Algoritmos de direcciones factibles para problemas de optimización no lineal con restricciones

Irene Echeverría Sanmartín

Trabajo de fin de grado en Matemáticas
Universidad de Zaragoza

Director del trabajo:
Pedro Mateo Collazos

Julio de 2023

Summary

This work focuses on feasible direction methods, essential techniques in solving constrained nonlinear optimization problems. By generating directions that comply with the problem's constraints and improve the objective function, these methods offer optimal solutions applicable to various disciplines such as engineering and economics. In particular, the work examines in-depth Rosen's projection method and Wolfe's reduced gradient method, providing a detailed analysis of these approaches, critically comparing them, and demonstrating their practical application through numerical examples and Python code.

As we see in chapter 1, feasible direction methods have emerged from the evolution of constrained nonlinear optimization, since Kelley's cutting-plane method in 1960 to modern techniques that combine various strategies to improve efficiency. This iterative process of moving from one feasible point to a better one is based on the Karush-Kuhn-Tucker (KKT) conditions, which integrate elements of Lagrange's multipliers method and ensure global optimality in convex problems.

By providing a deep understanding of these methods and their historical context, this work aspires to facilitate their adoption in solving optimization problems across a variety of fields, and to foster a solid understanding of the theory and practice of these approaches in the community of researchers and professionals.

In chapter 2, we analyze in depth Rosen´s gradient projection method, This method, developed in 1960 by Herbert Rosen, is a fundamental tool in the field of feasible direction methods. It relies on identifying a descent direction in the unrestricted variable space that is then projected into the restricted variable space. The contribution of this method is unquestionable, as it laid the groundwork for later optimization methods.

The essence of Rosen's gradient projection method lies on projecting the negative gradient in such a way as to improve the objective function, thus avoiding the generation of non-feasible points. To do this, use is made of projection matrices, symmetric and idempotent matrices with remarkable properties: positive semidefinite, can generate projection matrices from them, and can generate orthogonal linear subspaces.

The algorithm for Rosen's gradient projection method starts by selecting an initial feasible point and then iterates until convergence is reached or until a maximum number of iterations is reached. Each iteration consists of two steps: generating a feasible improvement direction using projection and line search by solving a one-variable nonlinear optimization problem to find a solution that improves the objective function.

Although the original version of the algorithm focuses on the case of nonlinear optimization problems with linear constraints, this method is also applicable to problems with nonlinear constraints, by modifying the projection matrix to include the gradients of these constraints. Likewise, in the present work, a modification of the algorithm is proposed to guarantee its convergence, by choosing an alternative direction if the normalized gradient is too small.

Finally, the proposed method is illustrated with an example of a nonlinear optimization problem in the objective function, for which the algorithm finds the solution in a few iterations.

In chapter 3, we deep into Wolfe's reduced gradient method. This one, proposed by Philip Wolfe in 1963 and later generalized by Abadie and Carpentier in 1969, is a robust approach within feasible direction methods for solving constrained nonlinear optimization problems. This method seeks to reduce the dimensionality of the optimization problem, by representing all variables in terms of an independent subset of them. The strategy is to decompose the solution into basic and non-basic components to construct

a feasible improvement direction, manipulating the gradient of the objective function.

The feasible direction is selected in such a way as to improve the objective function and satisfy the problem's constraints. In the work, we see that this feasible improvement direction exists if and only if the solution is not a KKT point, in other words, an optimal solution to the optimization problem.

The algorithm for this method assumes that the columns of the constraint matrix are linearly independent and that each extreme point of the feasible region has strictly positive components. It starts with an initial point that complies with the constraints. In each iteration, a search direction is generated and a line search is performed to determine the step size. The algorithm concludes when the search direction is zero, indicating that the current point is a KKT point. If this does not happen, the current point is updated and the process is repeated.

This work also provides an analysis of the convergence of the reduced gradient method to a KKT point. Relying on specific conditions related to a closed and non-empty set in $\mathbb{R}^n$ and a continuously differentiable function, it concludes that any accumulation point of a sequence generated by the reduced gradient algorithm is a KKT point.

Additionally, the resolution of an optimization problem using this method is detailed. Starting from a defined initial point, the previously described procedure is followed until the optimal solution is reached, thus demonstrating the effectiveness of the method in resolving the presented example.

Chapter 4 compares the theoretical development and Python implementation of Rosen's gradient projection and Wolfe's reduced gradient methods for solving non-linear optimization problems with constraints. The aim is to validate the understanding of the algorithms and to confirm the accuracy of the code implementations. This methods are implemented in Python in the same way as they have been theoretically exposed.

The implementations were tested on the examples proposed in the previous chapters, yielding results consistent with the theoretical ones. However, the number of iterations required in Python implementations can vary due to differences in numerical precision between theoretical and numerical approximations.

The results of the Python implementation of Rosen's method and Wolfe's method are discussed in detail, showing that the results are consistent with the theoretical solutions, despite the differences in the number of iterations required. This provides evidence that the codes are correct and effective for solving non-linear optimization problems with constraints.

Finally, in chapter 5, the advantages and disadvantages observed between both methods are commented and a list of possible continuations of the work is presented to deepen the knowledge of these and to be a starting point for future work.

Índice general

Capítulo 1

Introducción

Los métodos de direcciones factibles son una clase importante de métodos para resolver problemas de optimización no lineal con restricciones. Estos métodos se basan en la idea de generar direcciones factibles que satisfagan las restricciones del problema y que mejoren la función objetivo.

La motivación detrás del estudio y la comprensión profunda de estos métodos radica en su amplia gama de aplicaciones en diversas áreas, desde la ingeniería hasta la economía y las ciencias sociales. Estos métodos permiten a los investigadores aproximar soluciones óptimas a problemas complejos que involucran múltiples restricciones y variables en tiempo razonable. Algunas de las aplicaciones más comunes incluyen:

1. Diseño óptimo de estructuras mecánicas: Los métodos de direcciones factibles se utilizan para optimizar el diseño de estructuras mecánicas, como puentes y edificios. Estos métodos permiten a los ingenieros encontrar diseños que cumplan con todas las restricciones del problema y que minimicen el peso o maximicen la resistencia [1].

2. Planificación financiera: Estos métodos se utilizan para optimizar la planificación financiera, como la asignación óptima de recursos en una cartera diversificada, maximizando el rendimiento esperado o minimizando el riesgo [2].

3. Optimización en redes: Los métodos de direcciones factibles se utilizan para optimizar problemas en redes, como la asignación óptima de rutas o la programación óptima del transporte público. Con objetivo de minimizar el tiempo o el costo total [3].

4. Optimización en procesos químicos: Los métodos se utilizan para optimizar procesos químicos, como la producción óptima de productos químicos o la optimización de la mezcla de productos químicos [4].

En conclusión, los métodos de direcciones factibles demuestran ser una herramienta esencial en la resolución de problemas de optimización no lineal con restricciones, abarcando una amplia gama de aplicaciones en campos tan diversos como la ingeniería o la economía. A través de este trabajo, he profundizado en la comprensión de estos métodos y su relevancia, destacando cómo permiten a los investigadores y profesionales encontrar soluciones óptimas en situaciones complejas. Con este trabajo, espero contribuir a la difusión del conocimiento y fomentar la adopción de los estos métodos en la resolución de problemas de optimización en diversas áreas, lo que puede llevar a soluciones más eficientes y efectivas en una variedad de contextos.

1.1. Objetivos

El principal objetivo de este trabajo es proporcionar una revisión profunda y detallada de los métodos de direcciones factibles como herramienta para resolver problemas de optimización no lineal con

restricciones, centrándose especialmente en dos de los métodos más utilizados en la práctica: El método de proyección de Rosen y el método del gradiente reducido de Wolfe. A través de este análisis, se espera alcanzar una comprensión sólida de la teoría y aplicación de estos métodos en diversos campos donde se necesita resolver problemas de optimización no lineal con restricciones. Para lograr este objetivo, el trabajo se estructura en varias secciones, cada una con un propósito específico:

1. Introducción: En esta sección, se establecerá el contexto y la motivación necesarios para el estudio de los métodos de direcciones factibles, resaltando su importancia en la resolución de problemas de optimización no lineal y su aplicación en diversas áreas. A continuación, se proporcionará una descripción general de los métodos existentes para resolver problemas de optimización no lineal con restricciones, centrándose en los métodos de direcciones factibles e incluyendo su historia y fundamentación teórica.

2. Descripción detallada del método de proyección de Rosen: En esta parte del trabajo, se analizará a fondo el método de proyección de Rosen, incluyendo sus propiedades teóricas y algoritmo. Se presentarán un ejemplo numérico detallado para ilustrar cómo este método puede ser utilizado en la resolución de problemas de optimización no lineal con restricciones.

3. Descripción detallada del método del gradiente reducido de Wolfe: Similarmente, se describirá en detalle el método del gradiente reducido de Wolfe, incluyendo sus propiedades teóricas y algoritmo. Nuevamente, se incluirá la resolución detallada de un ejemplo numérico para ilustrar cómo este método puede ser aplicado en problemas de optimización no lineal con restricciones.

4. Implementación de los algoritmos: Este capítulo analiza la implementación en Python de los métodos de Rosen y Wolfe para resolver problemas de optimización no lineal con restricciones, comparándola con el desarrollo teórico llevado a cabo en los capítulos anteriores. En particular, mostramos que aunque el número de iteraciones puede variar, los resultados numéricos son consistentes con las soluciones teóricas, validando la efectividad y precisión de los códigos implementados.

5. Comparación de ambos métodos y conclusiones: En esta última sección, se realizará una comparación crítica de los métodos de Rosen y Wolfe, analizando sus ventajas y desventajas relativas en diferentes contextos y situaciones. Finalmente, se resumirán las principales conclusiones y se identificarán posibles áreas de investigación futura en el campo de los métodos de direcciones factibles.

Al final del trabajo, se añadirán apéndices en los que se implementarán en código Python los dos algoritmos principales descritos y se utilizarán para resolver varios problemas de optimización no lineal con restricciones.

1.2. Revisión bibliográfica

La optimización no lineal con restricciones es un campo de investigación, que como veremos posteriormente, ha experimentado un notable desarrollo a lo largo del tiempo. A lo largo de los años, se han propuesto numerosos métodos y enfoques para abordar estos problemas, y su evolución ha sido marcada por importantes hitos y avances que han dado forma al estado actual del conocimiento en este campo. En la siguiente revisión, se presenta un recorrido cronológico y coherente de los principales métodos y avances en la optimización no lineal con restricciones, poniendo de relieve las conexiones y la evolución entre ellos.

Kelley (1960) [5] propuso el método de planos de corte como uno de los primeros enfoques para resolver problemas de optimización con restricciones. Este método utiliza puntos de apoyo para aproximar el conjunto factible y encontrar la solución óptima mediante un proceso iterativo.

Poco después, Rosen (1960) [6] presentó el método de proyección del gradiente, basado en generar direcciones factibles que satisfagan las restricciones y mejoren la función objetivo. Este método fue

fundamental en la creación de la base teórica y práctica para los métodos de direcciones factibles, y sirvió como punto de partida para muchos avances y mejoras posteriores en la resolución de problemas de optimización no lineal con restricciones.

Uno de los avances clave en los métodos de direcciones factibles fue el método de proyección del gradiente de Wolfe (1969) [7], que extiende el trabajo inicial de Rosen (1960) y ofrece una alternativa más eficiente y robusta. El método de Wolfe introdujo mejoras en la selección de la dirección de búsqueda y el tamaño del paso, lo que permitieron una convergencia más rápida y precisa en comparación con el método de Rosen.

También a finales de los años 60, Fiacco y McCormick (1968) [8] desarrollaron los métodos de penalización y barrera, transformando el problema de optimización con restricciones en uno sin restricciones mediante la introducción de términos de penalización o barrera en la función objetivo.

Una década después, Powell (1978) [9] introdujo el método de Newton con restricciones, combinando el algoritmo de Newton con la consideración de restricciones en la resolución de problemas de optimización no lineal.

En la década de 1980, Karmarkar (1984) [10] revolucionó el campo de la programación lineal al proponer el método de puntos interiores, utilizando una trayectoria central dentro del conjunto factible para guiar la búsqueda de la solución óptima. Este enfoque fue posteriormente adaptado para abordar problemas de optimización no lineal con restricciones.

En la misma década se investigaron variantes y extensiones de los métodos clásicos, como el método de optimización secuencial cuadrática (SQP) propuesto por Powell (1983) [11] y refinado por Schittkowski (1985). Este enfoque combina las ideas de los métodos de Newton y la programación cuadrática para resolver problemas de optimización no lineal con restricciones.

Poco después, surgieron métodos de optimización basados en poblaciones, como los algoritmos genéticos, propuestos por Goldberg (1989) [12], y las colonias de hormigas, desarrolladas por Dorigo y Stützle (2004). Estos enfoques utilizan una población de soluciones candidatas que se actualizan iterativamente para explorar el espacio de búsqueda y encontrar la solución óptima.

En cuanto a enfoques híbridos, se han combinado y adaptado diferentes métodos para resolver problemas de optimización no lineal con restricciones de manera más eficiente y efectiva. Por ejemplo, el algoritmo de puntos interiores y barrera (IPM) desarrollado por Wright (1997) [13] combina las ventajas de los métodos de puntos interiores y barrera.

Otro enfoque híbrido es el algoritmo de optimización por enjambre de partículas con restricciones (CPSO) de Hu y Eberhart (2002) [14], que combina la idea del algoritmo de enjambre de partículas con la de los métodos de penalización para resolver problemas de optimización no lineal con restricciones.

En la década de los 2000, se propusieron métodos más sofisticados y refinados, como los métodos de búsqueda lineal y de confianza, introducidos por Conn, Gould y Toint (2000) [15], que demostraron ser efectivos para resolver problemas de optimización no lineal con restricciones. Además, Fletcher y Leyffer (2002) [16] desarrollaron algoritmos de filtrado que contribuyeron significativamente a mejorar la eficiencia y la robustez de los métodos de optimización no lineal con restricciones.

En 2006, Bonnans y Shapiro [17] introdujeron el concepto de soluciones de segundo orden. Esta idea busca proporcionar garantías más robustas de optimalidad local, superando las limitaciones de los enfoques de primer orden que eran dominantes en ese momento. La introducción de estas soluciones de segundo orden estableció un hito en la disciplina, ya que permitió mejorar tanto el rendimiento como la precisión de los métodos de direcciones factibles.

Durante la última década, la investigación en optimización no lineal ha continuado avanzando. En 2010, Andréa R. Vargas [18] propuso un enfoque que combina las ideas de los algoritmos de búsqueda lineal y los métodos de confianza. Este enfoque híbrido busca aprovechar la eficiencia inherente de los métodos de búsqueda lineal, al tiempo que incorpora la robustez que caracteriza a los métodos de confianza.

Por su parte, en 2013, Jian Li y Gonglin Yuan [19] presentaron un método que implementa una aproximación cuadrática de la función objetivo para mejorar la eficiencia en la búsqueda de soluciones en la optimización no lineal. Al utilizar esta aproximación cuadrática, se reduce drásticamente la necesidad de

calcular y actualizar las direcciones factibles, lo que resulta en una mejora significativa en el rendimiento.

En 2015, Kozma y Zilinskas [20] hicieron un aporte importante al combinar los métodos de direcciones factibles con el gradiente conjugado. Este enfoque mejoró la eficiencia de la búsqueda en el espacio de factibilidad y fue efectivo para problemas con restricciones complejas. Así, se abrió un camino para nuevas investigaciones que buscan combinaciones de diferentes métodos de optimización.

Por otro lado, en 2018, Zhou y sus colaboradores [21] introdujeron técnicas de aceleración de Nesterov a los métodos de direcciones factibles. La aceleración de Nesterov es una técnica que mejora la eficiencia de los algoritmos de optimización al acelerar su convergencia. La propuesta de Zhou y coautores permitió una convergencia más rápida a la solución óptima en los problemas de optimización no lineal con restricciones.

En conclusión, acabamos de ver cómo el campo de la optimización no lineal con restricciones ha evolucionado y se ha enriquecido a lo largo de los años, y cómo los diversos métodos y enfoques han influido y dado forma a las soluciones y técnicas actuales. Conocer esta evolución y las conexiones entre los distintos métodos permitirá un mejor entendimiento del campo y fomentará el desarrollo de nuevas soluciones y enfoques en el futuro.

1.3. Métodos de direcciones factibles

Los métodos de direcciones factibles son un enfoque para resolver problemas de optimización no lineal con restricciones basado en moverse de un punto factible a otro punto factible mejorado. Estos métodos típicamente determinan una dirección $\mathbf{d}_k$ desde un punto factible $\mathbf{x}_k$ tal que, para $\lambda > 0$ y lo suficientemente pequeño, se cumplen dos propiedades:

1. $\mathbf{x}_k + \lambda \mathbf{d}_k$ es un punto factible.
2. El valor de la función objetivo en $\mathbf{x}_k + \lambda \mathbf{d}_k$ es mejor que el valor de la función objetivo en $\mathbf{x}_k$.

Luego se resuelve un problema de optimización unidimensional para determinar la distancia de movimiento a lo largo de $\mathbf{d}_k$. Este proceso se repite para generar un nuevo punto $\mathbf{x}_{k+1}$, y el proceso continúa. Estos métodos a menudo convergen a soluciones de Karush-Kuhn-Tucker (KKT) o, a veces, a soluciones de Fritz-John (FJ). Como la factibilidad primal se mantiene durante todo el proceso de optimización, estos procedimientos, a menudo, se denominan también métodos primales.

Recordar que un punto KKT es un punto factible en el que se cumplen las condiciones KTT. Las condiciones KKT son un conjunto de condiciones necesarias que deben cumplir los puntos óptimos locales de un problema de optimización no lineal con restricciones de igualdad y desigualdad y que bajo ciertas condiciones, también son suficientes. Estas condiciones combinan elementos del método de los multiplicadores de Lagrange, condiciones de holgura complementaria y no negatividad de los multiplicadores de Lagrange asociados a las restricciones de desigualdad.

Para ilustrar, podemos suponer sin perdida de generalidad el problema de optimización:

$$\begin{aligned} &\text{Minimizar} && f(\mathbf{x}) \\ &\text{Sujeto a} && g_i(\mathbf{x}) \leq 0, \quad i = 1,...,m \\ &&& h_j(\mathbf{x}) = 0, \quad j = 1,...,l \end{aligned} \tag{1.1}$$

donde $\mathbf{x}$ es un vector de variables de decisión, $f(\mathbf{x})$ es la función objetivo no lineal que se desea minimizar, $g_i(\mathbf{x})$ son las restricciones de desigualdad no lineales, $h_j(\mathbf{x})$ son las restricciones de igualdad no lineales, m es el número de restricciones de desigualdad y p es el número de restricciones de igualdad.

Las condiciones KKT son las siguientes:

1. Condiciones de factibilidad primal:

$$g_i(\mathbf{x}) \leq 0,\ i = 1,...,m; \quad h_j(\mathbf{x}) = 0,\ j = 1,...,l.$$

2. El gradiente de la función Lagrangiana con respecto a $\mathbf{x}$ tiene que ser nulo:

   La función Lagrangiana se define como:

$$L(\mathbf{x},\mathbf{u},\mathbf{v}) = f(\mathbf{x}) + \sum_{i=1}^{m} u_i g_i(\mathbf{x}) + \sum_{j=1}^{l} v_j h_j(\mathbf{x}),$$

   donde u_i son los multiplicadores de Lagrange asociados a las restricciones de desigualdad $g_i(\mathbf{x})$ y v_j son los multiplicadores de Lagrange asociados a las restricciones de igualdad $h_j(\mathbf{x})$. Por tanto, la condición que describimos es:

$$\nabla_{\mathbf{x}} L(\mathbf{x},\mathbf{u},\mathbf{v}) = \nabla f(\mathbf{x}) + \sum_{i=1}^{m} u_i \nabla g_i(\mathbf{x}) + \sum_{j=1}^{l} v_j \nabla h_j(\mathbf{x}) = 0.$$

3. Condiciones de holgura complementaria:

$$u_i g_i(\mathbf{x}) = 0,\ i = 1,...,m.$$

   Esto implica que si $g_i(\mathbf{x}) < 0$, entonces $u_i = 0$, y si $u_i > 0$, entonces $g_i(\mathbf{x}) = 0$.

4. No negatividad de los multiplicadores de Lagrange asociados a las restricciones de desigualdad:

$$u_i \geq 0,\ i = 1,...,m.$$

Sea $\mathbf{x}^*$ un óptimo local del problema de optimización con restricciones. Supongamos que $\mathbf{x}^*$ satisface ciertas condiciones de regularidad, como que las restricciones de igualdad y desigualdad sean linealmente independientes en $\mathbf{x}^*$. La idea principal detrás de las condiciones KKT es que el gradiente de la función objetivo, $\nabla f(\mathbf{x}^*)$, debe ser una combinación lineal de los gradientes de las restricciones activas en $\mathbf{x}^*$ (restricciones de igualdad y restricciones de desigualdad que se cumplen con igualdad). Esto se debe a que, en un óptimo, no debe ser posible mejorar la función objetivo moviéndose en la dirección de las restricciones activas. En notación matemática, esto se puede expresar como:

$$\nabla f(\mathbf{x}^*) = -\sum_{i=1}^{m} u_i \nabla g_i(\mathbf{x}^*) - \sum_{j=1}^{l} v_j \nabla h_j(\mathbf{x}^*).$$

Es importante mencionar que, aunque las condiciones KKT son necesarias para la optimalidad en ciertas condiciones de regularidad, no son suficientes en general. Sin embargo, si las funciones involucradas son convexas, entonces las condiciones KKT también son suficientes para la optimalidad global.

En este primer capítulo, hemos establecido el contexto y subrayado la importancia de los métodos de direcciones factibles en la optimización no lineal con restricciones. Además, hemos realizado una revisión bibliográfica y teórica de estos métodos, presentando las condiciones KKT como herramienta fundamental en la resolución de problemas de optimización no lineal. Con una base sólida en estos conceptos esenciales, nos encontramos en una posición ideal para explorar en profundidad dos métodos de direcciones factibles destacados y altamente efectivos: el método de proyección de Rosen y el método del gradiente reducido de Wolfe.

En los capítulos 2 y 3, nos adentraremos en cada uno de estos métodos, analizando sus fundamentos teóricos y algoritmos, así como ilustrando su aplicación práctica mediante un ejemplo numérico relevante. Al finalizar esta exploración, estaremos en condiciones de comparar y contrastar estos dos enfoques.

Capítulo 2

Método de proyección del gradiente de Rosen

El método de proyección de gradiente de Rosen fue uno de los primeros métodos de direcciones factibles para resolver problemas de optimización no lineal con restricciones, desarrollado por el matemático estadounidense Herbert Rosen en 1960. En ese momento, Rosen trabajaba en el campo de la optimización no lineal y se enfrentaba a la dificultad de resolver problemas con restricciones, ya que los métodos existentes no eran eficientes para encontrar soluciones factibles. Para abordar este problema, Rosen desarrolló un método basado en la técnica de proyección.

El método de proyección de Rosen comienza por encontrar una dirección de descenso en el espacio de las variables no restringidas. Luego, se proyecta esta dirección sobre el espacio de las variables restringidas para obtener una dirección factible que satisfaga las restricciones. A continuación, se busca la longitud de paso óptima para avanzar hacia la solución.

Una de las ventajas de este método es que es fácil de implementar y funciona bien para problemas con restricciones lineales y no lineales. Sin embargo, una desventaja del método es que puede ser lento en algunos casos, especialmente cuando se tienen restricciones complicadas.

A pesar de sus limitaciones, el método de proyección de Rosen ha sido muy influyente en el desarrollo de los métodos de direcciones factibles para resolver problemas de optimización no lineal con restricciones y su enfoque de utilizar proyecciones para obtener soluciones factibles ha sido una técnica común en muchos métodos posteriores. Dos métodos basados en el enfoque de Rosen son el método de barreras de interior propuesto por Fiacco y McCormick [22], y el método de punto interior primal-dual introducido por Narendra Karmarkar [10].

2.1. Fundamentos teóricos y algoritmo

En la siguiente sección haremos una revisión de los fundamentos teóricos en los que se basa el método de proyección de gradiente de Rosen. Como sabemos,para un problema de optimización de mínimo, la dirección de máxima pendiente es la del gradiente negativo. Sin embargo, cuando hay restricciones presentes, moverse en la dirección de máxima pendiente puede llevar a puntos no factibles. Es aquí donde entra en juego el método de proyección de gradiente de Rosen, que proyecta el gradiente negativo de una manera que mejora la función objetivo mientras mantiene la factibilidad. En primer lugar, consideremos la siguiente definición de matriz de proyección.

Definición 1. Una matriz $\mathbf{P}$ $n \times n$ se dice matriz de proyección si $\mathbf{P} = \mathbf{P}^t$ y $\mathbf{PP} = \mathbf{P}$. Es decir, una matriz de proyección es una matriz simétrica e idempotente y cumple con la propiedad de que al aplicarla dos o más veces sobre un vector, el resultado es el mismo que aplicarla solo una vez.

Lema 1.

1. Si $\mathbf{P}$ es una matriz de proyección, entonces $\mathbf{P}$ es semidefinida positiva.

2. $\mathbf{P}$ es una matriz de proyección si y solo si $\mathbf{I}-\mathbf{P}$ es una matriz de proyección.

3. Si $\mathbf{P}$ es una matriz de proyección y $\mathbf{Q}=\mathbf{I}-\mathbf{P}$, entonces $\mathbf{L}=\{\mathbf{Px}:\mathbf{x}\in\mathbb{R}^n\}$ y $\mathbf{L}^\perp=\{\mathbf{Qx}:\mathbf{x}\in\mathbb{R}^n\}$ son subespacios lineales ortogonales. Además, cualquier punto $\mathbf{x}\in\mathbb{R}^n$ puede ser representado de manera única como $\mathbf{p}+\mathbf{q}$, donde $\mathbf{p}\in\mathbf{L}$ y $\mathbf{q}\in\mathbf{L}^\perp$.

Demostración. Sea $\mathbf{P}$ una matriz de proyección, y sea $\mathbf{x}\in\mathbb{R}^n$ arbitrario. Entonces $\mathbf{x}^t\mathbf{Px}=\mathbf{x}^t\mathbf{PPx}=\mathbf{x}^t\mathbf{P}^t\mathbf{Px}=\|\mathbf{Px}\|^2\geq 0$, y por lo tanto $\mathbf{P}$ es semidefinida positiva. Esto demuestra la Parte 1.

Por la definición 1, la Parte 2 es obvia.

Claramente, $\mathbf{L}$ y $\mathbf{L}^\perp$ son subespacios lineales. Note que $\mathbf{P}^t\mathbf{Q}=\mathbf{P}(\mathbf{I}-\mathbf{P})=\mathbf{P}-\mathbf{PP}=\mathbf{0}$, y por lo tanto, $\mathbf{L}$ y $\mathbf{L}^\perp$ son realmente ortogonales. Ahora, sea $\mathbf{x}$ un punto arbitrario en $\mathbb{R}^n$. Entonces $\mathbf{x}=\mathbf{Ix}=(\mathbf{P}+\mathbf{Q})\mathbf{x}=\mathbf{Px}+\mathbf{Qx}=\mathbf{p}+\mathbf{q}$, donde $\mathbf{p}\in\mathbf{L}$ y $\mathbf{q}\in\mathbf{L}^\perp$. Para mostrar la unicidad, suponga que $\mathbf{x}$ también puede ser representado como $\mathbf{x}=\mathbf{p}'+\mathbf{q}'$, donde $\mathbf{p}'\in\mathbf{L}$ y $\mathbf{q}'\in\mathbf{L}'$. Por sustracción se sigue que $\mathbf{p}-\mathbf{p}'=\mathbf{q}'-\mathbf{q}$. Dado que $\mathbf{p}-\mathbf{p}'\in\mathbf{L}$ y $\mathbf{q}'-\mathbf{q}\in\mathbf{L}^\perp$, y dado que el único punto en la intersección de $\mathbf{L}$ y $\mathbf{L}^\perp$ es el vector cero, se sigue que $\mathbf{p}-\mathbf{p}'=\mathbf{q}'-\mathbf{q}=\mathbf{0}$. Por lo tanto, la representación de $\mathbf{x}$ es única, y la prueba está completa. □

Resolución de problemas con restricciones lineales

Consideremos el siguiente problema:

$$\begin{aligned}&\text{Minimizar} && f(\mathbf{x})\\ &\text{Sujeto a} && \mathbf{Ax}\leq\mathbf{b},\\ &&& \mathbf{Qx}=\mathbf{q},\end{aligned}\tag{2.1}$$

donde $\mathbf{A}$ es una matriz $m\times n$, $\mathbf{Q}$ es una matriz $l\geq n$ dimensional, $\mathbf{b}$ es un vector m-dimensional, $\mathbf{q}$ es un vector l-dimensional y $f:\mathbb{R}^n\to\mathbb{R}$ es una función diferenciable.

El siguiente lema proporciona una caracterización de dirección factible y una condición suficiente para que un vector sea una dirección de mejora.

Lema 2. Consideremos el problema 2.1. Sea $\mathbf{x}$ una solución factible y supongamos que $\mathbf{A}_1\mathbf{x}=\mathbf{b}_1$ y $\mathbf{A}_2\mathbf{x}<\mathbf{b}_2$, donde $\mathbf{A}^t$ se descompone en $(\mathbf{A}_1^t,\mathbf{A}_2^t)$ y $\mathbf{b}^t$ se descompone en $(\mathbf{b}_1^t,\mathbf{b}_2^t)$. Entonces, un vector no nulo $\mathbf{d}$ es una dirección factible en $\mathbf{x}$ si y solo si $\mathbf{A}_1\mathbf{d}\leq 0$ y $\mathbf{Q}\,\mathbf{d}=0$. Si $\nabla f(\mathbf{x})^t\mathbf{d}<0$, entonces $\mathbf{d}$ es una dirección de mejora.

Demostración. Supongamos que $\mathbf{d}$ es una dirección factible. Entonces, para algún $\varepsilon>0$ suficientemente pequeño, $\mathbf{x}+\varepsilon\mathbf{d}$ es también factible, lo que implica que $\mathbf{A}_1\mathbf{d}\leq 0$ y $\mathbf{Q}\,\mathbf{d}=0$.

Por otro lado, si $\mathbf{A}_1\mathbf{d}\leq 0$ y $\mathbf{Q}\,\mathbf{d}=0$, entonces, para un $\varepsilon>0$ suficientemente pequeño, $\mathbf{x}+\varepsilon\mathbf{d}$ satisface las restricciones del problema, lo que implica que $\mathbf{d}$ es una dirección factible.

Si $\nabla f(\mathbf{x})^t\mathbf{d}<0$, entonces $\mathbf{d}$ es una dirección de descenso para la función objetivo y, por lo tanto, es una dirección de mejora. □

Dado un punto factible $\mathbf{x}$, la dirección de máxima pendiente es $-\nabla f(\mathbf{x})$. Sin embargo, moverse a lo largo de $-\nabla f(\mathbf{x})$ puede destruir la factibilidad. Para mantener la factibilidad, se proyecta $-\nabla f(\mathbf{x})$ de modo que nos movemos a lo largo de $\mathbf{d}=-\mathbf{P}\nabla f(\mathbf{x})$, donde $\mathbf{P}$ es una matriz de proyección adecuada. El lema 3 proporciona la forma de una matriz de proyección adecuada $\mathbf{P}$ y muestra que $-\mathbf{P}\nabla f(\mathbf{x})$ es efectivamente una dirección factible de mejora, siempre que $-\mathbf{P}\nabla f(\mathbf{x})\neq 0$.

Lema 3. Consideremos el problema 2.1. Sea $\mathbf{x}$ un punto factible tal que $\mathbf{A}_1\mathbf{x}=\mathbf{b}_1$ y $\mathbf{A}_2\mathbf{x}<\mathbf{b}_2$, donde $\mathbf{A}^t=(\mathbf{A}_1^t,\mathbf{A}_2^t)$ y $\mathbf{b}^t=(\mathbf{b}_1^t,\mathbf{b}_2^t)$. Además, supongamos que f es diferenciable en $\mathbf{x}$. Si $\mathbf{P}$ es una matriz de proyección tal que $\mathbf{P}\nabla f(\mathbf{x})\neq 0$, entonces $\mathbf{d}=-\mathbf{P}\nabla f(\mathbf{x})$ es una dirección de mejora de f en $\mathbf{x}$. Además, si $\mathbf{M}^t=(\mathbf{A}_1^t,\mathbf{Q}^t)$ tiene rango completo, y si $\mathbf{P}$ tiene la forma $\mathbf{P}=\mathbf{I}-\mathbf{M}^t(\mathbf{MM}^t)^{-1}\mathbf{M}$, entonces $\mathbf{d}$ es una dirección factible de mejora.

Demostración. Observa que

$$\nabla f(\mathbf{x})^t\mathbf{d} = -\nabla f(\mathbf{x})^t\mathbf{P}\nabla f(\mathbf{x}) = -\nabla f(\mathbf{x})^t\mathbf{P}^t\mathbf{P}\nabla f(\mathbf{x}) = -\|\mathbf{P}\nabla f(\mathbf{x})\|^2 < 0.$$

Por el lema 2, $\mathbf{d} = -\mathbf{P}\nabla f(\mathbf{x})$ es una dirección de mejora. Además, si $\mathbf{P} = \mathbf{I} - \mathbf{M}^t(\mathbf{M}\mathbf{M}^t)^{-1}\mathbf{M}$, entonces $\mathbf{M}\mathbf{d} = -\mathbf{M}\mathbf{P}\nabla f(\mathbf{x}) = 0$; es decir, $\mathbf{A}_1\mathbf{d} = 0$ y $\mathbf{Q}\mathbf{d} = 0$. Por el lema 2, $\mathbf{d}$ es una dirección factible, y la demostración está completa. □

Resolución del caso $\mathbf{P}\nabla f(\mathbf{x}) = 0$

Hemos visto que si $\mathbf{P}\nabla f(\mathbf{x}) \neq 0$, entonces $\mathbf{d} = -\mathbf{P}\nabla f(\mathbf{x})$ es una dirección factible de mejora. Ahora, supongamos que $\mathbf{P}\nabla f(\mathbf{x}) = 0$. Entonces,

$$0 = \mathbf{P}\nabla f(\mathbf{x}) = [\mathbf{I} - \mathbf{M}^t(\mathbf{M}\mathbf{M}^t)^{-1}\mathbf{M}]\nabla f(\mathbf{x}) = \nabla f(\mathbf{x}) + \mathbf{M}^t\mathbf{w} = \nabla f(\mathbf{x}) + \mathbf{A}_1^t\mathbf{u} + \mathbf{Q}^t\mathbf{v}, \tag{2.2}$$

donde $\mathbf{w} = -(\mathbf{M}\mathbf{M}^t)^{-1}\mathbf{M}\nabla f(\mathbf{x})$ y $\mathbf{w}^t = (\mathbf{u}^t, \mathbf{v}^t)$. Si $\mathbf{u} \geq 0$, entonces el punto $\mathbf{x}$ satisface las condiciones KKT y podemos detenernos. Si $\mathbf{u} \not\geq 0$, entonces, como muestra el teorema 1, se puede identificar una nueva matriz de proyección $\hat{\mathbf{P}}$ tal que $\mathbf{d} = -\hat{\mathbf{P}}\nabla f(\mathbf{x})$ es efectivamente una dirección factible de mejora.

Teorema 1. Consideremos el problema 2.1. Sea $\mathbf{x}$ una solución factible y supongamos que $\mathbf{A}_1\mathbf{x} = \mathbf{b}_1$ y $\mathbf{A}_2\mathbf{x} < \mathbf{b}_2$, donde $\mathbf{A}^t = (\mathbf{A}_1^t, \mathbf{A}_2^t)$ y $\mathbf{b}^t = (\mathbf{b}_1^t, \mathbf{b}_2^t)$. Supongamos que $\mathbf{M}^t = (\mathbf{A}_1^t, \mathbf{Q}^t)$ tiene rango completo y que $\mathbf{P} = \mathbf{I} - \mathbf{M}'(\mathbf{M}\mathbf{M}')^{-1}\mathbf{M}$. Además, supongamos que $\mathbf{P}\nabla f(\mathbf{x}) = 0$, y que $\mathbf{w} = -(\mathbf{M}\mathbf{M}')^{-1}\mathbf{M}\nabla f(\mathbf{x})$ y $(\mathbf{u}', \mathbf{v}') = \mathbf{w}'$. Si $\mathbf{u} \geq 0$, con $\mathbf{u} \not\geq 0$, sea u_j una componente negativa de $\mathbf{u}$, y sea $\hat{\mathbf{M}}' = (\hat{\mathbf{A}}_1', \mathbf{Q}')$, donde $\hat{\mathbf{A}}_1$ se obtiene de $\mathbf{A}_1$ eliminando la fila de $\mathbf{A}_1$ correspondiente a u_j. Ahora, definamos $\hat{\mathbf{P}} = \mathbf{I} - \hat{\mathbf{M}}'(\hat{\mathbf{M}}\hat{\mathbf{M}}')^{-1}\hat{\mathbf{M}}$, y sea $\mathbf{d} = -\hat{\mathbf{P}}\nabla f(\mathbf{x})$. Entonces, $\mathbf{d}$ es una dirección factible de mejora.

Demostración. Como $\mathbf{u} \not\geq 0$, sea u_j una componente negativa de $\mathbf{u}$. Definimos $\hat{\mathbf{P}}$ como en el enunciado del teorema. Primero demostramos que $\hat{\mathbf{P}}\nabla f(\mathbf{x}) \neq 0$. Por contradicción, supongamos que $\hat{\mathbf{P}}\nabla f(\mathbf{x}) = 0$. Por la definición de $\hat{\mathbf{P}}$ y dejando $\hat{\mathbf{w}} = -(\hat{\mathbf{M}}\hat{\mathbf{M}}^t)^{-1}\hat{\mathbf{P}}\nabla f(\mathbf{x})$, obtenemos

$$0 = \hat{\mathbf{P}}\nabla f(\mathbf{x}) = [\mathbf{I} - \hat{\mathbf{M}}^t(\hat{\mathbf{M}}\hat{\mathbf{M}}^t)^{-1}\hat{\mathbf{M}}]\nabla f(\mathbf{x}) = \nabla f(\mathbf{x}) + \hat{\mathbf{M}}^t\hat{\mathbf{w}}. \tag{2.3}$$

Nótese que $\mathbf{A}_1^t\mathbf{u} + \mathbf{Q}^t\mathbf{v}$ se puede escribir como $\hat{\mathbf{M}}^t\bar{\mathbf{w}} + u_j r_j^t$, donde r_j es la fila j-ésima de $\mathbf{A}_1$. Entonces, a partir de la ecuación anterior, obtenemos

$$0 = \nabla f(\mathbf{x}) + \hat{\mathbf{M}}'\bar{\mathbf{w}} + u_j r_j'. \tag{2.4}$$

Restando la ecuación 2.4 de la ecuación 2.3, se deduce que $0 = \hat{\mathbf{M}}^t(\hat{\mathbf{w}} - \bar{\mathbf{w}}) - u_j r_j^t$. Esto, junto con el hecho de que $u_j \neq 0$, viola la suposición de que $\mathbf{M}$ tiene rango completo ya que las filas de $\hat{\mathbf{M}}^t$ más las filas de r_j forman la matriz $\mathbf{M}^t$. Por lo tanto, $\hat{\mathbf{P}}\nabla f(\mathbf{x}) \neq 0$. En consecuencia, por el mismo razonamiento utilizado en lema 3, $\mathbf{d}$ es una dirección de mejora.

Ahora mostramos que $\mathbf{d}$ es una dirección factible. Nótese que $\hat{\mathbf{M}}\hat{\mathbf{P}} = 0$, de modo que

$$(\hat{\mathbf{A}}_1\mathbf{Q})\mathbf{d} = \hat{\mathbf{M}}\mathbf{d} = -\hat{\mathbf{M}}\hat{\mathbf{P}}\nabla f(\mathbf{x}) = 0. \tag{2.5}$$

Por el lema 3, $\mathbf{d}$ es una dirección factible si $\mathbf{A}_1\mathbf{d} \leq 0$ y $\mathbf{Q}\mathbf{d} = 0$. En vista de la ecuación 2.5, para demostrar que $\mathbf{d}$ es una dirección factible, basta con demostrar que $r_j\mathbf{d} \leq 0$. Premultiplicando la ecuación 2.4 por $r_j\hat{\mathbf{P}}$, y notando que $\hat{\mathbf{P}}\hat{\mathbf{M}}^t = 0$, se deduce que

$$0 = r_j\hat{\mathbf{P}}\nabla f(\mathbf{x}) + r_j\hat{\mathbf{P}}(\hat{\mathbf{M}}^t\bar{\mathbf{w}} + u_j r_j^t) = -r_j\mathbf{d} + u_j r_j\hat{\mathbf{P}}r_j^t. \tag{2.6}$$

Por el lema 1, $\hat{\mathbf{P}}$ es semidefinida positiva, de modo que $r_j\hat{\mathbf{P}}r_j^t \geq 0$. Dado que $u_j < 0$, la ecuación anterior implica que $r_j\mathbf{d} \leq 0$. Esto completa la demostración. □

Algoritmo del método de proyección del gradiente de Rosen para problemas con restricciones lineales

A continuación, resumimos el método de proyección del gradiente de Rosen para resolver problemas de optimización no lineal con restricciones de la forma 2.1. Asumimos que, para cualquier solución factible, el conjunto de restricciones vinculantes son linealmente independientes. De lo contrario, cuando las restricciones activas son dependientes, $\mathbf{M}\mathbf{M}^t$ es singular y el paso algorítmico principal no está definido. Además, en tal caso, los multiplicadores de Lagrange no son únicos y una elección arbitraria de eliminar una restricción puede hacer que el algoritmo se atasque en una solución actual no KKT.

Paso de inicialización:
$\mathbf{x}_1 \leftarrow$ Punto tal que $\mathbf{A}\mathbf{x}_1 \leq \mathbf{b}$ y $\mathbf{Q}\mathbf{x}_1 = \mathbf{q}$.
$(\mathbf{A}_1^t, \mathbf{A}_2^t), (\mathbf{b}_1^t, \mathbf{b}_2^t) \leftarrow \mathbf{A}^t, \mathbf{b}^t$ descompuestos tales que $\mathbf{A}_1\mathbf{x}_1 = \mathbf{b}_1$ y $\mathbf{A}_2\mathbf{x}_1 < \mathbf{b}_2$.
$k \leftarrow 1$.
Paso principal:
mientras $k \leq$ *número máximo de iteraciones* **hacer**
  Paso 1:
  $\mathbf{M}^t \leftarrow (\mathbf{A}_1^t, \mathbf{Q}^t)$.
  si $\boldsymbol{M} = \emptyset$ *y* $\nabla f(\boldsymbol{x}_k) = 0$ **entonces**
    Parar, $\mathbf{x}_k$ es un punto KKT.
  en otro caso
    si $\boldsymbol{M} = \emptyset$ *y* $\nabla f(\boldsymbol{x}_k) \neq 0$ **entonces**
      $\mathbf{d}_k \leftarrow -\nabla f(\mathbf{x}_k)$.
      Ir al Paso 2.
    en otro caso
      $\mathbf{P} \leftarrow \mathbf{I} - \mathbf{M}^t(\mathbf{M}\mathbf{M}^t)^{-1}\mathbf{M}$.
      $\mathbf{d}_k \leftarrow -\nabla \mathbf{P} f(\mathbf{x}_k)$.
      si $\boldsymbol{d}_k \neq 0$ **entonces**
        Ir al Paso 2.
      en otro caso
        $\mathbf{w} \leftarrow -(\mathbf{M}\mathbf{M}^t)^{-1}\mathbf{M}\nabla f(\mathbf{x}_k)$ descomponiéndose como $\mathbf{w}^t = (\mathbf{u}^t, \mathbf{v}^t)$.
        si $\boldsymbol{u} \geq 0$ **entonces**
          Parar, $\mathbf{x}_k$ es un punto KKT, con $\mathbf{w}$ generando los multiplicadores de Lagrange asociados.
        en otro caso
          Eligir una componente negativa de $\mathbf{u}$, digamos u_j, actualizar $\mathbf{A}_1$ eliminando la fila correspondiente a u_j y volver al Paso 1.

  Paso 2:
  $\lambda_k \leftarrow$ Una solución óptima del siguiente problema de búsqueda de línea:

$$\begin{aligned} &\text{Minimizar} && f(\mathbf{x}_k + \lambda \mathbf{d}_k) \\ &\text{Sujeto a} && 0 \leq \lambda \leq \lambda_{\max}, \end{aligned}$$

  donde

$$\lambda_{\max} = \begin{cases} \text{mín}\left\{ \frac{\hat{b}_i}{\hat{d}_i} : \hat{d}_i > 0 \right\} \text{ si } \hat{\mathbf{d}} \not\leq 0 \\ \infty \text{ si } \hat{\mathbf{d}} \leq 0 \end{cases}, \quad \hat{\mathbf{b}} = \mathbf{b}_2 - \mathbf{A}_2\mathbf{x}_k, \quad \hat{\mathbf{d}} = \mathbf{A}_2\mathbf{d}_k. \tag{2.7}$$

  $\mathbf{x}_{k+1} \leftarrow \mathbf{x}_k + \lambda_k \mathbf{d}_k$.
  $(\mathbf{A}_1^t, \mathbf{A}_2^t), (\mathbf{b}_1^t, \mathbf{b}_2^t) \leftarrow \mathbf{A}^t, \mathbf{b}^t$ descompuestos tales que $\mathbf{A}_1\mathbf{x}_{k+1} = \mathbf{b}_1$ y $\mathbf{A}_2\mathbf{x}_{k+1} < \mathbf{b}_2$.
  $k \leftarrow k+1$.
  Volver al paso 1.

Resolución de problemas con restricciones no lineales

Hasta ahora hemos discutido el método de proyección del gradiente para el caso de restricciones lineales. En este caso, la proyección del gradiente de la función objetivo sobre el espacio nulo de los gradientes de las restricciones vinculantes, o un subconjunto de las restricciones vinculantes, condujo a una dirección factible de mejora o a la conclusión de que se tenía un punto KKT. La misma estrategia se puede utilizar en presencia de restricciones no lineales. La única diferencia "aparente" es que en este caso la matriz $\mathbf{M}$ es la matriz cuyas filas son $\nabla g_i(\mathbf{x}_k)^t$ para $i \in I$ y $\nabla h_i(\mathbf{x}_k)^t$ para $i = 1, \ldots, p$.

Análisis de convergencia del método de proyección del gradiente

Para demostrar que el algoritmo presentado converge se necesita realizar una modificación del paso 1.

Paso 1: Sea $\mathbf{M}^t = (\mathbf{A}^t, \mathbf{Q}^t)$. Si $\mathbf{M}$ es vacío, stop si $\nabla f(\mathbf{x}_k) = \mathbf{0}$, en caso contrario, sea $\mathbf{d}_k = -\nabla f(\mathbf{x}_k)$ y continuar con el paso 2. En caso contrario, sean $\mathbf{P} = \mathbf{I} - \mathbf{M}^t(\mathbf{M}\mathbf{M}^t)^{-1}\mathbf{M}$ y $\mathbf{d}_k^I = -\mathbf{P}\nabla f(\mathbf{x}_k)$. Además, calcular $\mathbf{w} = -(\mathbf{M}\mathbf{M}^t)^{-1}\mathbf{M}\nabla f(\mathbf{x}_k)$ y sea $\mathbf{w}^t = (\mathbf{u}^t, \mathbf{v}^t)$. Si $\mathbf{u} \geq 0$, entonces stop si $\mathbf{d}_k^I = \mathbf{0}$; en caso contrario, $\mathbf{d}_k = \mathbf{d}_k^I \neq \mathbf{0}$ y continúa con el paso 2. Por otro lado, si $\mathbf{u} \ngeq 0$, sean $u_h = \text{mín}_j\{u_j\}$ y $\hat{\mathbf{M}}^t = (\hat{\mathbf{A}}_1^t, \mathbf{Q}^t)$, donde $\hat{\mathbf{A}}_1^t$ se obtiene de $\mathbf{A}_1$ eliminando la fila de $\mathbf{A}_1$ correspondiente a u_h, construir la matriz de proyección $\hat{\mathbf{P}} = \mathbf{I} - \hat{\mathbf{M}}^t(\hat{\mathbf{M}}\hat{\mathbf{M}}^t)^{-1}\hat{\mathbf{M}}$, y define $\mathbf{d}_k^{II} = -\mathbf{P}\nabla f(\mathbf{x}_k)$. Ahora, basado en una constante escalar $c > 0$, sea

$$\mathbf{d}_k = \begin{cases} \mathbf{d}_k^I & \text{si } \|\mathbf{d}_k^I\| > |u_h|c \\ \mathbf{d}_k^{II} & \text{en otro caso} \end{cases}, \tag{2.8}$$

y continúa con el paso 2.

Notar que si $\mathbf{M}$ es vacío o si $\mathbf{d}_k^I = \mathbf{0}$ en la ecuación anterior, los pasos del procedimiento son los mismos que antes. Por lo tanto, supongamos que $\mathbf{M}$ no es vacío y que $\mathbf{d}_k^I \neq \mathbf{0}$. Mientras que en el caso anterior, habríamos usado $\mathbf{d}_k = \mathbf{d}_k^I$ en este punto, ahora calculamos $\mathbf{w}$ y cambiamos a usar $\mathbf{d}_k^{II}$ en su lugar, siempre que resulte que $\mathbf{u} \ngeq 0$ y que $\|\mathbf{d}_k^I\|$ es "demasiado pequeño"según la medida $\|\mathbf{d}_k^I\| \leq |u_h|c$. En particular, si $c = 0$, entonces el Paso 1 es idéntico para los dos procedimientos. El siguiente resultado establece que, de hecho, el paso 1 que acabamos de describir genera una dirección factible que mejora.

Teorema 2. Considera la modificación anterior del paso 1 del método de proyección del gradiente. Entonces, o bien el algoritmo termina con una solución KKT en este paso, o bien genera una dirección factible que mejora.

Demostración. Por el teorema 1, si el proceso se detiene en este paso, lo hace con una solución KKT. Además, a partir de la discusión anterior, la afirmación se sigue del teorema 1 cuando $\mathbf{M}$ es vacío, o si $\mathbf{d}_k^I = \mathbf{0}$, o si $\mathbf{u} \geq 0$, o si $\|\mathbf{d}_k^I\| > |u_h|c$. Por lo tanto, supongamos que $\mathbf{M}$ no es vacío, $\mathbf{u} \ngeq 0$, y que $\mathbf{d}_k^I \neq \mathbf{0}$, pero $\|\mathbf{d}_k^I\| \leq |u_h|c$, de modo que, por 2.8, usamos $\mathbf{d}_k = \mathbf{d}_k^{II}$.

Para empezar, observe que $\mathbf{d}_k^{II} = -\mathbf{P}\nabla f(\mathbf{x}_k) \neq \mathbf{0}$, o de lo contrario, por 2.3, tendríamos $\mathbf{d}_k^I = -\mathbf{P}\nabla f(\mathbf{x}_k) = \mathbf{P}\hat{\mathbf{M}}^t\hat{\mathbf{w}} = \mathbf{0}$, ya que $\hat{\mathbf{M}}\mathbf{P}^t = \hat{\mathbf{M}}\mathbf{P} = 0$ porque $\mathbf{M}\mathbf{P} = 0$. Esto contradice $\mathbf{d}_k^I \neq \mathbf{0}$. Por lo tanto, $\hat{\mathbf{P}}\nabla f(\mathbf{x}_k) \neq \mathbf{0}$; así que por el lema 3, $\mathbf{d}_k^{II}$ es una dirección que mejora f en $\mathbf{x}_k$.

A continuación, mostraremos que $\mathbf{d}_k^{II}$ es una dirección factible. Como en la prueba del teorema 1, observando 2.5, basta con demostrar que $\mathbf{r}_h\mathbf{d}_k^{II} \leq 0$, donde $\mathbf{r}_h$ corresponde a la fila eliminada de $\mathbf{A}_1$. Como en 2.2 y 2.4, tenemos

$$\mathbf{P}\nabla f(\mathbf{x}_k) = \nabla f(\mathbf{x}_k) + \hat{\mathbf{M}}'\bar{\mathbf{w}} + u_h\mathbf{r}_h'.$$

Premultiplicando esto por $\mathbf{r}_h\hat{\mathbf{P}}$, obtenemos

$$\mathbf{r}_h\hat{\mathbf{P}}\mathbf{P}\nabla f(\mathbf{x}_k) = -\mathbf{r}_h\mathbf{d}_k^{II} + \mathbf{r}_h\hat{\mathbf{P}}\hat{\mathbf{M}}^t\bar{\mathbf{w}} + u_h\mathbf{r}_h\hat{\mathbf{P}}\mathbf{r}_h^t. \tag{2.9}$$

Dado que $\hat{\mathbf{M}}\mathbf{P} = 0$, tenemos $\hat{\mathbf{P}}\mathbf{P} = \mathbf{P}$, así que $\mathbf{r}_h\hat{\mathbf{P}}\mathbf{P} = \mathbf{r}_h\mathbf{P} = 0$ ya que $\mathbf{M}\mathbf{P} = 0$. Además, como $\hat{\mathbf{P}}\hat{\mathbf{M}}^t = 0$, por 2.9 $\mathbf{r}_h\mathbf{d}_k^{II} = u_h\mathbf{r}_h\hat{\mathbf{P}}\mathbf{r}_h^t \leq 0$, ya que $u_h < 0$, y $\hat{\mathbf{P}}$ es semidefinido positivo por el lema 1. □

2.2. Aplicación práctica

Consideremos el siguiente problema:

$$\begin{aligned} \text{Minimizar} \quad & 2x_1^2 + 2x_2^2 - 2x_1x_2 - 4x_1 - 6x_2 \\ \text{Sujeto a} \quad & x_1 + x_2 \leq 2, \\ & x_1 + 5x_2 \leq 5, \\ & -x_1 \leq 0, -x_2 \leq 0. \end{aligned}$$

Notar que $\nabla f(\mathbf{x}) = (4x_1 - 2x_2 - 4, 4x_2 - 2x_1 - 6)^t$. Vamos a resolver este problema comenzando desde el punto $(0,0)$. En cada iteración, buscamos la dirección para movernos mediante el paso 1 y luego realizamos una búsqueda lineal en esta dirección.

Iteración 1:

Dirección de búsqueda: En $\mathbf{x}_1 = (0,0)^t$, tenemos $\nabla f(\mathbf{x}_1) = (4,-6)^t$. Además, solo las restricciones de no negatividad son vinculantes en $\mathbf{x}_1$, de modo que

$$\mathbf{A}_1 = \begin{pmatrix} -1 & 0 \\ 0 & -1 \end{pmatrix}, \quad \mathbf{A}_2 = \begin{pmatrix} 1 & 1 \\ 1 & 5 \end{pmatrix}.$$

Luego tenemos

$$\mathbf{P} = \mathbf{I} - \mathbf{A}_1^t(\mathbf{A}_1\mathbf{A}_1^t)^{-1}\mathbf{A}_1 = \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}$$

y $\mathbf{d}_1 = -\mathbf{P}\nabla f(\mathbf{x}_1) = (0,0)^t$. Observando que no tenemos restricciones de igualdad en este problema:

$$\mathbf{w} = \mathbf{u} = (\mathbf{A}_1\mathbf{A}_1^t)^{-1}\mathbf{A}_1\nabla f(\mathbf{x}_1) = (-4,-6)^t.$$

Eligiendo $\mathbf{u}_4 = -6$, eliminamos el gradiente correspondiente de la cuarta restricción de $\mathbf{A}_1$. La matriz $\mathbf{A}_1$ se modifica para obtener $\hat{\mathbf{A}}_1 = (-1,0)$. La matriz de proyección modificada es entonces

$$\hat{\mathbf{P}} = \mathbf{I} - \hat{\mathbf{A}_1}^t(\hat{\mathbf{A}}_1\hat{\mathbf{A}}_1^t)^{-1}\hat{\mathbf{A}}_1 = \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix}$$

y la dirección $\mathbf{d}_1$ para movernos está dada por

$$\mathbf{d}_1 = -\hat{\mathbf{P}}\nabla f(\mathbf{x}_1) = -\begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix}\begin{pmatrix} -4 \\ -6 \end{pmatrix} = \begin{pmatrix} 0 \\ 6 \end{pmatrix}.$$

Búsqueda lineal: Cualquier punto $\mathbf{x}_2$ en la dirección $\mathbf{d}_1$ comenzando desde el punto $\mathbf{x}_1$ se puede escribir como $\mathbf{x}_2 = \mathbf{x}_1 + \lambda\mathbf{d}_1 = (0,6\lambda)^t$, y el valor de la función objetivo correspondiente es $f(\mathbf{x}_2) = 72\lambda^2 - 36\lambda$. El valor máximo de λ para el cual $\mathbf{x}_1 + \lambda\mathbf{d}_1$ es factible se obtiene de 2.8 como

$$\lambda_{\max} = \text{mín}\left\{\frac{2}{6}, \frac{5}{30}\right\} = \frac{1}{6}.$$

Por lo tanto, λ_1 es la solución óptima para el siguiente problema:

$$\begin{aligned} \text{Minimizar} \quad & 72\lambda^2 - 36\lambda \\ \text{Sujeto a} \quad & 0 \leq \lambda \leq \frac{1}{6}. \end{aligned}$$

La solución óptima es $\lambda_1 = \dfrac{1}{6}$, de modo que $\mathbf{x}_2 = \mathbf{x}_1 + \lambda_1\mathbf{d}_1 = \begin{pmatrix} 0, & 1 \end{pmatrix}^t$.

Iteración 2:

Dirección de búsqueda: En el punto $\mathbf{x}_2 = (0,1)^t$, tenemos $\nabla f(\mathbf{x}_2) = (-6,-2)^t$. Además, en este punto, las restricciones 2 y 3 son vinculantes, de modo que obtenemos

$$\mathbf{A}_1 = \begin{pmatrix} 1 & 5 \\ -1 & 0 \end{pmatrix}, \quad \mathbf{A}_2 = \begin{pmatrix} 1 & 1 \\ 0 & -1 \end{pmatrix}.$$

Luego tenemos

$$\mathbf{P} = \mathbf{I} - \mathbf{A}_1^t(\mathbf{A}_1\mathbf{A}_1^t)^{-1}\mathbf{A}_1 = \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}$$

y, por lo tanto, $-\mathbf{P}\nabla f(\mathbf{x}_2) = (0,0)^t$. Así, calculamos

$$\mathbf{u} = -(\mathbf{A}_1\mathbf{A}_1^t)\mathbf{A}_1\nabla f(\mathbf{x}_2) = \left(\frac{2}{5}, -\frac{28}{5}\right).$$

Dado que $u_3 < 0$, la fila $(-1,0)$ se elimina de $\mathbf{A}_1$, lo que da la matriz modificada $\hat{\mathbf{A}}_1 = (1,5)$. La matriz de proyección y el vector de dirección correspondiente están dados por

$$\hat{\mathbf{P}} = \mathbf{I} - \hat{\mathbf{A}_1}^t(\hat{\mathbf{A}}_1\hat{\mathbf{A}}_1^t)^{-1}\hat{\mathbf{A}}_1 = \begin{pmatrix} \frac{25}{26} & -\frac{5}{26} \\ -\frac{5}{26} & \frac{1}{26} \end{pmatrix}, \quad \mathbf{d}_2 = -\hat{\mathbf{P}}\nabla f(\mathbf{x}_2) = \left(\frac{70}{13}, \quad -\frac{14}{13}\right)^t.$$

Dado que la norma de la dirección $\mathbf{d}_2 = \left(\frac{70}{13}, -\frac{14}{13}\right)^t$ no es importante, tomamos $\mathbf{d}_2 = \left(5,-1\right)'$.

Búsqueda lineal: Nos interesan los puntos de la forma $\mathbf{x}_2 + \lambda\mathbf{d}_2 = \left(5\lambda, 1-\lambda\right)^t$ y $f(\mathbf{x}_2 + \lambda\mathbf{d}_2) = 62\lambda_2 - 28\lambda - 4$. El valor máximo de λ para el cual $\mathbf{x}_2 + \lambda\mathbf{d}_2$ es factible se obtiene de 2.8 como

$$\lambda_{\max} = \text{mín}\left\{\frac{1}{4}, \frac{1}{1}\right\} = \frac{1}{4}.$$

Por lo tanto, λ_2 es la solución al siguiente problema:

$$\begin{aligned} &\text{Minimizar} \quad 62\lambda - 28\lambda - 4 \\ &\text{Sujeto a} \quad\ 0 \le \lambda \le \frac{1}{4}. \end{aligned}$$

La solución óptima es $\lambda_2 = \frac{7}{31}$, de modo que $\mathbf{x}_3 = \mathbf{x}_2 + \lambda_2\mathbf{d}_2 = \left(\frac{35}{31}, \quad \frac{24}{31}\right)^t$.

Iteración 3:

Dirección de búsqueda: En el punto $\mathbf{x}_3 = \left(\frac{35}{31}, \frac{24}{31}\right)^t$, tenemos $\nabla f(\mathbf{x}_3) = \left(-\frac{32}{31}, -\frac{160}{31}\right)^t$. Además, la segunda restricción es vinculante, de modo que

$$\mathbf{A}_1 = \begin{pmatrix} 1 & 5 \end{pmatrix}, \quad \mathbf{A}_2 = \begin{pmatrix} 1 & 1 \\ -1 & 0 \\ 0 & -1 \end{pmatrix}.$$

Además, obtenemos

$$\mathbf{P} = \mathbf{I} - \mathbf{A}_1^t(\mathbf{A}_1\mathbf{A}_1^t)^{-1}\mathbf{A}_1 = \frac{1}{26}\begin{pmatrix} 25 & -5 \\ -5 & 1 \end{pmatrix}$$

y la dirección $\mathbf{d}_3 = -\mathbf{P}\nabla f(\mathbf{x}_3) = (0,0)^t$. Así, calculamos

$$\mathbf{u} = -(\mathbf{A}_1\mathbf{A}_1^t)\mathbf{A}_1\nabla f(\mathbf{x}_3) = \frac{32}{31} \ge 0.$$

Por lo tanto, el punto $\mathbf{x}_3$ es un punto KKT. Observamos que el gradiente de la restricción vinculante apunta en la dirección opuesta a $\nabla f(\mathbf{x}_3)$. En particular, $\nabla f(\mathbf{x}_3) + u_2\nabla g_2(\mathbf{x}_3) = 0$ para $u_2 = \frac{32}{31}$, lo que verifica que $\mathbf{x}_3$ es un punto KKT. En este ejemplo en particular, dado que f es estrictamente convexa, entonces, el punto $\mathbf{x}_3$ es efectivamente la solución óptima global del problema.

Capítulo 3

Método del gradiente reducido de Wolfe

En este capítulo, presentaremos y examinaremos en detalle otro enfoque dentro de los métodos de direcciones factibles para resolver problemas de optimización no lineal con restricciones: el método del gradiente reducido de Wolfe. Este método, propuesto por el matemático estadounidense Philip Wolfe en 1963, ofrece una alternativa al método de proyección del gradiente de Rosen, que fue el tema central del capítulo anterior.

El método del gradiente reducido de Wolfe se basa en la idea de reducir la dimensionalidad del problema, representando todas las variables en función de un subconjunto independiente de las mismas. Al hacerlo, se simplifica el problema original y se facilita la búsqueda de direcciones factibles de mejora. Inicialmente, Wolfe desarrolló este método para abordar problemas de programación no lineal con restricciones lineales. Sin embargo, más tarde, en 1969, Abadie y Carpentier generalizaron el método para manejar restricciones no lineales, ampliando así su aplicabilidad en una variedad de contextos y problemas de optimización.

En este capítulo 3, siguiendo la estructura del capítulo anterior, primero describiremos los fundamentos teóricos en los que se basa el método del gradiente reducido de Wolfe, proporcionando una base sólida para comprender su funcionamiento y sus propiedades. A continuación, enunciaremos el algoritmo asociado a este método y lo aplicaremos a un ejemplo concreto, lo cual nos permitirá ilustrar su aplicación práctica y facilitar su comprensión.

3.1. Fundamentos teóricos y algoritmo

Consideremos el problema

$$\begin{array}{ll} \text{Minimizar} & f(\mathbf{x}) \\ \text{Sujeto a} & \mathbf{A}\mathbf{x} = \mathbf{b}, \\ & \mathbf{x} \geq 0. \end{array} \tag{3.1}$$

donde $\mathbf{A}$ es una matrix $m \times n$ de rango m, b es un m-vector y f es una función continuamente diferenciable en $\mathbb{R}^n$. Hagamos la siguiente suposición de no degeneración: Cualquier conjunto de m columnas de $\mathbf{A}$ son linealmente independientes, y cada punto extremo de la región factible tiene m variables estrictamente positivas. Con esta suposición, cada solución factible tiene al menos m componentes positivas y, como máximo, $n-m$ componentes cero.

Sea $\mathbf{x}$ una solución factible. Por la suposición de no degeneración, $\mathbf{A}$ se puede descomponer en $(\mathbf{B},\mathbf{N})^*$ y $\mathbf{x}^t$ en $(\mathbf{x}_B^t, \mathbf{x}_N^t)$, donde $\mathbf{B}$ es una matriz invertible $m \times m$ y $\mathbf{x}_B > 0$. Aquí $\mathbf{x}_B$ se dice vector básico, y cada uno de sus componentes es estrictamente positiva. Los componentes del vector no básico $\mathbf{x}_N$ pueden ser positivas o cero. Sea $\nabla f(\mathbf{x})^t = (\nabla_{\mathbf{B}} f(\mathbf{x})^t, \nabla_{\mathbf{N}} f(\mathbf{x})^t)$, donde $\nabla_{\mathbf{B}} f(\mathbf{x})$ es el gradiente de f con respecto al vector básico $\mathbf{x}_B$, y $\nabla_{\mathbf{N}} f(\mathbf{x})$ es el gradiente de f con respecto al vector no básico $\mathbf{x}_N$. Recordar que una dirección $\mathbf{d}$ es una dirección factible de mejora de f en $\mathbf{x}$ si $\nabla f(\mathbf{x})^t \mathbf{d} < 0$, y si $\mathbf{A}\mathbf{d} = 0$ con $d_j \geq 0$ si $x_j = 0$. Ahora construimos un vector $\mathbf{d}$ que satisface estas propiedades y por tanto será una dirección

* Una descomposición similar ya se utilizó en la asignatura Investigación Operativa del tercer curso del grado de Matemáticas, como herramienta principal en el desarrollo del algoritmo Simplex para problemas de programación lineal.

factible de mejora. Primero, $\mathbf{d}^t$ se descompone en $(\mathbf{d}_B^t, \mathbf{d}_N^t)$. Notar que $0 = \mathbf{Ad} = \mathbf{Bd}_B + \mathbf{Nd}_N$ se cumple automáticamente si, para cualquier $\mathbf{d}_N$, $\mathbf{d}_B = -\mathbf{B}^{-1}\mathbf{Nd}_N$. Sea $\mathbf{r}^t = (\mathbf{r}_B^t, \mathbf{r}_N^t) = \nabla f(\mathbf{x})' - \nabla_\mathbf{B} f(\mathbf{x})^t \mathbf{B}^{-1}\mathbf{A} = (0, \nabla_\mathbf{N} f(\mathbf{x})^t - \nabla_\mathbf{B} f(\mathbf{x})^t \mathbf{B}^{-1}\mathbf{N})$ el gradiente reducido, y examinemos el término $\nabla f(\mathbf{x})^t \mathbf{d}$:

$$\nabla f(\mathbf{x})^t \mathbf{d} = \nabla_\mathbf{B} f(\mathbf{x})^t \mathbf{d}_B + \nabla_\mathbf{N} f(\mathbf{x})^t \mathbf{d}_N = (\nabla_\mathbf{N} f(\mathbf{x})^t - \nabla_\mathbf{B} f(\mathbf{x})^t \mathbf{B}^{-1}\mathbf{N})\mathbf{d}_N = \mathbf{r}_N^t \mathbf{d}_N.$$

Debemos elegir $\mathbf{d}_N$ de tal manera que $\mathbf{r}_N^t \mathbf{d}_N < 0$ y que $d_j \geq 0$ si $x_j = 0$.

Para ello, vamos a basarnos en el siguiente criterio. Para cada componente no básica j, $d_j = -r_j$ si $r_j \leq 0$, y $d_j = -x_j r_j$ si $r_j > 0$. Esto garantiza que $d_j \geq 0$ si $x_j = 0$, y evita tamaños de paso demasiado pequeños cuando $r_j > 0$, y $x_j > 0$ pequeño. Además, $\nabla f(\mathbf{x})^t \mathbf{d} \leq 0$, donde la desigualdad estricta se cumple si $\mathbf{d}_N \neq 0$.

En resumen, hemos descrito un procedimiento para construir una dirección factible de mejora. Este hecho, así como el hecho de que $\mathbf{d} = 0$ si y solo si $\mathbf{x}$ es un punto KKT, se demuestran en el teorema 3.

Teorema 3. Consideremos el problema 3.1. Sea $\mathbf{x}$ una solución factible tal que $\mathbf{x}^t = (\mathbf{x}_B^t, \mathbf{x}_N^t)$ y $\mathbf{x}_B > 0$, donde $\mathbf{A}$ se descompone en $(\mathbf{B}, \mathbf{N})$ y $\mathbf{B}$ es una matriz invertible $m \times m$. Sea $\mathbf{r}^t = \nabla f(\mathbf{x})^t - \nabla_B f(\mathbf{x})^t \mathbf{B}^{-1}\mathbf{A}$. Sea $\mathbf{d}^t = (\mathbf{d}_B^t, \mathbf{d}_N^t)$ la dirección formada de la siguiente manera: Para cada componente no básica j, $d_j = -r_j$ si $r_j \leq 0$ y $d_j = -x_j r_j$ si $r_j > 0$, y $\mathbf{d}_B = -\mathbf{B}^{-1}\mathbf{Nd}_N$. Si $\mathbf{d} \neq 0$, entonces $\mathbf{d}$ es una dirección factible de mejora. En otro caso, $\mathbf{d} = 0$ si y solo si $\mathbf{x}$ es un punto KKT.

Demostración. En primer lugar, notar que $\mathbf{d}$ es una dirección factible si y solo si $\mathbf{Ad} = 0$, y $d_j \geq 0$ si $x_j = 0$ para $j = 1, \ldots, n$. Por la definición de $\mathbf{d}_B$, $\mathbf{Ad} = \mathbf{Bd}_B + \mathbf{Nd}_N = \mathbf{B}(-\mathbf{B}^{-1}\mathbf{NdN}) + \mathbf{NdN} = 0$. Si x_j es básica, entonces $x_j > 0$. Si x_j no es básica, entonces d_j podría ser negativa solo si $x_j > 0$. Por lo tanto, $d_j \geq 0$ si $x_j = 0$, y por lo tanto, $\mathbf{d}$ es una dirección factible. Además,

$$\nabla f(\mathbf{x})^t \mathbf{d} = \nabla_\mathbf{B} f(\mathbf{x})^t \mathbf{d}_B + \nabla_\mathbf{N} f(\mathbf{x})^t \mathbf{d}_N = (\nabla_\mathbf{N} f(\mathbf{x})^t - \nabla_\mathbf{B} f(\mathbf{x})^t \mathbf{B}^{-1}\mathbf{N})\mathbf{d}_N = \sum_{j \notin I} r_j d_j,$$

donde I es el conjunto de índices de variables básicas. Teniendo en cuenta la definición de d_j, es obvio que $\mathbf{d} = 0$ o $\nabla f(\mathbf{x})^t \mathbf{d} < 0$. En este último caso, por el lema 2, $\mathbf{d}$ es efectivamente una dirección factible de mejora. Observe que $\mathbf{x}$ es un punto KKT si y solo si existen vectores $\mathbf{u}^t = (\mathbf{u}_B^t, \mathbf{u}_N^t) \geq (0,0)$ y $\mathbf{v}$ tales que

$$(\nabla_\mathbf{B} f(\mathbf{x})^t, \nabla_\mathbf{N} f(\mathbf{x})^t) + \mathbf{v}^t(\mathbf{B}, \mathbf{N}) - (\mathbf{u}_B^t, \mathbf{u}_N^t) = (0,0), \quad \mathbf{u}_B^t \mathbf{x}_B = 0, \quad \mathbf{u}_N^t \mathbf{x}_N = 0. \tag{3.2}$$

Dado que $\mathbf{x}_B > 0$ y $\mathbf{u}_B \geq 0$, entonces $\mathbf{u}_B^t \mathbf{x}_B = 0$ si y solo si $\mathbf{u}_B = 0$. De la primera ecuación en 3.2, se deduce que $\mathbf{v}^t = -\nabla_\mathbf{B} f(\mathbf{x})^t \mathbf{B}^{-1}$. Sustituyendo en la segunda ecuación en 3.2, se deduce que $\mathbf{u}_N^t = \nabla_\mathbf{N} f(\mathbf{x})^t - \nabla_\mathbf{B} f(\mathbf{x})^t \mathbf{B}^{-1}\mathbf{N}$. En otras palabras, $\mathbf{u}_N = \mathbf{r}_N$. Por lo tanto, las condiciones KKT se reducen a $\mathbf{r}_N \geq 0$ y $\mathbf{r}_N^t \mathbf{x}_N = 0$. Por la definición de $\mathbf{d}$, sin embargo, $\mathbf{d} = 0$ si y solo si $\mathbf{r}_N \geq 0$ y $\mathbf{r}_N^t \mathbf{x}_N = 0$. Por lo tanto, $\mathbf{x}$ es un punto KKT si y solo si $\mathbf{d} = 0$, y la prueba está completa. □

Algoritmo del método del gradiente reducido de Wolfe

A continuación, se presenta el algoritmo de gradiente reducido de Wolfe para resolver un problema de optimización no lineal con restricciones de la forma de 3.1. Supongamos que las m columnas de $\mathbf{A}$ son linealmente independientes y que cada punto extremo de la región factible tiene m componentes estrictamente positivas. Como veremos en breve, el algoritmo converge a un punto KKT, siempre que las variables básicas se elijan como las m variables mayores, rompiendo empates arbitrariamente.

Paso de inicialización:
$\mathbf{x}_1 \leftarrow$ Punto tal que $\mathbf{A}\mathbf{x}_1 = \mathbf{b}$ y $\mathbf{x}_1 \geq 0$.
$k \leftarrow 1$.
Paso principal:
mientras $k \leq$ *número máximo de iteraciones* **hacer**

Paso 1:

$$I_k \leftarrow \text{ conjunto de índices de las } m \text{ componentes más grandes de } \mathbf{x}_k \tag{3.3}$$

$$\mathbf{B} \leftarrow \{\mathbf{a}_j : j \in I_k\}, \quad \mathbf{N} \leftarrow \{\mathbf{a}_j : j \notin I_k\} \tag{3.4}$$

$$\mathbf{r}^t \leftarrow \nabla f(\mathbf{x}_k)' - \nabla_{\mathbf{B}} f(\mathbf{x}_k)' \mathbf{B}^{-1}\mathbf{A} \tag{3.5}$$

$$\mathbf{d}_j \leftarrow \begin{cases} -r_j & \text{si } j \notin I_k \text{ y } r_j \leq 0, \\ -x_j d_j & \text{si } j \notin I_k \text{ y } r_j > 0 \end{cases} \tag{3.6}$$

$$\mathbf{d}_{\mathbf{B}} \leftarrow -\mathbf{B}^{-1}\mathbf{N}\mathbf{d}_{\mathbf{N}} \tag{3.7}$$

$\mathbf{d}_k^t \leftarrow (\mathbf{d}_{\mathbf{B}}^t, \mathbf{d}_{\mathbf{N}}^t)$, donde $\mathbf{d}_{\mathbf{N}}$ y $\mathbf{d}_{\mathbf{B}}$ se obtienen a continuación de las ecuaciones 3.6 y 3.7, respectivamente.
si $d_k = 0$ **entonces**
Parar, $\mathbf{x}_k$ es un punto KKT. Observar que los multiplicadores de Lagrange asociados a $\mathbf{A}\mathbf{x} = \mathbf{b}$ y $\mathbf{x} \geq 0$ son, respectivamente, $\nabla_{\mathbf{B}} f(\mathbf{x}_k)^t \mathbf{B}^{-1}$ y $\mathbf{r}$. STOP

Paso 2:
$\lambda_k \leftarrow$ Una solución óptima del siguiente problema de búsqueda de línea:

$$\begin{aligned} &\text{Minimizar} && f(\mathbf{x}_k + \lambda \mathbf{d}_k) \\ &\text{Sujeto a} && 0 \leq \lambda \leq \lambda_{\max}, \end{aligned}$$

donde

$$\lambda_{\max} = \begin{cases} \min\limits_{1 \leq j \leq n} \left\{ -\dfrac{x_{jk}}{d_{jk}} : d_{jk} < 0 \right\} & \text{si } \mathbf{d}_k \ngeq 0 \\ \infty & \text{si } \mathbf{d}_k \geq 0 \end{cases} \tag{3.8}$$

y x_{jk}, d_{jk} son las componentes j-ésimas de $\mathbf{x}_k$ y $\mathbf{d}_k$, respectivamente.
$\mathbf{x}_{k+1} \leftarrow \mathbf{x}_k + \lambda_k \mathbf{d}_k$.
$k \leftarrow k+1$.
Volver al Paso 1.

Análisis de convergencia del método del gradiente reducido

A continuación, analizaremos la convergencia del método del gradiente reducido a un punto KKT a través del teorema 4. Para ello, emplearemos un argumento de contradicción que establece una secuencia que satisface las condiciones 1 a 4 del siguiente lema.

Lema 4. Sea S un conjunto cerrado no vacío en $\mathbb{R}^n$ y sea f:$\mathbb{R}^n \to \mathbb{R}$ continuamente diferenciable. Consideremos el problema

$$\begin{aligned} &\text{Minimizar} && f(\mathbf{x}) \\ &\text{Sujeto a} && \mathbf{x} \in S. \end{aligned}$$

Además, consideremos cualquier algoritmo de direcciones factibles cuyo mapa $\mathbf{A} = \mathbf{M}\mathbf{D}$ se define de la siguiente manera. Dado $\mathbf{x}$, $(\mathbf{x}, \mathbf{d}) \in \mathbf{D}(\mathbf{x})$ significa que $\mathbf{d}$ es una dirección factible de mejora de f en $\mathbf{x}$. Además, $\mathbf{y} \in \mathbf{M}(\mathbf{x}, \mathbf{d})$ significa que $\mathbf{y} = \mathbf{x} + \bar{\lambda}\mathbf{d}$, donde $\bar{\lambda}$ resuelve el siguiente problema de búsqueda

en línea:

$$\begin{array}{ll} \text{Minimizar} & f(\mathbf{x}+\lambda\mathbf{d}) \\ \text{Sujeto a} & \lambda \geq 0, \\ & \mathbf{x}+\lambda\mathbf{d} \in S. \end{array}$$

Sean $\{\mathbf{x}_k\}$ cualquier secuencia generada por dicho algoritmo y $\{\mathbf{d}_k\}$ la secuencia correspondiente de direcciones. Entonces no puede existir una subsecuencia $\{(\mathbf{x}_k,\mathbf{d}_k)\}_{\mathscr{K}}$ que satisfaga simultaneamente las siguientes propiedades:

1. $\mathbf{x}_k \to \mathbf{x}$ para $k \in \mathscr{K}$.
2. $\mathbf{d}_k \to \mathbf{d}$ para $k \in \mathscr{K}$.
3. $\mathbf{x}_k + \lambda\mathbf{d}_k \in S$ para todo $\lambda \in [0,\delta]$ y para cada $k \in \mathscr{K}$ para algún $\delta > 0$.
4. $\nabla f(\mathbf{x})^t\mathbf{d} < 0$.

Demostración. Supongamos, por contradicción, que existe una subsecuencia $\{(\mathbf{x}_k,\mathbf{d}_k)\}_{\mathscr{K}}$ que satisface las condiciones 1 a 4. Por la Condición 4, existe un $\varepsilon > 0$ tal que $\nabla f(\mathbf{x})^t\mathbf{d} = -2\varepsilon < 0$. Dado que $\mathbf{x}_k \to \mathbf{x}$ y $\mathbf{d}_k \to \mathbf{d}$ para $k \in \mathscr{K}$ y dado que f es diferenciable continuamente, existe un $\delta' > 0$ tal que

$$\nabla f(\mathbf{x}_k + \lambda\mathbf{d}_k)^t\mathbf{d}_k < -\varepsilon \text{ para } \lambda \in [0,\delta'] \text{ y para } k \in \mathscr{K} \text{ suficientemente grande.} \tag{3.9}$$

Ahora, sea $\bar{\delta} = \text{mín}\{\delta',\delta\} > 0$. Consideremos $k \in \mathscr{K}$ suficientemente grande. Por la Condición 3, y por la definición de $\mathbf{x}_{k+1}$, debemos tener $f(\mathbf{x}_{k+1}) \leq f(\mathbf{x}_k + \bar{\delta}\mathbf{d}_k)$. Por el teorema del valor medio, $f(\mathbf{x}_k + \bar{\delta}\mathbf{d}_k) = f(\mathbf{x}_k) + \bar{\delta}\nabla f(\hat{\mathbf{x}}_k)^t\mathbf{d}_k$, donde $\hat{\mathbf{x}}_k = \mathbf{x}_k + \lambda_k\bar{\delta}\mathbf{d}_k$ y $\lambda_k \in (0,1)$. Por 3.9 se deduce entonces que

$$f(\mathbf{x}_{k+1}) < f(\mathbf{x}_k) - \varepsilon\bar{\delta} \text{ para } k \in \mathscr{K} \text{ suficientemente grande.} \tag{3.10}$$

Dado que el algoritmo de direcciones factibles genera una secuencia de puntos con valores objetivos decrecientes, $\lim_{k\to\infty} f(\mathbf{x}_k) = f(\mathbf{x})$. En particular, tanto $f(\mathbf{x}_{k+1})$ como $f(\mathbf{x}_k)$ tienden a $f(\mathbf{x})$ si $k \in \mathscr{K}$ tiende a ∞. Así, de 3.10, obtenemos $f(\mathbf{x}) \leq f(\mathbf{x}) - \varepsilon\bar{\delta}$, lo cual es imposible, ya que $\varepsilon, \bar{\delta} > 0$. Esta contradicción muestra que no podría existir ninguna subsecuencia que satisfaga las propiedades 1 a 4. □

Teorema 4. Consideremos el problema 3.1 junto con las suposiciones que se hicieron cuando este se definió. Supongamos además que la secuencia $\{\mathbf{x}_k\}$ es generada por el algoritmo de gradiente reducido. Entonces, cualquier punto de acumulación de $\{\mathbf{x}_k\}$ es un punto KKT.

Demostración. Sea $\{\mathbf{x}_k\}_{\mathscr{K}}$ una subsecuencia convergente con límite $\hat{\mathbf{x}}$. Tenemos que mostrar que $\hat{\mathbf{x}}$ es un punto KKT. Supongamos, por contradicción, que $\hat{\mathbf{x}}$ no es un punto KKT. Vamos a mostrar una secuencia $\{(\mathbf{x}_k,\mathbf{d}_k)\}_{\mathscr{K}'}$, que satisface las condiciones 1 a 4 del lema 4, lo cual es imposible.

Sea $\{\mathbf{d}_k\}_{\mathscr{K}}$ la secuencia de direcciones asociada a $\{\mathbf{x}_k\}_{\mathscr{K}}$. Observemos que $\mathbf{d}_k$ está definido por 3.3 a 3.7. Sea I_k el conjunto que denota los índices de las m componentes más grandes de $\mathbf{x}_k$ utilizadas para calcular $\mathbf{d}_k$. Existe $\mathscr{K}' \subseteq \mathscr{K}$ tal que $I_k = \hat{I}$ para cada $k \in \mathscr{K}'$, donde $\hat{I}$ es el conjunto que denota los índices de las m componentes más grandes de $\hat{\mathbf{x}}$. Sea $\hat{\mathbf{d}}$ la dirección obtenida de 3.3 a 3.7 en $\hat{\mathbf{x}}$. Por el teorema 3, $\hat{\mathbf{d}} \neq \mathbf{0}$ y $\nabla f(\hat{\mathbf{x}})^t\hat{\mathbf{d}} < 0$. Dado que f es continuamente diferenciable, $\mathbf{x}_k \to \hat{\mathbf{x}}$ y $I_k = \hat{I}$ para $k \in \mathscr{K}'$, entonces, por 3.4 a 3.7, $\mathbf{d}_k \to \mathbf{d}$ para $k \in \mathscr{K}'$. En resumen, hemos mostrado una secuencia $\{(\mathbf{x}_k,\mathbf{d}_k)\}_{\mathscr{K}'}$, que satisface las condiciones 1, 2 y 4 del lema 4. Para completar la demostración, necesitamos demostrar que la parte 3 también es válida.

A partir de 3.9, recordemos que $\mathbf{x}_k + \lambda\mathbf{d}_k$ es factible para cada $\lambda \in [0,\delta_k]$, donde $\delta_k = \text{mín}\{\text{mín}\{-x_{ik}/d_{ik} : d_{jk} < 0\},\infty\} > 0$ para cada $k \in \mathscr{K}'$. Supongamos que $\text{ínf}\{\delta_k : k \in \mathscr{K}'\} = 0$. Entonces, existe un conjunto de índices $\mathscr{K}'' \subseteq \mathscr{K}'$ tal que $\delta_k = -x_{pk}/d_{pk}$ converge a 0 para $k \in \mathscr{K}''$, donde $x_{pk} > 0$, $d_{pk} < 0$ y p es un elemento de $\{1,\ldots,n\}$. Por 3.3 a 3.7, $\{d_{pk}\}_{\mathscr{K}''}$ es acotado, y dado que $\{\delta_k\}_{\mathscr{K}''}$ converge a 0, $\{x_{pk}\}_{\mathscr{K}''}$ converge a 0. Por lo tanto, $\hat{x}_p = 0$, es decir, $p \notin \hat{I}$. Pero $I_k = \hat{I}$ para $k \in \mathscr{K}''$, y por

lo tanto, $p \notin I_k$. Como $d_{pk} < 0$, de 3.6, $d_{pk} = -x_{pk} r_{pk}$. Entonces se sigue que $\delta_k = -x_{pk}/d_{pk} = 1/r_{pk}$. Esto muestra que $r_{pk} \to \infty$, lo cual es imposible, ya que $r_{pk} \to r_p \neq \infty$. Por lo tanto, $\text{ínf}\{\delta_k : k \in \mathscr{K}'\} = \delta > 0$. Hemos demostrado que existe un $\delta > 0$ tal que $\mathbf{x}_k + \lambda \mathbf{d}_k$ es factible para cada $\lambda \in [0, \delta]$ y para cada $k \in \mathscr{K}'$. Por lo tanto, se cumple la condición 3 del lema 4, y la prueba está completa. □

3.2. Aplicación práctica

Consideremos el siguiente problema:

$$\begin{array}{ll} \text{Minimizar} & 2x_1^2 + 2x_2^2 - 2x_1x_2 - 4x_1 - 6x_2 \\ \text{Sujeto a} & x_1 + x_2 + x_3 = 2, \\ & x_1 + 5x_2 + x_4 = 5, \\ & x_l, x_2, x_3, x_4 \geq 0. \end{array}$$

Vamos a resolver este problema utilizando el método de gradiente reducido de Wolfe partiendo del punto $\mathbf{x}_1 = (0,0,2,5)^t$. Notar que $\nabla f(\mathbf{x}) = (4x_1 - 2x_2 - 4, 4x_2 - 2x_1 - 6, 0, 0)^t$. Exhibiremos la información necesaria en cada iteración en una tabla similar a la tabla del método simplex. Sin embargo, dado que el vector gradiente cambia en cada iteración y que las variables no básicas podrían ser positivas, daremos explícitamente el vector gradiente y la solución completa en la parte superior de cada tabla. El vector de gradiente reducido $\mathbf{r}_k$ se mostrará como la última fila de cada tabla.

Iteración 1:

Dirección de búsqueda:

En el punto $\mathbf{x}_1 = (0,0,2,5)^t$, tenemos $\nabla f(\mathbf{x}_1) = (-4,-6,0,0)$. Por 3.3, tenemos $I_1 = \{3,4\}$, de modo que $\mathbf{B} = (\mathbf{a}_3, \mathbf{a}_4)$ y $\mathbf{N} = (\mathbf{a}_1, \mathbf{a}_2)$. Por 3.5, el gradiente reducido está dado por

$$\mathbf{r}^t = (-4,-6,0,0) - (0,0) \begin{pmatrix} 1 & 1 \\ 1 & 5 \\ 1 & 0 \\ 0 & 1 \end{pmatrix} = (-4,-6,0,0).$$

Notar que los cálculos para el gradiente reducido son similares a los cálculos para los coeficientes de la fila objetivo en el método simplex. Además, $r_i = 0$ para $i \in I_1$. La información en este punto se resume en la siguiente tabla:

			x_1	x_2	x_3	x_4
Solución $\mathbf{x}_1$			0	0	2	5
$\nabla f(\mathbf{x}_1)$			-4	-6	0	0
$\nabla_B f(\mathbf{x}_1) =$	0	x_3	1	1	1	0
	0	x_4	1	5	0	1
$\mathbf{r}$			-4	-6	0	0

Tabla 3.1: Resumen de la iteración 1 del método del gradiente reducido.

Dado lo anterior, tenemos que $\mathbf{d}_N = (\mathbf{d}_1, \mathbf{d}_2)^t = (4,6)^t$. Ahora calculamos $\mathbf{d}_B$ usando 3.7 para obtener $\mathbf{d}_B$.

$$\mathbf{d}_B = (\mathbf{d}_3, \mathbf{d}_4)^t = -\mathbf{B}^{-1}\mathbf{N}\mathbf{d}_N = -\begin{pmatrix} 1 & 1 \\ 1 & 5 \end{pmatrix} \begin{pmatrix} 4 \\ 6 \end{pmatrix} = (-10,-34)^t$$

Recordar que $\mathbf{B}^{-1}\mathbf{N}$ se guarda en las variables correspondientes a $\mathbf{N}$, es decir, x_1 y x_2. Por tanto, el vector de dirección es $\mathbf{d}_1 = (4,6,-10,-34)^t$.

Búsqueda lineal:

Comenzando desde $\mathbf{x}_1 = (0,0,2,5)^t$, ahora queremos minimizar la función objetivo a lo largo de la dirección $\mathbf{d}_1 = (4,6,-10,-34)^t$. El valor máximo de λ tal que $\mathbf{x}_1 + \lambda \mathbf{d}_1$ es factible se calcula usando 3.9 de la siguiente manera:

$$\lambda_{\max} = \text{mín}\left\{\frac{2}{10}, \frac{5}{34}\right\} = \frac{5}{34}.$$

Es fácil ver que $f(\mathbf{x}_1 + \lambda \mathbf{d}_1) = 56\lambda^2 - 52\lambda$, de modo que λ_1 es la solución al siguiente problema:

$$\begin{aligned} &\text{Minimizar} && 56\lambda^2 - 52\lambda \\ &\text{Sujeto a} && 0 \leq \lambda \leq \frac{5}{34}. \end{aligned}$$

La solución del problema anterior es $\lambda_1 = \frac{5}{34}$, por lo que $\mathbf{x}_2 = \mathbf{x}_1 + \lambda_1 \mathbf{d}_1 = \left(\frac{10}{17}, \frac{15}{17}, \frac{9}{17}, 0\right)'$.

Iteración 2:

Dirección de búsqueda:

En $\mathbf{x}_2 = \left(\frac{10}{17}, \frac{15}{17}, \frac{9}{17}, 0\right)^t$, de 3.3 obtenemos $I_2 = \{1,2\}$, $\mathbf{B} = (\mathbf{a}_1, \mathbf{a}_2)$ y $\mathbf{N} = (\mathbf{a}_3, \mathbf{a}_4)$. También tenemos $\nabla f(\mathbf{x}_2) = \left(-\frac{58}{17}, \frac{42}{17}, 0, 0\right)^t$. La información actual se registra en la siguiente tabla, donde las filas de x_1 y x_2 se obtienen mediante dos operaciones de pivote en la tabla de la iteración 1.

			x_1	x_2	x_3	x_4
Solución $\mathbf{x}_2$			$\frac{10}{17}$	$\frac{15}{17}$	$\frac{9}{17}$	0
$\nabla f(\mathbf{x}_2)$			$-\frac{58}{17}$	$-\frac{62}{17}$	0	0
$\nabla_B f(\mathbf{x}_2) =$	$-\frac{58}{17}$	x_1	1	0	$\frac{5}{4}$	$-\frac{1}{4}$
	0	x_2	0	1	$-\frac{1}{4}$	$\frac{1}{4}$
$\mathbf{r}$			0	0	$\frac{57}{17}$	$\frac{1}{17}$

Tabla 3.2: Resumen de la iteración 2 del método del gradiente reducido.

Utilizando 3.5,

$$\mathbf{r}^t = \left(-\frac{58}{17}, -\frac{62}{17}, 0, 0\right) - \left(-\frac{58}{17}, -\frac{62}{17}\right) \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ \frac{5}{4} & -\frac{1}{4} \\ -\frac{1}{4} & \frac{1}{4} \end{pmatrix} = \left(0, 0, \frac{57}{17}, \frac{1}{17}\right).$$

Por 3.6, $\mathbf{d}_3 = -\frac{9}{17}\frac{57}{17} = -\frac{513}{289}$ y $\mathbf{d}_4 = 0$, por lo que $\mathbf{d}_N = \left(-\frac{513}{289}, 0\right)^t$. De 3.7, obtenemos

$$\mathbf{d}_B = (\mathbf{d}_1, \mathbf{d}_2)^t = -\begin{pmatrix} \frac{5}{4} & -\frac{1}{4} \\ -\frac{1}{4} & \frac{1}{4} \end{pmatrix} \begin{pmatrix} -\frac{513}{289} \\ 0 \end{pmatrix} = \begin{pmatrix} \frac{2565}{1156} \\ -\frac{513}{1156} \end{pmatrix}.$$

La nueva dirección de búsqueda es, por lo tanto, $\mathbf{d}_2 = \left(\frac{2565}{1156}, -\frac{513}{1156}, -\frac{513}{289}, 0\right)^t$.

Búsqueda lineal:

Comenzando desde $\mathbf{x}_2 = \left(\frac{10}{17}, \frac{15}{17}, \frac{9}{17}, 0\right)^t$, deseamos minimizar la función objetivo a lo largo de la dirección $\mathbf{d}_2 = \left(\frac{2565}{1156}, -\frac{513}{1156}, -\frac{513}{289}, 0\right)^t$. El valor máximo de λ tal que $\mathbf{x}_2 + \lambda \mathbf{d}_2$ es factible se calcula usando 3.9 como antes:

$$\lambda_{\max} = \text{mín}\left\{\frac{-15/17}{-513/1156}, \frac{-9/17}{-513/289}\right\} = \frac{17}{57}.$$

Es fácil comprobar que $f(\mathbf{x}_2 + \lambda \mathbf{d}_2) = 12{,}21\lambda^2 - 5{,}95\lambda - 6{,}436$, de modo que λ_2 se obtiene resolviendo el siguiente problema:

$$\begin{aligned} &\text{Minimizar} && 12{,}21\lambda^2 - 5{,}95\lambda - 6{,}436 \\ &\text{Sujeto a} && 0 \leq \lambda \leq \frac{17}{57}. \end{aligned}$$

La solución del problema es $\lambda_2 = \frac{68}{279}$, por lo que $\mathbf{x}_3 = \mathbf{x}_2 + \lambda_2 \mathbf{d}_2 = \left(\frac{35}{31}, \frac{24}{31}, \frac{3}{31}, 0\right)$.

Iteración 3:

Dirección de búsqueda:

Ahora $I_3 = \{1,2\}$, por lo que $\mathbf{B} = (\mathbf{a}_1, \mathbf{a}_2)$ y $\mathbf{N} = (\mathbf{a}_3, \mathbf{a}_4)$. Dado que $I_3 = I_2$, la tabla en la iteración 2 se puede mantener. Sin embargo, ahora tenemos $\nabla f(\mathbf{x}_3) = \left(-\frac{32}{31}, -\frac{160}{31}, 0, 0\right)^t$.

			x_1	x_2	x_3	x_4
Solución $\mathbf{x}_3$			$\frac{35}{31}$	$\frac{24}{31}$	$\frac{3}{31}$	0
$\nabla f(\mathbf{x}_3)$			$-\frac{32}{31}$	$-\frac{160}{31}$	0	0
$\nabla_B f(\mathbf{x}_3) =$	$-\frac{32}{31}$	x_1	1	0	$\frac{5}{4}$	$-\frac{1}{4}$
	$-\frac{160}{31}$	x_2	0	1	$-\frac{1}{4}$	$\frac{1}{4}$
$\mathbf{r}$			0	0	0	$\frac{32}{31}$

Tabla 3.3: Resumen de la iteración 3 del método del gradiente reducido

Por 3.5,

$$\mathbf{r}^t = \left(-\frac{32}{31}, -\frac{160}{31}, 0, 0\right) - \left(-\frac{32}{31}, -\frac{160}{31}\right) \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ \frac{5}{4} & -\frac{1}{4} \\ -\frac{1}{4} & \frac{1}{4} \end{pmatrix} = \left(0, 0, 0, \frac{32}{31}\right).$$

De 3.6, $\mathbf{d}_N = (\mathbf{d}_3, \mathbf{d}_4)^t = (0,0)^t$, y de 3.7, $\mathbf{d}_B = (\mathbf{d}_1, \mathbf{d}_2)^t = (0,0)^t$. Por lo tanto, $\mathbf{d} = 0$, y la solución $\mathbf{x}_3$ es una solución KKT y, así, óptima para este problema. Los multiplicadores de Lagrange óptimos asociados a las restricciones de igualdad son $\nabla_{\mathbf{B}} f(\mathbf{x}_3)^t \mathbf{B}^{-1} = \left(0, -\frac{32}{31}\right)^t$, y los asociados a las restricciones de no negatividad son $(0,0,0,1)^t$.

Capítulo 4

Implementación de los algoritmos en Python

En este capítulo, presentamos una breve comparación entre el desarrollo teórico y la implementación en Python de los métodos de proyección del gradiente de Rosen y del gradiente reducido de Wolfe para la resolución de problemas de optimización no lineal con restricciones. Nuestro objetivo es validar nuestra comprensión de los algoritmos, su funcionamiento y corroborar la precisión de la implementación en código. Los algoritmos han sido implementados de manera fiel al desarrollo teórico presentado en los capítulos 2 y 3. Hemos decidido hacer esto con el fin de brindar una herramienta que pueda ser utilizada por terceros en la resolución de problemas similares.

Para evaluar la eficacia de las implementaciones, las hemos aplicado a los mismos ejemplos planteados en los capítulos anteriores, y se ha confirmado que los resultados obtenidos son consistentes. En el apéndice A, se encuentran los scripts completos de las implementaciones en Python de ambos algoritmos. Además, en el apéndice C, se muestra en detalle la resolución de otros problemas de optimización no lineal usando la implementación en Python de los métodos de Rosen y Wolfe. A continuación, presentamos la salida en terminal del script que implementa el método de Rosen aplicado al problema de optmización no lineal con restricciones de la sección 2.2:

```
Iteración: 0
xk: [0. 0.]
grad(f(xk)): [-4. -6.]
A1: [[-1  0]
 [ 0 -1]]
A2: [[1 1]
 [1 5]]
P: [[0. 0.]
 [0. 0.]]
dk: [0. 0.]
w: [-4. -6.]
u: [-4. -6.]
lambda_k: 0.16666319308046373
Iteración: 1
xk: [0.         0.99997916]
grad(f(xk)): [-5.99995832 -2.00008337]
A1: [[-1  0]]
A2: [[ 1  1]
 [ 1  5]
 [ 0 -1]]
P: [[0. 0.]
 [0. 1.]]
dk: [0.         2.00008337]
lambda_k: 3.980209692793515e-06
Iteración: 2
xk: [0.         0.99998712]
grad(f(xk)): [-5.99997424 -2.00005152]
A1: [[-1  0]]
A2: [[ 1  1]
 [ 1  5]
 [ 0 -1]]
P: [[0. 0.]
 [0. 1.]]
dk: [0.         2.00005152]
lambda_k: 2.4599440368540345e-06
Iteración: 3
xk: [0.         0.99999204]
grad(f(xk)): [-5.99998408 -2.00003184]
A1: [[ 1  5]
 [-1  0]]
A2: [[ 1  1]
 [ 0 -1]]
P: [[-2.22044605e-16 -3.46944695e-17]
 [ 5.55111512e-17 -4.44089210e-16]]
dk: [-1.40165414e-15 -5.55126537e-16]
w: [ 0.40000637 -5.59997771]
u: [ 0.40000637 -5.59997771]
lambda_k: 0.2096774193548397
Iteración: 4
xk: [1.12902776 0.77418649]
grad(f(xk)): [-1.03226192 -5.16130958]
A1: [[1 5]]
A2: [[ 1  1]
 [-1  0]
 [ 0 -1]]
P: [[ 0.96153846 -0.19230769]
 [-0.19230769  0.03846154]]
dk: [-2.56111646e-14  4.97217372e-15]
w: [1.03226192]
u: [1.03226192]
```

Figura 4.1: Iteraciones del método de Rosen para la resolución del problema de la sección 2.2.

Observamos que la solución obtenida con la implementación en Python es coherente con el resultado teórico presentado en el capítulo 2. Es importante señalar que el número de iteraciones requeridas por la implementación en Python puede variar en comparación con la solución teórica. Esto es normal y se debe a diferencias en la precisión numérica entre las dos aproximaciones. En el enfoque teórico, trabajamos con números racionales, lo que elimina los errores de redondeo. En contraste, la implementación en Python realiza cálculos numéricos, donde el número de iteraciones depende tanto de la tolerancia predefinida como de la precisión de la máquina. Esta variabilidad en el número de iteraciones no socava la validez de la implementación en Python. Más bien, ilustra las limitaciones y consideraciones prácticas al traducir algoritmos matemáticos en código ejecutable.

Para continuar con la comparación entre los cálculos teóricos y las implementaciones en Python, examinaremos a continuación el método del gradiente reducido de Wolfe. Al igual que con el método de Rosen, hemos aplicado el método de Wolfe manualmente y luego lo hemos implementado en Python. La salida del script en Python que implementa el método de Wolfe para resolver el problema de optimización no lineal con restricciones de la sección 3.2 se muestra a continuación. Por motivos de espacio, no hemos añadido todas las iteraciones en este capítulo. La ejecución completa se proporciona en el apéndice B.

```
Iteración: 0
x_k: [0. 0. 2. 5.]
grad_f: [-4. -6.  0.  0.]
I_k: [2 3]
B: [[1 0]
 [0 1]]
N: [[1 1]
 [1 5]]
r: [-4. -6.  0.  0.]
d_N: [4. 6.]
d_B: [-10. -34.]
d_k: [  4.   6. -10. -34.]
lambda_max: 0.14705882352941177
lambda_k: 0.1470528152427929
Iteración: 1
x_k: [5.88211261e-01 8.82316891e-01
5.29471848e-01 2.04281745e-04]
grad_f: [-3.41178874 -3.64715496
0.          0.        ]
I_k: [0 1]
B: [[1 1]
 [1 5]]
N: [[1 0]
 [0 1]]
r: [0.         0.
3.35294718 0.05884155]
d_N: [-1.77529114e+00 -1.20202554e-05]
d_B: [ 2.21911092 -0.44381978]
d_k: [ 2.21911092e+00 -4.43819780e-01
-1.77529114e+00 -1.20202554e-05]
lambda_max: 0.29824507959626917
lambda_k: 0.24370092815344244

...

Iteración: 7
x_k: [1.12903181e+00 7.74193638e-01
9.67745527e-02 1.15178382e-27]
grad_f: [-1.03226004 -5.16128907
0.          0.        ]
I_k: [1 0]
B: [[1 1]
 [5 1]]
N: [[1 0]
 [0 1]]
r: [0.00000000e+00 0.00000000e+00
2.78362519e-06 1.03225726e+00]
d_N: [-2.69384083e-07 -1.18893720e-27]
d_B: [-6.73460208e-08  3.36730104e-07]
d_k: [ 3.36730104e-07 -6.73460208e-08
-2.69384083e-07 -1.18893720e-27]
```

Figura 4.2: Iteraciones del método de Wolfe para la resolución del problema de la sección 3.2.

El análisis de la salida del script nos muestra que los resultados obtenidos a través de la implementación en Python del método de Wolfe concuerdan con la resolución manual previamente realizada. Este hecho respalda la corrección del código Python implementado. Como ocurre con el método de Rosen, el número de iteraciones requeridas por el método de Wolfe puede variar en comparación con la solución teórica. Esto se debe a las diferencias en la precisión numérica entre las dos aproximaciones y es una característica inherente a las implementaciones numéricas en comparación con los enfoques teóricos exactos. Es importante resaltar que, a pesar de estas diferencias en términos de número de iteraciones, la implementación en Python del método de Wolfe proporciona una solución precisa al problema de optimización no lineal con restricciones, lo que reafirma la utilidad de esta implementación y la validez de nuestras derivaciones teóricas.

En resumen, la consistencia entre nuestras resoluciones teóricas y las implementaciones en Python de los métodos de proyección del gradiente de Rosen y del gradiente reducido de Wolfe proporciona una fuerte evidencia de que los códigos son correctos y efectivos para resolver problemas de optimización no lineal con restricciones.

Capítulo 5

Comparación y conclusión

Como hemos visto en los capítulos 2 y 3, el método de proyección del gradiente de Rosen y el método del gradiente reducido de Wolfe son dos de las técnicas más robustas para resolver problemas de optimización no lineal con restricciones de igualdad y desigualdad. Aunque ambos se basan en la idea de seguir la dirección del gradiente negativo, difieren significativamente en su enfoque y en los detalles de su implementación, lo que lleva a diferencias importantes en su eficiencia, robustez, flexibilidad y aplicabilidad.

Para empezar, los dos métodos difieren en cómo manejan las restricciones del problema. El método de Rosen proyecta la dirección del gradiente negativo en el espacio factible en cada iteración, un enfoque que garantiza que cada paso permanezca dentro del conjunto factible. Esto se basa en la premisa de que, en el óptimo, el gradiente de la función objetivo será ortogonal al espacio factible. Este enfoque es intuitivo y simple de implementar, lo que hace que el método de Rosen sea una opción popular para muchos problemas de optimización.

Por otro lado, el método de Wolfe modifica la dirección del gradiente para tener en cuenta las restricciones activas, lo que puede considerarse como una forma de proyectar el gradiente no en el espacio factible, sino en el espacio de las restricciones activas. Este enfoque elimina la necesidad de realizar proyecciones en cada iteración, lo que puede ser computacionalmente costoso en problemas de alta dimensionalidad.

Además, ambos métodos tienen características de convergencia diferentes. Aunque la convergencia del método de Rosen se garantiza bajo ciertas condiciones, su velocidad de convergencia puede ser lenta, especialmente cerca del óptimo. Esto se debe a que el método puede sufrir de la denominada "convergencia zigzag", en la que las iteraciones pueden oscilar entre diferentes direcciones y hacer poco progreso hacia el óptimo. En contraste, el método de Wolfe generalmente tiene una velocidad de convergencia más rápida debido a su manejo más sofisticado de las restricciones. Al tomar en cuenta explícitamente las restricciones activas al calcular la dirección de búsqueda, el método puede avanzar más directamente hacia el óptimo. Sin embargo, su implementación puede ser más compleja, ya que requiere el cálculo y la actualización del gradiente reducido.

La robustez de estos métodos también es un punto importante a considerar. El método de Rosen tiende a ser más robusto en presencia de errores numéricos y perturbaciones en los datos. Su paso de proyección puede "corregir" ciertos errores, lo que le permite seguir realizando progresos incluso si la dirección del gradiente es ligeramente incorrecta. En contraste, el método de Wolfe puede ser menos robusto frente a errores y perturbaciones, ya que cualquier error en el cálculo del gradiente reducido se acumulará a lo largo de las iteraciones.

Cuando se considera la aplicabilidad de estos métodos, ambos tienen sus fortalezas y debilidades. El método de Rosen es fácil de implementar y es bastante general, lo que significa que se puede aplicar a una amplia gama de problemas. Sin embargo, su eficiencia puede verse afectada en problemas con un gran número de variables o restricciones, debido a la necesidad de realizar proyecciones en cada iteración. Por su parte, el método de Wolfe puede ser más eficiente en problemas de alta dimensionalidad, gracias a su enfoque más sofisticado para manejar las restricciones. También es especialmente útil en problemas

con muchas restricciones de igualdad. Sin embargo, su implementación puede ser más compleja y puede requerir más experiencia y conocimiento técnico.

En términos de flexibilidad, el método de Wolfe tiene una ventaja sobre el método de Rosen. Si las restricciones de un problema cambian, el gradiente reducido en el método de Wolfe se puede actualizar fácilmente para reflejar estos cambios. En contraste, cualquier cambio en las restricciones en el método de Rosen requeriría una nueva proyección de la dirección de búsqueda, lo que podría ser computacionalmente costoso.

En conclusión, tanto el método de proyección del gradiente de Rosen como el método del gradiente reducido de Wolfe tienen sus propias fortalezas y debilidades, y la elección entre ellos dependerá en gran medida del problema específico en cuestión. Para problemas con un pequeño número de variables y restricciones, el método de Rosen puede ser suficiente. Sin embargo, para problemas de alta dimensionalidad y/o con muchas restricciones de igualdad, el método de Wolfe puede ser una opción más eficiente.

5.1. Posibles direcciones futuras de investigación

Este trabajo de fin de grado ha proporcionado una exploración exhaustiva de dos métodos de direcciones factibles para resolver problemas de optimización no lineal con restricciones: el método de proyección de Rosen y el método del gradiente reducido de Wolfe. A través de una cuidadosa exposición teórica y la resolución de un ejemplo numérico con ambos métodos, hemos ganado un conocimiento valioso sobre sus características, ventajas y desventajas.

Dicho esto, la contribución de este trabajo a mi comprensión de los métodos de direcciones factibles es innegable y sienta las bases para continuar su aprendizaje de cara a una posible investigación posteriormente. A continuación, proponemos varias direcciones para completar el estudio de estos métodos:

1. Ampliar la base de experimentación numérica: La realización de experimentos numéricos adicionales con un conjunto diverso de problemas de optimización no lineal con restricciones permitiría obtener una comprensión más profunda de cómo los métodos de Rosen y Wolfe se desempeñan en diferentes situaciones, lo que permitiría entender cuál de ellos es más adecuado en ciertos casos.

2. Investigar mejoras y extensiones de los métodos: El estudio de variantes y mejoras de los métodos de proyección de Rosen y gradiente reducido de Wolfe podría conducir a la identificación de nuevas propiedades y características que pueden mejorar su rendimiento en la resolución de problemas de optimización no lineal con restricciones.

3. Combinar métodos de direcciones factibles con otros enfoques de optimización: La exploración de enfoques híbridos que integren los métodos de direcciones factibles con otros algoritmos de optimización, como el descenso de coordenadas, el algoritmo de Newton y la optimización basada en la población, podría permitir el desarrollo de soluciones más robustas y efectivas para problemas de optimización desafiantes y complejos.

4. Aplicar los métodos de direcciones factibles a problemas del mundo real: Un estudio sistemático de cómo los métodos de Rosen y Wolfe pueden ser aplicados a problemas prácticos en diversas áreas como la ingeniería, la economía, la logística, la biología y la inteligencia artificial, permitiría evaluar su relevancia y utilidad en situaciones del mundo real, así como identificar posibles limitaciones y áreas de mejora.

En resumen, este trabajo de fin de grado ha aportado una sólida base para el estudio y comprensión de los métodos de direcciones factibles en el contexto de la optimización no lineal con restricciones. Si bien no hemos determinado la superioridad de uno de los métodos sobre el otro de manera concluyente, hemos identificado varias direcciones futuras prometedoras para la investigación en este campo. Estas áreas de investigación tienen el potencial de enriquecer nuestra comprensión de estos métodos, mejorar su aplicabilidad y, en última instancia, contribuir al avance del conocimiento en el campo de la optimización no lineal con restricciones.

Bibliografía

[1] A. L. TITS, J. ZHOU, Fast Feasible Direction Methods, with Engineering Applications, *Institute for Systems Research Technical Reports* **TR 91-20**, (1991).

[2] M. OTSUKI, Discrete Procedures of Economic Planning: A Unified View from Feasible Direction Methods, *The Review of Economic Studies* **45**, Issue 1, (1978), 77-84.

[3] F. M. XU, C. X. XU, H. G. XUE, A Feasible Direction Algorithm Without Line Search for Solving Max-Bisection Problems, *Journal of Computational Mathematics*, **23**, (2005), 619-634.

[4] G. P. RANGAIAH, Z. FENG, A. F. HOADLEY, Multi-Objective Optimization Applications in Chemical Process Engineering: Tutorial and Review, *Processes* **8**(5), 508; (2020).

[5] J. E. KELLEY, The cutting-plane method for solving convex programs, *Journal of the Society for Industrial and Applied Mathematics* **8**(4), (1960), 703-712.

[6] J. B. ROSEN, The gradient projection method for nonlinear programming. Part I. Linear constraints, *Journal of the Society for Industrial and Applied Mathematics* **8**(1), (1960), 181-217.

[7] P. WOLFE, Convergence conditions for ascent methods, *SIAM Review* **11**(2), (1969), 226-235.

[8] A. V. FIACCO, G. P. MCCORMICK, Nonlinear programming: Sequential unconstrained minimization techniques, *John Wiley & Sons*, (1968).

[9] M. J. D. POWELL, A fast algorithm for nonlinearly constrained optimization calculations, *Numerical Analysis*, (1978), 144-157, Springer, Berlin, Heidelberg.

[10] N. KARMARKAR, A new polynomial-time algorithm for linear programming, *Combinatorica* **4**(4), (1984), 373-395.

[11] M. J. D. POWELL, Variable metric methods for constrained optimization, *Trust Region Methods*, (1983), 230-249, Springer, Berlin, Heidelberg; K. SCHITTKOWSKI, The nonlinear programming method of Wilson, Han, and Powell with an augmented Lagrangian type line search function. Part 1: Convergence analysis, *Numerische Mathematik* **48**(1), (1985), 85-114.

[12] D. E. GOLDBERG, Genetic algorithms in search, optimization, and machine learning, *Addison-Wesley*; M. DORIGO, T. STÜTZLE, Ant colony optimization, *MIT press*, (1989).

[13] S. J. WRIGHT, Primal-dual interior-point methods, *SIAM*, (1997).

[14] X. HU, R. C. EBERHART, Solving constrained nonlinear optimization problems with particle swarm optimization, *Proceedings of the Sixth World Multiconference on Systemics, Cybernetics and Informatics* **2**, (2002), 203-206.

[15] A. R. CONN, N. I. M. GOULD, P. L. TOINT, Trust region methods, *SIAM*, (2000).

[16] R. FLETCHER, S. LEYFFER, Nonlinear programming without a penalty function, *Mathematical Programming* **91**(2), (2002), 239-269.

[17] J. F. BONNANS, A. SHAPIRO, Perturbation Analysis of Optimization Problems, *Springer Series in Operations Research and Financial Engineering*, **1**, (2006).

[18] ANDRÉA R. VARGAS, A feasible directions algorithm for solving linearly constrained optimization problems, *Journal of Global Optimization*, **48**, (2010), 461-474.

[19] JIAN LI, GONGLIN YUAN, An improved feasible directions method for general constrained optimization, *Applied Mathematics and Computation*, **219**, (2013), 6645-6655.

[20] L. KOZMA, A. ZILINSKAS, Feasible direction method using conjugate gradients for unconstrained optimization, *Mathematical Modelling and Analysis*, **20**, (2015), 57-74.

[21] X. ZHOU, Y. CHEN, Y. YE, A feasible direction method for optimization with Nesterov acceleration, *Journal of Optimization Theory and Applications*, **176**, (2018), 461-482.

[22] A. V. FIACCO, G. P. MCCORMICK, *Nonlinear Programming: Sequential Unconstrained Minimization Techniques*, John Wiley & Sons, Inc., New York, USA, 1968.

[23] M. S. BAZARAA, H. D. SHERALI, C. M. SHETTY, *Nonlinear Programming. Theory and Algorithms*, Wiley, Hoboken, USA, third edition, 2006.

[24] D. G. LUENBERGER, Y. YE, *Linear and Nonlinear Programming*, Springer, New York, USA, third edition, 2008.

Apéndice A

Detalle de la implementación

A.1. Método de proyección del gradiente de Rosen

El presente script ha sido desarrollado utilizando Python, uno de los lenguajes de programación más versátiles y robustos disponibles en la actualidad. Esta elección no es casual, dado que Python ofrece una amplia gama de bibliotecas y módulos científicos y matemáticos muy útiles para abordar la resolución de problemas de optimización no lineal con restricciones.

En concreto, este script implementa el método de proyección del gradiente de Rosen, un algoritmo de optimización eficiente para resolver problemas de optimización no lineal con restricciones de igualdad y desigualdad. Este algoritmo se basa en la idea de proyectar sucesivamente el gradiente de la función objetivo sobre el conjunto factible de las restricciones, garantizando la convergencia a un óptimo local, siempre y cuando las restricciones sean convexas y la función objetivo sea diferenciable.

Este método y su implementación en Python son una herramienta valiosa para resolver problemas académicos de optimización, ya que permiten una comprensión más profunda de los métodos de direcciones factibles y su funcionamiento. Esto es especialmente relevante para aquellos interesados en adquirir habilidades en programación matemática y optimización.

```python
import numpy as np
from autograd import grad
from scipy.optimize import minimize_scalar

def decompose_A_and_b(x_k):
    equality_mask = np.isclose(A @ x_k, b)
    A1 = A[equality_mask]
    b1 = b[equality_mask]
    A2 = A[~equality_mask]
    b2 = b[~equality_mask]
    return A1, b1, A2, b2

def rosen_projection(f, A, b, Q, q, x1, tol=1e-6, max_iter=1000):
    m, n = A.shape
    l = Q.shape[0] if Q.size > 0 else 0
    x_k = x1
    k = 0
    A1, b1, A2, b2 = decompose_A_and_b(x_k)
    while k < max_iter:
        M_ = np.vstack([A1, Q]) if Q.size > 0 else A1
        if len(M_) == 0:
            if np.allclose(grad_f(x_k), 0, atol=tol):
                return x_k
            else:
                d_k = -grad_f(x_k)
        else:
            P = np.eye(n) - M_.T @ np.linalg.pinv(M_ @ M_.T) @ M_
            d_k = -P @ grad_f(x_k)
            if np.allclose(d_k, 0, atol=tol):
                w = -np.linalg.pinv(M_ @ M_.T) @ M_ @ grad_f(x_k)
                u = w[:len(A1)]
                if np.all(u >= 0):
```

```
                    return x_k
                else:
                    j = np.argmin(u)
                    A1 = np.delete(A1, j, axis=0)
                    b1 = np.delete(b1, j)
                    continue
        hat_b_k = b2 - A2 @ x_k
        hat_d_k = A2 @ d_k
        fun = lambda lambda_: f(x_k + lambda_ * d_k)
        if np.any(hat_d_k > 0):
            lambda_max = np.min(hat_b_k[hat_d_k > 0] / hat_d_k[hat_d_k > 0])
        else:
            lambda_max = np.inf
        result = minimize_scalar(fun, method='bounded', bounds=(0, lambda_max))
        lambda_k = result.x
        x_k += lambda_k * d_k
        A1, b1, A2, b2 = decompose_A_and_b(x_k)
        k += 1
    return x_k

def f(x):
    x1, x2 = x
    return 2*x1**2 + 2*x2**2 - 2*x1*x2 - 4*x1 - 6*x2
A = np.array([[1, 1], [1, 5], [-1, 0], [0, -1]])
b = np.array([2, 5, 0, 0])
Q = np.array([])
q = np.array([])
x1 = np.array([0.0, 0.0])
grad_f = grad(f)

print(rosen_projection(f, A, b, Q, q, x1))
```

Bloque de código A.1: Método de proyección del gradiente de Rosen

En este script, primero importamos los módulos necesarios, como *numpy* para operaciones matemáticas con arrays, *autograd* para el cálculo automático de gradientes y *scipy.optimize* para realizar la búsqueda lineal. La función principal es *rosen_proyection*, que implementa el método de proyección del gradiente de Rosen. Esta función toma como argumentos la función objetivo f, las matrices $\mathbf{A}$ y $\mathbf{Q}$ y los vectores $\mathbf{b}$, $\mathbf{q}$ y $\mathbf{x}_1$ que definen las restricciones y el punto de inicio. También acepta argumentos opcionales para la tolerancia del método y el número máximo de iteraciones.

La implementación se realiza a través de un bucle while, que itera hasta alcanzar el número máximo de iteraciones o hasta que se encuentre una solución. En cada iteración, se realiza la proyección del gradiente y se busca la dirección de descenso. Luego, se realiza una búsqueda lineal para encontrar el tamaño de paso óptimo. Finalmente, el punto de solución se actualiza y se reevalúa si se cumplen las condiciones de parada.

La función f es la función objetivo a minimizar y se define en la parte inferior del script, junto con las matrices y vectores que definen las restricciones del problema. Al final, el script imprime el resultado de la optimización.

En general, esta implementación en Python del método de proyección del gradiente de Rosen es un ejemplo poderoso y versátil de cómo la programación puede ser utilizada para resolver problemas matemáticos complejos y promover una mayor comprensión de los métodos de optimización.

A.2. Método del gradiente reducido de Wolfe

Continuando con el estudio de métodos de optimización con restricciones, presentamos ahora el método del gradiente reducido de Wolfe, otro algoritmo de optimización no lineal que se ha implementado en este trabajo utilizando Python. El método de Wolfe es una alternativa eficiente al método de Rosen que nos permite enfrentar problemas de optimización con restricciones.

El método del gradiente reducido de Wolfe se basa en el uso del gradiente de la función objetivo, pero con una variante interesante: en cada iteración, se seleccionan ciertas componentes del gradiente

para formar un subespacio, en el cual se busca la dirección de descenso. Esto puede ser beneficioso en problemas de gran escala, donde la selección de subespacios más pequeños puede mejorar la eficiencia del algoritmo.

```python
import numpy as np
from autograd import grad
from scipy.optimize import minimize_scalar

def wolfe_reduced_gradient(f, A, b, x1, tol=1e-6, max_iter=1000):
    m, n = A.shape
    x_k = x1
    grad_f = grad(f)
    for _ in range(max_iter):
        I_k = np.argsort(x_k)[-m:]
        B = A[:, I_k]
        N = np.delete(A, I_k, axis=1)
        r = grad_f(x_k) - grad_f(x_k)[I_k].T.dot(np.linalg.inv(B)).dot(A)
        I_N = np.delete(np.arange(n), I_k)
        r_N = r[I_N]
        x_N = x_k[I_N]
        d_N = np.where(r_N <= 0, -r_N, -x_N * r_N)
        d_B = -np.linalg.inv(B).dot(N).dot(d_N)
        d_k = np.zeros(n)
        d_k[I_k] = d_B
        d_k[I_N] = d_N
        if np.allclose(d_k, 0, atol=tol):
            return x_k
        lambda_max = min([-x_k[i]/d_k[i] for i in range(n) if d_k[i] < 0], default=np.inf
    )
        result = minimize_scalar(lambda x: f(x_k + x * d_k), bounds=(0, lambda_max),
    method='bounded')
        lambda_k = result.x
        x_k = x_k + lambda_k * d_k
    return x_k

def f(x):
    x1, x2 = x[:2]
    return 2*x1**2 + 2*x2**2 - 2*x1*x2 - 4*x1 - 6*x2

A = np.array([[1, 1, 1, 0], [1, 5, 0, 1]])
b = np.array([2, 5])
x1 = np.array([0.0, 0.0, 2.0, 5.0])

optimum = wolfe_reduced_gradient(f, A, b, x1)
print(f"Optimum found in: {optimum}")
```

Bloque de código A.2: Método del gradiente reducido de Wolfe

Este script Python implementa el método del gradiente reducido de Wolfe en la función *wolfe_reduced_gradient*. La función toma como argumentos la función objetivo f, la matriz $\mathbf{A}$, el vector $\mathbf{b}$ y el punto de inicio $\mathbf{x}_1$. Asimismo, tiene argumentos opcionales para la tolerancia del método y el número máximo de iteraciones.

El script comienza con la importación de los módulos necesarios. La función principal *wolfe_reduced_gradient* sigue dos pasos principales: en el primer paso, calcula el subespacio de búsqueda y la dirección de descenso; en el segundo paso, realiza una búsqueda lineal para encontrar el tamaño de paso óptimo.

La función objetivo a minimizar se define en la parte inferior del script, junto con la matriz $\mathbf{A}$, el vector $\mathbf{b}$ y el punto inicial $\mathbf{x}_1$. Finalmente, se ejecuta el algoritmo y se imprime el resultado de la optimización.

Así como el método de proyección del gradiente de Rosen, la implementación del método del gradiente reducido de Wolfe en Python nos permite explorar otras estrategias para resolver problemas de optimización con restricciones. Esto demuestra la flexibilidad y la capacidad de Python para aplicar y comparar diferentes métodos de optimización, promoviendo una mayor comprensión de los algoritmos y sus aplicaciones.

Apéndice B

Ejecución completa del algoritmo de Wolfe

En este apéndice presentamos la salida completa de la resolución de la implementación en Python del problema de optimización no lineal con restricciones de la sección 3.2:

```
Iteración: 0
x_k: [0. 0. 2. 5.]
grad_f: [-4. -6.  0.  0.]
I_k: [2 3]
B: [[1 0]
 [0 1]]
N: [[1 1]
 [1 5]]
r: [-4. -6.  0.  0.]
d_N: [4. 6.]
d_B: [-10. -34.]
d_k: [  4.   6. -10. -34.]
lambda_max: 0.14705882352941177
lambda_k: 0.1470528152427929
Iteración: 1
x_k: [5.88211261e-01 8.82316891e-01 5.29471848e-01 2.04281745e-04]
grad_f: [-3.41178874 -3.64715496  0.          0.        ]
I_k: [0 1]
B: [[1 1]
 [1 5]]
N: [[1 0]
 [0 1]]
r: [0.         0.         3.35294718 0.05884155]
d_N: [-1.77529114e+00 -1.20202554e-05]
d_B: [ 2.21911092 -0.44381978]
d_k: [ 2.21911092e+00 -4.43819780e-01 -1.77529114e+00 -1.20202554e-05]
lambda_max: 0.29824507959626917
lambda_k: 0.24370092815344244
Iteración: 2
x_k: [1.12901065e+00 7.74157599e-01 9.68317488e-02 2.01352398e-04]
grad_f: [-1.03227259 -5.16139091  0.          0.        ]
I_k: [1 0]
B: [[1 1]
 [5 1]]
N: [[1 0]
 [0 1]]
r: [ 0.00000000e+00  0.00000000e+00 -6.98942492e-06  1.03227958e+00]
d_N: [ 6.98942492e-06 -2.07851968e-04]
d_B: [ 5.37103483e-05 -6.06997732e-05]
d_k: [-6.06997732e-05  5.37103483e-05  6.98942492e-06 -2.07851968e-04]
lambda_max: 0.9687298091878863
lambda_k: 0.9687240347241601
Iteración: 3
x_k: [1.12895185e+00 7.74209630e-01 9.68385197e-02 1.20023365e-09]
grad_f: [-1.03261186 -5.16106518  0.          0.        ]
I_k: [1 0]
B: [[1 1]
 [5 1]]
N: [[1 0]
 [0 1]]
```

```
r: [0.00000000e+00 0.00000000e+00 4.98524655e-04 1.03211333e+00]
d_N: [-4.82763896e-05 -1.23877715e-09]
d_B: [-1.20687877e-05  6.03451773e-05]
d_k: [ 6.03451773e-05 -1.20687877e-05 -4.82763896e-05 -1.23877715e-09]
lambda_max: 0.9688858474328951
lambda_k: 0.9688800720390466
Iteración: 4
x_k: [1.12901032e+00 7.74197936e-01 9.67917456e-02 7.15442595e-15]
grad_f: [-1.0323546  -5.16122889  0.          0.        ]
I_k: [1 0]
B: [[1 1]
 [5 1]]
N: [[1 0]
 [0 1]]
r: [0.00000000e+00 0.00000000e+00 1.36028609e-04 1.03221857e+00]
d_N: [-1.31664465e-05 -7.38493134e-15]
d_B: [-3.29161162e-06  1.64580581e-05]
d_k: [ 1.64580581e-05 -3.29161162e-06 -1.31664465e-05 -7.38493134e-15]
lambda_max: 0.9687870639002298
lambda_k: 0.9687812890952163
Iteración: 5
x_k: [1.12902626e+00 7.74194748e-01 9.67789902e-02 4.26465385e-20]
grad_f: [-1.03228445 -5.16127353  0.          0.        ]
I_k: [1 0]
B: [[1 1]
 [5 1]]
N: [[1 0]
 [0 1]]
r: [0.00000000e+00 0.00000000e+00 3.71742044e-05 1.03224727e+00]
d_N: [-3.59768197e-06 -4.40217731e-20]
d_B: [-8.99420491e-07  4.49710246e-06]
d_k: [ 4.49710246e-06 -8.99420491e-07 -3.59768197e-06 -4.40217731e-20]
lambda_max: 0.9687601286244828
lambda_k: 0.9687163372165017
Iteración: 6
x_k: [1.12903062e+00 7.74193876e-01 9.67755051e-02 1.92777542e-24]
grad_f: [-1.03226528 -5.16128573  0.          0.        ]
I_k: [1 0]
B: [[1 1]
 [5 1]]
N: [[1 0]
 [0 1]]
r: [0.00000000e+00 0.00000000e+00 1.01644214e-05 1.03225511e+00]
d_N: [-9.83667012e-07 -1.98995604e-24]
d_B: [-2.45916753e-07  1.22958376e-06]
d_k: [ 1.22958376e-06 -2.45916753e-07 -9.83667012e-07 -1.98995604e-24]
lambda_max: 0.9687527694156928
lambda_k: 0.9681739707917424
Iteración: 7
x_k: [1.12903181e+00 7.74193638e-01 9.67745527e-02 1.15178382e-27]
grad_f: [-1.03226004 -5.16128907  0.          0.        ]
I_k: [1 0]
B: [[1 1]
 [5 1]]
N: [[1 0]
 [0 1]]
r: [0.00000000e+00 0.00000000e+00 2.78362519e-06 1.03225726e+00]
d_N: [-2.69384083e-07 -1.18893720e-27]
d_B: [-6.73460208e-08  3.36730104e-07]
d_k: [ 3.36730104e-07 -6.73460208e-08 -2.69384083e-07 -1.18893720e-27]
```

Apéndice C

Resolución numérica de otros problemas de optimización no lineal

En este capítulo vamos a enunciar varios problemas de optimización no lineal con restricciones y a resolverlos numéricamente usando las implementaciones en Python de los métodos de proyección del gradiente de Rosen y del gradiente reducido de Wolfe.

Consideremos los siguientes problemas de optimización no lineal con restricciones:

$$
\begin{aligned}
\text{Minimizar} \quad & 5\sum_{i=1}^{4} x_i - 5\sum_{i=1}^{4} x_i^2 - \sum_{i=5}^{13} x_i \\
\text{Sujeto a} \quad & 2x_1 + 2x_2 + x_{10} + x_{11} - 10 \leq 0, \\
& 2x_1 + 2x_3 + x_{10} + x_{12} - 10 \leq 0, \\
& 2x_1 + 2x_2 + x_{11} + x_{12} - 10 \leq 0, \\
& -8x_1 + x_{10} \leq 0, \\
& -8x_2 + x_{11} \leq 0, \\
& -8x_3 + x_{12} \leq 0, \\
& -2x_4 - x_5 + x_{10} \leq 0, \\
& -2x_6 - x_7 + x_{11} \leq 0, \\
& -2x_8 - x_9 + x_{12} \leq 0, \\
& -x_i \leq 0, \quad i = 1, \ldots, 13 \\
& x_i \leq 1, \quad i = 1, \ldots, 9, 13, \\
& x_i \leq 100, \quad i = 10, 11, 12.
\end{aligned}
\tag{C.1}
$$

$$
\begin{aligned}
\text{Minimizar} \quad & 3x_1^2 + 2x_1x_2 + 2x_2^2 - 4x_1 - 3x_1 - 10x_3 \\
\text{Sujeto a} \quad & x_1 + 2x_2 + x_3 \leq 8, \\
& -2x_1 + x_2 + x_4 \leq 1, \\
& -x_i \leq 0, \quad i = 1, 2, 3, 4.
\end{aligned}
\tag{C.2}
$$

$$
\begin{aligned}
\text{Minimizar} \quad & (2 - x_1)^2 - 8(x_2 - x_1)^2 + 2x_1^2 - 2x_1x_2 + e^{-2x_1 - x_2} \\
\text{Sujeto a} \quad & 5x_1 + 6x_2 \leq 30, \\
& -4x_1 + 3x_2 \leq 12, \\
& -x_i \leq 0, \quad i = 1, 2.
\end{aligned}
\tag{C.3}
$$

$$\begin{aligned} \text{Minimizar} \quad & 3e^{-2x_1+x_2} + 2x_1^2 + 2x_1x_2 + 3x_2^2 + x_1 + 3x_2 \\ \text{Sujeto a} \quad & 2x_1 + x_2 \leq 4, \\ & -x_1 + x_2 \leq 3, \\ & -x_i \leq 0, \quad i = 1,2. \end{aligned} \tag{C.4}$$

El problema C.1 es el problema G1 cuya referencia procede de la siguiente dirección web: Página web problema G1. Los problemas C.2, C.3 y C.4 se han sacado de los ejercicios propuestos del libro de Bazaraa, Sherali y Shetty "Nonlinear Programming. Theory and Algorithms"[23].

Para que el intérprete de Python pueda aplicar los algoritmos implementados y descritos en el apéndice A, traducimos los problemas de optimización anteriores a código Python como sigue:

```
# B.1
def f(x):
    x1, x2, x3, x4, x5, x6, x7, x8, x9, x10, x11, x12, x13 = x
    return 5*(x1 + x2 + x3 + x4) - 5*(x1*x1 + x2*x2 + x3*x3 + x4*x4) - (x5 + x6 + x7 + x8
     + x9 + x10 + x11 + x12 + x13)

A = np.array([
  [ 2,  2,  0,  0,  0,  0,  0,  0,  0,  1,  1,  0,  0],
  [ 2,  0,  2,  0,  0,  0,  0,  0,  0,  1,  0,  1,  0],
  [ 0,  2,  2,  0,  0,  0,  0,  0,  0,  0,  1,  1,  0],
  [-8,  0,  0,  0,  0,  0,  0,  0,  0,  1,  0,  0,  0],
  [ 0, -8,  0,  0,  0,  0,  0,  0,  0,  0,  1,  0,  0],
  [ 0,  0, -8,  0,  0,  0,  0,  0,  0,  0,  0,  1,  0],
  [ 0,  0,  0, -2, -1,  0,  0,  0,  0,  1,  0,  0,  0],
  [ 0,  0,  0,  0,  0, -2, -1,  0,  0,  0,  1,  0,  0],
  [ 0,  0,  0,  0,  0,  0,  0, -2, -1,  0,  0,  1,  0],
  [ 1,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0],
  [ 0,  1,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0],
  [ 0,  0,  1,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0],
  [ 0,  0,  0,  1,  0,  0,  0,  0,  0,  0,  0,  0,  0],
  [ 0,  0,  0,  0,  1,  0,  0,  0,  0,  0,  0,  0,  0],
  [ 0,  0,  0,  0,  0,  1,  0,  0,  0,  0,  0,  0,  0],
  [ 0,  0,  0,  0,  0,  0,  1,  0,  0,  0,  0,  0,  0],
  [ 0,  0,  0,  0,  0,  0,  0,  1,  0,  0,  0,  0,  0],
  [ 0,  0,  0,  0,  0,  0,  0,  0,  1,  0,  0,  0,  0],
  [ 0,  0,  0,  0,  0,  0,  0,  0,  0,  1,  0,  0,  0],
  [ 0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  1,  0,  0],
  [ 0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  1,  0],
  [ 0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  1],
  [-1,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0],
  [ 0, -1,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0],
  [ 0,  0, -1,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0],
  [ 0,  0,  0, -1,  0,  0,  0,  0,  0,  0,  0,  0,  0],
  [ 0,  0,  0,  0, -1,  0,  0,  0,  0,  0,  0,  0,  0],
  [ 0,  0,  0,  0,  0, -1,  0,  0,  0,  0,  0,  0,  0],
  [ 0,  0,  0,  0,  0,  0, -1,  0,  0,  0,  0,  0,  0],
  [ 0,  0,  0,  0,  0,  0,  0, -1,  0,  0,  0,  0,  0],
  [ 0,  0,  0,  0,  0,  0,  0,  0, -1,  0,  0,  0,  0],
  [ 0,  0,  0,  0,  0,  0,  0,  0,  0, -1,  0,  0,  0],
  [ 0,  0,  0,  0,  0,  0,  0,  0,  0,  0, -1,  0,  0],
  [ 0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0, -1,  0],
  [ 0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0, -1]])

b = np.array([10, 10, 10, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 100, 100, 100, 1,
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0])
Q = np.array([])
q = np.array([])
x1 = np.array([1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0])

# B.2
def f(x):
    x1, x2, x3, x4 = x
    return 3*x1*x1 + 2*x1*x2 + 2*x2*x2 - 4*x1 - 3*x2 - 10*x3

A = np.array([[1, 2, 1, 0], [-2, 1, 0, 1], [-1, 0, 0, 0], [0, -1, 0, 0], [0, 0, -1, 0],
    [0, 0, 0, -1]])
```

```
b = np.array([8, 1, 0, 0, 0, 0])
Q = np.array([])
q = np.array([])
x1 = np.array([8, 0, 0, 17])

# B.3
def f(x):
    x1, x2 = x
    return (2 - x1)*(2 - x1) - 8*(x2 - x1)*(x2 - x1) + 2*x1*x1 - 2*x1*x2 + e**(-2*x1 - x2
    )

A = np.array([[5, 6], [-4, 3], [-1, 0], [0, -1]])
b = np.array([30, 12, 0, 0])
Q = np.array([])
q = np.array([])
x1 = np.array([0, 0])

# B.4
def f(x):
    x1, x2= x
    return 3*e**(-2*x1 + x2) + 2*x1*x1 + 2*x1*x2 + 3*x2*x2 + x1 + 3*x2

A = np.array([[2, 1], [-1, 1], [-1, 0], [0, -1]])
b = np.array([4, 3, 0, 0])
Q = np.array([])
q = np.array([])
x1 = np.array([0, 0])
```

Bloque de código C.1: Otros problemas de optimización no lineal

A continuación, mostramos las salidas de las resoluciones de los problemas anteriores usando la implementación en Python del método de Rosen:

```
B.1
Iteración: 0
xk: [1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]
grad(f(xk)): [-5. -5. -5. -5. -1. -1. -1. -1. -1. -1. -1. -1. -1.]
A1: [[1 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 1 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 1 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 1 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 1 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 1 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 1 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 1 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 1 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 1 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 1 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 1 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 1]]
A2: [[ 2  2  0  0  0  0  0  0  0  1  1  0  0]
 [ 2  0  2  0  0  0  0  0  0  1  0  1  0]
 [ 0  2  2  0  0  0  0  0  0  0  1  1  0]
 [-8  0  0  0  0  0  0  0  0  1  0  0  0]
 [ 0 -8  0  0  0  0  0  0  0  0  1  0  0]
 [ 0  0 -8  0  0  0  0  0  0  0  0  1  0]
 [ 0  0  0 -2 -1  0  0  0  0  1  0  0  0]
 [ 0  0  0  0  0 -2 -1  0  0  0  1  0  0]
 [ 0  0  0  0  0  0  0 -2 -1  0  0  1  0]
 [ 0  0  0  0  0  0  0  0  0  1  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  1  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  1  0]
 [-1  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0 -1  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0 -1  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0 -1  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0 -1  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0 -1  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0 -1  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0 -1  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0 -1  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0 -1  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0 -1  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0 -1  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0 -1]]
```

```
P: [[0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]]
dk: [0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 1. 1. 0.]
lm:2.0
hat_bk: [ 4.  4.  4.  7.  7.  7.  2.  2.  2. 99. 99. 99.  1.  1.  1.  1.  1.  1.
  1.  1.  1.  1.  1.  1.  1.]
lambda_k: 1.9999959949686341
Iteración: 1
xk: [1.         1.         1.         1.         1.         1.
 1.         1.         1.         2.99999599 2.99999599 2.99999599
 1.        ]
grad(f(xk)): [-5. -5. -5. -5. -1. -1. -1. -1. -1. -1. -1. -1. -1.]
A1: [[2 2 0 0 0 0 0 0 0 1 1 0 0]
 [2 0 2 0 0 0 0 0 0 1 0 1 0]
 [0 2 2 0 0 0 0 0 0 0 1 1 0]
 [1 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 1 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 1 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 1 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 1 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 1 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 1 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 1 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 1 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 1]]
A2: [[-8  0  0  0  0  0  0  0  0  1  0  0  0]
 [ 0 -8  0  0  0  0  0  0  0  0  1  0  0]
 [ 0  0 -8  0  0  0  0  0  0  0  0  1  0]
 [ 0  0  0 -2 -1  0  0  0  0  1  0  0  0]
 [ 0  0  0  0  0 -2 -1  0  0  0  1  0  0]
 [ 0  0  0  0  0  0  0 -2 -1  0  0  1  0]
 [ 0  0  0  0  0  0  0  0  0  1  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  1  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  1  0]
 [-1  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0 -1  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0 -1  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0 -1  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0 -1  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0 -1  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0 -1  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0 -1  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0 -1  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0 -1  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0 -1  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0 -1  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0 -1]]
P: [[-1.33226763e-15 -7.94488517e-16 -3.42223158e-16  0.00000000e+00
   0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
   0.00000000e+00  6.66133815e-16 -5.55111512e-16  5.55111512e-16
   0.00000000e+00]
 [-7.97387978e-16 -4.44089210e-16 -9.45373234e-16  0.00000000e+00
   0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
   0.00000000e+00 -8.88178420e-16 -2.22044605e-16 -2.22044605e-16
   0.00000000e+00]
 [-2.76438514e-16  1.02381617e-16  0.00000000e+00  0.00000000e+00
   0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
   0.00000000e+00 -7.77156117e-16 -7.77156117e-16  4.44089210e-16
   0.00000000e+00]
 [ 0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
   0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
   0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
```

```
   0.00000000e+00]
 [ 0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
   0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
   0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
   0.00000000e+00]
 [ 0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
   0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
   0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
   0.00000000e+00]
 [ 0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
   0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
   0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
   0.00000000e+00]
 [ 0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
   0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
   0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
   0.00000000e+00]
 [ 0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
   0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
   0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
   0.00000000e+00]
 [ 0.00000000e+00  9.99200722e-16  6.66133815e-16  0.00000000e+00
   0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
   0.00000000e+00 -1.33226763e-15 -1.66533454e-15 -1.22124533e-15
   0.00000000e+00]
 [-4.44089210e-16 -4.44089210e-16 -7.77156117e-16  0.00000000e+00
   0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
   0.00000000e+00  1.11022302e-15  0.00000000e+00 -2.22044605e-16
   0.00000000e+00]
 [ 0.00000000e+00 -8.88178420e-16 -2.22044605e-16  0.00000000e+00
   0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
   0.00000000e+00  1.11022302e-16 -5.55111512e-16  2.10942375e-15
   0.00000000e+00]
 [ 0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
   0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
   0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
   0.00000000e+00]]
dk: [-1.16787627e-14 -1.22665197e-14 -1.98050751e-15  0.00000000e+00
  0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
  0.00000000e+00  4.10782519e-15 -7.43849426e-15 -3.88578059e-15
  0.00000000e+00]
w: [0.5 0.5 0.5 3.  3.  3.  5.  1.  1.  1.  1.  1.  1. ]
u: [0.5 0.5 0.5 3.  3.  3.  5.  1.  1.  1.  1.  1.  1. ]
Fin de la ejecución
Óptimo encontrado en: [1.         1.         1.         1.         1.         1.
 1.         1.         1.         2.99999599 2.99999599 2.99999599
 1.        ]
Función objetivo en el óptimo: -14.999987984905902

B.2
Iteración: 0
xk: [ 8.  0.  0. 17.]
grad(f(xk)): [ 44.  13. -10.   0.]
A1: [[ 1.  2.  1.  0.]
 [-2.  1.  0.  1.]
 [ 0.  0. -1.  0.]]
A2: [[-1.  0.  0.  0.]
 [ 0.  0.  0. -1.]]
P: [[ 1.33333333e-01 -6.66666667e-02  5.55111512e-17  3.33333333e-01]
 [-6.66666667e-02  3.33333333e-02  1.11022302e-16 -1.66666667e-01]
 [ 0.00000000e+00  0.00000000e+00 -2.22044605e-16  0.00000000e+00]
 [ 3.33333333e-01 -1.66666667e-01  0.00000000e+00  8.33333333e-01]]
dk: [-5.00000000e+00  2.50000000e+00 -2.22044605e-15 -1.25000000e+01]
lambda_k: 1.3599952467343315
Iteración: 1
xk: [ 1.20002377e+00  3.39998812e+00 -3.01979607e-15  5.94158209e-05]
grad(f(xk)): [ 10.00011883  13.         -10.           0.        ]
A1: [[ 1.  2.  1.  0.]
 [-2.  1.  0.  1.]
 [ 0.  0. -1.  0.]]
A2: [[-1.  0.  0.  0.]
 [ 0. -1.  0.  0.]
 [ 0.  0.  0. -1.]]
```

```
P: [[ 1.33333333e-01 -6.66666667e-02  5.55111512e-17  3.33333333e-01]
 [-6.66666667e-02  3.33333333e-02  1.11022302e-16 -1.66666667e-01]
 [ 0.00000000e+00  0.00000000e+00 -2.22044605e-16  0.00000000e+00]
 [ 3.33333333e-01 -1.66666667e-01  0.00000000e+00  8.33333333e-01]]
dk: [-4.66682511e-01  2.33341255e-01 -2.22044605e-15 -1.16670628e+00]
lambda_k: 4.682943099236314e-05
Iteración: 2
xk: [ 1.20000191e+00  3.39999904e+00 -3.01990005e-15  4.77962976e-06]
grad(f(xk)): [ 10.00000956  13.         -10.           0.        ]
A1: [[ 1.  2.  1.  0.]
 [-2.  1.  0.  1.]
 [ 0.  0. -1.  0.]]
A2: [[-1.  0.  0.  0.]
 [ 0. -1.  0.  0.]
 [ 0.  0.  0. -1.]]
P: [[ 1.33333333e-01 -6.66666667e-02  5.55111512e-17  3.33333333e-01]
 [-6.66666667e-02  3.33333333e-02  1.11022302e-16 -1.66666667e-01]
 [ 0.00000000e+00  0.00000000e+00 -2.22044605e-16  0.00000000e+00]
 [ 3.33333333e-01 -1.66666667e-01  0.00000000e+00  8.33333333e-01]]
dk: [-4.66667941e-01  2.33333971e-01 -2.22044605e-15 -1.16666985e+00]
lambda_k: 1.5648438243571474e-06
Iteración: 3
xk: [ 1.20000118e+00  3.39999941e+00 -3.01990353e-15  2.95397365e-06]
grad(f(xk)): [ 10.00000591  13.         -10.           0.        ]
A1: [[ 1.  2.  1.  0.]
 [-2.  1.  0.  1.]
 [ 0.  0. -1.  0.]]
A2: [[-1.  0.  0.  0.]
 [ 0. -1.  0.  0.]
 [ 0.  0.  0. -1.]]
P: [[ 1.33333333e-01 -6.66666667e-02  5.55111512e-17  3.33333333e-01]
 [-6.66666667e-02  3.33333333e-02  1.11022302e-16 -1.66666667e-01]
 [ 0.00000000e+00  0.00000000e+00 -2.22044605e-16  0.00000000e+00]
 [ 3.33333333e-01 -1.66666667e-01  0.00000000e+00  8.33333333e-01]]
dk: [-4.66667454e-01  2.33333727e-01 -2.22044605e-15 -1.16666864e+00]
lambda_k: 9.671276794739392e-07
Iteración: 4
xk: [ 1.20000073e+00  3.39999963e+00 -3.01990568e-15  1.82565611e-06]
grad(f(xk)): [ 10.00000365  13.         -10.           0.        ]
A1: [[ 1.  2.  1.  0.]
 [-2.  1.  0.  1.]
 [ 0.  0. -1.  0.]]
A2: [[-1.  0.  0.  0.]
 [ 0. -1.  0.  0.]
 [ 0.  0.  0. -1.]]
P: [[ 1.33333333e-01 -6.66666667e-02  5.55111512e-17  3.33333333e-01]
 [-6.66666667e-02  3.33333333e-02  1.11022302e-16 -1.66666667e-01]
 [ 0.00000000e+00  0.00000000e+00 -2.22044605e-16  0.00000000e+00]
 [ 3.33333333e-01 -1.66666667e-01  0.00000000e+00  8.33333333e-01]]
dk: [-4.66667154e-01  2.33333577e-01 -2.22044605e-15 -1.16666788e+00]
lambda_k: 5.977181627555615e-07
...
Iteración: 48
xk: [7.13907610e-09 1.14411498e-08 7.99999997e+00 9.17391968e-09]
grad(f(xk)): [ -3.99999993  -2.99999994 -10.           0.        ]
A1: [[ 1.  2.  1.  0.]
 [-1.  0.  0.  0.]
 [ 0.  0.  0. -1.]]
A2: [[-2.  1.  0.  1.]
 [ 0. -1.  0.  0.]
 [ 0.  0. -1.  0.]]
P: [[-2.22044605e-16  0.00000000e+00  0.00000000e+00  0.00000000e+00]
 [ 1.11022302e-16  2.00000000e-01 -4.00000000e-01  0.00000000e+00]
 [ 5.55111512e-17 -4.00000000e-01  8.00000000e-01  0.00000000e+00]
 [ 0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00]]
dk: [-8.88178405e-16 -3.40000001e+00  6.80000002e+00  0.00000000e+00]
lambda_k: 1.2853324547181051e-09
Iteración: 49
xk: [7.13907610e-09 7.07101946e-09 7.99999998e+00 9.17391968e-09]
grad(f(xk)): [ -3.99999994  -2.99999996 -10.           0.        ]
A1: [[ 1.  2.  1.  0.]
 [-1.  0.  0.  0.]
 [ 0. -1.  0.  0.]
```

```
 [ 0.  0.  0. -1.]]
A2: [[-2.  1.  0.  1.]
 [ 0.  0. -1.  0.]]
P: [[ 3.33066907e-16  0.00000000e+00  0.00000000e+00  0.00000000e+00]
 [ 4.44089210e-16  8.88178420e-16  0.00000000e+00  0.00000000e+00]
 [-9.99200722e-16  4.44089210e-16 -2.22044605e-16  0.00000000e+00]
 [ 0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00]]
dk: [ 1.33226761e-15  4.44089204e-15 -4.88498127e-15  0.00000000e+00]
w: [10.          6.00000006 17.00000004  0.        ]
u: [10.          6.00000006 17.00000004  0.        ]
Fin de la ejecución
Óptimo encontrado en: [7.13907610e-09 7.07101946e-09 7.99999998e+00 9.17391968e-09]
Función objetivo en el óptimo: -79.99999983695892

B.3
Iteración: 0
xk: [0. 0.]
grad(f(xk)): [-6. -1.]
A1: [[-1  0]
 [ 0 -1]]
A2: [[ 5  6]
 [-4  3]]
P: [[0. 0.]
 [0. 0.]]
dk: [0. 0.]
w: [-6. -1.]
u: [-6. -1.]
Iteración: 0
xk: [0. 0.]
grad(f(xk)): [-6. -1.]
A1: [[ 0 -1]]
A2: [[ 5  6]
 [-4  3]]
P: [[1. 0.]
 [0. 0.]]
dk: [6. 0.]
lambda_k: 0.9999940391390134
Iteración: 1
xk: [5.99996423 0.        ]
grad(f(xk)): [-63.99965464  83.99949314]
A1: [[ 5  6]
 [ 0 -1]]
A2: [[-4  3]
 [-1  0]]
P: [[-6.66133815e-16  0.00000000e+00]
 [-1.38777878e-16 -2.22044605e-16]]
dk: [-4.26323341e-14  9.76989800e-15]
w: [ 12.79993093 160.79907871]
u: [ 12.79993093 160.79907871]
Fin de la ejecución
Óptimo encontrado en: [5.99996423 0.        ]
Función objetivo en el óptimo: -199.99770489112498

B.4
Iteración: 0
xk: [0. 0.]
grad(f(xk)): [-5.  6.]
A1: [[-1.  0.]
 [ 0. -1.]]
A2: [[ 2.  1.]
 [-1.  1.]]
P: [[0. 0.]
 [0. 0.]]
dk: [0. 0.]
w: [-5.  6.]
u: [-5.  6.]
Iteración: 0
xk: [0. 0.]
grad(f(xk)): [-5.  6.]
A1: [[ 0. -1.]]
A2: [[ 2.  1.]
 [-1.  1.]]
P: [[1. 0.]
```

```
 [0. 0.]]
dk: [5. 0.]
lambda_k: 0.08205500350701746
Iteración: 1
xk: [0.41027502 0.        ]
grad(f(xk)): [-3.67394291e-05  5.14111844e+00]
A1: [[ 0. -1.]]
A2: [[ 2.  1.]
 [-1.  1.]
 [-1.  0.]]
P: [[1. 0.]
 [0. 0.]]
dk: [3.67394291e-05 0.00000000e+00]
lambda_k: 0.10773786208178042
Iteración: 2
xk: [0.41027898 0.        ]
grad(f(xk)): [1.83926252e-09 5.14111590e+00]
A1: [[ 0. -1.]]
A2: [[ 2.  1.]
 [-1.  1.]
 [-1.  0.]]
P: [[1. 0.]
 [0. 0.]]
dk: [-1.83926252e-09  0.00000000e+00]
w: [5.1411159]
u: [5.1411159]
Fin de la ejecución
Óptimo encontrado en: [0.41027898 0.        ]
Función objetivo en el óptimo: 2.067494602273939
```

De igual manera, presentamos las salidas de las resoluciones de los problemas presentados usando la implementación en Python del método de Wolfe. Notar que, en el caso del problema C.1, no mostramos dicha salida ya que el método de Wolfe no es capaz de resolverlo porque no se cumple la condición de no degeneración exigida sobre las soluciones básicas del recinto.

```
B.2
Iteración: 0
x_k: [ 8.  0.  0. 17.]
grad_f: [ 44.  13. -10.   0.]
I_k: [0 3]
B: [[ 1.  0.]
 [-2.  1.]]
N: [[2. 1.]
 [1. 0.]]
r: [  0. -75. -54.   0.]
d_N: [75. 54.]
d_B: [-204. -483.]
d_k: [-204.   75.   54. -483.]
lambda_max: 0.035196687370600416
lambda_k: 0.03519059587035381
Iteración: 1
x_k: [0.82111844 2.63929469 1.90029218 0.00294219]
grad_f: [  6.20530004   9.19941565 -10.           0.        ]
I_k: [2 1]
B: [[1. 2.]
 [0. 1.]]
N: [[ 1.  0.]
 [-2.  1.]]
r: [ 74.60413133   0.           0.         -29.19941565]
d_N: [-61.25882812  29.19941565]
d_B: [ 364.69297187 -151.71707188]
d_k: [ -61.25882812 -151.71707188  364.69297187   29.19941565]
lambda_max: 0.013404083422854957
lambda_k: 0.013398009980999609
Iteración: 2
x_k: [3.72051931e-04 6.06587847e-01 6.78645225e+00 3.94156257e-01]
grad_f: [ -2.78459199  -0.57290451 -10.           0.        ]
I_k: [1 2]
B: [[2. 1.]
 [1. 0.]]
N: [[ 1.  0.]
 [-2.  1.]]
```

```
r: [ 4.60695990e+01  1.55431223e-15  0.00000000e+00 -1.94270955e+01]
d_N: [-1.71402832e-02  1.94270955e+01]
d_B: [-19.46137606  38.9398924 ]
d_k: [-1.71402832e-02 -1.94613761e+01  3.89398924e+01  1.94270955e+01]
lambda_max: 0.021706288353778216
lambda_k: 0.021700209864097004
Iteración: 3
x_k: [1.04187035e-07 1.84271902e-01 7.63145609e+00 8.15728306e-01]
grad_f: [ -3.63145557  -2.26291218 -10.           0.        ]
I_k: [3 2]
B: [[0. 1.]
 [1. 0.]]
N: [[ 1.  2.]
 [-2.  1.]]
r: [ 6.36854443 17.73708782  0.          0.        ]
d_N: [-6.63519761e-07 -3.26844691e+00]
d_B: [3.26844559 6.53689449]
d_k: [-6.63519761e-07 -3.26844691e+00  6.53689449e+00  3.26844559e+00]
lambda_max: 0.05637904092779802
lambda_k: 0.05637301043894544
Iteración: 4
x_k: [6.67824285e-08 1.97103327e-05 7.99996051e+00 9.99980423e-01]
grad_f: [ -3.99996018  -2.99992103 -10.           0.        ]
I_k: [3 2]
B: [[0. 1.]
 [1. 0.]]
N: [[ 1.  2.]
 [-2.  1.]]
r: [ 6.00003982 17.00007897  0.          0.        ]
d_N: [-4.00697230e-07 -3.35077212e-04]
d_B: [0.00033428 0.00067056]
d_k: [-4.00697230e-07 -3.35077212e-04  6.70555121e-04  3.34275818e-04]
lambda_max: 0.05882325614349941
lambda_k: 0.05881696421329613
Iteración: 5
x_k: [4.32146338e-08 2.10828243e-09 7.99999995e+00 1.00000008e+00]
grad_f: [ -3.99999974  -2.99999991 -10.           0.        ]
I_k: [3 2]
B: [[0. 1.]
 [1. 0.]]
N: [[ 1.  2.]
 [-2.  1.]]
r: [ 6.00000026 17.00000009  0.          0.        ]
d_N: [-2.59287814e-07 -3.58408015e-08]
d_B: [-4.82734827e-07  3.30969417e-07]
d_k: [-2.59287814e-07 -3.58408015e-08  3.30969417e-07 -4.82734827e-07]
Fin de la ejecución
Óptimo encontrado en: [4.32146338e-08 2.10828243e-09 7.99999995e+00 1.00000008e+00]
Función objetivo en el óptimo: -79.9999997048714

B.3
Iteración: 0
x_k: [ 0.  0. 30. 12.]
grad_f: [-6. -1.  0.  0.]
I_k: [3 2]
B: [[0 1]
 [1 0]]
N: [[ 5  6]
 [-4  3]]
r: [-6. -1.  0.  0.]
d_N: [6. 1.]
d_B: [ 21. -36.]
d_k: [  6.   1. -36.  21.]
lambda_max: 0.8333333333333334
lambda_k: 0.8333286416304795
Iteración: 1
x_k: [4.99997185e+00 8.33328642e-01 1.68901303e-04 2.94999015e+01]
grad_f: [-42.33315698  56.6663279    0.           0.        ]
I_k: [0 3]
B: [[ 5  0]
 [-4  1]]
N: [[6 1]
 [3 0]]
```

```
r: [  0.         107.46611627   8.4666314    0.        ]
d_N: [-8.95545927e+01 -1.43002507e-03]
d_B: [107.46579724 698.52696705]
d_k: [ 1.07465797e+02 -8.95545927e+01 -1.43002507e-03  6.98526967e+02]
lambda_max: 0.00930525857521708
lambda_k: 0.00930177001305669
Iteración: 2
x_k: [5.99959398e+00 3.12416763e-04 1.55599538e-04 3.59974387e+01]
grad_f: [-63.99157826  83.9893109    0.           0.        ]
I_k: [0 3]
B: [[ 5  0]
 [-4  1]]
N: [[6 1]
 [3 0]]
r: [7.10542736e-15 1.60779205e+02 1.27983157e+01 0.00000000e+00]
d_N: [-0.05023012 -0.00199141]
d_B: [0.06067442 0.39338806]
d_k: [ 0.06067442 -0.05023012 -0.00199141  0.39338806]
lambda_max: 0.006219709825944455
lambda_k: 0.00621566517481643
Iteración: 3
x_k: [5.99997111e+00 2.03163307e-07 1.43221588e-04 3.59998838e+01]
grad_f: [-63.99972056  83.99958617   0.           0.        ]
I_k: [0 3]
B: [[ 5  0]
 [-4  1]]
N: [[6 1]
 [3 0]]
r: [7.10542736e-15 1.60799251e+02 1.27999441e+01 0.00000000e+00]
d_N: [-3.26685075e-05 -1.83322832e-03]
d_B: [0.00040585 0.0017214 ]
d_k: [ 4.05847874e-04 -3.26685075e-05 -1.83322832e-03  1.72139702e-03]
lambda_max: 0.006218934446076816
lambda_k: 0.006214890714724918
Iteración: 4
x_k: [5.99997363e+00 1.32102668e-10 1.31828274e-04 3.59998945e+01]
grad_f: [-63.99974863  83.99962473   0.           0.        ]
I_k: [0 3]
B: [[ 5  0]
 [-4  1]]
N: [[6 1]
 [3 0]]
r: [  0.         160.79932309  12.79994973   0.        ]
d_N: [-2.12420196e-08 -1.68739529e-03]
d_B: [0.0003375  0.00135008]
d_k: [ 3.37504547e-04 -2.12420196e-08 -1.68739529e-03  1.35008192e-03]
lambda_max: 0.006218931652217006
lambda_k: 0.006214887924179259
Iteración: 5
x_k: [5.99997573e+00 8.58969502e-14 1.21341302e-04 3.59999029e+01]
grad_f: [-63.99976961  83.9996541    0.           0.        ]
I_k: [0 3]
B: [[ 5  0]
 [-4  1]]
N: [[6 1]
 [3 0]]
r: [7.10542736e-15 1.60799378e+02 1.27999539e+01 0.00000000e+00]
d_N: [-1.38121761e-11 -1.55316307e-03]
d_B: [0.00031063 0.00124253]
d_k: [ 3.10632631e-04 -1.38121761e-11 -1.55316307e-03  1.24253057e-03]
lambda_max: 0.006218929542851246
lambda_k: 0.006214885817315685
...
Iteración: 93
x_k: [5.99999998e+000 3.04566944e-294 8.24169109e-008 3.59999999e+001]
grad_f: [-64.00001212  83.99999363   0.           0.        ]
I_k: [0 3]
B: [[ 5  0]
 [-4  1]]
N: [[6 1]
 [3 0]]
r: [  0.         160.80000817  12.80000242   0.        ]
d_N: [-4.89743671e-292 -1.05493666e-006]
```

```
d_B: [2.10987332e-07 8.43949327e-07]
lambda_max: 0.006218905156535062
lambda_k: 0.006214861459927195
Iteración: 94
x_k: [5.99999998e+000 1.98037482e-297 7.58606257e-008 3.59999999e+001]
grad_f: [-64.00001214  83.99999364   0.           0.        ]
I_k: [0 3]
B: [[ 5  0]
 [-4  1]]
N: [[6 1]
 [3 0]]
r: [  0.         160.80000821  12.80000243   0.        ]
d_N: [-3.18444287e-295 -9.71016193e-007]
d_B: [1.94203239e-07 7.76812954e-07]
d_k: [ 1.94203239e-007 -3.18444287e-295 -9.71016193e-007  7.76812954e-007]
Fin de la ejecución
Óptimo encontrado en: [5.99999998e+000 1.98037482e-297 7.58606257e-008 3.59999999e+001]
Función objetivo en el óptimo: -199.99999288477147

B.4
Iteración: 0
x_k: [0. 0. 4. 3.]
grad_f: [-5.  6.  0.  0.]
I_k: [3 2]
B: [[0 1]
 [1 0]]
N: [[ 2  1]
 [-1  1]]
r: [-5.  6.  0.  0.]
d_N: [ 5. -0.]
d_B: [  5. -10.]
d_k: [  5.  -0. -10.   5.]
lambda_max: 0.4
lambda_k: 0.08205500350701746
Iteración: 1
x_k: [0.41027502 0.         3.17944996 3.41027502]
grad_f: [-3.67394291e-05  5.14111844e+00  0.00000000e+00  0.00000000e+00]
I_k: [2 3]
B: [[1 0]
 [0 1]]
N: [[ 2  1]
 [-1  1]]
r: [-3.67394291e-05  5.14111844e+00  0.00000000e+00  0.00000000e+00]
d_N: [ 3.67394291e-05 -0.00000000e+00]
d_B: [-7.34788582e-05  3.67394291e-05]
d_k: [ 3.67394291e-05 -0.00000000e+00 -7.34788582e-05  3.67394291e-05]
lambda_max: 43270.26905871362
lambda_k: 0.10773786208178042
Iteración: 2
x_k: [0.41027898 0.         3.17944205 3.41027898]
grad_f: [1.83926252e-09 5.14111590e+00 0.00000000e+00 0.00000000e+00]
I_k: [2 3]
B: [[1 0]
 [0 1]]
N: [[ 2  1]
 [-1  1]]
r: [1.83926252e-09 5.14111590e+00 0.00000000e+00 0.00000000e+00]
d_N: [-7.54610743e-10 -0.00000000e+00]
d_B: [ 1.50922149e-09 -7.54610743e-10]
d_k: [-7.54610743e-10 -0.00000000e+00  1.50922149e-09 -7.54610743e-10]
Fin de la ejecución
Óptimo encontrado en: [0.41027898 0.         3.17944205 3.41027898]
Función objetivo en el óptimo: 2.067494602273939
```