FACULTAD DE CIENCIAS

Métodos de puntos interiores para Programación lineal: teoría y práctica

(Interior Point Methods for Linear Programming: Theory and Practice)

Trabajo de Fin de Grado
para acceder al
GRADO EN MATEMÁTICAS

Autor: Borja Fernández Ortiz
Directora: María Cecilia Pola Méndez
Junio-2025

Agradecimientos

Primero de todo, agradecer a Cecilia, por depositar su confianza en mi, dedicarme su tiempo y guiarme en el proceso que ha sido este trabajo de fin de grado. Ha sido una suerte y un placer tenerla a ella como tutora en el final de mi etapa universitaria.

También agradecer a mi familia, en especial a mi madre y abuelas, por apoyarme desde siempre y ser un pilar fundamental en mi vida. A mis amigos, a los que conozco desde siempre y a los que he conocido durante la carrera. Por los descansitos en la sala de estudio, los nervios antes de un examen y los cinco minutos para un café que terminaban convirtiéndose en media hora.

Agradecer por último a todos los profesores que me han enseñado a lo largo de mi formación, en especial a Marisa, sus clases de matemáticas fueron las que me dieron el último empujón para decidirme a entrar a la carrera y de no ser por ella no estaría aquí hoy.

Gracias.

Resumen:

Los métodos de puntos interiores han sido uno de los mayores logros de la optimización numérica de los últimos 50 años, convirtiéndose en unos buenos candidatos para resolver problemas de talla grande. En programación lineal, en vez de pivotar entre los vértices del conjunto de puntos admisibles para buscar una solución, como hace el Simplex, calculan candidatos por el interior de la región admisible.

En este trabajo se han estudiado los métodos de puntos interiores primales-duales para programación lineal. En la parte teórica se han tratado con detalle algunos aspectos como el concepto del camino central y la relación con los métodos de penalización. A continuación, se han considerado dos métodos de puntos interiores: el LSPF, para el que se estudia la convergencia global, y el MPC, por ser similar a los que se implementan en MATLAB. En la parte práctica, se han comparado experimentalmente los métodos de la *Optimization Toolbox* de MATLAB para programación lineal. Para ello se han usado dos colecciones de tests: problemas de la librería NETLIB y problemas propios generados aleatoriamente.

Palabras Clave: camino central, dualidad, métodos de puntos interiores, programación lineal.

Abstract:

Interior point methods have been one of the biggest achievements in numerical optimization over the last 50 years, turning out to be great candidates for solving large size problems. In linear programming, instead of jumping between the vertices of the feasible point set to search for a solution, like the Simplex method, interior point methods calculate candidates inside the feasible point set.

The main focus of this work are primal dual interior point methods for linear programming. Firstly, some theoretical aspects such as the concept of central path and its relationship with the penalty methods have been discussed. Next, the introducction of two algorithms: the LSPF, with a study of its convergence, and the MPC, because of its similarity with the algorithms implemented on MATLAB. Finallly, the practical section focuses on the comparison between all the methods available in MATLAB's Optimization Toolbox for linear programming. For this purpose, two colections of tests have been used: a set of problems of NETLIB's library and a set of randomly generated problems.

Key Words: Central path, duality, interior point methods, linear programming.

Índice general

0. Introducción **9**

1. Métodos de puntos interiores primales-duales **13**
1.1. Introducción a dualidad 13
1.2. El salto de dualidad 14
1.3. Dirección en los métodos primales-duales 15
1.4. El camino central 17
1.4.1. Relación con los métodos de penalización 22
1.4.2. Entornos del camino central 23
1.5. Algoritmo LSPF y su convergencia global 24
1.6. Dirección afín y dirección de centralización 28
1.7. Algoritmo MPC 29
1.7.1. Cálculo numérico de las direcciones 30

2. Resultados Numéricos **33**
2.1. Software: *linprog* de MATLAB 33
2.2. Experimentos con tests de NETLIB 33
Dual-Simplex vs. *Interior-Point* 34
Interior-Point vs. *Interior-Point-Legacy* 36
Conclusión 38
2.3. Experimentos generados de forma aleatoria 38
Resultados con las matrices de bloque triangular 39
Resultados con las matrices de bloque simétrico 41
Conclusiones 42
2.4. Tablas 43
Dual-Simplex vs. *Interior-Point* 43
Interior-Point vs. *Interior-Point-Legacy* 47
Problemas generados de forma aleatoria 51

A. **53**
A.1. Notación. 53
A.2. Resultados de optimización 53

Capítulo 0

Introducción

La programación lineal es la rama de la optimización que se ocupa de la minimización y maximización de funciones lineales cuyas variables están sometidas a restricciones también lineales. Los problemas que se resuelven con la programación lineal tienen una función objetivo, $f(x)$, que se va a minimizar o maximizar y que depende de un número finito de variables, $x \in \mathbb{R}^n$. Como la función es lineal se puede escribir como $c^T x$ para un $c \in \mathbb{R}^n$. Las restricciones que van a poder afectar a las variables son de igualdad y de desigualdad. Si el número de restricciones de igualdad se denota como n_I y el restricciones de desigualdad como n_D, un problema de programación lineal se escribe de la siguiente forma:

$$(PL) \begin{cases} \min\limits_{x \in \mathbb{R}^n} & f(x) = c^T x \\ \text{sujeto a} & a_i^T x = b_i \quad \text{para } i = 1, \ldots, n_I \\ & a_i^T x \leq b_i \quad \text{para } i = n_I + 1, \ldots, n_I + n_D \end{cases}, \tag{1}$$

donde $a_i \in \mathbb{R}^n$ y $b_i \in \mathbb{R}$. Como es habitual, en la parte teórica de este trabajo se tratarán los problemas de programación lineal en forma estándar. En estos las únicas restricciones de desigualdad tienen la forma: $x \geq 0$.

La programación lineal como disciplina de optimización numérica empieza en 1947 cuando Dantzig plantea un problema de minimización para el ejército estadounidense y desarrolla el conocido método del Simplex. Existen intentos de optimizar funciones lineales previos a 1947, pero ninguno tiene influencia significativa en lo que es la programación lineal hoy en día.

En 1946 a Dantzig se le pide mecanizar los procesos de planificación en el ejército de EEUU: encontrar una forma más rápida de gestionar el despliegue, entrenamiento y suministro logístico de materiales y tropas. Para su modelo Dantzig se inspiró en el trabajo de Wassily Leontief, que en 1932 publicó una estructura matricial que bautizó como *"Interindustry Input Output Model of the American Economy"*. Inicialmente no había una función objetivo en el modelo, ya que en el momento no existía forma de abordar tal concepto en la práctica computacional. Dantzig presenta un ejemplo en [1] para mostrar la dificultad de encontrar una solución en esos tiempos. Se quiere asignar 70 personas a 70 tareas, para ello se asocia el valor v_{ij} al beneficio de emparejar la persona i a la tarea j. A cada pareja entre una persona y una tarea se le asocia una variable, x_{ij}. Se asigna a cada variable el valor 1 si se produce el emparejamiento entre la persona y la tarea, y el 0 si no. Por lo tanto, se tienen 4900 variables, esto es un total de 70! posibilidades. Las restricciones son que

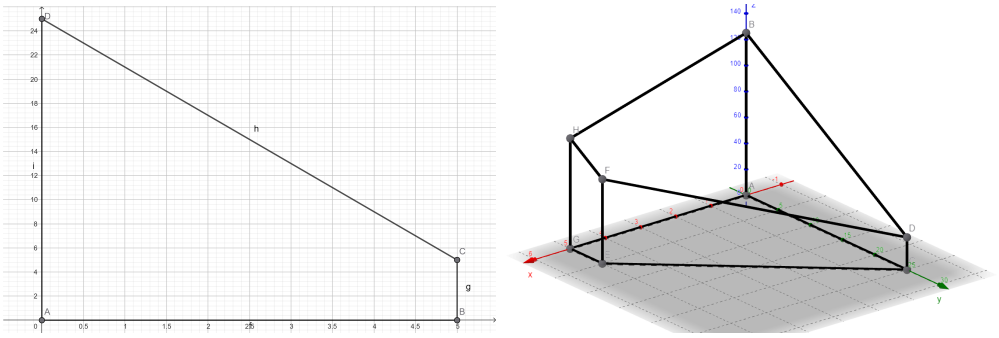
todos las tareas deben de estar cubiertas y que todas las personas deben de tener asignada una tarea. Así que se originan 140 restricciones. La dificultad era que había que comparar las 70! posibilidades para seleccionar cual de ellas proporciona una mayor suma total de los beneficios. Lo que suponía un gran inconveniente a la hora de buscar una solución, sobre todo considerando que el ejército tenía problemas de órdenes mayores que el de este ejemplo.

El desarrollo del Simplex supuso una de las mayores revoluciones en el campo de la optimización. Se convirtió en una herramienta muy útil para economistas, ingenieros y en la gestión tanto de trabajadores como de recursos. Con la capacidad de resolver problemas de optimización mediante el cálculo de nuevos candidatos a solución en los vértices del conjunto de admisibilidad, es capaz de resolver gran variedad de problemas, asegurando una convergencia global en un número finito de iteraciones. Desde su invención y hasta finales del siglo XX, el Simplex fue el único método que se utilizó para resolver problemas de programación lineal, pero, a pesar de garantizar la resolución, una cuestión teórica que quedaba pendiente era que no se tenían garantías de una complejidad polinomial del método.

En 1972 Klee y Minty [8] teorizan que, dado un problema de tamaño n, el método Simplex podía llegar a ejecutar hasta $2^n - 1$ iteraciones antes de encontrar una solución. En concreto esto ocurría con la regla de pivotación que más se usaba en el momento para el Simplex, propuesta por Dantzig. Para ello habría que generar un conjunto de puntos admisibles, con unas condiciones precisas, equivalente a un cubo $n - dimensional$, a este cubo se le llama de Klee-Minty. Una de las generalizaciones del problema de optimización con un cubo de Klee-Minty es el siguiente:

$$(KM_n) \begin{cases} \text{máx} & \sum_{i=1}^{n} 2^{n-i} x_i \\ \text{sujeto a} & x_i + \sum_{j=1}^{i-1} 2^{i+1-j} x_j \leq 5^i \quad i = 1, 2, \ldots, n \\ & x \geq 0 \end{cases} . \tag{2}$$

El conjunto de puntos admisibles de este problema para dos y tres variables se puede ver en la figura 1.



(a) $n = 2$ (b) $n = 3$

Figura 1: Representación gráfica del cubo de Klee-Minty para varias dimensiones.

Para mostrar algunas variantes del Simplex, se da la definición siguiente:

Definición 0.0.1. *Un conjunto de restricciones del problema (PL), I_k, se le llama* ***conjunto de trabajo en*** $x^{(k)}$ *si*

- *Las restricciones son activas en* $x^{(k)}$ $(a_i^T x^{(k)} = b_i,\ i \in I_k)$.
- *Los vectores* $\{a_i\}_{i\in I_k}$ *son linealmente independientes.*

La matriz A_k será la matriz cuyas columnas son los vectores $\{a_i\}_{i\in I_k}$. Al conjunto de trabajo se le hace referencia con el conjunto de índices de las restricciones y con la matriz asociada dichas restricciones, (I_k, A_k). A_k es una matriz de tamaño $n \times m_k$, siendo m_k el número de restricciones en el conjunto de trabajo.

A continuación, se presenta un esquema general del algoritmo del Simplex para la formulación (PL):

Algoritmo 0.0.1. ***(Simplex para formulación general)***

(1) Calcular un punto admisible $x^{(0)}$, y establecer $k = 0$. Obtener un conjunto de trabajo maximal, (I_k, A_k), asociado a $x^{(0)}$.

(2) Si $c \notin R(A_k) \Longrightarrow$ Ir a 3.
En otro caso, computar los candidatos a multiplicadores de Lagrange $\lambda^{(k)}$ resolviendo el sistema: $A_k^T \lambda = -c$.

- *Si $\lambda_i^{(k)} \geq 0$ para todas las restricciones de desigualdad de I_k, entonces se ha encontrado una solución global del problema. STOP.*
- *En otro caso, eliminar de I_k una restricción de desigualdad con $\lambda_q^{(k)} < 0$.*

(3) Calcular una dirección de descenso $d^{(k)}$ tal que $a_j^T d^{(k)} = 0$ si $j \in I_k$.
Si $\{j \notin I_k, :\ a_j^T d^{(k)} > 0\} = \emptyset$, no existe solución para (PL). STOP.

(4) Calcular el desplazamiento ρ_k mediante la fórmula:

$$\rho_k = \min_{j\notin I_k,\ a_j^T d^{(k)}>0} \frac{b_j - a_j^T x^{(k)}}{a_j^T d^{(k)}}.$$

Tomar $x^{(k+1)} = x^{(k)} + \rho_k d^{(k)}$.

(5) Añadir al conjunto de trabajo una restricción asociada al desplazamiento ρ_k. Volver a 2.

Hay dos momentos en los que hay varias opciones para proceder en el algoritmo: en el paso 2 con la elección de la restricción que abandona el conjunto de trabajo (si es que hay más de un candidato a multiplicador de Lagrange negativo) y en el paso 5 con la elección de la restricción que entra en el conjunto de trabajo (si hay varias asociadas a ese desplazamiento mínimo). Esas decisiones pueden marcar la convergencia del método. En el problema (KM), escogiendo como $x^{(0)}$ el origen, si el criterio de selección es el primer candidato a multiplicador de Lagrange negativo, el método no llegará a las $2^n - 1$ iteraciones,

si por otro lado se escoge el menor de todos esos candidatos, el método alcanzará esa cota de iteraciones antes de encontrar una solución.

A pesar de que este caso sea una excepción y que la mayoría de problemas se resuelvan en muchas menos iteraciones, es de aquí que surge el interés teórico en un método del que se pueda garantizar una complejidad polinomial. El primero en proponer un método para programación lineal con complejidad polinomial fue Khachiyan [5], quien prometía una complejidad $O(n^6L^2)$, para n el tamaño del problema y L un factor relacionado a los bits que se necesitan para almacenar los datos del problema. En la práctica el método de Khachiyan no resultó ser un buen competidor contra el Simplex. Años más tarde, en 1984, Karmakar [4] planteó su propio método, con una mejora en la complejidad, rebajándola a $O(n^{3.5}L^2)$. Este método dio buenos resultados, pero fue patentado y hasta 2006 no quedó libre de patente.

Después de que el método de Karmakar fuese patentado, se empezó a investigar en otros métodos, los que más atención tuvieron fueron los de puntos interiores primales-duales, pruebas computacionales demostraron que eran más eficientes que otros tipos de métodos de puntos interiores. En 1992 se tuvo una de las primeras implementaciones de un método de puntos interiores que dio buenos resultados, la implementación de Mehrotra [6]. Es a partir de este algoritmo del que surgen muchos de los códigos usados actualmente, por ejemplo $LIPSOL$, que es la base de los algoritmos incorporados en MATLAB bajo el nombre *linprog*.

Está ampliamente aceptado a día de hoy que los métodos de puntos de puntos interiores son una buena alternativa ante el método Simplex, sobre todo en problemas para talla grande [3]. Cabe destacar que con el auge de métodos distintos al Simplex, muchos desarrolladores comerciales de este algoritmo han recibido grandes presiones económicas, lo que ha derivado en una gran mejora de las implementaciones en los últimos años.

En este trabajo, en el capítulo 2 se estudian desde un punto de vista teórico los métodos de puntos interiores primales-duales. Primero se hace una breve introducción sobre dualidad, explicando términos que serán útiles más adelante, como es el salto de dualidad. Después se introduce el camino central y dos tipos de entorno útiles para el estudio de la convergencia global. Por último, en el capítulo 2 se introducen los conceptos de dirección afín y de centralización, que son la base de las direcciones de los métodos de puntos interiores primales-duales. En el capítulo 3 se presentan resultados numéricos obtenidos en el desarrollo del trabajo con el objetivo de comparar un método Simplex con métodos de puntos interiores. Se ha trabajado con dos tipos de problemas:

- Problemas de la biblioteca NETLIB [2].
- Problemas generados de forma aleatoria para este trabajo.

Capítulo 1

Métodos de puntos interiores primales-duales

Los métodos de puntos interiores tienen en común que reemplazan las restricciones de desigualdad $x \geq 0$, aplican la teoría de dualidad y usan el método de Newton para resolver sistemas de ecuaciones no lineales.

1.1. Introducción a dualidad

Se considera un problema de programación lineal en forma estándar al siguiente:

$$(P) \begin{cases} \min\limits_{x \in \mathbb{R}^n} & f(x) = c^T x \\ \text{sujeto a} & Ax = b \\ & x \geq 0 \end{cases}, \tag{1.1}$$

donde $c \in \mathbb{R}^n$, $b \in \mathbb{R}^m$ y $A \in \mathbb{R}^{m \times n}$ de rango máximo con $m < n$. Si $m = n$ para determinar la solución de (P) bastaría resolver el sistema de ecuaciones lineales, $Ax = b$. Además, en toda la memoria se va a suponer que existe solución para el problema.

Dado un problema, se define su conjunto de puntos admisibles como:

Definición 1.1.1. ***(Conjunto de puntos admisibles)*** *Se considera el conjunto de puntos admisibles de un problema (P) al conjunto formado por los puntos que cumplen todas las restricciones de dicho problema, y se denomina $\mathcal{A}(P)$.*

El siguiente resultado, demostrado en Optimización I, presenta la relación entre el problema (P), llamado **problema primal**, y su dual.

Teorema 1.1.1. *Sean $\bar{x}$ solución del problema de optimización (P) y $(\bar{\lambda}, \bar{s})$ multiplicadores de Lagrange asociados a $\bar{x}$. Entonces $(\bar{\lambda}, \bar{s})$ son solución global del problema:*

$$(PD) \begin{cases} \max\limits_{\lambda \in \mathbb{R}^m,\ s \in \mathbb{R}^n} & D(\lambda, s) = b^T \lambda \\ \textit{sujetos a} & A^T \lambda + s = c \\ & s \geq 0 \end{cases}. \tag{1.2}$$

Además, se tiene la igualdad de los valores óptimos de las dos funciones objetivo:

$$f(\bar{x}) = D(\bar{\lambda}, \bar{s}). \tag{1.3}$$

Al problema (PD) se le conoce como el **problema dual** de (P).

1.2. El salto de dualidad

A continuación, se presenta la relación de los valores de las funciones objetivo primal y dual en puntos admisibles.

Proposición 1.2.1. *Se cumple la siguiente relación:*

$$f(x) - D(\lambda, s) = x^T s \geq 0 \ , \tag{1.4}$$

donde $x \in \mathcal{A}(P)$ *y* $(\lambda, s) \in \mathcal{A}(PD)$.

Demostración. Sea $x \in \mathcal{A}(P)$ y $(\lambda, s) \in \mathcal{A}(PD)$ se verifican las siguientes igualdades

$$f(x) - D(\lambda, s) = c^T x - b^T \lambda = x^T (c - A^T \lambda) = x^T s, \tag{1.5}$$

y, como $x \geq 0$ y $s \geq 0$ se tiene $x^T s \geq 0$ □

Al valor $x^T s$ se le conoce como **salto de dualidad**.

Dada la relación (1.4) y la igualdad (1.3) se pude concluir que los puntos admisibles con salto de dualidad cero son solución de sus respectivos problemas.

Corolario 1.2.1. *Cuando se cumple* $\bar{x}^T \bar{s} = 0$ *para* $\bar{x} \in \mathcal{A}(P)$ *y* $(\bar{\lambda}, \bar{s}) \in \mathcal{A}(PD)$*, entonces se tiene que* $\bar{x}$ *es solución para el problema* (P) *y* $(\bar{\lambda}, \bar{s})$ *es solución del problema* (PD).

Demostración. Trivial. □

Además, el siguiente resultado da una cota superior de la distancia de la función objetivo en cualquier punto admisible al valor óptimo.

Proposición 1.2.2. *Si* $x \in \mathcal{A}(P)$ *y* $(\lambda, s) \in \mathcal{A}(PD)$*, entonces se tiene*

$$\gamma \leq f(x) \leq \gamma + x^T s.$$

para $\gamma = \min\{f(x) : \ x \in \mathcal{A}(P)\}$.

Demostración. Por definición de mínimo se cumple la primera desigualdad. Para comprobar la segunda desigualdad se toma $\bar{x} \in \mathcal{A}(P)$ tal que $\gamma = f(\bar{x})$. Por la proposición 1.2.1 se sabe que $0 \leq \gamma - D(\lambda, s)$, por lo tanto se cumple:

$$f(x) \leq \gamma + f(x) - D(\lambda, s) = \gamma + x^T s. \tag{1.6}$$

Con lo que queda demostrada la proposición. □

Usando las condiciones de optimalidad suficientes de primer orden (ver teorema A.2.6) se pueden resumir en cuatro las condiciones para que un vector $\bar{x}$ sea solución del problema (P): las restricciones de los problemas (P) y su dual (PD), el salto de dualidad igual a 0 y las condiciones de positividad sobre algunas variables duales:

$$A^T\bar{\lambda} + \bar{s} = c, \tag{1.7a}$$
$$A\bar{x} = b, \tag{1.7b}$$
$$\bar{x}_i\bar{s}_i = 0,\ i = 1, 2, \ldots, n, \tag{1.7c}$$
$$(\bar{x}, \bar{s}) \geq 0, \tag{1.7d}$$

donde $\bar{\lambda}, \bar{s}$ son los llamados multiplicadores de Lagrange (Teorema A.2.5). Estas son las llamadas condiciones de Karush-Kuhn-Tucker (KKT) para el problema (P). Como se puede observar, las tres primeras están asociadas a un sistema de ecuaciones no lineales. Además, las últimas corresponden a $2n$ inecuaciones.

1.3. Dirección en los métodos primales-duales

Para resolver el sistema no lineal asociado a las condiciones de KKT mediante alguna variación del método de Newton se define la siguiente función $F : \mathbb{R}^n_+ \times \mathbb{R}^m \times \mathbb{R}^n_+ \longrightarrow \mathbb{R}^{2n+m}$:

$$F(x, \lambda, s) = \begin{bmatrix} A^T\lambda + s - c \\ Ax - b \\ XSe \end{bmatrix}, \tag{1.8}$$

donde $X, S \in \mathbb{R}^{n\times n}$ son matrices con todos los elementos iguales a cero excepto sus diagonales principales, que son iguales a x y s, respectivamente, y $e = (1, \ldots, 1)^T \in \mathbb{R}^n$.

El objetivo será encontrar $(\bar{x}, \bar{\lambda}, \bar{s})$ tales que $F(\bar{x}, \bar{\lambda}, \bar{s}) = 0$. De esta forma se tendrá $\bar{x} \in \mathcal{A}(P)$, $(\bar{\lambda}, \bar{s}) \in \mathcal{A}(PD)$ y el salto de dualidad $\bar{x}^T\bar{s} = 0$ y por tanto una solución de los problemas primal y dual (ver el corolario 1.2.1).

Para aplicar el método de Newton al sistema no lineal asociado a (1.7a)-(1.7b)-(1.7c), dado $(x^{(k)}, \lambda^{(k)}, s^{(k)})$, se calcula una dirección resolviendo el siguiente sistema de ecuaciones lineales:

$$J(x^{(k)}, \lambda^{(k)}, s^{(k)}) \begin{bmatrix} \Delta x \\ \Delta\lambda \\ \Delta s \end{bmatrix} = -F(x^{(k)}, \lambda^{(k)}, s^{(k)}), \tag{1.9}$$

donde $J(x^{(k)}, \lambda^{(k)}, s^{(k)})$ es la matriz jacobiana de F en $(x^{(k)}, \lambda^{(k)}, s^{(k)})$. El sistema queda de la siguiente forma:

$$\begin{bmatrix} 0 & A^T & I \\ A & 0 & 0 \\ S^{(k)} & 0 & X^{(k)} \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta\lambda \\ \Delta s \end{bmatrix} = \begin{bmatrix} c - s^{(k)} - A^T\lambda^{(k)} \\ b - Ax^{(k)} \\ -X^{(k)}S^{(k)}e \end{bmatrix}. \tag{1.10}$$

Para comprobar que este sistema tiene una única solución habrá que verificar que la matriz jacobiana es inversible.

Proposición 1.3.1. *La matriz Jacobiana recién definida:*

$$J(x^{(k)}, \lambda^{(k)}, s^{(k)}) = \begin{pmatrix} 0 & A^T & I \\ A & 0 & 0 \\ S^{(k)} & 0 & X^{(k)} \end{pmatrix},$$

con $x^{(k)}, s^{(k)} \in \mathbb{R}^n,\ (x^{(k)}, s^{(k)}) > 0$ *y* $A \in \mathbb{R}^{m\times n}$*, de rango máximo, es inversible.*

Demostración. Para simplificar la notación, en esta demostración se evita escribir los superíndices. Para probar que la matriz $J(x, \lambda, s)$ es inversible se resolverá el siguiente sistema:

$$\begin{pmatrix} 0 & A^T & I \\ A & 0 & 0 \\ S & 0 & X \end{pmatrix} z = 0. \tag{1.11}$$

Si la única solución posible es $z = 0$, la matriz $J(x, \lambda, s)$ será inversible. Si se divide el vector z en tres partes:

$$z = \begin{pmatrix} z_1 \\ z_2 \\ z_3 \end{pmatrix},$$

con $z_1,\ z_3 \in \mathbb{R}^n$ y $z_2 \in \mathbb{R}^m$. Multiplicando por bloques en (1.11), se llega al sistema:

$$\begin{cases} A^T z_2 + & Iz_3 & = 0 \\ Az_1 & & = 0 \\ Sz_1 + & Xz_3 & = 0 \end{cases}. \tag{1.12}$$

Como se sabe que S y X son matrices diagonales con todos sus elementos positivos, son inversibles. Despejando z_1 de la tercera ecuación y z_3 de la primera ecuación se llega a:

$$\begin{cases} z_3 = -A^T z_2 \\ z_1 = -S^{-1} X z_3 \end{cases}, \tag{1.13}$$

y sustituyendo en la segunda ecuación de (1.12)

$$AS^{-1}XA^T z_2 = 0\ . \tag{1.14}$$

Se puede definir la matriz diagonal (y por lo tanto simétrica)

$$D = diag(\sqrt{x_1/s_1}, \sqrt{x_2/s_2}, \ldots, \sqrt{x_n/s_n}).$$

Como $S^{-1} = diag(1/s_1, \ldots, 1/s_n)$, se tiene que $S^{-1}X = DD = D^T D$. Por lo que la igualdad (1.14) queda de la siguiente forma:

$$AD^T DA^T z_2 = 0.$$

Se cumple lo siguiente:

$$AD^T DA^T z_2 = 0 \Longrightarrow z_2^T AD^T DA^T z_2 = 0 \Longrightarrow (DA^T z_2)^T DA^T z_2 = 0 \Longrightarrow \|DA^T z_2\|_2^2 = 0.$$

Como se tiene $\|DA^T z_2\|_2^2 = 0$ entonces, también se cumple $DA^T z_2 = 0$. Usando que D es una matriz inversible, se tiene $A^T z_2 = 0$. De ahí, como A^T es de rango máximo por columnas, se tiene que la única solución del sistema es $z_2 = 0$.

Como $z_2 = 0$, por las igualdades (1.13), se llega a $z_1 = 0$ y $z_3 = 0$. Por lo tanto $z = 0$ es la única solución del sistema (1.11) y entonces queda probado que la matriz jacobiana asociada a puntos interiores es inversible. □

Para puntos interiores $(x^{(k)}, s^{(k)})$, como la matriz $J(x^{(k)}, \lambda^{(k)}, s^{(k)})$ es inversible, el sistema (1.9) tiene solución única. En la práctica, los métodos primales-duales en lugar de usar la condición (1.7c) usan

$$x_i s_i = \tau > 0 \text{ para } i = 1, \ldots, \ n, \ \text{con } \tau \longrightarrow 0.$$

Estas ecuaciones formarán parte del camino central, como se verá en la siguiente sección.

1.4. El camino central

Para hablar de métodos de puntos interiores lo primero que habrá que hacer será definir el conjunto de puntos admisibles, $\mathcal{F}$, y estrictamente admisibles, $\mathcal{F}^0$,

$$\mathcal{F} = \{(x, \lambda, s) |\ Ax = b,\ A^T\lambda + s = c,\ (x, s) \geq 0\}, \tag{1.15a}$$
$$\mathcal{F}^0 = \{(x, \lambda, s) |\ Ax = b,\ A^T\lambda + s = c,\ (x, s) > 0\}. \tag{1.15b}$$

A partir de los conjuntos anteriores se define:

Definición 1.4.1. ***(Camino central)*** *Dado $\tau > 0$, se llama* ***camino central,*** *$\mathcal{C}$, al arco de puntos $(x_\tau, \lambda_\tau, s_\tau) \in \mathcal{F}^0$ que cumple $X_\tau S_\tau e = \tau e$.*

El camino central resultará útil en la búsqueda de soluciones de los problemas (P) y (PD). Para ver que el camino está bien definido, basta ver que si $\mathcal{F}^0 \neq \emptyset$, entonces existe una sola tripleta $(x_\tau, \lambda_\tau, s_\tau) \in \mathcal{C}$ para cada $\tau > 0$.

Se redefine el sistema no lineal asociado a las condiciones de KKT cambiando (1.7c) por la condición $x_i s_i = \tau$

$$A^T\lambda_\tau + s_\tau = c\ , \tag{1.16a}$$
$$Ax_\tau = b\ , \tag{1.16b}$$
$$x_{\tau i} s_{\tau i} = \tau\ ,\ \ i = 1, 2, \ldots, n\ , \tag{1.16c}$$
$$(x_\tau, s_\tau) > 0\ . \tag{1.16d}$$

Se tiene el siguiente teorema:

Teorema 1.4.1. *Si $\mathcal{F}^0 \neq \emptyset$, entonces para cada $\tau > 0$ existe una única tripleta $(x_\tau, \lambda_\tau, s_\tau)$ verificando las condiciones (1.16)*

Antes de probar el teorema, se va a formular un problema de optimización asociado al camino central y se demostrarán tres resultados relevantes. Sea

$$\mathcal{H}^0 = \{(x, s) |\ (x, \lambda, s) \in \mathcal{F}^0 \text{ para algún } \lambda \in \mathbb{R}^m\}\ ,$$

y $f_\tau : \mathbb{R}^n \times \mathbb{R}^n \longrightarrow \mathbb{R}$ una función tal que:

$$f_\tau(x,s) = \frac{x^T s}{\tau} - \sum_{i=1}^{n} \log(x_i s_i) \ . \tag{1.17}$$

Se define el siguiente problema de optimización.

$$(P_\tau) \left\{ \begin{array}{cc} \text{mín} & f_\tau(x,s) \\ \text{sujeto a} & (x,s) \in \mathcal{H}^0 \end{array} \right. . \tag{1.18}$$

Por último, antes de demostrar el Teorema 1.4.1, se enuncian y prueban tres resultados.

Lema 1.4.1. *Sea $\mathcal{F}^0 \neq \emptyset$. Entonces para cada $K \geq 0$, el conjunto*

$$\{(x,s) \mid (x,\lambda,s) \in \mathcal{F} \text{ para algún } \lambda \text{ y } x^T s \leq K\},$$

está acotado.

Demostración. Sea $(\bar{x}, \bar{\lambda}, \bar{s})$ un punto cualquiera de $\mathcal{F}^0$ y (x, λ, s) un punto en $\mathcal{F}$ que cumple $x^T s \leq K$, entonces como $A\bar{x} = b, Ax = b$ se cumple $A(\bar{x} - x) = 0$. De la misma forma $A^T(\bar{\lambda} - \lambda) + (\bar{s} - s) = 0$. Entonces se llega a la siguiente igualdad

$$(\bar{x} - x)^T(\bar{s} - s) = -(\bar{x} - x)^T A^T(\bar{\lambda} - \lambda) = 0.$$

Como se tiene que $x^T s \leq K$, sustituyendo en la igualdad, se llega a la siguiente desigualdad

$$\bar{x}^T s + \bar{s}^T x \leq K + \bar{x}^T \bar{s}.$$

Como $(\bar{x}, \bar{s}) > 0$, escogiendo componente más pequeño entre ambos vectores

$$\xi = \text{mín}\{\text{mín}\{x_i, s_i\} \mid i = 1, \ldots, n\}$$

se puede decir

$$\xi e^T(x + s) \leq \bar{x}^T s + \bar{s}^T x \leq K + \bar{x}^T \bar{s}.$$

Esta última desigualdad implica

$$0 \leq x_i \leq \tfrac{1}{\xi}(K + \bar{x}^T \bar{s}), \quad 0 \leq s_i \leq \tfrac{1}{\xi}(K + \bar{x}^T \bar{s}), \quad i = 1, 2, \ldots, n.$$

Como $(\bar{x}, \bar{s})$ es fijo, (x, s) está acotado. Probando así el resultado. □

Lema 1.4.2. *La función $g : (0, +\infty) \longrightarrow \mathbb{R}$ tal que $g(t) = t - \log t - 1$ verifica:*

- *g es estrictamente convexa;*
- *$g(t) \geq 0$, con igualdad si y solo si $t = 1$;*
- *$g(t) \uparrow +\infty$ si $t \longrightarrow 0$ o $t \longrightarrow +\infty$.*

Demostración. Para demostrar el lema:

1. Las derivadas primera y segunda de g son: $g'(t) = 1-1/t$, $g''(t) = 1/t^2$, luego $g''(t) > 0$ para $t \in (0, +\infty)$ por tanto estrictamente convexa (Teorema A.2.1).

2. Se tiene que $g'(t) = 0 \iff 1 = 1/t \iff t = 1$. Por lo tanto $t = 1$ es mínimo global de $g(t)$
(Apéndice **??**). Y por consecuente $g(t) \geq 0$ para cada $t \in (0, +\infty)$. De forma reciproca, $g(t) = 0$ si y solo si $t - \log t = 1 \iff e^t/t = e \iff t = 1$.

3. Basta calcular los dos límites:

   - $\lim\limits_{t \longrightarrow 0} g(t) = \lim\limits_{t \longrightarrow 0} t - \log t - 1 = -\lim\limits_{t \longrightarrow 0} \log t + 1 = +\infty$
   - $\lim\limits_{t \longrightarrow +\infty} g(t) = \lim\limits_{t \longrightarrow \infty} \log\left(\frac{e^{t-1}}{t}\right) = \log\left(\lim\limits_{t \longrightarrow \infty} \frac{e^{t-1}}{t}\right) = \log\left(\lim\limits_{t \longrightarrow \infty} e^{t-1}\right) = +\infty$. donde en la penúltima igualdad se usa la regla de L'Hopital

□

Lema 1.4.3. *Los conjuntos de nivel* $\{(x, s) \in \mathcal{A}(P_\tau) : f_\tau(x, s) \leq \psi\}$*, donde* ψ *es una constante dada, están acotados.*

Demostración. Para empezar, la función f_τ se puede reescribir de la siguiente forma:

$$\begin{aligned} f_\tau(x, s) &= \frac{1}{\tau} x^T s - \sum_{i=1}^{n} \log(x_i s_i) \\ &= \sum_{i=1}^{n} \left(\frac{x_i s_i}{\tau} - \log x_i s_i + \log \tau - \log \tau + 1 - 1 \right) \\ &= \sum_{i=1}^{n} \left(\frac{x_i s_i}{\tau} - \log \frac{x_i s_i}{\tau} - 1 \right) + n - n \log \tau \\ &= \sum_{i=1}^{n} g\left(\frac{x_i s_i}{\tau} \right) + n - n \log \tau, \end{aligned} \tag{1.19}$$

donde $g(t) = t - \log t - 1$.

Si $f_\tau(x, s) \leq \psi$, se tiene que :

$$\sum_{i=1}^{n} g\left(\frac{x_i s_i}{\tau} \right) + n - n \log \tau \leq \psi \iff \sum_{i=1}^{n} g\left(\frac{x_i s_i}{\tau} \right) \leq \bar{\psi},$$

donde $\bar{\psi} = \psi - n + n \log \tau$.

Además, como se ha visto en el Lema 1.4.2, como $g(t) \geq 0$,

$$g\left(\frac{x_i s_i}{\tau} \right) \leq \bar{\psi} - \sum_{j \neq i} g\left(\frac{x_j s_j}{\tau} \right) \leq \bar{\psi}, \quad i = 1, \ldots, n.$$

Por el tercer apartado del Lema 1.4.2, como $g(x_i s_i) \leq \bar{\psi}$, se tiene que debe de existir una constante $M > 0$ tal que

$$\frac{1}{M} \leq x_i s_i \leq M \quad i = 1, \ldots, n.$$

Entonces, sumando para cada i, se tiene lo siguiente:

$$\sum_{i=1}^{n} x_i s_i = x^T s \leq nM.$$

Por lo tanto,

$$\{(x,s) \in \mathcal{A}(P_\tau) : f_\tau(x,s) \leq \psi\} \subset \{(x,s) |\ (x,\lambda,s) \in \mathcal{F}^0 \text{ para algún } \lambda, \text{ y } x^T s \leq nM\}$$

y aplicando el Lema 1.4.1 se llega a que el conjunto está acotado. □

Ahora se procede a la demostración del Teorema 1.4.1.

Demostración. Para que el conjunto de restricciones no dependa de λ, se escoge una matriz Z tal que sus columnas sean vectores del núcleo de A, $N(A)$. Multiplicando por Z^T a la izquierda en la restricción $A^T\lambda + s = c$ se llega a $Z^T s = Z^T c$.

Sea entonces el problema de optimización, equivalente a (P_τ), para $\tau > 0$:

$$(P'_\tau) \quad \begin{cases} \text{mín} & f_\tau(x,s) \\ \text{sujeto a} & (x,s) > 0 \\ & Ax = b \\ & Z^T s = Z^T c \end{cases} . \tag{1.20}$$

Para comprobar que la solución del problema de optimización (P'_τ) es el punto buscado que cumple (1.16), basta ver que las condiciones (1.16) son las condiciones de optimalidad del problema. Las condiciones (1.16a),(1.16b) y (1.16d) se deducen de $\mathcal{A}(P'_\tau)$. Para ver que se cumple (1.16c) se aplica el Teorema A.2.5 y se llega a las siguientes dos igualdades:

$$\begin{cases} s/\tau - X^{-1}e + A^T\tilde{\lambda} = 0 \\ x/\tau - S^{-1}e + Z\tilde{\mu} = 0 \end{cases} .$$

Por lo tanto se cumple

$$\left(\tfrac{s}{\tau} - X^{-1}e\right)^T \left(\tfrac{x}{\tau} - S^{-1}e\right) = \tilde{\lambda}^T A Z \tilde{\mu}.$$

Como $AZ = 0$, se tiene $\tilde{\lambda}^T AZ\tilde{\mu} = 0$. Además, como X y S son diagonales, el producto es conmutativo. Por tanto, utilizando además las igualdades $s = Se$ y $x = Xe$, se tiene:

$$\left(\tfrac{1}{\tau}Se - X^{-1}e\right)^T \left(S^{-1/2}X^{1/2}\right)\left(S^{1/2}X^{-1/2}\right)\left(\tfrac{1}{\tau}Xe - S^{-1}e\right) = 0 \iff$$

$$\|\tfrac{1}{\tau}(XS)^{1/2}e - (XS)^{-1/2}e\|^2 = 0 \iff$$

$$\tfrac{1}{\tau}(XS)^{1/2}e - (XS)^{-1/2}e = 0.$$

La igualdad se cumplirá si y solo si $XSe = \tau e$, o lo que es lo mismo, $x_i s_i = \tau$ para todo $i \in \{1, \ldots, n\}$.

Una vez verificado que (1.16) son las condiciones de optimalidad del problema (P'_τ), se demuestra la existencia y unicidad de solución.

Para ver si existe mínimo, sea $\left(x^{(k)}, s^{(k)}\right)_{k\in\mathbb{N}} \subset \mathcal{A}(P'_\tau)$ una sucesión minimizante, tal que

$$f_\tau(x^{(k)}, s^{(k)}) \longrightarrow \inf_{\mathcal{A}(P'_\tau)} f_\tau, \text{ cuando } k \longrightarrow \infty.$$

Para ver que la sucesión está acotada, se escoge $c = f_\tau(x^{(0)}, s^{(0)})$ una constante finita. Se considera $\left(x^{(k)}, s^{(k)}\right)_{k\in\mathbb{N}} \subset \{(x, s) \in \mathcal{A}(P'_\tau) \mid f_\tau(x, s) \leq c\}$ y aplicando el Lema 1.4.3 se llega a que está acotada. Por estar la sucesión acotada en $\mathbb{R}^n \times \mathbb{R}^n$, espacio de dimensión finita, existe una subsucesión $(x^{(k_m)}, s^{(k_m)})_{m\in\mathbb{N}} \subset (x^{(k)}, s^{(k)})_{k\in\mathbb{N}}$ convergente a $(\bar{x}, \bar{s})$.

Para comprobar que $(\bar{x}, \bar{s}) \in \mathcal{A}(P'_\tau)$, por reducción al absurdo se supone que $(\bar{x}, \bar{s})$ se encuentra en la frontera de $\mathcal{A}(P'_\tau)$, esto es que al menos una de las coordenadas de $\bar{x}$ o $\bar{s}$ sea 0. Las coordenadas que son 0 cumplen $\bar{x}_j \bar{s}_j = 0$ y se tiene el límite

$$\lim_{k_m \longrightarrow \infty} log(x_j^{(k_m)} s_j^{(k_m)}) = -\infty.$$

Sea J el conjunto de coordenadas en las que $\bar{x}$ o $\bar{s}$ toma 0. Se tiene que

$$\lim_{k_m \longrightarrow \infty} f_\tau(x^{(k_m)}, s^{(k_m)}) = K - \sum_{j\in J} \lim_{k_m \longrightarrow \infty} log(x_j^{(k_m)} s_j^{(k_m)}) = +\infty,$$

donde $K = \frac{1}{\tau}\bar{x}^T\bar{s} - \sum_{i\in\{1,\ldots,n\}\setminus J} log(\bar{x}_i\bar{s}_i)$. Se llega a un absurdo, ya que $f_\tau(x^{(k_m)}, s^{(k_m)}) \leq c$. Por lo tanto $(\bar{x}, \bar{s}) \in \mathcal{A}(P'_\tau)$ y usando la continuidad de f_τ se tiene

$$f_\tau(\bar{x}, \bar{s}) = \lim_{k_m \longrightarrow \infty} f(x^{(k_m)}, s^{(k_m)}) = \lim_{k \longrightarrow \infty} f(x^{(k)}, s^{(k)}) = \inf_{\mathcal{A}(P'_\tau)} f_\tau \ .$$

Con lo que el problema tendrá al menos una solución.

Para determinar la unicidad de la solución, bastará con ver que se cumpla el Teorema A.2.4. Para ello hay que comprobar si f_τ es estrictamente convexa. Teniendo en cuenta f_τ como suma de dos funciones distintas: $g_\tau(x, s) = \frac{1}{\tau}x^T s$ y $h(x, s) = -\sum_{i=1}^n \log(x_i s_i)$.

Se puede reescribir g_τ como una función lineal sobre el conjunto $\mathcal{A}(P'_\tau)$. Se escoge un $\tilde{x}$ cualquiera tal que $A\tilde{x} = b$, entonces se tiene que para cada $(x, s) \in \mathcal{A}(P'_\tau)$:

$$x^T s = x^T(c - A^T\lambda) = x^T c - b^T\lambda = x^T c - (\tilde{x}A)^T\lambda = x^T c - \tilde{x}^T(c - s) = x^T c + \tilde{x}^T s - \tilde{x}^T c,$$

que es lineal respecto de x y respecto de s. Para ver que h es convexa se estudia su matriz hessiana. Teniendo en cuenta

$$\nabla_x h(x, s) = (-1/x_1, \ldots, -1/x_n)^T,$$
$$\nabla_s h(x, s) = (-1/s_1, \ldots, -1/s_n)^T,$$

y

$$\nabla^2_{xx} h(x, s) = diag(1/x_1^2, \ldots, 1/x_n^2), \quad \nabla^2_{xs} h(x, s) = 0$$
$$\nabla^2_{ss} h(x, s) = diag(1/s_1^2, \ldots, 1/s_n^2), \quad \nabla^2_{sx} h(x, s) = 0,$$

es evidente que la hessiana es definida positiva en $\mathcal{A}(P'_\tau)$. Entonces como $\mathcal{A}(P'_\tau)$ es un conjunto convexo, se pueden aplicar los teoremas A.2.1 y A.2.4. Por lo tanto existe solución para (P'_τ) y es única.

□

En la práctica, algunos métodos usan las condiciones (1.16) con $\tau = \sigma\mu$, donde μ será la medida de dualidad (sección 1.4.2) y $\sigma \in [0, 1]$ un factor de reducción. La ecuación para la dirección de Newton para puntos admisibles queda de la siguiente forma:

$$\begin{bmatrix} 0 & A^T & I \\ A & 0 & 0 \\ S^{(k)} & 0 & X^{(k)} \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta\lambda \\ \Delta s \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ -X^{(k)}S^{(k)}e + \sigma\mu e \end{bmatrix} \tag{1.21}$$

1.4.1. Relación con los métodos de penalización

Reemplazando las restricciones $x_i \geq 0$ del problema (P) por un término de barrera logarítmica, $\tau \log(x_i)$ (con $\tau > 0$), añadido a la función objetivo de (P) se obtiene el siguiente problema de optimización:

$$(PP_\tau) \begin{cases} \min\limits_{x>0} & \tilde{f}_\tau(x) = c^T x - \tau \sum_{i=1}^n \log\ x_i \\ \text{sujeto a} & Ax = b \end{cases} .$$

El parámetro de penalización τ sirve para controlar la relación entre el término asociado a las restriciones de desigualdad y la función objetivo del problema (P).

A continuación se relaciona el problema de penalización (P_τ) con las ecuaciones del camino central:

Proposición 1.4.1. *Dado el problema (PP_τ), se cumple que $\bar{x}_\tau$ es solución de (P_τ) si y solo si existen $\bar{s}_\tau \in \mathbb{R}^n, \bar{s}_\tau > 0,\ \bar{\lambda}_\tau \in \mathbb{R}^m$ que verifican (1.16).*

Demostración. Para demostrar la proposición:

$\Longrightarrow$ Si $\bar{x}_\tau$ es solución de (PP_τ), entonces es punto de Kuhn-Tucker (véase el teorema A.2.5). Así que el resultado se tiene usando

$$\nabla \tilde{f}_\tau(x) = c - \tau \begin{pmatrix} 1/x_1 \\ \vdots \\ 1/x_n, \end{pmatrix}$$

y tomando $\bar{s}_{\tau i} = \tau/\bar{x}_{\tau i}$.

La condición (1.16b) y $\bar{x}_\tau > 0$ se cumplen por las restricciones del problema (PP_τ). Como $\bar{x}_\tau > 0,\ \tau > 0$ y $\bar{s}_{\tau i} = \tau/\bar{x}_{\tau i}$, entonces $\bar{s}_{\tau i} > 0$. Por lo que se cumplen todas las condiciones (1.16).

$\Longleftarrow$ Para obtener el resultado basta aplicar las condiciones suficientes de primer orden (ver teorema A.2.6). Para ver que la función $\tilde{f}_\tau$ es convexa se puede aplicar el teorema A.2.1, teniendo en cuenta que la hessiana de $\tilde{f}_\tau$ es:

$$\nabla^2 \tilde{f}_\tau(x) = diag(\tau/x_1^2, \ldots, \tau/x_n^2).$$

Como $x > 0$ y $\tau > 0$, entonces la matriz hessiana es una matriz diagonal de elementos positivos, por tanto es definida positiva.

□

Los primeros métodos para resolver numéricamente los problemas de programación no lineal (problemas de optimización donde ambos, la función objetivo y las restricciones pueden ser no lineales) fueron los métodos de penalización en los que se trataba de resolver una sucesión de problemas sin restricciones (en los que la función objetivo combina la del problema de programación no lineal con unos términos asociados a las restricciones de ese problema). Por ejemplo, la función de barrera logarítmica fue introducida a mediados del siglo XX. Estos métodos de penalización sufrían de dificultades a la hora de su utilización en la práctica, originadas por el mal condicionamiento de las matrices hessianas cuando el parámetro de penalización no es pequeño.

La diferencia con el planteamiento de métodos de puntos interiores es que estos no hacen la minimización del problema de penalización en cada iteración para cada valor de τ, sino que calculan una dirección en relación con ese problema.

1.4.2. Entornos del camino central

Un vez definido el camino central $\mathcal{C}$ se pueden considerar entornos del mismo con el fin de que la búsqueda esté menos restringida. Sea $\mu_k = (x^{(k)})^T s^{(k)}/n$ la **medida de dualidad** actual asociada a el punto interior $(x^{(k)}, \lambda^{(k)}, s^{(k)})$. Destacan los dos siguientes entornos de $\mathcal{C}$:

$$\mathcal{N}_2(\theta) = \{(x, \lambda, s) \in \mathcal{F}^0 \mid \|XSe - \mu e\|_2 \leq \theta\mu\}, \tag{1.22}$$

con $\theta \in [0, 1)$, y

$$\mathcal{N}_{-\infty}(\gamma) = \{(x, \lambda, s) \in \mathcal{F}^0 \mid x_i s_i \geq \gamma\mu \text{ para } i = 1, 2 \ldots, n\}, \tag{1.23}$$

con $\gamma \in (0, 1]$ (usualmente se toman los valores $\theta = 0.5$ y $\gamma = 10^{-3}$). En el siguiente resultado se demuestra que los conjuntos del tipo $\mathcal{N}_2(\theta)$ son más restrictivos que los $\mathcal{N}_{-\infty}(\gamma)$. De ahí que los métodos de puntos interiores que usan entornos N_2 se llamen de paso corto ($Short - Step\ Path - Following$ o SPF[7]) y los métodos que usan entornos $N_{-\infty}$ se llamen de paso largo ($Long - Step\ Path - Following$ o LPF).

Proposición 1.4.2. *Se tiene que $\mathcal{N}_2(\theta) \subseteq \mathcal{N}_{-\infty}(\gamma)$ para $0 < \gamma \leq (1 - \theta)$.*

Demostración. Sea $(x, \lambda, s) \in \mathcal{N}_2(\theta)$ se cumple

$$|x_i s_i - \mu| \leq \|XSe - \mu e\|_2 \leq \theta\mu,$$

y, aplicando la desigualdad triangular y que $\mu > 0$ y $x_i s_i > 0$, se llega a la desigualdad deseada:

$$|\mu - x_i s_i| \leq \theta\mu \Longrightarrow |\mu| - |x_i s_i| \leq \theta\mu \Longrightarrow (1 - \theta)\mu \leq x_i s_i.$$

Luego $x_i s_i \geq \gamma\mu$ para cualquier γ tal que $\gamma \leq (1 - \theta)$. Por lo tanto queda probado el enunciado. □

La naturaleza menos restrictiva de los entornos $\mathcal{N}_{-\infty}$ da pie a que los algoritmos LPF sean más eficaces en la práctica que los SPF.

1.5. Algoritmo LSPF y su convergencia global

El siguiente procedimiento es un algoritmo LPF, que toma siempre el mayor desplazamiento posible dentro de un entorno $\mathcal{N}_{-\infty}(\gamma)$.

A continuación, se va a usar la siguiente notación

$$(x^{(k)}(\alpha), \lambda^{(k)}(\alpha), s^{(k)}(\alpha)) = (x^{(k)}, \lambda^{(k)}, s^{(k)}) + \alpha(\Delta x^{(k)}, \Delta\lambda^{(k)}, \Delta s^{(k)}), \tag{1.24}$$

donde $(x^{(k)}(\alpha), \lambda^{(k)}(\alpha), s^{(k)}(\alpha))$ es el último aproximante calculado y $(\Delta x^{(k)}, \Delta\lambda^{(k)}, \Delta s^{(k)})$ la última dirección de avance calculada.

Algoritmo 1.5.1. *(Long-Step Path-Following).*
Dado $\gamma,\ \sigma_{min},\ \sigma_{max}$ *con* $\gamma \in (0,1),\ 0 < \sigma_{min} < \sigma_{max} < 1$ *y* $(x^{(0)}, \lambda^{(0)}, s^{(0)}) \in \mathcal{N}_{-\infty}(\gamma)$
for k=1, 2,...

- *Escoger* $\sigma_k \in [\sigma_{min}, \sigma_{max}]$

- *Calcular la dirección* $(\Delta x^{(k)}, \Delta\lambda^{(k)}, \Delta s^{(k)})$ *resolviendo el sistema (1.21) con* $\mu_k = (x^{(k)})^T s^{(k)}/n$

- *Escoger* α_k *como el mayor desplazamiento en [0,1] tal que*

$$(x^{(k)}(\alpha_k), \lambda^{(k)}(\alpha_k), s^{(k)}(\alpha_k) \in \mathcal{N}_{-\infty}(\gamma). \tag{1.25}$$

- *Tomar* $(x^{(k+1)}, \lambda^{(k+1)}, s^{(k+1)}) = (x^{(k)}(\alpha_k), \lambda^{(k)}(\alpha_k), s^{(k)}(\alpha_k))$.

end(for)

El objetivo del Algoritmo 1.5.1 es construir una sucesión $((x^{(k)}, \lambda^{(k)}, s^{(k)}))_{k\in\mathbb{N}}$ con la medida de dualidad μ_k tendiendo a 0 y que sea convergente a $(\bar{x}, \bar{\lambda}, \bar{s})$, solución de (1.7). Para comprobar el ritmo de decrecimiento de $(\mu_k)_{k\in\mathbb{N}}$ se parte del siguiente resultado cuya demostración se encuentra en [9].

Lema 1.5.1. *Sean* $u, v \in \mathbb{R}^n$ *con* $u^T v \geq 0$. *Se tiene*

$$\|UVe\|_2 \leq 2^{-3/2}\|u + v\|_2^2\ . \tag{1.26}$$

Se tiene además que se cumple:

Lema 1.5.2. *Sea* $(\Delta x^{(k)}, \Delta\lambda^{(k)}, \Delta s^{(k)})$ *la solución del sistema (1.21). Se tiene*

$$\Delta x^T \Delta s = 0 \tag{1.27}$$

y

$$\mu(\alpha) = (1 - \alpha + \alpha\sigma)\mu\ . \tag{1.28}$$

Demostración. Basta multiplicar por Δx^T a la izquierda la primera igualdad del sistema (1.21). Se obtiene:

$$\Delta x^T(A^T\Delta\lambda + \Delta s) = 0 \Longleftrightarrow (A\Delta x)^T\Delta\lambda + \Delta x^T\Delta s = 0 \ .$$

Como se conoce que $A\Delta x = 0$, se deduce $\Delta x^T\Delta s = 0$. Esta igualdad es útil para probar (1.28). Se tiene que

$$\mu(\alpha) = \frac{x(\alpha)^T s(\alpha)}{n} = \frac{1}{n}\left(x^T s + \alpha(\Delta x^T s + x^T\Delta s)\right) \ . \tag{1.29}$$

De la tercera igualdad del sistema (1.21):

$$s_i^{(k)}\Delta x_i^{(k)} + x_i^{(k)}\Delta s_i^{(k)} = -x_i^{(k)}s_i^{(k)} + \sigma_k\mu \ \text{ para } i = 1\ldots, n \ ,$$

sumando las n ecuaciones se llega a que se cumple la siguiente

$$(s^{(k)})^T\Delta x^{(k)} + (x^{(k)})^T\Delta s^{(k)} = -(x^{(k)})^T s^{(k)} + n\sigma_k\mu.$$

Sustituyendo en (1.29) y usando que $\mu_k = (x^{(k)})^T s^{(k)}/n$ se tiene

$$\mu(\alpha) = \frac{1}{n}\left((x^{(k)})^T s^{(k)} + \alpha(-(x^{(k)})^T s^{(k)} + n\sigma\mu)\right) \ = \mu\left(1 - \alpha + \alpha\sigma\right) \ .$$

□

A continuación, se enuncia un resultado sobre una relación entre las componentes primal y dual de la dirección solución de (1.21) que servirá para calcular el ritmo de decrecimiento de $(\mu_k)_{k\in\mathbb{N}}$:

Lema 1.5.3. *Si $(x, \lambda, s) \in \mathcal{N}_{-\infty}(\gamma)$, entonces se tiene*

$$\|\Delta X\Delta S e\|_2 \leq 2^{-3/2}(1 + 1/\gamma)n\mu \ . \tag{1.30}$$

Demostración. Sea $D = X^{1/2}S^{-1/2}$. Al multiplicar la tercera igualdad del sistema (1.21) por $(XS)^{-1/2}$ se obtiene la igualdad

$$D^{-1}\Delta x + D\Delta s = (XS)^{-1/2}\left(-XSe + \sigma\mu e\right) \ .$$

Aplicando el lema 1.5.1 con $u = D^{-1}\Delta x$ y $v = D\Delta s$, sabiendo (1.27), se tiene:

$$\begin{aligned}\|D^{-1}\Delta X D\Delta e\|_2 &\leq 2^{-3/2}\|D^{-1}\Delta x + D\Delta s\|_2^2 \\ &= 2^{-3/2}\|(XS)^{-1/2}\left(-XSe + \sigma_k\mu e\right)\|_2^2 \ .\end{aligned}$$

Calculando la última norma se llega a lo siguiente:

$$\begin{aligned}\|D^{-1}\Delta X D\Delta e\|_2 &\leq 2^{-3/2}\left[x^T s - 2n\sigma\mu + (\sigma\mu)^2\sum_{i=1}^{n}\frac{1}{x_i s_i}\right] \\ &\leq 2^{-3/2}\left[n\mu - 2n\sigma\mu + (\sigma\mu)^2\frac{n}{\mu\gamma}\right] \\ &\leq 2^{-3/2}\left[1 - 2\sigma + \frac{\sigma^2}{\gamma}\right]n\mu \\ &\leq 2^{-3/2}\left(1 + 1/\gamma\right)n\mu \ ,\end{aligned}$$

donde se ha usado $x_i s_i \geq \gamma\mu$ y $\sigma_k \in [0, 1]$. Con lo que queda demostrado el lema. □

Para comprobar el ritmo de decrecimiento de μ_k se demuestra el siguiente teorema.

Teorema 1.5.1. *Dado el Algoritmo 1.5.1, existe una constante*

$$\delta = 2^{3/2}\gamma\frac{1-\gamma}{1+\gamma}\,\text{mín}\{\sigma_{min}(1-\sigma_{\text{mín}}), \sigma_{max}(1-\sigma_{\text{máx}})\}\ , \tag{1.33}$$

independiente de n, tal que la medida de dualidad para dos iteraciones seguidas cumple

$$\mu_{k+1} \leq \left(1 - \frac{\delta}{n}\right)\mu_k\ , \tag{1.34}$$

para todo $k \geq 0$.

Demostración. Primero, dado $(x^{(k)}, \lambda^{(k)}, s^{(k)}) \in \mathcal{N}_{-\infty}(\gamma)$ va a comprobar cual es el máximo α para el que

$$(x^{(k)}(\alpha), \lambda^{(k)}(\alpha), s^{(k)}(\alpha)) \in \mathcal{N}_{-\infty}(\gamma)\ .$$

Se tiene que para cada $i = 1, \ldots, n$ se cumple, aplicando el lema 1.5.3:

$$|\Delta x_i^{(k)}\Delta s_i^{(k)}| \leq \|\Delta X^{(k)}\Delta S^{(k)}e\|_2 \leq 2^{-3/2}(1+1/\gamma)n\mu_k\ . \tag{1.35}$$

Se tiene lo siguiente:

$$\begin{aligned} x_i^{(k)}(\alpha)s_i^{(k)}(\alpha) &= (x_i^{(k)} + \alpha\Delta x_i^{(k)})(s_i^{(k)} + \alpha\Delta s_i^{(k)}) \\ &= x_i^{(k)}s_i^{(k)} + \alpha\left(x_i^{(k)}\Delta s_i^{(k)} + \Delta x_i^{(k)}s_i^{(k)}\right) + \alpha^2\Delta x_i^{(k)}\Delta s_i^{(k)}\ . \end{aligned}$$

Aplicando (1.21), (1.35) y $x_i^{(k)}s_i^{(k)} \geq \gamma\mu_k$ se llega a

$$\begin{aligned} x_i^{(k)}(\alpha)s_i^{(k)}(\alpha) &\geq x_i^{(k)}s_i^{(k)} + \alpha(-x_i^{(k)}s_i^{(k)} + \sigma_k\mu_k) - \alpha^2|\Delta x_i^{(k)}\Delta s_i^{(k)}| \\ &\geq (1-\alpha)\gamma\mu_k + \alpha\sigma_k\mu_k - \alpha^2\ 2^{-3/2}(1+1/\gamma)n\mu_k\ . \end{aligned}$$

Además, por el lema 1.5.2 se tiene:

$$\mu_k(\alpha) = (1 - \alpha + \alpha\sigma_k)\mu_k\ .$$

Se puede asegurar que la condición $x_i^{(k)}(\alpha)s_i^{(k)}(\alpha) \geq \gamma\mu_k(\alpha)$ se cumple si se satisface

$$(1-\alpha)\gamma\mu_k + \alpha\sigma_k\mu_k - \alpha^2\ 2^{-3/2}(1+1/\gamma)n\mu_k \geq \gamma(1-\alpha+\alpha\sigma_k)\mu_k\ , \tag{1.38}$$

o equivalentemente:

$$\begin{aligned} \mu_k\gamma + \mu_k\left(\alpha(-\gamma+\sigma_k) - \alpha^2 2^{-3/2}n\left(1+\frac{1}{\gamma}\right)\right) \geq \gamma\mu_k + \gamma\mu_k\alpha(\sigma_k - 1) &\iff \\ \iff \alpha(-\gamma+\sigma_k) - \alpha\gamma(\sigma_k - 1) \geq \alpha^2 2^{-3/2}n\left(1+\frac{1}{\gamma}\right) &\iff \\ 2^{3/2}\frac{\sigma_k}{n}\frac{1-\gamma}{1+1/\gamma} \geq \alpha & \end{aligned}$$

Por lo tanto se tiene que

$$(x(\alpha), \lambda(\alpha), s(\alpha)) \in \mathcal{N}(\gamma), \quad \forall \alpha \in \left[0,\ 2^{3/2}\frac{\sigma_k}{n}\gamma\frac{1-\gamma}{1+\gamma}\right] \ .$$

Luego, en una iteración cualquiera k, tomando un paso α_k en el rango recién calculado, y usando (1.28), la reducción de μ_k cumple:

$$\begin{aligned}\mu_{k+1} = \mu_k(\alpha_k) &= \mu_k(1 + \alpha_k(\sigma_k - 1)) \\ &\leq \mu_k(1 + \frac{2^{3/2}}{n}\gamma\frac{1-\gamma}{1+\gamma}\sigma_k(\sigma_k - 1)) \ .\end{aligned}$$

Nótese que para $\sigma_k \in [\sigma_{min}, \sigma_{max}]$ la función $\sigma_k(1 - \sigma_k)$ es una función con mínimos en los extremos. Por lo tanto, como el valor σ_k escogido será el que más reduzca μ_k, se tomará

$$\delta = 2^{3/2}\gamma\frac{1-\gamma}{1+\gamma}\,\text{mín}\{\sigma_{min}(\sigma_{min} - 1), \sigma_{max}(\sigma_{max} - 1)\} \ , \tag{1.41}$$

con lo que se obtiene el resultado buscado

$$\mu_{k+1} \leq (1 - \frac{\delta}{n})\mu_k \ .$$

□

Como se puede observar, el ritmo de decrecimiento está relacionado con el tamaño del problema y con los valores $\gamma, \sigma_{min}, \sigma_{max}$ escogidos.

Dado el ritmo de decrecimiento del valor de la medida de dualidad, se puede probar que el método necesita un número de iteraciones de orden polinomial (en relación al tamaño del problema).

Teorema 1.5.2. *Dado $\epsilon \in (0, 1)$, existe un índice K con $K = O(n \log 1/\epsilon)$ tal que*

$$\mu_k \leq \epsilon\mu_0, \textit{para todo } k \geq K.$$

Demostración. A partir de la desigualdad (1.34) se llega a

$$\mu_k \leq \left(1 - \frac{\delta}{n}\right)^{(k)} \mu_0 \ .$$

Aplicando el logaritmo se tiene

$$\log \mu_k \leq k \log\left(1 - \frac{\delta}{n}\right) + \log \mu_0 \Longleftrightarrow \log \frac{\mu_k}{\mu_0} \leq k \log\left(1 - \frac{\delta}{n}\right). \tag{1.42}$$

Se conoce que

$$\forall \beta > -1 \quad \log(1 + \beta) \leq \beta \ . \tag{1.43}$$

Para aplicarlo se comprueba si $-\delta/n$ es mayor que -1. Basta observar (1.33) y ver que se cumplen las siguientes desigualdades

$$\gamma\frac{1-\gamma}{1+\gamma} \leq \gamma(1-\gamma) \leq 1/4 \ ,$$
$$\text{mín}\{\sigma_{min}(1-\sigma_{min}), \sigma_{max}(1-\sigma_{max})\} \leq 1/4,$$

ya que $\gamma, \sigma_{min}, \sigma_{max} \in (0,1)$. Por lo tanto se tiene que $\delta \leq 2^{3/2}/16 < 1 \leq n$. Una vez aplicado (1.43) a la desigualdad (1.42) se obtiene

$$\log\left(\frac{\mu_k}{\mu_0}\right) \leq k\left(-\frac{\delta}{n}\right).$$

Nótese que si $k\left(-\frac{\delta}{n}\right) \leq \log\epsilon$ entonces se cumplirá la desigualdad buscada $\frac{\mu_k}{\mu_0} \leq \epsilon$,por lo tanto se tiene

$$k\left(-\frac{\delta}{n}\right) \leq \log(\epsilon) \Longleftrightarrow k \geq \frac{n}{\delta}\log\left(\frac{1}{\epsilon}\right).$$

Se llega a que, para que se cumpla $\mu_k \leq \epsilon\mu_0$, el valor k debe de ser mayor o igual que $\frac{\delta}{n}\log\frac{1}{\epsilon}$ por lo que se tiene que $K = O\left(n\log\left(\frac{1}{\epsilon}\right)\right)$.

□

Se llega a la conclusión de que este algoritmo tiene una complejidad de $O\left(n\log\left(\frac{1}{\epsilon}\right)\right)$. Existen algoritmos con complejidades mejores, como es el caso de los algoritmos del tipo predictor-corrector. Estos algoritmos usan entornos del tipo $\mathcal{N}_2$ y, dos tipos distintos de pasos, mejorando la complejidad a $O\left(\sqrt{n}\log\left(\frac{1}{\epsilon}\right)\right)$ [9].

1.6. Dirección afín y dirección de centralización

En los métodos de puntos interiores que sigan el camino central destacan dos tipos de dirección que se definen a continuación:

Definición 1.6.1. ***(Dirección afín)*** *Se llama dirección afín , $(\Delta x^{aff}, \Delta\lambda^{aff}, \Delta s^{aff})$, a la solución resultante de resolver el sistema (1.21) para $\sigma = 0$.*

Definición 1.6.2. ***(Dirección de centralización)*** *Se llama dirección de centralización a la dirección $(\Delta x^{cnt}, \Delta\lambda^{cnt}, \Delta s^{cnt})$ resultante de resolver el sistema (1.21) para $\sigma = 1$.*

Dadas las direcciones afín y de centralización, se tiene que cualquier otra dirección calculada se pude escribir como combinación lineal de estas.

Teorema 1.6.1. *Sea $(x, \lambda, s) \in \mathcal{F}^0$, $\sigma \in [0,1]$ y $(\Delta x^\sigma, \Delta\lambda^\sigma, \Delta s^\sigma)$ una dirección, solución de (1.21). Entonces esa dirección se puede expresar como combinación de las direcciones recién definidas:*

$$(\Delta x^\sigma, \Delta\lambda^\sigma, \Delta s^\sigma) = (1-\sigma)(\Delta x^{aff}, \Delta\lambda^{aff}, \Delta s^{aff}) + \sigma(\Delta x^{cnt}, \Delta\lambda^{cnt}, \Delta s^{cnt}) \ . \qquad (1.45)$$

Demostración. Las direcciones afín y de centralización verifican

$$\begin{bmatrix} 0 & A^T & I \\ A & 0 & 0 \\ S & 0 & X \end{bmatrix} \begin{bmatrix} \Delta x^{aff} \\ \Delta \lambda^{aff} \\ \Delta S^{aff} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ -XS \end{bmatrix}, \quad \begin{bmatrix} 0 & A^T & I \\ A & 0 & 0 \\ S & 0 & X \end{bmatrix} \begin{bmatrix} \Delta x^{cnt} \\ \Delta \lambda^{cnt} \\ \Delta s^{cnt} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ -XSe + \mu e \end{bmatrix}.$$

Multiplicando por $(1-\sigma)$ la primera igualdad, por σ la segunda y sumando se obtiene

$$\begin{bmatrix} 0 & A^T & I \\ A & 0 & 0 \\ S & 0 & X \end{bmatrix} \left((1-\sigma) \begin{bmatrix} \Delta x^{aff} \\ \Delta \lambda^{aff} \\ \Delta S^{aff} \end{bmatrix} + \sigma \begin{bmatrix} \Delta x^{cnt} \\ \Delta \lambda^{cnt} \\ \Delta s^{cnt} \end{bmatrix} \right) = \begin{bmatrix} 0 \\ 0 \\ -XSe + \sigma\mu e \end{bmatrix}.$$

Luego, por la unicidad de solución del sistema lineal (1.21) (ver proposición 1.3.1) se debe de cumplir $(\Delta x^{\sigma}, \Delta \lambda^{\sigma}, \Delta s^{\sigma}) = (1-\sigma)(\Delta x^{aff}, \Delta \lambda^{aff}, \Delta s^{aff}) + \sigma(\Delta x^{cnt}, \Delta \lambda^{cnt}, \Delta s^{cnt})$. □

1.7. Algoritmo MPC

Una vez definidas las direcciones afín y de centralización ya se puede presentar el algoritmo predictor-corrector. El siguiente algoritmo es una de las variaciones más comunes del algoritmo de Mehrotra [6].

Algoritmo 1.7.1. *(MPC).*
Sea $(x^{(0)}, \lambda^{(0)}, s^{(0)}) \in \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^n$ *un punto inicial con* $(x^{(0)}, s^{(0)}) > 0$.
for k=0, 1, 2,. . .

- *Tomar* $(x, \lambda, s) = (x^{(k)}, \lambda^{(k)}, s^{(k)})$ *y calcular* $(\Delta x^{aff}, \Delta \lambda^{aff}, \Delta s^{aff})$ *la solución del sistema (1.21) con* $\sigma = 0$.

- *Calcular los desplazamientos, primal y dual*

$$\alpha_{aff}^{pri} = \arg\max\{\alpha \in [0,1] \mid x^{(k)} + \alpha \Delta x^{aff} \geq 0\}, \tag{1.46a}$$
$$\alpha_{aff}^{dual} = \arg\max\{\alpha \in [0,1] \mid s^{(k)} + \alpha \Delta s^{aff} \geq 0\}, \tag{1.46b}$$

  y la medida de dualidad

$$\mu_{aff} = (x^{(k)} + \alpha_{aff}^{pri} \Delta x^{aff})^T (s^{(k)} + \alpha_{aff}^{dual} \Delta s^{aff})/n \,. \tag{1.47}$$

- *Tomar* $\sigma = (\mu_{aff}/\mu)^3$.

- *Calcular* $(\Delta x^{cc}, \Delta \lambda^{cc}, \Delta s^{cc})$ *la solución del sistema lineal:*

$$\begin{bmatrix} 0 & A^T & I \\ A & 0 & 0 \\ S & 0 & X \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta \lambda \\ \Delta s \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ -\Delta X^{aff} \Delta S^{aff} e + \sigma \mu e \end{bmatrix}. \tag{1.48}$$

- *Calcular la dirección*

$$(\Delta x^k, \Delta\lambda^k, \Delta s^k) = (\Delta x^{aff}, \Delta\lambda^{aff}, \Delta s^{aff}) + (\Delta x^{cc}, \Delta\lambda^{cc}, \Delta s^{cc}) \tag{1.49}$$

y los desplazamientos

$$\alpha_{max}^{pri} = \arg\max\{\alpha \geq 0 \mid x^{(k)} + \alpha\Delta x^{(k)} \geq 0\} \ , \tag{1.50a}$$
$$\alpha_{max}^{dual} = \arg\max\{\alpha \geq 0 \mid s^{(k)} + \alpha\Delta s^{(k)} \geq 0\} \ . \tag{1.50b}$$

- *Tomar* $\alpha_k^{pri} = \min(0.99\alpha_{max}^{pri}, 1)$ *y* $\alpha_k^{dual} = \min(0.99\alpha_{max}^{dual}, 1)$.

- *Tomar*

$$x^{k+1} = x^{(k)} + \alpha_k^{pri}\Delta x^{(k)} \ ,$$
$$(\lambda^{k+1}, s^{k+1}) = (\lambda^{(k)}, s^{(k)}) + \alpha_k^{dual}(\Delta\lambda^{(k)}, \Delta s^{(k)}).$$

end for.

Este algoritmo es similar al que se implementa en MATLAB, que está basado en LIPSOL [10].

1.7.1. Cálculo numérico de las direcciones

La parte más costosa del proceso es resolver los sistemas (1.21) y (1.48). Estos sistemas tienen una matriz de coeficientes con la misma estructura y de tamaño $(2n+m)\times(2n+m)$. Utilizar un método convencional que trabaje con la matriz en su totalidad puede ser muy costoso, sobre todo si los problemas ya son de por si de talla grande. Para reducir el coste se puede aprovechar la estructura del sistema. Como la única diferencia entre ambos sistemas es el vector de términos independientes, se va a poder plantear el mismo procedimiento de resolución. Sea el sistema

$$\left\{\begin{array}{rcl} A^T\Delta\lambda + \Delta s &=& 0 \\ A\Delta x &=& 0 \\ S\Delta x + X\Delta s &=& u \end{array}\right. ,$$

para $u \in \mathbb{R}^n$ fijo. Se despeja Δs de la tercera ecuación y se sustituye el resultado en la primera, obteniendo

$$\left\{\begin{array}{rcl} A^T\Delta\lambda - X^{-1}S\Delta x &=& -X^{-1}u \\ A\Delta x &=& 0 \end{array}\right.$$
$$\Delta s = X^{-1}\left(u - S\Delta x\right).$$

Tomando $D = S^{-1/2}X^{1/2}$ se tiene

$$\left\{\begin{array}{rcl} A^T\Delta\lambda - D^{-2}\Delta x &=& -X^{-1}u \\ A\Delta x &=& 0 \end{array}\right. .$$

Por último, se multiplica la primera ecuación por AD^2 a la izquierda y se suma con la segunda se llegando a

$$AD^2A^T\Delta\lambda - AD^2D^{-2}\Delta x + A\Delta x = -AD^2X^{-1}u \ ,$$

obteniendo así el siguiente sistema (con una matriz de coeficientes simétrica y definida positiva $m \times m$)

$$AD(AD)^T \Delta\lambda = -AS^{-1}u \ .$$

Para el resto de variables se considera:

$$\Delta s = -A^T \Delta\lambda \quad \text{y} \quad \Delta x = S^{-1}\left(u - X\Delta s\right).$$

De forma que se tiene que $(\Delta x, \Delta\lambda, \Delta s)$ es solución de

$$\left\{ \begin{array}{rcl} AD(AD)^T \Delta\lambda & = & -AD^2X^{-1}u \\ \Delta x & = & S^{-1}\left(u - X\Delta s\right) \\ \Delta s & = & -A^T\Delta\lambda. \end{array} \right. \quad . \tag{1.52}$$

De donde para calcular la solución habrá que resolver un sistema de ecuaciones lineales de tamaño $m \times m$ con una matriz simétrica y definida positiva. Esto ayuda mucho en la resolución numérica del problema. Para empezar el tamaño de la matriz pasa de ser $(2n + m) \times (2n + m)$ a $m \times m$. Además, la matriz $AD(AD)^T$ cumple los requisitos para la factorización de Cholesky, generando estabilidad en el proceso de resolución numérica frente a otros tipos de factorización como la LU. En el caso de problemas de talla grande puede ser aconsejable elegir el método gradiente conjugado usando como precondicionador una factorización parcial de Cholesky.

Capítulo 2

Resultados Numéricos

2.1. Software: *linprog* de MATLAB

Para resolver los problemas considerados en las siguientes secciones se ha utilizado la *Optimization Toolbox* de MATLAB R2023a en un ordenador "Asus-UX481FL" con el sistema operativo Windows 11 Home y 16GB de RAM. Todos los resultados se han obtenido con la función *linprog*, usada con las tolerancias por defecto en todos los casos. Se han empleado los tres métodos disponibles en *linprog*: "Dual-Simplex"(S), "Interior-Point"(I) e "Interior-Point-Legacy"(L).

Cabe destacar que el algoritmo "Dual-Simplex" es una versión distinta del Simplex enunciado en la introducción del trabajo. Esencialmente, implementa un método Simplex para el problema dual [9]. Las opciones I y L se basan en métodos de puntos interiores de tipo predictor-corrector y no admisibles (o sea, que los términos de la derecha en (1.21) no son cero). Son métodos similares basados en [6] y [10], pero con preprocesamientos distintos, y el método I tiene una rutina de factorización más eficiente.

2.2. Experimentos con tests de NETLIB

Se han resuelto con *linprog* 109 problemas de programación lineal de la librería NETLIB [2]. Se han organizado los resultados en ocho tablas, las cuatro primeras donde se comparan los métodos "Dual-Simplex" e "Interior-Point", y las cuatro siguientes donde se comparan los dos métodos de puntos interiores implementados en *linprog* (I y L).

Los problemas de NETLIB presentan la siguiente estructura:

$$(P_{NETLIB}) \begin{cases} \min\limits_{x\in\mathbb{R}^n} & f(x) = c^T x \\ \text{sujeto a} & Ax = b \\ & lb \leq x \leq ub \end{cases}, \tag{2.1}$$

donde $c \in \mathbb{R}^n$,$lb_i \in [-\infty, +\infty)$, $ub_i \in (-\infty, +\infty]$ para $i = 1, \ldots, n$, $b \in \mathbb{R}^m$ y $A \in \mathbb{R}^{m\times n}$ de rango máximo por filas. Esta estructura difiere de la formulación del problema (P) en las cotas sobre las variables, que pasan de ser $x \geq 0$ para ser definidas por los vectores lb para las cotas inferiores y ub para las cotas superiores.

Dual-Simplex* vs. *Interior-Point

Primero se van a comparar los métodos S e I. Para ello se presentan los resultados obtenidos con MATLAB en las tablas 2.5, 2.6, 2.7 y 2.8 que siguen la siguiente estructura: la primera columna (*Problema*) muestra el nombre por el que se conoce el problema y se va a usar para ordenarlos alfabéticamente. Las cuatro columnas siguientes determinan su tamaño: el número de las variables (n), de restricciones de igualdad (m), de restricciones de cota ($Cotas$) y de valores no nulos en la matriz de restricciones de igualdad, A (nnz). Entre los problemas analizados hay problemas con menos de 100 variables, como son "afiro", "kb2", "sc50a" y "sc50b" con 51, 68, 78 y 78 variables, respectivamente. También hay problemas con más de 100000 variables, como son "ken_18", "osa_30", "osa_60" y "pds_20" con tamaños 154699, 104374, 243246 y 108175 variables, respectivamente.

Para analizar los resultados se presenta el número de iteraciones ($Iter$) que ha ejecutado el programa con los algoritmos indicados. Posteriormente se muestran sus respectivos tiempos de ejecución[1] en segundos (t), que aparece como 0.00 cuando se ha tardado menos de 5 milésimas de segundo. También se incluye una comparación de los valores de la función objetivo en los puntos solución calculados:

$$diff = (f(x_I) - f(x_S)) \,/\, \text{máx}\{1, f(x_S)\} \ ,$$

siendo x_I y x_S los puntos solución calculados con los métodos I y S, respectivamente. Por último, la columna Ad muestra una medida de la admisibilidad en cada vector solución

$$Ad(x) = \text{máx}\left\{\|Ax - b\|_\infty,\ \text{máx}\left\{lb - x, 0\right\},\ \text{máx}\left\{x - ub, 0\right\}\right\} \ .$$

Lo primero que destaca de las tablas 2.5-2.8 son las filas marcadas en rojo. Estas corresponden a los problemas para los que el programa *linprog* no ha terminado satisfactoriamente con alguna de las dos opciones:

- El método S no resuelve el problema "greenbea" al superar el número máximo de iteraciones fijadas por el programa (79900 para este problema en concreto). Se ha observado que el valor de la función objetivo aumenta y disminuye continuamente. Para poder encontrar una solución se puede hacer un escalado del problema: reduciendo las variables por un factor $1e-4$ y escalando la función con $1e-5$ se obtienen los resultados de la tabla 2.1 en la que se encuentran también los originales del método I.

- El método I aplicado al problema "pds_20" llega a las 67 iteraciones y, aunque el valor de la función en la última iteración, $2.382166e+10$, se acerca al objetivo óptimo dado por NETLIB ($2.3821659e+10$), el programa es incapaz de computar una dirección de descenso. Si se hace el mismo escalado que con el problema anterior, aplicado a I, se obtienen los resultados de la tabla 2.1.

- El método I falla al resolver el problema "srcs8". El programa cataloga el problema como no acotado y no llega a resolverlo. Si se aplica una escala únicamente a la función objetivo de $1e-2$ se llega a que I acaba resultados en tabla 2.1

[1] *Elapsed time* calculado con las funciones *tic* y *toc*

Problema	Iter(I|S)	t(I|S)	diff	Ad(I|S)
greenbea	126|5919	0.45|0.71	-7e-04	1e-08|4e-08
pds_20	46|34404	140.05|11.50	3e-05	4e-11|4e-12
scrs8	13|477	0.10|0.03	4e-06	1e-08|7e-14

Tabla 2.1: Resultados de los problemas escalados. "Interior-Point" vs. "Dual-Simplex"

En el análisis de los resultados se han considerado los obtenidos con estos escalamientos. Los resultados muestran lo siguiente:

- El número de iteraciones que utiliza el método S es significativamente mayor que el método I. Las iteraciones de I están comprendidas entre 6 y 126, alcanzando el mínimo en el problema "recipe" y el máximo en el problema "greenbea", con una media de aproximadamente 20 iteraciones por problema. Por otro lado, el método S alcanza su mínimo en el problema "afiro" con 9 iteraciones y su máximo en el problema "qap_15" con 94144, la media de iteraciones es aproximadamente 4566. Se puede decir que el número de iteraciones del S es considerablemente mayor, siendo aproximadamente la media de iteraciones del S 36 veces mayor que el máximo de iteraciones de I y la media de S 228 veces la de I.

- Con respecto a los tiempos, el problema que menos ha tardado con el método I es "sc50b" con 0.0034 segundos y el que más tiempo de ejecución tiene es "pds_20" con 140.05 segundos, la media de tiempos de I es 2.61 segundos. El método S ha tardado entre 0.0081 y 429.90 segundos, con el mínimo en el problema "bore3d" y el máximo en "qap15". La media de los tiempos del método S es 5.13. Cabe destacar que en muchos casos los tiempos son muy parecidos: en un total de 55 problemas la diferencia es menor que una centésima de segundo, siendo "bandm", "pilot4" o "maros" algunos ejemplos. En cuatro problemas la diferencia es menor que una milésima de segundo, siendo estos: "finnis", "fit1d", "scfxm2" y "sctap3". El problema en el que el S más tarda respecto I es "qap15" siendo la diferencia de 383 segundos aproximadamente y el problema en el que I más tiempo emplea respecto de S es "pds_20" con aproximadamente 131 segundos de diferencia. Viendo que la media de iteraciones de S es unas 228 veces la de I, sería de esperar que la relación entre los tiempos fuese similar, pero la diferencia es mucho menor. La razón detrás de esto reside en el tamaño de los sistemas (1.21) y (1.48). Por cada iteración del algoritmo I se resuelven dos sistemas del tamaño $(2n+m) \times (2n+m)$. Teniendo en cuenta que los valores de n y m llegan a tamaños como 154699 y 105127 ("ken_15"), respectivamente, es de esperar que las iteraciones de I sean mucho más costosas. Para mejorar la eficacia de esos métodos, los sistemas se resuelven aprovechando la estructura de la matriz de coeficientes como se muestra en la sección 1.7.1, pero, aún así, el sistema reducido a resolver en el caso de "ken_15" sigue siendo muy grande, de tamaño 105127×105127.

- En cuanto a la columna diff, los resultados muestran que de un total de 109 problemas analizados, para 96 la solución calculada con S es ligeramente mejor que la solución calculada con I, para 3 es igual y para 10 la solución de I es mejor. La distancia relativa entre los valores óptimos, $abs(diff)$, está acotada por $6e-4$ ("greenbea").

- En cuanto a la columna Ad, los resultados obtenidos con S están acotados superiormente por $6.95e-07$ ("qap12") y los de I por $1.90e-06$ ("wood1p"), con medias de un orden similar (10^{-8}).

Se puede concluir que en términos generales el método I, sobre la colección NETLIB, ha sido más rápido y ha usado muchas menos iteraciones, pero que las soluciones han sido ligeramente peores en cuanto al valor de la función objetivo y a la admisibilidad.

Para observar el papel que han jugado el número de variables en los resultados, en la tabla 2.2 se presenta la información correspondiente al grupo de problemas de NETLIB con más de 100000 variables. En todos los problemas se tiene que las iteraciones de S son

Problema	n	Iter(I|S)	t(I|S)	diff	Ad(I|S)
ken_18	154699	29|69257	9.83|6.34	7e-25	3e-09|0e+00
osa_30	104374	28|6558	1.30|3.84	6e-17	9e-09|1e-11
osa_60	243246	32|16518	3.58|21.48	5e-17	8e-08|4e-11
pds_20	108175	46|34404	140.05|11.50	3e-05	4e-11|4e-12

Tabla 2.2: Resultados con NETLIB y $n \geq 100000$. "Interior-Point" vs. "Dual-Simplex"

mucho mayores que las de I. Por otro lado, los tiempos de ejecución son más dispares, siendo en dos de los problemas mayor el tiempo de I y en los otros mayor el de S. Se tiene que la diferencia es más significativa en el problema "pds_20", uno para los que se hace un escalamiento para resolver con I, mientras que para el problema más grande, "osa_60", el método I es más rápido que el S. Por último, S ha encontrado soluciones ligeramente mejores, sobre todo la de "pds_20".

Interior-Point vs. *Interior-Point-Legacy*

Las tablas 2.9, 2.10, 2.11 y 2.12 contienen los resultados de los dos métodos de puntos interiores de *linprog* (I y L) al resolver la colección de test de NETLIB. En ellas se usa

$$diff = (f(x_I) - f(x_L)) \,/\, \text{máx}\{1, f(x_L)\},$$

siendo x_L la solución calculada con el método L. Además, se añade una columna respecto a las tablas 2.5-2.8 en la que se muestra una estimación del valor $Iter/n$ del Teorema 1.5.2 (K).

En las tablas 2.9-2.12 la única ejecución fallida con el método L es "qap12". Falla ya que aparece un error de tipo numérico, "NaN"[2]. Haciendo el mismo escalado que para el problema "greenbea" en la subsección anterior, se obtienen los siguientes resultados:

Problema	Iter(I|L)	t(I|L)	diff	Ad(I|L)
qap12	14|9	15.96|21.53	-6e-03	4e-12|4e-09

Tabla 2.3: Resultados para "qap12" escalado. "Interior-Point" vs. "Interior-Point-Legacy"

Teniendo en cuenta los resultados con la escala original (tablas 2.9-2.12) y los de los escalados (tablas 2.1 y 2.4), las conclusiones que se pueden sacar de las tablas son:

[2] *Not A Number*

- En cuanto a las iteraciones, el método L utiliza entre 7 y 111 iteraciones (problemas "afiro" y "qap15" respectivamente), con una media de 23 iteraciones por problema. El método I alcanza un máximo de 126 iteraciones en "greenbea", pero su media de iteraciones, 20, es ligeramente inferior que la de L.

- En cuanto al tiempo de ejecución del método L, se comprende entre 0.0078 ("afiro") y 1538.81 ("qap15") segundos, con una media de 19. Un incremento considerable con respecto al método I, cuyos tiempos mínimo y máximo son 0.0034 y 140.05 y cuya media es de 2.61 segundos aproximadamente. Con lo que se tiene que la media del método L es poco más de 6 veces la del I.

- En cuanto al valor $diff$, 54 de los resultados indican que el método L ha encontrado una solución mejor que la de I y 55 que el resultado de I es más óptimo. Los valores $diff$ están acotados por $6e-03$ ("qap12").

- La admisibilidad de L está acotada superiormente por $6e-06$ ("pds_10") con una media de $2e-07$. La media de las admisibilidades es 10 veces la de I ($2e-08$) y la cota superior 6 veces la de I ($1e-06$). Mostrando que las soluciones de I tienen una ligera mejor calidad en este aspecto.

- En cuanto la razon $Iter/n$, se tiene que los resultados son muy similares: las medias son $2.14e-02$ para I y $2.42e-02$ para L. Con valores oscilando de $1.32e-04$ ("osa_60") a $1.76e-01$ ("kb2") en I, y entre $1.4e-04$ ("osa_60") y $2.21e-1$ ("kb2") en L .

Por lo general el método L hace uso de más iteraciones que el método I, solo en 24 problemas las iteraciones de I superan las de L. Los resultados indican que I es más rápido que L, la media de I es considerablemente mayor. El mayor de los tiempos está en "qap15", 1538.81 segundos, seguido por "pds_20" que tarda 384.72 segundos, este salto hace parecer al problema "qap15" un caso excepcional. Si se ignora "qap15", la media de tiempos pasa a ser 7.01 segundos que sigue siendo mayor que la de I. Por lo que, en cualquier caso, el método I es más rápido que el L.

Si se comparan los problemas de más de 100000 variables

Problema	n	Iter(I|L)	t(I|L)	diff	Ad(I|L)
ken_18	154699	29|37	9.83|21.22	-4e-20	3e-09|5e-07
osa_30	104374	28|30	1.30|2.29	-2e-16	9e-09|2e-08
osa_60	243246	32|34	3.58|6.87	5e-17	8e-08|3e-08
pds_20	108175	46|62	140.05|384.72	3e-05	4e-11|8e-08

Tabla 2.4: Resultados con NETLIB $n \geq 100000$. "Interior-Point" vs. "Interior-Point-Legacy"

En todos los problemas el método I es más rápido, todos los problemas han tardado entre 1.7 y 2.7 veces el tiempo de I cuando se resuelven con L. El método L también hace uso de más iteraciones. Con respecto a las soluciones, la única con una diferencia mínimamente significativa es la de "pds_20" que marca la solución de L como mejor. Las admisibilidades

de I son por lo general mejores, pero para ambos métodos todas son menores del orden de 10^{-6}.

Conclusión

La única conclusión que se puede sacar con la colección de problemas NETLIB es que, salvo los problemas de talla grande, las diferencias son poco significativas.

2.3. Experimentos generados de forma aleatoria

Para ver el comportamiento de los algoritmos de *linprog* también se han generado una serie de problemas aleatorios con la formulación del problema (P). Se han seguido dos procedimientos distintos en los que se han generado la matriz A y vectores c y b aleatoriamente, utilizando la función "rand"[3] de MATLAB con la semilla "$default$". Para asegurar que el rango de la matriz A sea máximo, se ha optado por las siguientes dos opciones:

- La primera opción se basa en generar una matriz con un bloque triangular superior. Para ello se crea una matriz de tamaño $m \times n$ que consiste en una matriz identidad $m \times m$ y ceros. Después se procede al llenado de la parte superior a la diagonal con números aleatorios. Para controlar el llenado de la matriz se ha usado un parámetro *beta* con el valor 0.01. Las correspondientes instrucciones de MATLAB son las siguientes:

```
m=n/2;
A=[speye(m), zeros(m,n-m)];
for j=i+1:n
    r=-1+2*rand;
    if abs(r)<= beta
        A(i,j)=r;
    end
end
```

- La segunda opción se basa en crear una matriz con un bloque simétrico definido positivo. Para ello se genera una matriz de tamaño $m \times n$ que consiste en una identidad tamaño $m \times m$ y ceros. Al igual que en la matriz con bloque triangular, se toma $beta = 0.01$ para controlar el llenado de la matriz y que sea definida positiva. Se procede a llenar el primer bloque de forma simétrica y después el resto de la matriz. El código usado es el siguiente:

[3]Esta función retorna un escalar aleatorio de una distribución uniforme (0,1)

```
m=n/2;
A=[speye(m), zeros(m,n-m)];
for i=1:m
    for j=i+1:m
        r=-1+2*rand;
        if abs(r)<= beta
            A(i,j)=r;
            A(j,i)=r;
        end
    end
    for j=m+1:n
        r=-1+2*rand;
        if abs(r)<= beta
            A(i,j)=r;
        end
     end
end
```

Para asegurar la existencia de puntos admisibles, se ha generado un punto "$x_{sol} \geq 0$" y se ha calculado el vector de términos independientes con la fórmula $b = Ax_{sol}$. El vector de la función objetivo, c, se ha calculado de forma que todos sus componentes sean positivas, con el objetivo de que la función sea coerciva y por tanto el problema (P) tenga solución global (ver en el anexo la proposición A.2.3).

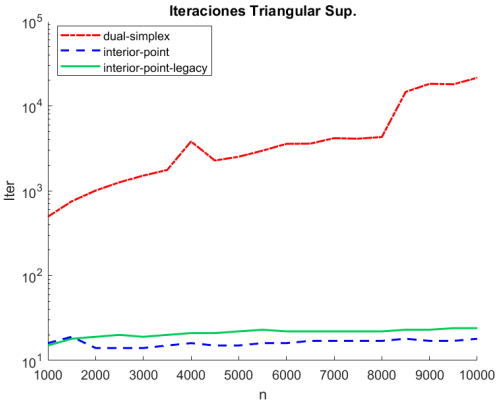
Los resultados obtenidos resolviendo los problemas aleatorios se muestran en dos tablas en la siguiente sección, la tabla 2.13 para los problemas con la matriz A generada con el bloque triangular superior y la 2.14 para los problemas con la matriz A generada con el bloque simétrico y definido positivo. La organización de las tablas es similar a la seguida en las 2.5-2.12, con la diferencia de que ahora las tablas incluyen los tres métodos de *linprog* (I, L y S) y que se prescinde de la columna $Cotas$ ya que todos los problemas tienen n cotas inferiores del tipo $x \geq 0$. Para comparar las soluciones entre sí, la última columna, $diff$, muestra, en orden, los valores

$$\frac{f(x_I) - f(x_S)}{\text{máx}\{1, f(x_S)\}}, \ \frac{f(x_I) - f(x_L)}{\text{máx}\{1, f(x_L)\}}, \ \frac{f(x_L) - f(x_S)}{\text{máx}\{1, f(x_S)\}}.$$

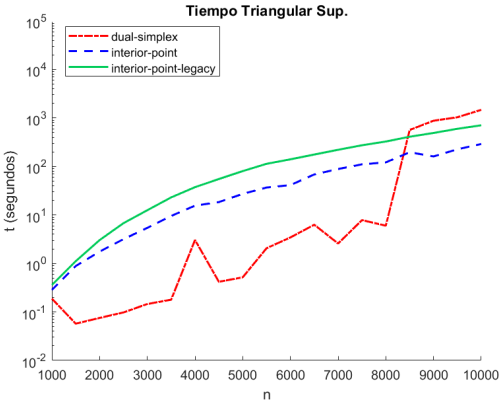
Los tamaños de los problemas van desde 1000 hasta 10000 variables, con saltos de 500 variables entre ellos. Se han resuelto un total de 38 problemas, 19 con cada definición de matriz.

Resultados con las matrices de bloque triangular

En la figura 2.1 se presenta a la izquierda el número de iteraciones y a la derecha el tiempo empleado para resolver los problemas aleatorios. Todo ello en relación al número de variables de cada problema.



(a) iteraciones



(b) tiempo

Figura 2.1: Resultados para tests aleatorios con matrices triangulares. Ver tabla 2.13.

Con respecto a los resultados, se puede destacar:

- En cuanto a las iteraciones, el método I emplea entre 14 y 19 iteraciones, con una media de 16 iteraciones por problema. El método L emplea entre 15 y 24 iteraciones, con una media de 21. El método S emplea entre 494 y 21458 iteraciones, con una media de 5806. La media de iteraciones de S es mucho mayor que el resto, siendo 363 veces la media de I y 276 la de L.

- Para el valor de las iteraciones respecto el tamaño del problema, se tiene que el método I oscila entre $1e-03$ y $2e-02$, con una media de $4e-03$. El valor para el método L oscila entre $2e-03$ y $2e-02$ con una media de $5e-03$. El valor para el método S oscila entre $4e-01$ y 2 con una media de $8e-01$. Los valores de S son los más grandes, el mínimo ya es 20 veces el máximo de los otros dos métodos.

- Con respecto a los tiempos, el método I ha tardado entre 0.28 y 290.73 segundos, con una media de 75. El método L ha tardado entre 0.36 y 707.53 segundos, con una media de 193.79. El método S ha tardado entre 0.05 y 1474.36 segundos, con una media de 209.98. La media de los tiempos de ejecución de S es casi 3 veces la media de I y la media de L es unas 2.5 veces la de I

- Con respecto a las admisibilidades, están acotadas por $7e-13$ para I, $2e-12$ para L y por $1e-10$ para S, las medias son del orden de 10^{-14}, 10^{-13} y 10^{-11}, respectivamente, despreciables teniendo en cuenta la precisión del ordenador.

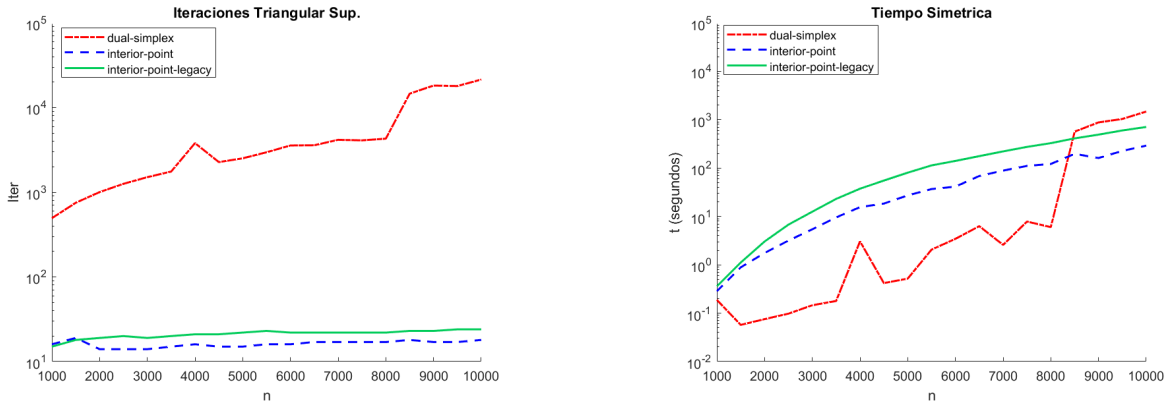
- Los resultados de $diff$ indican lo siguiente:

  - En 17 de 19 problemas, la solución de S es mejor que la solución de I.
  - En 11 problemas problemas la solución de L es mejor que la solución de I.
  - En 16 problemas la solución S es mejor que la solución L.

  Los valores de $diff$ están acotados en valor absoluto por $6e-07$ en la comparación entre I y S, $6e-07$ en la comparación entre I y L y por $9e-10$ en la comparación entre L y S.

Resultados con las matrices de bloque simétrico

Con los valores de la tabla 2.14 se han dibujado las gráficas de la figura 2.2.



(a) iteraciones (b) tiempo

Figura 2.2: Resultados para tests aleatorios con matrices simétricas. Ver la tabla 2.14.

De los resultados se puede destacar:

- En cuanto a las iteraciones, el método I emplea entre 14 y 19 iteraciones, con una media de 16 iteraciones por problema. El método L emplea entre 15 y 25 iteraciones, con una media de 21. El método S emplea entre 500 y 26367 iteraciones, con una media de 8002. La media de iteraciones de S es aproximadamente 500 veces la de I y 381 veces la de L.

- Para el valor de las iteraciones respecto el tamaño del problema, se tiene que el método I oscila entre $1e-03$ y $2e-02$, con una media de $4e-03$. El valor para el método L oscila entre $2e-03$ y $2e-02$ con una media de $5e-03$. El valor para el método S oscila entre $5e-01$ y 3 con una media de 1. La media de los valores de S es 250 veces la de I y 200 veces la de L.

- Con respecto a los tiempos, el método I ha tardado entre 0.34 y 549.53 segundos, con una media de 106.44. El método L ha tardado entre 0.61 y 735.48 segundos, con una media de 200.03. El método S ha tardado entre 0.34 y 20116.36 (5 horas y media) segundos, con una media de 2848.26.

- Con respecto a las admisibilidades, están acotadas por $2e-12$ para I, $1e-09$ para L y por $4e-07$ para S, las medias son del orden de 10^{-13}, 10^{-11} y 10^{-08}, respectivamente.

- Los resultados de $diff$ indican lo siguiente:

  - En 17 de 19 problemas, la solución de S es mejor que la de I.
  - En 13 problemas la solución de L es mejor que la de I.
  - En 18 problemas la solución S es mejor que la L.

Los resultados de $diff$ están acotados en valor absoluto por $7e-07$ en la comparación entre I y S, $7e-07$ en la comparación entre I y L y por $5e-10$ en la comparación entre L y S.

Conclusiones

Lo primero que se puede observar es que, con respecto a las iteraciones, el método S necesita considerablemente más iteraciones para resolver el mismo problema, diferencia que se vuelve más notable a medida que crece n como se puede ver en las gráficas 2.1a y 2.2a. Los métodos I y L no se ven casi afectados por el crecimiento de la dimensión del problema a diferencia del método S que experimenta un gran crecimiento de iteraciones a medida que aumenta n.

Por otro lado, en ambos casos, el método S empieza siendo más rápido que I y que L, pero en las últimas ejecuciones pasa a ser más lento. Se puede observar que el punto en el que S supera en tiempo de ejecución a los otros métodos se encuentra entre los problemas con 8000 y 9000 variables en la gráfica 2.1b, mientras que en la gráfica 2.2b se encuentra para los problemas que tienen entre 2000 y 3000 variables. Esto se debe a que el número de ceros en las matrices con bloque triangular es mucho mayor que en las definidas con bloque simétrico, lo que disminuye los cálculos. En los problemas con bloque simétrico, la media de tiempos de ejecución de I y L es mayor pero no llega al doble que la media de tiempos de los problemas con bloque triangular. Por otro lado, la media de tiempos de S en el caso simétrico es 14 veces la media de los problemas con bloque triangular.

En ambos tipos de problema se puede apreciar una perdida progresiva en la admisibilidad de las soluciones por parte del método S, mientras que en los métodos I y L se mantiene estable a medida que aumenta el valor de n (tablas 2.13 y 2.14). En cuanto a los incrementos de las medias de la admisibilidad en los problemas con estructura simétrica con respecto a la triangular, la media de I es 2 veces, la de L 168 veces y la de S 531 veces.

Las soluciones que por lo general son más óptimas son las proporcionadas por el método S, seguidas de L y por último las soluciones de I. Todas las distancias entre los resultados están acotadas por un error relativo $9.8977e-07$.

Es visible que para estos problemas los métodos de puntos interiores (I y L) ganan bastante en cuanto a iteraciones, tiempo y admisibilidad sobre todo a medida que aumenta el tamaño del problema. Aparte, se ven menos afectados por la distribución de elementos no nulos en las matrices de coeficientes de las restricciones. Además, dentro de los dos métodos de puntos interiores, es el método I el que menos iteraciones y tiempo emplea devolviendo puntos con mejor admisibilidad. Por otro lado, las soluciones aportadas por el Simplex suelen ser ligeramente más óptimas (columna $diff$), a cambio de tardar más tiempo en tallas grandes y necesitar más iteraciones para resolver los problemas. Por lo tanto, se puede decir que con estos experimentos, el método más rápido, pero que devuelve soluciones ligeramente menos óptimas es I y, ganando optimalidad pero perdiendo velocidad está L, para acabar con S que es el más lento pero el que devuelve soluciones más optimas.

2.4. Tablas

Dual-Simplex* vs. *Interior-Point

Problema	n	m	Cotas	nnz	Iter(I|S)	t(I|S)	diff	Ad(I|S)
25fv47	1876	821	1876	10705	21|2896	0.11|0.34	3e-14	3e-12|8e-12
80bau3b	12061	2262	15545	23264	28|3263	0.15|0.19	1e-15	1e-11|3e-11
adlittle	138	56	138	424	12|88	0.01|0.04	5e-18	7e-11|6e-14
afiro	51	27	51	102	7|9	0.00|0.01	0e+00	3e-14|1e-14
agg	615	488	615	2862	20|437	0.03|0.07	1e-21	9e-10|3e-08
agg2	758	516	758	4740	24|610	0.04|0.02	2e-22	2e-10|6e-11
agg3	758	516	758	4756	24|603	0.04|0.02	2e-21	2e-10|7e-11
bandm	472	305	472	2494	17|331	0.01|0.01	2e-13	5e-13|3e-12
beaconfd	295	173	295	3408	13|55	0.00|0.01	-3e-20	2e-11|1e-11
blend	114	74	114	522	9|90	0.00|0.01	-2e-15	4e-12|2e-13
bnl1	1586	643	1586	5532	24|892	0.03|0.03	5e-13	4e-13|2e-13
bnl2	4486	2324	4486	14996	30|1723	0.28|0.06	1e-11	3e-11|3e-12
bore3d	334	233	346	1448	12|47	0.00|0.01	2e-13	2e-08|2e-12
brandy	303	220	303	2202	16|236	0.01|0.01	5e-12	2e-09|1e-12
capri	482	271	615	1896	21|236	0.01|0.01	1e-12	2e-12|9e-13
cre_a	7248	3516	7248	18168	26|1302	0.07|0.05	-1e-23	3e-12|2e-12
cre_b	77137	9648	77137	260785	31|5853	1.16|1.05	-1e-23	4e-12|5e-13
cre_c	6411	3068	6411	15977	24|1151	0.08|0.07	4e-23	3e-12|5e-13
cre_d	73948	8926	73948	246614	28|3997	0.95|0.49	5e-19	2e-10|2e-13
cycle	3371	1903	3441	21234	19|1080	0.10|0.06	3e-06	2e-11|2e-11
czprob	3562	929	3791	10708	24|1194	0.02|0.04	3e-22	2e-12|5e-13
d2q06c	5831	2171	5831	33081	27|5958	0.41|1.00	8e-17	2e-08|3e-09
d6cube	6184	415	6184	37704	20|461	0.24|0.15	1e-14	6e-11|4e-11
degen2	757	444	757	4201	9|728	0.01|0.02	9e-13	1e-07|4e-15
degen3	2604	1503	2604	25432	12|2493	0.16|0.18	5e-12	4e-08|7e-15
dfl001	12230	6071	12243	35632	45|18449	28.85|6.36	1e-17	4e-10|8e-13
e226	472	223	472	2768	15|345	0.01|0.02	4e-09	1e-08|8e-14
etamacro	816	400	1033	2537	18|468	0.02|0.02	9e-11	4e-12|2e-12

Tabla 2.5: Problemas de NETLIB resueltos con Simplex y Puntos Interiores(I) con *linprog* (parte 1)

Problema	n	m	Cotas	nnz	Iter(I|S)	t(I|S)	diff	Ad(I|S)
fffff800	1028	524	1028	6401	28|281	0.03|0.01	2e-19	3e-11|3e-11
finnis	1064	497	1145	2760	21|417	0.01|0.01	4e-15	8e-10|3e-07
fit1d	1049	24	2075	13427	14|200	0.02|0.02	2e-13	4e-11|7e-12
fit1p	1677	627	2076	9868	13|735	0.09|0.04	3e-13	8e-13|3e-14
fit2d	10524	25	21024	129042	17|152	0.15|0.09	2e-13	1e-08|9e-12
fit2p	13525	3000	21025	50284	14|4871	0.70|1.37	7e-17	2e-13|2e-13
ganges	1706	1309	2103	6937	20|733	0.02|0.02	2e-19	2e-11|1e-11
gfrd_pnc	1160	616	1418	2445	28|302	0.01|0.01	8e-21	1e-11|2e-12
greenbea	5598	2392	5991	31070	126|79900	0.45|7.54	-	1e-08|-
greenbeb	5598	2392	6000	31070	34|4060	0.13|0.46	1e-17	7e-09|1e-11
grow15	645	300	1245	5620	15|1357	0.01|0.07	1e-24	5e-10|3e-09
grow22	946	440	1826	8252	15|1361	0.02|0.08	5e-23	5e-10|3e-07
grow7	301	140	581	2612	13|448	0.01|0.02	2e-23	2e-10|5e-09
israel	316	174	316	2443	19|243	0.02|0.01	1e-18	3e-11|1e-10
kb2	68	43	77	313	12|66	0.00|0.01	1e-13	4e-12|2e-12
ken_07	3602	2426	7204	8404	18|1639	0.02|0.03	1e-23	4e-12|0e+00
ken_11	21349	14694	42698	49058	25|10157	0.21|0.24	7e-24	1e-10|0e+00
ken_13	42659	28632	85318	97246	25|20952	0.67|0.89	4e-26	7e-12|0e+00
ken_18	154699	105127	309398	358171	29|69257	9.83|6.34	7e-25	3e-09|0e+00
lotfi	366	153	366	1136	22|148	0.01|0.01	4e-14	4e-12|2e-11
maros	1966	846	2001	10137	22|936	0.03|0.04	8e-16	5e-10|5e-10
maros_r7	9408	3136	9408	144848	20|2543	1.38|0.75	1e-20	3e-11|4e-08
modszk1	1620	687	1618	3168	21|408	0.03|0.02	-3e-15	9e-11|2e-10
osa_07	25067	1118	25067	144812	19|1386	0.19|0.20	2e-20	1e-10|5e-13
osa_14	54797	2337	54797	317097	27|3061	0.59|0.82	2e-16	1e-09|9e-13
osa_30	104374	4350	104374	604488	28|6558	1.30|3.84	6e-17	9e-09|1e-11
osa_60	243246	10280	243246	1408073	32|16518	3.58|21.48	5e-17	8e-08|4e-11
pds_02	7716	2953	9850	16571	35|1275	0.09|0.04	1e-26	4e-11|0e+00

Tabla 2.6: Problemas de NETLIB resueltos con Simplex y Puntos Interiores(I) con *linprog* (parte 2)

Problema	n	m	Cotas	nnz	Iter(I|S)	t(I|S)	diff	Ad(I|S)
pds_06	29351	9881	38591	63220	44|5956	3.37|0.32	0e+00	6e-11|0e+00
pds_10	49932	16558	66080	107605	50|11349	19.09|1.37	-1e-26	4e-11|4e-12
pds_20	108175	33874	143063	232647	67|34404	128.22|11.50	-	-|4e-12
perold	1506	625	1748	6148	26|1145	0.06|0.08	6e-14	4e-10|6e-09
pilot	4860	1441	6103	44375	32|5077	1.82|1.53	3e-10	1e-10|6e-07
pilot4	1123	410	1312	5264	23|654	0.05|0.05	8e-12	2e-09|4e-07
pilot87	6680	2030	8478	74949	31|12345	2.78|7.58	7e-09	4e-11|5e-09
pilot_ja	2267	940	2829	14977	29|1498	0.14|0.13	5e-12	2e-09|4e-08
pilot_we	2928	722	3220	9265	27|2747	0.07|0.23	1e-17	5e-10|3e-09
pilotnov	2446	975	2990	13331	21|1246	0.09|0.09	9e-20	6e-10|4e-07
qap12	8856	3192	8856	38304	14|62228	15.96|52.51	7e-07	4e-12|7e-07
qap15	22275	6330	22275	94950	18|94144	46.49|429.90	3e-07	4e-14|3e-11
qap8	1632	912	1632	7296	9|4823	0.55|0.59	2e-13	5e-13|2e-13
recipe	204	91	299	687	6|41	0.01|0.01	4e-11	9e-12|0e+00
sc105	163	105	163	340	11|77	0.00|0.01	6e-13	1e-13|2e-13
sc205	317	205	317	665	11|182	0.01|0.01	5e-09	5e-13|2e-12
sc50a	78	50	78	160	8|29	0.00|0.01	9e-11	6e-14|3e-14
sc50b	78	50	78	148	7|24	0.00|0.01	6e-18	6e-14|2e-13
scagr25	671	471	671	1725	17|552	0.01|0.02	9e-24	7e-12|7e-12
scagr7	185	129	185	465	15|166	0.00|0.01	2e-20	3e-12|2e-12
scfxm1	600	330	600	2732	20|398	0.02|0.02	0e+00	7e-12|8e-12
scfxm2	1200	660	1200	5469	21|837	0.03|0.03	5e-18	8e-12|7e-12
scfxm3	1800	990	1800	8206	20|1267	0.06|0.05	2e-17	1e-11|7e-12
scorpion	466	388	466	1534	11|118	0.01|0.01	5e-14	1e-13|7e-16
scrs8	1275	490	1275	3288	124|477	0.07|0.03	-	-|7e-14
scsd1	760	77	760	2388	7|99	0.01|0.01	4e-12	3e-14|3e-17
scsd6	1350	147	1350	4316	9|259	0.01|0.02	4e-12	3e-13|1e-15
scsd8	2750	397	2750	8584	9|879	0.01|0.05	7e-12	8e-11|3e-15

Tabla 2.7: Problemas de NETLIB resueltos con Simplex y Puntos Interiores(I) con *linprog* (parte 3)

Problema	n	m	Cotas	nnz	Iter(I|S)	t(I|S)	diff	Ad(I|S)
sctap1	660	300	660	1872	11|335	0.01|0.01	8e-16	4e-11|2e-14
sctap2	2500	1090	2500	7334	10|890	0.02|0.02	2e-18	3e-14|9e-16
sctap3	3340	1480	3340	9734	10|1196	0.02|0.02	1e-19	1e-14|2e-15
share1b	253	117	253	1179	18|172	0.01|0.01	4e-18	2e-10|1e-10
share2b	162	96	162	777	11|139	0.01|0.01	5e-12	7e-13|5e-13
shell	1777	536	2144	3558	15|342	0.01|0.02	2e-23	6e-11|0e+00
ship04l	2166	402	2166	6380	12|396	0.01|0.01	4e-18	2e-10|2e-14
ship04s	1506	402	1506	4400	12|315	0.01|0.01	2e-18	1e-10|3e-14
ship08l	4363	778	4363	12882	16|686	0.02|0.02	3e-22	4e-14|2e-14
ship08s	2467	778	2467	7194	14|437	0.01|0.02	3e-22	5e-14|2e-14
ship12l	5533	1151	5533	16276	14|962	0.02|0.03	9e-18	3e-11|2e-14
ship12s	2869	1151	2869	8284	14|593	0.01|0.02	3e-22	3e-14|3e-14
sierra	2735	1227	4771	8001	30|1006	0.04|0.02	2e-20	1e-11|2e-12
stair	614	356	696	4003	12|655	0.02|0.04	2e-11	4e-13|3e-12
standata	1274	359	1394	3230	12|72	0.01|0.01	3e-12	3e-11|3e-14
standgub	1383	361	1503	3338	12|72	0.01|0.01	3e-12	3e-11|3e-14
standmps	1274	467	1394	3878	22|204	0.02|0.01	3e-17	2e-12|4e-13
stocfor1	165	117	165	501	10|61	0.00|0.01	-4e-21	9e-13|5e-13
stocfor2	3045	2157	3045	9357	16|1441	0.04|0.05	-2e-20	2e-12|7e-12
stocfor3	23541	16675	23541	72721	24|11617	0.36|1.03	7e-14	3e-12|1e-11
truss	8806	1000	8806	27836	16|20915	0.10|3.58	7e-20	2e-13|2e-13
tuff	628	333	655	4561	14|152	0.01|0.01	7e-05	2e-10|1e-12
vtp_base	346	198	428	1051	11|59	0.00|0.01	-4e-21	4e-11|3e-11
wood1p	2595	244	2595	70216	16|128	0.09|0.04	7e-06	2e-06|5e-14
woodw	8418	1098	8418	37487	22|820	0.08|0.08	3e-05	5e-13|1e-13

Tabla 2.8: Problemas de NETLIB resueltos con Simplex y Puntos Interiores(I) con *linprog* (parte 4)

Interior-Point* vs. *Interior-Point-Legacy

Problema	n	m	Cotas	nnz	Iter(I|S)	t(I|S)	diff	Ad(I|S)	K(I|L)
25fv47	1876	821	1876	10705	21|25	0.11|0.26	3e-14	3e-12|3e-10	1e-02|1e-02
80bau3b	12061	2262	15545	23264	28|37	0.15|0.45	-6e-16	1e-11|1e-09	2e-03|3e-03
adlittle	138	56	138	424	12|12	0.01|0.03	5e-18	7e-11|5e-11	9e-02|9e-02
afiro	51	27	51	102	7|7	0.00|0.01	-3e-12	3e-14|1e-12	1e-01|1e-01
agg	615	488	615	2862	20|18	0.03|0.08	-1e-17	9e-10|2e-06	3e-02|3e-02
agg2	758	516	758	4740	24|16	0.04|0.07	-2e-18	2e-10|3e-08	3e-02|2e-02
agg3	758	516	758	4756	24|16	0.04|0.07	-1e-19	2e-10|7e-09	3e-02|2e-02
bandm	472	305	472	2494	17|17	0.01|0.05	2e-13	5e-13|6e-11	4e-02|4e-02
beaconfd	295	173	295	3408	13|12	0.00|0.02	-1e-14	2e-11|3e-10	4e-02|4e-02
blend	114	74	114	522	9|12	0.00|0.06	-1e-14	4e-12|8e-12	8e-02|1e-01
bnl1	1586	643	1586	5532	24|26	0.03|0.07	-3e-14	4e-13|3e-09	2e-02|2e-02
bnl2	4486	2324	4486	14996	30|32	0.28|0.35	1e-11	3e-11|3e-10	7e-03|7e-03
bore3d	334	233	346	1448	12|17	0.00|0.06	-3e-12	2e-08|6e-10	4e-02|5e-02
brandy	303	220	303	2202	16|16	0.01|0.07	5e-12	2e-09|2e-08	5e-02|5e-02
capri	482	271	615	1896	21|18	0.01|0.04	1e-12	2e-12|1e-07	4e-02|4e-02
cre_a	7248	3516	7248	18168	26|28	0.07|0.23	-1e-16	3e-12|7e-09	4e-03|4e-03
cre_b	77137	9648	77137	260785	31|40	1.16|7.53	-6e-21	4e-12|1e-09	4e-04|5e-04
cre_c	6411	3068	6411	15977	24|27	0.08|0.17	-5e-17	3e-12|5e-10	4e-03|4e-03
cre_d	73948	8926	73948	246614	28|34	0.95|6.55	-6e-18	2e-10|5e-10	4e-04|5e-04
cycle	3371	1903	3441	21234	19|27	0.10|0.36	3e-06	2e-11|5e-10	6e-03|8e-03
czprob	3562	929	3791	10708	24|31	0.02|0.11	-2e-17	2e-12|1e-08	7e-03|9e-03
d2q06c	5831	2171	5831	33081	27|31	0.41|0.96	2e-17	2e-08|4e-07	5e-03|5e-03
d6cube	6184	415	6184	37704	20|24	0.24|0.33	-3e-14	6e-11|1e-09	3e-03|4e-03
degen2	757	444	757	4201	9|14	0.01|0.09	9e-13	1e-07|3e-13	1e-02|2e-02
degen3	2604	1503	2604	25432	12|21	0.16|0.50	5e-12	4e-08|6e-10	5e-03|8e-03
dfl001	12230	6071	12243	35632	45|51	28.85|44.44	-5e-17	4e-10|3e-06	4e-03|4e-03
e226	472	223	472	2768	15|22	0.01|0.11	4e-09	1e-08|9e-10	3e-02|5e-02
etamacro	816	400	1033	2537	18|28	0.02|0.08	9e-11	4e-12|2e-11	2e-02|3e-02

Tabla 2.9: Comparación de métodos de puntos interiores (parte 1)

Problema	n	m	Cotas	nnz	Iter(I|S)	t(I|S)	diff	Ad(I|S)	K(I|L)
fffff800	1028	524	1028	6401	28|25	0.03|0.12	-5e-16	3e-11|2e-08	3e-02|2e-02
finnis	1064	497	1145	2760	21|26	0.01|0.07	5e-16	8e-10|5e-08	2e-02|2e-02
fit1d	1049	24	2075	13427	14|18	0.02|0.14	2e-13	4e-11|1e-08	1e-02|2e-02
fit1p	1677	627	2076	9868	13|15	0.09|0.24	3e-13	8e-13|5e-11	8e-03|9e-03
fit2d	10524	25	21024	129042	17|22	0.15|0.46	2e-13	1e-08|2e-09	2e-03|2e-03
fit2p	13525	3000	21025	50284	14|20	0.70|0.52	-4e-16	2e-13|7e-11	1e-03|1e-03
ganges	1706	1309	2103	6937	20|18	0.02|0.17	1e-16	2e-11|4e-06	1e-02|1e-02
gfrd_pnc	1160	616	1418	2445	28|19	0.01|0.03	-1e-17	1e-11|7e-10	2e-02|2e-02
greenbea	5598	2392	5991	31070	126|44	0.45|0.59	1e-09	1e-08|2e-06	2e-02|8e-03
greenbeb	5598	2392	6000	31070	34|35	0.13|0.49	-1e-16	7e-09|8e-07	6e-03|6e-03
grow15	645	300	1245	5620	15|17	0.01|0.06	1e-24	5e-10|5e-10	2e-02|3e-02
grow22	946	440	1826	8252	15|18	0.02|0.05	-8e-24	5e-10|2e-09	2e-02|2e-02
grow7	301	140	581	2612	13|15	0.01|0.02	-2e-23	2e-10|6e-10	4e-02|5e-02
israel	316	174	316	2443	19|22	0.02|0.10	2e-19	3e-11|2e-10	6e-02|7e-02
kb2	68	43	77	313	12|15	0.00|0.11	1e-13	4e-12|1e-09	2e-01|2e-01
ken_07	3602	2426	7204	8404	18|14	0.02|0.10	-3e-22	4e-12|4e-12	5e-03|4e-03
ken_11	21349	14694	42698	49058	25|20	0.21|0.56	-4e-20	1e-10|2e-10	1e-03|9e-04
ken_13	42659	28632	85318	97246	25|26	0.67|1.80	-2e-23	7e-12|7e-08	6e-04|6e-04
ken_18	154699	105127	309398	358171	29|37	9.83|21.22	-4e-20	3e-09|5e-07	2e-04|2e-04
lotfi	366	153	366	1136	22|18	0.01|0.03	-8e-13	4e-12|3e-09	6e-02|5e-02
maros	1966	846	2001	10137	22|32	0.03|0.19	8e-16	5e-10|1e-08	1e-02|2e-02
maros_r7	9408	3136	9408	144848	20|14	1.38|2.88	-5e-16	3e-11|3e-09	2e-03|1e-03
modszk1	1620	687	1618	3168	21|23	0.03|0.09	-3e-12	9e-11|7e-10	1e-02|1e-02
osa_07	25067	1118	25067	144812	19|30	0.19|0.61	-4e-19	1e-10|2e-10	8e-04|1e-03
osa_14	54797	2337	54797	317097	27|39	0.59|1.46	3e-17	1e-09|5e-09	5e-04|7e-04
osa_30	104374	4350	104374	604488	28|30	1.30|2.29	-2e-16	9e-09|2e-08	3e-04|3e-04
osa_60	243246	10280	243246	1408073	32|34	3.58|6.87	5e-17	8e-08|3e-08	1e-04|1e-04
pds_02	7716	2953	9850	16571	35|26	0.09|0.24	-3e-20	4e-11|5e-07	5e-03|3e-03

Tabla 2.10: Comparación de métodos de puntos interiores (parte 2)

Problema	n	m	Cotas	nnz	Iter(I|S)	t(I|S)	diff	Ad(I|S)	K(I|L)
pds_06	29351	9881	38591	63220	44|40	3.37|5.93	-4e-20	6e-11|1e-07	1e-03|1e-03
pds_10	49932	16558	66080	107605	50|51	19.09|35.12	-1e-19	4e-11|6e-06	1e-03|1e-03
pds_20	108175	33874	143063	232647	67|62	128.22|384.72	-	-|8e-08	-|6e-04
perold	1506	625	1748	6148	26|33	0.06|0.20	6e-14	4e-10|8e-09	2e-02|2e-02
pilot	4860	1441	6103	44375	32|36	1.82|1.63	3e-10	1e-10|3e-09	7e-03|7e-03
pilot4	1123	410	1312	5264	23|28	0.05|0.21	7e-12	2e-09|1e-07	2e-02|2e-02
pilot87	6680	2030	8478	74949	31|38	2.78|5.26	8e-09	4e-11|3e-09	5e-03|6e-03
pilot_ja	2267	940	2829	14977	29|30	0.14|0.35	4e-12	2e-09|1e-08	1e-02|1e-02
pilot_we	2928	722	3220	9265	27|56	0.07|0.52	-1e-18	5e-10|8e-07	9e-03|2e-02
pilotnov	2446	975	2990	13331	21|20	0.09|0.28	1e-18	6e-10|5e-10	9e-03|8e-03
qap12	8856	3192	8856	38304	14|103	15.96|213.50	-	4e-12|-	2e-03|-
qap15	22275	6330	22275	94950	18|111	46.49|1538.81	3e-07	4e-14|1e-08	8e-04|5e-03
qap8	1632	912	1632	7296	9|9	0.55|0.49	-1e-11	5e-13|3e-11	6e-03|6e-03
recipe	204	91	299	687	6|9	0.01|0.11	4e-11	9e-12|1e-09	3e-02|4e-02
sc105	163	105	163	340	11|10	0.00|0.05	6e-13	1e-13|8e-12	7e-02|6e-02
sc205	317	205	317	665	11|11	0.01|0.01	5e-09	5e-13|5e-11	3e-02|3e-02
sc50a	78	50	78	160	8|9	0.00|0.08	6e-11	6e-14|2e-11	1e-01|1e-01
sc50b	78	50	78	148	7|8	0.00|0.06	-8e-16	6e-14|4e-13	9e-02|1e-01
scagr25	671	471	671	1725	17|16	0.01|0.02	-7e-20	7e-12|5e-09	3e-02|2e-02
scagr7	185	129	185	465	15|12	0.00|0.01	-6e-18	3e-12|3e-10	8e-02|6e-02
scfxm1	600	330	600	2732	20|18	0.02|0.09	2e-19	7e-12|1e-09	3e-02|3e-02
scfxm2	1200	660	1200	5469	21|19	0.03|0.13	-9e-18	8e-12|1e-08	2e-02|2e-02
scfxm3	1800	990	1800	8206	20|20	0.06|0.09	-6e-16	1e-11|4e-08	1e-02|1e-02
scorpion	466	388	466	1534	11|14	0.01|0.04	5e-14	1e-13|2e-13	2e-02|3e-02
scrs8	1275	490	1275	3288	124|24	0.07|0.04	-	-|3e-11	-|2e-02
scsd1	760	77	760	2388	7|10	0.01|0.03	6e-13	3e-14|1e-13	9e-03|1e-02
scsd6	1350	147	1350	4316	9|12	0.01|0.02	4e-12	3e-13|2e-13	7e-03|9e-03
scsd8	2750	397	2750	8584	9|11	0.01|0.03	7e-12	8e-11|1e-13	3e-03|4e-03

Tabla 2.11: Comparación de métodos de puntos interiores (parte 3)

Problema	n	m	Cotas	nnz	Iter(I|S)	t(I|S)	diff	Ad(I|S)	K(I|L)
sctap1	660	300	660	1872	11|17	0.01|0.02	8e-16	4e-11|4e-11	2e-02|3e-02
sctap2	2500	1090	2500	7334	10|18	0.02|0.10	-1e-12	3e-14|2e-12	4e-03|7e-03
sctap3	3340	1480	3340	9734	10|18	0.02|0.06	-1e-18	1e-14|2e-12	3e-03|5e-03
share1b	253	117	253	1179	18|20	0.01|0.02	-7e-16	2e-10|6e-08	7e-02|8e-02
share2b	162	96	162	777	11|12	0.01|0.07	2e-12	7e-13|2e-09	7e-02|7e-02
shell	1777	536	2144	3558	15|19	0.01|0.14	-2e-22	6e-11|7e-08	8e-03|1e-02
ship04l	2166	402	2166	6380	12|12	0.01|0.10	-9e-18	2e-10|1e-11	6e-03|6e-03
ship04s	1506	402	1506	4400	12|13	0.01|0.02	3e-16	1e-10|2e-07	8e-03|9e-03
ship08l	4363	778	4363	12882	16|15	0.02|0.14	-5e-16	4e-14|1e-09	4e-03|3e-03
ship08s	2467	778	2467	7194	14|14	0.01|0.02	-8e-20	5e-14|2e-11	6e-03|6e-03
ship12l	5533	1151	5533	16276	14|15	0.02|0.06	-3e-15	3e-11|4e-10	3e-03|3e-03
ship12s	2869	1151	2869	8284	14|16	0.01|0.03	-6e-18	3e-14|2e-11	5e-03|6e-03
sierra	2735	1227	4771	8001	30|16	0.04|0.09	-2e-20	1e-11|2e-11	1e-02|6e-03
stair	614	356	696	4003	12|15	0.02|0.08	2e-11	4e-13|8e-09	2e-02|2e-02
standata	1274	359	1394	3230	12|16	0.01|0.13	3e-12	3e-11|1e-10	9e-03|1e-02
standgub	1383	361	1503	3338	12|16	0.01|0.16	3e-12	3e-11|8e-10	9e-03|1e-02
standmps	1274	467	1394	3878	22|24	0.02|0.06	3e-17	2e-12|2e-11	2e-02|2e-02
stocfor1	165	117	165	501	10|17	0.00|0.06	-1e-18	9e-13|8e-11	6e-02|1e-01
stocfor2	3045	2157	3045	9357	16|22	0.04|0.17	5e-18	2e-12|9e-09	5e-03|7e-03
stocfor3	23541	16675	23541	72721	24|34	0.36|1.22	7e-14	3e-12|3e-08	1e-03|1e-03
truss	8806	1000	8806	27836	16|18	0.10|0.15	-1e-16	2e-13|3e-11	2e-03|2e-03
tuff	628	333	655	4561	14|22	0.01|0.17	7e-05	2e-10|8e-12	2e-02|4e-02
vtp_base	346	198	428	1051	11|25	0.00|0.10	-9e-19	4e-11|7e-11	3e-02|7e-02
wood1p	2595	244	2595	70216	16|20	0.09|0.20	7e-06	2e-06|2e-12	6e-03|8e-03
woodw	8418	1098	8418	37487	22|27	0.08|0.34	3e-05	5e-13|1e-11	3e-03|3e-03

Tabla 2.12: Comparación de métodos de puntos interiores (parte 4)

Problemas generados de forma aleatoria

n	m	nnz	Iter(I|L|S)	t(I|L|S)	diff	Ad(I|L|S)	K(I|L|S)
1000	500	4231	16|15|494	0.29|0.36|0.19	4e-15|-2e-12|2e-12	8e-14|2e-14|7e-14	2e-02|2e-02|5e-01
1500	750	9253	19|18|751	0.89|1.11|0.06	6e-14|-4e-16|6e-14	3e-14|1e-14|2e-12	1e-02|1e-02|5e-01
2000	1000	15939	14|19|1004	1.76|3.03|0.08	5e-11|5e-11|3e-14	1e-15|2e-14|1e-12	7e-03|7e-03|5e-01
2500	1250	24657	14|20|1258	3.17|6.77|0.10	4e-09|4e-09|2e-11	2e-15|6e-14|4e-12	6e-03|6e-03|5e-01
3000	1500	35205	14|19|1507	5.41|12.53|0.15	4e-13|3e-13|1e-14	1e-15|2e-14|4e-12	5e-03|5e-03|5e-01
3500	1750	47799	15|20|1756	9.54|23.04|0.18	5e-14|-3e-10|3e-10	2e-15|8e-14|1e-11	4e-03|4e-03|5e-01
4000	2000	62138	16|21|3805	15.59|37.46|3.05	4e-11|2e-11|2e-11	2e-15|5e-14|1e-11	4e-03|4e-03|1e+00
4500	2250	78109	15|21|2272	18.44|55.12|0.42	3e-08|3e-08|4e-13	3e-15|5e-13|1e-11	3e-03|3e-03|5e-01
5000	2500	96223	15|22|2517	27.22|80.30|0.52	1e-13|-4e-14|1e-13	2e-15|4e-13|1e-11	3e-03|3e-03|5e-01
5500	2750	116526	16|23|2967	36.83|113.64|2.06	5e-08|5e-08|3e-14	3e-15|5e-13|2e-11	3e-03|3e-03|5e-01
6000	3000	138408	16|22|3570	41.31|140.50|3.44	1e-08|1e-08|6e-14	1e-13|6e-14|2e-11	3e-03|3e-03|6e-01
6500	3250	161670	17|22|3591	68.55|176.49|6.30	3e-08|3e-08|9e-10	3e-15|2e-12|2e-11	3e-03|3e-03|6e-01
7000	3500	187038	17|22|4162	88.69|221.11|2.58	-3e-13|-3e-10|3e-10	3e-15|2e-13|2e-11	2e-03|2e-03|6e-01
7500	3750	214492	17|22|4103	110.87|273.95|7.83	-1e-13|-4e-14|-1e-13	4e-15|8e-14|2e-11	2e-03|2e-03|5e-01
8000	4000	243820	17|22|4297	121.62|327.44|6.02	6e-07|6e-07|8e-10	8e-13|1e-12|3e-11	2e-03|2e-03|5e-01
8500	4250	275167	18|23|14617	196.53|410.98|566.86	1e-11|8e-12|3e-12	3e-15|6e-13|1e-10	2e-03|2e-03|2e+00
9000	4500	308720	17|23|18211	161.18|491.40|878.68	1e-08|1e-08|2e-12	8e-15|6e-14|1e-10	2e-03|2e-03|2e+00
9500	4750	343981	17|24|17974	226.49|599.33|1036.80	8e-11|7e-11|3e-12	3e-15|2e-13|1e-10	2e-03|2e-03|2e+00
10000	5000	379036	18|24|21458	290.73|707.53|1474.36	1e-11|1e-11|2e-12	7e-15|1e-13|1e-10	2e-03|2e-03|2e+00

Tabla 2.13: Resultados con matrices aleatorias de estructura triangular superior

n	m	nnz	Iter(I|L|S)	t(I|L|S)	diff	Ad(I|L|S)	K(I|L|S)
1000	500	4231	16|15|494	0.29|0.36|0.19	4e-15|-2e-12|2e-12	8e-14|2e-14|7e-14	1e-02|1e-02|5e-01
1500	750	9253	19|18|751	0.89|1.11|0.06	6e-14|-4e-16|6e-14	3e-14|1e-14|2e-12	1e-02|1e-02|5e-01
2000	1000	15939	14|19|1004	1.76|3.03|0.08	5e-11|5e-11|3e-14	1e-15|2e-14|1e-12	7e-03|7e-03|5e-01
2500	1250	24657	14|20|1258	3.17|6.77|0.10	4e-09|4e-09|2e-11	2e-15|6e-14|4e-12	6e-03|6e-03|5e-01
3000	1500	35205	14|19|1507	5.41|12.53|0.15	4e-13|3e-13|1e-14	1e-15|2e-14|4e-12	5e-03|5e-03|5e-01
3500	1750	47799	15|20|1756	9.54|23.04|0.18	5e-14|-3e-10|3e-10	2e-15|8e-14|1e-11	4e-03|4e-03|6e-01
4000	2000	62138	16|21|3805	15.59|37.46|3.05	4e-11|2e-11|2e-11	2e-15|5e-14|1e-11	4e-03|4e-03|6e-01
4500	2250	78109	15|21|2272	18.44|55.12|0.42	3e-08|3e-08|4e-13	3e-15|5e-13|1e-11	3e-03|3e-03|1e+00
5000	2500	96223	15|22|2517	27.22|80.30|0.52	1e-13|-4e-14|1e-13	2e-15|4e-13|1e-11	3e-03|3e-03|6e-01
5500	2750	116526	16|23|2967	36.83|113.64|2.06	5e-08|5e-08|3e-14	3e-15|5e-13|2e-11	3e-03|3e-03|5e-01
6000	3000	138408	16|22|3570	41.31|140.50|3.44	1e-08|1e-08|6e-14	1e-13|6e-14|2e-11	3e-03|3e-03|2e+00
6500	3250	161670	17|22|3591	68.55|176.49|6.30	3e-08|3e-08|9e-10	3e-15|2e-12|2e-11	2e-03|2e-03|5e-01
7000	3500	187038	17|22|4162	88.69|221.11|2.58	-3e-13|-3e-10|3e-10	3e-15|2e-13|2e-11	2e-03|2e-03|2e+00
7500	3750	214492	17|22|4103	110.87|273.95|7.83	-1e-13|-4e-14|-1e-13	4e-15|8e-14|2e-11	2e-03|2e-03|3e+00
8000	4000	243820	17|22|4297	121.62|327.44|6.02	6e-07|6e-07|8e-10	8e-13|1e-12|3e-11	2e-03|2e-03|2e+00
8500	4250	275167	18|23|14617	196.53|410.98|566.86	1e-11|8e-12|3e-12	3e-15|6e-13|1e-10	2e-03|2e-03|3e+00
9000	4500	308720	17|23|18211	161.18|491.40|878.68	1e-08|1e-08|2e-12	8e-15|6e-14|1e-10	2e-03|2e-03|6e-01
9500	4750	343981	17|24|17974	226.49|599.33|1036.80	8e-11|7e-11|3e-12	3e-15|2e-13|1e-10	2e-03|2e-03|5e-01
10000	5000	379036	18|24|21458	290.73|707.53|1474.36	1e-11|1e-11|2e-12	7e-15|1e-13|1e-10	2e-03|2e-03|3e+00

Tabla 2.14: Resultados para matrices aleatorias con simetría con *linprog*

Apéndice A

A.1. Notación.

$\mathbb{R}^{m\times n}$	Conjunto de matrices reales de tamaño $m \times n$.
$\mathbb{R}_+$	Conjunto de valores no negativos de la recta real $\{x \in \mathbb{R} \ : \ x \geq 0\}$.
log	Logaritmo neperiano : $\log_e$.
X	Matriz diagonal asociada al vector x: $X = diag(x)$.
e	Vector $(1, 1, \ldots, 1)^T$.
$N(A)$	Núcleo de la matriz A. $\{x \in \mathbb{R}^n \mid Ax = 0\}$.
$\mathcal{A}(P)$	Conjunto de puntos admisibles del problema (P).
μ	Medida de dualidad definida como el producto escalar de x y s entre el tamaño del problema n: $\mu = x^T s/n$ para $x, s \in \mathbb{R}^n,\ (x, s) > 0$.
$x \geq 0$ **con** $x \in \mathbb{R}^n$	$x_i \geq 0$ para $i = 1, \ldots, n$.

A.2. Resultados de optimización

Teorema A.2.1. *Sea $\Omega \subset \mathbb{R}^n$ un conjunto abierto, $f : \Omega \longrightarrow \mathbb{R}$ C^2, $K \subset \Omega$ un conjunto convexo. Entonces:*

a) f es convexa sobre K sii $(x-y)^T \nabla^2 f(x)(x-y) \geq 0, \quad \forall x, y \in K$

b) f es estrictamente convexa sobre $K \Longleftarrow (x-y)^T \nabla^2 f(x)(x-y) > 0, \quad \forall x, y \in K,\ x \neq y$.

Definición A.2.2. *Sea $f : K \subset \mathbb{R}^n \longrightarrow \mathbb{R}$ para $K \neq \emptyset$. Se dice que es una* ***función coerciva*** *en K si*

$$f(x^k) \longrightarrow +\infty, \quad \forall (x^k)_{k\in\mathbb{N}} \subset K \text{ tal que } \|x^k\| \longrightarrow +\infty \text{ cuando } k \longrightarrow +\infty$$

Siendo el problema de optimización

$$(\tilde{P}) \begin{cases} \min\limits_{x\in\mathbb{R}^n} & f(x) \\ \text{sujeto a} & x \in K \subseteq \Omega \end{cases}, \tag{A.1}$$

con $K \in \mathbb{R}^n$ no vacío y Ω abierto. Se enuncian los siguientes resultados:

Proposición A.2.3. *Si f es coerciva en K, continua y K es cerrado, entonces existe al menos una solución (global) para $(\tilde{P})$.*

Teorema A.2.4. *Si existe solución para $(\tilde{P})$, K es un conjunto convexo y f es una función estrictamente convexa en K, entonces el problema $(\tilde{P})$ admite una única solución (global).*

Teorema A.2.5. *(Regla de los multiplicadores de Lagrange). Sean $f, h_i, g_j : \Omega \longrightarrow \mathbb{R}$ funciones clase C^1,$1 \leq i \leq n_I, 1 \leq j \leq n_D$ y*

$$K = \{x \in \Omega : \ h_i(x) = 0, \ i = 1, \ldots, n_I; \ g_j(x) \leq 0 \ j = 1, \ldots, n_D\}.$$

Si $\bar{x} \in \Omega$ es una solución del problema $(\tilde{P})$, entonces existen $1 + n_D + n_I$ números: $\bar{\alpha} \in \mathbb{R}_+, \{\bar{\lambda}_i\}_{i=1}^{n_I} \subset \mathbb{R}$ y $\{\bar{s}_j\}_{j=1}^{n_D} \subset \mathbb{R}_+$ tales que:

$$\bar{\alpha} + \sum_{i=1}^{n_I} |\bar{\lambda}_i| + \sum_{j=1}^{n_D} \bar{s}_j > 0 \ , \tag{A.2a}$$

$$\bar{\alpha}\nabla f(\bar{x}) + \sum_{i=1}^{n_I} \bar{\lambda}_i \nabla h_i(\bar{x}) + \sum_{j=1}^{n_D} \bar{s}_j \nabla g(\bar{x}) = 0 \ , \tag{A.2b}$$

$$\bar{s}_j \geq 0 \quad y \quad \bar{s}_j g_j(\bar{x}) = 0, \quad 1 \leq j \leq n_D \ , \tag{A.2c}$$

$$\tag{A.2d}$$

$$h_i(\bar{x}) = 0, \quad 1 \leq i \leq n_I, \quad g_j(\bar{x}) \leq 0, \quad 1 \leq j \leq n_D \ . \tag{A.2e}$$

Puede tomarse $\bar{\alpha} = 1$ si además de las hipótesis ya mencionadas, las restricciones que forman el conjunto K son lineales.

Cuando $\bar{\alpha} = 1$, las condiciones se conocen como ***condiciones de Kuhn-Tucker*** *y al punto $\bar{x}$ que las verifica como* ***punto de Kuhn-Tucker****.*

Si se tiene el problema de optimización

$$(P) \begin{cases} \min\limits_{x\in\mathbb{R}^n} & f(x) = c^T x \\ \textit{sujeto a} & Ax = b \\ & x \geq 0 \end{cases}, \tag{A.3}$$

con $c \in \mathbb{R}^n, A \in \mathbb{R}^{m\times n}$ y $b \in \mathbb{R}^m$, para $m < n$, y $\bar{x}$ un punto admisible. Entonces $\bar{x}$ es solución de (P) si y solo si $\exists \bar{\lambda} \in \mathbb{R}^m, \bar{s} \in \mathbb{R}^n$ tales que

$$c - A^T\bar{\lambda} + \bar{s} = 0 \ , \tag{A.4a}$$

$$\bar{s}_i > 0 \quad y \quad \bar{s}_i\bar{x}_i = 0 \text{ para } i = 1, \ldots, n \ , \tag{A.4b}$$

$$A\bar{x} - b = 0 \quad y \quad \bar{x}_i \geq 0 \text{ para } i = 1, \ldots, n. \tag{A.4c}$$

Teorema A.2.6. *(Condiciones suficientes de primer orden) Sean* $f, h_i, g_j : \Omega \longrightarrow \mathbb{R}$ *funciones de clase* C^1, $1 \leq i \leq n_I, 1 \leq j \leq n_D$, *tales que*

- f *es una función convexa en* K.
- $K = \{x \in \Omega : \ h_i(x) = 0, \ i = 1, \ldots, n_I; \ g_j(x) \leq 0 \ j = 1, \ldots, n_D\}$ *es un conjunto convexo.*

Si $\bar{x} \in K$ *es un punto Kuhn-Tucker entonces* $\bar{x}$ *es solución global del problema* $(\tilde{P})$.

Bibliografía

[1] G. B. Dantzig. Linear programming. *Operations Research*, 50(1):42–47, 2002.

[2] D. M. Gay. Electronic mail distribution of linear programming test problems. *Mathematical Programming Society COAL Newsletter*, 1985.

[3] J. Gondzio. Interior point methods 25 years later. *European Journal of Operational Research*, 218(3):587–601, 2012.

[4] N. Karmarkar. A new polynomial-time algorithm for linear programming. In *Proceedings of the Sixteenth Annual ACM Symposium on Theory of Computing*, pages 302–311, 1984.

[5] L. G. Khachiyan. A polynomial algorithm in linear programming. In *Doklady Akademii Nauk*, volume 244, pages 1093–1096. Russian Academy of Sciences, 1979.

[6] S. Mehrotra. On the implementation of a primal-dual interior point method. *SIAM Journal on Optimization*, 2:575–601, 11 1992.

[7] S. J. Wright. *Primal-Dual Interior-Point Methods*. SIAM, 1997.

[8] V. Klee y G. J. Minty. How good is the simplex algorithm. *Inequalities*, 3(3):159–175, 1972.

[9] J. Nocedal y S. J. Wright. *Numerical Optimization*. Springer, New York, NY, USA, second edition, 2006.

[10] Y. Zhang. Solving large-scale linear programs by interior-point methods under the MATLAB environment. *Optimization Methods and Software*, 10(1):1–31, 1998.