FACULTADE DE MATEMÁTICAS

Traballo Fin de Grao

Métodos numéricos en optimización

Cheyenne Amoroso Sanmiguel

2019/2020

UNIVERSIDADE DE SANTIAGO DE COMPOSTELA

GRAO DE MATEMÁTICAS

Traballo Fin de Grao

Métodos numéricos en optimización

Cheyenne Amoroso Sanmiguel

Julio 2020

UNIVERSIDADE DE SANTIAGO DE COMPOSTELA

Trabajo propuesto

Área de Coñecemento: Matemática Aplicada
Título: Métodos numéricos en optimización
Breve descrición do contido
Implementación de métodos numéricos de optimización con y sin restricciones utilizando Python. Comparación con métodos de librerías de código libre.
Recomendacións
Buenas capacidades de programación.

Índice general

Resumen

En el presente Trabajo de Fin de Grado titulado *'Métodos numéricos en optimización'* se trata el problema de ajuste no lineal. Para ello, se describen una serie de métodos numéricos que permiten la resolución de dicho problema. Entre los métodos tratados se encuentran el método de Gauss-Newton, el método de Levenberg-Marquardt y algunos métodos quasi-Newton. Además, se analiza el desempeño de los métodos aplicándolos sobre una base de datos de problemas Benchmark de distintos grados de dificultad, para lo cual los cálculos y la programación se llevan a cabo a través del lenguaje de programación interpretado *Python*. Por último, y para completar la información, como anexo, se incluye también el código empleado para cada uno de los métodos, siendo algunos de implementación propia y otros procedentes de bibliotecas libres propias de *Python*.

Abstract

This Final Degree Project titled *'Numerical methods in optimization'* it is about nonlinear least squares problems. A few of numerical methods that allow solving nonlinear least squares problems are described for it. Methods discussed include the Gauss-Newton method, the Levenberg-Marquardt method, and some quasi-Newton methods. The project shows the behaviour of the algorithms applying them to a database of Benchmark problems of varying levels of difficulty. The interpreted language *Python* has been used to perform the algorithms. The code used for each of the methods is also included to supplement the description, some of them being self-implemented and other belonging to free libraries of *Python*.

Introducción

La optimización es una rama relativamente reciente de la matemática aplicada, la matemática computacional y la investigación operativa que tiene amplias aplicaciones, dado que una gran variedad de problemas en todas las áreas de las matemáticas, ciencias aplicadas, ingeniería, economía o medicina se pueden plantear en términos de optimización. Los modelos matemáticos se desarrollan, generalmente, para analizar y comprender fenómenos complejos. La mayoría de los procedimientos de toma de decisiones implican la solución explícita de un problema de optimización para poder realizar así la mejor elección. La optimización conlleva el estudio de las condiciones óptimas de los problemas, la construcción de modelos, la determinación del método algorítmico de solución, el establecimiento de la teoría de convergencia de los algoritmos y experimentos numéricos con problemas típicos y problemas de la vida real. Los problemas de optimización a menudo surgen como subproblemas críticos dentro de otros procesos numéricos.

Pierre de Fermat y Joseph Louis Lagrange encontraron fórmulas basadas en el cálculo para identificar valores óptimos. A su vez Isaac Newton y Carl Friedrich Gauss propusieron métodos iterativos para aproximar el óptimo. Sin embargo, no fue hasta finales de la década de los cuarenta del siglo pasado, cuando George B. Dantzig publicó el algoritmo símplex, que la optimización empezó a tratarse como una rama independiente. El término programación lineal para referirse a ciertos problemas de optimización se debe a Dantzig, aunque gran parte de la teoría ya había sido introducida por Leonid Kantorovich en 1939. En el mismo año en el que Dantzig publicó su algoritmo, John von Neumann desarrolló la teoría de la dualidad. Después de la década de 1950, cuando los métodos de gradiente conjugado y quasi-Newton se presentaron, la programación no lineal se desarrolló enormemente. Actualmente, existen métodos de optimización modernos con los cuales es posible resolver problemas más complicados. De esta forma, los métodos de optimización se han convertido en una herramienta indispensable.

Un problema general de optimización comienza con un conjunto de variables o pa-

rámetros independientes pudiendo incluir condiciones o restricciones que definen valores aceptables de dichos parámetros y con una función objetivo o función coste que depende en cierta medida de las variables. Tales restricciones se denominan las acotaciones del problema. La solución del problema es un conjunto de valores admisibles de las variables para las cuales la función objetivo asume un valor "óptimo".

Generalmente los problemas de optimización se plantean en forma estándar para así facilitar el procedimiento de la creación de métodos con el objetivo de hallar una solución. Esta forma estándar surge de la naturaleza de la mayoría de los problemas de optimización. Se escribe bajo la forma siguiente: dado un conjunto abierto $\Omega \subseteq \mathbb{R}^n$, una función $f : \Omega \subseteq \mathbb{R}^n \longrightarrow \mathbb{R}$ y un subconjunto $U \subseteq \Omega$, encontrar un elemento $\boldsymbol{u} \in U$ que minimice la función f en U, es decir,

$$\min_{\boldsymbol{\omega} \in U} \quad f(\boldsymbol{\omega}) \tag{1}$$

donde $\boldsymbol{\omega} \in \mathbb{R}^n$ es una variable de decisión, $f(\boldsymbol{\omega})$ es la función objetivo y $U \subset \mathbb{R}^n$ es el conjunto de restricciones. El problema se dirá sin restricciones si el conjunto U es abierto, en cuyo caso tomaremos $U = \Omega$ para simplificar la notación. En particular, si $\Omega = \mathbb{R}^n$ diremos que estamos ante un problema de optimización sin restricciones de la forma siguiente:

$$\min_{\boldsymbol{\omega} \in \mathbb{R}^n} \quad f(\boldsymbol{\omega}). \tag{2}$$

Un problema de optimización con restricciones sigue la siguiente forma general:

$$\begin{aligned} \min_{\boldsymbol{\omega} \in \mathbb{R}^n} \quad & f(\boldsymbol{\omega}) \\ \text{s.t} \quad & c_i(\boldsymbol{\omega}) = 0, \quad i \in E, \\ & d_i(\boldsymbol{\omega}) \leq 0, \quad i \in I. \end{aligned} \tag{3}$$

Se cumple que la función objetivo y las funciones de restricción son funciones escalares de valor real.

Al encontrarnos con un problema de optimización es conveniente determinar ciertas características del problema para así poder resolverlo de la forma más eficiente. Las distinciones más obvias que podemos realizar son aquellas que tratan las diferencias en las características matemáticas de las funciones objetivo y de restricción. Un tipo de problema que se puede enmarcar es aquel en el que la f es de la forma $f(\boldsymbol{\omega}) = \sum_{i=1}^{m} [r_i(\boldsymbol{\omega})]^2$, donde $\boldsymbol{r} : \mathbb{R}^n \to \mathbb{R}^m$ es una función, en general, no lineal. Dicho problema se denomina problema de suma de cuadrados de funciones no lineales (*nonlinear least-squares problems*) y es el

que trataremos en el marco de este TFG. Estos siguen la fórmula general siguiente:

$$\min_{\boldsymbol{\omega}\in\mathbb{R}^n} f(\boldsymbol{\omega}) = \min_{\boldsymbol{\omega}\in\mathbb{R}^n} \frac{1}{2}\|\boldsymbol{r}(\boldsymbol{\omega})\|_2^2 = \min_{\boldsymbol{\omega}\in\mathbb{R}^n} \frac{1}{2}\boldsymbol{r}(\boldsymbol{\omega})^T\boldsymbol{r}(\boldsymbol{\omega}) = \min_{\boldsymbol{\omega}\in\mathbb{R}^n} \frac{1}{2}\sum_{i=1}^{m}[r_i(\boldsymbol{\omega})]^2, \quad m \geq n. \quad (4)$$

Cuando se trata de un problema sin restricciones, se puede interpretar el problema como la resolución de un sistema de m ecuaciones no lineales,

$$r_i(\boldsymbol{\omega}) = 0, \qquad i = 1, ..., m,$$

donde $r_i(\boldsymbol{\omega})$ denota la función residual. Cuando $m > n$ diremos que el sistema está sobredeterminado mientras que si $m = n$ diremos que se trata de un sistema compatible determinado.

Una de las áreas más importantes donde podemos encontrarnos este tipo de problemas es en el campo de la estadística; más concretamente, en el ámbito de la regresión. El objetivo de la regresión es explicar la dependencia de una variable aleatoria univariante y, a la cual llamaremos variable dependiente o respuesta, respecto a una variable aleatoria, posiblemente multivariante, x, que denominaremos variable independiente o explicativa. Sea $\{(\boldsymbol{x_i}, y_i)\}_{i=1}^m$ una muestra donde $\boldsymbol{x_i} \in \mathbb{R}^p$, $y_i \in \mathbb{R}$ son los valores de las variable explicativa y respuesta, respectivamente, correspondientes al i-ésimo individuo. Sea $\boldsymbol{\theta} \in \mathbb{R}^n$ el vector de parámetros desconocidos, consideraremos el modelo de regresión no lineal que vendrá expresado como sigue:

$$y_i = h(\boldsymbol{x_i}, \boldsymbol{\theta}) + \varepsilon_i, \quad i = 1, ..., m.$$

Pensaremos en el término ε_i como los errores asociados a cada individuo, esto es, la distorsión en el valor esperado y_i que difiere de la respuesta $h(\boldsymbol{x}_i, \boldsymbol{\theta})$. Supondremos los errores distribuidos normalmente con media 0 y varianza constante desconocida, σ^2, que se estimará a partir de los datos.

El objetivo en este trabajo es estimar el vector de parámetros $\boldsymbol{\theta} \in \mathbb{R}^n$, para lo cual consideraremos como estimador aquel que minimice la suma residual de cuadrados (*Residual Sum of Squares*), RSS. Denotaremos dicho estimador por $\hat{\boldsymbol{\theta}}$. El problema de minimizar la RSS es equivalente a minimizar la función

$$\frac{1}{2}\sum_{i=1}^{m}(y_i - h(\boldsymbol{x_i}, \boldsymbol{\theta}))^2.$$

Debido a la no linealidad de h estamos ante un problema de optimización no lineal. Por consiguiente, obtenemos que nuestro problema sigue la fórmula general (4) de la siguiente

manera:

$$\min_{\boldsymbol{\theta}\in\mathbb{R}^n} f(\boldsymbol{\theta}) = \min_{\boldsymbol{\theta}\in\mathbb{R}^n} \frac{1}{2}\boldsymbol{r}(\boldsymbol{\theta})^T\boldsymbol{r}(\boldsymbol{\theta}) = \min_{\boldsymbol{\theta}\in\mathbb{R}^n} \frac{1}{2}\sum_{i=1}^{m}[r_i(\boldsymbol{\theta})]^2, \quad m \geq n,$$

siendo $r_i(\boldsymbol{\theta}) = y_i - h(\boldsymbol{x_i}, \boldsymbol{\theta})$, $r_i : \mathbb{R}^n \to \mathbb{R}$.

La interpretación geométrica que podemos dar a este tipo de problemas es encontrar un punto en la superficie $\boldsymbol{z} = \boldsymbol{r}(\boldsymbol{\omega})$ en $\mathbb{R}^m$ lo más próximo posible al origen. En el caso de la regresión consideramos la superficie $\boldsymbol{z} = (h(\boldsymbol{x_1}, \boldsymbol{\theta}), ..., h(\boldsymbol{x_m}, \boldsymbol{\theta})) \in \mathbb{R}^m$.

El trabajo se estructura en dos capítulos como sigue: en el Capítulo 1 estudiaremos los aspectos más relevantes de algoritmos utilizados en el problema general de estimación de parámetros. Para ello nos guiaremos principalmente por Sun y Yuan (2006), Bjorck (1996) y Gill et al. (1981). En el Capítulo 2 se mostrará el desempeño de los algoritmos sobre una base de datos tomada de Shen et al. (2013), realizándose una comparación de los distintos métodos explicados y las distintas implementaciones de cada uno de ellos. Los cálculos y la programación se llevarán a cabo a través de *Python* mediante las bibliotecas libres Virtanen et al. (2020), van der Walt et al. (2011) y Hunter (2007). Se incluirán además tres apéndices, en el primero de ellos encontraremos una serie de conceptos básicos para una mejor comprensión de lo expuesto en el primer capítulo. En el segundo se mostrarán algunos métodos más generales para la resolución del problema expuesto. Finalmente, se incluirá en un tercer apéndice el código implementado por la autora de este trabajo para aquellos métodos que no fuera posible encontrar programado en ninguna biblioteca de *Python*.

Capítulo 1

Métodos de numéricos

En este capítulo estudiaremos diferentes métodos numéricos en optimización para problemas de suma de cuadrados de funciones no lineales. Los algoritmos para la optimización no lineal sin restricciones con múltiples variables se pueden clasificar en los siguientes tres grupos:

1. Búsqueda sin el uso de derivadas. Ejemplos de esto son el método simplex, el algoritmo de Hooke y Jeeves, el método de Rosenbrock y el método de las direcciones conjugadas.

2. Búsqueda usando información de la derivada primera. En este caso tenemos el método de máximo descenso, el método del gradiente conjugado y los métodos quasi-Newton.

3. Búsqueda usando información de la derivada segunda como, por ejemplo, el método de Newton.

A lo largo de este capítulo asumiremos que las componentes $r_i(\boldsymbol{\omega})$ son de clase dos. Denotaremos al jacobiano de $\boldsymbol{r}(\boldsymbol{\omega})$ por $J(\boldsymbol{\omega}) \in \mathbb{R}^{m\times n}$ donde

$$J(\boldsymbol{\omega})_{ij} = \frac{\partial r_i(\boldsymbol{\omega})}{\partial \omega_j} \tag{1.1}$$

y a las matrices Hessianas de $r_i(\boldsymbol{\omega})$ son $G_i(\boldsymbol{\omega}) \in \mathbb{R}^{n\times n}$, donde

$$G_i(\boldsymbol{\omega})_{jk} = \frac{\partial^2 r_i(\boldsymbol{\omega})}{\partial \omega_j \partial \omega_k}, \quad i = 1, ..., m. \tag{1.2}$$

Por otra parte, el gradiente de $f(\boldsymbol{\omega})$ se denotará con

$$\nabla f(\boldsymbol{\omega}) = \sum_{i=1}^{m} r_i(\boldsymbol{\omega}) \nabla r_i(\boldsymbol{\omega}) = J(\boldsymbol{\omega})^T \boldsymbol{r}(\boldsymbol{\omega}), \tag{1.3}$$

y su Hessiana con

$$\nabla^2 f(\boldsymbol{\omega}) = \sum_{i=1}^{m} (\nabla r_i(\boldsymbol{\omega}) \nabla r_i(\boldsymbol{\omega})^T + r_i(\boldsymbol{\omega}) \nabla^2 r_i(\boldsymbol{\omega})) = J(\boldsymbol{\omega})^T J(\boldsymbol{\omega}) + S(\boldsymbol{\omega}), \tag{1.4}$$

donde

$$S(\omega) = \sum_{i=1}^{m} r_i(\boldsymbol{\omega}) \nabla^2 r_i(\boldsymbol{\omega}). \tag{1.5}$$

Existen dos formas de ver el problema (4). La primera de ellas, como ya mencionábamos en la introducción, es interpretarlo como la resolución de un sistema sobredeterminado de m ecuaciones no lineales

$$r_i(\boldsymbol{\omega}) = 0, \quad i = 1, ..., m.$$

En este caso podemos aproximar $\boldsymbol{r(\omega)}$ por un modelo lineal centrado en un punto dado, $\boldsymbol{\omega}_k$,

$$\tilde{\boldsymbol{r}}_{\boldsymbol{k}}(\boldsymbol{\omega}) = \boldsymbol{r}(\boldsymbol{\omega}_k) + J(\boldsymbol{\omega}_k)(\boldsymbol{\omega} - \boldsymbol{\omega}_k). \tag{1.6}$$

Cuando $m > n$ tenemos el sistema sobredeterminado $\boldsymbol{r}_k(\boldsymbol{\omega}) = 0$, el cual podemos resolver como un problema de mínimos cuadrados lineal para así aproximar una solución.

La segunda forma de ver este problema es tratarlo como un caso especial de optimización en $\mathbb{R}^n$. Para ello empleamos un modelo cuadrático en torno a $\boldsymbol{\omega}_k$

$$q_k(\boldsymbol{\omega}) = f(\boldsymbol{\omega}_k) + \nabla f(\boldsymbol{\omega}_k)^T (\boldsymbol{\omega} - \boldsymbol{\omega}_k) + \frac{1}{2} (\boldsymbol{\omega} - \boldsymbol{\omega}_k)^T \nabla^2 f(\boldsymbol{\omega}_k)(\boldsymbol{\omega} - \boldsymbol{\omega}_k). \tag{1.7}$$

Haciendo uso de las ecuaciones (1.3) y (1.4) llegamos a la siguiente expresión

$$q_k(\boldsymbol{\omega}) = \frac{1}{2} \boldsymbol{r}(\boldsymbol{\omega}_k)^T \boldsymbol{r}(\boldsymbol{\omega}_k) + (J(\boldsymbol{\omega}_k)^T \boldsymbol{r}(\boldsymbol{\omega}_k))^T (\boldsymbol{\omega} - \boldsymbol{\omega}_k) + \frac{1}{2} (\boldsymbol{\omega} - \boldsymbol{\omega}_k)^T (J(\boldsymbol{\omega}_k)^T J(\boldsymbol{\omega}_k) + S(\boldsymbol{\omega}_k))(\boldsymbol{\omega} - \boldsymbol{\omega}_k). \tag{1.8}$$

Empleando el método de Newton, el mínimo de la función q_k se puede calcular de la forma

$$\boldsymbol{\omega}_{k+1} = \boldsymbol{\omega}_k - (J(\boldsymbol{\omega}_k)^T J(\boldsymbol{\omega}_k) + S(\boldsymbol{\omega}_k))^{-1} J(\boldsymbol{\omega}_k)^T \boldsymbol{r}(\boldsymbol{\omega}_k), \tag{1.9}$$

el cual converge localmente de forma cuadrática.

El principal problema de este método es que al calcular las mn^2 derivadas segundas necesarias para obtener el término $S(\boldsymbol{\omega}_k)$ el coste computacional puede ser relativamente alto. Por lo tanto, puede resultar útil omitir dicho término. De (1.5) deducimos que cuando $r_i(\boldsymbol{\omega})$, $i = 1, ..., n$, es próximo a cero o es levemente no lineal, en cuyo caso $\nabla^2 r_i(\boldsymbol{\omega})$ se aproxima a cero, el término $S(\boldsymbol{\omega})$ es pequeño y puede omitirse. A partir de este razonamiento se siguen los métodos de Gauss-Newton y Levenberg-Marquardt que trataremos más

en detalle a continuación. En este capítulo trataremos también los métodos quasi-Newton que sí utilizan información de la derivada segunda, aunque no directamente y sí mediante aproximaciones recursivas.

1.1. Método de Gauss-Newton

1.1.1. Descripción y propiedades

El método Gauss-Newton puede pensarse de dos maneras distintas: la primera de ellas es pensar que se obtiene omitiendo el término de segundo grado, $S(\omega)$, en el modelo cuadrático (1.8). De esta forma llegamos a la siguiente aproximación

$$q_k^G(\boldsymbol{\omega}) = \frac{1}{2}\boldsymbol{r}(\boldsymbol{\omega}_k)^T\boldsymbol{r}(\boldsymbol{\omega}_k) + (J(\boldsymbol{\omega}_k)^T\boldsymbol{r}(\boldsymbol{\omega}_k))^T(\boldsymbol{\omega}-\boldsymbol{\omega}_k) + \frac{1}{2}(\boldsymbol{\omega}-\boldsymbol{\omega}_k)^T J(\boldsymbol{\omega}_k)^T J(\boldsymbol{\omega}_k)(\boldsymbol{\omega}-\boldsymbol{\omega}_k), \tag{1.10}$$

y aplicando el método de Newton a la condición necesaria de mínimo se obtiene la solución mediante

$$\boldsymbol{\omega}_{k+1} = \boldsymbol{\omega}_k - (J(\boldsymbol{\omega}_k)^T J(\boldsymbol{\omega}_k))^{-1} J(\boldsymbol{\omega}_k)^T \boldsymbol{r}(\boldsymbol{\omega}_k). \tag{1.11}$$

Cabe destacar que para que esto tenga sentido es necesario que la matriz $J(\boldsymbol{\omega}_k)$ sea de rango completo.

Como hemos explicado anteriormente, este razonamiento será útil cuando $r_i(\boldsymbol{\omega}),\ i = 1, ..., n$, sea próximo a cero o levemente no lineal. Si nos encontramos en este caso es esperable que el comportamiento del método sea similar al método de Newton dado en (1.9). En particular, en el caso de que nos encontremos con un problema consistente tal que $r(\tilde{\omega}) = 0$, donde $\tilde{\boldsymbol{\omega}}$ es solución, la velocidad de convergencia local será idéntica con ambos métodos. Sin embargo, en el caso de residuos grandes la velocidad de convergencia local será inferior en el método de Gauss-Newton.

La segunda forma de pensar este método es como una serie de aproximaciones de $\boldsymbol{r}(\boldsymbol{\omega})$ de la forma (1.6). Si denotamos por $\boldsymbol{s}_k$ a la corrección de la aproximación $\boldsymbol{\omega}_k$, dicha corrección se obtendrá al resolver el problema de mínimos cuadrados

$$\min_{\boldsymbol{s}\in\mathbb{R}^n} \quad \frac{1}{2}||\boldsymbol{r}(\boldsymbol{\omega}_k) + J(\boldsymbol{\omega}_k)\boldsymbol{s}||_2^2. \tag{1.12}$$

De esta forma, la nueva aproximación será $\boldsymbol{\omega}_{k+1} = \boldsymbol{\omega}_k + \boldsymbol{s}_k$.

En el caso de que la matriz $J(\boldsymbol{\omega}_k)$ tenga rango completo la solución al problema lineal de mínimos cuadrados es única y vemos que coincide con la obtenida mediante el razonamiento

anterior

$$\boldsymbol{s}_k = -(J(\boldsymbol{\omega}_k)^T J(\boldsymbol{\omega}_k))^{-1} J(\boldsymbol{\omega}_k)^T \boldsymbol{r}(\boldsymbol{\omega}_k). \tag{1.13}$$

Tenemos que $\nabla f(\boldsymbol{\omega}_k)$ es distinto de cero, por lo que deducimos que $\boldsymbol{s}_k$ se trata de una dirección de descenso ya que

$$\boldsymbol{s}_k^T \nabla f(\boldsymbol{\omega}_k) = \boldsymbol{s}_k^T J(\boldsymbol{\omega}_k)^T \boldsymbol{r}(\boldsymbol{\omega}_k) = -\boldsymbol{s}_k^T J(\boldsymbol{\omega}_k)^T J(\boldsymbol{\omega}_k) \boldsymbol{s}_k \leq 0, \tag{1.14}$$

donde la desigualdad será estricta si no se cumple que $J(\boldsymbol{\omega}_k)\boldsymbol{s}_k = 0$. Esta última igualdad es equivalente a $J(\boldsymbol{\omega}_k)^T \boldsymbol{r}(\boldsymbol{\omega}_k) = \nabla f(\boldsymbol{\omega}_k) = \mathbf{0}$.

En el caso de que $J(\boldsymbol{\omega}_k)$ esté mal condicionado o sea singular también podremos calcular $\boldsymbol{s}_k$ usando descomposiciones de $J(\boldsymbol{\omega}_k)$ como, por ejemplo, la factorización QR o la descomposición en valores singulares o SVD (*Singular Value Decomposition*). Si $J(\boldsymbol{\omega}_k)$ no es de rango completo podemos tomar $\boldsymbol{s}_k$ como

$$\boldsymbol{s}_k = -J(\boldsymbol{\omega}_k)^I \boldsymbol{r}(\boldsymbol{\omega}_k), \tag{1.15}$$

donde $J(\boldsymbol{\omega}_k)^I$ denota la matriz pseudo-inversa de $J(\boldsymbol{\omega}_k)$.

A continuación analizaremos la convergencia del método. Para ello enunciaremos tres teoremas que nos permitirán concluir lo siguiente:

1. Si $S(\tilde{\boldsymbol{\omega}}) = 0$, el método tiene convergencia cuadrática.
2. Si $S(\tilde{\boldsymbol{\omega}})$ es pequeño comparado con $J(\tilde{\boldsymbol{\omega}})^T J(\tilde{\boldsymbol{\omega}})$ el método es localmente Q-linealmente convergente.
3. Si $S(\tilde{\boldsymbol{\omega}})$ es demasiado grande el método no convergerá.

Teorema 1.1.1. *Sea $f : \mathbb{R}^n \longrightarrow \mathbb{R}, \quad f \in \mathcal{C}^2$. Supongamos que $\tilde{\boldsymbol{\omega}}$ es el mínimo local del problema (4) y que $J(\tilde{\boldsymbol{\omega}})^T J(\tilde{\boldsymbol{\omega}})$ es definida positiva. Supongamos además que la sucesión $\{\boldsymbol{\omega}_k\}$ generada por el Algoritmo 1.1.5 converge a $\tilde{\boldsymbol{\omega}}$. Entonces, si $\nabla^2 f(\boldsymbol{\omega})$ y $(J(\boldsymbol{\omega})^T J(\boldsymbol{\omega}))^{-1}$ son Lipschitzianas en un entorno de $\tilde{\boldsymbol{\omega}}$, se cumple*

$$||\boldsymbol{\omega}_{k+1} - \tilde{\boldsymbol{\omega}}||_2 \leq ||(J(\tilde{\boldsymbol{\omega}})^T J(\tilde{\boldsymbol{\omega}}))^{-1}||_2 \; ||S(\tilde{\boldsymbol{\omega}})||_2 \; ||\boldsymbol{\omega}_k - \tilde{\boldsymbol{\omega}}||_2 + \mathcal{O}(||\boldsymbol{\omega}_k - \tilde{\boldsymbol{\omega}}||_2^2) \tag{1.16}$$

Demostración. Ver Sun y Yuan (2006) (p. 357-358).

□

Teorema 1.1.2. *Sea $f : \Omega \subset \mathbb{R}^n \longrightarrow \mathbb{R}, \quad f \in \mathcal{C}^2(\Omega)$ donde Ω es un conjunto convexo y abierto. Sea $J(\boldsymbol{\omega})$ de Lipschitz continua en Ω, es decir,*

$$||J(\boldsymbol{u}) - J(\boldsymbol{v})||_2 \leq \gamma||\boldsymbol{u} - \boldsymbol{v}||_2, \quad \forall \boldsymbol{u}, \boldsymbol{v} \in \Omega, \tag{1.17}$$

y $||J(\boldsymbol{\omega})||_2 \leq \alpha$, $\forall \boldsymbol{\omega} \in \Omega$. Supongamos que existen $\tilde{\boldsymbol{\omega}} \in \Omega$ y $\lambda, \sigma \geq 0$ tal que, si $J(\tilde{\boldsymbol{\omega}})^T \boldsymbol{r}(\tilde{\boldsymbol{\omega}}) = 0$, λ es el menor valor propio de $J(\tilde{\boldsymbol{\omega}})^T J(\tilde{\boldsymbol{\omega}})$ y

$$||(J(\boldsymbol{\omega}) - J(\tilde{\boldsymbol{\omega}}))^T \boldsymbol{r}(\tilde{\boldsymbol{\omega}})||_2 \leq \sigma||\boldsymbol{\omega} - \tilde{\boldsymbol{\omega}}||_2, \quad \forall \boldsymbol{\omega} \in \Omega. \tag{1.18}$$

Si $\sigma < \lambda$, entonces, para cualquier $c \in (1, \frac{\lambda}{\sigma})$, existe $\varepsilon > 0$ tal que $\forall \boldsymbol{\omega}_0 \in N(\tilde{\boldsymbol{\omega}}, \varepsilon)$ la secuencia generada por el método de Gauss-Newton mediante el Algoritmo 1.1.5 está bien definida, converge a $\tilde{\boldsymbol{\omega}}$ y satisface

$$\begin{aligned} ||\boldsymbol{\omega}_{k+1} - \tilde{\boldsymbol{\omega}}||_2 &\leq \frac{c\sigma}{\lambda}||\boldsymbol{\omega}_k - \tilde{\boldsymbol{\omega}}||_2 + \frac{c\alpha\gamma}{2\lambda}||\boldsymbol{\omega}_k - \tilde{\boldsymbol{\omega}}||_2^2 \\ ||\boldsymbol{\omega}_{k+1} - \tilde{\boldsymbol{\omega}}||_2 &\leq \frac{c\sigma + \lambda}{2}\lambda||\boldsymbol{\omega}_k - \tilde{\boldsymbol{\omega}}||_2 < ||\boldsymbol{\omega}_k - \tilde{\boldsymbol{\omega}}||_2. \end{aligned} \tag{1.19}$$

Demostración. Ver Sun y Yuan (2006) (p. 358-359).

□

Teorema 1.1.3. *Consideremos que las suposiciones del Teorema 1.1.1 o las del Teorema 1.19 se verifican. Si $\boldsymbol{r}(\tilde{\boldsymbol{\omega}}) = 0$, entonces existe $\varepsilon > 0$ tal que para cualquier $\boldsymbol{\omega}_0 \in N(\tilde{\boldsymbol{\omega}}, \varepsilon)$ la sucesión $\{\boldsymbol{\omega}_k\}$ generada por el método de Gauss-Newton converge de forma cuadrática a $\tilde{\boldsymbol{\omega}}$.*

Demostración. Ver Sun y Yuan (2006) (p. 360).

□

El siguiente ejemplo (Fletcher, 1980, pág. 94) muestra que el método de Gauss-Newton funciona bien para problemas con residuos pequeños. Sin embargo, cuando sean grandes o los problemas sean claramente no lineales el método no será localmente convergente.

Ejemplo 1.1.4. Consideramos el problema con m=2 y n=1 dado por

$$r_1(\boldsymbol{\omega}) = \boldsymbol{\omega} + 1, \qquad r_2(\boldsymbol{\omega}) = \lambda\boldsymbol{\omega}^2 + \boldsymbol{\omega} + 1$$

donde λ es un parámetro cualquiera. Tenemos el siguiente problema escrito de forma estándar

$$\min_{\boldsymbol{\omega} \in \mathbb{R}^n} f(\boldsymbol{\omega}) = \min_{\boldsymbol{\omega} \in \mathbb{R}^n} \frac{1}{2} \sum_{i=1}^{2} r_i(\boldsymbol{\omega})^2 = \min_{\boldsymbol{\omega} \in \mathbb{R}^n} \frac{1}{2}[(\boldsymbol{\omega} + 1)^2 + (\lambda\boldsymbol{\omega}^2 + \boldsymbol{\omega} + 1)^2],$$

de manera que $\tilde{\boldsymbol{\omega}} = 0$. Llegamos a que, en este caso particular,

$$\boldsymbol{\omega}_{k+1} = \frac{2\lambda^2\boldsymbol{\omega}_k^3 + \lambda\boldsymbol{\omega}_k^2 + 2\lambda\boldsymbol{\omega}_k}{1 + (2\lambda\boldsymbol{\omega}_k + 1)^2}.$$

Observamos que si $\lambda = 0$ entonces $\boldsymbol{\omega}_1 = 0 = \tilde{\boldsymbol{\omega}}$, lo que muestra que, cuando el problema es lineal, el método de Gauss-Newton lo resuelve en una única iteración. En el caso de que $\lambda \neq 0$

$$\boldsymbol{\omega}_{k+1} = \lambda\boldsymbol{\omega}_k + \mathcal{O}(||\boldsymbol{\omega}_k||_2^2).$$

Así, cuando λ es suficientemente pequeño la convergencia del método será lineal, mientras que si $|\lambda| > 1$ el método no será convergente.

1.1.2. Implementación del método de Gauss-Newton

El método presentado en la sección anterior no está, bajo mi conocimiento, programado en ninguna biblioteca libre de Python, por lo que ha sido implementado por la autora de este trabajo. El código utilizado podemos encontrarlo en el apéndice C.1. El algoritmo programado, basado en la descripción dada del método de Gauss-Newton, es el siguiente:

Algoritmo 1.1.5. *(Gauss-Newton)*
Paso 0. Dados $\boldsymbol{\omega}_0$, $\varepsilon > 0$, $k := 0$.
Paso 1. Si $||\nabla f(\boldsymbol{\omega}_k)||_2 \leq \varepsilon$, *stop.*
Paso 2. Cálculo de $\boldsymbol{s}_k$ *resolviendo*

$$J(\boldsymbol{\omega}_k)^T J(\boldsymbol{\omega}_k)\boldsymbol{s}_k = -J(\boldsymbol{\omega}_k)^T \boldsymbol{r}(\boldsymbol{\omega}_k).$$

Paso 3. $\boldsymbol{\omega}_{k+1} = \boldsymbol{\omega}_k + \boldsymbol{s}_k$, $k := k + 1$. *Volver al Paso 1.* □

Como ya hemos comentado, cabe la posibilidad de que $J(\boldsymbol{\omega}_k)$ no sea una matriz de rango completo. Como solución se propone calcular $\boldsymbol{s}_k$ como $\boldsymbol{s}_k = -J(\boldsymbol{\omega}_k)^I \boldsymbol{r}(\boldsymbol{\omega}_k)$ donde $J(\boldsymbol{\omega}_k)^I$ denota la matriz pseudo-inversa de $J(\boldsymbol{\omega}_k)$.

1.2. Método de Gauss-Newton amortiguado

Según los problemas observados en el ejemplo mostrado en la sección anterior, en la práctica se utiliza un método de Gauss-newton con búsqueda de línea o también llamado método de Gauss-Newton amortiguado (*damped Gauss-Newton*). Sea α_k un escalar y $\boldsymbol{s}_k$ la solución de 1.12, que denominaremos dirección de Gauss-Newton, el método es como sigue

$$\boldsymbol{\omega}_{k+1} = \boldsymbol{\omega}_k + \alpha_k \boldsymbol{s}_k. \tag{1.20}$$

Se verifica que $\boldsymbol{s}_k$ se mantiene invariante ante transformaciones lineales sobre la variable independiente $\boldsymbol{\omega}$ y, como ya vimos anteriormente, es una dirección de descenso.

Para que esta modificación del método sea un algoritmo viable tenemos que tener cuidado con la elección de α_k. Generalmente existen dos maneras que facilitan la elección del escalar:

1. Tomar α_k como el mayor número de la secuencia $1, \frac{1}{2}, \frac{1}{4}, ...$ tal que se cumpla la desigualdad
$$||\boldsymbol{r}(\boldsymbol{\omega}_k)||_2^2 - ||\boldsymbol{r}(\boldsymbol{\omega}_k + \alpha_k \boldsymbol{s}_k)||_2^2 \geq \frac{1}{2}\alpha_k ||J(\boldsymbol{\omega}_k)\boldsymbol{s}_k||_2^2 \tag{1.21}$$
2. Tomar α_k como la solución de
$$\min_{\alpha} ||\boldsymbol{r}(\boldsymbol{\omega}_k + \alpha_k)||_2^2 \tag{1.22}$$

Observación 1.2.1. El primer método explicado para la búsqueda de α_k se trata básicamente de la regla de búsqueda de paso Armijo-Goldstein [1] , mientras que el segundo es el cálculo del paso óptimo.

Tomando de forma adecuada α_k tenemos que este método siempre toma pasos de descenso, por lo tanto es localmente convergente. De hecho, se trata de un método globalmente convergente en la mayoría de los casos pero continúa siendo muy lento cuando el residuo es grande o los problemas son claramente no lineales.

1.2.1. Implementación del método de Gauss-Newton amortiguado

Al igual que ocurría con el método de Gauss-Newton este método no está programado en ninguna biblioteca libre de Python, por lo que ha sido implementado por la autora de este trabajo. En el apéndice C.2 podemos encontrar el código utilizado. El algoritmo programado, basado en la descripción dada, es el siguiente:

Algoritmo 1.2.2. *(Gauss-Newton amortiguado)*
Paso 0. Dados $\boldsymbol{\omega}_0$, $\varepsilon > 0$, $k := 0$.
Paso 1. Si $||\nabla f(\boldsymbol{\omega}_k)||_2 \leq \varepsilon$, *stop.*
Paso 2. Cálculo de $\boldsymbol{s}_k$ *resolviendo*

$$J(\boldsymbol{\omega}_k)^T J(\boldsymbol{\omega}_k)\boldsymbol{s}_k = -J(\boldsymbol{\omega}_k)^T \boldsymbol{r}(\boldsymbol{\omega}_k).$$

Paso 3. Cálculo de α_k. *Dos opciones:*

[1] La regla de Armijo-Goldstein sigue el esquema siguiente: (i) Se elige $t > 0$, $\beta \in (0,1)$, $\varepsilon \in (0,1)$.
(ii) Se toma $p = 0$ y se comprueba si se verifica $q(\beta^p t) \leq q(0) + \beta^p t \varepsilon q'(0)$.
(iii) Si se cumple se toma $\alpha = \beta^p t$, en caso contrario se hace $p = p + 1$ y se vuelve a (i).

1. *Tomar* $\alpha_k = \max\left\{1, \frac{1}{2}, \frac{1}{4}, ...\right\}$ *tal que se verifique*

$$||\boldsymbol{r}(\boldsymbol{\omega}_k)||_2^2 - ||\boldsymbol{r}(\boldsymbol{\omega}_k + \alpha_k \boldsymbol{s}_k)||_2^2 \geq \frac{1}{2}\alpha_k ||J(\boldsymbol{\omega}_k)\boldsymbol{s}_k||_2^2$$

2. *Tomar* α_k *como la solución de*

$$\min_{\alpha} ||\boldsymbol{r}(\boldsymbol{\omega}_k + \alpha_k)||_2^2$$

Paso 4. $\boldsymbol{\omega}_{k+1} = \boldsymbol{\omega}_k + \alpha_k \boldsymbol{s}_k$, $k := k+1$. *Volver al Paso 1.* □

1.3. Método de Levenberg-Marquardt

En la sección 1.2 veíamos que el método Gauss-Newton amortiguado era globalmente convergente en la mayoría de los casos. Sin embargo, cuando $J(\boldsymbol{\omega})$ no sea de rango completo este método tendrá dificultades para trabajar o el algoritmo convergerá a un punto no estacionario. Para solucionar este problema podemos tener en cuenta las segundas derivadas (Sección 1.4), o bien, estabilizar el método Gauss-Newton amortiguado.

Para llevar a cabo la segunda estrategia mencionada consideramos la técnica de región de confianza (*trust-region technique*), con la que obtenemos el método de Levenberg-Marquardt. Este algoritmo fue publicado por primera vez por Kenneth Levenberg, Levenberg (1944), y redescubierto en 1963 por Donald Marquardt, Marquardt (1963).

1.3.1. Descripción y propiedades

El método de Gauss-Newton podíamos pensarlo de dos maneras distintas: una de ellas era pensarlo como una serie de aproximaciones de $r(\omega)$ de la forma (1.6) obteniendo un problema lineal de mínimos cuadrados (1.12). Desafortunadamente, esta linealización no es efectiva para todos los $(\boldsymbol{\omega} - \boldsymbol{\omega_k})$. Para solucionar esto empleamos la técnica de región de confianza y añadimos una restricción. Consideramos entonces el siguiente modelo de región de confianza:

$$\begin{aligned} \min_{\boldsymbol{\omega}\in\mathbb{R}^n} \quad & \frac{1}{2}||\boldsymbol{r}(\boldsymbol{\omega}_k) + J(\boldsymbol{\omega}_k)(\boldsymbol{\omega} - \boldsymbol{\omega}_k)||_2^2 \\ \text{s.t} \quad & ||\boldsymbol{\omega} - \boldsymbol{\omega}_k||_2 \leq \Delta_k, \end{aligned} \tag{1.23}$$

que podemos verlo como un problema no lineal de mínimos cuadrados con restricciones. Este modelo puede reescribirse como

$$\begin{aligned} \min_{\boldsymbol{\omega}\in\mathbb{R}^n} \quad & \frac{1}{2}\boldsymbol{r}(\boldsymbol{\omega}_k)^T\boldsymbol{r}(\boldsymbol{\omega}_k) + \boldsymbol{r}(\boldsymbol{\omega}_k)^T J(\boldsymbol{\omega}_k)(\boldsymbol{\omega}-\boldsymbol{\omega}_k) + \frac{1}{2}(\boldsymbol{\omega}-\boldsymbol{\omega}_k)^T J(\boldsymbol{\omega}_k)^T J(\boldsymbol{\omega}_k)(\boldsymbol{\omega}-\boldsymbol{\omega}_k) \\ \text{s.t} \quad & ||\boldsymbol{\omega}-\boldsymbol{\omega}_k||_2 < \Delta_k. \end{aligned}$$

Denotamos $\boldsymbol{s} = \boldsymbol{\omega} - \boldsymbol{\omega}_k$. La solución al problema (1.23) se obtendrá resolviendo el sistema

$$(J(\boldsymbol{\omega}_k)^T J(\boldsymbol{\omega}_k) + \mu_k I)\boldsymbol{s} = -J(\boldsymbol{\omega}_k)^T \boldsymbol{r}(\boldsymbol{\omega}_k), \tag{1.24}$$

de donde obtenemos que

$$\boldsymbol{\omega}_{k+1} = \boldsymbol{\omega}_k - (J(\boldsymbol{\omega}_k)^T J(\boldsymbol{\omega}_k) + \mu_k I)^{-1} J(\boldsymbol{\omega}_k)^T \boldsymbol{r}(\boldsymbol{\omega}_k). \tag{1.25}$$

El parámetro $\Delta_k \geq 0$ se encarga de controlar las iteraciones y limita el tamaño de $\boldsymbol{s}_k$. Nótese que el parámetro $\boldsymbol{s}_k$ está bien definido incluso cuando $J(\boldsymbol{\omega}_k)$ no es de rango completo. Si se verifica que $||(J(\boldsymbol{\omega}_k)^T J(\boldsymbol{\omega}_k))^{-1} J(\boldsymbol{\omega}_k)^T \boldsymbol{r}(\boldsymbol{\omega}_k)||_2 \leq \Delta_k$ entonces se cumple que $\mu_k = 0$. En otro caso, existe $\mu_k > 0$ tal que la solución $\boldsymbol{s}_k$ satisface $||\boldsymbol{s}_k||_2 = \Delta_k$ y

$$(J(\boldsymbol{\omega}_k)^T J(\boldsymbol{\omega}_k) + \mu_k I)\boldsymbol{s}_k = -J(\boldsymbol{\omega}_k)^T \boldsymbol{r}(\boldsymbol{\omega}_k). \tag{1.26}$$

Otra forma de ver este método es como una mezcla entre Gauss-Newton y el método de descenso de gradiente (*steepest descent method*), de tal manera que este método permite elegir dos direcciones distintas. En el caso de que $\mu_k = 0$ la dirección sería la misma que en el Gauss-Newton, mientras que si μ_k es demasiado grande el sistema (1.24) se reduciría a

$$\mu_k I\boldsymbol{s} = -J(\boldsymbol{\omega}_k)^T \boldsymbol{r}(\boldsymbol{\omega}_k). \tag{1.27}$$

y la dirección que obtendríamos estaría muy próxima a la de máximo descenso.

A continuación enunciaremos una serie de propiedades del método Levenberg-Marquardt donde denotaremos $\boldsymbol{s} = \boldsymbol{s}(\mu)$ como la solución del sistema

$$(J(\boldsymbol{\omega})^T J(\boldsymbol{\omega}) + \mu I)\boldsymbol{s} = -J(\boldsymbol{\omega})^T \boldsymbol{r}(\boldsymbol{\omega}).$$

Teorema 1.3.1. *Cuando μ aumenta de forma monótona desde cero, $||\boldsymbol{s}(\mu)||_2$ decrecerá de forma estrictamente monótona.*

Demostración. Ver Sun y Yuan (2006, pág. 363-364).
□

Teorema 1.3.2. *El ángulo entre* $\boldsymbol{s}$ *y* $-\nabla f(\boldsymbol{\omega})$ *no incrementa monotonamente cuando* μ *aumenta.*

Demostración. Ver Sun y Yuan (2006, pág. 364-365).

□

Teorema 1.3.3. *Sea* $\mu_k > 0$ *y* $\boldsymbol{s}_k$ *la solución de* (1.24)*. Entonces* $\boldsymbol{s}_k$ *es solución global del subproblema*

$$\begin{aligned} \min_{\boldsymbol{s}\in\mathbb{R}^n} \quad & \frac{1}{2}||\boldsymbol{r}(\boldsymbol{\omega}_k) + J(\boldsymbol{\omega}_k)\boldsymbol{s}||_2^2 \\ s.t \quad & ||\boldsymbol{s}||_2 \leq ||\boldsymbol{s}_k||_2. \end{aligned}$$

Demostración. Ver Sun y Yuan (2006, pág. 365-366).

□

Teorema 1.3.4. *El vector* $\boldsymbol{s}_k$ *es solución del problema* (1.23) *para algún* $\Delta_k > 0$ *si, y sólo si, existe* $\mu \geq 0$ *verificando*

$$\begin{aligned} & (J(\boldsymbol{\omega}_k)^T J(\boldsymbol{\omega}_k) + \mu I)\boldsymbol{s}_k = -J(\boldsymbol{\omega}_k)^T \boldsymbol{r}(\boldsymbol{\omega}_k) \\ & \mu(\Delta_k - ||\boldsymbol{s}_k||_2) = 0, \\ & ||\boldsymbol{s}_k||_2 \leq \Delta_k. \end{aligned}$$

Demostración. Ver Sun y Yuan (2006, pág. 366).

□

Dado que este método puede pensarse como un caso especial de los métodos de región de confianza, al analizar su convergencia veremos que muchos de los resultados derivarán de los obtenidos para dichos métodos (ver Sun y Yuan, 2006, cap. 6). En Moré (1983) se realiza un análisis más general de los métodos de región de confianza para optimización no lineal. La convergencia del método de Levenberg-Marquardt, al implementarlo como un método de región de confianza, seguirá siendo lenta para residuos grandes o problemas claramente no lineales.

1.3.2. Implementación

Existen diversas maneras de implementar el método de Levenberg-Marquardt; sin embargo, la mayoría de las implementaciones no son robustas, o no tienen una justificación teórica sólida. Una de las formas más eficientes, propuesta por Moré (1978), es implementarlo como un algoritmo reescalado de región de confianza. Dicha implementación está

contenida en el paquete de software MINPACK (ver Moré et al., 1980).

La implementación del método de Levenberg-Marquardt que se propone en Moré (1978) consiste en calcular $\boldsymbol{s}$ como la solución del siguiente sistema de ecuaciones

$$\boldsymbol{s}_k = -(J(\boldsymbol{\omega}_k)^T J(\boldsymbol{\omega}_k) + \mu_k D_k^T D_k)^{-1} J(\boldsymbol{\omega}_k)^T \boldsymbol{r}(\boldsymbol{\omega}_k), \tag{1.28}$$

que se corresponde con el siguiente subproblema de región de confianza

$$\min_{\boldsymbol{s}\in\mathbb{R}^n} \quad \frac{1}{2}||\boldsymbol{r}(\boldsymbol{\omega}_k) + J(\boldsymbol{\omega}_k)\boldsymbol{s}||_2^2 \tag{1.29}$$

$$\text{s.t} \qquad ||D_k\boldsymbol{s}||_2 \leq \Delta_k, \tag{1.30}$$

donde $\Delta_k > 0$ es el radio de la región de confianza y D_k es una matriz diagonal que tiene en cuenta el reescalado del problema. En el caso de que $J(\boldsymbol{\omega}_k)$ sea singular y $\mu_k = 0$ podemos definir la solución del sistema como

$$D_k\boldsymbol{s}(0) = \lim_{\mu_k\to 0^+} D_k\boldsymbol{s}(\mu_k) = -(J(\boldsymbol{\omega}_k)D_k^{-1})^T\boldsymbol{r}(\boldsymbol{\omega}_k). \tag{1.31}$$

Existen entonces dos posibilidades: en primer lugar, puede darse que $\mu_k = 0,\ ||D_k\boldsymbol{s}(0)|| \leq \Delta_k$, en cuyo caso la solución al sistema será $\boldsymbol{s}(0)$; si esto no ocurre entonces $\mu_k > 0,\ ||D_k\boldsymbol{s}(0)|| = \Delta_k$ y tendremos entonces que $\boldsymbol{s}(\mu_k)$ es la única solución del sistema.

Se sugiere la siguiente iteración

Algoritmo 1.3.5. *Paso 1. Dado $\Delta_k > 0$, encontrar $\mu_k \geq 0$ tal que*

$$(J(\boldsymbol{\omega}_k)^T J(\boldsymbol{\omega}_k) + \mu_k D_k^T D_k)\boldsymbol{s}_k = -J(\boldsymbol{\omega}_k)^T\boldsymbol{r}(\boldsymbol{\omega}_k).$$

Paso 2. Si $||\boldsymbol{r}(\boldsymbol{\omega}_k + \boldsymbol{s}_k)||_2 \leq ||r(\omega_k)||_2$, tomamos $\boldsymbol{\omega}_{k+1} = \boldsymbol{\omega}_k + \boldsymbol{s}_k$ y calculamos $J(\boldsymbol{\omega}_{k+1})$. En otro caso, tomamos $\boldsymbol{\omega}_{k+1} = \boldsymbol{\omega}_k$ y $J(\boldsymbol{\omega}_{k+1}) = J(\boldsymbol{\omega}_k)$.
Paso 3. Elegimos Δ_{k+1} y D_{k+1}. □

A continuación, discutimos cómo realizar el algoritmo anterior de manera eficiente y fidedigna.

La forma más sencilla de obtener la dirección $\boldsymbol{s}_k$ es utilizar la descomposición de Cholesky en el sistema lineal

$$(J(\boldsymbol{\omega}_k)^T J(\boldsymbol{\omega}_k) + \mu_k D_k^T D_k)\boldsymbol{s}_k = -J(\boldsymbol{\omega}_k)^T\boldsymbol{r}(\boldsymbol{\omega}_k). \tag{1.32}$$

Observación 1.3.6. Otra manera de resolver esto sería emplear la descomposición QR con pivotamiento de columna. La principal ventaja de eutilizar (1.32) es la velocidad ya que es posible resolver el problema hasta dos veces más rápido que con la descomposición QR. Por otro lado, estas ecuaciones son no fidedignas cuando $\mu_k = 0$ y $J(\boldsymbol{\omega}_k)$ está próximo de ser de rango deficiente. Además, el cálculo de $J(\boldsymbol{\omega}_k)^T J(\boldsymbol{\omega}_k)$ o $D_k^T D_k$ puede provocar que la solución salga de la región de confianza establecida, mientras que esto no ocurre al utilizar la descomposición QR. Debido a esto se considera que la pérdida de velocidad se compensa al ganar confiabilidad y robustez.

La elección de Δ_k depende de la proporción entre la actual reducción y la predicción de la reducción obtenida por la corrección de la función objetivo. En el caso de mínimos cuadrados de funciones no lineales, esta proporción viene dada por

$$\rho_k = \frac{||\boldsymbol{r}(\boldsymbol{\omega}_k)||_2^2 - ||\boldsymbol{r}(\boldsymbol{\omega}_k + \boldsymbol{s}_k)||_2^2}{||\boldsymbol{r}(\boldsymbol{\omega}_k)||_2^2 - ||\boldsymbol{r}(\boldsymbol{\omega}_k) + J(\boldsymbol{\omega}_k)\boldsymbol{s}_k||_2^2} \tag{1.33}$$

que mide la similitud entre el modelo linealizado y la función no lineal.

El esquema para actualizar Δ_k tiene el objetivo de mantener el valor de (1.33) en un nivel razonable. Por lo tanto, si ρ_k está próximo a la unidad podríamos querer aumentar Δ_k, en el caso de que no estuviera cerca de la unidad deberíamos disminuir el valor de Δ_k.

Dado que el propósito de la matriz D_k es tener en cuenta el reescalado del problema se elige

$$D_k = diag(d_1^{(k)}, ..., d_n^{(k)}), \tag{1.34}$$

donde

$$d_i^{(0)} = ||\partial_i \boldsymbol{r}(\boldsymbol{\omega}_0)||, \tag{1.35}$$

$$d_i^{(k)} = \text{máx}\{d_i^{(k-1)}, ||\partial_i \boldsymbol{r}(\boldsymbol{\omega}_k)||\}, \ k \geq 1. \tag{1.36}$$

El algoritmo propuesto en Moré (1978) es el siguiente:

Algoritmo 1.3.7. *(Levenberg-Marquardt, versión de Moré)*
Paso 0. $k := 0$, $\boldsymbol{\omega}_0$ *dado,* $D_0 = diag(d_1^{(0)}, ..., d_n^{(0)})$ *donde* $d_i^{(0)} = ||\partial_i \boldsymbol{r}(\boldsymbol{\omega}_0)||$.
Paso 1. Sea $\sigma \in (0,1)$. *Si* $||D_k J(\boldsymbol{\omega}_k)^- \boldsymbol{r}(\boldsymbol{\omega}_k)||_2 \leq (1+\sigma)\Delta_k$, *tomamos* $\mu_k = 0$ *y* $\boldsymbol{s}_k = -J(\boldsymbol{\omega}_k)^- \boldsymbol{r}(\boldsymbol{\omega}_k)$ [2]*; en otro caso determinamos* $\mu_k > 0$ *tal que si verifica* (1.32) *entonces*

$$(1-\sigma)\Delta_k \leq ||D_k \boldsymbol{s}_k|| \geq (1+\sigma)\Delta_k.$$

[2] $J(\boldsymbol{\omega}_k)^-$ denota una matriz inversa simétrica generalizada de $J(\boldsymbol{\omega}_k)$ en el sentido de que $J(\boldsymbol{\omega}_k)J(\boldsymbol{\omega}_k)^-$ es simétrico y $J(\boldsymbol{\omega}_k)J(\boldsymbol{\omega}_k)^- J(\boldsymbol{\omega}_k) = J(\boldsymbol{\omega}_k)$.

Paso 2. Calcular la proporción ρ_k entre la actual reducción y la predicción de la reducción de la función objetivo según (1.33).
Paso 3. Si $\rho_k \leq 0.0001$, tomamos $\boldsymbol{\omega}_{k+1} = \boldsymbol{\omega}_k$ y $J(\boldsymbol{\omega}_{k+1}) = J(\boldsymbol{\omega}_k)$.
Si $\rho_k > 0.0001$, tomamos $\boldsymbol{\omega}_{k+1} = \boldsymbol{\omega}_k + \boldsymbol{s}_k$ y calculamos $J(\boldsymbol{\omega}_{k+1})$.
Paso 4. Si $\rho_k \leq \frac{1}{4}$, tomamos $\Delta_{k+1} \in \left[\frac{1}{10}\Delta_k, \frac{1}{2}\Delta_k\right]$. Si $\rho_k \in \left[\frac{1}{4}, \frac{1}{3}\right]$ y $\mu_k = 0$ o $\rho_k \geq \frac{3}{4}$, entonces tomamos $\Delta_{k+1} = 2||D_k \boldsymbol{s}_k||_2$.
Paso 5. Actualizamos D_{k+1} mediante (1.34) *y* (1.36). *Volvemos al Paso 1.*

En Moré (1978) se expone la demostración del siguiente resultado probando así la convergencia del Algoritmo 1.3.7

Teorema 1.3.8. *Sea $\boldsymbol{r} : \mathbb{R}^n \longrightarrow \mathbb{R}^m$ continuamente diferenciable. Sea $\{\boldsymbol{\omega}_k\}$ la secuencia generada por el Algoritmo 1.3.7. Entonces*

$$\liminf_{k \to +\infty} ||(J(\boldsymbol{\omega}_k) D_k^{-1})^T \boldsymbol{r}(\boldsymbol{\omega}_k)||_2 = 0. \tag{1.37}$$

Este resultado garantiza que el gradiente reescalado será lo suficientemente pequeño. Si $\{J(\boldsymbol{\omega}_k)\}$ está acotado, (1.37) implica que

$$\liminf_{k \to +\infty} ||J(\boldsymbol{\omega}_k)^T \boldsymbol{r}(\boldsymbol{\omega}_k)||_2 = 0. \tag{1.38}$$

Además, si $\nabla \boldsymbol{r}(\boldsymbol{\omega})$ es uniformemente continuo entonces

$$\lim_{k \to +\infty} ||J(\boldsymbol{\omega}_k)^T \boldsymbol{r}(\boldsymbol{\omega}_k)||_2 = 0. \tag{1.39}$$

En la biblioteca libre Scipy de Python encontramos distintos módulos que proporcionan un contenedor para las rutinas de MINPACK. En este trabajo se emplearán tres de ellos *scipy.optimize.leastsq*, *scipy.optimize.least_squares* y *scipy.optimize.curve_fit*. Analizaremos su comportamiento y lo compararemos tanto con las implementaciones de los restantes métodos descritos en este trabajo como entre ellos. Cabe destacar que el módulo *curve_fit* implementa el algoritmo de Levenberg-Marquardt a través de *leastsq*, por lo que cabe esperar que sea preferible emplear este segundo.

1.4. Métodos Quasi-Newton

En las secciones anteriores hemos visto que en el caso de que el residuo sea grande o nuestro problema sea claramente no lineal, los métodos de Gauss-Newton y Levenberg-Marquardt no terminan de funcionar bien. La convergencia será lenta y solamente lineal. Esto se debe a que ambos métodos consideran que $J(\boldsymbol{\omega})^T J(\boldsymbol{\omega})$ es una buena aproximación para $\nabla^2 f(\boldsymbol{\omega}) = J(\boldsymbol{\omega})^T J(\boldsymbol{\omega}) + S(\boldsymbol{\omega})$, es decir, no estamos empleando la información que

nos aporta el término de segundo orden $S(\boldsymbol{\omega}) = \sum_{i=1}^{m} \boldsymbol{r}_i(\boldsymbol{\omega})\nabla^2\boldsymbol{r}_i(\boldsymbol{\omega})$. Esta suposición no está justificada para los llamados problemas de residuos grandes, en los que $||\boldsymbol{r}(\tilde{\boldsymbol{\omega}})||$ no es "pequeño". Supongamos que $||\nabla_i^2 f(\boldsymbol{\omega})||$ es de orden uno para $i = 1, ..., m$. La matriz $S(\boldsymbol{\omega})$ siempre será significativa si $\boldsymbol{r}(\tilde{\boldsymbol{\omega}})$ excede los valores propios más pequeños de $J(\tilde{\boldsymbol{\omega}})^T J(\tilde{\boldsymbol{\omega}})$. Por lo tanto, definimos un problema residual grande como uno en el que el residuo óptimo es grande en relación con los valores propios pequeños de $J(\tilde{\boldsymbol{\omega}})^T J(\tilde{\boldsymbol{\omega}})$, pero no con respecto a su valor propio más grande.

Cuando las segundas derivadas de $f(\boldsymbol{\omega})$ son asequibles podemos emplear, como comentábamos al principio, el método de Newton para resolver estos problemas. Sin embargo, raramente el término $S(\boldsymbol{\omega})$ se puede obtener con un coste computacional razonable. Surgen entonces nuevos métodos empleando diversas aproximaciones quasi-Newton para el término de orden dos. Notemos que esta aproximación debe de ser definida positiva para garantizar así la obtención de una dirección de descenso.

1.4.1. Método Dogbox

En los métodos vistos hasta el momento se han seguido dos enfoques generales. El primero de ellos sigue una búsqueda lineal a lo largo de una dirección de descenso debidamente seleccionada, como ocurría con el métdodo de Gauss-Newton, mientras que el segundo permite pasos de un tamaño restringido para así intentar mantener la fiabilidad de la aproximación cuadrática. Los enfoques que siguen esta segunda estrategia reciben el nombre de técnicas de región de confianza. Dichas técnicas las veíamos, por ejemplo, en el método de Levenberg-Marquardt. En Voglis y Lagaris (2004) se presenta un nuevo método que sigue esta segunda estrategia adoptando una forma rectangular para la región de confianza, el método de Dogbox, que consiste en una modificación de la técnica dogleg expuesta en Powell (1970) adecuándolo a regiones de confianza rectangulares. Esta geometría tiene la ventaja obvia de la linealidad de las restricciones del subproblema y, además, permite la fácil adaptación a problemas acotados.

Descripción y propiedades

El método Dogbox hace uso de la información que aporta $S(\boldsymbol{\omega})$ realizando una aproximación de la matriz Hessiana $\nabla^2 f(\boldsymbol{\omega})$, que denotaremos por M. Para asegurarnos de que dicha aproximación es definida positiva emplearemos una aproximación BFGS. En cada iteración k escribimos el modelo cuadrático de $f(\omega)$

$$q_k(\boldsymbol{\omega}) = \frac{1}{2} r(\boldsymbol{\omega}_k)^T \boldsymbol{r}(\boldsymbol{\omega}_k) + (J(\boldsymbol{\omega}_k)^T \boldsymbol{r}(\boldsymbol{\omega}_k))^T (\boldsymbol{\omega} - \boldsymbol{\omega}_k) + \frac{1}{2}(\boldsymbol{\omega} - \boldsymbol{\omega}_k)^T M_k (\boldsymbol{\omega} - \boldsymbol{\omega}_k), \quad (1.40)$$

y definimos la región de confianza como

$$T_k = \{\boldsymbol{\omega} \in \mathbb{R}^n,\ ||\boldsymbol{\omega} - \boldsymbol{\omega}_k||_\infty \leq \Delta_k\}. \tag{1.41}$$

En el caso de estar tratando un problema sin restricciones la región de confianza será un hipercubo. Tenemos entonces que el problema original se reduce a resolver el subproblema siguiente

$$\begin{aligned} &\min_{\boldsymbol{\omega}\in\mathbb{R}^n} \quad q_k(\boldsymbol{\omega}) = \frac{1}{2}\boldsymbol{r}(\boldsymbol{\omega}_k)^T\boldsymbol{r}(\boldsymbol{\omega}_k) + (J(\boldsymbol{\omega}_k)^T\boldsymbol{r}(\boldsymbol{\omega}_k))^T(\boldsymbol{\omega} - \boldsymbol{\omega}_k) + \frac{1}{2}(\boldsymbol{\omega} - \boldsymbol{\omega}_k)^T M_k(\boldsymbol{\omega} - \boldsymbol{\omega}_k), \\ &\text{s.t} \quad ||\boldsymbol{\omega} - \boldsymbol{\omega}_k||_\infty \leq \Delta_k. \end{aligned} \tag{1.42}$$

Debido a que el término $f(\boldsymbol{\omega}_k)$ es una constante respecto de $\boldsymbol{\omega}$ podemos omitir este término. De esta forma llegamos al siguiente problema

$$\begin{aligned} &\min_{\boldsymbol{\omega}\in\mathbb{R}^n} \quad (J(\boldsymbol{\omega}_k)^T\boldsymbol{r}(\boldsymbol{\omega}_k))^T(\boldsymbol{\omega} - \boldsymbol{\omega}_k) + \frac{1}{2}(\boldsymbol{\omega} - \boldsymbol{\omega}_k)^T M_k(\boldsymbol{\omega} - \boldsymbol{\omega}_k), \\ &\text{s.t} \quad ||\boldsymbol{\omega} - \boldsymbol{\omega}_k||_\infty \leq \Delta_k, \end{aligned} \tag{1.43}$$

que se trata de un problema estrictamente convexo debido a que la aproximación M_k de la Hessiana es definida positiva. Esto nos permite aplicar la técnica dogleg donde la dirección viene definida como

$$\boldsymbol{s}(a) = \begin{cases} aC, & 0 \leq a \leq 1 \\ C + (a-1)(N-C), & 1 \leq a \leq 2 \end{cases}$$

siendo $C = \frac{\boldsymbol{r}(\boldsymbol{\omega_k})^T J(\boldsymbol{\omega_k}) J(\boldsymbol{\omega_k})^T \boldsymbol{r}(\boldsymbol{\omega_k})}{((\boldsymbol{r}(\boldsymbol{\omega_k})^T J(\boldsymbol{\omega_k}) B_k J(\boldsymbol{\omega_k})^T \boldsymbol{r}(\boldsymbol{\omega_k})} J(\boldsymbol{\omega_k})^T \boldsymbol{r}(\boldsymbol{\omega_k})$, el paso de Cauchy, y $N = -M_k^{-1}(J(\boldsymbol{\omega_k})^T \boldsymbol{r}(\boldsymbol{\omega_k}))$, el paso de Newton que minimiza q_k.

Asumiendo que M_k es una matriz definida positiva se cumple que el modelo cuadrático $q_k(\boldsymbol{s}(a))$ decrece de forma monótona cuando a aumenta. Distinguimos tres casos:

Caso 1: $N \in T_k$.

Caso 2: $C \in T_k,\ N \notin T_k$.

Caso 3: $C \notin T_k,\ N \notin T_k$.

Los casos 1 y 2 se tratan de la misma forma que en caso artículo original de Powell (1970); sin embargo, en el caso 3 realizamos una pequeña modificación del algoritmo. En lugar de tomar el máximo paso posible en la dirección $C(PC = bC, b \leq 1)$ procedemos hacia N en la dirección $N - PC$ hasta encontrar el límite. Siguiendo esta modificación tenemos que

$$\boldsymbol{s}(a) = \begin{cases} aC, & 0 \leq a \leq b \\ bC + (a-1)(N - bC), & b \leq a \leq 1 + b \end{cases}$$

donde $b = \frac{||PC||_2}{||C||_2} \in [0,1]$. Trivialmente $q_k(\boldsymbol{s}(a))$ disminuye de forma monótona alcanzando así un valor más bajo para el modelo.

Es probable que el algoritmo exhiba una convergencia lenta cuando el rango de $J(\boldsymbol{\omega})$ es menor que el número de variables.

Implementación

De la descripción dada del método de Dogbox en la subsección anterior se sigue el siguiente algoritmo:

Algoritmo 1.4.1. *(Dogbox)*

Paso 0. Elegimos el iterante inicial $\boldsymbol{\omega}_0$*, el parámetro inicial de la región de confianza* Δ_0 *y fijamos* $k := 0$.

Paso 1. Construimos el modelo cuadrático en torno a $\boldsymbol{\omega}_k$*, es decir,*

$$\begin{aligned} &\min_{\boldsymbol{\omega}\in\mathbb{R}^n} \quad (J(\boldsymbol{\omega}_k)^T \boldsymbol{r}(\boldsymbol{\omega}_k))^T(\boldsymbol{\omega}-\boldsymbol{\omega}_k) + \frac{1}{2}(\boldsymbol{\omega}-\boldsymbol{\omega}_k)^T M_k(\boldsymbol{\omega}-\boldsymbol{\omega}_k), \\ &s.t \quad ||\boldsymbol{\omega}-\boldsymbol{\omega}_k||_\infty \leq \Delta_k, \end{aligned}$$

Paso 2. Calculamos $\boldsymbol{s}_k$*:*

if $N = -M_k(J(\boldsymbol{\omega_k})^T \boldsymbol{r}(\boldsymbol{\omega_k}))$ *es factible*

$\boldsymbol{s}_k = N$

else

if $C = \frac{\boldsymbol{r}(\boldsymbol{\omega_k})^T J(\boldsymbol{\omega_k}) J(\boldsymbol{\omega_k})^T \boldsymbol{r}(\boldsymbol{\omega_k})}{\boldsymbol{r}(\boldsymbol{\omega_k})^T J(\boldsymbol{\omega_k}) B_k J(\boldsymbol{\omega_k})^T \boldsymbol{r}(\boldsymbol{\omega_k})} J(\boldsymbol{\omega_k})^T \boldsymbol{r}(\boldsymbol{\omega_k})$ *es factible*

encontrar el máximo α *tal que* $C + \alpha(N-C) \in T_k$,

$s_k = C + \alpha(N-C)$

else

encontrar el máximo β *tal que* $PC \equiv \beta C \in T_k$ *y el máximo* α *tal que* $PC + \alpha(N-PC) \in T_k$,

$s_k = PC + \alpha(N-PC)$

Paso 3. Calculamos el radio $\delta_k = \frac{f(\boldsymbol{\omega}_k) - f(\boldsymbol{\omega}_k + s_k)}{q_k(0) - q_k(s_k)}$.

Paso 4. Elegimos el nuevo punto $\boldsymbol{\omega}_{k+1}$*:*

if $\delta_k \leq 0.1$

$\boldsymbol{\omega}_{k+1} = \boldsymbol{\omega}_k$

else

$\boldsymbol{\omega}_{k+1} = \boldsymbol{\omega}_k + \boldsymbol{s}_k$

Paso 5. Actualizamos Δ_k*:*

if $\delta_k < 0.25$

$\Delta_{k+1} = ||\boldsymbol{s}_k||_2/4$

else if $\delta_k > 0.75$ *y* $||s_k||_2 = \Delta_k$
$\Delta_{k+1} = 2\Delta_k$
else
$\Delta_{k+1} = \Delta_k$

Paso 6. $k := k + 1$. *Volver al Paso 1.* □

Al igual que ocurría con el método de Levenberg-Marquardt, en la biblioteca libre Scipy encontramos distintos módulos que implementan este algoritmo. En esta memoria utilizaremos dos: *scipy.optimize.least_ squares* y *scipy.optimize.curve_ fit*. Analizaremos su comportamiento y lo compararemos tanto con las implementaciones de los restantes métodos descritos en este trabajo como entre ellos.

1.4.2. Método NL2SOL

La aplicación directa de los métodos quasi-Newton al problema de mínimos cuadrados no lineales descrito anteriormente, en ocasiones en la práctica, no resultan tan eficientes. Una de las razones por las que ocurre esto es que estos métodos ignoran la información que aporta $J(\boldsymbol{\omega}_k)$, pero a menudo $J(\boldsymbol{\omega}_k)^T J(\boldsymbol{\omega}_k)$ es la parte dominante de $\nabla^2 f(\boldsymbol{\omega})$. Una estrategia posible es, en lugar de aproximar $\nabla^2 f(\boldsymbol{\omega})$, incluir una aproximación quasi-Newton del término desconocido de la segunda derivada, $S(\omega)$. Sea B_k la aproximación de la matriz $S(\omega_k)$, se tiene entonces

$$(B_k + J(\boldsymbol{\omega}_k)^T J(\boldsymbol{\omega}_k))\boldsymbol{s}_k = -J(\boldsymbol{\omega}_k)^T \boldsymbol{r}_k. \tag{1.44}$$

Dado que

$$S(\omega_{k+1}) = \sum_{i=1}^{m} r_i(\boldsymbol{\omega}_{k+1}) \nabla^2 r_i(\boldsymbol{\omega}_{k+1})$$

podemos tomar

$$B_{k+1} = \sum_{i=1}^{m} r_i(\boldsymbol{\omega}_{k+1})(H_i)_{k+1}$$

como la aproximación de $S(\boldsymbol{\omega}_{k+1})$, donde $(H_i)_{k+1}$ es una aproximación de $\nabla^2 r_i(\boldsymbol{\omega}_{k+1})$. Tenemos entonces

$$\begin{aligned} B_{k+1}(\boldsymbol{\omega}_{k+1} - \boldsymbol{\omega}_k) &= \sum_{i=1}^{m} (r_i(\boldsymbol{\omega}_{k+1})(H_i)_{k+1})(\boldsymbol{\omega}_{k+1} - \boldsymbol{\omega}_k) \\ &= \sum_{i=1}^{m} r_i(\boldsymbol{\omega}_{k+1})(\nabla r_i(\boldsymbol{\omega}_{k+1}) - \nabla r_i(\boldsymbol{\omega}_k)) \\ &= J(\boldsymbol{\omega}_{k+1})^T \boldsymbol{r}(\boldsymbol{\omega}_{k+1}) - J(\boldsymbol{\omega}_k)^T \boldsymbol{r}(\boldsymbol{\omega}_{k+1}) =: \boldsymbol{y}_k, \end{aligned} \tag{1.45}$$

lo cual es una condición quasi-Newton impuesta sobre B_k.

En Dennis et al. (1981) se sugiere una regla de aproximación para $S(\boldsymbol{\omega}_k)$ simple, general y geométrica. La idea es elegir un conjunto de características deseables para la aproximación y posteriormente seleccionar $S(\boldsymbol{\omega}_{k+1})$ como el punto factible más cercano a $S(\boldsymbol{\omega}_k)$. Veamos entonces qué propiedades debe verificar $S(\boldsymbol{\omega}_{k+1})$. Es razonable comenzar con $S(\boldsymbol{\omega}_0) = 0$ dado que es barato computacionalmente y, además, es lógico pensar que $q_0 = q_0^G$. Supongamos entonces que $S(\boldsymbol{\omega}_k)$ está disponible. Recordamos que queremos aproximar $\sum_{i=1}^m r_i(\boldsymbol{\omega})\nabla^2 r_i(\boldsymbol{\omega})$, por lo tanto es obvio que debe ser simétrico. Dennis et al. (1981) se basa en el siguiente resultado para proporcionar un formula de actualización así como sus propiedades.

Notación 1.4.2. Sea $A \in \mathcal{M}_{m\times n}(\mathcal{C})$, se define la norma de Frobenius, llamada también norma de Hilbert–Schmidt, como $||A||_F := (tr(A^TA))^{\frac{1}{2}} = \left(\sum_{i=1}^m \sum_{j=1}^n |a_{ij}|^2\right)^{\frac{1}{2}}$.

Teorema 1.4.3. *Sea $\boldsymbol{v}_k^T\boldsymbol{s}_k > 0$ y $T \in \mathbb{R}^{n\times n}$ una matriz simétrica y definida positiva tal que*

$$TT^T\boldsymbol{s}_k = \boldsymbol{v}_k$$

donde

$$\boldsymbol{v}_k = \nabla f(\boldsymbol{\omega}_{k+1}) - \nabla f(\boldsymbol{\omega}_k).$$

Entonces la actualización

$$B_{k+1} = B_k + \frac{(\boldsymbol{y}_k - B_k\boldsymbol{s}_k)\boldsymbol{v}_k^T + \boldsymbol{v}_k(\boldsymbol{y}_k - B_k\boldsymbol{s}_k)^T}{\boldsymbol{s}_k^T\boldsymbol{v}_k} - \frac{(\boldsymbol{y}_k - B_k\boldsymbol{s}_k)^T\boldsymbol{s}_k}{(\boldsymbol{s}_k^T\boldsymbol{v}_k)^2}\boldsymbol{v}_k\boldsymbol{v}_k^T \tag{1.46}$$

es la única solución del problema de minimización

$$\begin{aligned} \text{mín} \quad & ||T^{-T}(B_{k+1} - B_k)T^{-1}||_F \\ s.t \quad & (B_{k+1} - B_k) \text{ simétrica}, B_{k+1}\boldsymbol{s}_k = \boldsymbol{y}_k. \end{aligned}$$

Dennis et al. (1981) emplearon la condición quasi-Newton (1.45) y la fórmula de actualización (1.46), y presentaron un nuevo algoritmo quasi-Newton con región de confianza que recibe el nombre de NL2SOL. En cada etapa es necesario resolver el subproblema

$$\begin{aligned} \min_{\boldsymbol{\omega}\in\mathbb{R}^n} \quad & \frac{1}{2}\boldsymbol{r}(\boldsymbol{\omega}_k)^T\boldsymbol{r}(\boldsymbol{\omega}_k) + (J(\boldsymbol{\omega}_k)^T\boldsymbol{r}(\boldsymbol{\omega}_k))^T(\boldsymbol{\omega} - \boldsymbol{\omega}_k) \\ & + \frac{1}{2}(\boldsymbol{\omega} - \boldsymbol{\omega}_k)^T(J(\boldsymbol{\omega}_k)^TJ(\boldsymbol{\omega}_k) + B_k)(\boldsymbol{\omega} - \boldsymbol{\omega}_k) \\ \text{s.t} \quad & ||\boldsymbol{\omega} - \boldsymbol{\omega}_k||_2 \leq \Delta_k. \end{aligned} \tag{1.47}$$

Se cumple que, en el caso en el que el residuo sea cero, la matriz B_k debería desaparecer o, al menos, acercarse a 0. Sin embargo, en el algoritmo propuesto, la actualización (1.46) no

garantiza que desaparezca la matriz B_k. Cuando esto ocurre, el método de Gauss-Newton es superior al NL2SOL. Además, esto puede interferir con la convergencia superlineal del método.

Para solucionar dicho problema utilizaremos una modificación directa del autoescalado. La idea es actualizar $\gamma_k B_k$ en lugar de B_k para obtener B_{k+1}. Podemos tomar como factor de escalado

$$\gamma_k = \frac{|(\boldsymbol{\omega}_{k+1} - \boldsymbol{\omega}_k)^T \boldsymbol{y}_k|}{|(\boldsymbol{\omega}_{k+1} - \boldsymbol{\omega}_k)^T B_k (\boldsymbol{\omega}_{k+1} - \boldsymbol{\omega}_k)|}. \tag{1.48}$$

Dado que queremos que B_k no sea demasiado grande tomamos

$$\gamma_k = \text{mín}\left\{ \frac{|(\boldsymbol{\omega}_{k+1} - \boldsymbol{\omega}_k)^T \boldsymbol{y}_k|}{|(\boldsymbol{\omega}_{k+1} - \boldsymbol{\omega}_k)^T B_k (\boldsymbol{\omega}_{k+1} - \boldsymbol{\omega}_k)|},\ 1 \right\} \tag{1.49}$$

A partir de los experimentos numéricos reflejados en Dennis et al. (1981) observamos que para problemas con residuos grandes, el algoritmo quasi-Newton NL2SOL es significativamente ventajoso. Sin embargo, para problemas con residuos significativamente pequeños, el rendimiento de NL2SOL y del algoritmo de Levenberg-Marquardt implementado en Moré (1978) es similar. En el caso de los problemas con residuo nulo seguimos prefirendo el método de Gauss-Newton.

Capítulo 2

Resultados numéricos

A lo largo de este capítulo analizaremos el comportamiento de los algoritmos descritos empleando el lenguaje de programación interpretado *Python*[1] . Como ya hemos comentado, haremos uso tanto de bibliotecas libres como SciPy, donde encontramos implementados los métodos de Levenberg-Marquardt y Dogbox, como de código propio en el caso de los algoritmos restantes.

La validación de los algoritmos descritos la realizaremos mediante problemas de distintos grados de dificultad tomados de *Statistical Reference Datsets Project* (STRD) desarrollado por el *Staff Statistical Engineering Division and the Mathematical and Computational Sciences Division* del *National Institute of Standards and Techonology* (NIST), que proporciona bases de datos referenciadas con valores certificados. En primer lugar, haremos una descripción de los distintos problemas para, posteriormente, mostrar y analizar los resultados obtenidos.

Encontrar el "mejor" código es una tarea casi imposible y muy dependiente de la definición de "mejor". Para cualquiera que sea el criterio utilizado, el procedimiento de prueba debe intentar medir la capacidad del código para encontrar soluciones. Los problemas de mínimos cuadrados no lineales son intrínsecamente difíciles y, generalmente, es posible encontrar un conjunto de datos que haga fallar incluso a los códigos más robustos. Por lo

[1]Se trata de un lenguaje de programación multiparadigma, ya que soporta programación orientada a objetos, programación imperativa y, en menor medida, programación funcional. Es un lenguaje interpretado, dinámico y multiplataforma. Se administra por la Python Software Foundation y posee una licencia de código abierto, denominada Python Software Foundation License, que es compatible con la Licencia pública general de GNU a partir de la versión 2.1.1, e incompatible en ciertas versiones anteriores. En este trabajo se ha usado la versión 3.7.1.

tanto, la mayoría de las evaluaciones de software de mínimos cuadrados no lineales también deben incluir una medida de la confiabilidad del código, es decir, el código debe reconocer correctamente cuándo ha encontrado una solución. Los conjuntos de datos empleados son particularmente adecuados para tales pruebas de robustez y confiabilidad. Se incluyen problemas de mínimos cuadrados no lineales generados y "del mundo real". Los valores certificados son soluciones "mejores disponibles", obtenidas con precisión de 128 bits y confirmadas por, al menos, dos algoritmos y paquetes de software diferentes que utilizan derivadas analíticas.

2.1. Descripción de los problemas

En esta sección presentaremos 3 problemas test ordenados por grado de dificultad, según la clasificación del NIST. Emplearemos dichos problemas para analizar tanto el buen comportamiento de los algoritmos descritos en el primer capítulo como su implementación.

2.1.1. Test 1: Gauss1

Los datos empleados en este problema son datos generados que se corresponden con dos normales en una base de decrecimiento exponencial con errores normales de media cero y varianza 6.25. Sea $\{(x_i, y_i)\}_{i=1}^{250}$ una muestra donde $x_i, y_i \in \mathbb{R}$ son los valores de las variables explicativa y respuesta, respectivamente, correspondientes al i-ésimo individuo. Sea $\boldsymbol{\theta} \in \mathbb{R}^8$ el vector de parámetros desconocidos. Consideraremos el ajuste no lineal dado por:

$$y_i = h(\boldsymbol{x_i}, \boldsymbol{\theta}) + \varepsilon_i, \quad \varepsilon_i \in N(0, 6.25), \quad i = 1, ..., 250,$$

donde

$$h(x, \boldsymbol{\theta}) = \theta_1 \cdot exp(-\theta_2 x) + \theta_3 \cdot exp\left(\frac{-(x-\theta_4)^2}{\theta_5^2}\right) + \theta_6 \cdot exp\left(\frac{-(x-\theta_7)^2}{\theta_8^2}\right).$$

El problema a resolver sigue la fórmula general (4):

$$\min_{\boldsymbol{\theta}\in\mathbb{R}^8} f(\boldsymbol{\theta}) = \min_{\boldsymbol{\theta}\in\mathbb{R}^8} \frac{1}{2}\boldsymbol{r}(\boldsymbol{\theta})^T\boldsymbol{r}(\boldsymbol{\theta}) = \min_{\boldsymbol{\theta}\in\mathbb{R}^8} \frac{1}{2}\sum_{i=1}^{250}[r_i(\boldsymbol{\theta})]^2,$$

siendo $r_i(\boldsymbol{\theta}) = y_i - h(\boldsymbol{x_i}, \boldsymbol{\theta})$, $r_i : \mathbb{R}^8 \to \mathbb{R}$, $i = 1, ..., 250$.

En la Figura 2.1 representamos el ajuste benchmark tomando como valores certificados $\boldsymbol{\theta}$ =(9.8778210871e+01, 1.0497276517e-02, 1.0048990633e+02, 6.7481111276e+01,

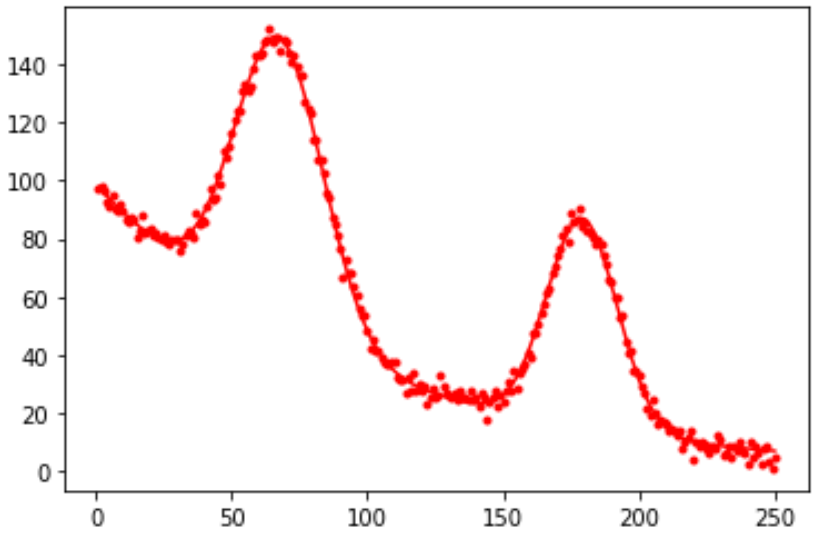

Figura 2.1: Ajuste de Gauss1 con la función del benchmark.

2.3129773360e+01, 7.1994503004e+01, 1.7899805021e+02, 1.8389389025e+01). Empleando dichos valores certificados obtenemos que $RSS(\boldsymbol{\theta})$ =1.3158222432e+03.

Al realizar la implementación de este problema utilizaremos dos iterantes iniciales distintos:

$$\boldsymbol{\theta_0^1} = (97.0, 0.009, 100.0, 65.0, 20.0, 70.0, 178.0, 16.5),$$

$$\boldsymbol{\theta_0^2} = (94.0, 0.0105, 99.0, 63.0, 25.0, 71.0, 180.0, 20.0).$$

2.1.2. Test 2: Hahn1

Los datos empleados en este problema son el resultado de un estudio del NIST que involucra la dilatación térmica del cobre. La variable respuesta es el coeficiente de dilatación térmica, y la variable independiente es la temperatura medida en grados Kelvin. Sea $\{(x_i, y_i)\}_{i=1}^{236}$ una muestra donde $x_i, y_i \in \mathbb{R}$ son los valores de las variables explicativa y respuesta, respectivamente, correspondientes al i-ésimo individuo. Sea $\boldsymbol{\theta} \in \mathbb{R}^7$ el vector de parámetros desconocidos. Consideraremos el ajuste no lineal dado por:

$$y_i = h(\boldsymbol{x_i}, \boldsymbol{\theta}) + \varepsilon_i, \quad i = 1, ..., 236,$$

donde

$$h(x, \boldsymbol{\theta}) = \frac{\theta_1 + \theta_2 x + \theta_3 x^2 + \theta_4 x^3}{1 + \theta_5 x + \theta_6 x^2 + \theta_7 x3}.$$

El problema a resolver sigue la fórmula general (4):

$$\min_{\boldsymbol{\theta}\in\mathbb{R}^7} f(\boldsymbol{\theta}) = \min_{\boldsymbol{\theta}\in\mathbb{R}^7} \frac{1}{2}\boldsymbol{r}(\boldsymbol{\theta})^T\boldsymbol{r}(\boldsymbol{\theta}) = \min_{\boldsymbol{\theta}\in\mathbb{R}^7} \frac{1}{2}\sum_{i=1}^{236}[r_i(\boldsymbol{\theta})]^2,$$

siendo $r_i(\boldsymbol{\theta}) = y_i - h(\boldsymbol{x_i}, \boldsymbol{\theta})$, $r_i : \mathbb{R}^7 \to \mathbb{R}$, $i = 1, ..., 236$.

En la Figura 2.2 representamos el ajuste benchmark tomando como valores certificados $\boldsymbol{\theta}$ =(1.0776351733e+00, -1.2269296921e-01, 4.0863750610e-03,-1.4262662514e-06, -5.7609940901e-03, 2.4053735503e-04, -1.2314450199e-07). Empleando dichos valores certificados obtenemos que $RSS(\boldsymbol{\theta})$ =1.5324382854e+00.

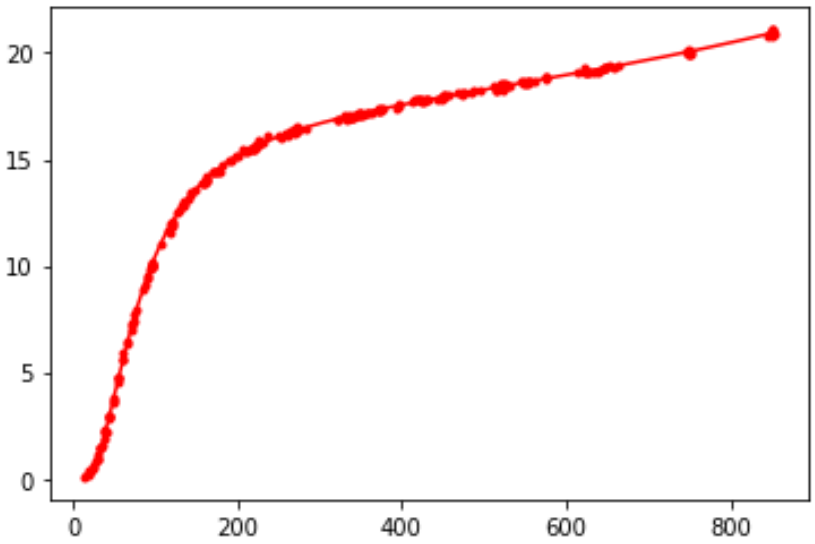

Figura 2.2: Ajuste de Hahn1 con la función del benchmark.

Al realizar la implementación de este problema utilizaremos dos iterantes iniciales distintos:

$$\boldsymbol{\theta_0^1} = (10, -1, 0.05, -0.00001, -0.05, 0.001, -0.000001),$$

$$\boldsymbol{\theta_0^2} = (1, -0.1, 0.005, -0.000001, -0.005, 0.0001, -0.0000001).$$

2.1.3. Test 3: Bennett5

Los datos empleados en este problema son el resultado de un estudio del NIST que involucra el modelado de magnetización de superconductividad, donde la variable respuesta es el magnetismo, y la variable independiente es el tiempo tomando como unidad de medida

los minutos. Sea $\{(x_i, y_i)\}_{i=1}^{154}$ una muestra donde $x_i, y_i \in \mathbb{R}$ son los valores de las variables explicativa y respuesta, respectivamente, correspondientes al i-ésimo individuo. Sea $\boldsymbol{\theta} \in \mathbb{R}^3$ el vector de parámetros desconocidos. Consideraremos el ajuste no lineal dado por:

$$y_i = h(\boldsymbol{x_i}, \boldsymbol{\theta}) + \varepsilon_i, \quad i = 1, ..., 154,$$

donde

$$h(x, \boldsymbol{\theta}) = \theta_1(\theta_2 + x)^{\frac{-1}{\theta_3}}.$$

El problema a resolver sigue la fórmula general (4):

$$\min_{\boldsymbol{\theta} \in \mathbb{R}^3} f(\boldsymbol{\theta}) = \min_{\boldsymbol{\theta} \in \mathbb{R}^3} \frac{1}{2} \boldsymbol{r}(\boldsymbol{\theta})^T \boldsymbol{r}(\boldsymbol{\theta}) = \min_{\boldsymbol{\theta} \in \mathbb{R}^3} \frac{1}{2} \sum_{i=1}^{154} [r_i(\boldsymbol{\theta})]^2,$$

siendo $r_i(\boldsymbol{\theta}) = y_i - h(\boldsymbol{x_i}, \boldsymbol{\theta})$, $r_i : \mathbb{R}^3 \to \mathbb{R}$, $i = 1, ..., 154$.

En la Figura 2.3 representamos el ajuste benchmark tomando como valores certificados $\boldsymbol{\theta} = (-2.5235058043e + 03, 4.6736564644e + 01, 9.3218483193e - 01)$. Empleando dichos valores certificados obtenemos que $RSS(\boldsymbol{\theta}) = 5.2404744073e - 04$.

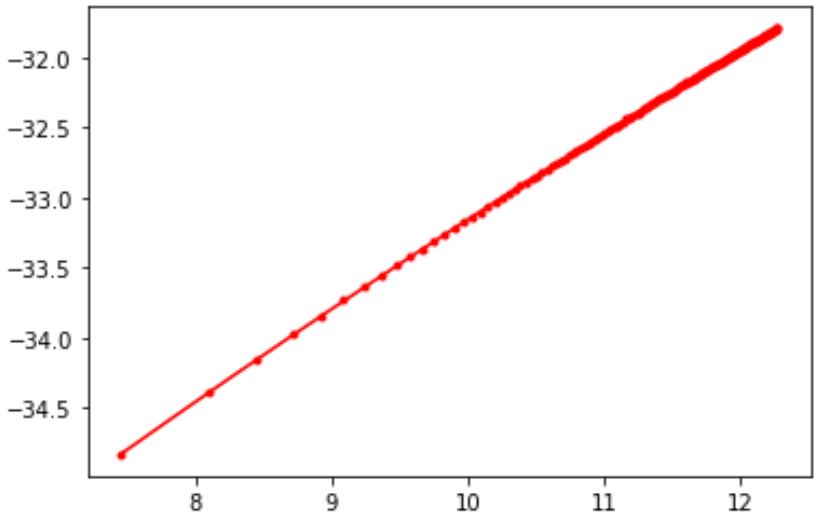

Figura 2.3: Ajuste de Bennett5 con la función del benchmark.

Al realizar la implementación de este problema utilizaremos dos iterantes iniciales distintos:

$$\boldsymbol{\theta_0^1} = (-2000, 50, 0.8),$$

$$\boldsymbol{\theta_0^2} = (-1500, 45, 0.85).$$

2.2. Resultados obtenidos

En esta sección presentaremos los resultados obtenidos al emplear los métodos numéricos descritos en el Capítulo 1. Para cada problema test se presentarán dos tablas resumen según el iterante inicial que hayamos empleado. En ellas se mostrará el tiempo de cálculo para obtener la solución, el error cometido y, por último, la suma residual de cuadrados. Se incluirán también las representaciones gráficas de los distintos ajustes realizados.

Recordamos que para los métodos de Gauss-Newton y Gauss-Newton amortiguado la implementación será propia, dicha implementación está icluída en el Apéndice C. Para el método de Levenberg-Marquardt emplearemos tres módulos distintos pertenecientes a la biblioteca libre Scipy (*least_ squares*, *curve_ fit*, *leastsq*) para comparar así su funcionamiento. Finalmente, para el método Dogbox usaremos dos módulos pertenecientes a la biblioteca libre Scipy (*least_ squares*, *curve_ fit*).

Cabe destacar que, debido a que las implementaciones de los métodos Gauss-Newton y Gauss-Newton amortiguado son propias, es esperable que requieran mayor cantidad de tiempo de cálculo para alcanzar la solución que aquellas implementaciones incluidas en bibliotecas contrastadas.

2.2.1. Resultados Test 1: Gauss1

En el Cuadro 2.1 se presentan los resultados obtenidos para el problema test 1 utilizando como iterante inicial $\boldsymbol{\theta_0^1}$. A la vista de los resultados expuestos no podemos afirmar que existan diferencias significativas de comportamiento entre los métodos utilizados, obteniendo soluciones prácticamente idénticas y muy cercanas a los valores certificados. Observamos además que, en este caso, el método que tiene un menor coste computacional es el método de Levenberg-Marquardt. Comparando las tres distintas implementaciones realizadas de este método obtenemos que con *leastsq* el tiempo se reduce a la mitad respecto a la implementación con *least_ squares*. Destaca también el elevado tiempo de cálculo requerido por el método de Gauss-Newton amortiguado, siendo diez veces más elevado que el método de Gauss-Newton. En la Figura 2.4 se representan los ajustes obtenidos donde no se aprecia diferencia entre cada una de las soluciones obtenidas.

	Tiempo	Error	RSS
Gauss-Newton	1.77925e-01 s	3.6274264199e+01	1.3158222432e+03
Gauss-Newton amortiguado	3.31221e+00 s	3.6274264199e+01	1.3158222432e+03
Levenberg-Marquardt (curve_fit)	4.22978e-03 s	3.6274264199e+01	1.3158222432e+03
Levenberg-Marquardt (least_squares)	6.31428e-03 s	3.6274264199e+01	1.3158222432e+03
Levenberg-Marquardt (leastsq)	3.02863e-03 s	3.6274264199e+01	1.3158222432e+03
Dogbox (curve_fit)	7.81941e-03 s	3.6274264199e+01	1.3158222432e+03
Dogbox (least_squares)	7.46346e-03 s	3.6274264199e+01	1.3158222432e+03

Cuadro 2.1: Resultados obtenidos para Gauss1 con el iterante inicial $\boldsymbol{\theta_0^1}$.

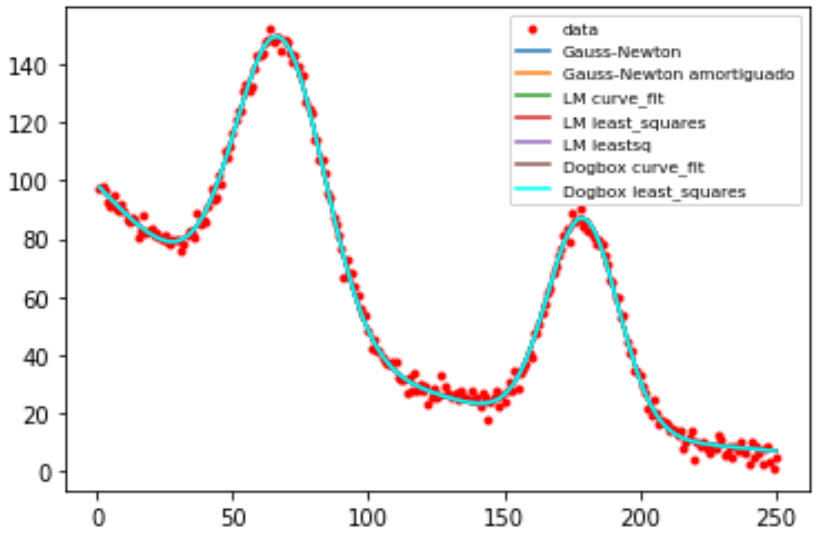

Figura 2.4: Ajustes de Gauss1 partiendo del iterante inicial $\boldsymbol{\theta_0^1}$.

En el Cuadro 2.2 se presentan los resultados obtenidos empleando como iterante inicial $\boldsymbol{\theta_0^2}$. De nuevo, las soluciones obtenidas empleando cada uno de los métodos son prácticamente idénticas y próximas a los valores certificados dados. En la Figura 2.5 se representan los ajustes obtenidos, donde no es posible apreciar diferencias entre las diferentes soluciones obtenidas. Según los resultados expuestos se tiene que el método de Levenberg-Marquardt empleando la implementación *leastsq* es el método que requiere un menor tiempo de cálculo para alcanzar la solución. Sin embargo, a diferencia de lo que ocurría al utilizar el iterante

inicial $\boldsymbol{\theta}_0^2$, las dos implementaciones restantes del médodo Levenberg-Marquardt son más lentas que las utilizadas para el método Dogbox. La implementación del método Dogbox haciendo uso de *least_ squares* es dos veces más rápida que las utilizadas para el método de Levenberg-Marquardt. Destaca, de nuevo, el elevado tiempo de cálculo requerido por el método de Gauss-Newton amortiguado

	Tiempo	Error	RSS
Gauss-Newton	3.63491e-01 s	3.6274264199e+01	1.3158222432e+03
Gauss-Newton amortiguado	3.38936e+00 s	3.6274264199e+01	1.3158222432e+03
Levenberg-Marquardt (curve_fit)	7.97653e-03 s	3.6274264199e+01	1.3158222432e+03
Levenberg-Marquardt (least_squares)	6.97708e-03 s	3.6274264199e+01	1.3158222432e+03
Levenberg-Marquardt (leastsq)	1.39413e-03 s	3.6274264199e+01	1.3158222432e+03
Dogbox (curve_fit)	5.01251e-03 s	3.6274264199e+01	1.3158222432e+03
Dogbox (least_squares)	4.00710e-03 s	3.6274264199e+01	1.3158222432e+03

Cuadro 2.2: Resultados obtenidos para Gauss1 con el iterante inicial $\boldsymbol{\theta}_0^2$.

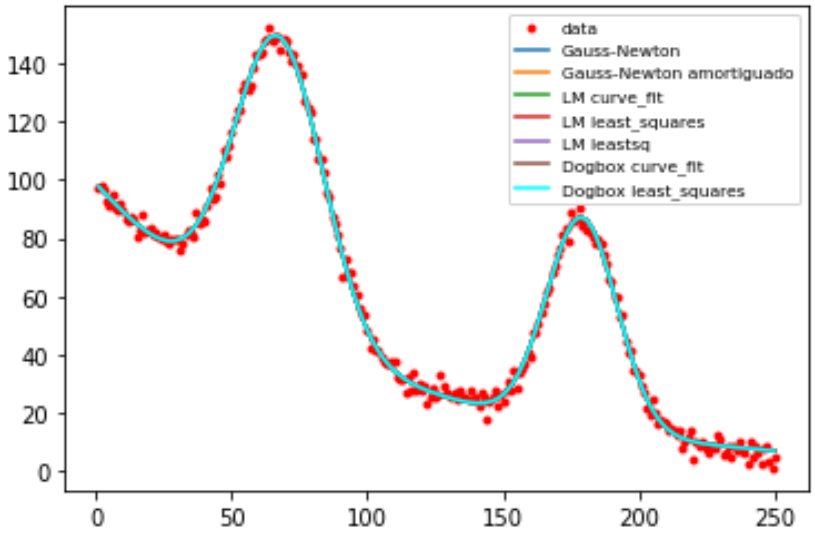

Figura 2.5: Ajustes de Gauss1 partiendo del iterante inicial $\boldsymbol{\theta}_0^2$.

A la vista de los resultados presentados en el Cuadro 2.1 y en el Cuadro 2.2 observamos

que al utilizar el módulo *leastsq* con los dos iterantes iniciales considerados, el tiempo de cálculo necesario para alcanzar la solución se reduce a la tercera parte cuando empleamos el segundo iterante inicial, $\boldsymbol{\theta}_0^2$. En ambos casos destacábamos el elevado tiempo de cálculo requerido por el método de Gauss-Newton amortiguado, siendo este diez veces mayor que el utilizado por el método de Gauss-Newton. Aunque ya comentábamos al comienzo que era esperable que ambos métodos necesitaran una mayor cantidad de tiempo para alcanzar la solución la diferencia entre ellos es muy notable. Concluimos, por lo tanto, que en este caso no es recomendable emplear el método de Gauss-Newton amortiguado frente al Gauss-Newton.

2.2.2. Resultados Test 2: Hahn1

En el Cuadro 2.3 se presentan los resultados obtenidos para el problema test 2 empleando como iterante inicial $\boldsymbol{\theta_0^1}$. Al emplear este iterante inicial el método de Gauss-Newton no llega a converger; a partir de la iteración 25 el paso comienza a oscilar no llegando nunca a encontrar la solución. Esto se debe a que al tomar este iterante inicial no podemos asegurar la convergencia local del método. En contraposición a lo que ocurría en el problema anterior, el método de Gauss-Newton amortiguado sí que supone una mejora respecto al Gauss-Newton dado que sí que llega a converger obteniendo una solución muy próxima a los valores certificados dados. De todos modos, el coste computacional continúa siendo alto. Respecto al resto de los métodos implementados no se aprecian diferencias de comportamiento. Las soluciones obtenidas son muy similares y próximas a los valores certificados propuestos, siendo el método de Levenberg-Marquardt el que alcanza dicha solución en una cantidad de tiempo menor. La implementación más eficiente del método resulta ser la que se realiza a través del módulo *leastsq*. En la figura 2.6 se representan los ajustes obtenidos donde podemos apreciar que el método de Gauss-Newton no converge a la solución.

En el Cuadro 2.4 se presentan los resultados obtenidos empleando como iterante inicial $\boldsymbol{\theta_0^2}$. En este caso, la implementación del método Gauss-Newton no ha presentado ningún problema llegando a alcanzar la solución. Aunque el método de Gauss-Newton amortiguado alcanza una solución próxima a los valores certificados el coste computacional es demasiado alto. Debido a que el método de Gauss-Newton converge no es recomendable emplear el método de Gauss-Newton amortiguado. En la figura 2.7 se representan los ajustes obtenidos. De nuevo, el método más eficiente es el método de Levenberg-Marquardt siendo la implementación más eficiente la que se realiza mendiante el módulo *leastsq*.

	Tiempo	Error	RSS
Gauss-Newton	*	*	*
Gauss-Newton amortiguado	7.297993e+00 s	1.2379169137	1.5324382853
Levenberg-Marquardt (curve_fit)	4.432158e-01 s	1.2379202026	1.5324464282
Levenberg-Marquardt (least_squares	1.306224e-02 s	1.2379169137	1.5324382853
Levenberg-Marquardt (leastsq)	3.989460e-03 s	1.2379169137	1.5324382853
Dogbox (curve_fit)	2.393556e-02 s	1.2379202026	1.5324464282
Dogbox (least_squares)	2.594066e-02 s	1.2379202026	1.5324464282

Cuadro 2.3: Resultados obtenidos para Hahn1 con el iterante inicial $\boldsymbol{\theta_0^1}$.

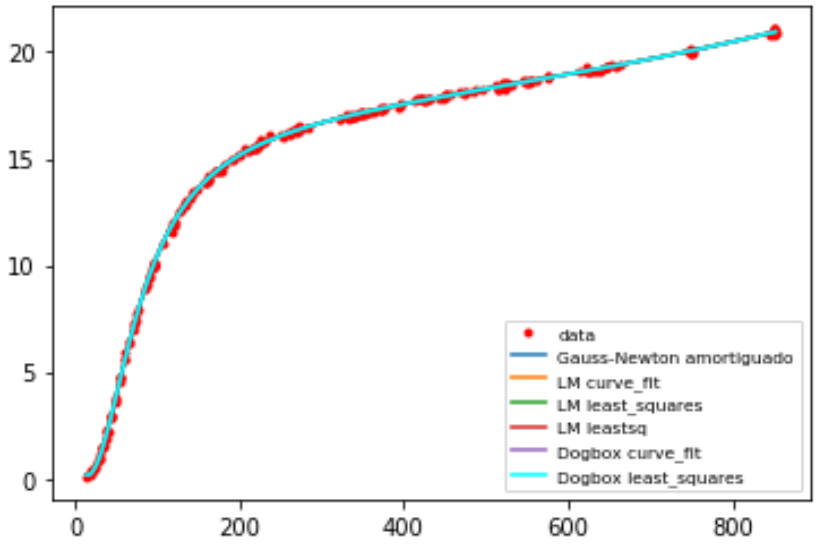

Figura 2.6: Ajustes de Hahn1 partiendo del iterante inicial $\boldsymbol{\theta_0^1}$.

Los resultados obtenidos empleando cada uno de los iterantes iniciales al utilizar el módulo *leastsq* son prácticamente idénticos alcanzando en ambos casos los valores certificados aportados en la descripción del problema en tiempos similares. Cabe destacar que al emplear el iterante inicial $\boldsymbol{\theta_0^1}$ el método de Gauss-Newton no se encuentra en las condicicones de convergencia mientras que con el iterante inicial $\boldsymbol{\theta_0^2}$ sí que converge.

	Tiempo	Error	RSS
Gauss-Newton	7.201097e-01 s	1.2379169137	1.5324382853
Gauss-Newton amortiguado	2.608703e+00 s	1.2379169137	1.5324382853
Levenberg-Marquardt (curve_fit)	4.089022e-02 s	1.2379202253	1.5324464842
Levenberg-Marquardt (least_squares)	8.764740e-03 s	1.2379169137	1.5324382853
Levenberg-Marquardt (leastsq)	3.981590e-03 s	1.2379169137	1.5324382853
Dogbox (curve_fit)	1.695466e-02 s	1.2379202253	1.5324464842
Dogbox (least_squares)	1.620960e-02 s	1.2379202253	1.5324464842

Cuadro 2.4: Resultados obtenidos para Hahn1 con el iterante inicial $\boldsymbol{\theta_0^2}$.

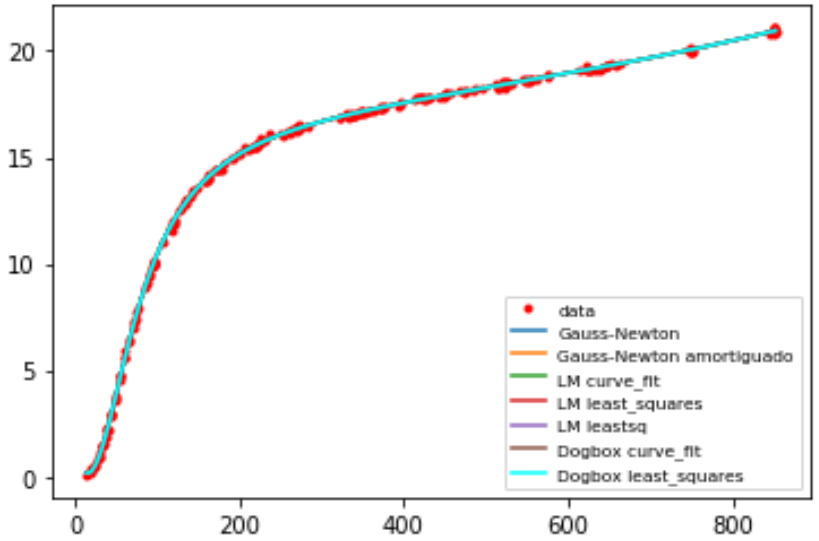

Figura 2.7: Ajustes de Hahn1 partiendo del iterante inicial $\boldsymbol{\theta_0^2}$.

2.2.3. Resultados Test 3: Bennett5

En el Cuadro 2.5 se presentan los resultados obtenidos para el problema test 3 empleando como iterante inicial $\boldsymbol{\theta_0^1}$. Al utilizar las implementaciones *leastsq* y *least_squares* el método Levenberg-Marquardt no alcanza los valores certificados que se proponen en la descripción del problema; sin embargo, al emplear el módulo *curve_fit* sí que se alcanzan dichos valores. Aunque en el Cuadro 2.5 se observa que el método que require un menor tiempo computacional es Levenberg-Marquardt empleando el módulo *leastsq* no considera-

mos que sea el más eficiente debido a que no se alcanza la solución propuesta. Por lo tanto, a la vista de los resultados expuestos, obtenemos que entre los métodos implementados que sí alcanzan los valores certificados el que tiene un menor coste computacional es el método de Levenberg-Marquardt al emplear el módulo *curve_fit*. Destaca el método de Gauss-Newton amortiguado alacanzando la solución en un tiempo muy competitivo. En la figura 2.8 se representan los ajustes obtenidos.

	Tiempo	Error	RSS
Gauss-Newton	1.585631e+00 s	2.2892082490e-02	5.2404744072e-04
Gauss-Newton amortiguado	4.452369e-01 s	2.2892082490e-02	5.2404744072e-04
Levenberg-Marquardt (curve_fit)	8.676696e-02 s	2.2892082490e-02	5.2404744073e-04
Levenberg-Marquardt (least_squares)	2.393365e-02 s	2.3478732372e-02	5.512508738e-04
Levenberg-Marquardt (leastsq)	1.595163e-02 s	2.4900365520e-02	6.200282030e-04
Dogbox (curve_fit)	1.087105e-01 s	2.2892082490e-02	5.2404744073e-04
Dogbox (least_squares)	9.129357e-02 s	2.2892082490e-02	5.2404744073e-04

Cuadro 2.5: Resultados obtenidos para Bennett5 con el iterante inicial $\boldsymbol{\theta_0^1}$.

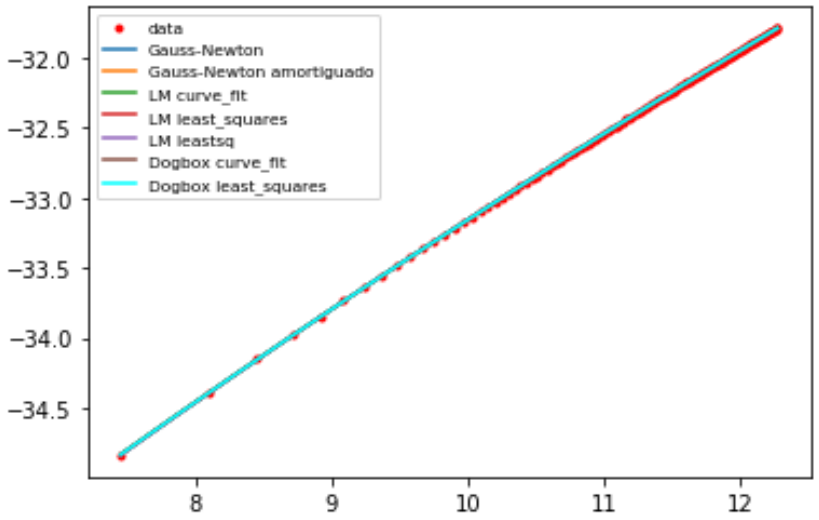

Figura 2.8: Ajustes de Bennett5 partiendo del iterante inicial $\boldsymbol{\theta_0^1}$.

En el Cuadro 2.6 se presentan los resultados obtenidos empleando como iterante inicial $\boldsymbol{\theta_0^2}$. No se aprecian diferencias de comportamiento entre los métodos utilizados alcanzándose en todos los casos los valores certificados. De nuevo, obtenemos que el método más eficiente es Levenberg-Marquardt mediante el módulo *curve_fit*. En la figura 2.9 se representan los ajustes obtenidos.

	Tiempo	Error	RSS
Gauss-Newton	1.681291e+00 s	2.2892082490e-02	5.2404744073e-04
Gauss-Newton amortiguado	2.098632e-01 s	2.2892082490e-02	5.2404744073e-04
Levenberg-Marquardt (curve_fit)	1.097035e-02 s	2.2892082490e-02	5.2404744073e-04
Levenberg-Marquardt (least_squares)	2.493477e-02 s	2.2894309265e-02	5.2414939672e-04
Levenberg-Marquardt (leastsq)	1.595736e-02 s	2.2892082491e-02	5.2404744080e-04
Dogbox (curve_fit)	1.994944e-02 s	2.2892082490e-02	5.2404744073e-04
Dogbox (least_squares)	1.499319e-02 s	2.2892082490e-02	5.2404744073e-04

Cuadro 2.6: Resultados obtenidos para Bennett5 con el iterante inicial $\boldsymbol{\theta_0^2}$.

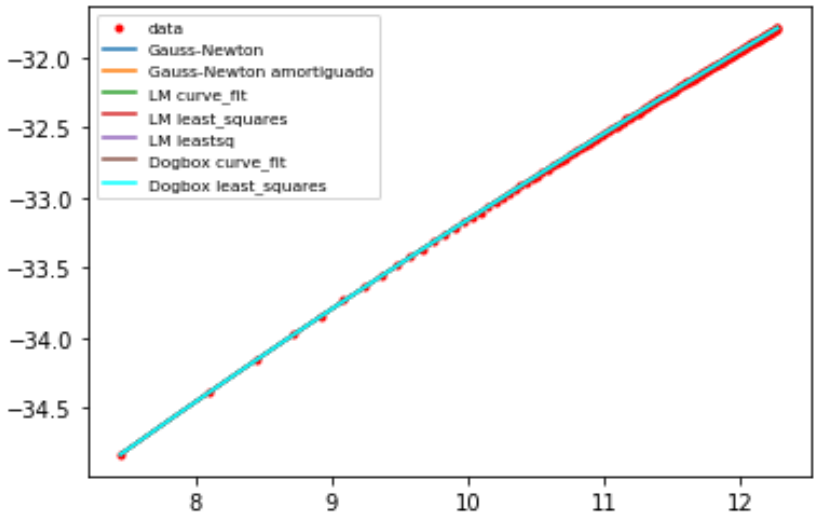

Figura 2.9: Ajustes de Bennett5 partiendo del iterante inicial $\boldsymbol{\theta_0^2}$.

Según los resultados expuestos en el Cuadro 2.5 y en el Cuadro 2.6 apreciamos que em-

pleando el método de Levenberg-Marquardt mediante el módulo *curve_fit* el coste computacional se reduce en una octava parte al utilizar como iterante inicial a $\boldsymbol{\theta_0^2}$ respecto a utilizar $\boldsymbol{\theta_0^1}$. Destaca que en ambos casos el método de Gauss-Newton amortiguado mejora al método de Gauss-Newton alcanzando la solución en un tiempo muy competitivo respecto al resto. Debemos tener siempre en consideración que ambos métodos, Gauss-Newton y Gauss-Newton amortiguado, son de implementación propia.

Capítulo 3

Conclusiones

El objetivo de este Trabajo de Fin de Grado era analizar el funcionamiento de algunos de los métodos existentes para la resolución del problema de ajuste no lineal y comparar sus distintas implementaciones presentes en *Python*.

Las conclusiones generales tras haber realizado el presente Trabajo de Fin de Grado son las siguientes:

- Se han complementado los conocimientos adquiridos en el Grado de Matemáticas impartidos en las asignaturas de Métodos Numéricos: 'Cálculo Numérico nunha Variable', 'Análise Numérica Matricial' y 'Métodos Numéricos en Optimización e Ecuacións Diferenciais'.
- Se han aprendido los fundamentos de una serie de métodos numéricos que permiten la resolución de problemas de optimización de mínimos cuadrados no lineales con el fin poder aplicar dichos conocimientos a realizar un análisis de los propios métodos, su implementación en ordenador y aplicarlos a problemas concretos.
- Se han adquirido nuevos conocimientos en el ámbito de la programación, aprendiendo un nuevo lenguaje que no se incluye en el programa formativo del Grado en Matemáticas como es *Python*.

Se han extraído también una serie de conclusiones particulares tras la realización del trabajo:

- El método de Levenberg-Marquardt aplicado en los problemas benchmark descritos en el presente Trabajo de Fin de Grado ha demostrado ser significativamente más eficiente en el cálculo de parámetros frente al resto de métodos expuestos.

- Aunque en los resultados expuestos no sea posible apreciarlo con claridad, se deduce que en los casos en los que el residuo sea pequeño o el problema no sea claramente no lineal los métodos de Gauss-Newton y Gauss-Newton amortiguado resultan ser muy competitivos. Esto no es posible apreciarlo debido a que ambos han sido implementados por la autora de este trabajo.

A juicio de la autora del presente Trabajo de Fin de Grado habría resultado interesante realizar la implementación de los métodos propuestos en el Apéndice B; en particular, el método propuesto por Fletcher-Xu.

Apéndice A

Conceptos básicos

En este apéndice expondremos una serie de conceptos básicos a tener en cuenta para una mejor comprensión del trabajo. Para ello nos guiaremos por Viaño y Burguera (2013) donde podrá encontrarse más información.

Sea $\Omega \subseteq \mathbb{R}^n$ un conjunto abierto y $f : \Omega \subseteq \mathbb{R}^n \longrightarrow \mathbb{R}$. Diremos que un punto $\boldsymbol{u} \in \Omega$ es mínimo local o relativo de f si existe $\varepsilon > 0$ tal que $B(\boldsymbol{u}, \varepsilon) \subset \Omega$ y, además, $f(\boldsymbol{u}) \leq f(\omega),\ \forall \omega \in B(\boldsymbol{u}, \varepsilon)$. Si la desigualdad es estricta diremos que es un mínimo local estricto. Diremos que se trata de un mínimo global o absoluto si $f(\boldsymbol{u}) \leq f(\omega),\ \forall \boldsymbol{\omega} \in \Omega$. De nuevo, si la desigualdad es estricta diremos que se trata de un mínimo global estricto. Por definición, el mínimo global estricto, si existe, es único.

Sea $\boldsymbol{\omega} \in \Omega$ y supongamos que f es derivable en $\boldsymbol{\omega}$. Denotamos por $\nabla f(\boldsymbol{\omega})$ al vector gradiente de f en $\boldsymbol{\omega}$. Tenemos la siguiente condición necesaria de mínimo relativo.

Teorema A.0.1. *Sea $\Omega \subseteq \mathbb{R}^n$ un conjunto abierto, no vacío y sea $f : \Omega \subseteq \mathbb{R}^n \longrightarrow \mathbb{R}$ un funcional con un mínimo relativo en $\boldsymbol{u} \in \Omega$. Si f es derivable en $\boldsymbol{u}$ entonces se verifica que $\nabla f(\boldsymbol{u}) = \theta$.*

Supongamos ahora que f es dos veces derivable en $\boldsymbol{\omega} \in \Omega$, denotamos por $\nabla^2 f(\boldsymbol{\omega})$ la matriz Hessiana de f en $\boldsymbol{\omega}$.

Teorema A.0.2. *Sea $\Omega \subseteq \mathbb{R}^n$ un conjunto abierto, no vacío y sea $f : \Omega \subseteq \mathbb{R}^n \longrightarrow \mathbb{R}$ un funcional con un mínimo relativo en $\boldsymbol{u} \in \Omega$. Se supone f derivable en un entorno de u y dos veces derivable en $\boldsymbol{u}$. Entonces se verifica:*

1. $\nabla f(\boldsymbol{u}) = \theta$.
2. *La matriz hessiana de f es semidefinida positiva.*

Veamos ahora una serie de condiciones suficientes que nos permiten afirmar la existencia de un mínimo relativo

Teorema A.0.3. *Sea $\Omega \subseteq \mathbb{R}^n$ un conjunto abierto y no vacío, $\boldsymbol{u} \in \Omega$. Sea $f : \Omega \subseteq \mathbb{R}^n \longrightarrow \mathbb{R}$ un funcional derivable en un entorno de $\boldsymbol{u}$ y tal que $\nabla f(\boldsymbol{u}) = \theta$. Entonces:*

1. *Si la función f es dos veces derivable en $\boldsymbol{u}$ y su matriz hessiana es definida positiva en $\boldsymbol{u}$ entonces la función f admite un mínimo relativo estricto en u.*

2. *Si la función f es dos veces derivable en una bola $B(\boldsymbol{u}, r) \subseteq \Omega$ de forma que la matriz hessiana es semidefinida positiva en todos los puntos de la bola entonce f admite un mínimo relativo en $\boldsymbol{u}$.*

Apéndice B

Quasi-Newton

En este apéndice se mostrarán algunos métodos numéricos más generales para la resolución de los problemas de minimización de suma de cuadrados de funciones no lineales. Estos métodos podemos incluirlos dentro del marco de los quasi-Newton. Recordemos que la estrategia a seguir era introducir una aproximación quasi-Newton del término $S(\boldsymbol{\omega}_k)$ que denotábamos por B_k de manera que

$$(B_k + J(\boldsymbol{\omega}_k)^T J(\boldsymbol{\omega}_k))\boldsymbol{s}_k = -J(\boldsymbol{\omega}_k)^T \boldsymbol{r}(\boldsymbol{\omega}_k).$$

Tomando

$$B_{k+1} = \sum_{i=1}^{m} r_i(\boldsymbol{\omega}_{k+1})(H_i)_{k+1},$$

donde $(H_i)_{k+1}$ era una aproximación de $\nabla^2 r_i(\boldsymbol{\omega}_{k+1})$, llegábamos a que B_{k+1} tenía que satisfacer

$$\begin{aligned} B_{k+1}(\boldsymbol{\omega}_{k+1} - \boldsymbol{\omega}_k) &= \sum_{i=1}^{m} (r_i(\boldsymbol{\omega}_{k+1})(H_i)_{k+1})(\boldsymbol{\omega}_{k+1} - \boldsymbol{\omega}_k) \\ &= \sum_{i=1}^{m} r_i(\boldsymbol{\omega}_{k+1})(\nabla r_i(\boldsymbol{\omega}_{k+1}) - \nabla r_i(\boldsymbol{\omega}_k)) \\ &= J(\boldsymbol{\omega}_{k+1})^T \boldsymbol{r}(\boldsymbol{\omega}_{k+1}) - J(\boldsymbol{\omega}_k)^T \boldsymbol{r}(\boldsymbol{\omega}_{k+1}) =: \boldsymbol{y}_k. \end{aligned}$$

De forma análoga, si pedimos que se verifique

$$(B_{k+1} + J(\boldsymbol{\omega}_{k+1})^T J(\boldsymbol{\omega}_{k+1}))\boldsymbol{s}_k = J(\boldsymbol{\omega}_{k+1})^T \boldsymbol{r}_{k+1} - J(\boldsymbol{\omega}_k)^T \boldsymbol{r}(\boldsymbol{\omega}_k)$$

obtenemos que B_{k+1} debe cumplir

$$B_{k+1}\boldsymbol{s}_k = J(\boldsymbol{\omega}_{k+1})^T \boldsymbol{r}(\boldsymbol{\omega}_{k+1}) - J(\boldsymbol{\omega}_k)^T \boldsymbol{r}(\boldsymbol{\omega}_k) - J(\boldsymbol{\omega}_{k+1})^T J(\boldsymbol{\omega}_{k+1})\boldsymbol{s}_k := \tilde{\boldsymbol{y}}_k.$$

De esta forma llegamos a una nueva condición quasi-Newton.

B.1. Método de Betts

Siguiendo esta estrategia, en Betts (1976), se describe un nuevo algoritmo donde se propone emplear la siguiente fórmula de rango uno

$$B_{k+1} = B_k + \frac{(\tilde{\boldsymbol{y}}_k - B_k \boldsymbol{s}_k)(\tilde{\boldsymbol{y}}_k - B_k \boldsymbol{s}_k)^T}{\boldsymbol{s}_k(\tilde{\boldsymbol{y}}_k - B_k \boldsymbol{s}_k)}.$$

Esta fórmula de rango uno ha sido reportada por varios autores, incluyendo entre ellos a Broyden, Murtagh y Sargent.

Dado que se utiliza un algoritmo recursivo de rango uno para generar B la estimación completa no estará disponible hasta que no se completen las n iteraciones. La contribución de B_k afecta a la determinación de la dirección s_k siempre que las n iteraciones hayan sido realizadas. De lo contrario, la dirección de búsqueda s_k se calcula directamente como una dirección de Gauss, lo que equivale a tomar $\nabla^2 f(\boldsymbol{\omega}_k) = J(\boldsymbol{\omega}_k)^T J(\boldsymbol{\omega}_k)$. Como ocurría con el método NL2SOL es razonable tomar en la primera iteración la aproximación de Gauss.

La aproximación de Gauss resulta algo más ventajosa si los residuos son lineales o casi nulos pero este algoritmo continúa siendo muy competitivo respecto al método de Gauss-Newton. La contribución de B se vuelve significativa para residuos no lineales y puntos donde $r_i(\boldsymbol{\omega})$ es grande. El algoritmo resulta efectivo en problemas generales que se caracterizan por residuos distintos de cero, principalmente porque la contribución de B está incluída en la estimación total de la Hessiana. En Levenberg-Marquardt ya se incluía esta matriz B pero se representaba como λI. Aunque no se llegó a probar,los resultados numéricos dados en Betts (1976) tienden a indicar que el algoritmo es cuadráticamente convergente, lo que se debe a la contribución de la matriz B.

B.2. Método de Bartholomew-Biggs

En Bartholomew-Biggs (1977) se propone un nuevo algoritmo de forma análoga al método NL2SOL expuesto en el Capítulo 1. Con respecto a la cuestión de elegir las fórmulas de actualización necesitamos que estas mantengan a la matriz B_k simétrica, pero no hay razón para suponer que deba ser definida positiva. Sin embargo, es preferible que $B_{k+1} + J^T(\boldsymbol{\omega}_{k+1})J(\boldsymbol{\omega}_{k+1})$ sí sea definida positiva; de lo contrario $(B_{k+1} + J^T(\boldsymbol{\omega}_{k+1})J(\boldsymbol{\omega}_{k+1}))s_{k+1} = J^T(\boldsymbol{\omega}_{k+1})\boldsymbol{r}(\boldsymbol{\omega}_{k+1})$ puede no dar una dirección de descenso. Las

dos modificaciones consideradas fueron la actualización de rango dos dada por Powell

$$B_{k+1} = B_k + \frac{\boldsymbol{y}_k - B_k\boldsymbol{s}_k)\boldsymbol{s}_k^T + \boldsymbol{s}_k(\boldsymbol{y}_k - B_k s_k)^T}{\boldsymbol{s}_k^T\boldsymbol{s}_k} - \frac{\boldsymbol{s}_k\boldsymbol{s}_k^T(\boldsymbol{y}_k - B_k\boldsymbol{s}_k)^T\boldsymbol{s}_k}{(\boldsymbol{s}_k^T\boldsymbol{s}_k)^2}$$

y la fórmula de rango uno propuesta por Wolfe

$$B_{k+1} = B_k + \frac{(\boldsymbol{y}_k - B_k\boldsymbol{s}_k)(\boldsymbol{y}_k - B_k\boldsymbol{s}_k)^T}{(\boldsymbol{y}_k - B_k\boldsymbol{s}_k)^T\boldsymbol{s}_k}.$$

Supongamos que la aproximación B_k es exactamente igual a $\sum_{i=1}^m r_i(\boldsymbol{\omega}_k)\nabla^2 r_i(\boldsymbol{\omega}_k)$ y que todas las subfunciones $r_i(\boldsymbol{\omega}_k)$ son cuadráticas por lo que las matrices $\nabla^2 r_i(\boldsymbol{\omega}_k)$ son constantes. Debido a que, en general, $\boldsymbol{r}_{k+1} \neq \boldsymbol{r}_k$, la verdadera matriz $\sum_{i=1}^m r_i(\boldsymbol{\omega}_{k+1})\nabla^2 r_i(\boldsymbol{\omega}_{k+1})$ difiere de B_k por una matriz de rango n como resultado de los cambios realizados en $r_i(\boldsymbol{\omega}_k)$. Por lo tanto, no se puede garantizar, simplemente mediante el uso de una actualización basada en que B_{k+1} sea correcto incluso siendo B_k correcto. Se propone entonces una estrategia de actualización basada en el caso especial de que $\boldsymbol{r}_{k+1} = \gamma_k\boldsymbol{r}_k$ para algún escalar γ_k. Si se verifican las condiciones expuestas tenemos que $\sum_{i=1}^m r_i(\boldsymbol{\omega}_{k+1})\nabla^2 r_i(\boldsymbol{\omega}_{k+1}) = \gamma_k B_k$. Si definimos $\gamma_k = \frac{\boldsymbol{r}_{k+1}^T\boldsymbol{r}_k}{\boldsymbol{r}_k^T\boldsymbol{r}_k}$ y reescalamos la matriz B_k tenemos que la matriz B_{k+1} debería reflejar los cambios realizados en $\sum_{i=1}^m r_i(\boldsymbol{\omega}_k)\nabla^2 r_i(\boldsymbol{\omega}_k)$. Se proponen entonces dos modificaciones de las fórmulas de actualización

$$B_{k+1} = \gamma_k B_k + \frac{(\boldsymbol{y}_k - \gamma_k B_k\boldsymbol{s}_k)\boldsymbol{s}_k^T + \boldsymbol{s}_k(\boldsymbol{y}_k - \gamma_k B_k s_k)^T}{\boldsymbol{s}_k^T\boldsymbol{s}_k} - \frac{\boldsymbol{s}_k\boldsymbol{s}_k^T(\boldsymbol{y}_k - \gamma_k B_k\boldsymbol{s}_k)^T\boldsymbol{s}_k}{(\boldsymbol{s}_k^T\boldsymbol{s}_k)^2}$$

y

$$B_{k+1} = \gamma_k B_k + \frac{(\boldsymbol{y}_k - \gamma_k B_k\boldsymbol{s}_k)(\boldsymbol{y}_k - \gamma_k B_k\boldsymbol{s}_k)^T}{(\boldsymbol{y}_k - \gamma_k B_k\boldsymbol{s}_k)^T\boldsymbol{s}_k}.$$

La cuestión de la mejor actualización para B_k se encuentra todavía abierta. Como mencionábamos al principio, es deseable que la actualización mantenga definida positiva la matriz $(B_{k+1} + J^T(\boldsymbol{\omega}_{k+1})J(\boldsymbol{\omega}_{k+1}))$. Sería interesante desarrollar un algoritmo que tuviera la capacidad de cambiar automáticamente de una aproximación Gauss-Newton a una aproximación basada en una de las actualizaciones dadas cuando hay indicios de que el problema a tratar es considerado difícil.

B.3. Método de Fletcher-Xu

En la Sección 1.1 hemos visto que para un problema de residuos pequeños, el método de Gauss-Newton converge a una velocidad lineal rápida que, con precisión limitada, puede ser preferible a la convergencia superlineal del método BFGS. Fletcher y Al-Baali (1985) proponen un método híbrido, HY1, intentando combinar las mejores características de ambos

métodos de manera que tomen pasos BFGS para un problema con residuo distinto de cero (*nonzero residual problem* , NZRP) y pasos de Gauss-Newton para un problema con residuo nulo (*zero residual problem* , ZRP). Al realizar una comparación numérica del método HY1 con Gauss-Newton, BFGS y otros métodos, se obtuvo que HY1 es un método muy efectivo basándose en la conjetura de que se trata de un método superlinealmente convergente. Sin embargo, en Fletcher y Xu (1987) se prueba que HY1 no retiene la propiedad de convergencia superlineal del método BFGS mediante un contraejemplo (ver Fletcher y Xu, 1987, sec. 2). Esto se debe a que el método sugerido no siempre logra distinguir entre las situaciones NZRP y ZRP. Sugieren entonces dos nuevos métodos híbridos modificando el método HY1 que sí son superlinealmente convergentes, y que describiremos a continuación.

En Fletcher y Xu (1987, pág. 382) se prueba que $\frac{f(\boldsymbol{\omega}_k)-f(\boldsymbol{\omega}_{k+1})}{f(\boldsymbol{\omega}_k)}$ puede emplearse para distiguir entre los casos NZRP y ZRP. De esta forma se sugiere el siguiente método, que denotaremos por HY2, en donde la actualización de la matriz B se hace de la forma:

$$B_{k+1} = \begin{cases} BFGS(B_k, \boldsymbol{s}_k, \tilde{\boldsymbol{y}}_k) & \text{si } \frac{f(\boldsymbol{\omega}_k)-f(\boldsymbol{\omega}_{k+1})}{f(\boldsymbol{\omega}_k)} < \varepsilon, \\ J(\boldsymbol{\omega}_{k+1})^T J(\boldsymbol{\omega}_{k+1}) & \text{otro caso,} \end{cases}$$

donde $\varepsilon \in (0,1)$. Según los resultados numéricos presentados en dicho artículo se tiene que cuando $\boldsymbol{\omega}$ está alejado de $\tilde{\boldsymbol{\omega}}$, generalmente se toman pasos de Gauss-Newton. Dichos resultados también indican que HY2 es algo más eficiente que HY1, y que la variación que existe al tomar distintos valores de ε en el rango $[0.1\ , 0.5]$ no es significativa, por lo que los autores recomiendan tomar $\varepsilon = 0.2$.

El segundo método híbrido sugerido, que denotaremos porHY3, es más similar a HY1 ya que emplea también la medida de error de la Hessiana, $\Delta(B, \boldsymbol{s}_k, \tilde{\boldsymbol{y}}_k)$. Sea $W_{k+1} = [B_{k+1}]^{-1}$, definimos

$$\Delta(B_k, \delta_k, \gamma_k) = ||W_{k+1} - B_k^{-1}||_{B_{k+1}} = (a^2 - 2b + 1)^{\frac{1}{2}}$$

donde

$$a = \frac{\tilde{\boldsymbol{y}}_k^T W_k \tilde{\boldsymbol{y}}_k}{\boldsymbol{s}_k^T \tilde{\boldsymbol{y}}_k}, \quad b = \frac{\boldsymbol{s}_k^T \tilde{\boldsymbol{y}}_k}{\boldsymbol{s}_k^T B_k \boldsymbol{s}_k}.$$

El método HY3 sugerido viene dado por

$$B_{k+1} = \begin{cases} BFGS(B_k, \boldsymbol{s}_k, \tilde{\boldsymbol{y}}_k) & \text{si } \frac{f(\boldsymbol{\omega}_k)-f(\boldsymbol{\omega}_{k+1})}{f(\boldsymbol{\omega}_k)} < \varepsilon \text{ y } \frac{\Delta(J(\boldsymbol{\omega}_{k+1})^T J(\boldsymbol{\omega}_{k+1}),\delta,\gamma)}{\Delta(J(\boldsymbol{\omega}_k)^T J(\boldsymbol{\omega}_k),\delta,\gamma)} \geq \mu, \\ J(\boldsymbol{\omega}_{k+1})^T J(\boldsymbol{\omega}_{k+1}) & \text{otro caso,} \end{cases}$$

donde $\mu \in (0,1)$.

En ambos métodos, al igual que en HY1, la matriz inicial B_0 se toma igual a la matriz $J(\boldsymbol{\omega}_0)^T J(\boldsymbol{\omega}_0)$ lo que implica que el primer paso es un paso de Gauss-Newton. En el caso en el que $J(\boldsymbol{\omega}_0)^T J(\boldsymbol{\omega}_0)$ no sea definida positiva se tomará B_0 como la matriz definida positiva más próxima a $J(\boldsymbol{\omega}_0)^T J(\boldsymbol{\omega}_0)$. En los tres métodos se define $\tilde{\boldsymbol{y}}_k := J(\boldsymbol{\omega}_{k+1})^T \boldsymbol{r}(\boldsymbol{\omega}_{k+1}) - J(\boldsymbol{\omega}_k)^T \boldsymbol{r}(\boldsymbol{\omega}_k) - J(\boldsymbol{\omega}_{k+1})^T J(\boldsymbol{\omega}_{k+1}) \boldsymbol{s}_k$. Existe la posibilidad de que $\boldsymbol{s}_k^T \tilde{\boldsymbol{y}}_k = 0$, en cuyo caso $B_{k+1} \boldsymbol{s}_k = 0$ y, por lo tanto, B_{k+1} será singular. Para evitar esto tomamos

$$\begin{cases} \tilde{\boldsymbol{y}}_k & \text{si } \boldsymbol{s}_k^T \tilde{\boldsymbol{y}}_k \geq 0.01 \boldsymbol{s}_k^T \boldsymbol{y}_k, \\ \boldsymbol{y}_k & \text{otro caso,} \end{cases}$$

en la fórmula BFGS. Recordamos que $\boldsymbol{y}_k := J(\boldsymbol{\omega}_{k+1})^T \boldsymbol{r}(\boldsymbol{\omega}_{k+1}) - J(\boldsymbol{\omega}_k)^T \boldsymbol{r}(\boldsymbol{\omega}_{k+1})$.

Se cumple que cualquiera de los métodos, HY2 y HY3, podría ser equivalente al método BFGS para el cual la convergencia global es una pregunta abierta. A partir de los resultados expuestos en Fletcher y Xu (1987) deducimos que los dos métodos son muy similares entre si (ambos son superlinealmente convergentes bajo condiciones leves cuando $\nabla^2 f(\tilde{\boldsymbol{\omega}})$ es definida positiva) pero no hay razones teóricas para preferir HY3. Para el uso práctico se recomienda emplear HY2 ya que es más simple y económico de operar que HY3.

Apéndice C

Código Python

C.1. Gauss-Newton

```
#--- Gauss-Newton
import numpy as np
----

def solve(res, jacobian, x0, tol = 1e-10, maxits = 500):
    """Gauss-Newton algorithm for solving nonlinear least squares problems.

    Parameters
    ----------
    res: evaluate the residuals of a nonlinear system for a given set of parameters.
    jaconian: evaluate the Jacobian matrix of said system for the same parameters.
    x0 : tuple, list or ndarray
    Initial guesses or starting estimates for the system.
    tol : float
    Tolerance threshold. The problem is considered solved when this value
    becomes smaller than the magnitude of the correction vector.
    Defaults to 1e-10.
    maxits : int
    Maximum number of iterations of the algorithm to perform.
    Defaults to 500.

    Return
    ------
```

```
    sol : ndarray
    Resultant values.

    """

    dx = np.ones(len(x0))   # Correction vector
    xn = np.array(x0)       # Approximation of solution
    i=0

    while  (i < maxits) and (dx[np.absolute(dx) > tol].size > 0):
        # dx = np.dot(np.linalg.pinv(jacobian(xn)), -res(xn))
        jt = np.matrix.transpose(jacobian(xn))
        dx = np.dot(np.linalg.inv(np.dot(jt,jacobian(xn))), -np.dot(jt, res(xn)))
        xn += dx            # x_{n + 1} = x_n + dx_n
        i += 1

    return xn
```

C.2. Gauss-Newton amortiguado

```
#--- Damped Gauss-Newton
import numpy as np
----

def armijo(f, J, val, direction, alpha_k, rho):
    for j in range(100):
        if ((np.linalg.norm(f(val)))**2 - (np.linalg.norm(f(val+alpha_k*direction)))**2
             >= 0.5*alpha_k*(np.linalg.norm(np.dot(J(val),direction)))**2):
            break
        else:
            alpha_k=rho*alpha_k

    return alpha_k

def solve(res, jacobian, x0, tol = 1e-10, maxits = 500):
    """Gauss-Newton algorithm for solving nonlinear least squares problems.
```

```
    Parameters
    ----------
    res: evaluate the residuals of a nonlinear system for a given set of parameters.
    jaconian: evaluate the Jacobian matrix of said system for the same parameters.
    x0 : tuple, list or ndarray
    Initial guesses or starting estimates for the system.
    tol : float
    Tolerance threshold. The problem is considered solved when this value
    becomes smaller than the magnitude of the correction vector.
    Defaults to 1e-10.
    maxits : int
    Maximum number of iterations of the algorithm to perform.
    Defaults to 500.

    Return
    ------
    sol : ndarray
    Resultant values.

    """

    dx = np.ones(len(x0))   # Correction vector
    xn = np.array(x0)       # Approximation of solution
    alpha_k=1
    i=0

    while (i < maxits) and (np.linalg.norm(alpha_k*dx) > tol):
        # dx = np.dot(np.linalg.pinv(jacobian(xn)), -res(xn))
        jt = np.matrix.transpose(jacobian(xn))
        dx = np.dot(np.linalg.inv(np.dot(jt,jacobian(xn))), -np.dot(jt, res(xn)))
        alpha_k = armijo(res, jacobian, xn, dx, 1, 0.5)
        xn += alpha_k*dx            # x_{n + 1} = x_n + alpha_k*dx_n
        i += 1

    return xn
```

Bibliografía

Bartholomew-Biggs, M. C. (1977). The estimation of the hessian matrix in nonlinear least squares problems with non-zero residual. *Mathematical Programming*, 12:67–80.

Betts, J. T. (1976). Solving the nonlinear least-square problem: Application of a general method. *Journal of Optimations Theory and Applications*, 18(4):469–483.

Bjorck, A. (1996). Least squares methods. In Ciarlet, P. G. y Lions, J. L., editors, *Handbook of Numerical Analysis, Vol. I.* North-Holland.

Dennis, J. E., Gay, D. M., y Welsch, R. E. (1981). An adaptive nonlinear least-squares algorithm. *ACM Transactions on Math. Software*, 7(3):348–368.

Fletcher, R. (1980). *Practical Methods of Optimization, Vol. 1, UnconstrainedOptimization.* John Wiley and Sons, New York.

Fletcher, R. y Al-Baali, M. (1985). Variational methods for nonlinear least squares. *Operational Res. Soc.*, 36:405–421.

Fletcher, R. y Xu, C. (1987). Hybrid methods of nonlinear least squares. *IMA Journal of Numerical Analysis*, 7:371–389.

Gill, P. E., Murray, W., y Wright, M. H. (1981). *Practical Optimization.* Academic Press, London.

Hunter, J. D. (2007). Matplotlib: A 2D Graphics Environment. *Computing in Science and Engineering*, 9:90–95.

Levenberg, K. (1944). A method for the solution of certain problems in leas squares. *Quart. Appl. Math*, 2:164–168.

Marquardt, D. W. (1963). An algorithm for least-squares estimation of nonlinear parameters. *SIAM J. Appl. Math*, 11(2):431–441.

Moré, J. J. (1978). The levenberg-marquardt algorithm: implementation and theory. In Watson, G., editor, *Lecture Notes in Mathematics 630: Numerical Analysis*, Berlín,Heidelberg. Springer-Verlag.

Moré, J. J. (1983). Recent developments in algorithms and software for trust region methods. In Bachem, A., Grotschel, M., y Korte, B., editors, *Mathematical Programming: The State of the Art*, Berlín,Heidelberg. Springer-Verlag.

Moré, J. J., Garbow, B. S., y Hilstrom, K. E. (1980). *User Guide for Minpack-1.* Argonne National Laboratory, Argonne, Illinois.

Powell, M. J. D. (1970). A new algorithm for unconstrained optimization. In Rosen, J., Mangasarian, O., y Ritter, K., editors, *Nonlinear Programming*, pages 31–65, London. Academic Pres.

Shen, V. K., Siderius, D. W., Krekelberg, W. P., y Hatch, H. W. (2013). Nist standard reference simulation website. Technical Report 173, National Institute of Standards and Technology, Gaithersburg MD, 20899. `http://doi.org/10.18434/T4M88Q`.

Sun, W. y Yuan, Y.-X. (2006). *Optimization Theory and Methods.* Springer US.

van der Walt, S., Colbert, S. C., y Varoquaux, G. (2011). The NumPy Array: A Structure for Efficient Numerical Computation. *Computing in Science and Engineering*, 13:22–30.

Viaño, J. M. y Burguera, M. (2013). *Lecciones de Métodos Numérico: 4. Optimización.* Andavira Editora, S.L., Santiago de Copostela.

Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., y Reddy, T. *et al. (2020). SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python.* Nature Methods*, 17:261–272.*

Voglis, C. y Lagaris, I. E. (2004). A rectangular trust region dogleg approach for unconstrained and bound constrained nonlinear optimization. In Proc. WSEAS International Conference on Applied Mathematics*, pages 1–7.*