{\nabla}V\end{array}\right\}\vec{\nabla}\cdot\vec{J}=\sigma\vec{\nabla}\cdot\vec{E} \tag{2.3}$$

$$\sigma\vec{\nabla}\cdot\vec{E}=-\sigma\vec{\nabla}^2V=0\rightarrow\vec{\nabla}^2V=0 \tag{2.4}$$

Además de cumplir la ecuación de Laplace, $\vec{\nabla}^2V=0$, se debe cumplir que:

1. El potencial, V_e, es constante en el electrodo. Esta condición, implica que el campo eléctrico debe ser perpendicular en la superficie del electrodo, y por tanto, se puede emplear el método de simulación de cargas que se explicará más adelante.
2. $\vec{J}$ solo puede existir en el medio natural y es paralelo a la superficie del terreno en sus alrededores. Este requisito permite sustituir la condición de frontera impuesta por un electrodo imagen, véase la figura 2.3.

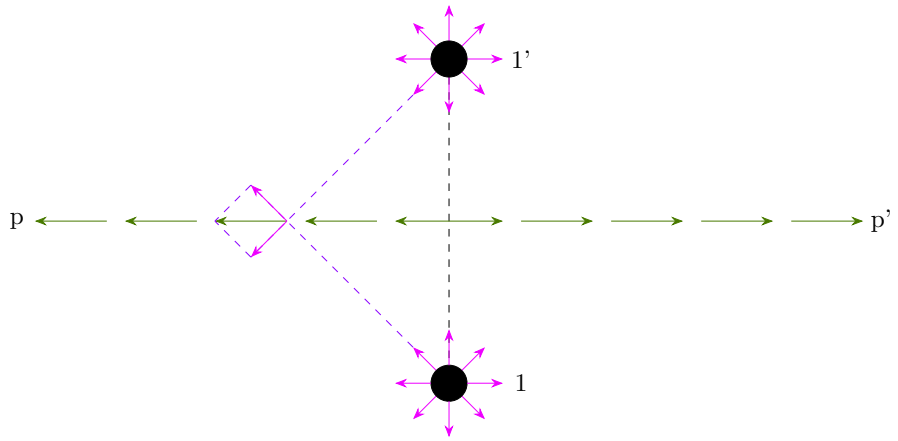


Figura 2.3: Representación del vector $\vec{J}$ creado por una carga, 1 y su imagen, $1'$.

A continuación, en el libro se especifica como han de calcularse los parámetros mencionados por el reglamento:

1. Perfil de potenciales: potenciales que se generan en relación con tierra, $V_\infty=0$.
2. El potencial adquirido por el electrodo, V_e.
3. El valor de la tensión de contacto en un punto del terreno $P0$: diferencia de potencial entre cualquier superficie metálica conectada al electrodo, V_e y un punto cualquiera del terreno a potencial V_{P0}.

$$V_{contacto}=V_e-V_{P0} \tag{2.5}$$

4. El valor de la tensión de paso en un punto del terreno $P0$: máximo del valor absoluto de las diferencias de potencial entre $P0$ y todos los puntos $P1$ que distan un metro de $P0$.

$$V_{paso}=max\left(V_{P0}-V_{P1}\right) \tag{2.6}$$

5. La resistencia de puesta a tierra: definida como la resistencia del medio conductor cuando se inyecta una corriente al electrodo.

$$R_{pat} = \frac{V_e}{I} \tag{2.7}$$

2.1.2.3. Cálculo del perfil de potenciales

En el libro se describen procedimientos de cálculo analíticos para electrodos sencillos. No obstante, dado que los electrodos a analizar en este estudio son complejos, es necesario recurrir a métodos numéricos. Los dos métodos mencionados en el libro son el método de Howe y el CSM (Método de Simulación de Cargas).

- Método de Howe: es un método simplificado de cálculo que resulta válido para geometrías sencillas. Se usa en la metodología UNESA [15] que sirve como base para los valores del reglamento . No obstante, debido a sus suposiciones, como se desarrolla en el capítulo 2 del libro, se puede obtener un error elevado en el cálculo. Las suposiciones del método son:

  1. El potencial creado por un electrodo j en cualquier punto de un electrodo i, V_{ji}, es constante e igual al valor medio.
  2. La densidad de corriente, λ, es constante y se define como el cociente de la intensidad que atraviesa el electrodo y su longitud.

$$\lambda = \frac{I}{L} \tag{2.8}$$

  3. La densidad de corriente es igual para todos los electrodos. Por tanto, la densidad de corriente equivale a la intensidad de corriente dividida entre la longitud total de la puesta a tierra (se debe tener en cuenta el electrodo imagen).

- CSM: es un método numérico clásico [16] [17] que se basa en suponer que en el interior del electrodo hay puntos de corriente distribuidos de tal manera que consiguen que la superficie del electrodo sea equipotencial. El detalle del cálculo se explicará en la *Subsección 3.1.1.*

2.1.2.4. Capítulo 2. Caracterización de los electrodos de puesta a tierra

En este capítulo se muestra cómo obtener los parámetros eléctricos mencionados anteriormente mediante aplicaciones prácticas de los métodos de Howe y CSM. Además, a continuación se introduce una serie de coeficientes que sirven como parámetros para validar el diseño de la puesta a tierra de la subestación:

1. Coeficiente de resistencia:

$$k_r = \frac{V_e}{\rho I_{falta}} \left[\frac{\Omega}{\Omega m}\right] \tag{2.9}$$

2. Coeficiente de tensión de paso máxima:

$$k_p = \frac{V_p}{\rho I_{falta}} \left[\frac{\Omega}{\Omega m}\right] \tag{2.10}$$

3. Coeficiente de tensión de contacto máxima:

$$k_c = \frac{V_c}{\rho I_{falta}} \left[\frac{\Omega}{\Omega m}\right] \tag{2.11}$$

En la práctica, en lugar de dividir por la resistividad del terreno y la intensidad de falta, se suponen unitarias y simplemente se aplican para obtener el valor de las tensiones.

A continuación, se ilustra mediante una tabla la diferencia media entre los coeficientes obtenidos entre ambos métodos para distintas configuraciones. No obstante, para no incluir todas las tablas, se calcula un promedio de la diferencia relativa.

$$\varepsilon(\,\%) = \frac{|k_{Howe} - k_{CSM}|}{k_{CSM}} \tag{2.12}$$

	$\overline{\varepsilon}(\,\%)$
k_r	2,00
k_p	8,71
k_c	33,24

Tabla 2.1: Tabla comparativa entre los coeficientes geométricos entre el método de Howe y el CSM. Estudio de una sola pica, picas en hilera, anillo sin picas, anillo con cuatro picas, anillo con ocho picas, doble anillo sin picas , doble anillo con cuatro picas enterrado a distinta profundidad, doble anillo con ocho picas enterrado a distinta profundidad y una malla. Se observan desviaciones mayores en geometrías más complejas y sin picas.

2.1.3. IEEE Guide for Safety in AC Substation Grounding [7]

Este documento es la referencia internacional en cuanto a la normativa sobre puestas a tierra en subestaciones de corriente alterna. Un sistema de puesta a tierra seguro debe cumplir con la capacidad de soportar corrientes eléctricas en condiciones de falta sin exceder los límites operativos y asegurar que cualquier persona en la cercanía de las instalaciones puestas a tierra no corra peligro de recibir una descarga eléctrica.

A continuación, la normativa muestra una figura que da una idea del potencial que se crea en una malla rectangular que sirve como referencia para dar validez a la forma de los potenciales obtenidos.

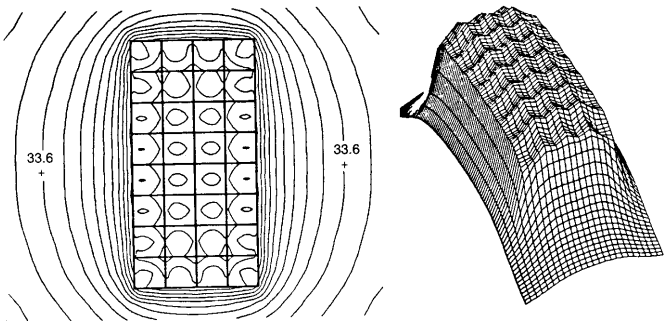


Figura 2.4: Líneas equipotenciales creadas por una malla rectangular según la norma IEEE-80.

Cabe destacar también que, dado que este documento desarrolla temas similares a las fuentes mencionadas anteriormente, se omite la información repetida.

2.1.3.1. Principales consideraciones de diseño

Según la norma, las puestas a tierra están formadas por una malla de electrodos con picas colgando de ellos. Este hecho permitirá realizar varias simplificaciones en el preprocesamiento de los datos. Las razones detrás de esta geometría son las siguientes:

1. En subestaciones, un solo electrodo por si mismo es insuficiente. Cuando varios electrodos se conectan entre sí y a las partes conductoras, se acaba formando inevitablemente una malla.
2. Si la conductividad del terreno es demasiado baja, se emplean picas para estabilizar los gradientes de potencial creados en la superficie.

Teniendo en cuenta las anteriores consideraciones, en la normativa se proponen unas guías generales como punto de partida:

1. Un bucle continuo de conductor debería rodear el perímetro para encerrar el máximo área posible. De esta manera, se evitan grandes densidades de corriente y así, reducir los gradientes de potencial. Además, se reduce la resistencia de puesta a tierra.
2. Normalmente se distribuyen los conductores en líneas paralelas.
3. Una malla suele estar enterrada por lo menos a 0,5m de profundidad y con los conductores espaciados entre 3 y 7m.

2.1.3.2. Diseño de sistemas de puesta a tierra

A continuación se describen los pasos que han de seguirse para el diseño según la normativa:

1. Determinar el área donde se pondrá la puesta a tierra y medir la resistividad del terreno.
2. Se determina el diámetro del conductor en función de la corriente de falta y del tiempo que tardan en actuar las protecciones.
3. Se determina la tensión de contacto y de paso admisibles.
4. Se crea un diseño preliminar incluyendo el conductor en bucle y los conductores de los dispositivos.
5. Se calcula la resistencia de la puesta a tierra. Por la complejidad de las puestas a tierra a estudiar se emplean métodos numéricos.
6. Se calcula la corriente que circula por el sistema de puesta a tierra en función de la corriente de falta máxima.
7. Se comprueba que a esta corriente, I_G, no se supera la tensión de contacto admisible mediante la siguiente ecuación $R_{pat} \times I_G < V_{contacto}$. En caso de que no se supere esta tensión el análisis ha finalizado.

8. Se comprueba si la tensión máxima entre un punto del terreno es menor que la tensión de contacto admisible. Si no se cumple se debe modificar el diseño.
9. Se comprueba si la tensión de paso máxima es menor que la tensión de paso admisible. Si no se cumple se debe modificar el diseño.
10. Una vez modificado el diseño se vuelve al paso 5.

Para aclarar los pasos anteriores la normativa ofrece el siguiente diagrama:

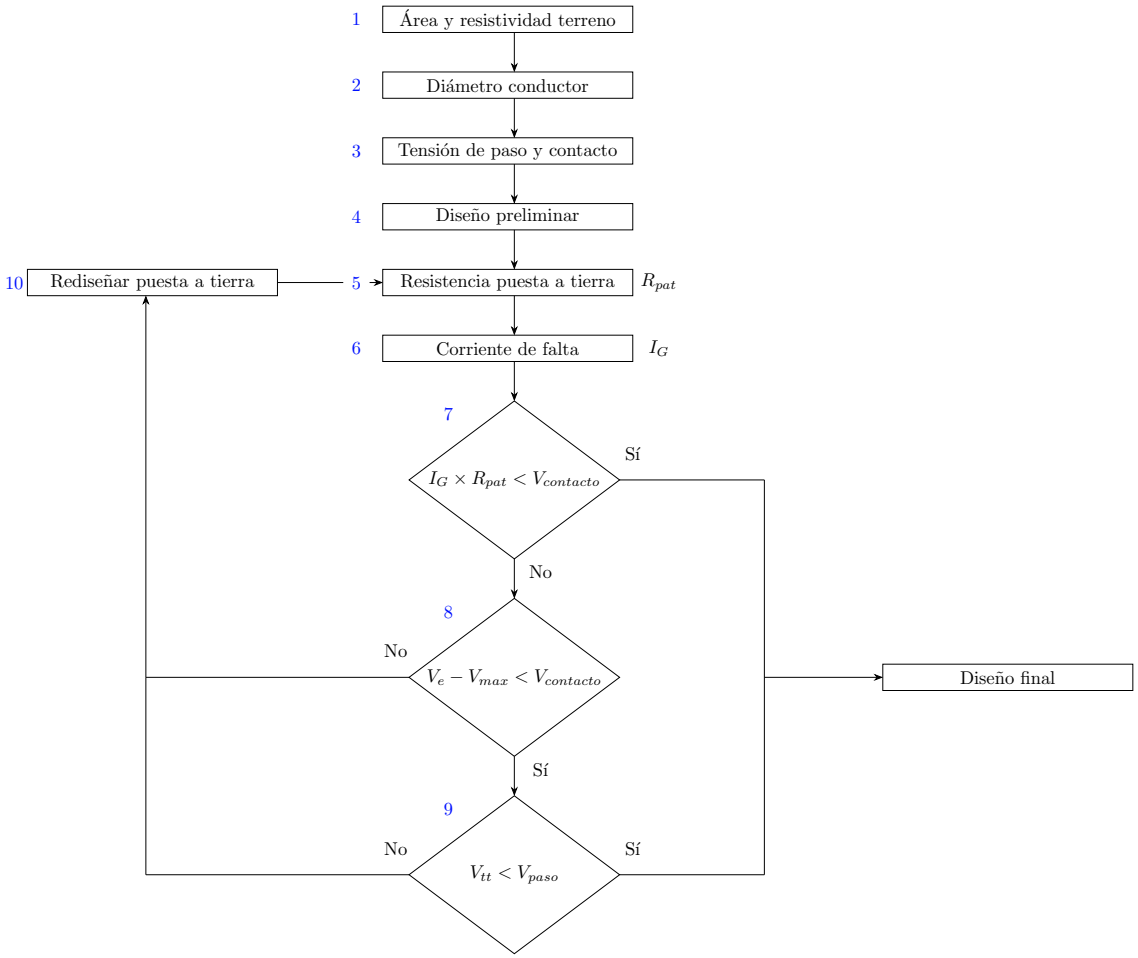


Figura 2.5: Diagrama de flujo para el diseño de una puesta a tierra.

Por último, aunque la normativa incluye numerosas expresiones para estimar los parámetros en función de la malla a emplear, se permite el uso de métodos numéricos siempre que se asegure un modelo equivalente. Esto reafirma la importancia de optimizar el cálculo mediante el método CSM.

2.2. Métodos de procesamiento geométrico.

En el proyecto se han empleado numerosos algoritmos para el procesamiento geométrico que se desarrollan en la sección 3.2.1 de los cuales se puede destacar la literatura existente sobre la descomposición en polígonos convexos, definidos a continuación.

2.2.1. Optimal Convex Decompositions [8]

Este libro presenta una introducción a los conceptos básicos a tener en cuenta a la hora de descomponer un polígono en sus componentes básicas. Para ello, se considera el siguiente problema:

- Dado un polígono P, ¿Cuál es la mínima cantidad de polígonos convexos que forman P?

Este problema en la literatura recibe el nombre de optimal convex decomposition, OCD, que en castellano significa descomposición óptima convexa. A lo largo de este artículo se desarrollan las ideas básicas de la implementación del algoritmo y la demostración de que el problema es resoluble en tiempo polinomial.

2.2.1.1. Definiciones geométricas

Antes de plantear los algoritmos para resolver el problema es necesario definir algunos de los conceptos fundamentales:

1. Convexo: se habla de toda región de tal forma que escogidos dos puntos cualesquiera contenidos en ella se pueden unir mediante una línea también contenida en la región.
2. Vértice: punto de intersección entre dos lados consecutivos que delimitan la región a estudiar.
3. Punto de concavidad o notch: todo vértice que tiene un ángulo interno mayor a 180°.
4. Punto de Steiner: puntos introducidos durante la resolución al problema como nuevos vértices en los polígonos.

En base a estas definiciones, para dividir un polígono en regiones convexas es necesario eliminar todos los puntos de concavidad, c. Por lo tanto, como para eliminar estos puntos se deben trazar líneas que eliminen hasta dos puntos de concavidad a la vez, el número de polígonos resultantes estará comprendido entre:

$$\frac{c}{2} + 1 \leq N_{convexos} \leq c + 1 \tag{2.13}$$

Como eliminar los puntos de concavidad tiene varias posibles soluciones, en el artículo se proponen estructuras en forma de X e Y para intentar alcanzar las descomposiciones convexas de mínima cardinalidad.

Un primer acercamiento al problema consiste en realizar una descomposición "sencilla"de tal manera que se recorren los puntos de concavidad y, mediante una recta, se interseca una arista del polígono creando un punto de Steiner. De esta manera, se asegura que todas las líneas dibujadas permanecen dentro del polígono y que ningún segmento conecta dos puntos de concavidad. No obstante, este procedimiento tiene la problemática de generar exactamente c+1 polígonos.

A continuación, en el artículo se desarrollan las descomposiciones en X e Y, así como la demostración de que resuelven la descomposición óptima convexa. Estos patrones, son estructuras que ayudan a eliminar los notches al introducir segmentos

adicionales que forman árboles en el grafo de la descomposición. Además, se desarrolla que cualquier patrón en X, salvo aquellos con vértices de grado 4, es convertible en un patrón en Y que tiene la ventaja de no tener notches en su interior.

Por tanto, el proceso de eliminar los puntos de concavidad consiste en introducir un patrón en forma de X e Y para iterativamente repetir el proceso hasta eliminar todos los puntos de concavidad en los subpolígonos. De esta manera, se logra obtener una descomposición en $c + 1 - p$ convexos, donde p es el número de patrones.

2.2.1.2. Algoritmo mediante selección de patrones

En esta sección del artículo se describe un algoritmo en tiempo polinomial para descomponer en convexos utilizando patrones en X e Y. Para ello, se estudia la existencia de patrones que conecten k notches en tiempo polinómico.

El método se basa en la construcción de un patrón en forma de Y, que resulta en un conjunto desconectado de tres árboles (véase la Figura 2.6). El algoritmo de descomposición utiliza una estrategia dinámica para encontrar el máximo número de patrones compatibles y, posteriormente, divide el problema en subproblemas más pequeños. Estos subproblemas se resuelven de manera recursiva, creando subpolígonos hasta que todos los notches se eliminan.

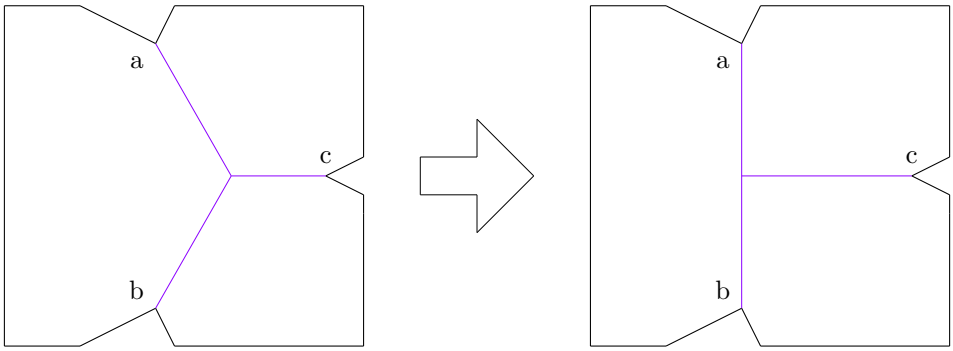


Figura 2.6: Eliminación de 3 notches mediante un patrón en Y refinado.

Una vez obtenidos los posibles subpolígonos mediante varias funciones auxiliares descritas en el artículo se itera para ensamblar los patrones hasta obtener una descomposición convexa válida. Este proceso acaba generando varios candidatos de los cuales se selecciona aquel que introduzca menor cantidad de nuevos polígonos.

No obstante, este algoritmo tiene el problema de generar una gran multitud de patrones que acaban provocando un coste temporal de computación elevado. Por ello, para evitarlo se justifica la introducción de patrones donde el punto de concavidad sea colineal con uno de sus bordes lo que permite emplear una menor cantidad de patrones.

2.2.1.3. Conclusiones

En el artículo se consigue demostrar que la descomposición mediante puntos de Steiner tiene una complejidad temporal $O(p + q)$ donde p es número de vértices

mientras que q es el número de puntos de concavidad. Por otro lado, el algoritmo propuesto mediante patrones tiene una complejidad temporal $O(n + q^3)$ que tiene como ventaja crear menos regiones.

Finalmente, se discute que, dependiendo de cuál sea la prioridad principal del problema, se debe elegir entre uno de los algoritmos, ya que la penalización por no obtener la cantidad mínima de polígonos tiene una cota superior de dos veces este número de subregiones.

2.2.2. Polygon Decomposition [9]

Este artículo revisa el estado del arte de los algoritmos para la descomposición de polígonos y sus principales aplicaciones en casos bidimensionales.Para ello, se definen los polígonos básicos empleados para estas descomposiciones, de los cuales se pueden emplear los siguientes para representar plantas de subestaciones:

- **Triangulación**: este método consiste en subdividir el polígono original en triángulos. Sin embargo, debido a su relevancia, la triangulación se ha convertido en un área de estudio en si misma, y por lo tanto, no se aborda en este artículo.
- **Descomposición en convexos**: este método consiste en subdividir el polígono original en subpolígonos convexos. Para polígonos con agujeros se demuestra que el problema de descomponer el polígono en la mínima cantidad de subpolígonos es NP-complejo mientas que si no contiene agujeros el problema es resoluble en tiempo polinomial. Además, en el artículo se mencionan algoritmos donde la subdivisión tiene como objetivo encontrar la descomposición con menor longitud de arista.
- **Subpolígonos monótonos**: este método consiste en subdividir el polígono de tal manera que, en cada subpolígono, las cadenas de vértices proyectadas sobre una recta aparecen en el mismo orden que en la cadena. Resolver el problema con agujeros es NP-completo.
- **Otros polígonos**: este método normalmente consiste en subdividir el polígono en regiones trapezoidales aunque también se emplean otro tipo de cuadriláteros. Este tipo de descomposición es de especial relevancia en polígonos ortogonales definidos a continuación.

En la segunda mitad del artículo, se desarrolla el estado del arte de la descomposición de polígonos ortogonales. Un polígono ortogonal está formado únicamente por líneas verticales y horizontales. Estos algoritmos son útiles en aplicaciones como el procesamiento de imágenes o la fabricación de microchips. No obstante, en este trabajo no se desarrolla este apartado debido a que la geometría de una puesta a tierra no puede asegurarse ortogonal, ya que las delimitaciones del terreno no siempre lo son.

2.3. Algoritmos de optimización. Optimization Methods: From Theory to Design [10]

Este libro sirve como punto de partida para conocer los diferentes algoritmos de optimización, así como para la aplicación práctica de estos a distintos problemas de ingeniería. Aunque el libro discute diversas técnicas de optimización, como el diseño de experimentos, la metodología de superficie de respuesta, la optimización determinista, la optimización estocástica y el diseño robusto en ingeniería, solo se estudian los capítulos centrados en la optimización determinista y la optimización estocástica, dado que el problema a resolver ya está adecuadamente descrito.

No obstante, antes de pasar a describir los fundamentos de los distintos algoritmos, en el libro se describe la secuencia de pasos para resolver adecuadamente un problema de optimización tal y como se describe en la figura 2.7.

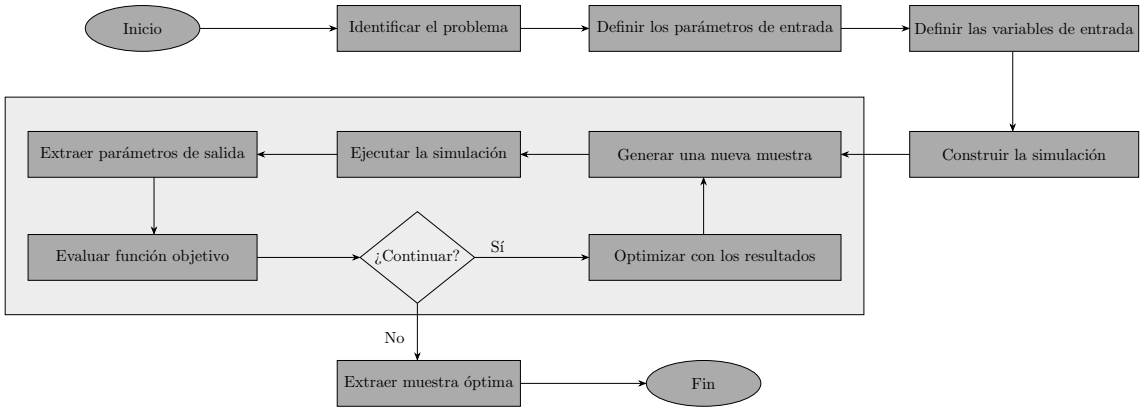


Figura 2.7: Esquema de los pasos a realizar para resolver un problema de optimización.

2.3.1. Optimización determinista

Es la optimización que se estudia clásicamente en matemáticas. Estos algoritmos se basan en el cálculo del gradiente y, muchas veces, en la Matriz Hessiana. Su principal ventaja es que son algoritmos mucho más rápidos en comparación con la optimización estocástica, ya que requieren un menor número de evaluaciones de la simulación. Además, los resultados obtenidos con estos algoritmos son fácilmente replicables. No obstante, estos algoritmos deterministas, al buscar puntos estacionarios en la variable de salida, podrían encontrar un mínimo local en lugar del mínimo global. Estos algoritmos se dividan en algoritmos con restricciones y sin ellas.

De manera general, un algoritmo de optimización sin restricciones parte de un punto $x^{(1)}$ y genera una secuencia de puntos $\left\{x^{(i)}\right\}$ en el espacio de diseño que converge a la solución x^*. Para ello, sea una recta el conjunto de puntos:

$$x(\alpha) = x' + \alpha s \tag{2.14}$$

Donde x' es un punto del espacio de diseño y s un vector director unitario. Para poder calcular el gradiente y el Hessiano en el libro se asume que la función de respuesta es por lo menos de clase C^2, esto como se verá más adelante es cierto para

la función del potencial creado por una puesta a tierra. Una función se dice que es de clase C^m si es continua, derivable y con derivadas continuas hasta el orden m.

Por la regla de la cadena se pueden obtener la primera y segunda derivada:

$$\frac{\mathrm{d}f(x)}{\mathrm{d}\alpha} = \sum_{i=1}^{k} \frac{\mathrm{d}x_i(\alpha)}{\mathrm{d}\alpha}\frac{\partial f(x_i)}{\partial x_i} = \sum_{i=1}^{k} s_i \frac{\partial f(x_i)}{\partial x_i} = s^T \nabla f(x) = s^T g(x) = g(x)^T s \quad (2.15)$$

$$\frac{\mathrm{d}^2 f(x)}{\mathrm{d}\alpha^2} = \frac{\mathrm{d}}{\mathrm{d}\alpha}\left[s^T \nabla f(x)\right] = s^T \nabla \left[\nabla f(x)^T s\right] = s^T \nabla^2 f(x) s = s^T G(x) s \quad (2.16)$$

Donde k es la dimensión del espacio de diseño, $g(x)$ el gradiente y $G(x)$ el Hessiano de la variable de respuesta.

Las condiciones necesarias para que un punto x^* sea el mínimo de la función $f(x)$ son:

$$\begin{cases} \dfrac{\mathrm{d}f(x)}{\mathrm{d}\alpha} = s^T g(x^*) = 0 \ \forall s \\ \dfrac{\mathrm{d}^2 f(x)}{\mathrm{d}\alpha^2} = s^T G(x^*) s \geq 0 \ \forall s \end{cases} \rightarrow \begin{cases} g(x^*) = 0 \\ G(x^*) \text{ definida positiva} \end{cases} \quad (2.17)$$

Una vez obtenidos los criterios para determinar que la solución x^* es óptima, se dice que un algoritmo converge con orden p si $\exists i_0$ y $\exists a$ de tal manera que $\forall n > i_0$ se cumple que la función $h^{(i+1)} = x^{(i)} - x^*$:

$$\frac{\left\|h^{(i+1)}\right\|}{\|h^{(i)}\|^p} \leq a \leftrightarrow h^{(i+1)} \text{ es del orden de magnitud de } \left\|h^{(i)}\right\|^p \quad (2.18)$$

Los problemas de optimización sin restricciones se basan normalmente en aproximaciones genéricas de la función objetivo. Para ello, se emplean dos estrategias distintas en las que se basan los distintos algoritmos:

1. **Búsqueda lineal**: esta estrategia se basa en los siguientes pasos para cada una de sus i iteraciones:

   - Determinar una dirección $s^{(i)}$
   - Encontrar un $\alpha^{(i)}$ para que se minimice $f\left(x^{(i)} + \alpha^{(i)} s^{(i)}\right)$
   - Calcular el siguiente punto $x^{(i+1)} = x^{(i)} + \alpha^{(i)} s^{(i)}$

   En las asignaturas de cálculo se estudia que la dirección $s^{(i)} = -g^{(i)}$ es aquella en la que se obtiene la máxima variación (pendiente) de la función a optimizar en cada iteración. No obstante, para acelerar el proceso, dependiendo del algoritmo, se elige una dirección $s^{(i)}$ distinta cuyo ángulo cumple que:

   $$cos\left(\theta^{(i)}\right) = \frac{-g^{(i)T} s^{(i)}}{\left\|g^{(i)}\right\|_2 \left\|s^{(i)}\right\|_2}, \quad \theta^{(i)} \leq \frac{\pi}{2} - \mu, \quad \mu > 0 \quad (2.19)$$

   Aún así, elegir una dirección y requerir que $f\left(x^{(i+1)}\right) < f\left(x^{(i)}\right)$ no asegura la convergencia, ya que permite reducciones minúsculas en la función objetivo. Por esta razón, normalmente se emplean las condiciones de Wolfe:

- La primera condición de Wolfe considera válidos solo aquellos puntos que cumplen:

$$f\left(x^{(i)}+\alpha s^{(i)}\right) \leq f\left(x^{(i)}\right)+\varepsilon_1 \alpha g\left(x^{(i)}\right)^T s^{(i)},\ 0<\varepsilon_1<1 \tag{2.20}$$

  Esta condición no es más que requerir que los puntos estén por debajo de una recta con una pendiente proporcional al gradiente.

- La primera condición de Wolfe considera válidos solo aquellos puntos que cumplen:

$$g\left(x^{(i)}+\alpha s^{(i)}\right) \geq \varepsilon_2 g\left(x^{(i)}\right),\ \varepsilon_1<\varepsilon_2<1 \tag{2.21}$$

  Esta condición obliga a que el punto elegido tenga una pendiente "menor", ya que a medida que uno se acerca al mínimo, la pendiente tiende a 0. Aunque en la inecuación haya un mayor o igual, esto se debe a que se desciende y las pendientes son negativas.

2. **Región de confianza:** esta estrategia se basa en asumir que la función objetivo, $f(x)$, se puede aproximar fácilmente por una función cuadrática, $q^{(i)}(\delta)$, en un entorno $\Omega^{(i)}$ de $x^{(i)}$ al truncar la función $f\left(x^{(i)}+\delta\right)$ mediante una serie de Taylor. A continuación, se define el radio $h^{(i)}$ en el entorno de $x^{(i)}$ como:

$$\Omega^{(i)}=\left\{x:\left\|x-x^{(i)}\right\| \leq h^{(i)}\right\} \tag{2.22}$$

Dentro de esta región, se busca la solución $\delta^{(i)}$ de tal manera que satisfaga:

$$\underset{\delta}{\text{minimizar}}\ q^{(i)}(\delta),\ \text{de tal manera que}\ \|\delta\| \leq h^{(i)} \tag{2.23}$$

Una vez obtenido δ se calcula el siguiente punto $x^{(i+1)}=x^{(i)}+\delta^{(i)}$. Para medir la validez de la aproximación cuadrática, se definen: la reducción actual en la función objetivo $\Delta f^{(i)}$, la reducción prevista $\Delta q^{(i)}$ y su cociente $r^{(i)}$.

$$\Delta f^{(i)}=f\left(x^{(i)}\right)-f\left(x^{(i+1)}\right) \tag{2.24}$$

$$\Delta q^{(i)}=q^{(i)}(0)-q^{(i)}\left(\delta^{(i)}\right)=f\left(x^{(i)}\right)-q^{(i)}\left(\delta^{(i)}\right) \tag{2.25}$$

$$r^{(i)}=\frac{\Delta f^{(i)}}{\Delta q^{(i)}} \tag{2.26}$$

El cociente $r^{(i)}$ es una medida de la exactitud de la aproximación cuadrática. La principal diferencia entre este método y el de búsqueda lineal reside en que en estos métodos primero se elige un paso y posteriormente la dirección de descenso, mientras que en los algoritmos de búsqueda lineal se lleva a cabo el proceso contrario.

Para una iteración n los pasos son los siguientes, el radio inicial, $h^{(0)}$, se debe elegir antes de iniciar el algoritmo:

- Calcular el gradiente $g^{(i)}$ y el Hessiano $G^{(i)}$ de $f^{(i)}$ para obtener la función $q^{(i)}$ mediante Taylor
- Encontrar la solución $\delta^{(i)}$

- Evaluar $f\left(x^{(i)}+\delta^{(i)}\right)$ y $r^{(i)}$
- Para elegir $h^{(i+1)}$ se sigue el siguiente criterio:
  - Si $r^{(i)}<0{,}25 \rightarrow h^{(i+1)}=\left\|\frac{\delta^{(i)}}{4}\right\|$
  - Si $r^{(i)}>0{,}75$ y $\left\|\delta^{(i)}\right\|=h^{(i)} \rightarrow h^{(i+1)}=2h^{(i)}$
  - En caso contrario $h^{(i+1)}=h^{(i)}$
- Para elegir $x^{(i+1)}$ se sigue el siguiente criterio
  - Si $r^{(i)} \leq 0 \rightarrow x^{(i+1)}=x^{(i)}$
  - En caso contrario $x^{(i+1)}=x^{(i)}+\delta^{(i)}$

A continuación, el libro propone distintos algoritmos de optimización sin restricciones:

- **Método de Newton**: en este algoritmo se aproxima la función objetivo a través de su serie de Taylor:

$$f\left(x^{(i)}+\delta\right) \approx q^{(i)}(\delta)=f^{(i)}+g^{(i)T}\delta+\frac{1}{2}\delta^{T}G^{(i)}\delta \tag{2.27}$$

  El algoritmo se basa en elegir el paso δ de tal manera que se minimice $q^{(i)}$. En ese caso se cumple que:

$$G^{(i)}\delta=-g^{(i)} \tag{2.28}$$

  Por tanto el paso en cada iteración sería:

$$x^{(i+1)}=x^{(i)}+\delta^{(i)} \tag{2.29}$$

  El problema de este procedimiento es que requiere que la matriz Hessiana sea definida positiva para asegurar convergencia. Por ello, se podrían emplear métodos de descenso máximo, $s^{(i)}=-g^{(i)}$, cuando la matriz Hessiana no sea definida positiva, o bien añadirle un sesgo, $G'^{(i)}=G^{(i)}+\lambda I$, donde $G'^{(i)}$ sería la matriz definida positiva a usar.

- **Métodos Cuasi-Newton**: es una variación del método de Newton para funciones donde la derivada no esté definida. Para ello se estima el gradiente a través de diferencias finitas mientras que el Hessiano se aproxima de distintas maneras en función de qué algoritmo de Cuasi-Newton se implemente.

- **Métodos de dirección conjugada**: en este algoritmo se comienza con un punto $x^{(1)}$ donde se selecciona como dirección de búsqueda $s^{(1)}=-g^{(i)}$. A continuación, se van eligiendo como direcciones de búsqueda un conjunto de vectores conjugado:

$$s^{(i)T}Gs^{(j)}=0,\ \forall i \neq j \tag{2.30}$$

  Para ello, se emplea la siguiente expresión:

$$s^{(i+1)}=-g^{(i)}+\beta^{(i)}s^{(i)},\ \beta^{(i)}=\frac{g^{(i+1)T}g^{(i+1)}}{g^{(i)T}g^{(i)}} \tag{2.31}$$

  Aunque para asegurar convergencia es preferible el uso de la siguiente expresión:

$$\beta^{(i)}=\frac{\left[g^{(i+1)}-g^{(i)}\right]^{T}g^{(i+1)}}{g^{(i)T}g^{(i)}} \tag{2.32}$$

Los algoritmos de gradiente conjugado son menos eficientes y robustos que los métodos de Cuasi-Newton. Sin embargo, tienen la ventaja de contar con una expresión muy sencilla para el cálculo de la dirección de búsqueda, $s^{(i)}$. Aunque esta sigue una de las expresiones previamente mencionadas, cada ciertas iteraciones es necesario reiniciar el cálculo de la dirección. Se explica este algoritmo más en detalle en la sección 3.3.1, pues es uno de los algoritmos empleados para el estudio.

- **Métodos de Levenberg-Marquadt**: es un algoritmo basado en el uso de la región de confianza donde para calcular el valor δ se emplea la siguiente expresión:

$$\left(G^{(i)} + \lambda I\right) \delta^{(i)} = -g^{(i)},\ \lambda \geq 0 \tag{2.33}$$

El parámetro λ se escoge de tal manera que la matriz $\left(G^{(i)} + \lambda I\right)$ sea definida positiva. Son una combinación entre el método de Newton donde $\lambda = 0$ y los métodos de descenso máximo $\lambda = \infty$. Tiene mejor estabilidad que el método de Newton.

2.3.2. Optimización determinista con restricciones

Un problema de optimización con restricciones genérico se puede expresar de la siguiente forma:

$$\begin{aligned} \underset{x}{\text{minimizar}}\ & f(x), \quad x \in \mathbb{R}^n \\ \text{sujeto a}\ & c_j(x) = 0,\ j \in E \\ & c_j(x) \geq 0,\ j \in J \end{aligned} \tag{2.34}$$

Donde E e J son los conjuntos de las restricciones de igualdad y desigualdad respectivamente. Se dice que un punto es válido si satisface estas restricciones, y al conjunto de puntos que cumple esta condición se le denomina región válida, R. Se asume que las restricciones y la función objetivo son continuas y que la región R es cerrada. Entonces si la región válida no es el conjunto vacío existe solución x^*.

Se define el conjunto de activo de restricciones en el punto x como:

$$\mathbb{A} = \mathbb{A}(x) = \{j : c_j(x) = 0\} \rightarrow A^T x = b \tag{2.35}$$

Existen diferentes problemas de optimización y una gran variedad de técnicas se aplican para resolverlos. Un ejemplo de estos problemas son los siguientes:

- **Programación lineal**: la función objetivo y las restricciones son lineales. Para resolver el problema se emplea el método símplex para optimización lineal.
- **Programación cuadrática**: la función objetivo es cuadrática y las restricciones son lineales. Para resolver el problema se emplea eliminación y multiplicadores de Lagrange para las restricciones de igualdad y métodos de conjunto activo para las restricciones de desigualdad. El problema se reduce a una función cuadrática sin restricciones.
- **Programación lineal general con restricciones**: la función objetivo es suave y las restricciones son lineales. Estos problemas se resuelven con los mismos métodos que los de programación cuadrática donde el problema se reduce en un problema de optimización sin restricciones genérico.

- **Programación no lineal**: la función objetivo y las restricciones son funciones suaves. Estos problemas se resuelven con métodos de barrera o con programación cuadrática secuencial.

2.3.2.1. Condiciones de minimalidad

En un problema de optimización con restricciones, la condición de que el gradiente sea nulo no es suficiente para garantizar un óptimo. Por esta razón, se introduce el concepto de multiplicadores de Lagrange, los cuales permiten manejar las restricciones. Considerando esto, una condición necesaria para que x^* sea un minimizador local es que el gradiente de la función objetivo sea igual a una combinación lineal de los gradientes de las restricciones:

$$\nabla f\left(x^*\right) = \sum_{j \in E \cup I} \lambda_j^* \nabla c_j(x^*) \rightarrow \nabla \left[f(x^*) - \sum_{j \in E \cup I} \lambda_j^* c_j(x^*) \right] \rightarrow \nabla_x \mathcal{L}\left(x^*, \lambda^*\right) = 0 \tag{2.36}$$

Donde $\mathcal{L}\left(x, \lambda\right)$ es el Lagrangiano.

La intuición detrás de la ecuación 2.36 se puede visualizar en la figura 2.8, donde se representan las curvas de nivel de una función parabólica $f(x, y) = a(x-b)^2+c(y-d)^2$ junto con una restricción lineal $c(x, y) = 0$. En este ejemplo, el punto óptimo se alcanza en el lugar donde el gradiente de la restricción (apunta hacia la dirección viable), $\nabla c(x, y)$, es proporcional al gradiente de la función objetivo, $\nabla f(x, y)$, ya que se trata de un problema de minimización. Esto es coherente con la interpretación geométrica del gradiente, el cual siempre apunta en la dirección de mayor variación de la función. Por lo tanto, si los gradientes no son proporcionales, significa que todavía existe una dirección en la que la función puede seguir disminuyendo.

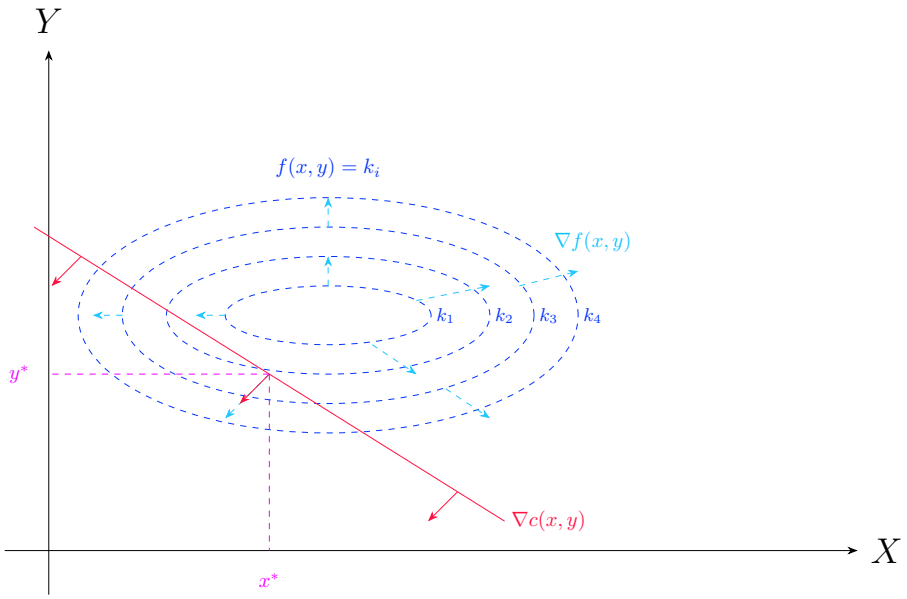


Figura 2.8: Representación geométrica de la condición de minimalidad mediante el Lagrangiano.

A partir del Lagrangiano, se definen las condiciones de primer orden de Karush–Kuhn–Tucker para que x^* sea un minimizador local:

- Condición de punto estacionario: $\nabla \mathcal{L}\left(x^*, \lambda^*\right) = 0$
- Condición de igualdad: $c_i(x^*) = 0 \; i \in E$
- Condición de desigualdad: $c_i(x^*) \geq 0 \; i \in I$
- Condición de signo o primaria: $\lambda_i^* \geq 0 \; i \in I$. Se debe penalizar a un valor x si no cumple las restricciones.
- Condición de holgura o dual: $\lambda_i^* c_i(x^*) = 0$. Las restricciones no deben afectar al valor de la función objetivo en el mínimo

Para que la solución local, x^*, sea un minimizador es también necesario que cumpla las condiciones de segundo orden o de curvatura. Partiendo del punto x^* si se añade un paso a partir de esta solución local se obtiene que:

$$f\left(x^* + \delta\right) = \mathcal{L}\left(x^* + \delta, \lambda^*\right) = \mathcal{L}\left(x^*, \lambda^*\right) + \delta^T \nabla \mathcal{L}\left(x^*, \lambda^*\right) + \frac{1}{2}\delta^T \left[\nabla^2 \mathcal{L}\left(x^*, \lambda^*\right)\right] \delta + \text{error} \tag{2.37}$$

$$f\left(x^* + \delta\right) = f(x^*) + \frac{1}{2}\delta^T W^* \delta + \text{error} \tag{2.38}$$

Donde W^* es el Hessiano del Lagrangiano y, por tanto, la condición de segundo grado para que x^* sea un minimizador es:

$$s^T W^* s \geq 0, \; \forall s : A^{*T} s = 0 \tag{2.39}$$

2.3.2.2. Métodos más comunes

- **Métodos de eliminación**: son métodos de optimización en los cuales las restricciones son de igualdad. Si la matriz Hessiana es definida positiva y la función objetivo a optimizar puede aproximarse por una función cuadrática, se pueden usar las restricciones para reducir la cantidad de variables de entrada.

  Una vez reducido el número de variables mediante las restricciones, el problema se transforma en un problema de optimización sin restricciones, ya estudiado anteriormente. Sustituyendo la solución de este problema en las restricciones originales, se obtiene el valor óptimo x^*.

- **Métodos de Lagrange**: se aplica a problemas con restricciones de igualdad y la función objetivo a optimizar puede aproximarse por una función cuadrática. Se parte de la definición del Lagrangiano:

$$\mathcal{L}\left(x, \lambda\right) = f(x) - \lambda^T \left(A^T x - b\right) = \frac{1}{2} x^T G x + g^T x - \lambda^T \left(A^T x - b\right) \tag{2.40}$$

  Si se aplican las condiciones de punto estacionario:

$$\left.\begin{array}{r} \nabla_x \mathcal{L} = 0 \rightarrow Gx + g - A\lambda = 0 \\ \nabla_\lambda \mathcal{L} = 0 \rightarrow A^T x - b = 0 \end{array}\right\} \rightarrow \begin{pmatrix} G & -A \\ -A^T & 0 \end{pmatrix} \begin{pmatrix} x \\ \lambda \end{pmatrix} = - \begin{pmatrix} g \\ b \end{pmatrix} \tag{2.41}$$

  Por tanto, la solución del problema sería:

$$\begin{pmatrix} x^* \\ \lambda^* \end{pmatrix} = - \begin{pmatrix} G & -A \\ -A^T & 0 \end{pmatrix}^{-1} \begin{pmatrix} g \\ b \end{pmatrix} \tag{2.42}$$

- **Métodos de conjunto activo:** son métodos utilizados para gestionar restricciones de desigualdad. En estos, se van añadiendo algunas de las inecuaciones al conjunto activo, donde se calcula un punto que satisface las restricciones (como las desigualdades $c_j(x) \leq 0$, que se consideran como $c_j(x) = 0$ en el conjunto activo). El proceso itera sobre la frontera de la región delimitada por estas inecuaciones hasta que se cumplen las condiciones de optimalidad de Karush-Kuhn-Tucker (KKT).
- **Métodos de barrera o función de penalización**: son métodos donde se sustituye el problema de optimización con el conjunto restricciones, $c_j(x) = 0 \ \forall j \in E$ y $c_j(x) \geq 0 \ \forall j \in J$ en un problema de optimización sin restricciones donde se cumple que la función de coste, $f(x)$:

$$f(x) = \begin{cases} f(x) \ si \ x \in c_j(x) \\ \infty \ si \ x \notin c_j(x) \end{cases} \tag{2.43}$$

  No obstante, esta función no es suave y, por tanto se prefiere usar funciones lineales o logarítmicas que mediante un parámetro se pueden ajustar para conseguir un comportamiento similar al de la función 2.43. Estos métodos se pueden emplear para problemas no lineales. Se explica la aplicación de este algoritmo más en detalle en la sección 3.3.2, pues es uno de los algoritmos empleados para el estudio.
- **Métodos programación cuadrática secuencial**: son métodos empleados en problemas de optimización donde la función objetivo o las restricciones son no lineales. Estos métodos se utilizan normalmente en conjunción con técnicas de cuasi-Newton y consisten en realizar aproximaciones sucesivas de la función Lagrangiana hasta resolver el problema.

2.3.3. Optimización estocástica

Los algoritmos de optimización estocástica son métodos de búsqueda que incorporan elementos aleatorios en su proceso. Muchos de estos algoritmos están inspirados en fenómenos naturales que exhiben aleatoriedad, y buscan replicar tales procesos. Generalmente, estos algoritmos se basan en el uso de poblaciones que, con el tiempo, convergen hacia una solución óptima. El desafío clave radica en encontrar un equilibrio entre la exploración exhaustiva del espacio de diseño y la velocidad de convergencia.

A continuación se exponen brevemente los algoritmos mencionados en el libro:

- **Algoritmo de temple simulado**: el proceso comienza con la evaluación del valor de la función objetivo en un punto cualquiera del espacio de diseño, $f\left(x^{(1)}\right)$ y una temperatura inicial, T_0. A continuación, se siguen los siguientes pasos:
  - Elegir una ley de enfriamiento de tal manera que se usará para determinar si una configuración es viable. Típicamente se usa la siguiente:

$$T^{(i)} = c \cdot T^{(i-1)}, \ 0 \leq c \leq 1 \tag{2.44}$$

    Donde c es la constante de enfriamiento.

- Elegir una regla para obtener el siguiente punto de la iteración, $x^{(i+1)}$, de manera que permita alta movilidad a altas temperaturas y baja movilidad cuando la temperatura se acerca a 0.
- Una vez obtenido un punto para aceptar el nuevo punto se realiza la siguiente comprobación:

$$t^{(i)} \leq exp\left[-\frac{f\left(x^{(i+1)}\right) - f\left(x^{(i)}\right)}{T^{(i)}}\right] \tag{2.45}$$

Donde $t^{(i)}$ es un número aleatorio entre 0 y 1. Cabe recalcar que si $f\left(x^{(i+1)}\right) \leq f\left(x^{(i)}\right)$ no es necesaria la comprobación ya que la configuración siempre será aceptada.

Este algoritmo tiene la ventaja de permitir muestras que no necesariamente mejoran la función objetivo, lo que facilita superar mínimos locales. Sin embargo, es menos eficiente que otros métodos estocásticos, excepto en problemas donde el espacio de diseño es discreto.

- **Optimización por enjambre de partículas**: este algoritmo se basa en explorar el espacio de diseño haciendo converger las distintas soluciones hacia el individuo óptimo, imitando el vuelo de las aves. Para ello, se define la posición $x_j^{(i)}$, donde $j = 1, \ldots, m$, de cada individuo en la iteración i, siendo m el número de individuos.

$$x_j^{(i)} = x_j^{(i-1)} + v_j^{(i)} \tag{2.46}$$

Donde $v_j^{(i)}$ es el vector de velocidad de un individuo j. Para calcular la velocidad de un individuo se emplea la siguiente expresión:

$$v_j^{(i)} = C_1 v_j^{(i-1)} + C_2 r_2 \left[\bar{x}_j - x_j^{(i-1)}\right] + C_3 r_3 \left[\bar{x} - x_j^{(i-1)}\right] \tag{2.47}$$

Donde los factores C_1, C_2 y C_3 son la inercia, el factor de aprendizaje cognitivo y el factor de aprendizaje social, respectivamente. Los valores r_2 y r_3 son números aleatorios entre 0 y 1. Los valores $\bar{x}_j$ y $\bar{x}$ representan el mejor personal y el mejor global, respectivamente.

El factor de inercia es un valor utilizado para permitir que cada individuo explore el espacio de diseño y debe ir disminuyendo para garantizar la convergencia. Por otro lado, los factores de aprendizaje cognitivo y aprendizaje social se ajustan en función de la rapidez requerida.

- **Algoritmos evolutivos**: estos algoritmos, al igual que los genéticos, se basan en generar una población a lo largo de varias generaciones, seleccionando a los individuos con mejor desempeño. Para generar una nueva población, se emplean operadores específicos. Uno de los operadores clave es la mutación, que consiste en realizar pequeñas modificaciones aleatorias sobre uno o varios genes del individuo. Un gen representa una variable de la solución, y la mutación puede implicar un cambio en su valor dentro de un rango predefinido.

Los pasos de este algoritmo son los siguientes, y se repiten hasta que se cumplan los criterios de terminación, salvo la inicialización:

- Inicialización
- Mutación: se crea un individuo mutante para cada individuo en la población
- Cruce: el mutante se combina con su padre para crear un individuo de prueba
- Evaluación: se comprueba la aptitud de los individuos
- Selección: se selecciona entre el individuo de prueba y su padre para que sobreviva para la siguiente generación

- **Algoritmos genéticos**: a diferencia de los algoritmos evolutivos, los algoritmos genéticos dan mayor importancia al operador de cruce. Para ello, normalmente se discretiza el espacio de diseño, y para generar nuevos individuos, se divide el vector de bits de los padres y se recombinan las partes para formar los hijos. En la sección 3.4 se explicarán estos procesos con mayor detalle, ya que es uno de los algoritmos empleados para la optimización del potencial de la puesta a tierra.

  Los pasos del algoritmo genético son los siguientes y salvo la inicialización se repiten hasta cumplir con las condiciones de terminación:

  - Inicialización en el espacio de diseño discreto y evaluación de la aptitud de los individuos
  - Creación de nuevos individuos mediante los siguientes métodos:
    - Selección de unos pocos individuos de élite
    - Cruce con alta probabilidad de dos padres para dar lugar a dos hijos
    - Mutación de los hijos, este mecanismo normalmente tiene una baja probabilidad de ocurrir
  - Reemplazo de la población anterior y evaluación de la función objetivo

Capítulo 3

Fundamentos generales

En este capítulo se desarrollan los fundamentos teóricos necesarios para comprender las implementaciones realizadas.

3.1. Método de simulación de cargas (CSM)

3.1.1. Obtención de la densidad lineal de carga

Para el cálculo de los potenciales y las densidades de carga se emplea el método de simulación de cargas descrito en [6] para una estructura conductora compleja. En esta estructura cada electrodo se compone de M cargas distribuidas uniformemente situadas en $\vec{r}_{jk}$ que representan puntos fuente (en rojo en el dibujo) que generan potencial sobre los puntos de campo situados en $\vec{r}_{is}$ (en azul en el dibujo).

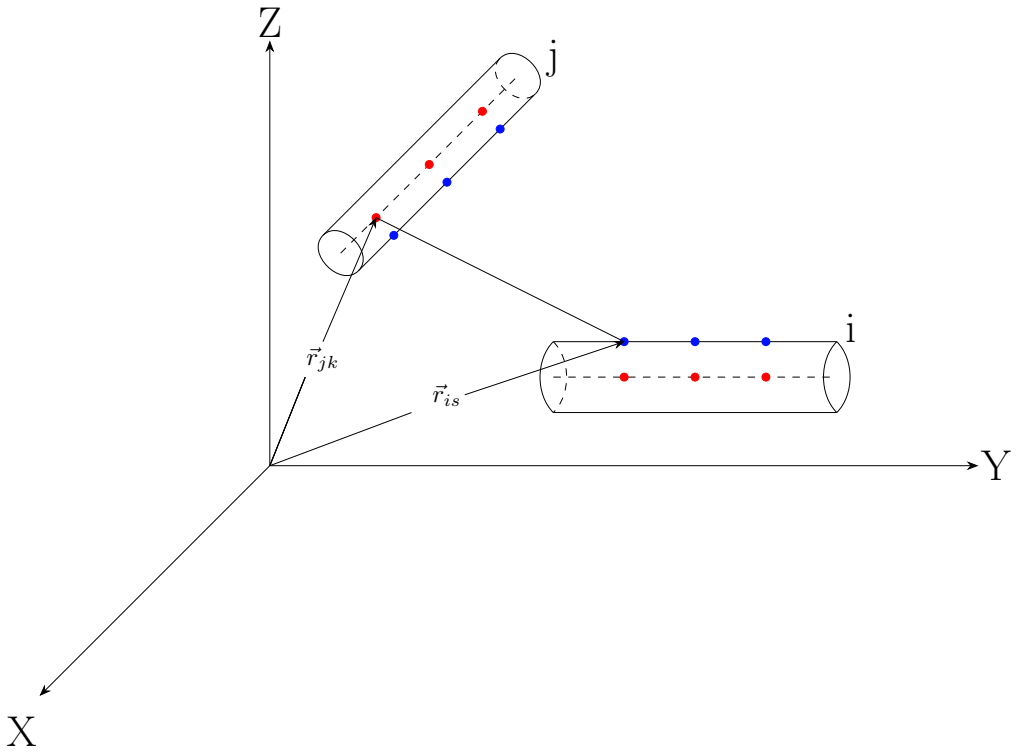


Figura 3.1: Representación del método CSM.

En este caso se cumple que el potencial generado por un punto fuente en un punto campo de otro electrodo por analogía electrostática es:

$$V_{jk}(\vec{r}_{is}) = L_j \frac{\rho}{4\pi} \frac{\lambda(\vec{r}_{jk})}{\|\vec{r}_{jk} - \vec{r}_{is}\|} \iff i \neq j \tag{3.1}$$

Donde L_j es la longitud del electrodo j y $\lambda(\vec{r}_{jk})$ la densidad lineal de corriente en el punto $\vec{r}_{jk}$.

Si el punto fuente se encuentra en el mismo electrodo que el punto campo se emplea la expresión del potencial creado por un hilo conductor de longitud L que puede consultarse la deducción en [6].

$$V_{jk}(\vec{r}_{js}) = \rho \frac{\lambda(\vec{r}_{jk})}{4\pi} \ln \left[\frac{\frac{L_j}{2} + z_{jsk} + \sqrt{R^2 + \left(\frac{L}{2} + z_{jsk}\right)^2}}{-\frac{L_j}{2} + z_{jsk} + \sqrt{y^2 + x^2 + \left(\frac{L}{2} - z_{jsk}\right)^2}} \right] \iff i = j \tag{3.2}$$

$$z_{jsk} = \|\vec{r}_{js} - \vec{r}_{jk}\| \tag{3.3}$$

Por tanto, aplicando el principio de superposición, el campo creado por el electrodo j sobre un punto de campo será:

$$V_j(\vec{r}_{is}) = \begin{cases} L_j \frac{\rho}{4\pi} \sum_{k=1}^{M} \frac{\lambda(\vec{r}_{jk})}{\|\vec{r}_{jk} - \vec{r}_{is}\|} \iff i \neq j \\ \\ \frac{\rho}{4\pi} \sum_{k=1}^{M} \lambda(\vec{r}_{jk}) \ln \frac{\frac{L_j}{2} + z_{jsk} + \sqrt{R^2 + \left(\frac{L}{2} + z_{jsk}\right)^2}}{-\frac{L_j}{2} + z_{jsk} + \sqrt{y^2 + x^2 + \left(\frac{L}{2} - z_{jsk}\right)^2}} \iff i = j \end{cases} \tag{3.4}$$

Donde se define LN_{jsk} como:

$$LN_{jsk} = \ln \frac{\frac{L_j}{2} + z_{jsk} + \sqrt{R^2 + \left(\frac{L}{2} + z_{jsk}\right)^2}}{-\frac{L_j}{2} + z_{jsk} + \sqrt{y^2 + x^2 + \left(\frac{L}{2} - z_{jsk}\right)^2}} \tag{3.5}$$

Si se tiene en cuenta la contribución de los 2N electrodos, el potencial total en un punto campo cualquiera será:

$$V(\vec{r}_{is}) = \sum_{j=1, i \neq j}^{2N} \sum_{k=1}^{M} L_j \frac{\rho}{4\pi} \frac{\lambda(\vec{r}_{jk})}{\|\vec{r}_{jk} - \vec{r}_{is}\|} + \frac{\rho}{4\pi} \sum_{k=1}^{M} \lambda(\vec{r}_{ik}) LN_{isk} \tag{3.6}$$

Este potencial se expresa mediante un vector de tamaño $2NM \times 1$.

$$\vec{V} = \begin{pmatrix} V(\vec{r}_{11}) \\ V(\vec{r}_{12}) \\ \vdots \\ V(\vec{r}_{1M}) \\ V(\vec{r}_{21}) \\ \vdots \\ V(\vec{r}_{2NM}) \end{pmatrix} \in \mathbb{R}^{2NM\times 1} \tag{3.7}$$

Sacando factor común la densidad de corriente se obtiene el siguiente sistema de ecuaciones donde toda la geometría de la malla queda recogida en la matriz R de tamaño 2NM.

$$\vec{V} = R\vec{\lambda} \tag{3.8}$$

$$\begin{pmatrix} V(\vec{r}_{11}) \\ V(\vec{r}_{12}) \\ \vdots \\ V(\vec{r}_{1M}) \\ V(\vec{r}_{21}) \\ \vdots \\ V(\vec{r}_{2NM}) \end{pmatrix} = \frac{1}{4\pi} \begin{pmatrix} LN_{111} & \cdots & LN_{11M} & \frac{L_2}{\|\vec{r}_{21}-\vec{r}_{11}\|} & \cdots & \frac{L_2}{\|\vec{r}_{2M}-\vec{r}_{11}\|} & \cdots & \frac{L_{2N}}{\|\vec{r}_{2NM}-\vec{r}_{11}\|} \\ \vdots & & \vdots & \vdots & & \vdots & & \vdots \\ LN_{1M1} & \cdots & LN_{1MM} & \frac{L_2}{\|\vec{r}_{21}-\vec{r}_{1M}\|} & \cdots & \frac{L_2}{\|\vec{r}_{2M}-\vec{r}_{1M}\|} & \cdots & \frac{L_{2N}}{\|\vec{r}_{2NM}-\vec{r}_{1M}\|} \\ \frac{L_1}{\|\vec{r}_{11}-\vec{r}_{21}\|} & \cdots & \frac{L_1}{\|\vec{r}_{1M}-\vec{r}_{21}\|} & LN_{211} & \cdots & LN_{21M} & \cdots & \frac{L_{2N}}{\|\vec{r}_{2NM}-\vec{r}_{21}\|} \\ \vdots & & \vdots & \vdots & & \vdots & & \vdots \\ \frac{L_1}{\|\vec{r}_{11}-\vec{r}_{2M}\|} & \cdots & \frac{L_1}{\|\vec{r}_{1M}-\vec{r}_{2M}\|} & LN_{2M1} & \cdots & LN_{2MM} & \cdots & \frac{L_{2N}}{\|\vec{r}_{2NM}-\vec{r}_{2M}\|} \\ \vdots & & \vdots & \vdots & & \vdots & & \vdots \\ \frac{L_1}{\|\vec{r}_{11}-\vec{r}_{2NM}\|} & \cdots & \frac{L_1}{\|\vec{r}_{1M}-\vec{r}_{2NMM}\|} & \frac{L_2}{\|\vec{r}_{2M}-\vec{r}_{2N1}\|} & \cdots & \frac{L_2}{\|\vec{r}_{2M}-\vec{r}_{2NM}\|} & \cdots & LN_{2NMM} \end{pmatrix} \cdot \begin{pmatrix} \lambda(\vec{r}_{11}) \\ \lambda(\vec{r}_{12}) \\ \vdots \\ \lambda(\vec{r}_{1M}) \\ \lambda(\vec{r}_{21}) \\ \vdots \\ \lambda(\vec{r}_{2NM}) \end{pmatrix}$$

Para calcular la densidad de corrientes, dado que todo el electrodo debe estar al mismo potencial, se asume que la tensión y la resistividad del terreno son iguales a 1. Tras ello se despeja la densidad de corriente.

$$\vec{\lambda} = R^{-1}\vec{1} \tag{3.9}$$

A continuación, se obtiene la corriente total cuando el voltaje de la puesta a tierra es de 1 voltio:

$$I = \sum_{l=1}^{2NM} \lambda_l L_l \tag{3.10}$$

Por último, se obtiene la densidad de corriente por unidad, *dens* y la resistencia de puesta a tierra, k_r, mediante las siguientes ecuaciones:

$$dens = \frac{\vec{\lambda}}{I} \tag{3.11}$$

$$k_r = \frac{R_{pat}}{\rho} = \frac{V_e}{I\rho} = \frac{1}{\frac{I}{2}} = \frac{2}{I} \tag{3.12}$$

Es importante recalcar que el 2 de la ecuación 3.12 se usa para tener en cuenta el efecto del electrodo imagen.

3.1.2. Cálculo del potencial en un punto

Una vez obtenido el vector de densidades, $\vec{\lambda}_{p.u.}$, el cálculo del potencial en un punto cualquiera del espacio se realiza mediante la aplicación directa de la expresión 3.1 (en este caso no es necesario que $i \neq j$) y aplicando el principio de superposición.

$$V(\vec{r}) = \frac{\rho I_{falta}}{4\pi} \sum_{j=1}^{2N} \sum_{k=1}^{M} \frac{L_j}{M} \frac{\lambda(\vec{r}_{jk})}{\|\vec{r}_{jk} - \vec{r}\|} \tag{3.13}$$

3.2. Fundamentos teóricos algoritmos de preprocesamiento

Los datos de partida del problema son tres matrices con la definición de los electrodos y sus radios en metros:

- P1 $\rightarrow$ Matriz $2N \times 3$ con las coordenadas del inicio de los electrodos.
- P2 $\rightarrow$ Matriz $2N \times 3$ con las coordenadas del final de los electrodos.
- r $\rightarrow$ Matriz $2N \times 1$ con los radios de los electrodos.

El problema con estos datos es que no siempre son adecuados, por lo que es necesario procesarlos para poder comprimirlos, obtener resultados más precisos en el cálculo numérico y establecer las condiciones de contorno del algoritmo genético.

Antes de abordar los fundamentos del procesamiento de datos, es importante mencionar que la geometría de la subestación puede proyectarse sobre el plano ($z = 0$) [7], lo que permite trabajar con la geometría en $\mathbb{R}^2$. Sin embargo, existen algunas excepciones, como las picas y los electrodos imagen, que deben considerarse durante el preprocesamiento. Se puede ver un ejemplo de una puesta a tierra de una subestación en la figura 3.2.

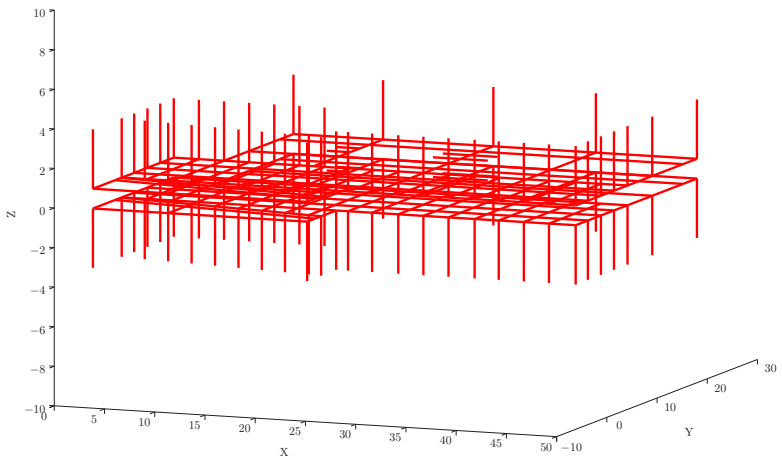


Figura 3.2: Representación de una puesta a tierra donde se observa claramente como salvo las picas se pueden distinguir claramente 2 planos.

3.2.1. Rutinas preprocesado de electrodos

3.2.1.1. Ecuación general de una recta

Uno de los pilares fundamentales para poder trabajar con la geometría de los electrodos es la de poder convertir las coordenadas de las parejas de nodos inicial y final en pendientes y punto de corte de las ecuaciones de las rectas del espacio que unen dichos nodos:

$$y = mx + n \tag{3.14}$$

Como se conocen las coordenadas de 2 puntos que conforman la recta:

$$\left.\begin{array}{l} P_{1y} = mP_{1x} + n \\ P_{2y} = mP_{2x} + n \end{array}\right\} \rightarrow \begin{cases} m = \dfrac{P_{2y} - P_{1y}}{P_{2x} - P_{1x}} \\ \\ n = (P_{2y} - P_{1y}) \left[\dfrac{P_{1y}}{P_{2y} - P_{1y}} - \dfrac{P_{1x}}{P_{2x} - P_{1x}} \right] \end{cases} \tag{3.15}$$

Para implementar de manera más cómoda las ecuaciones se opera la ecuación 3.15 hasta obtener su forma implícita:

$$-\frac{x}{P_{2x} - P_{1x}} + \frac{y}{P_{2y} - P_{1y}} = \frac{P_{1y}}{P_{2y} - P_{1y}} - \frac{P_{1x}}{P_{2x} - P_{1x}} \tag{3.16}$$

En forma matricial:

$$\begin{pmatrix} -\dfrac{1}{P_{2x} - P_{1x}} & \dfrac{1}{P_{2y} - P_{1y}} \end{pmatrix} \cdot \begin{pmatrix} x \\ y \end{pmatrix} = \frac{P_{1y}}{P_{2y} - P_{1y}} - \frac{P_{1x}}{P_{2x} - P_{1x}} \tag{3.17}$$

En estas ecuaciones se observa como existen dos casos límites donde se producen singularidades:

1. Cuando $P_{2x} - P_{1x} = 0 \rightarrow$ Los puntos que delimitan el segmento en x están alineados. Por tanto, la recta tiene la expresión $y = P_{1y}$.
2. Cuando $P_{2y} - P_{1y} = 0 \rightarrow$ Los puntos que delimitan el segmento en y están alineados. Por tanto, la recta tiene la expresión $x = P_{1x}$.

Es importante recalcar que no se pueden dar ambas situaciones a la vez ya que para el procesamiento geométrico no se tienen en cuenta las picas.

3.2.1.2. Condición para la intersección entre dos segmentos

Partiendo de la forma matricial deducida anteriormente, es sencillo obtener la compatibilidad del sistema de ecuaciones y, por tanto, conocer si dos segmentos intersecan. El sistema que describe la intersección entre un electrodo i y un electrodo j quedaría de la siguiente forma:

$$\begin{cases} \begin{pmatrix} -\frac{1}{\Delta x(i)} & \frac{1}{\Delta y(i)} \\ -\frac{1}{\Delta x(j)} & \frac{1}{\Delta y(j)} \end{pmatrix} \cdot \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} \frac{P_{1y}(i)}{\Delta y(i)} - \frac{P_{1x}(i)}{\Delta x(i)} \\ \frac{P_{1y}(j)}{\Delta y(j)} - \frac{P_{1x}(j)}{\Delta x(j)} \end{pmatrix} \\ x_{min} \leq x \leq x_{max} \\ y_{min} \leq y \leq y_{max} \\ x_{min} = Max\{min[P_{1x}(i), P_{2x}(i)], min[P_{1x}(j), P_{2x}(j)]\} \\ x_{max} = min\{Max[P_{1x}(i), P_{2x}(i)], Max[P_{1x}(j), P_{2x}(j)]\} \\ y_{min} = Max\{min[P_{1y}(i), P_{2y}(i)], min[P_{1y}(j), P_{2y}(j)]\} \\ y_{max} = min\{Max[P_{1y}(i), P_{2y}(i)], Max[P_{1y}(j), P_{2y}(j)]\} \end{cases} \tag{3.18}$$

Donde, (i) y (j) denotan el electrodo i y j respectivamente, min la función mínimo, Max la función máximo y los incrementos siendo:

$$\Delta x = P_{2x} - P_{1x} \tag{3.19}$$

$$\Delta y = P_{2y} - P_{1y} \tag{3.20}$$

Para entender el razonamiento de selección de los límites de las variables se adjunta la figura 3.3:

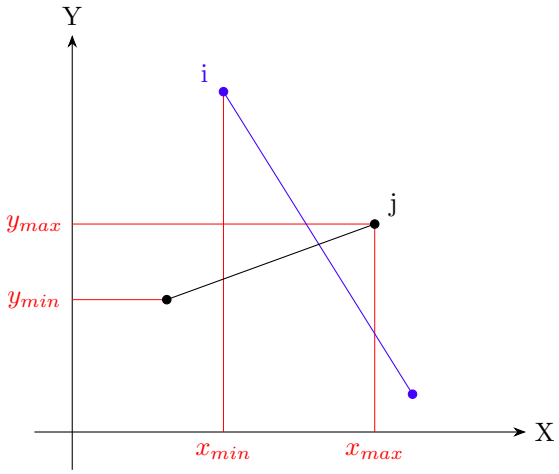


Figura 3.3: Intersección entre dos electrodos.

Por tanto, para que se produzca la intersección de dos electrodos se debe estudiar la compatibilidad del sistema descrito mediante la ecuación 3.18. Para ello, como el sistema es de la forma:

$$Ax = B \rightarrow x = A^{-1}B \tag{3.21}$$

$$A = \begin{pmatrix} -\frac{1}{\Delta x(i)} & \frac{1}{\Delta y(i)} \\ -\frac{1}{\Delta x(j)} & \frac{1}{\Delta y(j)} \end{pmatrix} \tag{3.22}$$

$$B = \begin{pmatrix} \frac{P_{1y}(i)}{\Delta y(i)} - \frac{P_{1x}(i)}{\Delta x(i)} \\ \frac{P_{1y}(j)}{\Delta y(j)} - \frac{P_{1x}(j)}{\Delta x(j)} \end{pmatrix} \tag{3.23}$$

Si el determinante de la matriz A es nulo, no existirá solución y, en consecuencia, no habrá intersección entre los electrodos. Aunque podría argumentarse que, en el caso de electrodos superpuestos, este razonamiento no es completamente aplicable, es importante señalar que los electrodos superpuestos se consideran un error en el diseño de la puesta a tierra y, por lo tanto, no se admiten y se pueden eliminar en el preproceso.

3.2.2. Cálculo de la frontera

El cálculo de la frontera que encierra la puesta a tierra es fundamental. Como se estudio anteriormente, según [7], en el diseño de las puestas a tierra se realiza un

bucle exterior mediante electrodos que contiene el perímetro de la subestación. Por tanto, la región de estudio debe ser aquella encerrada por la poligonal que forma el perímetro de la subestación.

Para obtener la frontera se transforman los datos para tenerlos en un formato donde su manejo sea más cómodo. Para ello, se convierten las matrices $P1$ y $P2$ en grafos con la mínima cantidad de puntos necesaria y sus conexiones. Una primera aproximación para resolver el problema podría ser el cálculo de una envolvente convexa [18]. No obstante, como se ve en la figura 3.4, la envolvente convexa puede dar resultados que no corresponden con la frontera real de la subestación.

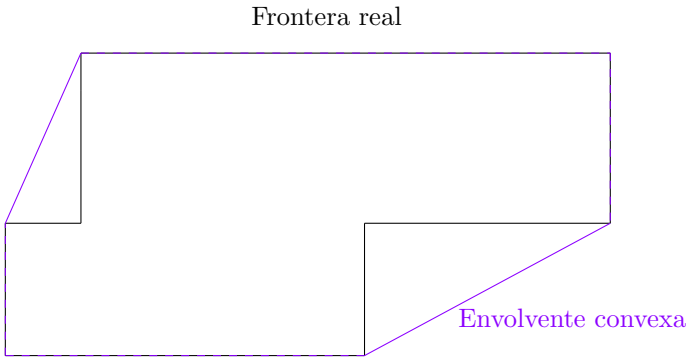


Figura 3.4: Comparación entre frontera real y envolvente convexa.

Se define la frontera como una poligonal cuyos lados son electrodos de la puesta a tierra, de modo que todos los electrodos queden dentro de la región delimitada por esta poligonal. Para el cálculo de la frontera se toman los siguientes supuestos:

1. No existen electrodos conectados por un único punto a la malla de puesta a tierra en su exterior. Véase figura 3.5.
2. Existe una frontera delimitada por electrodos exteriores.
3. No existen electrodos superpuestos.

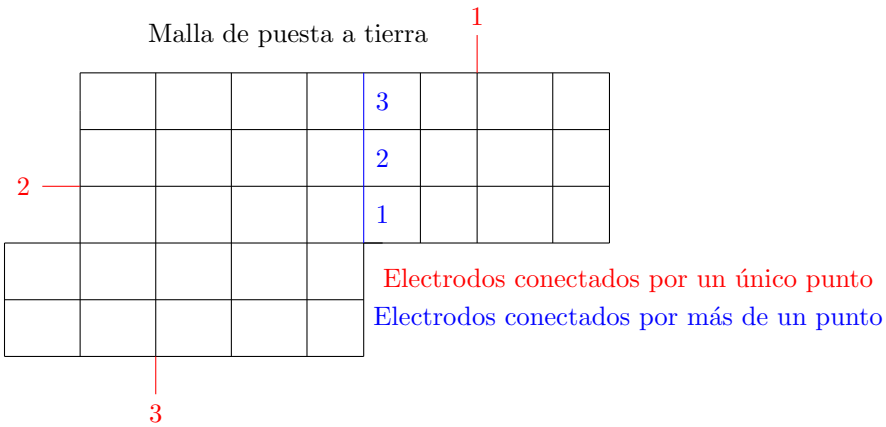


Figura 3.5: Malla de puesta a tierra con electrodos conectados por un único punto a la malla de puesta a tierra en su exterior en rojo.

Siguiendo los supuestos anteriores, se comienza con dos puntos en la frontera de la malla. El primer punto se selecciona buscando el punto más al noreste, es decir, el que tiene la mayor coordenada x e y. Posteriormente, se selecciona un punto adyacente que esté lo más alejado posible del centroide de la malla. Este método es válido porque, al no haber electrodos conectados únicamente en un punto, el primer punto siempre será una esquina. Mientras que el segundo punto no puede estar en el interior de la frontera, ya que eso implicaría que hay un punto adyacente a la esquina más alejado del centroide.

Cuando se tiene un primer segmento, este debe orientarse en sentido antihorario ya que la frontera calculada tendrá en este sentido. No obstante, como solo a partir de dos puntos no puede saberse su orientación no queda más remedio que tratar de calcular la frontera y sí no se ha llegado al resultado deseado simplemente recalcular cambiando de sentido el segmento. Para visualizar mejor el porqué a partir de un solo segmento no se orientar véase la figura 3.6.

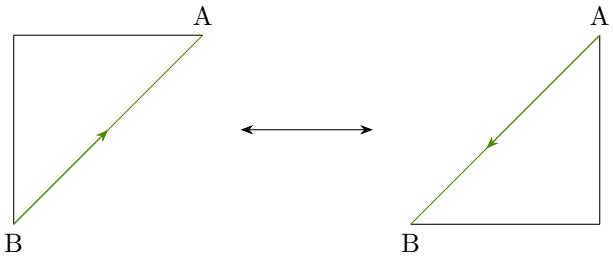


Figura 3.6: Representación comparativa para ver como un mismo segmento puede cambiar de sentido en la frontera en función de la geometría.

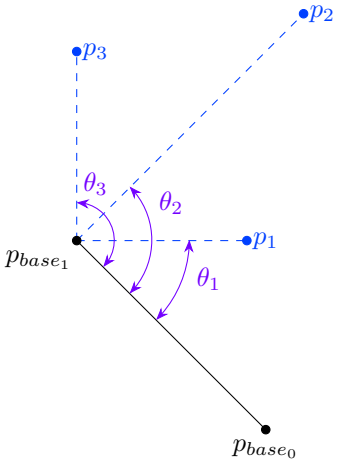


Figura 3.7: Representación del algoritmo para seleccionar un punto en la frontera.

Una vez obtenido el segmento orientado, la idea básica para añadir puntos es seguir el sentido antihorario de la frontera. Si el segmento no está orientado adecuadamente se verifica al final del algoritmo. El procedimiento consiste en calcular el ángulo en sentido positivo entre el segmento base y el segmento formado por el punto final del segmento base y cada uno de sus puntos adyacentes. Se selecciona como siguiente punto de la frontera aquel que forme el menor ángulo respecto al segmento base. Para visualizar el procedimiento ver la figura 3.7.

Una vez seleccionado el punto que forma el menor ángulo con el segmento base, se añade este punto a la frontera. El nuevo segmento base se forma entre p_{base_1} y p_1. Se repite este proceso hasta volver a uno de los puntos iniciales del algoritmo. Si no se regresa al punto inicial después de recorrer los n puntos del grafo, se debe repetir

el algoritmo cambiando la orientación de los puntos base iniciales.

3.2.2.1. Comprobación de la orientación de una polilínea cerrada

El criterio para obtener la orientación de una polilínea cerrada se basa en la fórmula del área de Gauss. Una deducción para dicha fórmula se encuentra en el artículo [19]. Aunque el artículo no desarrolla directamente una expresión para la orientación de una polilínea sino para el área encerrada por ella, empleando las propiedades de los determinantes se puede llegar a un criterio para obtener la orientación.

Fórmula del área de Gauss. Sea un polígono formado por los n vértices $(x_0, y_0), (x_1, y_1), ..., (x_{n-2}, y_{n-2}), (x_{n-1}, y_{n-1})$ orientados en sentido antihorario, entonces su área, A, es igual a la siguiente expresión:

$$2A = \begin{vmatrix} x_0 & x_1 \\ y_0 & y_1 \end{vmatrix} + \begin{vmatrix} x_1 & x_2 \\ y_1 & y_2 \end{vmatrix} + ... + \begin{vmatrix} x_{n-2} & x_{n-1} \\ y_{n-2} & y_{n-1} \end{vmatrix} + \begin{vmatrix} x_{n-1} & x_0 \\ y_{n-1} & y_0 \end{vmatrix} \tag{3.24}$$

Si en el computo de los determinantes los puntos estuviesen orientados en sentido horario se obtendría la suma, S:

$$S = \begin{vmatrix} x_{n-1} & x_{n-2} \\ y_{n-1} & y_{n-2} \end{vmatrix} + \begin{vmatrix} x_{n-2} & x_{n-3} \\ y_{n-2} & y_{n-3} \end{vmatrix} + ... + \begin{vmatrix} x_1 & x_0 \\ y_1 & y_0 \end{vmatrix} + \begin{vmatrix} x_0 & x_{n-1} \\ y_0 & y_{n-1} \end{vmatrix} \tag{3.25}$$

Empleando la propiedad de que intercambiar columnas en un determinante cambia el signo del mismo se obtiene que:

$$S = -\begin{vmatrix} x_{n-2} & x_{n-1} \\ y_{n-2} & y_{n-1} \end{vmatrix} - \begin{vmatrix} x_{n-3} & x_{n-2} \\ y_{n-3} & y_{n-2} \end{vmatrix} - ... - \begin{vmatrix} x_0 & x_1 \\ y_0 & y_1 \end{vmatrix} - \begin{vmatrix} x_{n-1} & x_0 \\ y_{n-1} & y_0 \end{vmatrix} = -2A \tag{3.26}$$

Por tanto, para saber si una polilínea cerrada esta orientada en sentido horario o antihorario basta con calcular la suma de estos determinantes y comprobar el signo de la suma. Si se obtiene un número negativo los puntos están en sentido horario y en sentido antihorario en caso contrario.

3.2.3. Subdivisión convexa

Aunque en la bibliografía se mencionan distintos métodos para obtener una subdivisión en convexos, dado que las puestas a tierra suelen tener geometrías sin demasiados puntos de concavidad para que la ejecución de la obra sea razonable, se realiza una descomposición "sencilla"[8] de la frontera. Para ello, el primer paso es calcular los puntos de concavidad.

3.2.3.1. Cálculo de puntos de concavidad

El proceso para calcular los puntos de concavidad o notches (ángulo interior $> 180°$) el proceso es bastante sencillo. Se va recorriendo la frontera y sacando los vectores directores, véase la figura 3.8. A continuación, se obtiene la matriz de rotación que desplazaría el vector v_1 para que sea solidario al eje x, $R(-\alpha)$.

La expresión de una matriz de rotación genérica en el plano es:

$$R(\varphi) = \begin{pmatrix} cos(\varphi) & -sen(\varphi) \\ sen(\varphi) & cos(\varphi) \end{pmatrix} \tag{3.27}$$

Donde φ es el ángulo en sentido positivo teniendo como base el eje x. Por tanto, si se quiere obtener obtener $R(-\alpha)$ es necesario obtener la expresión del coseno y seno de α. Estas expresiones son fácilmente deducibles del esquema de la figura 3.8.

$$cos(-\alpha) = cos(\alpha) = \frac{(1,0)\cdot\vec{v}_1}{\|\vec{v}_1\|} = \frac{v_{1x}}{\|\vec{v}_1\|} \tag{3.28}$$

$$sen(-\alpha) = -sen(\alpha) = -\frac{(0,1)\cdot\vec{v}_1}{\|\vec{v}_1\|} = -\frac{v_{1y}}{\|\vec{v}_1\|} \tag{3.29}$$

Por tanto, la matriz de rotación quedaría de la siguiente forma:

$$R(-\alpha) = \frac{1}{\|\vec{v}_1\|}\begin{pmatrix} v_{1x} & v_{1y} \\ -v_{1y} & v_{1x} \end{pmatrix} \tag{3.30}$$

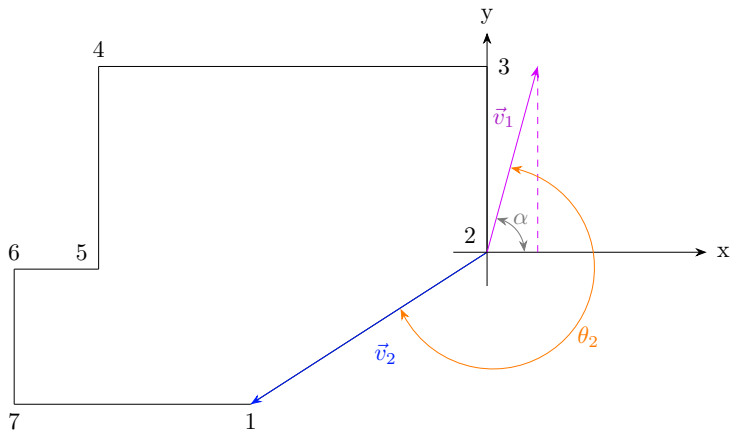


Figura 3.8: Representación de los vectores directores empleados para la obtención de los puntos de concavidad.

Por último, para comprobar si el vértice es un punto de concavidad se rota el vector v_2 mediante la matriz de rotación $R(-\alpha)$ y si la coordenada y de v_2' cumple la ecuación 3.31 se trata de un punto de concavidad. Para ayudar a la comprensión del razonamiento se pueden consultar la figura 3.9 donde se analiza el vértice 2 que no provoca concavidad y la figura 3.10 donde se analiza el vértice 5 que efectivamente provoca concavidad.

$$(0,1)\cdot v_2' = (0,1)\cdot[R(-\alpha)\cdot v_2] < 0 \tag{3.31}$$

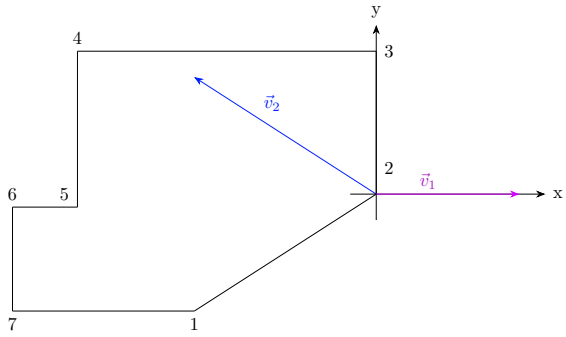


Figura 3.9: Representaciones vectores rotados para vértice normal.

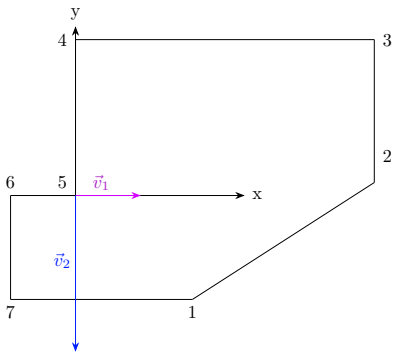


Figura 3.10: Representaciones vectores rotados para un notch.

3.2.3.2. División en subpolígonos a partir de los puntos de concavidad

Una vez obtenidos los puntos de concavidad para obtener la subdivisión en convexos se van recorriendo los notches donde se biseca el ángulo interior formado por los vectores directores vistos anteriormente. Para calcular el vector bisector se forma un cuadrado mediante los vectores directores y se calcula su diagonal de la siguiente forma (véase figura 3.11):

$$\vec{v}_{diag} = \vec{v}_1 \left\|\vec{v}_2\right\| + \vec{v}_2 \left\|\vec{v}_1\right\| \tag{3.32}$$

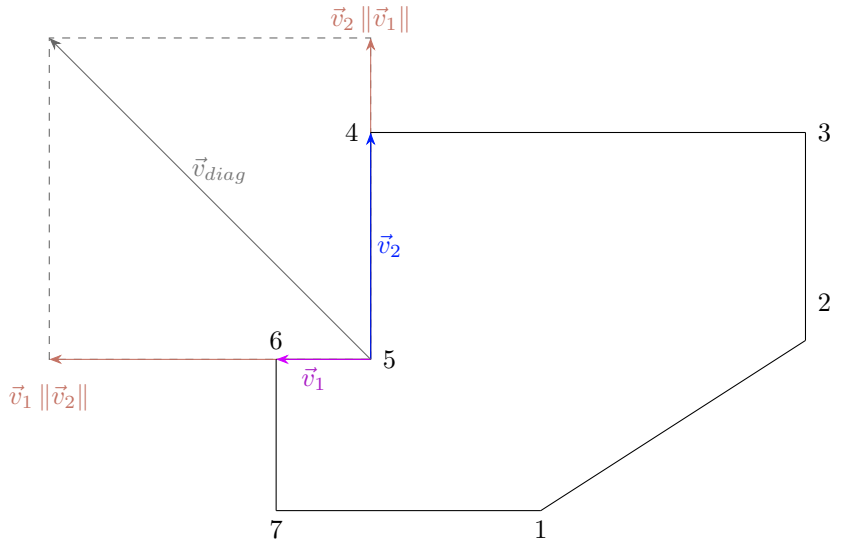


Figura 3.11: Representación del vector bisector v_{diag} de un punto de concavidad.

Una vez obtenido el vector bisector se calcula la intersección formada por la recta con dirección el bisector y punto el notch con la frontera del electrodo. Una vez obtenida la intersección se divide el polígono en dos subpolígonos, uno a la derecha del bisector y otro a la izquierda (véase figura 3.12), dentro de los cuales se vuelve a comprobar si son convexos y en caso contrario continuar el proceso hasta que todos los subpolígonos resultantes sean convexos.

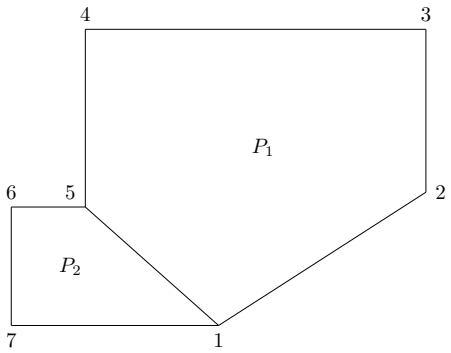


Figura 3.12: Polígonos resultantes de la división en convexos de una frontera.

3.2.4. Conversión a inecuaciones

Como las rutinas de optimización necesitan tener la expresión de las regiones mediante inecuaciones se debe procesar los subpolígonos convexos obtenidos para pasar de poligonales a sistemas de inecuaciones para que la expresión final sea de la forma:

$$A \begin{pmatrix} x \\ y \end{pmatrix} \leq B \tag{3.33}$$

Para ello, se emplean las ecuaciones estudiadas en la sección 3.2.1.1. Un detalle importante es asegurarse de que cada fila del sistema de ecuaciones tiene el signo correcto. Para verificar esto, se calcula un punto auxiliar en el interior de la frontera, el cual debe cumplir con la inecuación correspondiente. Si no la cumpliese se debe multiplicar la fila por -1. Para calcular dicho punto basta con obtener el punto medio de 3 vértices. El punto siempre estará en el interior del polígono al ser convexo. Para entender este procedimiento véase la figura 3.13.

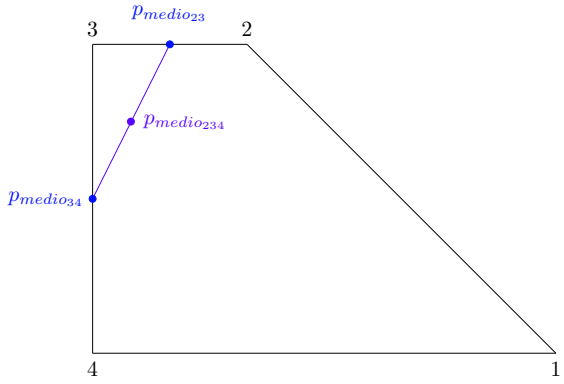


Figura 3.13: Representación del punto medio auxiliar para el cálculo del signo de las inecuaciones.

3.3. Algoritmo determinista

3.3.1. Gradiente conjugado

Para el fundamento teórico de este algoritmo se emplea la deducción presentada en [20], la cual es referenciada en [10] usado para escribir el estado del arte de los algoritmos de optimización. Sea la función a minimizar la ecuación 3.13, expresado matemáticamente:

$$\underset{x}{\text{minimizar}}\ f(x) = K\sum_{j=1}^{2N}\sum_{k=1}^{M}\frac{L_j\lambda(r_{jk})}{\Delta_{jk}},\quad x \in X \subseteq \mathbb{R}^3 \tag{3.34}$$

$$K = \frac{\rho I_{falta}}{4\pi M},\ \Delta_{jk} = \|r_{jk} - x\| \tag{3.35}$$

El gradiente, $g(x)$, y matriz Hessiana, $G(x)$, se pueden obtener fácilmente:

$$g(x) = K\sum_{j=1}^{2N}\sum_{k=1}^{M}\frac{L_j\lambda(r_{jk})}{\Delta_{jk}^3}\begin{pmatrix}\delta_0\\ \delta_1\\ \delta_2\end{pmatrix} \tag{3.36}$$

$$G(x) = K\sum_{j=1}^{2N}\sum_{k=1}^{M}\frac{L_j\lambda(r_{jk})}{\Delta_{jk}^5}\begin{pmatrix}3\delta_0^2 - \Delta_{jk}^2 & 3\delta_0\delta_1 & 3\delta_0\delta_2\\ 3\delta_0\delta_1 & 3\delta_1^2 - \Delta_{jk}^2 & 3\delta_1\delta_2\\ 3\delta_0\delta_2 & 3\delta_1\delta_2 & 3\Delta_2^2 - \Delta_{jk}^2\end{pmatrix} \tag{3.37}$$

Donde

$$\begin{aligned}\delta_0 &= r_{0jk} - x_0\\ \delta_1 &= r_{1jk} - x_1\\ \delta_2 &= r_{2jk} - x_2\end{aligned} \tag{3.38}$$

Los algoritmos de gradiente conjugado son métodos de búsqueda lineal donde se elige la dirección, $s^{(i)}$, en base al gradiente de tal manera que se cumpla que:

$$s^{(i)T}Gs^{(j)} = 0,\ \forall i \neq j \tag{3.39}$$

Aunque en el método no se emplea la matriz Hessiana, G, esta debe ser definida positiva para asegurar la convergencia. Debido a la dificultad de demostrar que la matriz Hessiana presentada en la ecuación 3.37 es definida positiva, se asume que en un entorno cercano a la solución tendrá un comportamiento similar al de una matriz definida positiva. Teniendo esto en cuenta, se procede a la demostración de la convergencia del algoritmo siguiendo la deducción presentada en [20].

Antes de pasar a la deducción es importante recalcar que para facilitar la comprensión se usará la siguiente notación del producto escalar:

$$\langle a, b\rangle = a^T b \tag{3.40}$$

Previamente se explica el proceso de biortogonalización. Sea una matriz G de tamaño $n \times n$ simétrica y definida positiva, se construyen dos secuencias en $\mathbb{R}^n$, $g^{(0)}, g^{(1)}, ...$ y $s^{(0)}, s^{(1)}, ...$ de tal manera que:

$$\left\langle g^{(i)}, g^{(j)}\right\rangle = 0\ \forall i \neq j \tag{3.41}$$

$$\left\langle s^{(i)}, Gs^{(j)}\right\rangle = 0\ \forall i \neq j \tag{3.42}$$

Para conseguir este resultado se puede seguir el método de ortogonalización de Gram-Schmidt. Sea $g^{(0)} \in \mathbb{R}$ arbitrario. Se establece que $s^{(0)} = g^{(0)}$. A continuación, para satisfacer las condiciones de ortogonalidad se cumple que:

$$g^{(i+1)} = g^{(i)} - \lambda^{(i)} G s^{(i)} + \sum_{k=0}^{i-1} \alpha^{(k)} g^{(k)},\ \lambda^{(i)} = \frac{\left\langle g^{(i)}, g^{(i)} \right\rangle}{\left\langle g^{(i)}, G s^{(i)} \right\rangle},\ \alpha^{(k)} = 0\ \forall k \tag{3.43}$$

$$s^{(i+1)} = g^{(i+1)} + \gamma^{(i)} s^{(i)} + \sum_{k=0}^{i-1} \beta^{(k)} s^{(k)},\ \gamma^{(i)} = -\frac{\left\langle G s^{(i)}, g^{(i+1)} \right\rangle}{\left\langle G s^{(i)}, s^{(i)} \right\rangle},\ \beta^{(k)} = 0\ \forall k \tag{3.44}$$

Para obtener demostrar las ecuaciones 3.43 y 3.44, se recurre a las condiciones 3.41 y 3.42. Para ello, basta con demostrar que se cumple ortogonalidad para todo $j < i + 1$.

- Si $j = i$, el sumatorio se cancela y la expresión se cumple por las propiedades del producto escalar

$$\left\langle g^{(i+1)}, g^{(i)} \right\rangle = \left\langle g^{(i)}, g^{(i)} \right\rangle - \frac{\left\langle g^{(i)}, g^{(i)} \right\rangle}{\left\langle g^{(i)}, G s^{(i)} \right\rangle} \left\langle G s^{(i)}, g^{(i)} \right\rangle + \sum_{k=0}^{i-1} \alpha^{(k)} \underbrace{\left\langle g^{(k)}, g^{(i)} \right\rangle}_{0} \tag{3.45}$$

$$0 = \left\langle g^{(i)}, g^{(i)} \right\rangle - \frac{\left\langle g^{(i)}, g^{(i)} \right\rangle}{\left\langle g^{(i)}, G s^{(i)} \right\rangle} \left\langle G s^{(i)}, g^{(i)} \right\rangle \tag{3.46}$$

$$\left\langle s^{(i+1)}, G s^{(i)} \right\rangle = \left\langle g^{(i+1)}, G s^{(i)} \right\rangle - \frac{\left\langle G s^{(i)}, g^{(i+1)} \right\rangle}{\left\langle G s^{(i)}, s^{(i)} \right\rangle} \left\langle s^{(i)}, G s^{(i)} \right\rangle + \sum_{k=0}^{i-1} \beta^{(k)} \underbrace{\left\langle s^{(k)}, G s^{(i)} \right\rangle}_{0} \tag{3.47}$$

$$0 = \left\langle g^{(i+1)}, G s^{(i)} \right\rangle - \frac{\left\langle G s^{(i)}, g^{(i+1)} \right\rangle}{\left\langle G s^{(i)}, s^{(i)} \right\rangle} \left\langle s^{(i)}, G s^{(i)} \right\rangle \tag{3.48}$$

- Si $j < i$ es necesario asumir que $\left\langle G s^{(i)}, g^{(j)} \right\rangle = 0$:

$$\left\langle g^{(i+1)}, g^{(j)} \right\rangle = \underbrace{\left\langle g^{(i)}, g^{(j)} \right\rangle}_{0} - \frac{\left\langle g^{(i)}, g^{(i)} \right\rangle}{\left\langle g^{(i)}, G s^{(i)} \right\rangle} \underbrace{\left\langle G s^{(i)}, g^{(j)} \right\rangle}_{0} + \sum_{k=0}^{i-1} \alpha^{(k)} \left\langle g^{(k)}, g^{(j)} \right\rangle \tag{3.49}$$

$$0 = \alpha^{(j)} \left\langle g^{(j)}, g^{(j)} \right\rangle \rightarrow \alpha^{(j)} = 0 \tag{3.50}$$

$$\left\langle s^{(i+1)}, G s^{(j)} \right\rangle = \left\langle g^{(i+1)}, G s^{(j)} \right\rangle - \frac{\left\langle G s^{(i)}, g^{(i+1)} \right\rangle}{\left\langle G s^{(i)}, s^{(i)} \right\rangle} \underbrace{\left\langle s^{(i)}, G s^{(j)} \right\rangle}_{0} + \sum_{k=0}^{i-1} \beta^{(k)} \left\langle s^{(k)}, G s^{(j)} \right\rangle \tag{3.51}$$

$$0 = \underbrace{\left\langle g^{(i+1)}, \frac{g^{(j)} - g^{(j+1)}}{\lambda^{(j)}} \right\rangle}_{0} + \beta^{(j)} \left\langle s^{(j)}, G s^{(j)} \right\rangle \rightarrow \beta^{(j)} = 0 \tag{3.52}$$

Por tanto, las ecuaciones 3.43 y 3.44 se reducen a:

$$g^{(i+1)} = g^{(i)} - \lambda^{(i)} G s^{(i)},\ \lambda^{(i)} = \frac{\left\langle g^{(i)}, g^{(i)} \right\rangle}{\left\langle g^{(i)}, G s^{(i)} \right\rangle} \tag{3.53}$$

$$s^{(i+1)} = g^{(i+1)} + \gamma^{(i)} s^{(i)},\ \gamma^{(i)} = -\frac{\left\langle G s^{(i)}, g^{(i+1)} \right\rangle}{\left\langle G s^{(i)}, s^{(i)} \right\rangle} \tag{3.54}$$

A continuación, se demuestran las siguientes expresiones que servirán como base para el algoritmo:

$$\left\langle s^{(i)}, g^{(k)} \right\rangle = 0,\ \forall i < k \tag{3.55}$$

$$\lambda^{(i)} = \frac{\left\langle g^{(i)}, g^{(i)} \right\rangle}{\left\langle g^{(i)}, G s^{(i)} \right\rangle} = \frac{\left\langle g^{(i)}, s^{(i)} \right\rangle}{\left\langle s^{(i)}, G s^{(i)} \right\rangle} \tag{3.56}$$

$$\gamma^{(i)} = -\frac{\left\langle G s^{(i)}, g^{(i+1)} \right\rangle}{\left\langle G s^{(i)}, s^{(i)} \right\rangle} = \frac{\left\langle g^{(i+1)}, g^{(i+1)} \right\rangle - \left\langle g^{(i+1)}, g^{(i)} \right\rangle}{\left\langle g^{(i)}, g^{(i)} \right\rangle} \tag{3.57}$$

Para demostrar la expresión 3.55:

$$\left\langle s^{(i)}, g^{(k)} \right\rangle = \left\langle s^{(i)}, g^{(k)} - \lambda^{(k-1)} G s^{(k-1)} \right\rangle = \left\langle s^{(i)}, g^{(k)} \right\rangle = \left\langle s^{(i)}, g^{(i+1)} \right\rangle \tag{3.58}$$

$$= \left\langle s^{(i)}, g^{(i+1)} \right\rangle = \left\langle s^{(i)}, g^{(i)} - \lambda^{(i)} G s^{(i)} \right\rangle = \left\langle s^{(i)}, g^{(i)} \right\rangle - \lambda^{(i)} \left\langle s^{(i)}, G s^{(i)} \right\rangle \tag{3.59}$$

$$= \left\langle g^{(i)} + \gamma^{(i-1)} s^{(i-1)}, g^{(i)} \right\rangle - \lambda^{(i)} \left\langle s^{(i)}, G s^{(i)} \right\rangle \tag{3.60}$$

$$= \left\langle g^{(i)}, g^{(i)} \right\rangle - \lambda^{(i)} \left\langle s^{(i)}, G s^{(i)} \right\rangle + \gamma^{(i-1)} \left\langle s^{(i-1)}, g^{(i)} \right\rangle \tag{3.61}$$

En la ecuación 3.61 se puede observar como el resultado depende recursivamente del producto anterior $\left\langle s^{(i-1)}, g^{(i)} \right\rangle$. Por tanto en el límite, $i = 0$, se tendrá el siguiente resultado:

$$\left\langle s^{(0)}, g^{(1)} \right\rangle \rightarrow s^{(0)} = g^{(0)} \rightarrow \left\langle s^{(0)}, g^{(1)} \right\rangle = 0 \tag{3.62}$$

Para demostrar la expresión 3.56:

$$\lambda^{(i)} = \frac{\left\langle g^{(i)}, g^{(i)} \right\rangle}{\left\langle g^{(i)}, G s^{(i)} \right\rangle} = \frac{\left\langle s^{(i)} - \gamma^{(i)} s^{(i-1)}, g^{(i)} \right\rangle}{\left\langle s^{(i)} - \gamma^{(i)} s^{(i-1)}, G s^{(i)} \right\rangle} = \frac{\left\langle g^{(i)}, s^{(i)} \right\rangle}{\left\langle s^{(i)}, G s^{(i)} \right\rangle} \tag{3.63}$$

Para demostrar la expresión 3.57:

$$\gamma^{(i)} = -\frac{\left\langle G s^{(i)}, g^{(i+1)} \right\rangle}{\left\langle G s^{(i)}, s^{(i)} \right\rangle} = -\frac{\left\langle g^{(i+1)} - g^{(i)}, g^{(i+1)} \right\rangle}{\left\langle g^{(i+1)} - g^{(i)}, s^{(i)} \right\rangle} = \frac{\left\langle g^{(i+1)}, g^{(i+1)} \right\rangle - \left\langle g^{(i)}, g^{(i+1)} \right\rangle}{\left\langle g^{(i)}, s^{(i)} \right\rangle} \tag{3.64}$$

$$= \frac{\left\langle g^{(i+1)}, g^{(i+1)} \right\rangle - \left\langle g^{(i)}, g^{(i+1)} \right\rangle}{\left\langle g^{(i)}, g^{(i)} + \gamma^{(i+1)} s^{(i-1)} \right\rangle} = \frac{\left\langle g^{(i+1)}, g^{(i+1)} \right\rangle - \left\langle g^{(i+1)}, g^{(i)} \right\rangle}{\left\langle g^{(i)}, g^{(i)} \right\rangle} \tag{3.65}$$

Una vez asentados estos fundamentos de la biortogonalización, en el libro el autor propone el algoritmo Polak-Ribière:

Algoritmo 1 Polak-Ribiére

1: Seleccionar un $x^{(0)} \in \mathbb{R}^n$. Si $\nabla f\left(x^{(0)}\right) = 0$, parar; sino, ir al paso 1.
2: Asignar $i = 0$ y asignar $g^{(0)} = s^{(0)} = -\nabla f\left(x^{(0)}\right)$
3: Calcular $\lambda^{(i)} > 0$ tal que

$$f\left(x^{(i)} + \lambda^{(i)} s^{(i)})\right) = \text{mín}\{f\left(x^{(i)} + \lambda s^{(i)}\right) \mid \lambda \geq 0\}$$

4: Asignar

$$x^{(i+1)} = x^{(i)} + \lambda^{(i)} s^{(i)}$$

5: Calcular $\nabla f\left(x^{(i+1)}\right)$.
6: Si $\nabla f\left(x^{(i+1)}\right) = 0$, parar; sino, asignar

$$g^{(i+1)} = -\nabla f\left(x^{(i+1)}\right),$$

7: Si $i \bmod n \leq n + 1$,

$$s^{(i+1)} = g^{(i+1)} + \gamma^{(i)} s^{(i)}, \quad \text{con} \quad \gamma^{(i)} = \frac{\langle g^{(i+1)} - g^{(i)}, g^{(i+1)} \rangle}{\langle g^{(i)}, g^{(i)} \rangle},$$

sino,

$$s^{(i+1)} = g^{(i+1)}$$

asignar $i = i + 1$, e ir al paso 2.

Para comprobar que esta formulación efectivamente corresponde con la teoría desarrollada anteriormente basta con recurrir al desarrollo en serie de Taylor:

$$-g^{(i+1)} = -g(x^{(i+1)}) + \lambda^{(i)} G^{(i)} s^{(i)}, \; G^{(i)} = \int_0^1 G\left(z^{(i)} + t\lambda^{(i)} s^{(i)}\right) dt \tag{3.66}$$

A continuación, para demostrar que el algoritmo converge rápidamente, se mostrará que para una matriz definida positiva, el ángulo, $\theta^{(i)}$, formado entre el gradiente, $g^{(i)}$, y la dirección, $s^{(i)}$, es pequeño, y por tanto, los pasos tomados se acercan a la pendiente de máximo descenso aumentando la eficiencia del algoritmo.

$$\gamma^{(i)} = -\frac{\left\langle Gs^{(i)}, g^{(i+1)} \right\rangle}{\left\langle Gs^{(i)}, s^{(i)} \right\rangle} \rightarrow \left|\gamma^{(i)}\right| \leq \frac{\left\|g^{(i+1)}\right\| \left\|G^{(i)}\right\| \left\|s^{(i)}\right\|}{m \left\|s^{(i)}\right\|^2} \leq \frac{M \left\|g^{(i+1)}\right\|}{m \left\|s^{(i)}\right\|} \tag{3.67}$$

Donde m y M son el menor y mayor autovalor de $G^{(i)}$ respectivamente. Por tanto, que se cumple:

$$n \|x\|^2 \leq \left\langle x, G^{(i)} x \right\rangle \leq M \|x\|^2 \tag{3.68}$$

Usando la desigualdad triangular:

$$s^{(i+1)} = g^{(i+1)} + \gamma^{(i)} s^{(i)} \rightarrow \left\|s^{(i+1)}\right\| \leq \left\|g^{(i+1)}\right\| + \left|\gamma^{(i)}\right| \left\|s^{(i)}\right\| \tag{3.69}$$

Incluyendo 3.67 en 3.69 se obtiene:

$$\left\|s^{(i+1)}\right\| \leq \left\|g^{(i+1)}\right\| \left(1 + \frac{M}{m}\right) \tag{3.70}$$

Para terminar con la demostración, el valor del ángulo $\theta^{(i)}$ queda acotado por:

$$cos\left(\theta^{(i)}\right) = \frac{\left\langle s^{(i+1)}, g^{(i+1)} \right\rangle}{\left\|s^{(i+1)}\right\| \left\|g^{(i+1)}\right\|} = \frac{\left\langle g^{(i+1)} + \gamma^{(i)} s^{(i)}, g^{(i+1)} \right\rangle}{\left\|s^{(i+1)}\right\| \left\|g^{(i+1)}\right\|} = \frac{\left\|g^{(i+1)}\right\|^2}{\left\|s^{(i+1)}\right\| \left\|g^{(i+1)}\right\|} \tag{3.71}$$

$$cos\left(\theta^{(i)}\right) = \frac{\left\|g^{(i+1)}\right\|}{\left\|s^{(i+1)}\right\|} \geq \frac{\left\|g^{(i+1)}\right\|}{\left\|g^{(i+1)}\right\|\left(1+\frac{M}{m}\right)} = \frac{1}{1+\frac{M}{m}} \tag{3.72}$$

Con una cota inferior para el coseno se garantiza que el ángulo tiene una cota superior.

Una vez introducido el algoritmo usado para el cálculo de la dirección, como el método de gradiente es un algoritmo de búsqueda lineal una vez obtenida la dirección de búsqueda, $s^{(i)}$, es necesario determinar un paso adecuado, $\lambda^{(i)}$. La elección del paso se realiza de tal manera que se minimice la función objetivo 3.34 en la siguiente recta:

$$x^{(i+1)} = x^{(i)} + \lambda^{(i)} s^{(i)} \tag{3.73}$$

Para obtener el valor del paso que minimiza la función objetivo se recurre a las condiciones de Wolfe. Mediante las condiciones de Wolfe se puede tener un criterio objetivo para poder considerar un paso válido. Las condiciones extraídas de [10] son las siguientes:

- La primera condición de Wolfe considera válidos solo aquellos puntos que cumplen:

$$f\left(x^{(i)} + \alpha s^{(i)}\right) \leq f\left(x^{(i)}\right) + \varepsilon_1 \alpha g\left(x^{(i)}\right)^T s^{(i)},\ 0 < \varepsilon_1 < 1 \tag{3.74}$$

  Esta condición no es más que requerir que los puntos estén por debajo de una recta con una pendiente proporcional al gradiente.

- La primera condición de Wolfe considera válidos solo aquellos puntos que cumplen:

$$g\left(x^{(i)} + \alpha s^{(i)}\right) \geq \varepsilon_2 g\left(x^{(i)}\right),\ \varepsilon_1 < \varepsilon_2 < 1 \tag{3.75}$$

  Esta condición obliga a que el punto elegido tenga una pendiente "menor", ya que a medida que uno se acerca al mínimo, la pendiente tiende a 0. Aunque en la inecuación haya un mayor o igual, esto se debe a que se desciende y las pendientes son negativas.

3.3.2. Algoritmos con restricciones

Para la elaboración de este apartado la información empleada se ha extraído del siguiente libro [21]. La función a minimizar es la planteada en la ecuación 3.34, pero con las restricciones de desigualdad obtenidas mediante el procesamiento geométrico:

$$\begin{gathered}\underset{x}{\text{minimizar}}\ f(x) = K\sum_{j=1}^{2N}\sum_{k=1}^{M}\frac{L_j\lambda(r_{jk})}{\Delta_{jk}},\quad x \in X \subseteq \mathbb{R}^3 \\ \text{sujeto a } x_3 = 0\ ;\ B_j - A_j x' \geq 0,\ j \in J\ x' \in \mathbb{R}^2\end{gathered} \tag{3.76}$$

Con su respectivo Lagrangiano:

$$L(x,\lambda) = f(x) - \sum_j \lambda_j\left(B_j - A_j x\right) - \lambda_z x_3 \tag{3.77}$$

3.3.2.1. Deducción de las condiciones Karush–Kuhn–Tucker (KKT)

Como los potenciales a optimizar se estudian a nivel de suelo, $x_3 = 0$, se puede tomar que $x = x' \in \mathbb{R}^2$, y así reducir el espacio de diseño. Las matrices A_j y B_j son los coeficientes que definen la región convexa a optimizar. Antes de deducir las condiciones KKT, se establecen algunas definiciones que serán útiles más adelante.

Se dice que un punto x es factible si cumple las restricciones planteadas en el sistema.

Se dice que una restricción de desigualdad es activa si se cumple $B_j - A_j x = 0$, e inactiva si $B_j - A_j x > 0$. Así, se define el conjunto de restricciones activas como $\mathcal{A}(x) = \{j \mid B_j - A_j x = 0\}$.

Para construir el Lagrangiano, se introducen multiplicadores λ_j que penalizan cualquier violación de las restricciones. Esto asegura que el algoritmo minimice $f(x)$ sin ignorar las condiciones impuestas. Así, el Lagrangiano se define como:

$$L(x, \lambda) = f(x) - \sum_j \lambda_j (B_j - A_j x) - \lambda_z x_3 \tag{3.78}$$

Donde $\lambda_j \geq 0$ garantiza que las restricciones activas reduzcan el valor de la función objetivo.

Se define que la condición de cualificación linealmente restringida (LICQ, por sus siglas en inglés, *Linear Independence Constraint Qualification*) que se cumple cuando dado x^* y $\mathcal{A}(x^*)$ los gradientes de las restricciones del conjunto activo son linealmente independientes. En el caso de restricciones definidas mediante polígonos convexos en $\mathbb{R}^2$, las condiciones LICQ se cumplen siempre (Véase figura 3.14 y la figura 3.15).

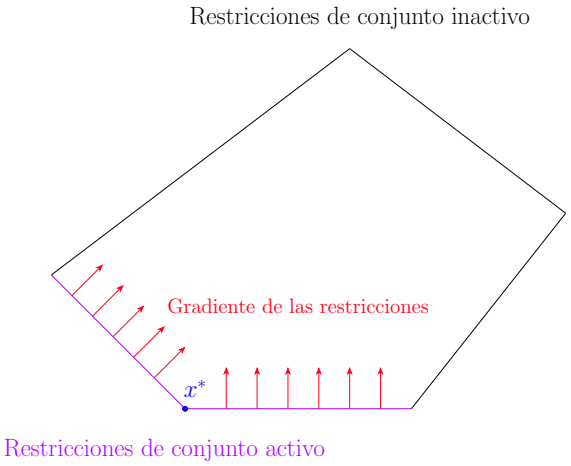


Figura 3.14: El óptimo, x^*, coincide con un vértice del polígono. Como los lados del polígono no pueden ser coincidentes, los gradientes siempre son linealmente independientes.

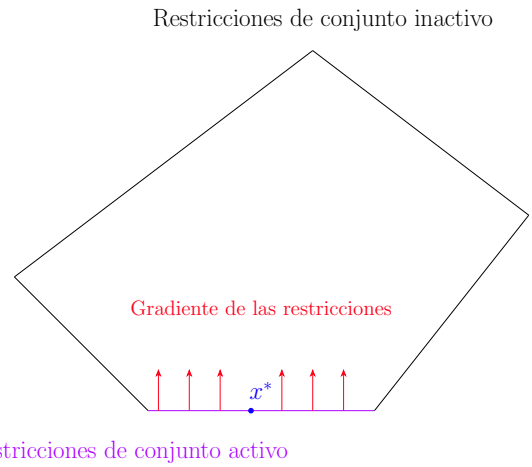


Figura 3.15: El óptimo, x^*, cae en un lado del polígono. Al solo estar activa una restricción la condición se cumple trivialmente.

Las condiciones LICQ garantizan que existe unicidad en el calculo de los multiplicadores de Lagrange y, por tanto aportan estabilidad al calculo.

Teniendo esto en cuenta, se vuelven a plantear las condiciones KKT:

$$\begin{cases} \nabla_x L(x^*, \lambda^*) = 0 \\ x_3^* = 0 \\ B_j - A_j x^* \geq 0 \\ \lambda_j^* \geq 0 \\ \lambda_j^* \left[B_j - A_j x^*\right] = 0 \end{cases} \tag{3.79}$$

A partir de estas condiciones, es evidente que cuando una restricción es inactiva, esta no influye en la solución. Esto se puede justificar directamente observando lo siguiente:

$$\lambda_j^* \left[B_j - A_j x^*\right] = 0 \implies B_j - A_j x^* > 0 \implies \lambda_j^* = 0 \tag{3.80}$$

Secuencias factibles

Para demostrar rigurosamente que una restricción inactiva no afecta la solución, se recurre al concepto de secuencias factibles. Se define una secuencia factible como toda secuencia $\{z_k\}_{k=0}^{\infty} \in \mathbb{R}^3$ tal que, dado un punto x^*, cumple las siguientes propiedades:

- $z_k \neq x^* \quad \forall k$,
- $\lim\limits_{k\to\infty} z_k = x^*$,
- z_k es factible para valores grandes de k.

El conjunto de todas estas secuencias se denota como $\mathcal{T}(x^*)$, el cual tiene la propiedad de que, para valores grandes de k, respecto a la solución local en x^*, se cumple:

$$f(z_k) \geq f(x^*) \tag{3.81}$$

Con base en esto, se define una dirección límite, d_k, para una secuencia factible como:

$$d_k = \frac{z_k - x^*}{\|z_k - x^*\|}, \quad \text{y en el límite:} \quad d = \lim_{k\to\infty} \frac{z_k - x^*}{\|z_k - x^*\|} \tag{3.82}$$

A partir de esto, se enuncia el siguiente teorema: si x^* es una solución local, entonces todas las secuencias factibles z_k en $\mathcal{T}(x^*)$ satisfacen:

$$\nabla f(x^*)^T \cdot d \geq 0 \tag{3.83}$$

Para demostrarlo, se emplea una reducción al absurdo. Supóngase que existe z_k tal que $\nabla f(x^*)^T \cdot d < 0$ para alguna dirección límite d. Truncando la función por Taylor, se tiene:

$$f(z_k) = f(x^*) + (z_k - x^*)^T \nabla f(x^*) + o\left(\|z_k - x^*\|\right) \tag{3.84}$$

Dado que los términos de segundo orden se vuelven despreciables cuando $k \to \infty$, se puede establecer:

$$o(\|z_k - x^*\|) < \kappa \|z_k - x^*\| \cdot |\nabla f(x^*)^T \cdot d|, \quad \text{con } 0 < \kappa < 1 \tag{3.85}$$

Sustituyendo en la ecuación (3.84) y asumiendo $\kappa = 1/2$, se obtiene:

$$f(z_k) < f(x^*) + \frac{1}{2} \|z_k - x^*\| \cdot \nabla f(x^*)^T \cdot d \tag{3.86}$$

Esto contradice la suposición de que x^* es un mínimo local, ya que implicaría la existencia de un punto z_k donde la función toma un valor menor que en x^*. Por lo tanto, debe cumplirse la ecuación (3.83).

Con este teorema, se justifica por qué se pueden ignorar las restricciones inactivas al formular las condiciones de óptimo. Si una restricción es inactiva, debido al carácter asintótico de las secuencias factibles, permanecerá inactiva para toda la secuencia $\{z_k\}$.

Para garantizar que la última condición de las ecuaciones KKT se cumpla, se asigna a los multiplicadores de Lagrange el siguiente valor, asegurando además que las restricciones inactivas no afectan a la solución:

$$\lambda_j^* = \begin{cases} \lambda_j, & j \in \mathcal{A}(x^*) \\ 0, & \text{en otro caso} \end{cases} \tag{3.87}$$

Es importante destacar que, para garantizar que el punto encontrado sea un mínimo, deben verificarse las condiciones de segundo orden, es decir, la curvatura de la matriz Hessiana. Sin embargo, es válido asumir que la función es convexa en el entorno de la solución. Esto es un resultado de ejecutar esta optimización tras ejecutar el algoritmo genético.

3.3.2.2. Método de punto interior

El primer paso que se realiza cuando se elaboran estos algoritmos consiste en reescribir las condiciones KKT de la siguiente manera:

$$\begin{cases} \nabla f(x) + A^T \lambda = 0 \\ S\lambda - \mu \mathcal{I} = 0 \\ B - Ax - s = 0 \\ \lambda \geq 0 \\ s \geq 0 \\ \mu \geq 0 \end{cases} \tag{3.88}$$

Donde:

- A es la matriz de coeficientes de las restricciones de desigualdad.
- λ es el vector de multiplicadores de Lagrange asociados a las restricciones.
- s es el vector de variables de holgura que transforma las restricciones de desigualdad en restricciones de igualdad.
- S es una matriz diagonal cuyas entradas corresponden a las variables de holgura s.
- μ es un parámetro de barrera que controla la proximidad a la frontera de las restricciones.
- $\mathcal{I}$ es un vector cuyas entradas son todas iguales a 1.

Como se puede observar, si el valor de μ tiende a cero ($\mu \to 0$) las condiciones KKT coinciden con las planteadas en 3.88. Por lo tanto, el algoritmo de punto interior consiste en calcular iterativamente las condiciones KKT perturbadas hasta obtener una solución viable, reduciendo progresivamente el valor de μ.

Aunque existen diversos métodos para disminuir este valor, una estrategia común es la siguiente:

$$\mu^{(i+1)} \in \left[0, \sigma\mu^{(i)}\right] \tag{3.89}$$

Donde σ es un valor comprendido entre 0 y 1, que garantiza la convergencia del parámetro de barrera. Este recibe dicho nombre porque una manera efectiva de asegurar el cumplimiento de las condiciones KKT es modificar la función objetivo de modo que el problema se transforme en:

$$k(x) = f(x) - \mu \sum_j \log\left[B_j - A_j x\right] \tag{3.90}$$

Que como problema con restricciones, incluyendo el Lagrangiano:

$$\begin{aligned} \underset{x}{\text{minimizar}} \quad & L_k(x, s, \lambda) = f(x) - \mu \sum_j \log s_j - \sum_j \lambda_j \left(B_j - A_j x\right) \\ \text{sujeto a:} \quad & B - Ax - s = 0 \\ & S\lambda - \mu\mathcal{I} = 0 \end{aligned} \tag{3.91}$$

No es necesario imponer desigualdades adicionales sobre μ, ya que, por construcción, siempre es no negativa. Tampoco sobre s, debido a que el logaritmo no está definido para valores negativos. Finalmente, no es necesario restringir λ, ya que toma valores positivos debido a su relación con los parámetros λ y μ, como se muestra en la ecuación 3.92.

A continuación, para eliminar las restricciones se despejan las ecuaciones:

$$\begin{aligned} s &= B - Ax \\ \lambda &= \mu S^{-1}\mathcal{I} \end{aligned} \tag{3.92}$$

Como la matriz S es diagonal:

$$S = \begin{bmatrix} s_1 & 0 & \cdots & 0 & \cdots \\ 0 & s_2 & \cdots & 0 & \cdots \\ \vdots & \vdots & & \vdots & \\ 0 & 0 & \cdots & s_j & \cdots \\ \vdots & \vdots & & \vdots & \ddots \end{bmatrix} \tag{3.93}$$

Su inversa queda como:

$$S^{-1} = \begin{bmatrix} \dfrac{1}{B_1 - A_1 x} & 0 & \cdots & 0 & \cdots \\ 0 & \dfrac{1}{B_2 - A_2 x} & \cdots & 0 & \cdots \\ \vdots & \vdots & & \vdots & \\ 0 & 0 & \cdots & \dfrac{1}{B_j - A_j x} & \cdots \\ \vdots & \vdots & & \vdots & \ddots \end{bmatrix} \tag{3.94}$$

Por tanto, sustituyendo en la ecuación inicial se obtiene:

$$\underset{x}{\text{minimizar}} \quad L_k(x) = f(x) - \mu \sum_j \log\left(B - Ax\right) - \mathcal{K} \tag{3.95}$$

Que como se puede observar es un problema que se puede resolver mediante el método de gradiente conjugado discutido anteriormente. Con su respectivo gradiente:

$$g(x) = \nabla f(x) + \mu \sum_j \frac{A}{B - Ax} \tag{3.96}$$

3.4. Algoritmo genético

El uso de un algoritmo genético resulta especialmente interesante para la optimización de los potenciales de puesta a tierra, ya que la función, como se observa en la figura 3.16, presenta numerosos valles. Debido a esto, los algoritmos deterministas tienden a quedar atrapados en mínimos locales. Para la elaboración de este apartado, la información utilizada se ha extraído del libro [22].

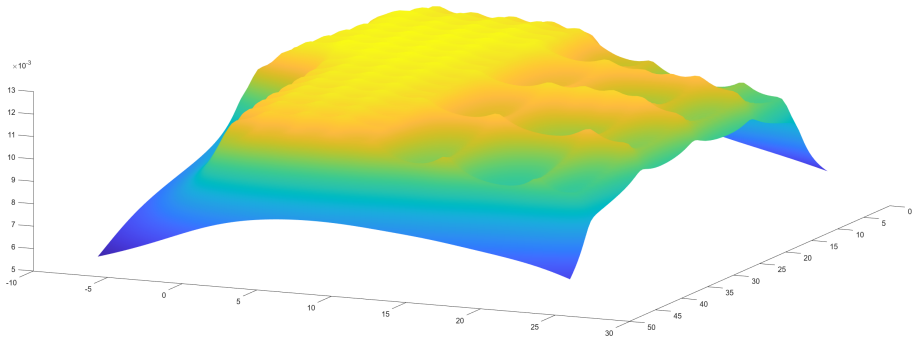


Figura 3.16: Representación del potencial de un electrodo de puesta a tierra.

3.4.1. Introducción

Antes de analizar el funcionamiento de estos algoritmos, es necesario aclarar que las técnicas estudiadas en este libro están diseñadas para problemas de maximización de la función objetivo. No obstante, este problema es dual al de minimización, y basta con realizar el siguiente cambio en la función objetivo:

$$f_2(x) = -f_1(x) \tag{3.97}$$

3.4.1.1. Funcionamiento básico

El funcionamiento de un algoritmo genético básico es relativamente simple, aunque la explicación de por qué se obtienen los resultados deseados es más compleja. El primer paso consiste en crear una población inicial de n vectores de bits generados al azar, los cuales codifican de manera discreta los valores que pueden tomar las variables en el espacio de diseño.

A continuación, en las sucesivas iteraciones del algoritmo se realizan los siguientes 3 pasos con el siguiente ejemplo de valores iniciales:

Numero	Cadena	Valor $f(x)$	% del total
1	10110	596	48.7
2	10100	373	30.4
3	00110	226	18.4
4	11110	30	2.5
Total		1225	100.0

Tabla 3.1: Tabla con los valores iniciales elegidos como ejemplo de algoritmo genético.

1. **Reproducción**: En este paso, se eligen los individuos a copiar en función de su valor en la función objetivo. Este operador selecciona las cadenas (strings) de manera que aquellos con un mayor valor en la función objetivo tengan una mayor probabilidad de ser elegidos. Esto se asemeja a una "ruleta trucada' ' (véase la figura 3.17), donde la ruleta se gira y el individuo seleccionado para la siguiente generación es aquel en el que cae.

   De esta manera, se introduce un componente de aleatoriedad en la selección, pero con un sesgo que favorece la reproducción de los mejores individuos, asegurando así la transmisión de características favorables a las siguientes generaciones.

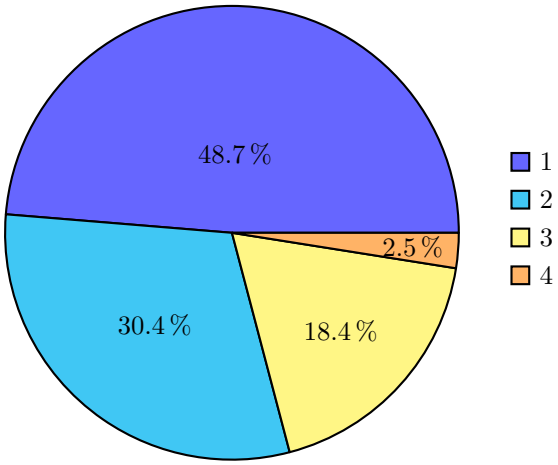


Figura 3.17: Representación gráfica de la "ruleta trucada' ' que se gira para obtener los vectores a reproducir.

2. **Cruce**: una vez elegidos los individuos para la reproducción se realiza el cruce. Para ello, se eligen 2 individuos al azar de los posibles candidatos. A continuación, se selecciona de manera uniforme un número entero k que representa la posición de corte en la cadena, asegurando que k esté entre 1 y el penúltimo carácter de la cadena, es decir, $1 \leq k \leq l-1$.

   Después de determinar k, se intercambian las subcadenas resultantes para generar dos nuevos individuos. A continuación, se muestra un ejemplo:

   Sean A_1 y A_2 los individuos elegidos con $k = 4$. Los nuevos individuos A'_1 y A'_2 son:

$$\left.\begin{array}{l} A_1 = 0 \quad 1 \quad 1 \quad 0 \quad | \quad 1 \\ A_2 = 1 \quad 1 \quad 0 \quad 0 \quad | \quad 0 \end{array}\right\} \rightarrow \begin{cases} A'_1 = 0 \quad 1 \quad 1 \quad 0 \quad 0 \\ A'_2 = 1 \quad 1 \quad 0 \quad 0 \quad 1 \end{cases} \tag{3.98}$$

3. **Mutación**: como paso final y con un rol secundario, este operador se utiliza para recuperar bits (1s o 0s) que, de otro modo, podrían perderse de forma irreversible. La mutación se aplica con una pequeña probabilidad, modificando aleatoriamente algunos de los bits en los individuos resultantes.

3.4.2. Fundamentos matemáticos

3.4.2.1. Notación

Para describir cadenas de caracteres, se denotarán mediante letras mayúsculas, mientras que se utilizarán letras minúsculas para representar cada uno de los caracteres individuales (a veces denominados alelos o características en la literatura). Es importante destacar que las características no necesariamente tienen un orden fijo.

Por ejemplo, las siguientes son cadenas válidas:

$$\begin{aligned} A &= a_1 a_2 a_3 a_4 a_5 a_6 a_7 \\ A' &= a_2 a_5 a_7 a_6 a_4 a_3 a_1 \end{aligned} \tag{3.99}$$

3.4.2.2. Teorema fundamental de los algoritmos genéticos

Para el planteamiento del algoritmo y su funcionalidad, es útil extender el alfabeto binario de las cadenas de bits, $\{0, 1\}$, incorporando un símbolo de irrelevancia (don't care), representado por $\{*\}$. Con esta extensión, ahora se puede trabajar con el esquema, un constructo formado por el alfabeto $V+ = \{0, 1, *\}$ que permite referirse a un subconjunto de cadenas. Por ejemplo, el esquema:

$$\{* \quad 1 \quad 1 \quad 1 \quad *\} \tag{3.100}$$

Se describe el siguiente subconjunto:

$$\{01110, \quad 01111, \quad 11110, \quad 11111\} \tag{3.101}$$

El concepto de esquema será útil pues permite describir las posibles permutaciones que pueden existir sobre una subcadena concreta.

Para caracterizar mejor las diferencias entre distintos esquemas, como $A_1 = 0 * 0 * ** $ y $A_2 = 0 * * * 0*$, se introducen dos parámetros clave:

- **Orden** $[o(H)]$: número de posiciones fijas (1s o 0s) en el esquema.
- **Longitud característica** $[\delta(H)]$: distancia entre el primer y último dígito fijo de un esquema. Si solo hay un dígito esta distancia es 0.

Sea un instante t en el que existen m ejemplos del esquema H contenidos en la población $A(t)$., Estos ejemplos se denotan como $m = m(H, t)$. Durante el proceso de reproducción, la probabilidad de que una cadena A_i sea seleccionada está dada por:

$$p_i = \frac{f_i}{\sum f_i}. \tag{3.102}$$

Una vez generada la nueva población, se espera que el número de ejemplos del esquema H sea:

$$m(H, t+1) = m(H, t) \cdot n \cdot \frac{f(H)}{\sum f_i} = m(H, t) \cdot \frac{f(H)}{\bar{f}} \tag{3.103}$$

Donde n es el tamaño de la población y $f(H)$ representa el valor promedio de la función objetivo del esquema H.

Como se aprecia, un esquema tiende a preservarse si el cociente entre su valor objetivo y el valor promedio de la población es elevado. En caso contrario, es probable que desaparezca. Por tanto, algunos patrones se mantienen mientras que otros tienden a desaparecer.

Para analizar el funcionamiento de esta población se puede modelar que la función $f(H)$ mediante una constante c de la siguiente manera:

$$f(H) = (1 + c)\, \bar{f} \tag{3.104}$$

Por tanto, si se asume que el valor de c no cambia, se puede conocer la población de un esquema en función del valor en $t = 0$:

$$m(H, t) = m(H, 0) \cdot (1 + c)^t \tag{3.105}$$

Vista la naturaleza exponencial de preservación o eliminación de individuos durante la reproducción. A continuación, es necesario discutir el operador de cruce, puesto que es el mecanismo que permite explorar el espacio de diseño.

Aunque una cadena haya sido seleccionada durante la reproducción, dos esquemas diferentes que representen dicha cadena pueden presentar probabilidades distintas de preservarse en el cruce. Consideremos el siguiente ejemplo con la cadena A y los esquemas H_1 y H_2:

$$\begin{aligned} A &= 011100 \\ H_1 &= *1****0 \\ H_2 &= ***10** \end{aligned} \tag{3.106}$$

Se observa que, claramente, el esquema H_1 tiene una mayor probabilidad de desaparecer durante el cruce, ya que, en promedio, el punto de corte se ubicará entre las posiciones fijas. Para cuantificar este efecto se emplea la longitud característica, $\delta(H)$ (a mayor longitud, mayor es la probabilidad de que el esquema se "rompa"), y la longitud de la cadena, l (dado que solo se pueden realizar cortes en $l-1$ sitios).

La probabilidad de desaparición, p_d, se define como:

$$p_d = \frac{\delta(H)}{l-1}. \tag{3.107}$$

Por consiguiente, asumiendo una probabilidad de cruce p_{sc}, la probabilidad de que el esquema sobreviva al cruce, p_s, está acotada inferiormente por:

$$p_{sc} \geq 1 - p_c \frac{\delta(H)}{l-1} \tag{3.108}$$

Una vez realizado el cruce, cada cadena puede sufrir una mutación con probabilidad p_m. Por ello, la probabilidad de que un esquema sobreviva a la mutación, p_{sm}, depende de que ninguno de los dígitos que lo definen sea alterado. En consecuencia, se tiene que

$$p_{sm} = (1-p_m)^{o(H)} \approx 1 - o(H) \cdot p_m \tag{3.109}$$

Donde $o(H)$, el orden del esquema, indica el número de dígitos fijos y la aproximación es válida para valores pequeños de p_m.

Con estas probabilidades calculadas y sabiendo que cada uno de los operadores es independiente del anterior, se tiene la población de un esquema a lo largo de las iteraciones:

$$m(H, t+1) \geq m(H,t) \cdot \frac{f(H)}{\bar{f}} \cdot \left[1 - p_c \frac{\delta(H)}{l-1}\right] \cdot [1 - o(H) \cdot p_m] \tag{3.110}$$

Como la probabilidad de mutar es pequeña se puede aproximar por:

$$m(H, t+1) \geq m(H,t) \cdot \frac{f(H)}{\bar{f}} \cdot \left[1 - p_c \frac{\delta(H)}{l-1} - o(H) \cdot p_m\right] \tag{3.111}$$

Con este desarrollo, se ha obtenido el resultado principal del teorema principal de los algoritmos genéticos o el algoritmo del esquema ("Schema Algorythm").

3.4.2.3. Esquemas útiles. Hipótesis del bloque de construcción

La potencia real de los esquemas se fundamenta en que, aunque se procesen n estructuras, el algoritmo genético actúa como si procesara n^3 esquemas. Este resultado se conoce como "paralelismo implícito". Es decir, aunque se procesen n individuos, se obtiene un procesamiento útil de n^3 esquemas en paralelo, sin necesidad de asignar recursos ni memoria adicionales. A continuación, se desarrollan los supuestos pertinentes.

Sea una población de n individuos de longitud l. Solo se consideran esquemas con una probabilidad de supervivencia superior a una constante p_s. Si solo se consideran validos los esquemas con un error, ε:

$$\varepsilon < 1 - p_s \tag{3.112}$$

Como:

$$p_s \geq 1 - p_c \frac{\delta(H)}{l-1} - o(H) \cdot p_m = 1 - p_c \frac{l_s - 1}{l-1} - o(H) \cdot p_m \tag{3.113}$$

Entonces se puede obtener la longitud característica máxima de los esquemas l_s. Asumiendo que $o(H) \cdot p_m \approx 0$ y que $p_c \approx 1$:

$$l_s < \varepsilon(l-1) + 1 \tag{3.114}$$

Con una longitud definida, se puede estimar una cota inferior de los esquemas únicos procesados en una población aleatoria inicial. Para ello, se consideran todos los esquemas de longitud menor o igual a l_s. Supongamos el siguiente ejemplo con una ventana de tamaño $l_s = 4$ y una longitud total $l = 9$:

$$A = \boxed{1 \quad 0 \quad 1 \quad 1} \quad 0 \quad 0 \quad 0 \quad 1 \quad 0 \tag{3.115}$$

Si se fija el quinto bit, se obtienen todos los esquemas de la forma:

$$A = \boxed{\% \quad \% \quad \% \quad 1} \quad * \quad * \quad * \quad * \quad * \tag{3.116}$$

Donde el símbolo % indica que dicho bit puede ser fijo o un don't care. Claramente, existen $2^{l_s - 1}$ esquemas de esta forma, ya que $l_s - 1 = 4$ posiciones pueden ser fijas o don't care. Para ver el número total de estas ventanas, basta con desplazar la ventana una posición cada vez:

$$A = \quad * \quad \boxed{\% \quad \% \quad \% \quad 0} \quad * \quad * \quad * \quad * \tag{3.117}$$

No obstante, esta estimación sobreestima la verdadera cantidad, puesto que existe una alta probabilidad de obtener duplicados de bajo orden. Por ello, se introduce una constante de proporcionalidad K (menor que 1), que normalmente se elige como $K = 0{,}5$. Por tanto:

$$n_s \geq n \cdot 2^{l_s} \cdot \frac{l - l_s + 1}{4} \tag{3.118}$$

Si se escoge la población de tal forma que:

$$n = 2^{\frac{l_s}{2}} \tag{3.119}$$

Se obtiene que, efectivamente, el número de esquemas es el planteado anteriormente:

$$n_s \geq n^3 \cdot \frac{l - l_s + 1}{4} \quad \rightarrow \quad n_s = C\, n^3 \tag{3.120}$$

Donde C es una constante.

Una vez analizados los esquemas, se obtiene una imagen más clara del potencial de estos nuevos bloques de construcción, fundamentales para el funcionamiento del algoritmo genético. En lugar de buscar directamente cadenas con un valor elevado en la función objetivo y explorar todas las combinaciones posibles, el problema se simplifica mediante la construcción progresiva a partir de esquemas parciales de alto rendimiento.

Capítulo 4

Desarrollo

En el siguiente capítulo se describe la implementación de los algoritmos.

4.1. Implementación de las rutinas de cálculo

4.1.1. Proceso de simulación de cargas

Se usa la rutina `ProcesoCSMG` para implementar este algoritmo:

Algoritmo 2 ProcesoCSMG

Entrada: P1, P2, r, M

Salida: Kr, dens

1: Crear los puntos campo y fuente $\vec{p} = (x, y, z)$ y las longitudes L_j de cada uno de los electrodos para el cálculo de la matriz de resistencias R a partir de $P1$, $P2$ y el número de segmentos M.
2: Inicializar la matriz de resistencias $R = 0$.
3: **for** cada combinación de puntos del campo y fuente **do**
4: **if** punto campo y fuente estan en el mismo electrodo **then** ▷ Resistencia propia

$$R_{ij} = \frac{1}{4\pi} \ln \left[\frac{\frac{L_j}{2} + z_{jsk} + \sqrt{r^2 + \left(\frac{L}{2} + z_{jsk}\right)^2}}{-\frac{L_j}{2} + z_{jsk} + \sqrt{y^2 + x^2 + \left(\frac{L}{2} - z_{jsk}\right)^2}} \right] \tag{4.1}$$

$$z_{jsk} = \|\vec{p}_{js} - \vec{p}_{jk}\|$$

5: **else** ▷ Resistencia mutua

$$R_{ij} = \frac{L_j}{4\pi \|\vec{p}_{jk} - \vec{p}_{is}\|} \tag{4.2}$$

6: **end if**
7: **end for**
8: Calculo de la densidad de corriente:

$$\lambda = R^{-1} \cdot \text{ones}(2N, 1) \rightarrow I = \sum_{i}^{2N} \sum_{j}^{M} \lambda_{ij} \rightarrow \textbf{dens} = \frac{\lambda}{I} \tag{4.3}$$

9: Calculo de Kr

$$\textbf{Kr} = \frac{2}{I} \tag{4.4}$$

En el algoritmo las variables de entrada son:

- P1 y P2 son dos matrices de tamaño $2N \times 3$ con las coordenadas (x, y, z) $[m]$ de inicio y fin de cada conductor que compone el electrodo.
- r es el radio del conductor en metros.
- M es el número de segmentos utilizado para el calculo para decidir el número de subdivisiones que tendrá cada conductor. En el apartado 4.1.1.1 se desarrolla un criterio para la elección de este parámetro.

Las variables de salida son:

- Kr es el parámetro de la resistencia de puesta a tierra para un valor de intensidad de corriente $I = 1A$ y resistividad del terreno $\rho = 1\Omega/m$.
- dens es la densidad de corriente que circula por cada conductor del electrodo si la intensidad de corriente es de $I = 1A$.

4.1.1.1. Elección del parámetro M

Para elegir el valor de M se podría utilizar la rutina `obtenerParametroM` descrita a continuación en función de la tolerancia tol:

Algoritmo 3 Obtener Parámetro M

Entrada: P1, P2, r, tol

Salida: M

1: Inicializar el índice $index = 1$, la diferencia $dif(1) = \text{NaN}$ y la variable $kr(1) = \text{ProcesoCSMG}(P1, P2, r, 1)$.
2: **while** *true* **do** $index+ = 1$
3: Calcular $kr(i) = \text{ProcesoCSMG}(P1, P2, r, i)$.
4: Calcular la diferencia $dif(i) = kr(i) - kr(i-1)$.
5: **if** $|dif(i)| < tol$ **then**
6: Establecer $M = index$.
7: Salir del bucle.
8: **end if**
9: **end while**

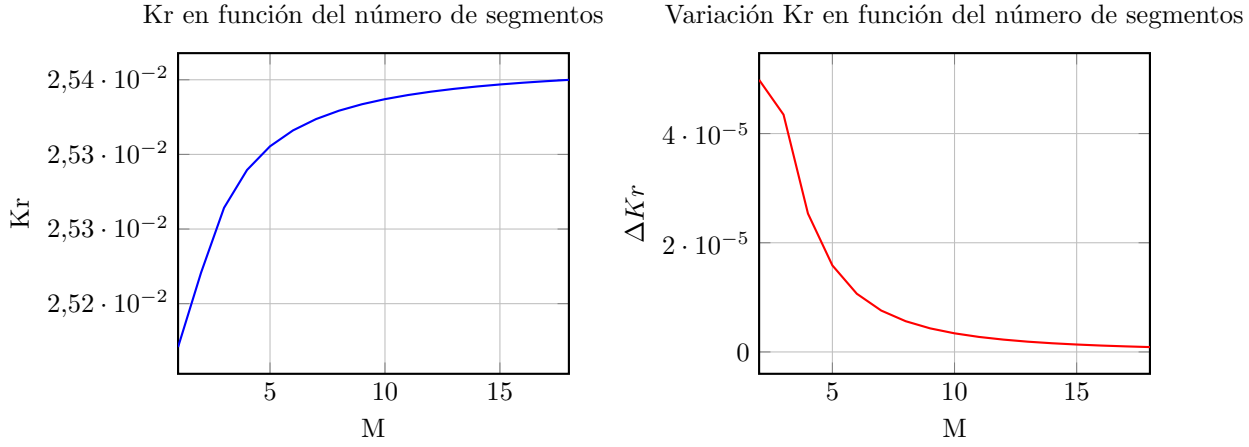


Figura 4.1: Obtención del parámetro M para una tolerancia $tol = 10^{-6}$ obteniéndose un número de segmentos óptimo de $M = 18$.

Para elegir el valor de este parámetro, M, se puede realizar un estudio pormenorizado de cada electrodo para tener un número variable de segmentos con el algoritmo anterior.En trabajo se utiliza un número fijo de segmentos de tal manera que la tolerancia entre los valores de K_r sea menor a 10^{-6}. Normalmente ese valor corresponde a un número de segmentos de $M = 10$ o $M = 12$.

4.1.2. Cálculo del potencial

El cálculo del potencial se realiza directamente por superposición tal y como se estudio en los fundamentos. Para ello, se utiliza la rutina `PotencialCSM` descrita a continuación:

Algoritmo 4 PotencialCSM

Entrada: punto, P1, P2, dens

Salida: pot

1: Calculo directo del potencial mediante un bucle:

$$pot = \frac{1}{4\pi} \sum_{j=1}^{2N} \sum_{k=1}^{M} \frac{L_j}{M} \frac{\text{dens}_{jk}}{\left\| \vec{r}_{jk} - \vec{P} \right\|} \tag{4.5}$$

Donde punto son las coordenadas $P = (x, y, z)$ del punto en el que se desea calcular el potencial, pot es el potencial en dicho punto para una resistividad $\rho = 1\Omega/m$ y una corriente de falta $I_f = 1A$. El resto de variables se han descrito anteriormente.

4.1.3. Cálculo del gradiente

El cálculo del gradiente se fundamenta en la derivada del potencial descrita anteriormente en los fundamentos. Para ello se usa la rutina `GradienteCSM` descrita a continuación.:

Algoritmo 5 GradienteCSM

Entrada: punto, P1, P2, dens

Salida: grad

1: Calculo directo del potencial mediante un bucle:

$$grad = \frac{1}{4\pi} \sum_{j=1}^{2N} \sum_{k=1}^{M} \frac{L_j}{M} \frac{dens_{jk}}{\left\| \vec{r}_{jk} - \vec{P} \right\|^3} \begin{pmatrix} r_{xjk} - P_x \\ r_{yjk} - P_y \\ r_{zjk} - P_z \end{pmatrix} \tag{4.6}$$

En este caso, la salida *grad* es una matriz de dimensiones 2×1 con los valores del gradiente en la coordenada X e Y. Dado que los algoritmos trabajan con el potencial sobre el plano $z = 0$, se descarta la tercera coordenada del gradiente.

4.2. Implementación de las rutinas de preproceso

4.2.1. Rutinas de remallado

Estas rutinas son empleadas para manipular los valores P1, P2 y r:

4.2.1.1. Rutina de fusión de electrodos alineados

Para comprimir los electrodos se emplea la rutina `fusionAlineados`.

Algoritmo 6 fusionAlineados

Entrada: P1, P2, r

Salida: P1, P2, r

while varie el tamaño del electrodo **do**
  for all conductores **do**
    Calcular el vector director para cada conductor

$$d = \frac{P_2 - P_1}{\|P_2 - P_1\|} \tag{4.7}$$

    if $d_i == \pm d_j$ **then** ▷ Dos conductores tienen la misma dirección
      if conectados(conductor i, conductor j) **then** ▷ Comprobar contigüidad
        Simplificar (conductor i, conductor j)
      end if
    end if
  end for
end while

En esta rutina, es importante comentar el motivo detrás del primer bucle **While**. Este bucle se implementa para tener en cuenta que, debido a la disposición de los electrodos, no siempre es posible simplificarlos todos en una sola iteración. Por lo tanto, la rutina debe ejecutarse hasta que ya no sea posible continuar con la simplificación.

4.2.1.2. Rutina de mallado en electrodos cortos

Este mallado se utiliza para obtener la representación descomprimida de un electrodo. Descomprimir un electrodo implica transformar los electrodos "largos" (que pueden intersecarse con otros conductores) en electrodos "cortos" (que solo pueden intersecarse con otros conductores en sus extremos). Este proceso de descompresión se lleva a cabo mediante la rutina `mallado` y se puede visualizar en la figura 4.2.

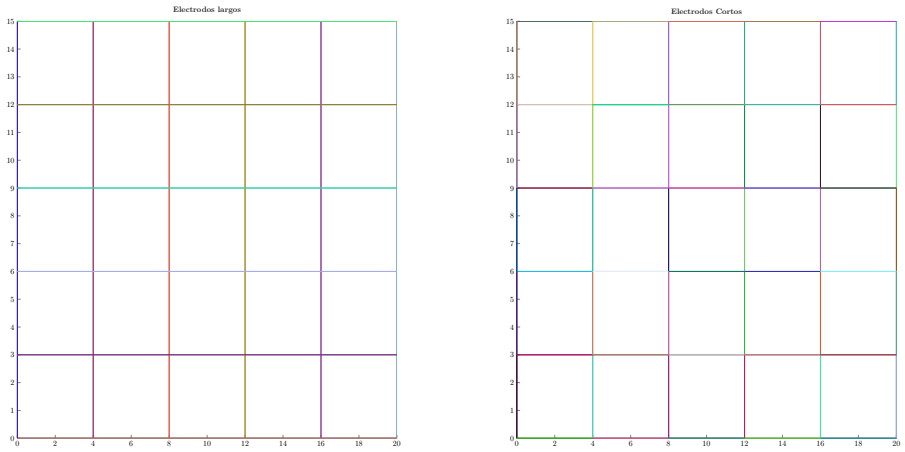


Figura 4.2: Representación mediante electrodos largos a la izquierda y electrodos cortos a la derecha para mostrar la compresión de un electrodo.

Algoritmo 7 mallado

Entrada: P1, P2, r

Salida: P1, P2, r

```
[MatTransf,Intersección]=matriz_transfer(P1,P2)        ▷ Función Auxiliar
Reordenar intersecciones para que no se solapen electrodos.
for all conductores do
    if existe intersección then
        if punto de intersección != extremo del electrodo then
            Subdividir el electrodo teniendo en cuenta el punto de intersección
        end if
    end if
end for
```

A continuación se describe la implementación de la rutina para el calculo de la matriz de transferencia.

Algoritmo 8 matriz_transfer

Entrada: P1, P2

Salida: D, C_x, C_y

for all conductores **do**
  Obtener forma matricial primer conductor:

$$A_1 \cdot \begin{pmatrix} x \\ y \end{pmatrix} = B_1 \tag{4.8}$$

  for all resto de conductores **do**
    Obtener forma matricial segundo conductor:

$$A_2 \cdot \begin{pmatrix} x \\ y \end{pmatrix} = B_2 \tag{4.9}$$

    Calcular el siguiente determinante para ver si el sistema es compatible determinado:

$$\begin{vmatrix} A_1 \\ A_2 \end{vmatrix} \tag{4.10}$$

    if determinante != 0 **then** ▷ Sistema compatible determinado
      Calcular intersección entre electrodo 1 y electrodo 2
      if punto de intersección en el interior de un electrodo **then**
        Asignar salidas

$$D_{ij} = 1,\ C_{x_{ij}} = \text{interseeción}_x,\ C_{y_{ij}} = \text{interseeción}_y \tag{4.11}$$

      end if
    end if
  end for
end for

Las variables de entrada de este algoritmo se mencionaron anteriormente mientras que las variables de salida son:

- D la matriz de transferencia, una matriz cuyo coefiente $[i, j]$ es 1 si los conductores $[i, j]$ tienen una intersección y un 0 en caso contrario.
- C_x y C_y las coordenadas (x, y) de las intersección si $D_{ij} = 1$.

4.2.1.3. Rutina de mallado en electrodos finos en la frontera

El algoritmo denominado `remallado fino` se emplea para generar un mallado en el que la frontera se subdivide en segmentos aún más pequeños (limitándose exclusivamente a la frontera, como se muestra en la figura 4.3) que los obtenidos al descomprimir un electrodo. Esta rutina adquiere especial relevancia en situaciones donde resulta crucial determinar si el valor mínimo se encuentra en una esquina del electrodo o en su interior. Este aspecto se ejemplifica en la figura 2.4, extraída de la norma IEEE-80.

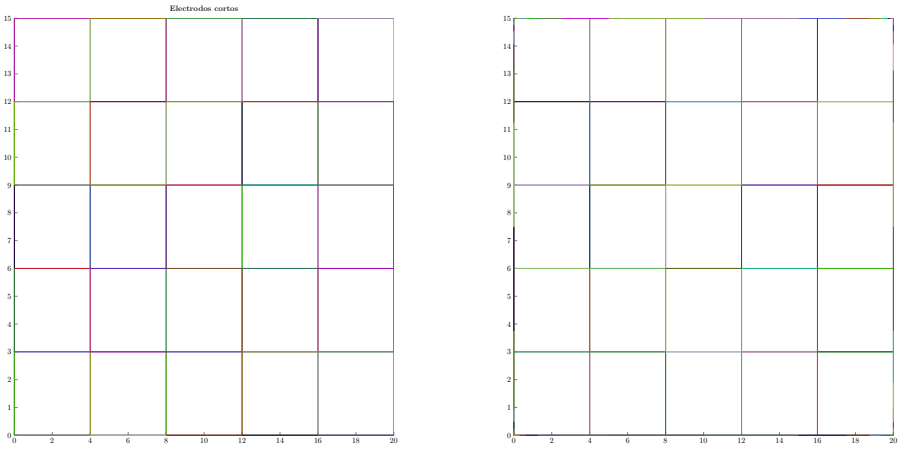


Figura 4.3: Representación de los resultados del remallado fino. A la izquierda se encuentra el electrodo representado mediante electrodos cortos y a la derecha el resultante de la rutina de `remalladofino`

.

Algoritmo 9 remalladoFino

Entrada: P1, P2, r

Salida: P1, P2, r

1: Calcular la frontera del electrodo utilizando la rutina `frontera`.
2: Separar los electrodos, almacenando por separado aquellos pertenecientes a la frontera y los que no.
3: Comprimir los electrodos en la frontera mediante `fusionAlineados`
4: **for all** conductores en la frontera **do**
5: Refinar los conductores mediante la rutina `refinado`
6: **end for**
7: Volver a combinar los electrodos

A continuación, se describe la rutina de refinado, una función auxiliar del algoritmo `remallado fino`. Este refinado se lleva a cabo mediante una homotecia con razón 1/2, de manera que el segmento más largo se ubica en el punto medio del segmento original, mientras que los segmentos más cortos se posicionan en los extremos.

Algoritmo 10 refinado

Entrada: Punto1, Punto2, numeroDivisiones

Salida: P1salida, P2salida

1: Calcular el punto medio

$$\frac{\text{Punto1} + \text{Punto2}}{2} \tag{4.12}$$

2: Calcular el valor del incremento inicial

$$\Delta X = \frac{\text{Punto1} - centro}{2} \tag{4.13}$$

3: Calcular el primer punto `P1aux`

$$P1aux = centro - \Delta X \tag{4.14}$$

4: **for** 2 a numero de divisiones **do**

5: Asignar a `P2aux` el valor del `P1aux` anterior

6: Calcular el nuevo incremento como la mitad del interior

$$\Delta X = \frac{\Delta X}{2} \tag{4.15}$$

7: Calcular el siguiente `P1aux`

$$P1aux = P2aux - \Delta X \tag{4.16}$$

8: **end for**

9: Crear la segunda mitad del segmento mediante simetría a través del punto central.

Las variables de entrada de este algoritmo son:

- Punto1 y Punto2 los extremos del segmento a refinar mediante la homotecia.
- numeroDivisiones es un parámetro para definir cuantos subdivisiones existirán desde el punto medio del segmento hasta uno de sus extremos.

Las variables de salida son:

- P1salida y P2salida la nueva matriz de inicio y finales creados a partir de los extremos del segmento.

4.2.1.4. Rutina de mallado en electrodos con un tamaño inferior a uno dado

Esta rutina está diseñada para gestionar electrodos con variaciones significativas en sus dimensiones, asegurando una distribución adecuada de segmentos en el mallado. Para evitar que el número de segmentos, representado por el parámetro M, sea insuficiente en electrodos de menor tamaño, se ajusta la longitud de los mismos, estableciendo un tamaño máximo permitido. De esta manera, se optimiza la discretización del dominio y se mejora la calidad del proceso de calculo.

Algoritmo 11 remalladoLongitud

Entrada: P1, P2, r

Salida: P1, P2, r, maxLenght

1: **for all** conductores **do**
2: Calcular longitud conductor
3: **if** Longitud > maxLenght **then**
4: Calcular vector director unitario:

$$\text{vd} = \frac{P1 - P2}{\|P1 - P2\|} \tag{4.17}$$

5: **for** j = 1 a floor($Longitud/maxLenght$) **do**
6: Añadir al final el conductor con:

$$\begin{aligned} P1 &= (j-1) * \text{vd} * maxLenght + P1 \\ P2 &= j * \text{vd} * maxLenght + P1 \\ r &= r \end{aligned} \tag{4.18}$$

7: **end for**
8: Añadir el ultimo trozo del conductor para completarlo.
9: **end if**
10: **end for**

La nueva entrada:

- maxLenght es la longitud máxima de los electrodos.

4.2.2. Cálculo de la frontera

La rutina de cálculo de la frontera (`frontera`) constituye uno de los pilares esenciales de este trabajo, ya que permite determinar las condiciones de contorno, las cuales representan la mitad del problema a resolver.

Algoritmo 12 frontera

Entrada: P1, P2, dibujar

Salida: poligonal, indicesFrontera

1: **if** no se piden indices frontera **then**
2: Comprimir electrodos para normalizar los conductores mediante `fusionAlineados`.
3: Descomprimir los electrodos para tenerlos en el formato de electrodos cortos mediante `mallado`.
4: **end if**
5: Identificar los puntos únicos que definen el electrodo en la variable `points`:
- Eliminar las picas.
- Proyectar los puntos para trabajar sobre el plano.
- Construir matriz de conexión que describe la relación entre los puntos.

6: Generar la matriz `edge`, donde cada fila contiene los índices de los puntos conectados al punto (i).
7: Calcular el centroide del electrodo como el promedio de las coordenadas de los puntos únicos.
8: Determinar el `puntoBaseInicial` como la esquina superior derecha del electrodo para garantizar un punto de partida consistente.
9: Determinar el segundo `puntoBase` como el punto más alejado del centroide que conecta con el `puntoBaseInicial`.
10: Ejecutar la rutina `computoFrontera` con los puntos base en este orden.
11: **if** `falloFrontera` **then**
12: Ejecutar la rutina `computoFrontera` invirtiendo el orden de los puntos base.
13: **end if**
14: Generar la variable `poligonal` a partir de `indicesFrontera`.
15: Retornar la frontera comprimida, optimizada para el formato requerido por las siguientes rutinas.

Las variables de entrada de este algoritmo son:

- P1 y P2 las matrices que definen el electrodo.
- dibujar es un booleano que si tiene el valor de 1 dibuja los resultados obtenidos.

Las variables de salida son:

- poligonal es un conjunto de puntos que constituye la frontera del electrodo. Aunque no forma una región cerrada por sí misma, bastaría con añadir el primer punto al final para cerrarla.
- indicesFrontera es una variable usada en la rutina `remalladofino` para determinar que puntos de P1 y P2 constituyen la frontera.

La siguiente rutina es una rutina auxiliar empleada en `frontera`.

Algoritmo 13 computoFrontera

Entrada: frontier, point, edge, conexionesDegeneradas

Salida: falloFrontera, frontier

1: **while** true **do**
2: Calcular `segmentoBase`

$$segmentoBase = puntoBase - puntoBaseAnterior \tag{4.19}$$

3: Asignar `anguloBase` inicial

$$anguloBase = 2\pi \tag{4.20}$$

4: **for all** nodos conectados al `puntoBase` excepto el `puntoBaseAnterior` **do**
5: Calcular el `anguloBase` entre el `nodo` y el `segmentoBase`

$$\begin{aligned} &\texttt{angulo1} = \text{atan2}\,(\texttt{nodo} - \texttt{puntoBase}) \\ &\texttt{angulo2} = \text{atan2}\,(\texttt{puntoBaseAnterior} - \texttt{puntoBase}) \\ &\texttt{anguloBase} = \texttt{angulo1} - \texttt{angulo2} \end{aligned} \tag{4.21}$$

6: Elegir como siguiente punto de la frontera y por tanto nuevo `puntoBase` aquel que forme un ángulo menor.
7: **end for**
8: **if** superado numero de iteraciones **then**
9: Fallo en la convergencia. Salir de la rutina.

$$falloFrontera = 1 \tag{4.22}$$

10: **end if**
11: **if** puntoBase == puntoInicialFrontera **then**
12: Calcular orientación
13: **if** orientación == horario **then**
14: Frontera mal calculada

$$falloFrontera = 1 \tag{4.23}$$

15: **end if**
16: Salir de la rutina.
17: **end if**
18: **end while**

Las variables de entrada de este algoritmo son:

- frontier son los puntos iniciales o base necesarios para el calculo de la frontera.
- point es una matriz que contiene todos los puntos únicos que forman el electrodo.
- edge es una estructura tipo `cell` en la que cada fila (i) contiene los puntos con los que se conecta el punto (i).
- conexionesDegeneradas es un parámetro que controla la existencia de conexiones a través de un único punto. Aunque este tipo de conexiones no se presenta en la optimización de puestas a tierra en subestaciones, se incluye este parámetro para dotar a la rutina de mayor flexibilidad.

Las variables de salida son:

- falloFrontera es un booleano que indica fallo en uno de los siguientes supuestos:
  - La poligonal está orientada en sentido horario, lo cual supone un error, ya que en el cálculo se asume que la frontera debe estar orientada en sentido antihorario. Si el resultado es horario, esto indica que los puntos de partida se han elegido en el orden incorrecto.
  - Se ha superado el número de puntos del electrodo teniendo en cuenta posibles conexiones degeneradas.
- frontier es la poligonal de abierta de la frontera.

4.2.3. Subdivisión de la figura en convexos

La subdivisión en convexos, descrita en los fundamentos, se implementa mediante la rutina `descomposicionConvexa`. No se detalla a continuación, ya que simplemente invoca la rutina `notchCalculator`, utilizada para obtener los vértices no convexos (notches), seguida de la rutina `subdividir`, que realiza la división en convexos a partir de los notches.

4.2.3.1. Cálculo de puntos de no convexidad

Algoritmo 14 notchCalculator

Entrada: poli

Salida: notches_, amount, position

1: **for all** vertices en la frontera **do**
2: Calcular los vectores directores de los 2 segmentos conectados al vértice:

$$\begin{aligned} v_1 &= \text{Vértice}_{\text{posterior}} - \text{Vértice}_{\text{actual}} \\ v_2 &= \text{Vértice}_{\text{anterior}} - \text{Vértice}_{\text{actual}} \end{aligned} \tag{4.24}$$

3: Se computa los valores de los senos y cosenos de la matriz de rotación:

$$\begin{aligned} \cos(\theta) &= \frac{v_{1x}}{\|v_1\|} \\ \sin(\theta) &= -\frac{v_{1y}}{\|v_1\|} \end{aligned} \tag{4.25}$$

4: Se computa la matriz de rotación:

$$R = \begin{pmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{pmatrix} \tag{4.26}$$

5: Se rota el segundo vector director:

$$v_2' = R \cdot v_2 \tag{4.27}$$

6: **if** $v_{2y}' < 0$ **then**
7: Existe notch, añadir a la lista este vértice.
8: Aumentar la cantidad de notches en 1.
9: **end if**
10: **end for**

La variable de entrada de este algoritmo es la poligonal que define la frontera. Las variables de salida son:

- notches_ es una matriz que contiene las coordenadas de los notches.
- amount es la cantidad de notches que hay en la frontera.
- position es la posición en la matriz de coordenadas de los notches, facilita el trabajo con la propia poligonal de la frontera.

4.2.3.2. Subdivisión de los polígonos

Algoritmo 15 subdividir

Entrada: poli, notches_, position

Salida: poligonales

1: Calculo de los vectores directores en el punto de concavidad con dirección el vértice:
▷ Solo se trata un punto de no concavidad cada iteración. Por tanto solo se trabaja con el primer valor en position.

$$\begin{aligned} v_1 &= \text{Vértice}_{\text{position}} - \text{Vértice}_{\text{position+1}} \\ v_2 &= \text{Vértice}_{\text{position}} - \text{Vértice}_{\text{position-1}} \end{aligned} \tag{4.28}$$

2: Calcular el vector bisector de los vectores directores:

$$v = \|v_1\| \cdot v_2 + \|v_2\| \cdot v_1 \tag{4.29}$$

3: Expresar el vector director como una recta. ▷ Visto en anteriores algoritmos.
4: Intersecar esta recta con el resto de las aristas de la poligonal de la frontera para obtener las posibles intersecciones donde crear un nuevo vértice. ▷ Visto en anteriores algoritmos.
5: Crear un nuevo punto en la intersección más cercana al vértice con el que se esta trabajando. ▷ Visto en anteriores algoritmos.
6: Devolver las poligonales siguientes:
- Poligonal_1: Conjunto de vértices a un lado de la bisectriz.
- Poligonal_2: Conjunto de vértices al otro lado de la bisectriz.

Las variables de entrada de este algoritmo son las de salida de la rutina `notchCalculator`. La variable de salida es:

- poligonales es una estructura que contiene cada una de las fronteras abiertas de las subregiones convexas de la malla de la subestación.

4.2.4. Cálculo de los coeficientes de las inecuaciones

La rutina `coeff` implementa el algoritmo para el cálculo de las inecuaciones necesarias en el algoritmo genético.

Algoritmo 16 coeff

Entrada: poligonal
Salida: coeficientes

1: Calcular la descomposición convexa de la poligonal mediante la función **descomposicionConvexa**
2: **for all** poligonales **do**
3: Calcular un punto auxiliar, D, para ajustar los signos de las inecuaciones como el punto medio de los puntos medios de 2 segmentos en la frontera.
4: **for all** segmentos en la poligonal **do**
5: Calcular las matrices de coeficientes, A, B, que definen la recta que define dicho segmento. ▷ Visto en anteriores algoritmos.
6: **if** $A \cdot D > B$ **then** ▷ La desigualdad no esta en su formato canónico
7: Cambiar el signo las matrices de coeficientes A, B.
8: **end if**
9: **end for**
10: **end for**

La variable de entrada de este algoritmo es la salida de la rutina `subdividir`. La variable de salida es:

- coeficientes es una estructura que contiene las matrices A y , las cuales definen las inecuaciones correspondientes a cada una de las regiones a estudiar.

4.2.5. Integración de las rutinas

La siguiente rutina integra las funciones descritas en las secciones 4.2.2, 4.2.3 y 4.2.4. Teniendo esto en cuenta, basta con invocar esta rutina para realizar todo el cálculo del procesamiento geométrico.

Algoritmo 17 condicionesContorno

Entrada: P1, P2, dibujar
Salida: coeficientes

1: Llamar a la función `frontera` con parámetros `P1`, `P2` y `dibujar = 0`, para calcular la poligonal. `poligonal` ← `frontera`(P1, P2, 0)
2: Llamar a la función `coeff` con `poligonal` para obtener los coeficientes. `coeficientes` ← `coeff`(poligonal)
3: **return** `coeficientes`

Las variables de entrada de este algoritmo son las mismas que las de la rutina `frontera` mientras que la salida es la misma que en la rutina `coeff`.

4.3. Implementación de las rutinas de optimización

Para implementar adecuadamente los algoritmos se ha leído la siguiente documentación [23][24][25][26][27][28][29].

4.3.1. Función preparada para la optimización

Para la optimización como se requieren dos funciones distintas. Una para obtener el máximo y otra para el mínimo. Se crea una función base a partir de las cuales se crean funciones que la adaptan según si se busca el máximo o el mínimo y según se requiera o no el gradiente (se utiliza en el algoritmo determinista).

Para ello se emplea la rutina base `PotencialCSM_opt`.

Algoritmo 18 PotencialCSM_opt

Entrada: pot, grad

Salida: punto, a

1: Descomponer la variable `a` en sus componentes para permitir el calculo del potencial y gradiente.
   - **P1** ← **rows**(a,1:3)
   - **P2** ← **rows**(a,4:6)
   - **dens** ← **rows**(a,7:16) ▷ Asumiendo 10 segmentos

2: pot ← **PotencialCSM**(punto,P1,P2,dens)
3: grad ← **GradienteCSM**(punto,P1,P2,dens)

Las variables de entrada de esta rutina son:

- punto es el punto donde se calcula el potencial.
- a es una matriz que contiene el resto de datos necesarios para el calculo del potencial.

Las variables de salida son:

- pot es el valor del potencial en el punto.
- grad es el valor del gradiente en el punto.

Para minimizar, se emplea la rutina `negPotencial_opt`, que se encarga de negar el valor del potencial y el gradiente, ya que maximizar la función $f(x)$ equivale a minimizar $-f(x)$. Por último, como el gradiente solo espera una única salida, se ajusta mediante las rutinas `PotencialCSM_opt_GA` y `negPotencial_opt_GA` para garantizar dicho comportamiento. Debido a su simplicidad, dichas rutinas no se desarrollan a continuación.

4.3.2. Función fmincon

Se utiliza esta función definida en MATLAB con los siguientes parámetros (se omiten parámetros que no aplican al problema):

Parámetros	**Descripción o valor del parámetro**
Función a optimizar (`fun`)	PotencialCSM convertido en una función anónima con una única entrada, la otra entrada es el parámetro `a` descrito anteriormente.
Gradiente especificado (`SpecifyObjectiveGradient`)	GradienteCSM convertido en una función anónima con una única entrada, la otra entrada es el parámetro `a` descrito anteriormente.
Punto inicial (x_0)	Obtenido como el punto donde se encuentra mínimo/máximo entre los óptimos obtenidos por el algoritmo genético.
Restricciones de desigualdad lineales (A `y` b) $A \cdot x <= b$	Los coeficientes de las desigualdades A y b correspondientes con la región donde se encuentra el punto mínimo/máximo entre los óptimos obtenidos por el algoritmo genético.
Algoritmo (`Algorithm`)	Punto interior.
Comprobación gradientes (`CheckGradients`)	Se hacen pruebas para ver que el gradiente calculado mediante diferencias finitas y el analítico son similares. Con una cota superior del error de 10^{-5}.
Tolerancia restricciones (`ConstraintTolerance`)	Se utiliza el valor por defecto 10^{-6}.
Número máximo de evaluaciones de la función (`MaxFunctionEvaluations`)	Se utiliza el valor por defecto de 3000 evaluaciones de la función.
Número máximo de iteraciones (`MaxIterations`)	Se utiliza el valor por defecto de 1000 iteraciones.
Tolerancia en las condiciones de primer orden (`OptimalityTolerance`)	Se utiliza el valor por defecto 10^{-6}.
Mínimo paso permitido (`StepTolerance`)	Se utiliza el valor por defecto 10^{-10}.
Actualización del parámetro de barrera (`BarrierParamUpdate`)	Se utiliza el actualizador monótono donde el parámetro: $$\mu^{(i+1)} = \sigma\mu^{(i)} \quad (4.30)$$ ▪ Si solo se necesitan 1 o 2 iteraciones para obtener precisión suficiente: $\sigma = \frac{1}{100}$. ▪ En cualquier otro caso $\sigma = \frac{1}{5}$.
Valor inicial del parámetro de barrera (`InitBarrierParam`)	Se utiliza el valor por defecto 0,1.
Algoritmo sin restricciones para cuando se simplifican las condiciones KKT (`SubproblemAlgorithm`)	Se utiliza el algoritmo de gradiente conjugado. Como el calculo de la matriz Hessiana es caro este algoritmo es más eficiente.

Tabla 4.1: Tabla con los parámetros empleados por el algoritmo determinista.

4.3.3. Función ga

Se utiliza esta función definida en MATLAB con los siguientes parámetros (se omiten parámetros que no aplican al problema):

Parámetros	**Descripción o valor del parámetro**
Función a optimizar (`fun`)	PotencialCSM convertido en una función anónima con una única entrada, la otra entrada es el parámetro `a` descrito anteriormente.
Número de variables (`nvars`)	Se utilizan 2 variables: x,y.
Restricciones de desigualdad lineales (A `y` b) $A \cdot x <= b$	Los coeficientes de las desigualdades A y b correspondientes con la región donde se lanza el algoritmo genético. Existen tantas regiones como puntos de no concavidad.
Tolerancia restricciones (`ConstraintTolerance`)	Se utiliza el valor por defecto 10^{-3}.
Función de creación de la población inicial (`CreationFcn`)	Se crea una población inicial aleatoria que satisface las restricciones. Muchos individuos se sitúan en la frontera y se crea una población con buena dispersión.
Función de cruce (`CrossoverFcn`)	Se crea al hijo como una media ponderada de los padres.
Fracción de población que crea la función de cruce (`CrossoverFraction`)	Se utiliza el valor por defecto 0,8.
Individuos con supervivencia garantizada (`EliteCount`)	Se utiliza el valor por defecto 3.
Función para rescalar la aptitud de los individuos (`FitnessScalingFcn`)	Se usa para reducir la dispersión de los valores de aptitud de los distintos individuos. Para rescalarlos un individuo de rango r (su posición si se ordenan los individuos en función de su aptitud) tiene su aptitud rescalada en $1/\sqrt{r}$.
Tolerancia en el cambio de la función objetivo para detener las iteraciones (`FunctionTolerance`)	Se utiliza el valor por defecto 10^{-6}.
Número máximo de generaciones (`MaxGenerations`)	Se utiliza el valor por defecto 200.
Número máximo de generaciones sin un cambio en la aptitud del mejor individuo (`MaxStallGenerations`)	Se utiliza el valor por defecto 50.
Función que determina la mutación de los hijos (`MutationFcn`)	Se muta en la dirección de anteriores individuos exitosos. De esta manera, no se puede violar las restricciones.
Tamaño de la población (`PopulationSize`)	Se utiliza el valor por defecto 50.
Función que selecciona a los padres para el cruce (`SelectionFcn`)	Se crea una línea en la que cada individuo ocupa una posición proporcional a su aptitud. Se define un paso como: $$paso = 1/\text{Número de padres a seleccionar} \quad (4.31)$$ El primer punto de selección se elige de manera aleatoria dentro del intervalo $[0, paso]$. El resto de los individuos se seleccionan avanzando en la línea con incrementos iguales al paso definido.

Tabla 4.2: Tabla con los parámetros empleados por el algoritmo genético.

4.3.4. Integración de las rutinas

Para obtener el mínimo y el máximo de un electrodo de puesta a tierra se emplea una función que integra gran parte de los métodos de procesamiento geométrico junto con los métodos de optimización para obtener los puntos donde se obtiene el potencial máximo y mínimo.

Para ello se emplea la rutina `PotencialOptimo`.

Algoritmo 19 PotencialOptimo

Entrada: xmin, fmin, xmax, fmax

Salida: P1, P2, dens, poligonal

1: Calcular los coeficientes [`A,B`] que conforman las distintas regiones mediante **coeff**(poligonal)
2: Crear las funciones anónimas para los problemas de minimización, maximización, función fmincon y algoritmo genético junto con las opciones de estas últimas.
3: **for all** regiones convexas **do**
4: [`x,fval`] ← **ga**(`minimizar`,`A,B`)
5: Añadir `x` y `fval` a los vectores `puntoResultado` y `potencialOptimo` respectivamente.
6: **end for**
7: Calcular el mínimo en `potencialOptimo` para obtener `puntoOptimizar`.
8: [`xmin,fmin`] ← **fmincon**(`minimizar`,`puntoOptimizar`,`A,B`)
9: **for all** regiones convexas **do**
10: [`x,fval`] ← **ga**(`maximizar`,`A,B`)
11: Añadir `x` y `fval` a los vectores `puntoResultado` y `potencialOptimo` respectivamente.
12: **end for**
13: Calcular el máximo en `potencialOptimo` para obtener `puntoOptimizar`.
14: [`xmax,fmax`] ← **fmincon**(`maximizar`,`puntoOptimizar`,`A,B`)

Las variables de entrada de esta rutina son:

- P1, P2, dens los parámetros obtenidos mediante el cálculo descritos en secciones anteriores.
- poligonal la región donde se realizará el estudio de máximos y mínimos. Para subestaciones normalmente se usara la poligonal que delimita la frontera, pero se permite cualquier otra poligonal en sentido antihorario.

Las variables de salida son:

- xmin y xmax los puntos $[x, y]$ donde se producen el mínimo y máximo respectivamente.
- fmin y fmax los valores de la función objetivo correspondientes con el mínimo y el máximo respectivamente.

4.4. Implementación de la interfaz gráfica

La interfaz gráfica se implementa en MATLAB mediante el uso de scripts con una clase GUI por cada interfaz o figura. A continuación se presentan varias secciones. En la primera sección se describe la implementación de clases y en la segunda sección se presenta el diagrama funcional de la interfaz.

4.4.1. Diagramas UML de clases.

En esta sección se presentan los diagramas UML de las dos colecciones de clases que se han diseñado para llevar a cabo la implementación de la interfaz gráfica.

La clase GUI actúa como la clase principal encargada de construir cada una de las ventanas de la interfaz gráfica. Esto significa que, cada vez que se desee cambiar de ventana, será necesario crear una nueva instancia de esta clase. Para añadir elementos a una ventana, se utiliza el método `add()`, junto con el constructor del elemento que se desea incorporar.

No se han integrado las imágenes como elementos accesibles directamente desde la clase GUI, ya que solo se emplean en la ventana inicial. Además, dado que se partió de un código implementado íntegramente con un enfoque funcional, y con el fin de evitar reescribir toda la aplicación, se optó por una implementación parcial.

Cabe destacar también la relevancia de los métodos `addAuxData()` y `getAuxData()`, que permiten almacenar datos auxiliares bajo un nombre específico, con el objetivo de reutilizarlos en otras ventanas de la interfaz. Esto resulta útil, por ejemplo, para conservar la opción seleccionada en un menú desplegable o recuperar un texto introducido previamente.

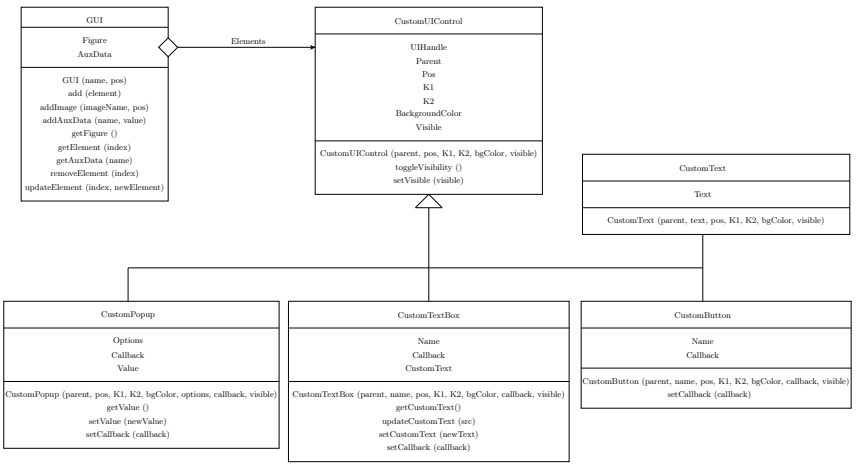


Figura 4.4: Diagrama UML de la implementación de la clase GUI. Esta clase está compuesta por distintos elementos: botones, menús desplegables (popups) y campos de texto. Todos ellos encapsulados en sus respectivas clases derivadas. El acceso a estos elementos se realiza a través de la clase base CustomUIControl, que proporciona los datos necesarios para su creación y manipulación. Aunque no es la solución más óptima, la clase GUI incluye métodos para añadir imágenes y almacenar datos auxiliares, lo cual facilita la gestión y el posicionamiento preciso de los controles.

La clase ElectrodeCollection constituye el componente principal para el almacenamiento de toda la información relacionada con los electrodos. Aunque en esta aplicación concreta presenta cierta incompletitud, debido a que aún conserva vestigios del diseño funcional original, se trata de una clase extremadamente versátil. Su diseño permite añadir fácilmente todos los métodos que requieran acceso a la información contenida en los vectores P1, P2 y r. A pesar de estas limitaciones, esta estructura ofrece una notable agilidad en la creación y eliminación dinámica de electrodos, algo que sería más complejo de implementar con otro enfoque de diseño.

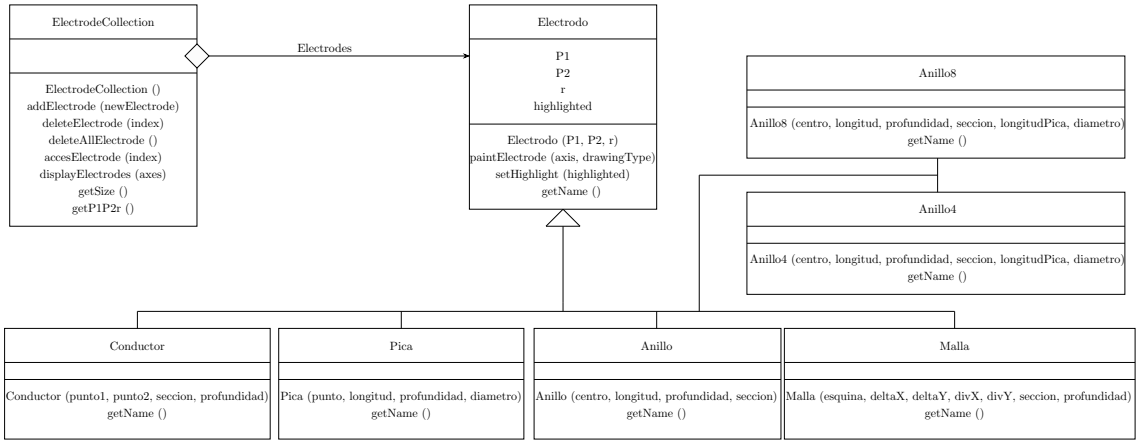
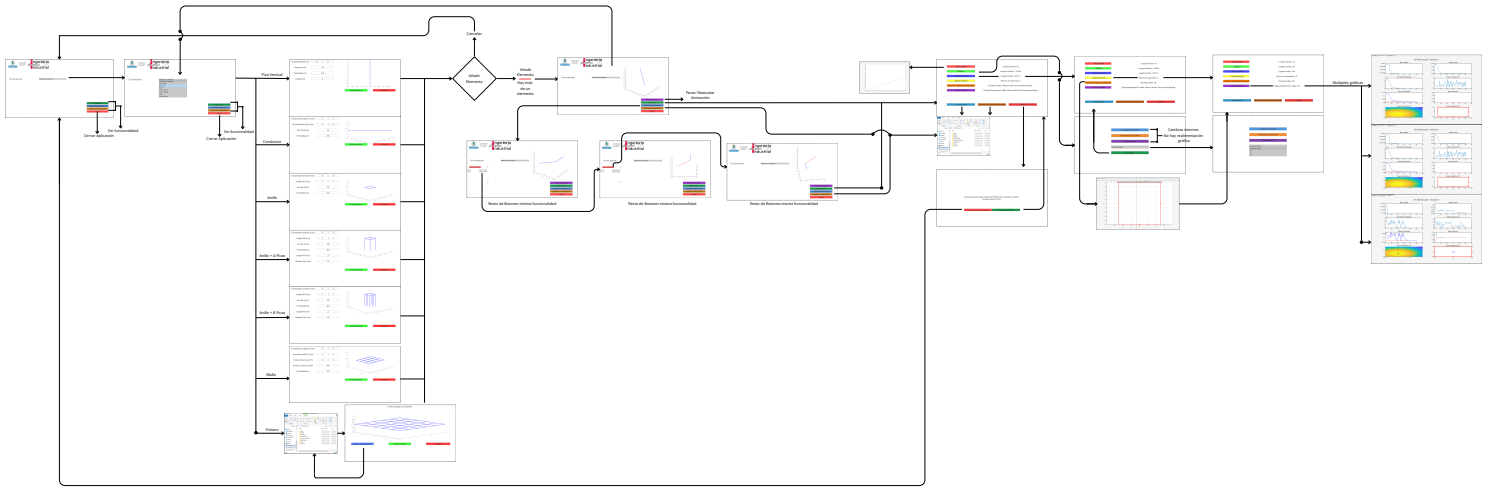


Figura 4.5: Diagrama UML de la implementación de los electrodos. Los distintos electrodos estándar se modelan mediante clases derivadas de Electrodo, cada una con su propio constructor. La API incluye métodos para gestionar un vector de electrodos, así como operaciones para construirlos, dibujarlos en 3D (resaltándolos y coloreándolos de rojo) y obtener su nombre para su visualización en la interfaz.

4.4.2. Diagrama Funcional de la interfaz

Figura 4.6: Diagrama funcional de la interfaz gráfica, dividido en dos pantallas principales: dibujo de electrodos y preprocesado/cálculo. En la primera pantalla el usuario puede añadir electrodos estándar o importarlos desde un archivo (`.csv`, `.mat`), visualizar una animación del conjunto, eliminarlos o exportarlos, y avanzar al preprocesado (actualmente no está disponible regresar desde esta pantalla). En la segunda pantalla se genera y se puede procesar y visualizar el mallado, calcular la densidad de corriente y la región de estudio (de forma automática o manual). Para optimizar la función potencial se deben haber completados los pasos previos y, por último, se exportan los resultados.



Capítulo 5

Resultados y discusión

En este capítulo se muestran los resultados obtenidos de aplicar las rutinas desarrolladas con anterioridad.

5.1. Resultados

5.1.1. Calculo geométrico

5.1.1.1. Estadisticas iniciales de cada malla.

Antes de comenzar con el trabajo específico sobre las distintas mallas, se presentan a modo de resumen sus características geométricas iniciales.

En total, se analizan nueve mallas. Las cuatro primeras se emplean exclusivamente para demostrar la robustez y el funcionamiento del procesamiento geométrico. Por otro lado, las cinco mallas restantes se utilizan tanto para el análisis de los métodos geométricos como para la evaluación de los métodos de optimización.

Nombre	**Longitud Mínima** $[m]$	**Longitud Máxima** $[m]$	**Longitud Media** $[m]$	**Número de Segmentos**	**Kr** $\left[\frac{\Omega}{\Omega m}\right]$
GeometriaIrregular.mat	0.1898	2.000	1.6745	100	0.1411
GeometriaIrregularSimple.mat	0.03530	0.6902	0.1427	132	0.6227
MetodosGeometricos.mat	2.500	10.00	5.163	732	$5.888 \cdot 10^{-3}$
RutinasMalladoTest.mat	0.3410	0.7571	0.6005	12	1.080
MallaSimpleRectangulo.mat	3.000	4.000	3.500	120	$2.539 \cdot 10^{-2}$
Planta fotovoltaica.mat	0.1080	134.2	9.330	558	$2.484 \cdot 10^{-3}$
Subestacion 1.mat	0.5130	68.54	11.13	166	$9.033 \cdot 10^{-3}$
Subestacion 2.mat	$3.443 \cdot 10^{-5}$	13.95	3.057	556	$1.411 \cdot 10^{-2}$
Subestacion 3.mat	2.000	38.00	25.12	50	$1.374 \cdot 10^{-2}$

Tabla 5.1: Estadísticas iniciales de longitud mínima, máxima y media de segmento, número de segmentos, y coeficiente de resistencia de las nueve mallas analizadas en el estudio.

5.1.1.2. Rutinas remallado

En esta sección se presentan los resultados de las rutinas de remallado aplicadas a todos los electrodos analizados, con el objetivo de ilustrar el funcionamiento general de dichas rutinas y evidenciar los cambios geométricos que estas inducen. La única malla no analizada es la de la estación fotovoltaica porque presenta una geometría tan dispersa que la visualización de los resultados no es apreciable.

Figura 5.1: Resultados de lanzar la rutina `remalladosTest.m` sobre la malla `GeometriaIrregular.mat`. Como se puede observar, dado que la malla está compuesta únicamente por un solo anillo y no contiene electrodos con una longitud superior a 1 metro, únicamente tiene efecto la rutina de remallado fino.

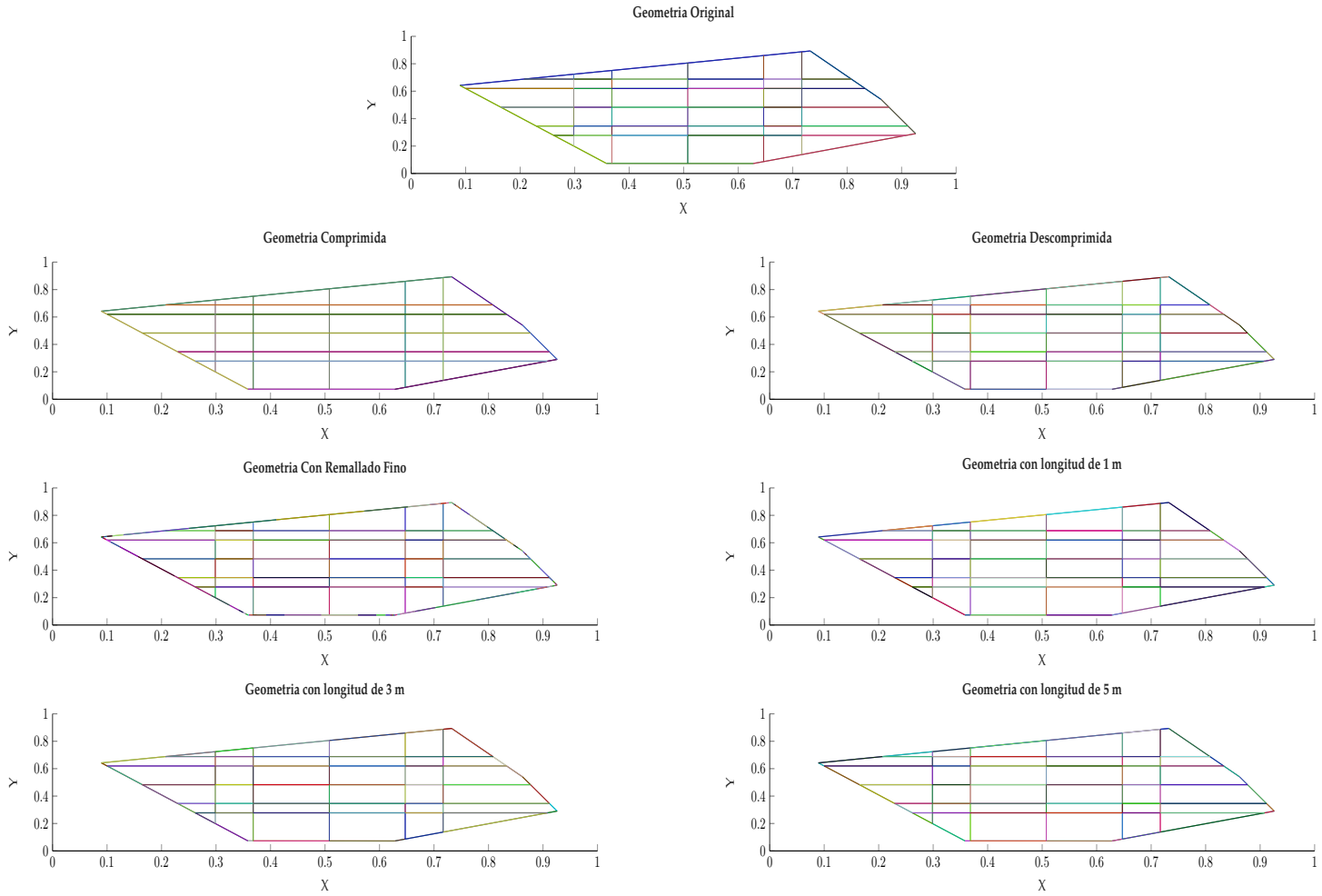


Figura 5.2: Resultados de lanzar la rutina `remalladosTest.m` sobre la malla `GeometriaIrregularSimple.mat`. En este caso, además, es posible apreciar la diferencia entre la geometría comprimida y la descomprimida de la malla. Mientras que la geometría comprimida no refleja las intersecciones, la descomprimida sí las representa.

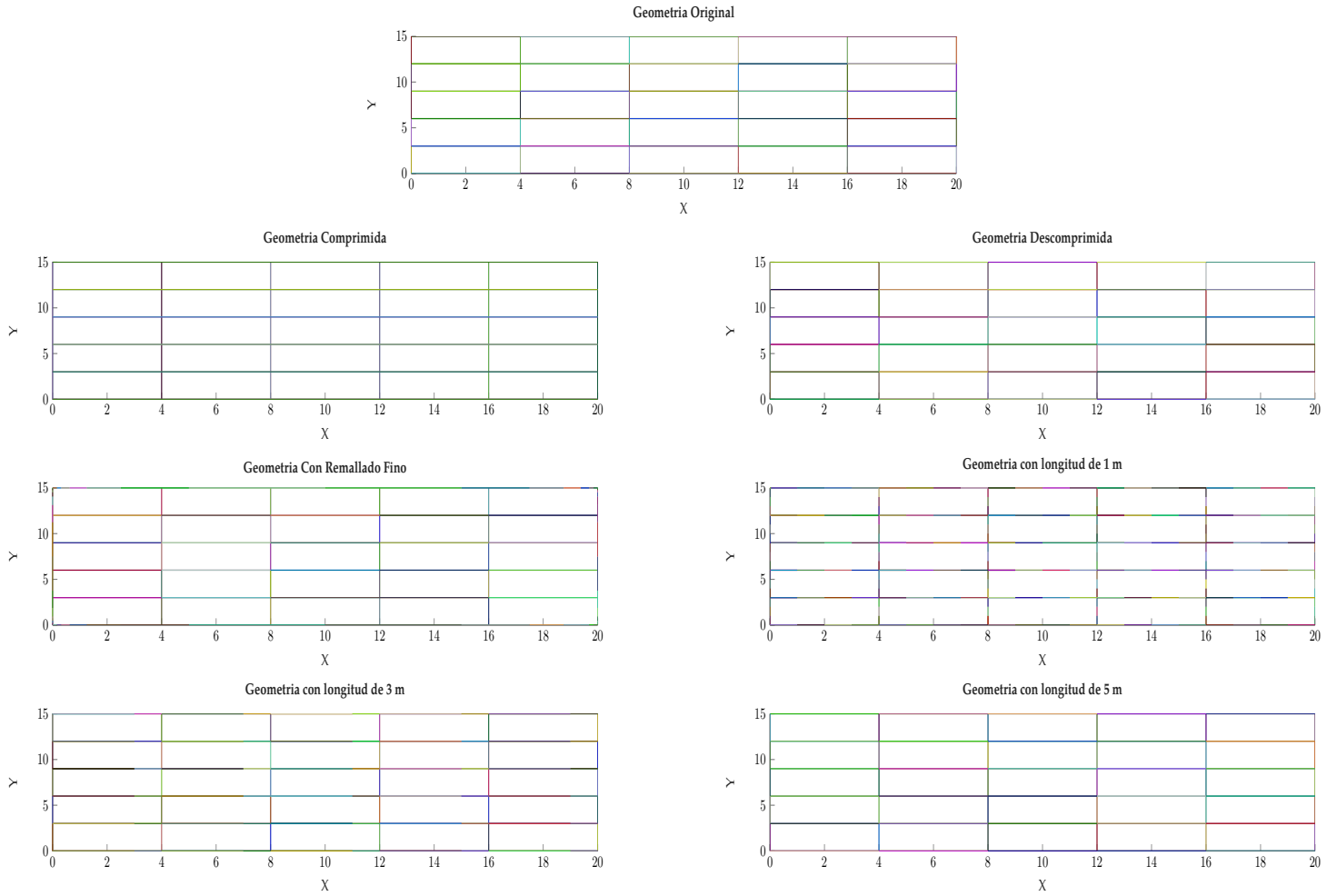


Figura 5.3: Resultados de lanzar la rutina `remalladosTest.m` sobre la malla `MallaSimpleRectangulo.mat`. Este ejemplo ilustra de manera óptima todos los tipos de mallado, ya que su regularidad permite distinguir claramente los diferentes cambios que realiza cada rutina.

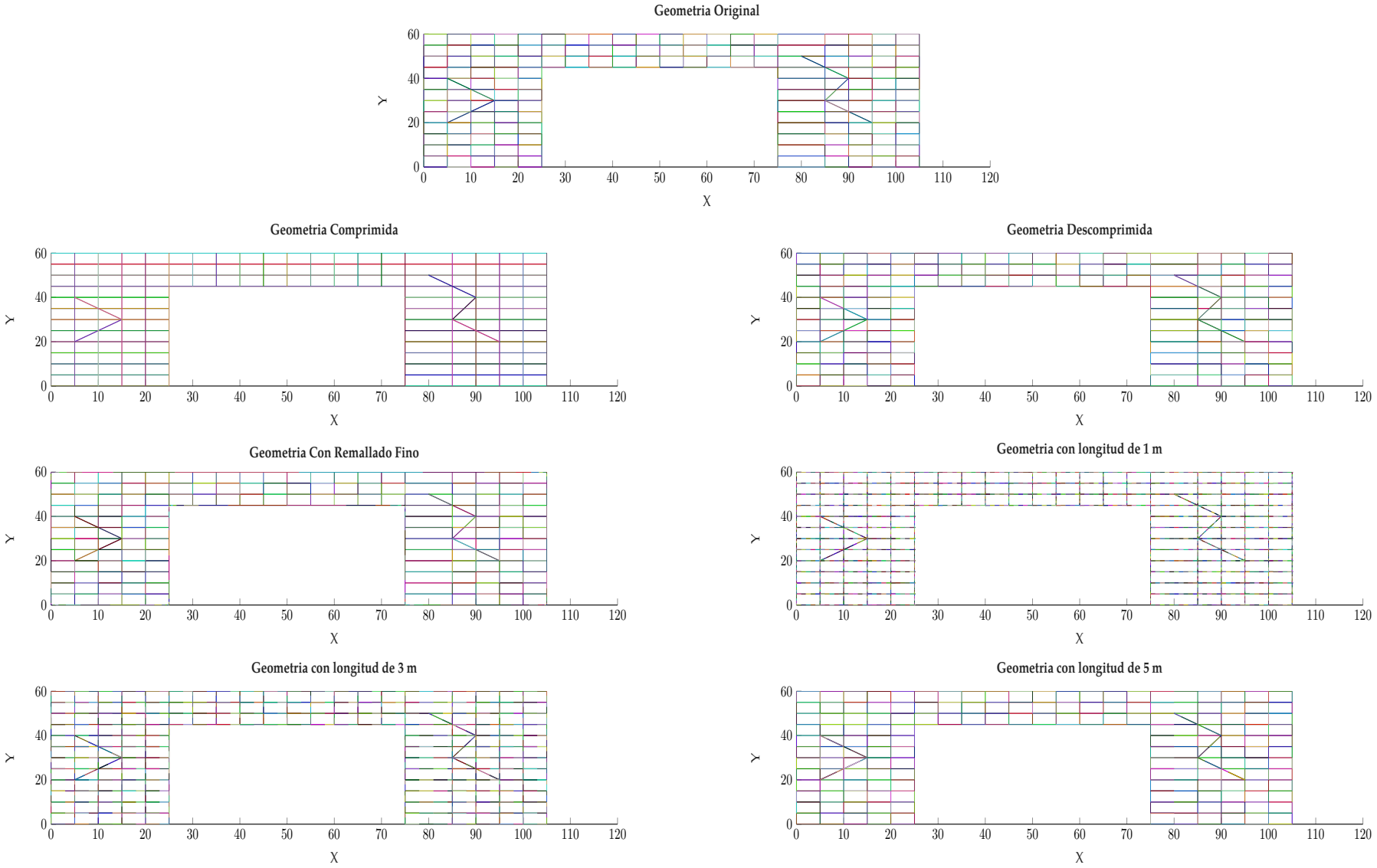


Figura 5.4: Resultados de lanzar la rutina `remalladosTest.m` sobre la malla `MetodosGeometricos.mat`. En este ejemplo se puede observar como las rutinas son estables incluso con geometrías con gran número de electrodos.

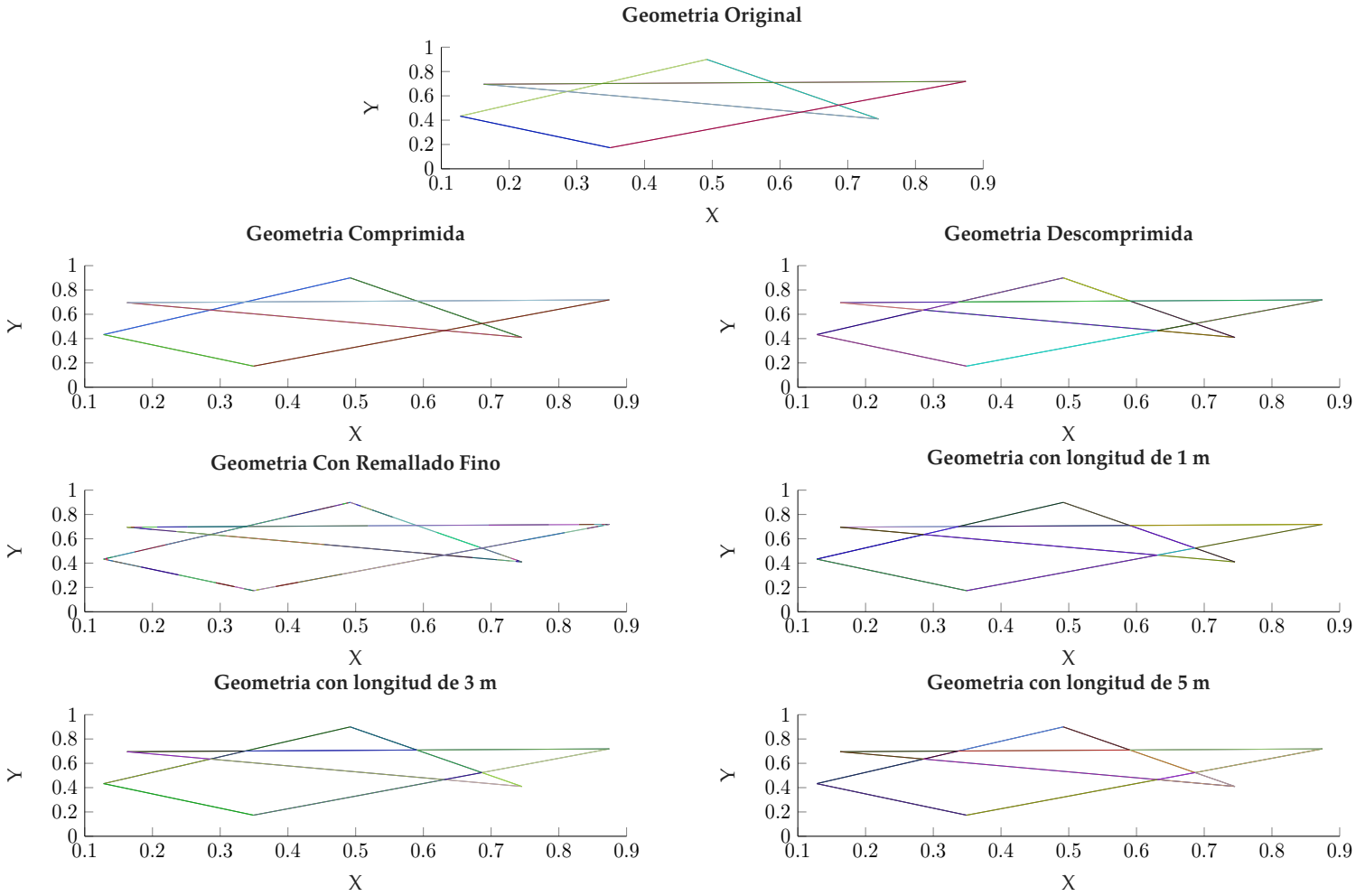


Figura 5.5: Resultados de lanzar la rutina `remalladosTest.m` sobre la malla `RutinasRemalladoTest.mat`. En este ejemplo se puede observar como las rutinas son estables incluso con geometrías irregulares.

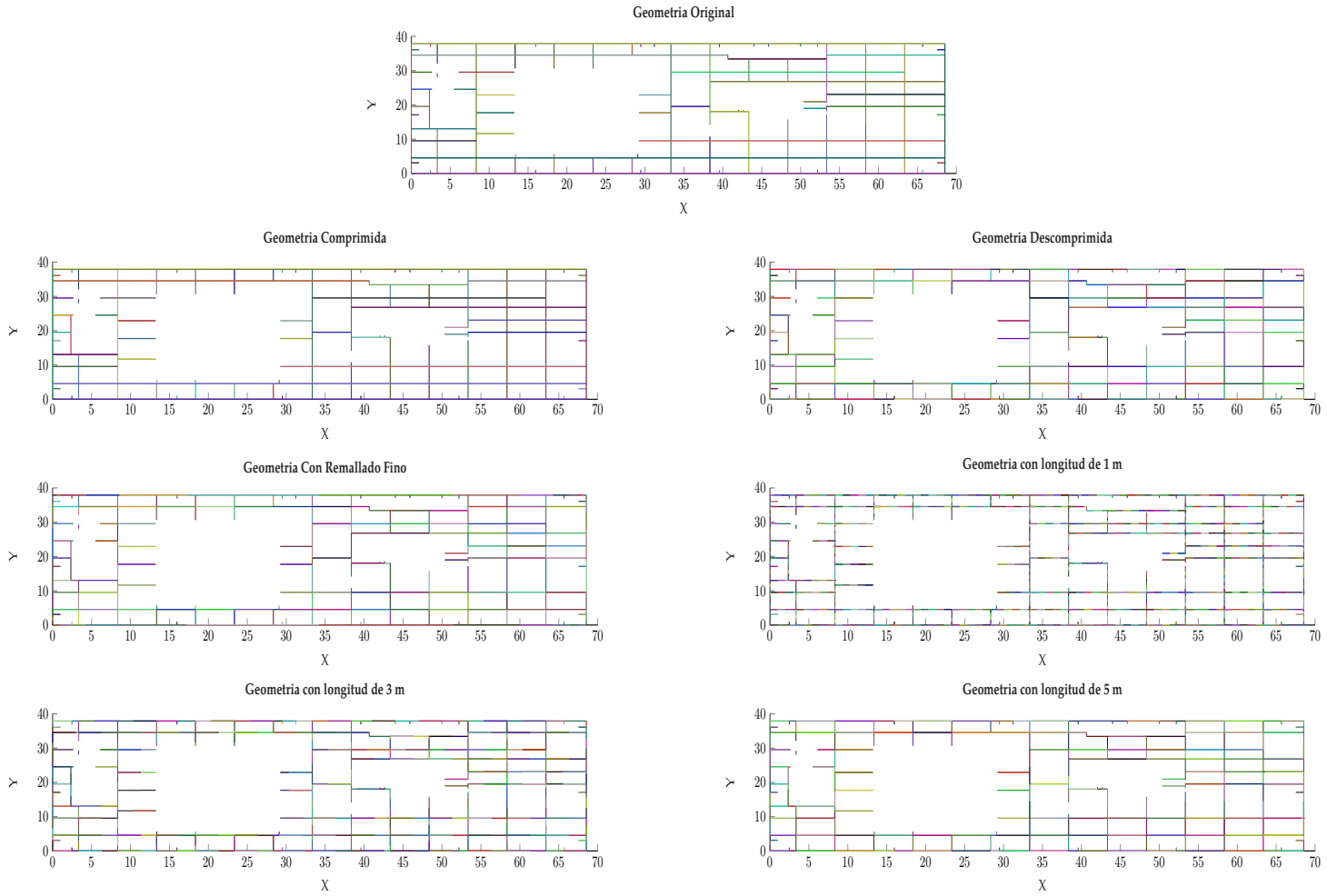


Figura 5.6: Resultados de lanzar la rutina `remalladosTest.m` sobre la malla `Subestacion 1.mat`. Se introduce como ejemplo porque será una de las geometrías a analizar en la optimización.

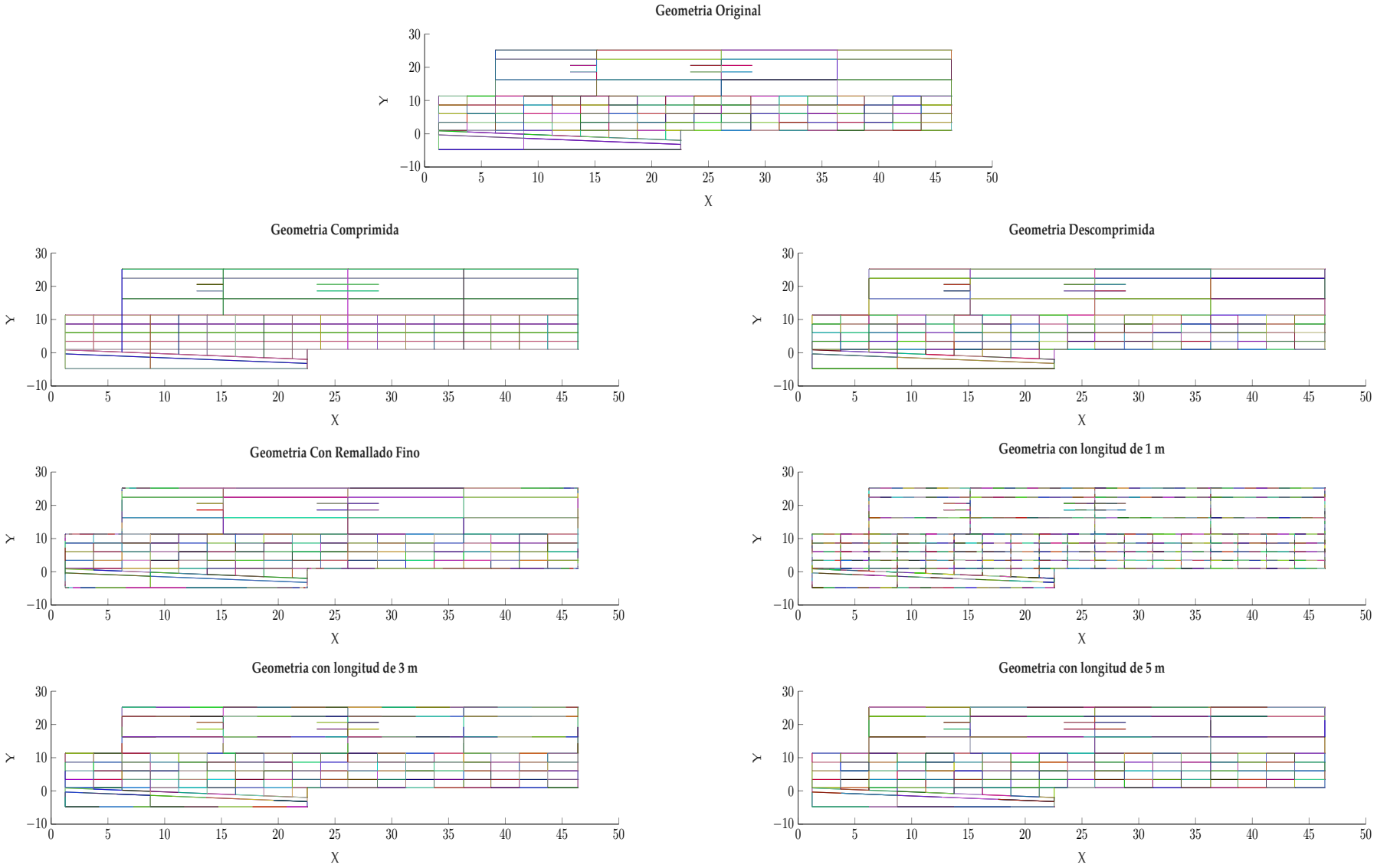


Figura 5.7: Resultados de lanzar la rutina `remalladosTest.m` sobre la malla `Subestacion 2.mat`. Se introduce como ejemplo porque será una de las geometrías a analizar en la optimización.

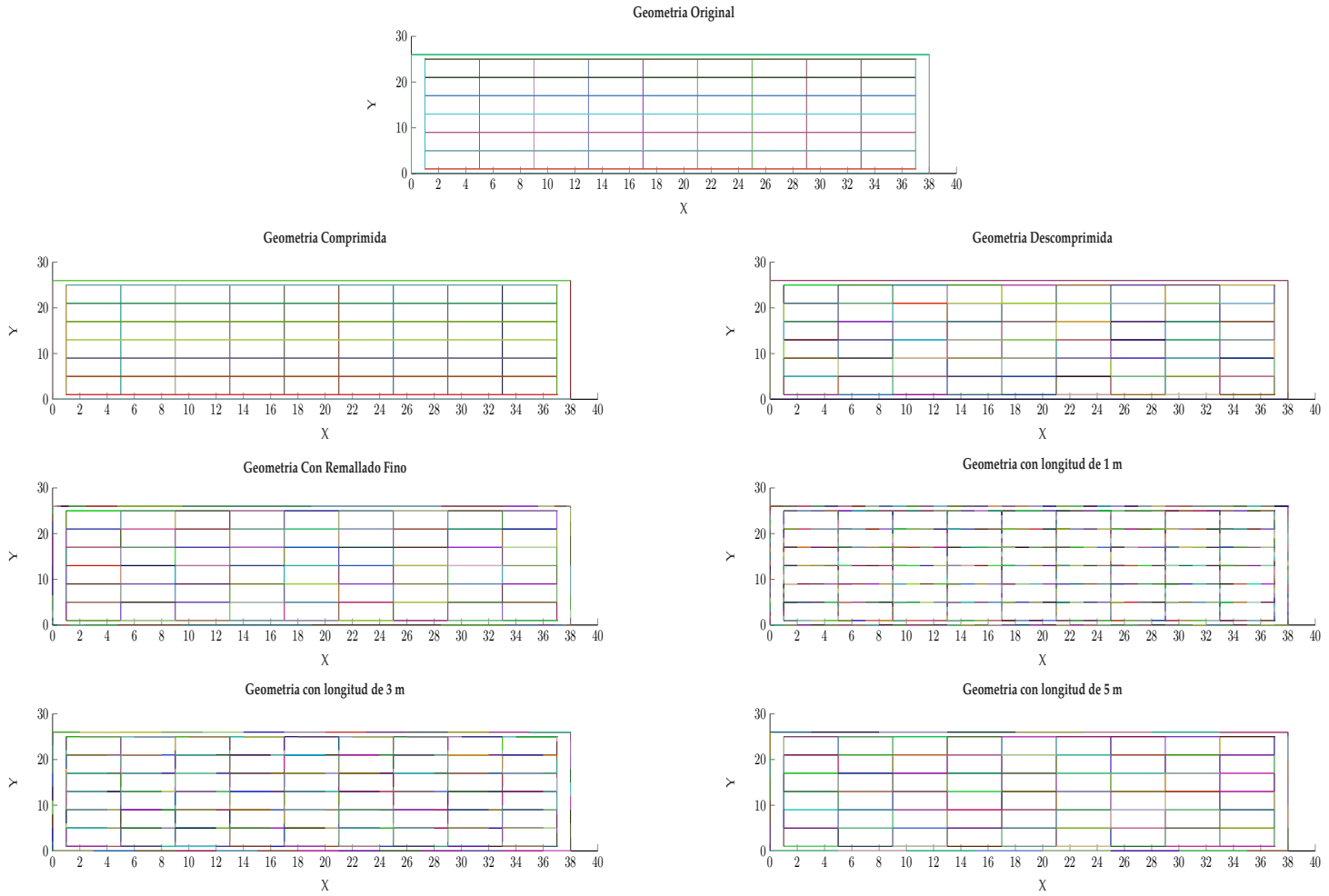


Figura 5.8: Resultados de lanzar la rutina `remalladosTest.m` sobre la malla `Subestacion 3.mat`. Se introduce como ejemplo porque será una de las geometrías a analizar en la optimización.

5.1.1.3. Calculo de la frontera

En esta sección se presentan los resultados de una rutina fundamental para el correcto funcionamiento del programa: el cálculo de la frontera. Como se observa en los ejemplos siguientes, esta rutina permite, a partir de la información del electrodo (representado en rojo en la gráfica superior), obtener la poligonal de la frontera (representada en azul en la gráfica inferior) mediante el procesamiento geométrico descrito en la sección 4.2.2.

Entre estos ejemplos cabe destacar el funcionamiento en las figuras 5.12, donde a pesar de la geometría irregular se consigue marcar correctamente el contorno, en 5.13, donde pese a la existencia de electrodos desconectados y a una geometría con "caminos sin retorno" se logra encontrar el rectángulo envolvente, y por último en la figura 5.15, donde aunque las dos partes de la malla no están conectadas se determina sin dificultad la frontera exterior.

En el caso del ejemplo `Planta fotovoltaica.mat` no se analiza la frontera, ya que, al tratarse de electrodos con una estructura en forma de "árbol", es extremadamente complicado definir la envolvente o frontera. En este caso, la región de estudio debe introducirse manualmente, ya sea mediante la interfaz gráfica o utilizando la rutina `poligonalGinput.m`.

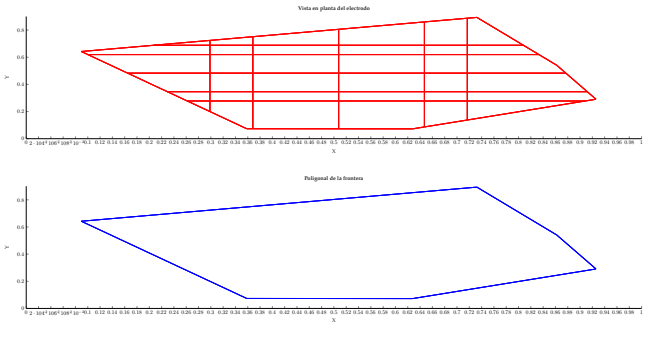


Figura 5.9: Resultados del calculo de la frontera de la malla `GeometriaIrregularSimple.mat`. En este ejemplo se observa cómo, empezando en la esquina superior derecha, el algoritmo permanece en el exterior mediante iteraciones sucesivas, obteniéndose así la región de estudio delimitada por la frontera.

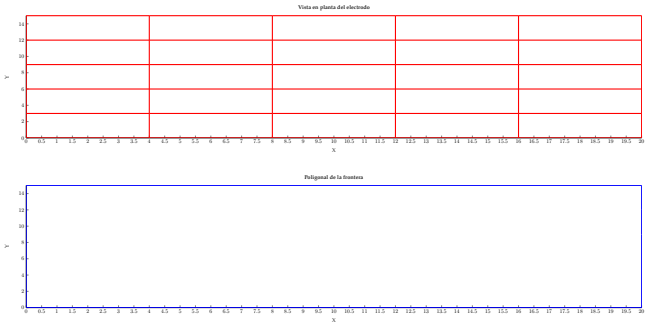


Figura 5.10: Resultados del calculo de la frontera de la malla `MallaSimpleRectangulo.mat`. Este ejemplo, aunque no tiene un gran valor desde el punto de vista geométrico, es especialmente relevante, ya que demuestra que el algoritmo funciona correctamente en electrodos regulares del tipo malla o anillo, que son relativamente comunes en su aplicación.

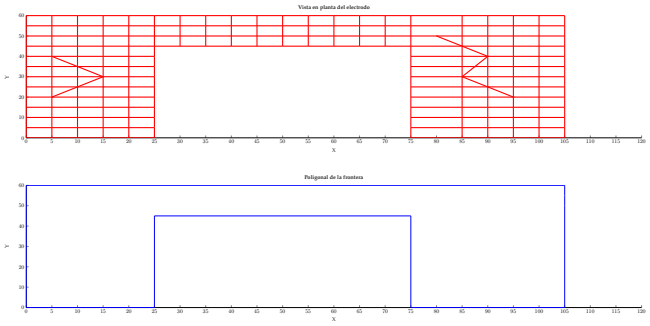


Figura 5.11: Resultados del calculo de la frontera de la malla `MetodosGeometricos.mat`. El cálculo de la frontera en esta malla es relevante, ya que demuestra el correcto funcionamiento del algoritmo incluso cuando se trabaja con un gran número de electrodos en la malla.

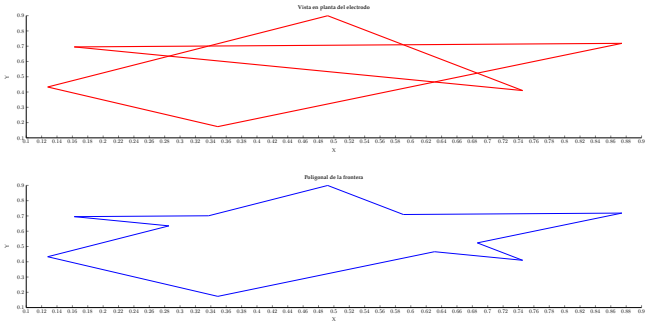


Figura 5.12: Resultados del calculo de la frontera de la malla `RutinasRemalladoTest.mat`. Este ejemplo destaca el correcto funcionamiento del algoritmo en geometrías altamente irregulares, donde es fundamental poder calcular la frontera siguiendo los electrodos. Soluciones como la envolvente convexa o las herramientas implementadas por defecto en MATLAB para el cálculo de envolventes de puntos no reflejan adecuadamente la frontera real. Por lo tanto, este ejemplo demuestra la efectividad de esta rutina.

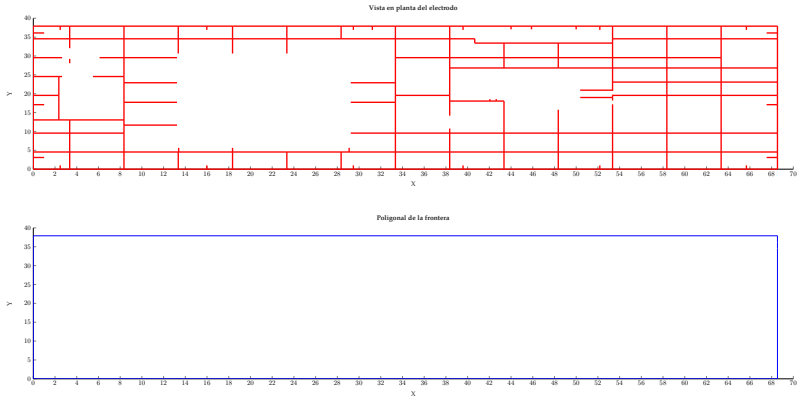


Figura 5.13: Resultados del calculo de la frontera de la malla `Subestacion 1.mat`. En este ejemplo destaca que, aunque existan electrodos que están conectados a la malla solo por un extremo, o incluso totalmente desconectados de ella, la rutina para el cálculo de la frontera permanece en el exterior y no se ve afectada por dichas irregularidades.

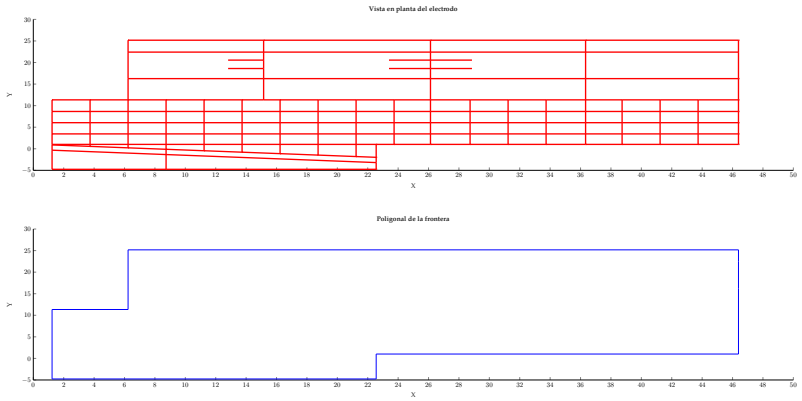


Figura 5.14: Resultados del calculo de la frontera de la malla `Subestacion 2.mat`. Este ejemplo no es especialmente destacable. No obstante, dado que corresponde a uno de los electrodos estudiados en la optimización, se presenta el resultado de su frontera.

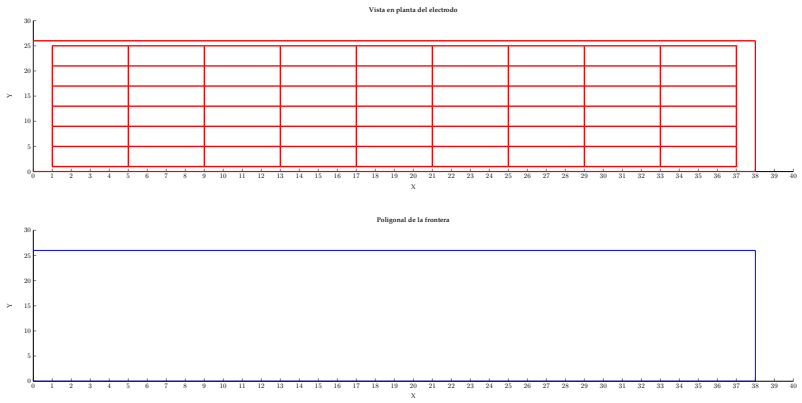


Figura 5.15: Resultados del calculo de la frontera de la malla `Subestacion 3.mat`. En este ejemplo destaca que la malla está compuesta, geométricamente, por dos mallas separadas. No obstante, la rutina detecta correctamente la frontera exterior como la adecuada.

5.1.1.4. Descomposición en convexos

En esta sección se presentan los resultados de la rutina para la descomposición en regiones convexas, correspondiente a la segunda rutina fundamental del cálculo geométrico, descrita en la sección 4.2.3. Como se observa en los ejemplos siguientes, se logra descomponer adecuadamente las regiones no convexas, las cuales no pueden ser analizadas mediante los métodos de optimización definidos para regiones convexas. Además, se pone de manifiesto la eficacia de este método, ya que permite excluir determinadas zonas de la región de estudio, como se muestra en la figura 5.20, con la ventaja de generar un número relativamente bajo de nuevas subregiones.

Figura 5.16: Resultados de la descomposicion en convexos de la frontera calculada anteriormente de la malla `GeometriaIrregular 1.mat`. En este ejemplo se puede observar cómo la rutina actúa de forma recursiva para eliminar regiones no convexas. Como se aprecia, la última división se realiza al intersectar con una arista introducida en un paso anterior. Esta operación transforma la región no convexa en cuatro subregiones convexas.

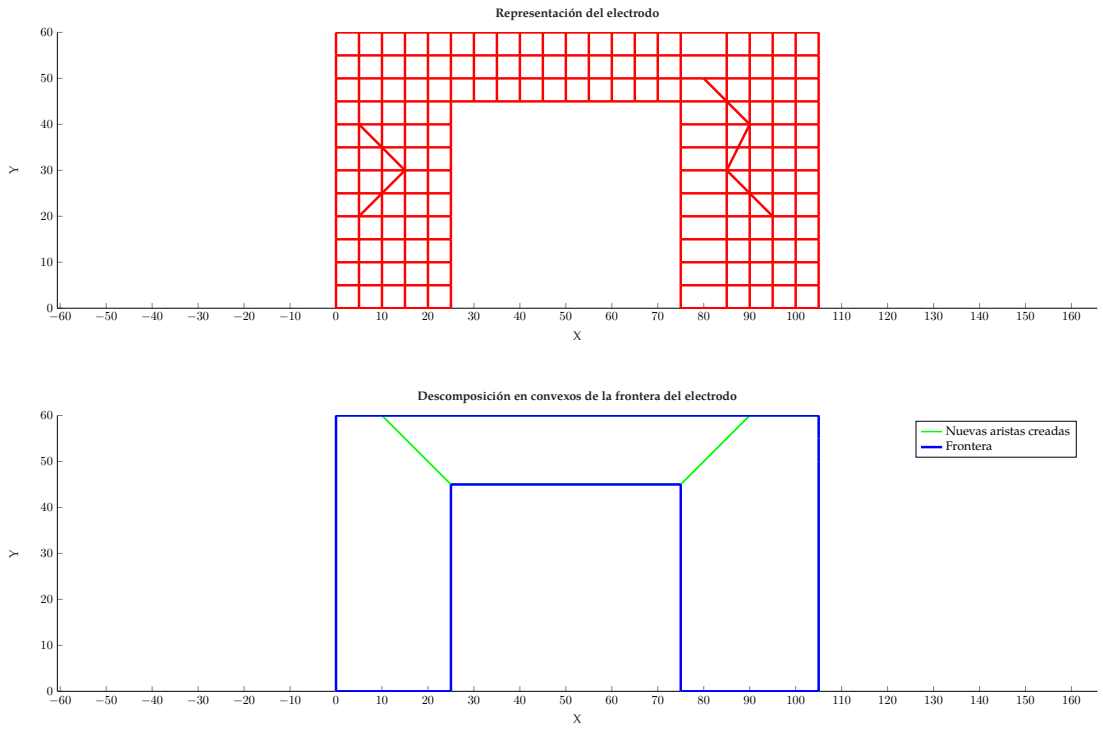


Figura 5.17: Resultados de la descomposicion en convexos de la frontera calculada anteriormente de la malla `MetodosGeometricos.mat`. En este ejemplo, aunque el cálculo de los convexos no resulta especialmente complejo, se puede apreciar cómo la rutina consigue, en casos sencillos como este, obtener un número reducido de regiones, cercano al mínimo necesario. Esta operación transforma la región no convexa en tres subregiones convexas.

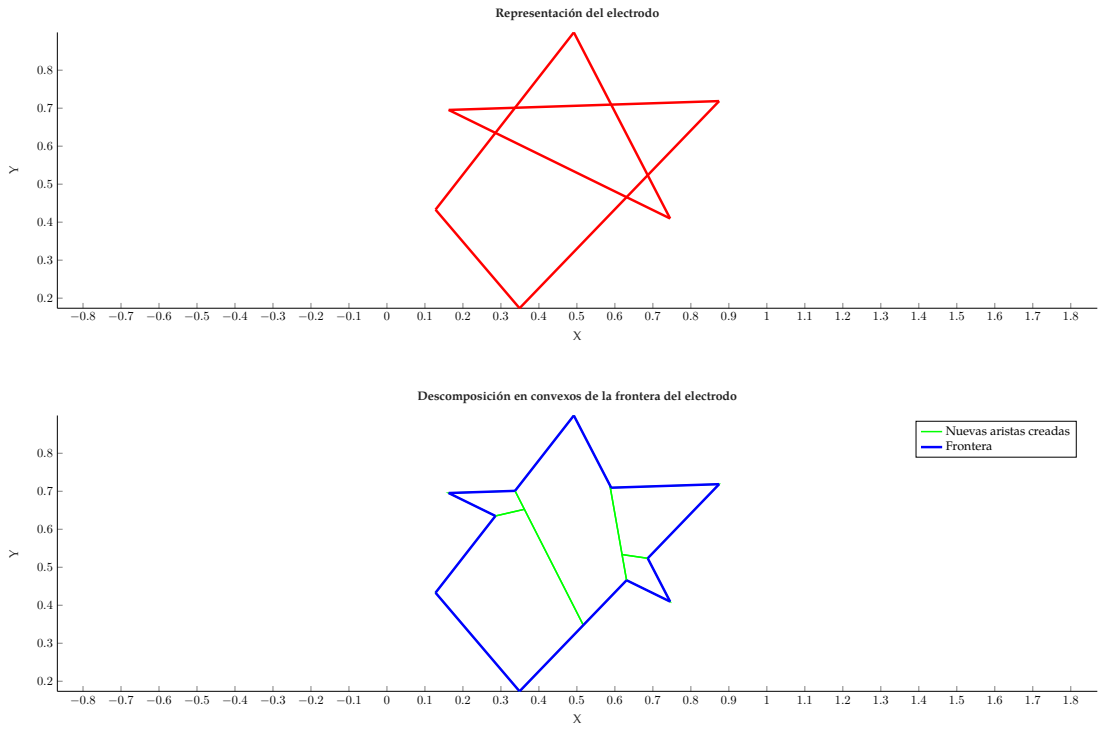


Figura 5.18: Resultados de la descomposicion en convexos de la frontera calculada anteriormente de la malla `RutinasRemalladoTest.mat`. Este ejemplo refuerza la idea del funcionamiento recursivo de la subdivisión en regiones convexas, ya que, al igual que en el ejemplo 5.16, se puede observar cómo las aristas creadas son intersectadas progresivamente al eliminar las concavidades. No obstante, aunque no sea especialmente evidente, en este caso se genera una microregión convexa en las proximidades del punto $[0{,}6, 0{,}7]$, debido a que la intersección no coincide exactamente con dicho punto. Esta pequeña región no se formaría con otros métodos de descomposición. Por tanto, en la operación ha transformado la región no convexa en seis subregiones convexas.

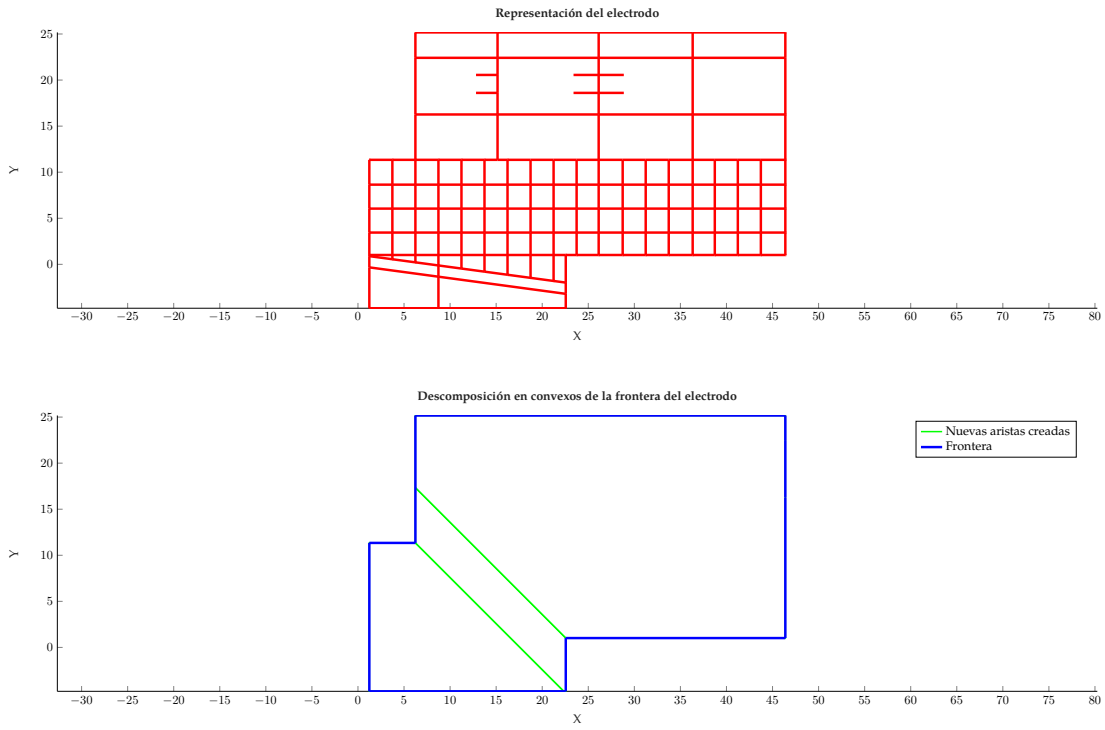


Figura 5.19: Resultados de la descomposicion en convexos de la frontera calculada anteriormente de la malla `Subestacion 2`. Este ejemplo, al igual que el anterior, permite visualizar que no siempre se alcanza una descomposición óptima, ya que en cada paso solo se elimina un punto de concavidad sin tener en cuenta la existencia de otros puntos similares. Como resultado, en este caso se generan tres regiones en lugar del número óptimo, que sería dos.

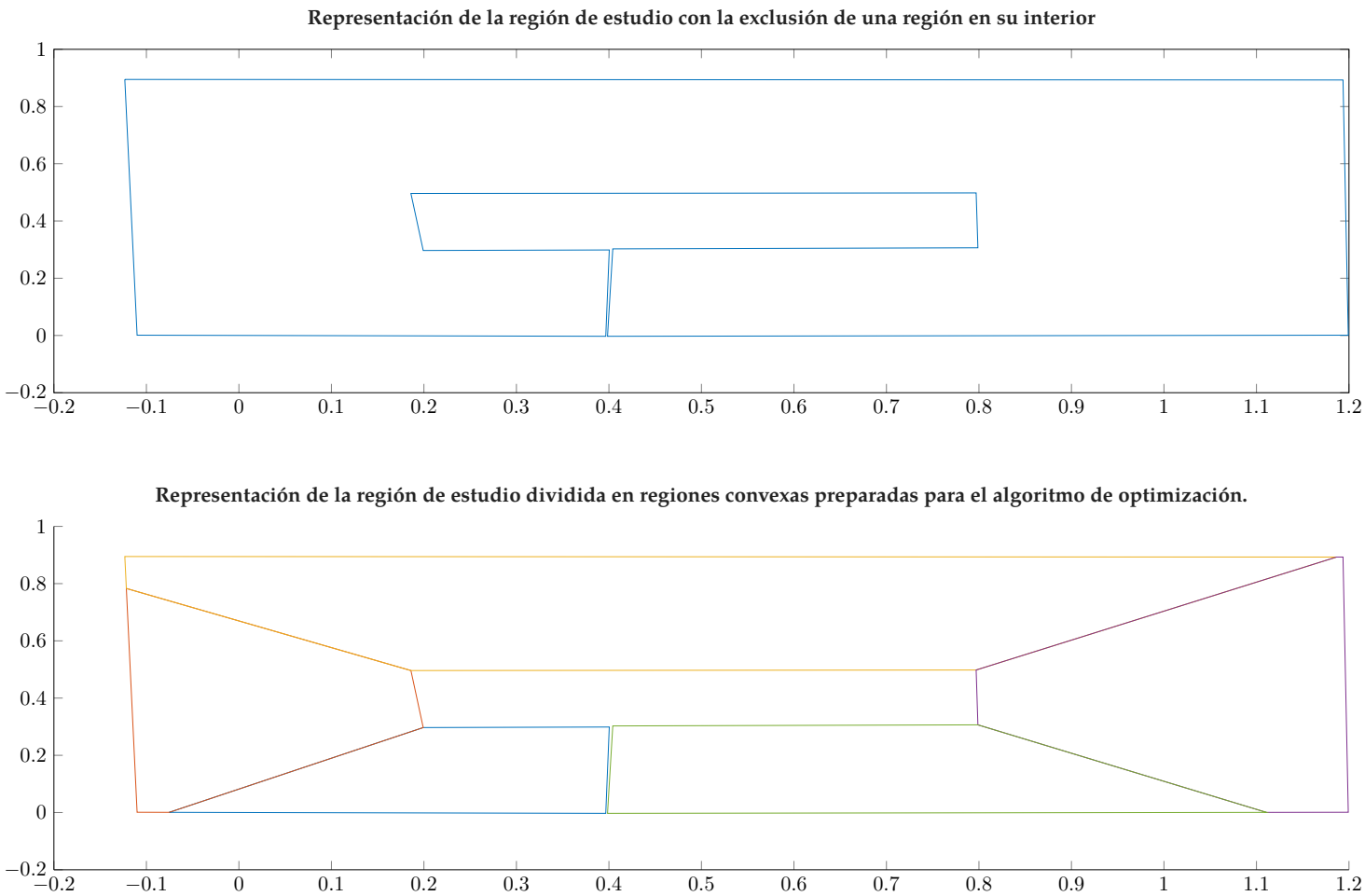


Figura 5.20: Resultados de la descomposición en convexos de una poligonal introducida manualmente, que ilustra cómo puede definirse una región de estudio excluyendo deliberadamente ciertas zonas. Esta posibilidad aporta una robustez importante al cálculo, ya que permite enfocar el análisis en áreas específicas de la subestación cuando existe incertidumbre sobre su seguridad.
Este tipo de regiones es especialmente relevante cuando se estudia la seguridad de la puesta a tierra, ya que, en ocasiones, el mínimo del potencial de contacto podría situarse en una zona no accesible. Por ejemplo, en el interior del recinto de un transformador, lo cual no representa un peligro real. En estos casos, resulta útil excluir dicha región del análisis para evaluar si, fuera de ella, la instalación sigue siendo segura. Esa zona no accesible no supone un riesgo, precisamente por su inaccesibilidad.

5.1.2. Optimización de la función

5.1.2.1. Eleccion factorizacion o gradiente conjugado

En esta sección se presentan los resultados en base a los cuales se decide que algoritmo usar en la función `fmincon`. Se prefiere el uso de gradiente conjugado por tener tiempos de ejecución menores.

Malla	Cálculo del paso	Punto óptimo	Función Objetivo	Tiempo (s)
Irregular				
Mínimo	Factorización	[0.000009,0.000009]	0.018455	0.073761
	Gradiente Conjugado	[0.000010,0.000009]	0.018455	0.036569
Máximo	Factorización	[8.032689,8.947708]	0.023693	0.005863
	Gradiente Conjugado	[8.032689,8.947708]	0.023693	0.005973
Cuadrado				
Mínimo	Factorización	[0.000009,0.000009]	0.018455	0.210349
	Gradiente Conjugado	[0.000009,0.000009]	0.018455	0.105206
Máximo	Factorización	[8.032690,6.052289]	0.023693	0.007048
	Gradiente Conjugado	[8.032690,6.052289]	0.023693	0.004241
Subestación				
Mínimo	Factorización	[44.25050,19.98706]	0.009846	0.075156
	Gradiente Conjugado	[44.25050,19.98706]	0.009846	0.023481
Máximo	Factorización	[21.23730,3.441349]	0.012995	0.029293
	Gradiente Conjugado	[21.23730,3.441349]	0.012995	0.020208

Tabla 5.2: Comparación de resultados obtenidos mediante el algoritmo fmincon usando el método de gradiente conjugado o el método de factorización para el calculo del paso. Se usan los ejemplos *GeometriaIrregularSimple.mat*, *MallaSimpleRectangulo.mat* y *Subestacion 2.mat*.

5.1.2.2. Preproceso realizado para las mallas a optimizar

Antes de proceder a la optimización de las mallas, y con el objetivo de mejorar los resultados obtenidos mediante el método de simulación de cargas (CSM), se lleva a cabo un preprocesado de las mallas utilizando rutinas de remallado. En las mallas `MallaSimpleRectangulo.mat`, `Subestacion 1.mat` y `Subestacion 3.mat` se realiza únicamente la descompresión.Por su parte, en la malla `Planta fotovoltaica.mat` se aplica, además, un remallado con una longitud máxima de 5 metros. Por último, en la malla `Subestacion 2.mat` se efectúa el remallado fino.

Variable	MallaSimpleRectangulo.mat	Planta fotovoltaica.mat	Subestacion 1.mat	Subestacion 2.mat	Subestacion 3.mat
Longitud Mínima $[m]$	1.000	0.016	0.039	$3.443{\cdot}10^{-5}$	2.000
Longitud Máxima $[m]$	2.000	5.000	10.00	13.95	38.00
Longitud Media $[m]$	1.833	3.917	3.609	2.341	4.794
Número de Segmentos	360	1574	512	726	262
Kr inicial $\left[\frac{\Omega}{\Omega m}\right]$	$2.539{\cdot}10^{-2}$	$2.484{\cdot}10^{-3}$	$9.033{\cdot}10^{-3}$	$1.411{\cdot}10^{-2}$	$1.374{\cdot}10^{-2}$
Kr final $\left[\frac{\Omega}{\Omega m}\right]$	$2.539{\cdot}10^{-2}$	$2.516{\cdot}10^{-3}$	$8.900{\cdot}10^{-3}$	$1.411{\cdot}10^{-2}$	$1.373{\cdot}10^{-2}$

Tabla 5.3: Estadísticas finales de longitud mínima, máxima y media de segmento, número de segmentos, y coeficiente de resistencia tanto inicial como final de las cinco mallas analizadas en el estudio de óptimo.

5.1.2.3. Resultados optimización

En esta sección se presentan los resultados obtenidos tras optimizar las mallas mencionadas anteriormente, es decir, tras determinar el mínimo y el máximo de la función potencial en la región de estudio, definida o bien por la frontera o bien de forma manual. Para cada malla, se incluyen las siguientes tablas:

- Una tabla con los resultados obtenidos mediante la aplicación de un algoritmo genético combinado con el método del gradiente conjugado.
- Una tabla con los resultados de la rutina `testOptimo.m`, la cual implementa un cálculo por fuerza bruta con una resolución de 1000 puntos tanto en el eje X como en el eje Y, evaluando la función en todos esos puntos discretos del espacio. Este método corresponde al procedimiento utilizado previamente para la optimización.
- Una tabla con los tiempos de ejecución de cada algoritmo: uno por cada región convexa utilizando el algoritmo genético, seguido por el gradiente conjugado asociado al mínimo o máximo del conjunto de dichas regiones, así como el tiempo de ejecución de la rutina `testOptimo.m`, utilizada como referencia o benchmark.

Finalmente, se incluyen representaciones gráficas que ilustran tanto las regiones analizadas en las mallas correspondientes a las subestaciones, como los resultados obtenidos con los distintos algoritmos empleados. No se incluye la región de estudio de la planta fotovoltaica, ya que esta puede variar considerablemente en función de cómo se dibuje. En este caso, lo único relevante es que dicha región contenga por completo el electrodo, ya que lo importante en este estudio es identificar el valor máximo que puede alcanzar el potencial.

1. `MallaSimpleRectangulo.mat`

Número de regiones	Punto mínimo	Función mínimo	Punto máximo	Función máximo
1	$[9.4558{\cdot}10^{-6}, 9.385{\cdot}10^{-6}]$	$1.855{\cdot}10^{-2}$	[8.033,8.948]	$2.370{\cdot}10^{-1}$

Tabla 5.4: Resultados de la ejecución de la optimización mediante el algoritmo mixto genético-gradiente para cada una de las subregiones de la frontera de la malla `mallaSimpleRectangulo.mat`. La región de estudio se puede visualizar en la figura 5.9.

Número de puntos	Punto mínimo	Función mínimo	Punto máximo	Función máximo
10^6	[0.0200,0.0150]	$1.855{\cdot}10^{-2}$	[8.028,8.948]	$2.370{\cdot}10^{-1}$

Tabla 5.5: Resultados de ejecución de la rutina `testOptimo` para la malla `mallaSimpleRectangulo.mat`. Como se puede observar, los resultados del calculo son prácticamente idénticos. Por tanto, se puede dar veracidad a los resultados obtenidos.

Algoritmo	**Tiempo (s)**	Operación
Ga	2.271	Minimiza
fmincon	0.097	Minimiza
Ga	2.080	Maximiza
fmincon	0.008	Maximiza
Suma algogitmo mixto	4.456	ambos
testOptimo	97.275	ambos

Tabla 5.6: Resultados de tiempos de ejecución de los diferentes algoritmos de optimización empleados en la malla `mallaSimpleRectangulo.mat`. Como se puede observar la aplicación del nuevo algoritmo supone una mejoría de tiempos del 95.42 %.

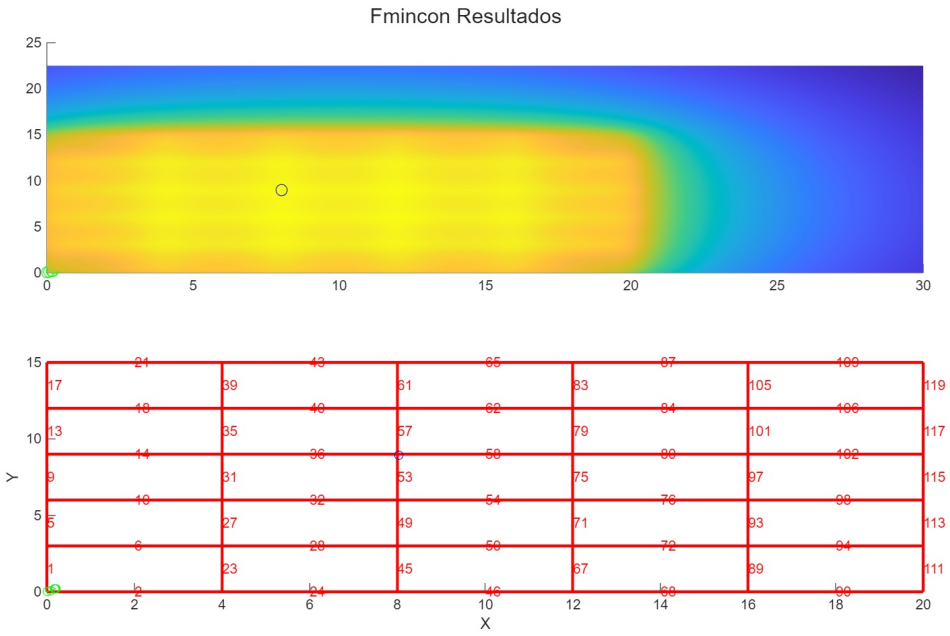


Figura 5.21: Resultados de la rutina `fmincon` ejecutada con los máximo y mínimo obtenidos por el algoritmo genético en la malla `mallaSimpleRectangulo.mat`.

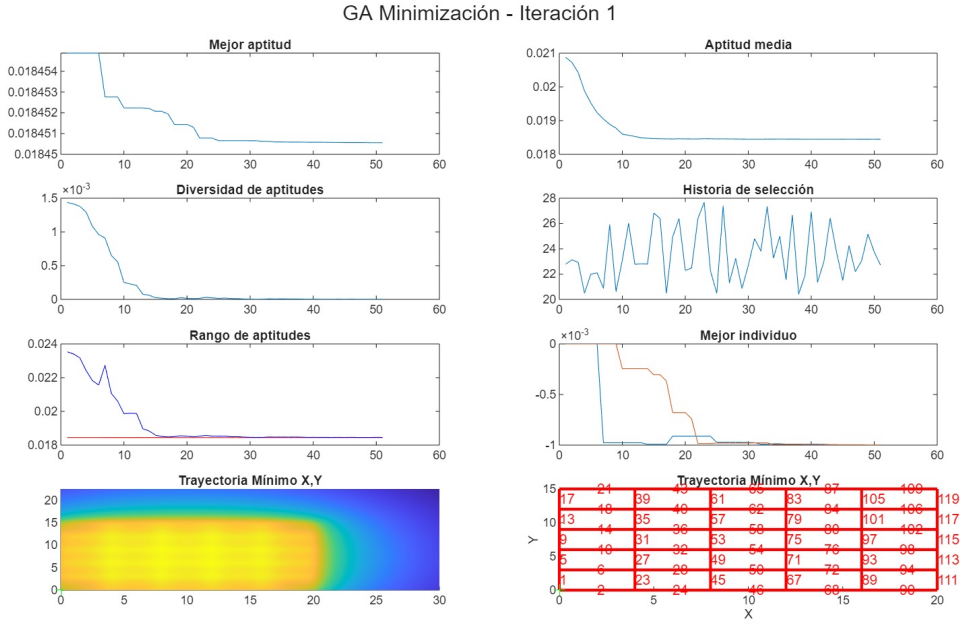


Figura 5.22: Resultados de la rutina `ga` ejecutada con el objetivo de minimizar la malla `mallaSimpleRectangulo.mat`.

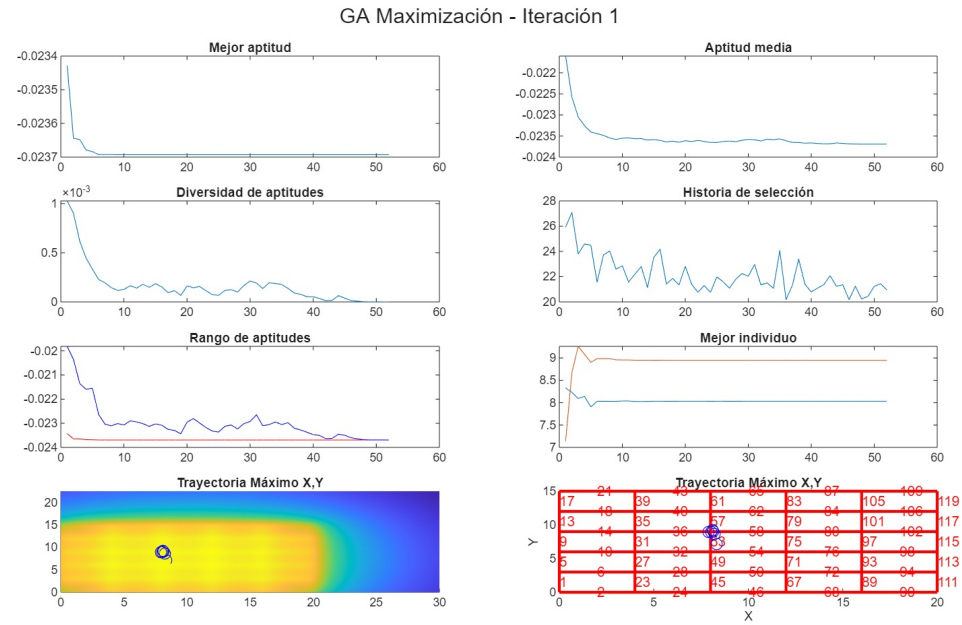


Figura 5.23: Resultados de la rutina `ga` ejecutada con el objetivo de maximizar la malla `mallaSimpleRectangulo.mat`. Se puede claramente observar como el algoritmos genético va analizando distintos individuos al mover el individuo óptimo representado con el círculo azul.

2. `Subestacion 1.mat`

Número de regiones	Punto mínimo	Función mínimo	Punto máximo	Función máximo
1	[20.60,18.57]	$6.736 \cdot 10^{-3}$	[53.37,26.87]	$8.611 \cdot 10^{-3}$

Tabla 5.7: Resultados de la ejecución de la optimización mediante el algoritmo mixto genético-gradiente para cada una de las subregiones de la frontera de la malla `Subestacion 1.mat`. La región de estudio se puede visualizar en la figura 5.13.

Número de puntos	Punto mínimo	Función mínimo	Punto máximo	Función máximo
10^6	[20.58,18.59]	$6.736 \cdot 10^{-3}$	[53.38,26.86]	$8.611 \cdot 10^{-3}$

Tabla 5.8: Resultados de ejecución de la rutina `testOptimo` para la malla `Subestacion 1.mat`. Como se puede observar, los resultados del calculo son prácticamente idénticos. Por tanto, se puede dar veracidad a los resultados obtenidos.

Algoritmo	Tiempo (s)	Operación
Ga	8.169	Minimiza
fmincon	0.085	Minimiza
Ga	7.861	Maximiza
fmincon	0.029	Maximiza
Suma algogitmo mixto	16.144	ambos
testOptimo	440.844	ambos

Tabla 5.9: Resultados de tiempos de ejecución de los diferentes algoritmos de optimización empleados en la malla `Subestacion 1.mat`. Como se puede observar la aplicación del nuevo algoritmo supone una mejoría de tiempos del 96.34 %.

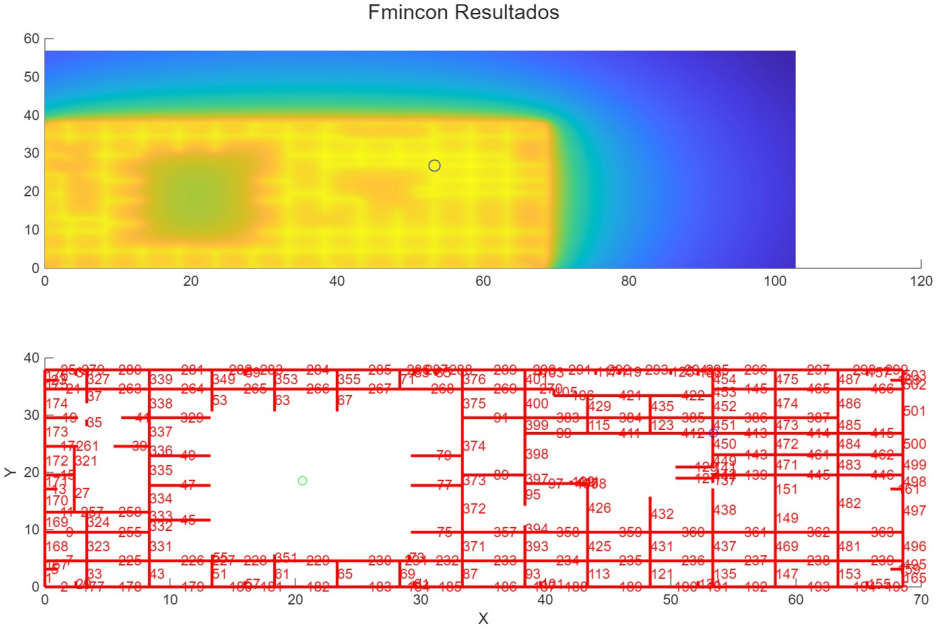


Figura 5.24: Resultados de la rutina `fmincon` ejecutada con los máximo y mínimo obtenidos por el algoritmo genético en la malla `Subestacion 1.mat`.

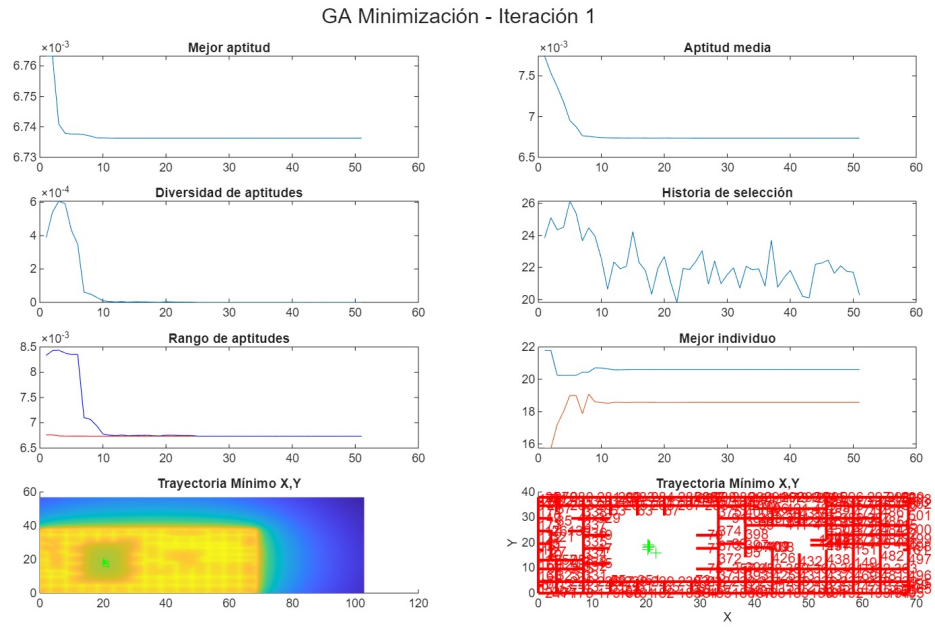


Figura 5.25: Resultados de la rutina `ga` ejecutada con el objetivo de minimizar la malla `Subestacion 1.mat`.

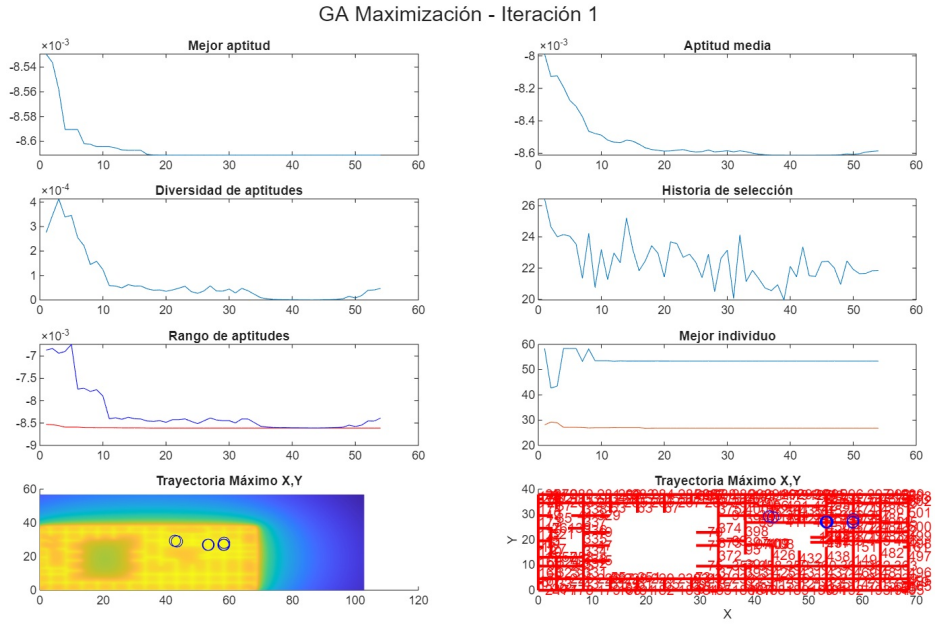


Figura 5.26: Resultados de la rutina `ga` ejecutada con el objetivo de maximizar la malla `Subestacion 1.mat`.

3. `Subestacion 2.mat`

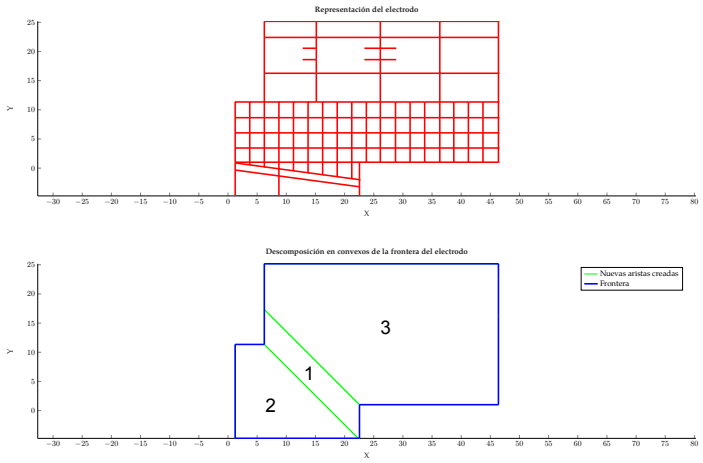


Figura 5.27: Representación gráfica de las regiones analizadas por el algoritmo genético durante la optimización.

Número de regiones	Punto mínimo	Función mínimo	Punto máximo	Función máximo
3	[36.39,25.17]	$9.652 \cdot 10^{-3}$	[21.24,3.442]	$1.300 \cdot 10^{-2}$

Tabla 5.10: Resultados de la ejecución de la optimización mediante el algoritmo mixto genético-gradiente para cada una de las subregiones de la frontera de la malla `Subestacion 2.mat`. Las regiones de estudio se pueden visualizar en la figura 5.27.

Número de puntos	Punto mínimo	Función mínimo	Punto máximo	Función máximo
10^6	[21.26,3.446]	$9.719 \cdot 10^{-3}$	[46.34,25.14]	$1.300 \cdot 10^{-2}$

Tabla 5.11: Resultados de ejecución de la rutina `testOptimo` para la malla `Subestacion 2.mat`. En este ejemplo, el método de barrido no consigue identificar el mínimo real, incurriendo en un error del 0.69 %. Aunque esta diferencia es baja desde un punto de vista práctico, implica una sobreestimación de la seguridad real de la subestación. En consecuencia, el algoritmo genético ofrece resultados más precisos y fiables.

Algoritmo	**Tiempo (s)**	Operación
Ga 1	11.237	Minimiza
Ga 2	10.499	Minimiza
Ga 3	10.606	Minimiza
fmincon	0.393	Minimiza
Ga 1	10.954	Maximiza
Ga 2	10.696	Maximiza
Ga 3	10.918	Maximiza
fmincon	0.072	Maximiza
Suma algogitmo mixto	65.375	ambos
testOptimo	492.001	ambos

Tabla 5.12: Resultados de tiempos de ejecución de los diferentes algoritmos de optimización empleados en la malla `Subestacion 2.mat`. Como se puede observar, la aplicación del nuevo algoritmo representa una mejora del 86.71 % en los tiempos de ejecución. No obstante, el hecho de que la frontera no sea convexa y, por tanto, deban analizarse más regiones, conlleva una reducción de tiempos menos pronunciada.

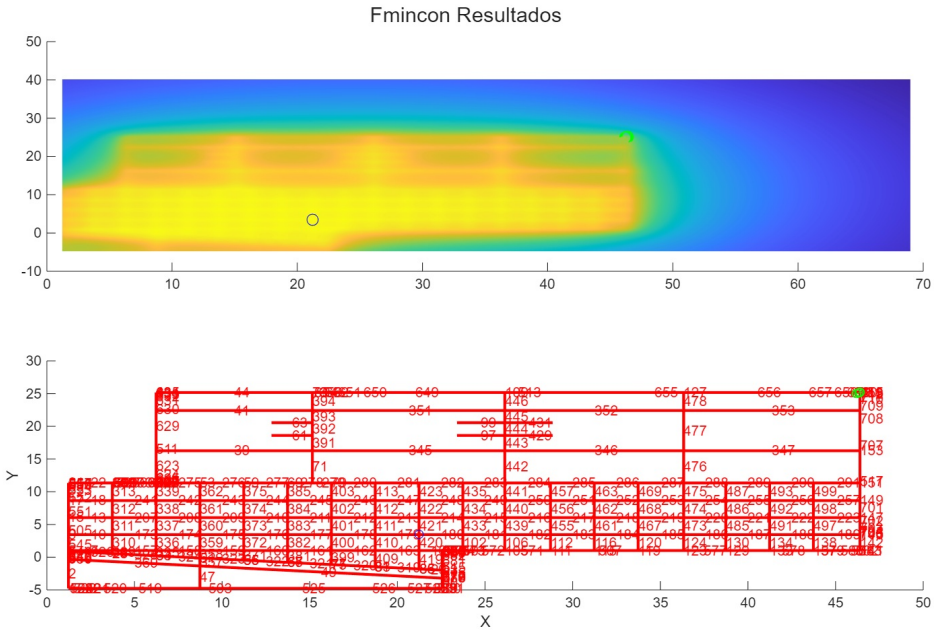


Figura 5.28: Resultados de la rutina `fmincon` ejecutada con los máximo y mínimo obtenidos por el algoritmo genético en la malla `Subestacion 2.mat`.

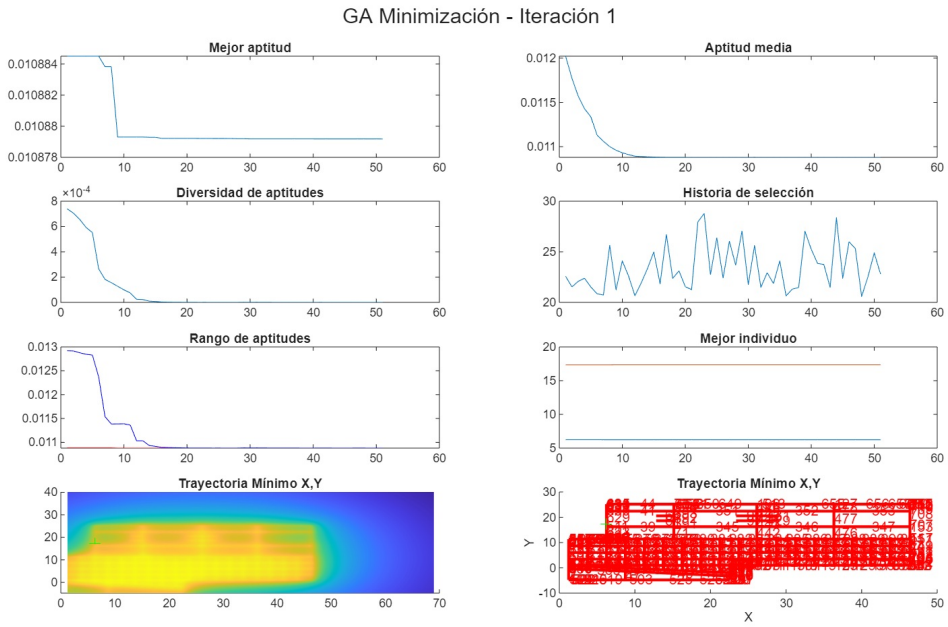


Figura 5.29: Resultados de la rutina `ga` ejecutada con el objetivo de minimizar la malla `Subestacion 2.mat` en la primera subregión.

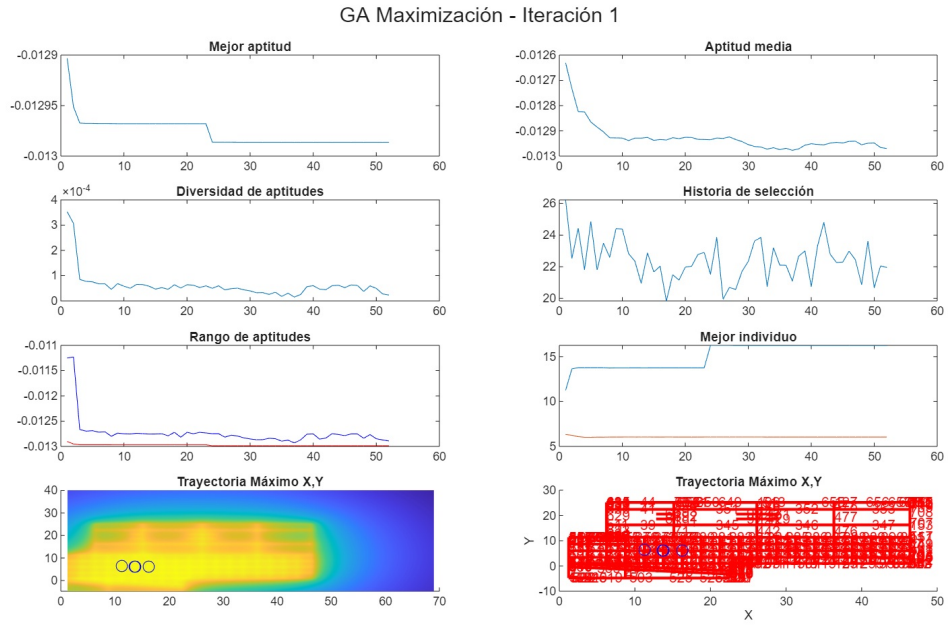


Figura 5.30: Resultados de la rutina `ga` ejecutada con el objetivo de maximizar la malla `Subestacion 2.mat` en la primera subregión.

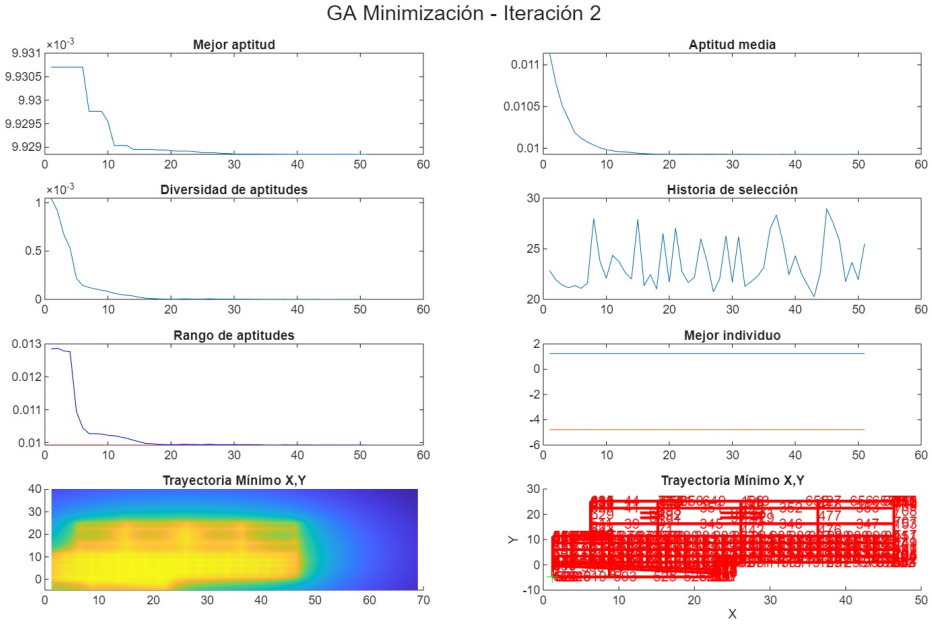


Figura 5.31: Resultados de la rutina `ga` ejecutada con el objetivo de minimizar la malla `Subestacion 2.mat` en la segunda subregión.

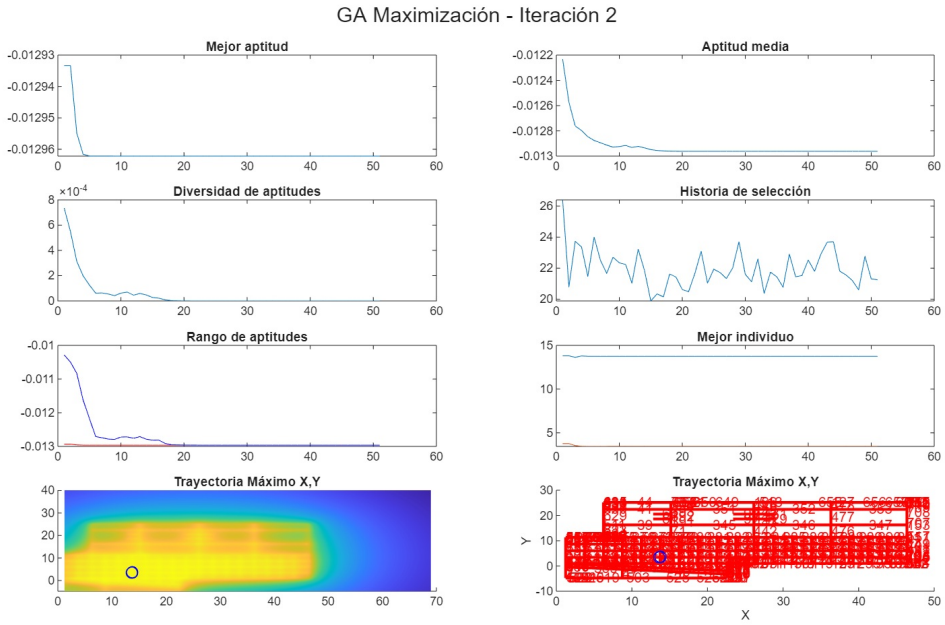


Figura 5.32: Resultados de la rutina `ga` ejecutada con el objetivo de maximizar la malla `Subestacion 2.mat` en la segunda subregión.

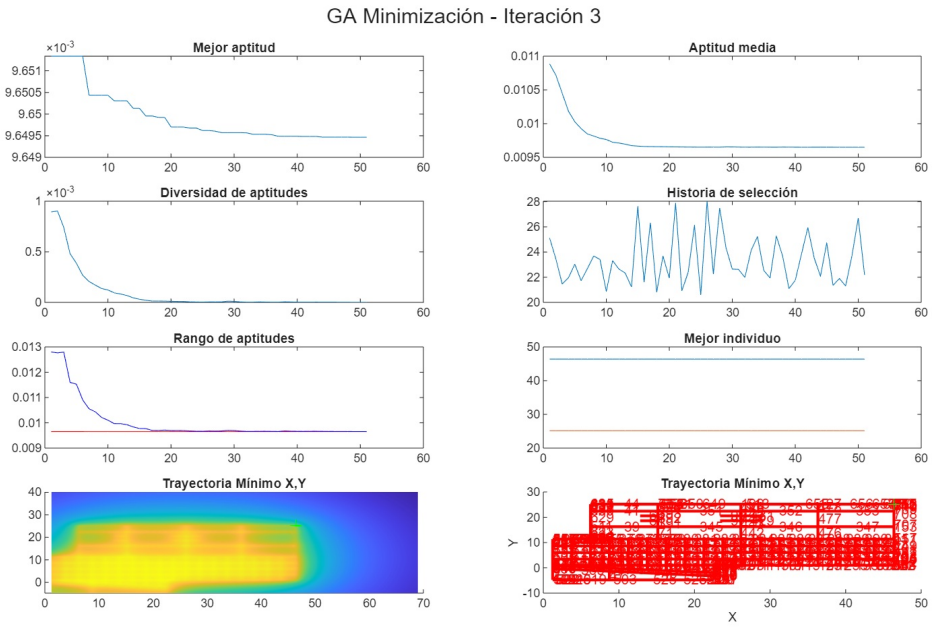


Figura 5.33: Resultados de la rutina `ga` ejecutada con el objetivo de minimizar la malla `Subestacion 2.mat` en la tercera subregión.

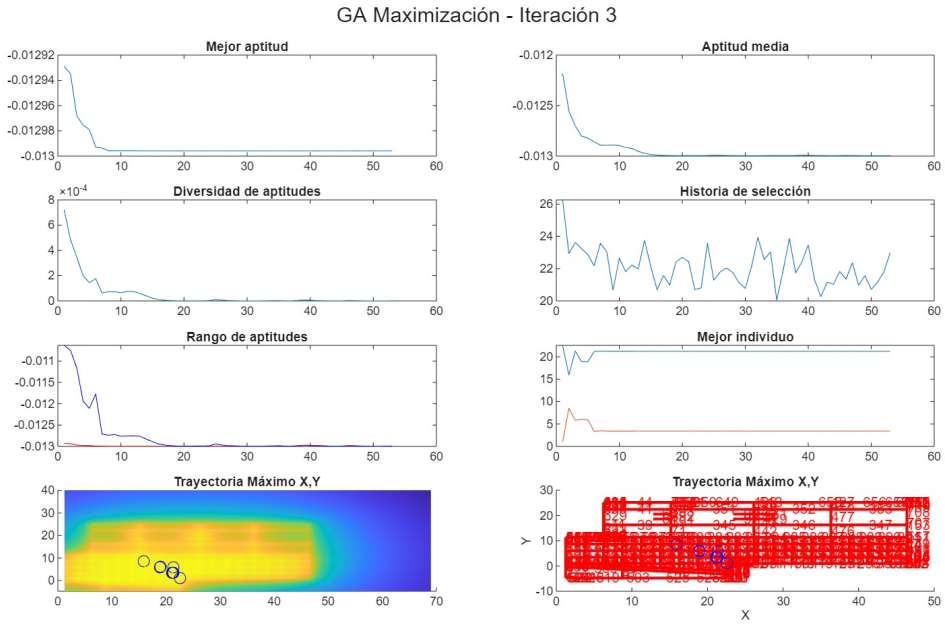


Figura 5.34: Resultados de la rutina `ga` ejecutada con el objetivo de maximizar la malla `Subestacion 2.mat` en la tercera subregión.

4. `Subestacion 3.mat`

Número de regiones	Punto mínimo	Función mínimo	Punto máximo	Función máximo
1	$[38.00, 5.000 \cdot 10^{-4}]$	$1.050 \cdot 10^{-2}$	[20.99,13.00]	$1.319 \cdot 10^{-2}$

Tabla 5.13: Resultados de la ejecución de la optimización mediante el algoritmo mixto genético-gradiente para cada una de las subregiones de la frontera de la malla `Subestacion 3.mat`. La región de estudio se puede visualizar en la figura 5.15.

Número de puntos	Punto mínimo	Función mínimo	Punto máximo	Función máximo
10^6	[37.96,0.0260]	$1.056 \cdot 10^{-2}$	[20.99,12.99]	$1.319 \cdot 10^{-2}$

Tabla 5.14: Resultados de ejecución de la rutina `testOptimo` para la malla `Subestacion 3.mat`. En este ejemplo, el método de barrido no consigue identificar el mínimo real, incurriendo en un error del 0.57 %. Aunque esta diferencia es baja desde un punto de vista práctico, implica una sobreestimación de la seguridad real de la subestación. En consecuencia, el algoritmo genético ofrece resultados más precisos y fiables.

Algoritmo	**Tiempo (s)**	Operación
Ga	4.964	Minimiza
fmincon	0.275	Minimiza
Ga	4.352	Maximiza
fmincon	0.026	Maximiza
Suma algogitmo mixto	9.617	ambos
testOptimo	219.653	ambos

Tabla 5.15: Resultados de tiempos de ejecución de los diferentes algoritmos de optimización empleados en la malla `Subestacion 3.mat`. Como se puede observar, la aplicación del nuevo algoritmo representa una mejora del 95.62 % en los tiempos de ejecución.

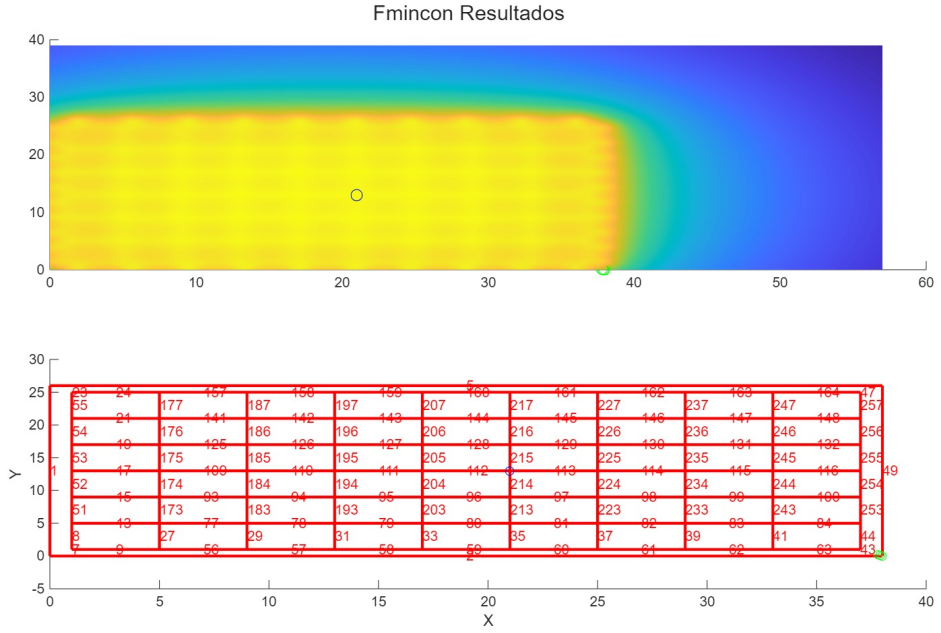


Figura 5.35: Resultados de la rutina `fmincon` ejecutada con los máximo y mínimo obtenidos por el algoritmo genético en la malla `Subestacion 3.mat`.

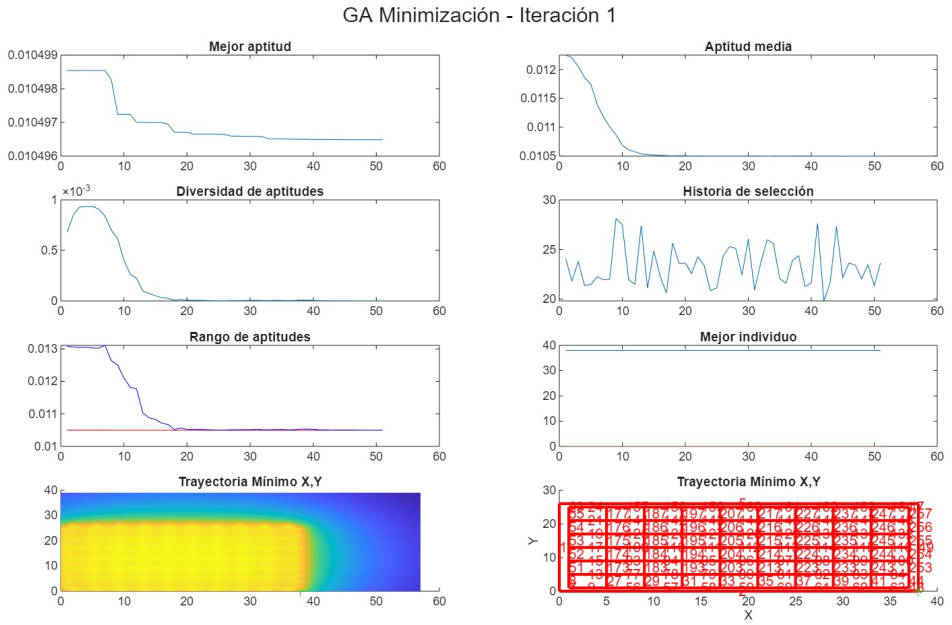


Figura 5.36: Resultados de la rutina `ga` ejecutada con el objetivo de minimizar la malla `Subestacion 3.mat`.

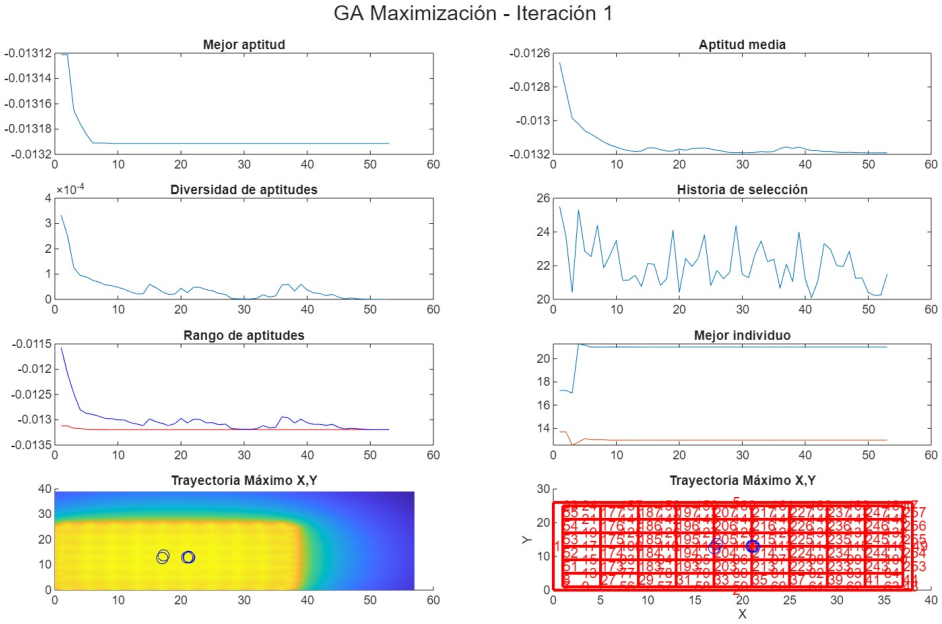


Figura 5.37: Resultados de la rutina `ga` ejecutada con el objetivo de maximizar la malla `Subestacion 3.mat`.

5. `Planta fotovoltaica.mat`

Número de regiones	Punto mínimo	Función mínimo	Punto máximo	Función máximo
1	$[4.313{\cdot}10^5, 4.341{\cdot}10^6]$	$7.260{\cdot}10^{-4}$	$[4.313{\cdot}10^5, 4.341{\cdot}10^6]$	$2.442{\cdot}10^{-3}$

Tabla 5.16: Resultados de la ejecución de la optimización mediante el algoritmo mixto genético-gradiente una región convexa dibujada a mano de la malla `Planta fotovoltaica.mat`. La región de estudio no se incluye porque como se ha mencionado anteriormente, lo interesante en esta malla es el estudio del máximo. Además, por sus grandes dimensiones no se pueden distinguir las coordenadas del punto con la precisión empleada para las tablas.

Número de puntos	Punto mínimo	Función mínimo	Punto máximo	Función máximo
10^6	$[4.313{\cdot}10^5, 4.341{\cdot}10^6]$	$6.656{\cdot}10^{-4}$	$[4.313{\cdot}10^5, 4.341{\cdot}10^6]$	$2.439{\cdot}10^{-3}$

Tabla 5.17: Resultados de ejecución de la rutina `testOptimo` para la malla `Planta fotovoltaica.mat`. En este caso, ya se puede apreciar cómo el método de barrido no logra encontrar el máximo real, el cual sí ha sido identificado por el algoritmo genético. Por tanto, el uso de este método representa también una desventaja desde el punto de vista del análisis de seguridad, aunque en la práctica esta diferencia resulte poco significativa, al ser del orden del 0.12 %.

Algoritmo	**Tiempo (s)**	Operación
Ga	23.052	Minimiza
fmincon	0.268	Minimiza
Ga	24.085	Maximiza
fmincon	0.079	Maximiza
Suma algogitmo mixto	47.484	ambos
testOptimo	1133.508	ambos

Tabla 5.18: Resultados de tiempos de ejecución de los diferentes algoritmos de optimización empleados en la malla `Planta fotovoltaica.mat`. Como se puede observar la aplicación del nuevo algoritmo supone una mejoría de tiempos del 95.81 %.

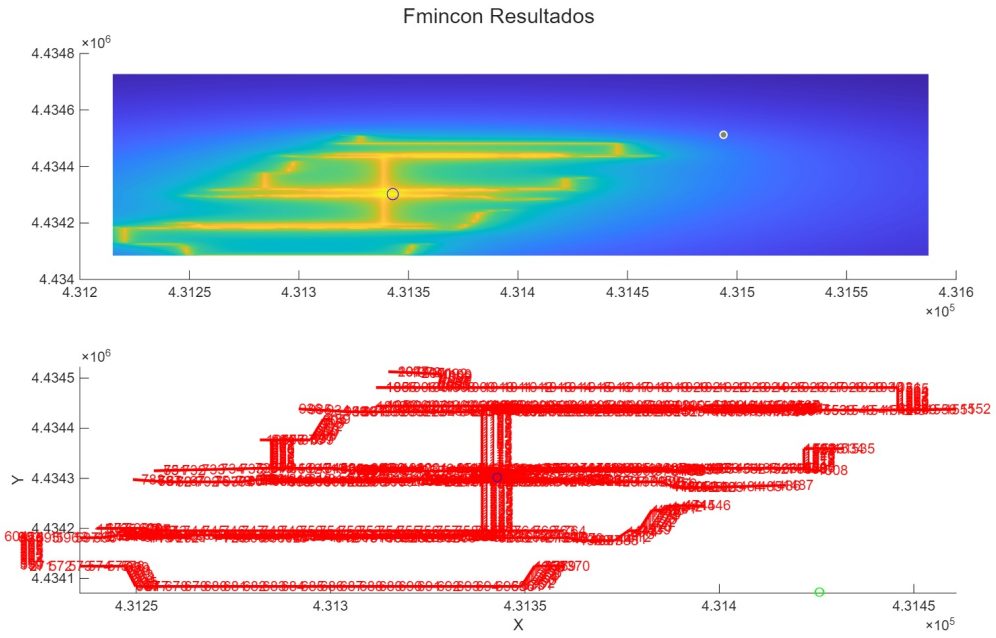


Figura 5.38: Resultados de la rutina `fmincon` ejecutada con los máximo y mínimo obtenidos por el algoritmo genético en la malla `Planta fotovoltaica.mat`.

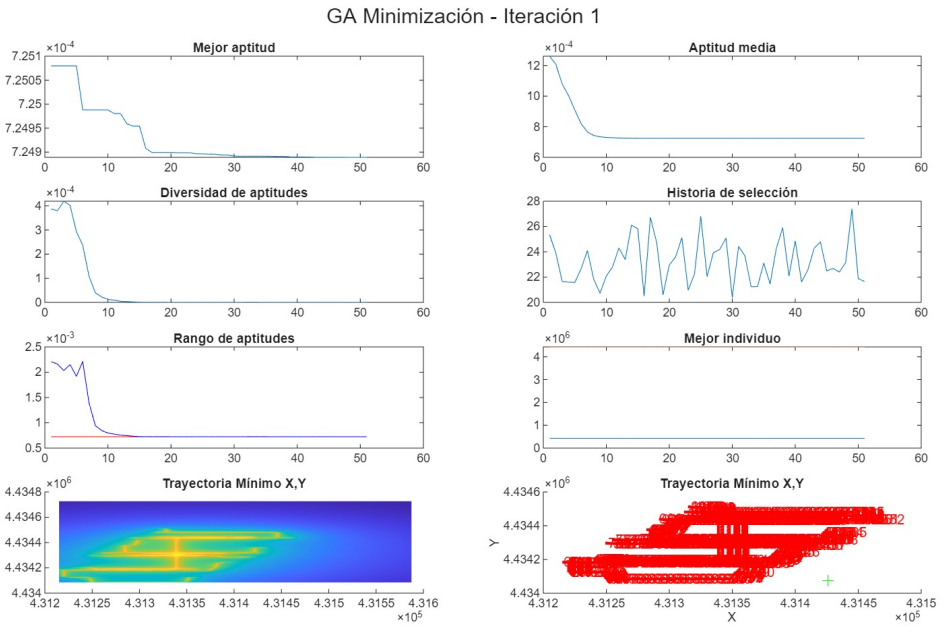


Figura 5.39: Resultados de la rutina `ga` ejecutada con el objetivo de minimizar la malla `Planta fotovoltaica.mat`.

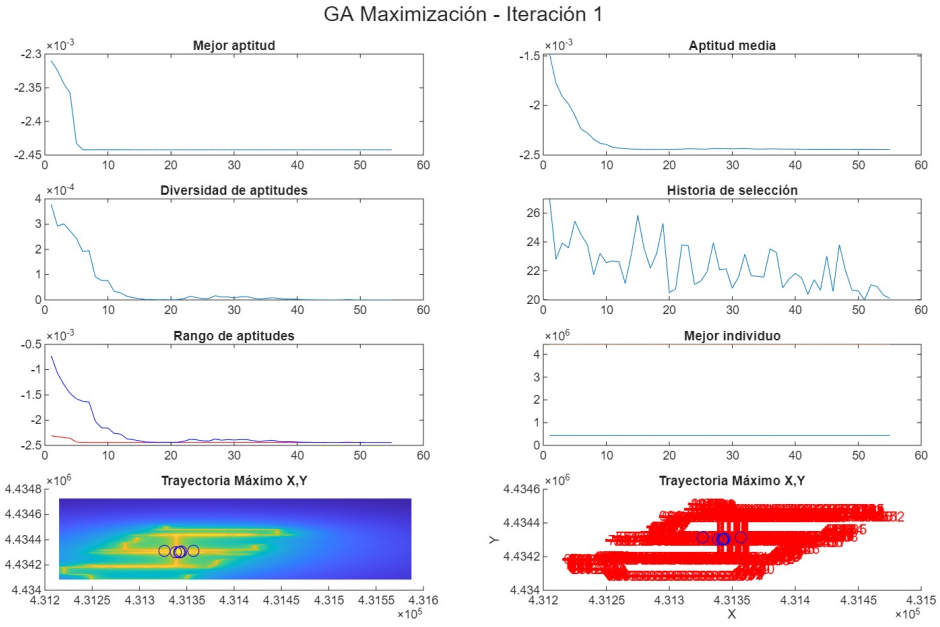


Figura 5.40: Resultados de la rutina `ga` ejecutada con el objetivo de maximizar la malla `Planta fotovoltaica.mat`.

5.2. Discusión

En esta sección se discuten los resultados, considerando el estado del arte y las metodologías alternativas que podrían haberse aplicado, así como las razones que justificaron las decisiones tomadas en este trabajo.

5.2.1. Métodos geométricos

5.2.1.1. Calculo de la frontera

Aunque en el cálculo de la frontera no se han mencionado artículos específicos del estado del arte que describan el proceso, en MATLAB existen dos funciones integradas que también podrían utilizarse para calcular envolventes. Sin embargo, a continuación se exponen las razones por las que no se optó por estas opciones:

- Función `convhull` [30]: esta función, como su nombre indica, permite el cálculo de la envolvente convexa de una nube de puntos. Aunque este enfoque podría parecer razonable, ya que determina una región que incluye la subestación, en situaciones como las mostradas en los ejemplos 5.7 o 5.4, como el potencial siempre decae rápidamente fuera del electrodo de puesta a tierra. Esto podría hacer que el mínimo se ubique en una zona fuera del electrodo, que no tuviese relevancia para la seguridad de la instalación, por lo que el resultado carecería de significado.
- Función `boundary` [31]: esta función, mediante el uso de un parámetro llamado `shrinkFactor`, permite calcular una envolvente que contenga la nube de puntos que define el electrodo. Aunque este enfoque podría parecer adecuado, ya que, si se determina correctamente el valor del parámetro, podría generar un resultado suficientemente bueno, y por tanto no parecería necesario elaborar un algoritmo propio para el cálculo, presenta las siguientes limitaciones:
  - Determinar el valor del parámetro `shrinkFactor` es sumamente complejo. Además, cuando se cambia el electrodo de puesta a tierra, cada geometría tiene un valor de parámetro asociado distinto. Por lo tanto, los resultados de la frontera obtenida se vuelven impredecibles, ya que no se ajustan de manera precisa a la geometría real.
  - Si el parámetro `shrinkFactor` tiene un valor demasiado elevado (cercano a 1), se corre el riesgo, al igual que con el cálculo de la envolvente convexa, de incluir en la región zonas de estudio poco relevantes para la seguridad de la instalación, lo que podría entorpecer los resultados.
  - Si el parámetro `shrinkFactor` tiene un valor demasiado bajo (cercano a 0), a diferencia del caso anterior. Donde un valor elevado podía pecar de excesiva seguridad. Aquí el riesgo es el de pecar de inseguridad. Con un `shrinkFactor` demasiado bajo la envolvente podría omitir regiones esenciales donde se ve comprometida la seguridad de la puesta a tierra, lo que conduciría a dar por segura una instalación que en realidad presenta zonas vulnerables. De este modo, se perdería el objetivo principal de estas comprobaciones que es el de garantizar que la instalación es realmente segura.

Por tanto, a la vista de los resultados y de las alternativas la opción lógica que se ha decidido es la de usar la implementación propia para el calculo de la frontera.

5.2.1.2. Descomposición convexa

En lo que respecta a la descomposición en regiones convexas, resulta relevante comparar la descomposición realizada con aquellas que son pertinentes para este trabajo y que han sido mencionadas en el estado del arte. En particular, es importante contextualizarla en relación con la triangulación presentada en [9], así como con la descomposición en patrones X e Y descrita en [8].

Aunque la triangulación es un método ampliamente utilizado y reconocido por ofrecer resultados robustos y fiables, tiende a generar un número significativamente mayor de regiones en comparación con las descomposiciones trapezoidales que produce el método actual. Si bien estas regiones triangulares son geométricamente más simples, lo que podría facilitar la verificación de restricciones, en este caso particular, dichas restricciones son lineales y su comprobación no conlleva un coste computacional elevado. Por ello, esta ventaja no compensa necesariamente el impacto negativo derivado del aumento en la cantidad de regiones.

En la práctica, un incremento en el número de regiones puede afectar de forma considerable al rendimiento del proceso global, especialmente porque la etapa más costosa es la ejecución del algoritmo genético. Un mayor número de regiones implica evaluar más puntos, lo que puede suponer, en el peor de los casos, analizar hasta dos o tres veces más elementos y, en consecuencia, obtener resultados de menor calidad.

Por esta misma razón, resulta atractivo considerar la implementación de las descomposiciones en patrones X e Y descritas en [8]. Este enfoque sería particularmente interesante de aplicar en las mallas mostradas en las Figuras 5.16, 5.17 y 5.19, donde podría valer la pena comprobar si se logra una reducción en el número de regiones generadas. No obstante, cabe señalar que las regiones no convexas que delimitan las zonas de estudio tienden a ser relativamente cuadradas y no presentan ángulos tan agudos como los de las figuras mencionadas, las cuales fueron diseñadas específicamente para evaluar el comportamiento de las herramientas geométricas.

A pesar de su potencial, la elevada complejidad de estas rutinas ha hecho que no se hayan implementado. Además, no existe una garantía clara de que en casos como el de la subestación representada en la Figura 5.19 se logre reducir el número de regiones, por ejemplo, de tres a dos. Por otro lado, aunque algunas subestaciones presentan geometrías no convexas, el uso principal de las herramientas de descomposición se orienta al estudio de regiones con exclusiones interiores, como en el ejemplo de la Figura 5.20, donde dichos métodos no aportarían una ventaja significativa.

En consecuencia, se concluye que los métodos actualmente implementados son adecuados y suficientes para satisfacer los requisitos del programa. Aunque no se descarta que, en futuras versiones del programa, se evalúe la viabilidad de implementar métodos más complejos como los patrones X e Y si se evidencia una mejora sustancial en escenarios más exigentes.

5.2.2. Métodos de optimización

A la vista de los resultados obtenidos, se puede concluir que la implementación del algoritmo mixto (basado en un algoritmo genético combinado con el método del gradiente conjugado) supone una mejora significativa respecto al método de fuerza bruta de barrido uniforme de la región de estudio, utilizando N puntos espaciados con un paso t, en los que se calcula el potencial.

En el ejemplo concreto, al buscar estimaciones precisas de los valores máximos y mínimos del potencial, fue necesario utilizar una gran cantidad de puntos, lo que incrementó notablemente el tiempo de ejecución. En contraste, el algoritmo mixto logró una mejora media en los tiempos del 93.98 %. Además, el método clásico no siempre consiguió resultados comparables a los obtenidos mediante el algoritmo genético, lo que refuerza la eficacia de la nueva implementación.

Otro aspecto relevante a considerar en este estudio es que el principal factor que ralentiza el tiempo de ejecución del nuevo algoritmo mixto es la cantidad de regiones en las que se descompone la región de estudio. Para una misma malla y un número máximo de individuos constante en la población del algoritmo genético, el tiempo de ejecución se mantiene relativamente estable.

Cabe destacar que, aunque no se ha incluido en el estudio, la variación en el número de individuos por población afecta de forma significativa al rendimiento de la rutina. Por este motivo, se ha optado por mantener el valor por defecto, ya que ofrecía resultados razonables en todos los casos analizados.

Finalmente, resulta conveniente comparar los resultados obtenidos con los criterios y parámetros definidos por el reglamento para el estudio de las puestas a tierra, tal y como se expone en la Sección 2.1.2.4.

El coeficiente de resistencia k_r, cuyo valor se presenta en la Tabla 5.3, sirve como un parámetro clave para evaluar el comportamiento del sistema de puesta a tierra en suelos monocapa, según lo establecido por el reglamento.

El coeficiente de tensión de contacto k_c, en el reglamento es la diferencia entre k_r y el potencial mínimo en la zona accesible cuando la intensidad de falta es de 1 p.u. No obstante, puede ser útil para el estudio del potencial definir un coeficiente $\hat{k}_c$ que recoja la diferencia entre el potencial máximo y el mínimo obtenidos mediante el proceso de optimización. Este valor para las mallas estudiadas se puede ver en la siguiente tabla:

Variable	MallaSimpleRectangulo.mat	Planta fotovoltaica.mat	Subestacion 1.mat	Subestacion 2.mat	Subestacion 3.mat
$k_c \left[\frac{\Omega}{\Omega m}\right]$	$6.840 \cdot 10^{-3}$	$1.790 \cdot 10^{-3}$	$2.164 \cdot 10^{-3}$	$4.458 \cdot 10^{-3}$	$3.230 \cdot 10^{-3}$
$\hat{k}_c \left[\frac{\Omega}{\Omega m}\right]$	$2.185 \cdot 10^{-1}$	$1.716 \cdot 10^{-3}$	$1.875 \cdot 10^{-3}$	$3.348 \cdot 10^{-3}$	$2.690 \cdot 10^{-3}$

Tabla 5.19: Valor del parámetro k_c y $\hat{k}_c$ para las mallas analizadas en el trabajo fin de grado.

En cuanto al coeficiente de tensión de paso k_p, no se ha realizado un análisis específico en este trabajo. Según la normativa IEEE-80 (véase Figura 2.5), si un diseño cumple con el criterio de tensión de contacto, no es necesario verificar la tensión de paso, salvo en situaciones particulares, como la presencia de aceras equipotenciales.

No se ha considerado en el presente estudio, ya que su inclusión no supondría ninguna dificultad dentro del enfoque de optimización propuesto. Bastaría con modificar la función objetivo y las restricciones del problema para permitir su análisis de forma adecuada. La función potencial se sustituiría por la tensión de paso, definida como la máxima variación de potencial entre un punto y aquellos situados a 1 m.

Capítulo 6

Conclusiones

Se presentan a continuación las conclusiones del proyecto y desarrollos futuros para mejorar la implementación.

6.1. Conclusión

Una vez finalizado el proyecto, y en relación con los objetivos planteados al inicio de este Trabajo Fin de Grado, se puede concluir que estos han sido en gran medida alcanzados.

Se han desarrollado herramientas de geometría computacional con un alto grado de robustez y un funcionamiento exitoso, lo que ha permitido mejorar significativamente el análisis geométrico tanto en la fase de cálculo como en la de optimización.

La resistencia del electrodo de puesta a tierra se ha calculado mediante el parámetro k_r, cuyos valores se recogen en la Tabla 5.3. Además, mediante la rutina `potencialCSM.m`, ha sido posible calcular los potenciales en cualquier punto del espacio.

Otro de los objetivos, la optimización del sistema, se ha logrado de forma satisfactoria, obteniendo mejoras tanto en los niveles de seguridad como en los tiempos de ejecución, con una reducción media en los tiempos de calculo del 93.98 %.

En cuanto a la tensión de contacto, representada por el parámetro $\hat{k}_c$, esta ha sido calculada con éxito y sus resultados se presentan en la Tabla 5.19.

Por último, también se ha desarrollado una herramienta informática con interfaz gráfica (GUI), que facilita notablemente el trabajo y la interacción del usuario con las distintas funcionalidades implementadas.

6.2. Desarrollos futuros

A partir del trabajo realizado se proponen mejoras y desarrollos futuros en tres direcciones: la mejora de la interfaz gráfica, cambios en el cálculo de la región de estudio y el análisis del uso de optimización determinista con una semilla adecuada.

6.2.1. Mejora de la interfaz gráfica

A pesar de haber empleado programación orientada a objetos para encapsular información y facilitar la creación de interfaces, el diseño actual de las clases presenta áreas de mejora que pueden contribuir a solucionar algunos errores y optimizar la composición de los distintos elementos del trabajo. Entre las posibles mejoras se destacan:

- **Permitir retorno entre fases de cálculo y diseño:** Para optimizar el ciclo de trabajo, sería deseable posibilitar que el usuario regrese de la fase de cálculo a la de dibujo de electrodos. De este modo, tras obtener los potenciales, podría ajustar rápidamente el diseño hasta alcanzar los valores deseados sin tener que reiniciar todo el proceso.
- **Refactorización del código:** La estructura actual de archivos y la forma en que se organiza la interfaz gráfica dificultan el mantenimiento y la comprensión de su funcionalidad. Una reorganización del código, así como una mejor coordinación entre las distintas rutinas de la aplicación, permitirían obtener un código más limpio y facilitaría futuras mejoras.
- **Integración de rutinas en las clases:** Algunas rutinas esenciales no se han incorporado de manera nativa en la estructura de clases. Aunque su integración podría aumentar la reusabilidad y modularidad del código, se priorizó el desarrollo y perfeccionamiento de dichas rutinas, relegando en cierta medida una integración más profunda en la arquitectura orientada a objetos.
- **Coordinación de clases y aplicación de patrones de diseño:** Si bien se han implementado patrones como el Composite y se ha aprovechado la delegación mediante callbacks, la estructura actual evidencia la ausencia de patrones clave. En particular, la incorporación del patrón Factoría facilitaría la creación y coordinación de los distintos elementos de la interfaz, aportando mayor flexibilidad y modularidad al sistema.
- **Optimización del diseño visual y de la experiencia de usuario:** La retroalimentación insuficiente, tanto visual como sonora, en la interacción con algunos elementos dificulta la comprensión del estado de la aplicación. Además, mejorar la disposición y coherencia visual de los elementos (por ejemplo, en cuanto a colores, tipografías y estilos) contribuiría a una experiencia de usuario más intuitiva y atractiva. No obstante, al igual que en otros aspectos, se ha preferido dedicar tiempo a los métodos de cálculo que al aspecto de diseño de la aplicación.

6.2.2. Cambios en el cálculo

6.2.2.1. Definición de la función objetivo

Para permitir la optimización de la tensión de paso y el cálculo del coeficiente k_p, basta con definir una función `potencialPaso`, análoga a la función `potencial`, que para cada punto calcule la máxima diferencia de potencial existente a una distancia de 1 metro. De este modo, sería suficiente con utilizar la rutina de optimización habitual y obtener el valor máximo de la función mediante esta nueva definición de `potencialPaso`.

6.2.2.2. Definición de la región de estudio

Actualmente, el cálculo de la región de estudio se realiza mediante input manual o a través del cálculo automático de la frontera. En el caso del input manual, no existe inconveniente para la optimización, ya que es el usuario quien determina qué región le interesa analizar. No obstante, a efectos del cálculo del potencial de contacto, sería conveniente implementar una rutina que expanda dicha frontera hacia el exterior una longitud L. De este modo, se tendría también en cuenta la posible existencia de condiciones desfavorables en el entorno inmediato.

6.2.3. Optimización determinista con una semilla adecuada

Con el objetivo de mejorar la eficiencia de la optimización, se plantea estudiar un método alternativo al uso de algoritmos genéticos para abordar el problema de minimización en la malla. Este enfoque se centra en electrodos de puesta a tierra tipo subestación, en los cuales la geometría de la malla se sitúa en el interior de un lazo cerrado. La propuesta consiste en calcular una base de ciclos mediante procesamiento geométrico, es decir, determinar aquellas submallas cuya unión conforma el electrodo completo, asegurando que la intersección entre dos ciclos cualesquiera sea nula para evitar solapamientos.

Una vez obtenida esta base de ciclos, que constituye la parte compleja del cálculo, se procede a aplicar algoritmos deterministas basados en gradiente a los centroides de cada ciclo. Es importante destacar que la elección del centroide como semilla se fundamenta en una observación heurística. Normalmente, en distintos potenciales estudiados, los óptimos se hallan en el centro o en la frontera del ciclo, aunque no se garantiza que siempre sea el caso. No obstante, emplear el centroide como punto inicial proporciona un punto de partida razonable para la optimización paralela de cada ciclo, lo que podría resultar en una alternativa viable."

En este sentido, se busca determinar si el método determinista ofrece ventajas en términos de rapidez y eficiencia en comparación con el enfoque genético, que, si bien es robusto, a menudo requiere de una significativa cantidad de iteraciones para alcanzar la convergencia.

No obstante, este método presenta desafíos, como la necesidad de un cálculo geométrico previo y la posible adaptación de la malla cuando ésta no cumpla con ciertas características geométricas deseadas. Por ello, se debe evaluar detalladamente si dicha metodología resulta en una mejora significativa en la eficiencia del cálculo.

Apéndice A

Definiciones básicas

1. Algoritmo Genético: método de optimización estocástico que en función de la aptitud (valor en la función objetivo) replica, reproduce y muta a los individuos.
2. Coeficiente de resistencia k_r: magnitud que cuantifica la resistencia de puesta a tierra del electrodo. Puede emplearse para calcular el potencial del electrodo. Es una magnitud unitaria.
3. Coeficiente de tensión de contacto k_c: diferencia de potencial entre el mínimo de potencial y el potencial del electrodo k_r. Es una magnitud unitaria.
4. Coeficiente de tensión de paso k_p: máxima diferencia de potencial entre dos puntos de la instalación situados a 1 m. Es una magnitud unitaria.
5. Coeficiente de tensión de paso de "gigante" $\hat{k}_c$: diferencia entre el potencial máximo y mínimo obtenidos en la optimización. Es una magnitud unitaria.
6. Condiciones de Karush-Kuhn-Tucker (KKT): condiciones empleadas en métodos de optimización con restricciones para garantizar que un punto cumpla con ellas.
7. Condiciones de Wolfe: condiciones aplicadas a métodos de gradiente para garantizar que se toman pasos con un tamaño aceptable (ni muy grandes ni muy pequeños). Así se agiliza la convergencia de los métodos.
8. Conductor o electrodo: se habla de los elementos que conforman una malla de puesta a tierra.
9. Convexo: se habla de toda región de tal forma que escogidos dos puntos cualesquiera contenidos en ella se pueden unir mediante una línea también contenida en la región.
10. Descomposición convexa: consiste en descomponer una poligonal para que todas las subregiones queden delimitadas por poligonales que encierran regiones convexas.
11. Malla comprimida: malla compuesta por electrodos largos, es decir, se admiten intersecciones entre electrodos.
12. Malla descomprimida: malla formada por electrodos cortos, es decir, sin intersecciones entre ellos. Para ello, los electrodos largos se dividen en segmentos más pequeños en cada punto de intersección, formando así electrodos cortos.

13. Frontera de una malla: poligonal que encierra todos los electrodos de una malla de tal manera que todos los segmentos de la poligonal deben coincidir con electrodos.

14. Malla: conjunto de electrodos conectados entre sí.

15. Método de Gradiente Conjugado: método de optimización determinista, aplicado a problemas sin restricciones, que busca el mínimo de una función avanzando en direcciones conjugadas entre sí, definidas mediante la matriz Hessiana.

16. Método de Punto Interior: método de optimización determinista con restricciones. Incorpora las restricciones mediante una función de barrera.

17. Método de simulación de cargas (CSM): método numérico tipo diferencias finitas para resolver, de forma aproximada, las ecuaciones de Maxwell en régimen estacionario.

18. Optimización: proceso mediante el cual se obtienen el máximo y mínimo de una función

19. Optimización determinista: abarca los métodos de optimización clásicos que normalmente se basan en gradientes.

20. Optimización estocástica: abarca los métodos de optimización basados en el uso de elementos aleatorios en su búsqueda.

21. Malla de Puesta a tierra: malla enterrada a través de la cual se drena la corriente de falta hacia la tierra para garantizar la protección de personas y equipos.

22. Punto de concavidad (notch): todo vértice que tiene un ángulo interno mayor a 180°.

23. Punto de Steiner: puntos introducidos durante la resolución al problema como nuevos vértices en los polígonos.

Bibliografía

[1] The MathWorks Inc. Matlab version: 23.2 (r2023b), 2024.

[2] Leslie Lamport et al. The latex project, 2024.

[3] Tikzmaker. `https://tikzmaker.com/editor`, 2024.

[4] Nico Schlömer. matlab2tikz. `https://github.com/matlab2tikz/matlab2tikz`, 2024. Consultado 23 junio, 2024.

[5] Gobierno de España. Instrucción técnica complementaria itc-rat 1. terminología e itc-rat 13. instalaciones de puesta a tierra. `https://www.boe.es/eli/es/rd/2014/05/09/337`, 9 de mayo 2014. I. Disposiciones generales. Ministerio de Industria, Energía y Turismo.

[6] Jorge Moreno Mohino, Pascual Simon Comin, Gabriel Asensio Madrid, Gregorio Denche Castejon, Eduardo Faleiro Usanos, Daniel Garcia Puertas, Pedro Navarro Martinez, and Francisco José Pazos Filguiera. *Sistemas de puesta a tierra en instalaciones de alta tension*. Garceta, 1 edition, 2015.

[7] IEEE Power Engineering Society. Substations Committee and IEEE SA Standards Board. *IEEE guide for safety in AC substation grounding*. Institute Of Electrical And Electronics Engineers, 01 2000.

[8] Bernard Chazelle and David P. Dobkin. Optimal convex decompositions. In Godfried T. TOUSSAINT, editor, *Computational Geometry*, volume 2 of *Machine Intelligence and Pattern Recognition*, pages 63–133. North-Holland, 1985.

[9] J. Mark Keil. Polygon decomposition. In Jörg-Rüdiger Sack and Jorge Urrutia, editors, *Handbook of Computational Geometry*, pages 491–518. North Holland / Elsevier, 2000.

[10] Marco Cavazzuti. *Optimization Methods: From Theory to Design. Scientific and Technological Aspects in Mechanics*. Springer Science & Business Media, 1 edition, 09 2012.

[11] E. Faleiro, G. Asensio, J. Moreno, P. Simón, G. Denche, and D. García. Modelling and simulation of the grounding system of a class of power transmission line towers involving inhomogeneous conductive media. *Electric Power Systems Research*, 136:154–162, 2016.

[12] Eduardo Faleiro, Gabriel Asensio, and Jorge Moreno. An estimate of the uncertainty in the grounding resistance of electrodes buried in two-layered soils with non-flat surface. *Energies*, 10(2), 2017.

[13] Eduardo Faleiro, Gabriel Asensio, Gregorio Denche, and Julissa Moreno. A fast method to compute the grounding resistance of a coated electrode using the coated electrode equivalent radius. *International Journal of Electrical Power & Energy Systems*, 137(3):107879, May 2022. License: CC BY-NC-ND 4.0.

[14] Gabriel Asensio, Eduardo Faleiro, Jorge Moreno, Daniel García, and Gregorio Denche. Joule heating in grounding electrodes under fault conditions: Effects on system potentials and electrode efficiency. Preprint submitted to *Applied Sciences*, 2025, 15, x FOR PEER REVIEW, 2025.

[15] Método unesa - anexo ii: Valores de coeficientes, 2023. Consultado: 2024-09-21.

[16] Pei bai Zhou. Numerical analysis of electromagnetic fields. In *Numerical Analysis of Electromagnetic Fields*, Electric Energy Systems and Engineering Series, pages 215–251. Springer-Verlag Berlin Heidelberg, Berlin, Heidelberg, 1 edition, 1993.

[17] Erling D. Sunde. Earth conduction effects in transmission systems. In *Earth Conduction Effects in Transmission Systems*, The Bell Telephone Laboratories Series, pages 38–98. D. Van Nostrand, New York, 1949. Access-restricted item, contributed by the Internet Archive.

[18] Timothy M. Chan. Optimal output-sensitive convex hull algorithms in two and three dimensions. *Discrete & Computational Geometry*, 16(4):361–368, 1996.

[19] Bart Braden. The surveyor's area formula. *The College Mathematics Journal*, 17(4):326–337, 1986.

[20] E. Polak. Computational methods in optimization: A unified approach. volume 77 of *Mathematics in Science and Engineering*, pages 49–55. Academic Press, New York and London, 1971. Department of Electrical Engineering and Computer Sciences, University of California, Berkeley, California.

[21] Jorge Nocedal and Stephen J. Wright. Numerical optimization. Springer Series in Operations Research and Financial Engineering, pages 333 – 617. Springer, New York, NY, 2 edition, 07 2006. Copyright Information: Springer-Verlag New York 2006.

[22] David E. Goldberg. *Genetic algorithms in search, optimization, and machine learning*. Addison-Wesley, Reading, Mass, reprinted with corrections edition.

[23] MathWorks. Can you certify a solution is global?, 2024. Consultado: 2024-07-11.

[24] MathWorks. Comparison of six solvers, 2024. Consultado: 2024-07-11.

[25] MathWorks. Find minimum of function using genetic algorithm, 2024. Consultado: 2024-07-11.

[26] MathWorks. Genetic algorithm options, 2024. Consultado: 2024-07-11.

[27] MathWorks. How the genetic algorithm works, 2024. Consultado: 2024-07-11.

[28] MathWorks. Isolated global minimum, 2024. Consultado: 2024-07-11.

[29] MathWorks. Refine start points, 2024. Consultado: 2024-07-11.

[30] MathWorks. boundary, 2025. Consultado: 2025-04-24.

[31] MathWorks. convhull, 2025. Consultado: 2025-04-24.