Sección de Matemáticas
Universidad de La Laguna

Virginia Vargas Mesa

Problemas Complementarios Lineales

Linear Complementary Problems

Trabajo Fin de Grado
Grado en Matemáticas
La Laguna, Septiembre de 2019

Dirigido por
Carlos González Martín

Carlos González Martín
Departamento de Matemáticas, Estadística e Investigación Operativa
Universidad de La Laguna
38200 La Laguna, Tenerife

Agradecimientos

A mi tutor Carlos González Martín por su guía e inestimable ayuda. Sin él este trabajo no hubiera sido posible.

A mi familia porque gracias a su apoyo absoluto, su fe ciega y un amor incalculable he conseguido llegar hasta aquí.

A Yanet, mi incondicional amiga, que desde el principio hasta el fin ha estado ahí apoyándome, sobre todo en los momentos más bajos. Por esas risas, alegrías, peleas y llantos que hemos compartido a lo largo de esta etapa, sin ti nada hubiera sido lo mismo.

A Daniela, por su eterna amistad, sus inigualables consejos y por esas dosis de locura que hacen falta en la vida.

A Adrián y Michael, quienes han sido uno de los mayores apoyo, por estar ahí cuando hacía falta y por su lealtad sin límites.

A todos mis amigos, porque no hay palabras para agradecer todas esas sonrisas y momentos inolvidables vividos juntos.

Y por último, pero no menos importante, quiero agradecer a mi profesor de segundo de bachillerato, Francisco, quien me demostró lo bonitas que son las matemáticas.

Virginia Vargas Mesa
La Laguna, 12 de septiembre de 2019

Resumen · Abstract

Resumen

En este trabajo se estudian los problemas de Programación Complementaria Lineal. La importancia de este estudio viene enfatizada por el hecho de que, entre otros, los problemas de Programación Lineal, los de Programación Cuadrática y los de Juegos Bimatriciales pueden modelizarse como problemas de Programación Complementaria Lineal.

Para su resolución, se presenta el algoritmo de pivotaje complementario de Lemke, se estudia en detalle y se aplica a la resolución de distintos casos prácticos.

El trabajo se completa con la prolongación del estudio anterior a la resolución de problemas de Programación Cuadrática convexa y Juegos Bimatriciales.

Por último, se usa la versión programada en python del algoritmo de Lemke, para resolver algunos ejemplos prácticos relevantes.

Palabras clave: *Programación Complementaria Lineal – Programación Lineal – Programación Cuadrática – Juegos Bimatriciales – Algoritmo de pivotaje complementario de Lemke.*

Abstract

In this project the problems of Linear Complementary Programming are studied. The importance of this study is emphasized by the fact that, among others, Linear Programming, Quadratic Programming and Bimatrix Games problems can be modeled as problems of Linear Complementary Programming.

For its resolution, Lemke's complementary pivot algorithm is presented, studied in detail and applied to the resolution of different case studies.

The work is completed with the prolongation of the previous study to the resolution of problems of Convex Quadratic Programming and Bimatricial Games.

Finally, the python version of Lemke's algorithm is used to solve some relevant practical examples.

Keywords: *Linear Complementary Problems – Linear Programming – Quadratic Programming – Bimatrix Games – Lemke's complementary pivoting algorithm.*

Índice general

Introducción

Un problema complementario lineal (LCP) consiste en la resolución de un sistema particular de ecuaciones lineales, con variables no negativas y cumpliendo una condición de complementariedad por pares específicos. La resolución de problemas de Programación Lineal, de Programación Cuadrática con restricciones lineales y de Juegos Bimatriciales se puede acometer a través de un LCP conveniente. La modelización como LCP se puede utilizar en el estudio de otros muchos casos (ver [2]).

Entre los algoritmos desarrollados para resolver LCPs nos interesa destacar el algoritmo de pivotaje complementario, o algoritmo de Lemke. Este algoritmo, que trabaja de forma parecida a como lo hace el Método del Simplex, se ha utilizado, por ejemplo, para generar cálculos de equilibrios económicos y para resolver determinados sistemas de ecuaciones y problemas de programación no lineal. (Ver [5])

Por otro lado, son muy prometedores los métodos iterativos planteados para la resolución de los LCP enfocados a los programas lineales de gran tamaño que no pueden afrontarse con el método del simplex por sus dimensiones y, por tanto, con dificultades computacionales. Por ello, el estudio de los LCP promete grandes beneficios en el ámbito de la optimización.(Ver [5])

Este trabajo se estructura en cuatro capítulos y un apéndice.

Después de esta introducción, a lo largo del primer capítulo hablamos del problema general y de alguno de los problemas relacionados con él, como es el caso de la programación lineal.

En el capítulo dos estudiamos el algoritmo de Lemke, utilizando ejemplos para mostrar su funcionamiento y finalizando con el estudio de su convergencia.

A continuación, en el capítulo siguiente, estudiamos y aplicamos este algoritmo en el caso de los problemas de programación cuadrática y los juegos bimatriciales, resolviendo los correspondientes ejemplos.
Para finalizar el trabajo, usamos la versión programada del algoritmo en python para la resolución de ejemplos presentados en capítulos previos y de algunos ejemplos prácticos de LCP.

1

Planteamiento del problema. Problemas relacionados. Ejemplos

1.1. Planteamiento del problema

El LCP fue propuesto por Cottle y Dantzig en 1968 [3] y se modeliza de la siguiente forma:

M es una matriz cuadrada de orden n y q un vector columna en $\mathbb{R}^n$, debemos encontrar $w = (w_1, \ldots, w_n)^T$ y $z = (z_1, \ldots, z_n)^T$ pertenecientes a $\mathbb{R}^n$, que satisfagan las siguientes condiciones:

$$\begin{gathered} w - Mz = q \\ w \geqslant 0, z \geqslant 0 \quad \text{y} \quad w_i z_i = 0, \quad \forall i \in \{1, ..., n\}. \end{gathered} \tag{1.1}$$

En este problema los únicos datos conocidos son el vector q y la matriz M. Esto sirve para representar al LCP con el par (q, M). Si la dimensión de los vectores y la matriz es n, entonces decimos que el LCP es de orden n y posee $2n$ variables.

Ejemplo 1.1 *Si tomamos un sistema de orden 2 donde* $M = \begin{pmatrix} 1 & 1 \\ 1 & 2 \end{pmatrix}$ *y* $q = (-2, -3)$*, tendríamos el siguiente sistema:*

$$\begin{gathered} w_1 - z_1 - z_2 = -2 \\ w_2 - z_1 - 2z_2 = -3 \\ w_1, w_2, z_1, z_2 \geqslant 0 \quad y \quad w_1 z_1 = w_2 z_2 = 0 \end{gathered} \tag{1.2}$$

Esta expresión se puede escribir en forma matricial como:

$$w_1 \begin{pmatrix} 1 \\ 0 \end{pmatrix} + w_2 \begin{pmatrix} 0 \\ 1 \end{pmatrix} + z_1 \begin{pmatrix} -1 \\ -1 \end{pmatrix} + z_2 \begin{pmatrix} -1 \\ -2 \end{pmatrix} = \begin{pmatrix} -2 \\ -3 \end{pmatrix} \tag{1.3}$$

$$w_1, w_2, z_1, z_2 \geqslant 0 \qquad y \qquad w_1 \cdot z_1 = w_2 \cdot z_2 = 0 \tag{1.4}$$

Para que se satisfaga 1.3, en cada par (w_i, z_i), al menos una de las variables debe ser cero. Una forma de resolver el problema sería igualar a cero una variable de cada par. Las variables que restan son llamadas variables utilizables. Si el sistema resultante al sustituir las variables nulas tiene una solución donde las variables utilizables no son negativas, es decir, cumplen las condiciones 1.4, entonces se tiene una solución del problema.

Si en el Ejemplo 1.1 planteado anteriormente tomamos $w_1 = w_2 = 0$, las variables utilizables serán z_1 y z_2. Sustituyendo en 1.3 nos queda el siguiente sistema:

$$z_1 \begin{pmatrix} -1 \\ -1 \end{pmatrix} + z_2 \begin{pmatrix} -1 \\ -2 \end{pmatrix} = \begin{pmatrix} -2 \\ -3 \end{pmatrix} = \begin{pmatrix} q_1 \\ q_2 \end{pmatrix} = q \tag{1.5}$$

$$z_1 \geqslant 0, z_2 \geqslant 0$$

Este sistema tiene una solución si y sólo si el vector $q = (q_1, q_2)^T$ se puede expresar como combinación lineal no negativa de los vectores que acompañan a las variables z_1 y z_2. En el Ejemplo 1.1, $(-2, -3)^T$, puede escribirse como combinación lineal no negativa de los vectores $(-1, -1)^T$ y $(-1, -2)^T$.

Si dibujamos el conjunto de combinaciones lineales no negativas de estos vectores, obtenemos un cono:

Para que el LCP tenga una solución con z_1 y z_2 como variables utilizables, el vector dado q debe encontrarse en este cono. Verificamos que el punto $(-2, -3)^T$ se encuentra en el cono y por tanto la solución de 1.3 es $z_1 = 1$ y $z_2 = 1$, por lo que una solución para 1.2 es $(w_1, w_2; z_1, z_2) = (0, 0; 1, 1)$. El cono de la figura 1.1 es conocido como un cono complementario asociado con el LCP 1.2 (ver [5]).

Conos complementarios

Los conos complementarios son generalizaciones de cuadrantes u ortantes. En el LCP, estos son determinados por la matriz M, no teniendo en cuenta el vector q. Dada M una matriz cuadrada de orden n, tenemos que $C(M)$, la clase de conos complementarios asociada a M, se obtiene calculando $(I_{.j}, -M_{.j})$ a los que denotaremos como $A_{.j}$.

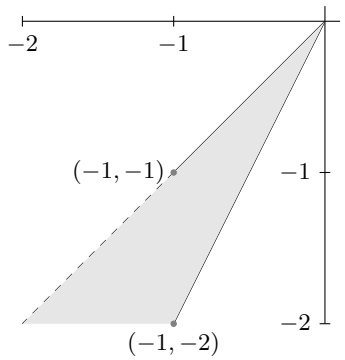

Figura 1.1. Cono complementario

El conjunto ordenado $(A_{\cdot 1}, ..., A_{\cdot n})$ es conocido como un conjunto complementario de vectores, donde el

$$Pos(A_{\cdot 1}, ..., A_{\cdot n}) = \{y : y = \alpha_1 A_{\cdot 1} + ... + \alpha_n A_{\cdot n}; \alpha_1 \geqslant 0, ...\alpha_n \geqslant 0\}$$

es llamado cono complementario en la clase $C(M)$. Hay 2^n conos complementarios, ya que los vectores $A_{\cdot j} \in \mathbb{R}^{2n}$

Conos complementarios degenerados y no degenerados

Un cono complementario en $C(M)$, $Pos(A_{\cdot 1}, ...A_{\cdot n})$ se dice que es no degenerado si tiene un interior no vacío, es decir, si $\{A_{\cdot 1}, ..., A_{\cdot n}\}$ es un conjunto linealmente independiente y se dice degenerado si su interior es vacío, o cuando $\{A_{\cdot 1}, ..., A_{\cdot n}\}$ es un conjunto linealmente dependiente.

El LCP (q, M), siendo M la matriz cuadrada de orden n y $q \in \mathbb{R}^n$ el vector columna, es semejante al problema de encontrar un cono complementario en $C(M)$ que contenga al punto q. Al ser así, encontrar un conjunto complementario de vectores $(A_{\cdot 1}, ..., A_{\cdot n})$ de tal manera que:

i) $A_{\cdot j} \in \{I_{\cdot j}, -M_{\cdot j}\}$ para $j \in \{1, ..., n\}$
ii) q sea combinación lineal no negativa $(A_{\cdot 1}, ..., A_{\cdot n})$

es equivalente a encontrar $w \in \mathbb{R}^n$; $z \in \mathbb{R}^n$ que cumplan:

$$\sum_{j=1}^{n} I_{\cdot j} w_j - \sum_{j=1}^{n} M_{\cdot j} z_j = q,$$

$$w_j \geqslant 0, \quad z_j \geqslant 0 \quad \forall j \in \{1, ...n\}, \quad \text{y} \quad w_j = 0 \quad \text{o} \quad z_j = 0 \quad \forall j \in \{1, ..., n\}.$$

O, de forma equivalentemente, resolver:

$$w - Mz = q \tag{1.6}$$

$$w \geqslant 0, \quad z \geqslant 0 \tag{1.7}$$

$$w_j \cdot z_j = 0, \forall j \in \{1, ..., n\} \tag{1.8}$$

La condición 1.8 se puede escribir como $\sum_{j=1}^{n} w_j z_j = w^T z = 0$. Esta se conoce como restricción de complementariedad. Cualquier solución del LCP (q, M) cumple que en el par (w_j, z_j) si una de las variables es positiva la otra debe ser nula, por tanto cada variable es el complemento de la otra.

En 1.6 tomamos el vector w_j como $I_{.j}$ y z_j como $-M_{.j}$, por lo que $(I_{.j}, -M_{.j})$ es el par complementario de vectores en LCP, con $j = 1, \ldots, n$. Tenemos que siendo $y_j \in \{w_j, z_j\}$ y $A_{.j}$ el vector correspondiente a y_j, entonces $A_{.j} \in \{I_{.j}, -M_{.j}\}$.

El vector $y = (y_1, ..., y_n)$ es un vector complementario de variables en este LCP, $(A_{.1}, ..., A_{.n})$ es un conjunto complementario de vectores correspondientes a y y A es la matriz formada por los vectores columna $A_{.1}, ..., A_{.n}$. A dicha matriz A la llamamos matriz complementaria del problema.

Si A es una matriz cuyos vectores columna son $A_{.1}, ..., A_{.n}$ en ese orden, con $A_{.1}, ..., A_{.n}$ linealmente independientes, entonces A es la base complementaria de 1.6 correspondiente al vector básico complementario y.

El cono $Pos(A_{.1}, ..., A_{.n}) = \{x : x = \alpha_1 A_{.1} + ... + \alpha_n A_{.n}, \alpha_1 \geqslant 0, ..., \alpha_n \geqslant 0\}$ es el cono complementario en la clase $C(M)$ asociado al conjunto complementario de vectores $(A_{.1}, ..., A_{.n})$, o el vector complementario asociado a y. Para que (w, z) sea solución del LCP, se deben satisfacer todas las restricciones 1.6, 1.7 y 1.8.

Definición 1.1 *Un vector básico factible complementario para el LCP es un vector básico complementario que cumple que q puede expresarse como combinación no negativa de los vectores columnas pertenecientes a la base complementaria, así cada vector básico factible complementario da una solución del LCP.*

La unión de todos los conos complementarios asociados a M lo denotaremos como $K(M)$. Este es el conjunto de todos los vectores q para los que LCP tiene al menos una solución. El vector $\bar{z}$ nos proporcionará una solución del LCP (q, M), si y solo si, $(\bar{w} = M\bar{z} + q, \bar{z})$ es solución de este LCP.

Para el LCP de orden 2 visto en 1.2 los vectores complementarios y las matrices complementarias son los siguientes:

Vectores complementarios de las variables	Correspondiente matriz complementaria
(w_1, w_2)	$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$
(w_1, z_2)	$\begin{pmatrix} 1 & 0 \\ -1 & -2 \end{pmatrix}$
(z_1, w_2)	$\begin{pmatrix} -1 & -1 \\ 0 & 1 \end{pmatrix}$
(z_1, z_2)	$\begin{pmatrix} -1 & -1 \\ -1 & -2 \end{pmatrix}$

Cuadro 1.1. Vectores y matrices complementarias en el LCP 1.2 de orden 2 .

Dado que estas matrices complementarias son no singulares, éstas serán bases complementarias y los vectores complementarios serán básicos en este LPC. Además, como en 1.2, $q = (-5, -6)^T$ se puede expresar como combinación lineal no negativa de la matriz complementaria asociada a (z_1, z_2), sabemos que $q = (-5, -6)^T$ es un vector básico complementario factible para el LCP planteado.

El método de enumeración total para el LCP [5]

Como se ha descrito, en un LCP (q, M) de orden n, por 1.8 sabemos que cualquier solución (w, z) del sistema, debe cumplir que o bien $w_j = 0$ o $z_j = 0$. Esta propiedad le da un carácter combinatorio que inevitablemente nos lleva a un método de enumeración para este tipo de problemas. Sabemos que hay 2^n vectores complementarios de variables, tomamos $y^r = (y_1^r, ..., y_n^r), r = 1, \ldots, 2^n$ donde $y^r \in \{w_j, z_j\}$ para cada $j = 1$ a n, sean todos estos vectores complementarios de variables. Sea A_r la matriz complementaria asociada a y^r, con $r = 1, \ldots, 2^n$, formamos el siguiente sistema:

$$A_r \cdot y^r = q$$
$$y^r \geqslant 0.$$

Este puede resolverse aplicando la fase I del Método Simplex para resolución de problemas de Programación Lineal o con otros métodos de resolución de sistemas de ecuaciones o inecuaciones lineales. Si este sistema tiene una solución factible $\bar{y^r}$, entonces decimos que $y^r = \bar{y^r}$, donde las variables que no pertenezcan a y^r son iguales a cero, es solución de LCP (q, M). Cuando la matriz complementaria A_r es singular, el sistema 1.1 puede tener varias soluciones factibles.

Como hemos dicho, cada solución factible de 1.1 nos proporciona una solución de LCP (q, M). Si esto ocurre para cada $r = 1, \ldots, 2^n$ podríamos obtener todas las soluciones del LCP.
El ejemplo 1.1, con $n = 2$ se ha solucionado mediante este procedimiento.

El método de enumeración es conveniente para sistemas pequeños como lo es para $n = 2$, ya que $2^2 = 4$. Además, en este caso se puede comprobar que tiene solución para cualquier r dibujando el cono complementario correspondiente y verificando si este contiene a q. En el caso de $n > 2$ el método se complica y deja de ser útil a medida que el valor de n aumenta, puesto que 2^n crece rápidamente.

1.2. Problemas relacionados

Los problemas de programación lineal, programación cuadrática y los juegos bimatriciales pueden modelizarse como un LCP, transformándose el problema fundamental en un sistema cuyo objetivo es encontrar w y z tal que:

$$\begin{aligned} w - Mz &= q \\ w \geqslant 0, z &\geqslant 0 \end{aligned} \tag{1.9}$$

$$w_i z_i = 0, \quad \forall i \tag{1.10}$$

En este apartado, nos centraremos en los problemas de programación lineal y se estudiarán los problemas de programación cuadrática y los juegos bimatriciales en el capítulo 3.

1.2.1. Problemas de programación lineal

Usamos la forma simétrica siguiente:

$$\begin{aligned} \text{Minimizar} \quad & c^T x \\ \text{Sujeto a} \quad & Ax \geqslant b \\ & x \geqslant 0 \end{aligned} \tag{1.11}$$

Este problema primal expresado en forma Dual será:

$$\begin{aligned} \text{Maximizar} \quad & b^T y \\ \text{Sujeto a} \quad & A^T y \leqslant b \\ & x \geqslant 0 \end{aligned}$$

En este caso, sea A un matriz de orden $m \times n$, por la condición de holguras complementarias sabemos que para que x sea solución del problema primal e y solución del problema dual debe verificarse que $u^T x = 0$ y $v^T y = 0$, con $u = c - A^T y$ un vector de holgura primal y $v = Ax - b$ un vector de holgura dual. Por lo que nos queda que debe satisfacerse:

$$\begin{pmatrix} u \\ v \end{pmatrix} - \begin{pmatrix} 0 & -A^T \\ A & 0 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} c^T \\ -b \end{pmatrix}$$
$$\begin{pmatrix} u \\ v \end{pmatrix} \geqslant \begin{pmatrix} x \\ y \end{pmatrix} \geqslant 0 \quad \text{y} \quad \begin{pmatrix} u \\ v \end{pmatrix}^T \begin{pmatrix} x \\ y \end{pmatrix} = 0 \tag{1.12}$$

Recíprocamente, si u, v, y, x satisfacen las condiciones 1.12 podemos decir que x es una solución óptima de 1.11. Hay que tener en cuenta que en 1.12 todos los vectores se escriben de forma particionada, por ejemplo (u, v) es el vector $(u_1, ..., u_n, v_1, ..., v_m)^T$. Si tomamos $N = m + n$,

$$w = \begin{pmatrix} u \\ v \end{pmatrix}, \quad z = \begin{pmatrix} x \\ y \end{pmatrix}, \quad M = \begin{pmatrix} 0 & -A^T \\ A & 0 \end{pmatrix}, \quad q = \begin{pmatrix} c^T \\ -b \end{pmatrix},$$

Podemos ver que 1.12 es un LCP de orden N de la forma 1.6, 1.7, 1.8, por lo que la solución de LP 1.11 se puede obtener resolviendo el LCP 1.12. Además, los pares complementarios de variables en el LCP 1.11 son los del par de LP primarios, duales 1.11 y su dual.

Ejemplo 1.2 *Tenemos el LP siguiente:*

$$\begin{aligned} \text{Minimizar} \quad & 5x_1 + 12x_2 + 4x_3 \\ \text{sujeto a} \quad & 4x_1 + 6x_2 - 2x_3 - x_4 = 20 \\ & 2x_1 - 4x_2 + 4x_3 - x_5 = 16 \\ & x_j \geqslant 0 \quad \text{para} \quad j = 1, ...5. \end{aligned}$$

Hacemos los cambios necesarios hasta obtener el problema en la forma 1.11. En este caso eliminamos las variables no negativas x_4 y x_5 para transformar las restricciones de igualdad en desigualdades, quedando el LP de la siguiente manera:

$$\begin{aligned} \text{Minimizar} \quad & 5x_1 + 12x_2 + 4x_3 \\ \text{sujeto a} \quad & 4x_1 + 6x_2 - 2x_3 \geqslant 20 \\ & 2x_1 - 4x_2 + 4x_3 \geqslant 16 \\ & x_i \geqslant 0 \quad \text{para} \quad i = 1, 2, 3. \end{aligned}$$

Y su dual será:

$$\begin{aligned} \text{Maximizar} \quad & 20y_1 + 16y_2 \\ \text{sujeto a} \quad & 4y_1 + 2y_2 \leqslant 5 \\ & 6y_1 - 4y_2 \leqslant 12 \\ & -2y_1 + 4y_2 \leqslant 4 \\ & y_j \geqslant 0 \quad \text{para} \quad j = 1, 2. \end{aligned}$$

Sean u_i, con $i = 1, 2, 3$ las variables de holgura dual no negativas asociadas con la restricción dual correspondiente a la variable principal x_i y sea (v_1, y_1), (v_2, y_2) la variable inactiva no negativa y la variable dual respectivamente, asociada con las dos restricciones primarias en ese orden.
Entonces, los sistemas primarios y duales junto con las condiciones de holgura complementarias para la optimalidad son:

$$\begin{aligned} 4x_1 + 6x_2 - 2x_3 - v_1 &= 20 \\ 2x_1 - 4x_2 + 4x_3 - v_2 &= 16 \\ 4y_1 + 2y_2 + u_1 &= 5 \\ 6y_1 - 4y_2 + u_2 &= 12 \\ -2y_1 + 4y_2 + u_3 &= 4 \\ x_i, u_i, y_j, v_j &\geqslant 0 \\ x_i u_i = y_j v_j &= 0 \\ \text{para todo} \quad i = 1, 2, 3 \quad \text{y} \quad j &= 1, 2. \end{aligned}$$

Este es el LCP del problema de programación lineal planteado en este ejemplo. Además podemos observar en la siguiente tabla que todas las variables cumplen que son $\geqslant 0$ y que $x_1u_1 = x_2u_2 = x_3u_3 = y_1v_1 = y_2v_2 = 0$.

u_1	u_2	u_3	v_1	v_2	x_1	x_2	x_3	y_1	y_2	
1	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*4*	*2*	*5*
0	*1*	*0*	*0*	*0*	*0*	*0*	*0*	*6*	*-4*	*12*
0	*0*	*1*	*0*	*0*	*0*	*0*	*0*	*-2*	*4*	*4*
0	*0*	*0*	*1*	*0*	*-4*	*-6*	*2*	*0*	*0*	*20*
0	*0*	*0*	*0*	*1*	*-2*	*4*	*-4*	*0*	*0*	*16*

2

Algoritmo de pivotaje complementario de Lemke. Aplicación a ejemplos.

2.1. Algoritmo de pivotaje complementario de Lemke

Existen varios algoritmos para resolver el problema complementario lineal, el método más conocido es el método de Lemke, también conocido como algoritmo de pivotaje complementario, ya que elige la variable de entrada por la regla de pivotaje complementario.

Procedemos entonces a describir este método para resolver LCPs, que converge bajo ciertos supuestos de no degeneración y cuando M satisface ciertas propiedades (ver, por ejemplo, [2], [5] y [3])

Definición 2.1 *Sea (w_j, z_j) un par de variables complementarias. Una solución (w, z) para el LCP se llama una solución factible complementaria.*

Asimismo, decimos que esta es una solución básica factible complementaria si el par (w, z) es una solución básica factible para el LCP, siendo básica una única variable del par (w_j, z_j), $\forall j = 1, ..., n$.

Si $q \geqslant 0$, haciendo $w = q$ y $z = 0$ se obtienen una solución básica factible complementaria. Por otro lado, si $q \ngeqslant 0$, se puede considerar una nueva columna e_n, siendo este un vector de tamaño n cuyas entradas son 1, y una variable artificial, z_0 generándose el siguiente sistema:

$$w - Mz - e_n z_0 = q \tag{2.1}$$

$$w_j \geqslant 0, z_j \geqslant 0, z_0 \geqslant 0 \quad \forall j = 1, ..., n. \tag{2.2}$$

$$w_j z_j = 0 \quad \forall j = 1, ..., n. \tag{2.3}$$

Evidentemente, si z=0 tenemos el LCP inicial.

Haciendo $z_0 = \text{máximo}\{-q_i : 1 \leqslant i \leqslant n\}$, $z = 0$ y $w = q + 1z_0$, obtenemos una solución del sistema 2.1, 2.2 y 2.3. Trabajaremos hasta obtener una solución del problema complementario lineal en la que, si es posible, la variable artificial, z_0, se anule.

Definición 2.2 *Partiendo del sistema definido en 2.1, 2.2 y 2.3,denominamos solución básica factible casi complementaria a una solución factible* (w, z, z_0) *para este problema, de tal manera que cumpla las siguientes condiciones:*

1) (w, z, z_0) *es una solución factible básica para 2.1 y 2.2*
2) Para algún $s \in \{1, \ldots, n\}$ *ni* w_s*, ni* z_s *son básicas.*
3) z_0 *y exactamente una variable de cada par complementario* (w_j, z_j) *son básicas, para* $j = 1, \ldots, n$ *y* $j \neq s$*.*

Si (w, z, z_0) *es una solución básica factible casi complementaria, donde las variables* w_s *y* z_s *son no básicas, obtenemos una solución factible básica complementaria adyacente* $(\hat{w}, \hat{z}, \hat{z_0})$*, al realizar un pivotaje introduciendo la variable* w_s *o* z_s *en la base en lugar de una variable diferente a* z_0*.*

Queda claro que cada solución básica factible casi complementaria posee, como máximo, dos soluciones básicas casi complementarias adyacentes. Si al introducir en la base w_s o z_s, es expulsada de la base la variable artificial z_0 se consigue una solución básica factible complementaria, entonces habrán menos de dos soluciones factibles básicas casi complementarias adyacentes.

2.1.1. Método de Pivotaje Complementario

El método de resolución del problema complementario lineal, conocido como algoritmo de pivotaje complementario o Algoritmo de Lemke,genera soluciones básicas factibles casi adyacentes hasta que se obtiene una solución básica factible complementaria o se llegue a la imposibilidad de pivotar y, por tanto, a la denominada terminación en rayo.

Más adelante se mostrará para que condiciones de M este algoritmo converge en un número finito de pasos.

Un esquema del algoritmo es el siguiente:

- **Inicio:**
  Si $q \geqslant 0$, paramos. La solución básica factible complementaria es $(w, z) = (q, 0)$. En caso contrario, transcribimos el sistema 2.1 y 2.2 a una tabla de la siguiente manera:

w	z	z_0	
I	$-M$	$-e$	q

$w \geqslant q0$, $z \geqslant q0$, $z_0 \geqslant q0$.

  Sea $-q_s = \text{maximo}\{-q_i : 1 \leqslant i \leqslant n\}$, transformamos la tabla pivotando la fila s y la columna z_0. Por lo que, las variables básicas, z_0 y w_j, con $j = 1, \ldots, n$ y $j \neq s$ no son negativas. Tomamos $y_s = z_s$ y continuamos con la etapa general.

- **Etapa general:**

  1) Sea d_s la nueva columna de la tabla actual bajo la variable y_s. Si $d_s \leqslant 0$, vamos al paso 4). En caso contrario, determinamos r mediante la prueba de razón mínima, donde $\bar{q}$ es la columna actualizada del lado derecho que indica los valores de las variables básicas:

$$\frac{\bar{q_r}}{d_{rs}} = minimo\left\{\frac{\bar{q_i}}{d_{is}} : d_{is} > 0\right\}$$

  Si en la fila r la variable básica es z_0, pasamos al paso 3). De no ser así, vamos al paso 2).

  2) En la fila r, la variable básica es w_l o z_l, para algún valor $l \neq s$. Entra a la base la variable y_s, por lo que la tabla se actualiza al pivotar la fila r y la columna y_s. Si la variable que ha abandonado la base es w_l, entonces tomamos $y_s = z_l$, por otro lado, si la variable que ha salido de la base es z_l, entonces $y_s = w_l$. Una vez hecho esto, volvemos al paso 1).

  3) En este paso y_s entra en la base, abandonando esta z_0. Para ello pivotamos la columna y_s y la fila z_0, obteniéndose finalmente una solución básica factible complementaria. Paramos el algoritmo.

  4) Acaba con terminación en rayo. Un rayo $R = \{(w, z, z_0) + \lambda d : \lambda \geqslant 0\}$ se encuentra tal que, cada punto en R satisface 2.1, 2.2 y 2.3. En este caso, (w, z, z_0) sería la solución básica factible casi complementaria obtenida en la última tabla y d es una dirección extrema del sistema, tal que en

la fila correspondiente a y_s posee un 1, $-d_s$ en las filas de las variables básicas actuales y cero en las demás.

Ejemplo 2.1 *Tenemos el problema LCP planteado:*

$$\begin{aligned} &w - Mz = q \\ &w \geqslant 0, z \geqslant q0 \\ &w_i z_i = 0 \\ &\forall i. \end{aligned}$$

donde los valores de q y M son los siguientes:

$$q = \begin{pmatrix} 3 \\ 5 \\ -9 \end{pmatrix} \quad \text{y} \quad M = \begin{pmatrix} 1 & -1 & -1 \\ -1 & 1 & -1 \\ 1 & 1 & 2 \end{pmatrix} \tag{2.4}$$

Por lo que la tabla será la siguiente:

V.B.	w_1	w_2	w_3	z_1	z_2	z_3	z_0	
w_1	1	0	0	−1	1	1	−1	3
w_2	0	1	0	1	−1	1	−1	5
w_3	0	0	1	−1	−1	−2	−1	−9

Queremos que z_0 entre en el vector de las variables básicas, por lo que tomamos el valor más negativo de q para saber cuál es la variable que debe salir para que z_0 ocupe su lugar. En este caso es w_3. Haciendo uso del pivotaje, obtenemos lo siguiente:

V.B.	w_1	w_2	w_3	z_1	z_2	z_3	z_0	
w_1	1	0	−1	0	2	3	0	12
w_2	0	1	−1	2	0	3	0	14
z_0	0	0	−1	1	1	2	1	9

Sabemos que por el pivotaje complementario, al salir la variable w_3 debe entrar su complementaria, es decir, z_3. Y haciendo uso de la prueba de relación mínima nos queda que la variable que debe salir es w_1.

V.B.	w_1	w_2	w_3	z_1	z_2	z_3	z_0		Min
w_1	1	0	−1	0	2	3	0	12	12/3
w_2	0	1	−1	2	0	3	0	14	14/3
z_0	0	0	−1	1	1	2	1	9	9/2

Por lo que en la siguiente tabla la variable a entrar será z_1 por ser w_1 la que ha salido del vector básico y al hacer la relación mínima y la regla lexicomínima, tomamos w_2 como la variable que debe salir.

V.B.	w_1	w_2	w_3	z_1	z_2	z_3	z_0		Min
z_3	1/3	0	−1/3	0	2/3	1	0	4	
w_2	−1	1	−2	2	−2	0	0	2	2/2
z_0	−2/3	0	−5/3	1	−1/3	0	1	1	1/1

Al salir w_2, entra z_2. Como observamos a continuación, finalmente la variable a salir es z_0 por lo que se obtiene:

V.B.	w_1	w_2	w_3	z_1	z_2	z_3	z_0		Min
z_3	1/3	0	−1/3	0	2/3	1	0	4	12/3
z_1	−1/2	1/2	−1	1	−1	0	0	1	
z_0	−11/6	−1/2	−2/3	0	2/3	0	1	0	0

V.B.	w_1	w_2	w_3	z_1	z_2	z_3	z_0	
z_3	3/6	1/2	1/3	0	0	1	−1	4
z_1	−3/4	−1/4	−2	1	0	0	3/2	1
z_2	−1/4	−3/4	−1	0	1	0	3/2	0

Como z_0 ha salido del vector básico entonces el algoritmo finaliza, obteniéndose una solución básica factible complementaria con $w = 0$ y $z = (1, 0, 4)$.

Ejemplo 2.2 *Si ahora planteamos un LCP dónde los datos iniciales son:*

$$q = \begin{pmatrix} -3 \\ 2 \\ -1 \end{pmatrix} \quad \text{y} \quad M = \begin{pmatrix} -1 & 2 & -1 \\ -1 & -2 & 3 \\ -2 & -1 & -1 \end{pmatrix} \tag{2.5}$$

Por lo que la tabla sería:

V.B.	w_1	w_2	w_3	z_1	z_2	z_3	z_0	
w_1	1	0	0	1	−2	1	−1	−3
w_2	0	1	0	1	2	−3	−1	2
w_3	0	0	1	2	1	1	−1	−1

Convertimos a z_0 en una variable básica, al ser -3 el valor más negativo de q_i, la variable a salir es w_1:

V.B.	w_1	w_2	w_3	z_1	z_2	z_3	z_0	
z_0	−1	0	0	−1	2	−1	1	3
w_2	−1	1	0	0	4	−2	0	5
w_3	−1	0	1	1	3	0	0	2

Al ser salir w_1, debe entrar z_1. El único candidato a salir de la base es el w_3.

V.B.	w_1	w_2	w_3	z_1	z_2	z_3	z_0	
z_0	−2	0	1	0	5	−1	1	5
w_2	−1	1	0	0	4	−2	0	5
z_1	−1	0	1	1	3	0	0	2

Al haber salido la variable w_3, le corresponde entrar a la variable z_3 pero observamos que la columna pivote no tiene elementos positivos. Por tanto, el algoritmo finaliza en rayo. No siendo posible solucionar el problema con el algoritmo.

2.1.2. Convergencia del algoritmo

A lo largo de este apartado estudiaremos algunos resultados sobre la convergencia de este algoritmo tomados de [2]. En primer lugar, veamos un lema que nos asegura que el algoritmo debe detenerse en un número finito de pasos.

Esto ocurre independientemente de que el algoritmo finalice con una solución factible básica o con terminación en rayo. Además, si M cumple ciertas condiciones, podemos asegurar que el algoritmo finaliza con una solución básica factible complementaria.

Lema 2.1 *En el sistema 2.1, 2.2 y 2.3, supongamos que toda solución básica factible casi complementaria es no degenerada, dicho de otro modo, toda variable básica es positiva. Por lo que ninguno de los puntos que genera el algoritmo de pivotaje de Lemke se repite y el algoritmo debe acabar en un número finito de pasos.*

Demostración. Tenemos una solución básica factible casi complementaria (w, z, z_0), donde w_s y z_s son no básicas. Por tanto, (w, z, z_0) tendrá, como máximo, dos soluciones básicas factibles casi complementarias adyacentes, una conseguida al introducir w_s en la base, y la otra, al introducir en la base z_s. Al suponer no degeneración, cada una de estas soluciones será diferente de (w, z, z_0).

Veamos ahora que ninguna de las soluciones básicas complementarias generadas por el algoritmo se repite.

Siendo $(w, z, z_0)_v$ el punto que se genera en una iteración v. Procediendo por reducción al absurdo, suponemos que $(w, z, z_0)_{k+\alpha} = (w, z, z_0)_k$ para ciertos k y α enteros positivos, siendo $k + \alpha$ el índice más pequeño en el que se observa una repetición.

Al haber supuesto la no degeneración del problema, $\alpha > 1$. Así mismo, por las reglas del algoritmo se tiene que $\alpha > 2$. Pero al ser $(w, z, z_0)_{k+\alpha-1}$ adyacente a $(w, z, z_0)_{k+\alpha}$, entonces también es adyacente a $(w, z, z_0)_k$. Si $k = 1$, como (w, z, z_0) posee exactamente una solución básica factible casi complementaria adyacente, $(w, z, z_0)_{k+\alpha-1} = (w, z, z_0)_{k+1}$, lo que nos llevaría a una repetición en $k + \alpha$–1, esto supondría una contradicción con nuestra suposición inicial, donde afirmábamos que esta repetición ocurre en $k + \alpha$.

En el caso de que $k \geqslant 2$, $(w, z, z_0)_{k+\alpha-1}$ es adyacente a $(w, z, z_0)_k$, por lo que es igual a $(w, z, z_0)_{k+1}$ o a $(w, z, z_0)_{k-1}$. Se produce entonces, en la iteración $(w, z, z_0)_{k+\alpha-1}$, una repetición, contradiciendo así la suposición. Por lo que los puntos que genera el algoritmo son distintos.
Como hay un número finito de soluciones básicas factibles casi complementarias y estas no se repiten, entonces el algoritmo finaliza en un número finito de pasos, ya sea con una solución básica factible complementaria o con una terminación en rayo.

Para comprobar la convergencia principal del algoritmo son necesarios los siguientes enunciados:

Lema 2.2 *Sea el sistema definido por 2.1, 2.2 y 2.3, supongamos que cada solución básica factible casi complementaria es no degenerada.*
Además, supongamos que este problema es resuelto por el algoritmo de Lemke, finalizando este con una terminación en rayo. Al finalizar el algoritmo obtenemos la solución básica factible casi complementaria $(\bar{w}, \bar{z}, \bar{z_0})$ *y la dirección extrema* $(\hat{w}, \hat{z}, \hat{z_0})$*, por lo que* $R = \{(\bar{w}, \bar{z}, \bar{z_0}) + \lambda(\hat{w}, \hat{z}, \hat{z_0}) : \lambda \geqslant 0\}$*. Y nos queda que:*

1. $(\hat{w}, \hat{z}, \hat{z_0}) \neq (0, 0, 0), \quad (\hat{w}, \hat{z}) \geqslant 0 \quad \hat{z_0} \geqslant 0$
2. $\hat{w} - M\hat{z} - e\hat{z_0} = 0$
3. $\bar{w}^t\bar{z} = \bar{w}^t\hat{z} = \hat{w}^t\bar{z} = \hat{w}^t\hat{z} = 0$
4. $\hat{z} \neq 0$
5. $\hat{z}^t M\hat{z} = -e^t\hat{z}\hat{z_0} \leqslant 0$

Demostración. Puesto que $(\hat{w}, \hat{z}, \hat{z_0})$ es una dirección extrema, entonces 1. y 2. son inmediatas. Además, sabemos que cada punto de R satisface 2.3, por lo que se cumple que $0 = (\bar{w} + \lambda\hat{w})^t(\bar{z} + \lambda\hat{z})$ para cada $\lambda \geqslant 0$. Si a esto le añadimos la no negatividad de de $\bar{w}$, $\hat{w}$, $\bar{z}$, y $\hat{z}$, entonces se tiene que:

$$\bar{w}^t\bar{z} = \bar{w}^t\hat{z} = \hat{w}^t\bar{z} = \hat{w}^t\hat{z} = 0 \tag{2.6}$$

Y por tanto se cumple 3.. Probemos ahora 4.. Demostrémoslo por reducción al absurdo.
Supongamos que $\hat{z} = 0$. Además, $\hat{z_0} > 0$ porque si no $\hat{z_0} = 0$; y por 2., obtendríamos que $\hat{w} = 0$, y llegaríamos a que se contradice 1., es decir, que $(\hat{w}, \hat{z}, \hat{z_0}) \neq (0, 0, 0)$. Tenemos entonces, que si $\hat{z} = 0$, $\hat{z_0} > 0$ y $\hat{w} = 1\hat{z_0}$. Por 2.4, tenemos que $0 = \hat{w}^t\bar{z}$. Por lo que, $e^t\bar{z} = 0$ y como $\bar{z} \geqslant 0$, entonces $\bar{z} = 0$. Todo componente de $\bar{z}$ es no básico debido a la suposición de no degeneración, además, $\bar{z_0}$ es básico y deben tenerse $n-1$ componentes básicos de $\bar{w}$. Dado que $\bar{w} - M\bar{z} - e\bar{z_0} = q$ y $\bar{z} = 0$, entonces tenemos $\bar{z_0} = \text{maximo}\{-q_i : 1 \leqslant i \leqslant n\}$. Por lo que la solución básica factible casi complementaria $(\bar{w}, \bar{z}, \bar{z_0})$ seria la solución inicial, siendo esto imposible por el lema 2.1.
Por lo tanto $\hat{z} \neq 0$. Multiplicando $\hat{w} - M\hat{z} - e\hat{z_0} = 0$ por $\hat{z}^t$ y por 2.4 sabemos que $\hat{z}^t\hat{w} = 0$, obteniéndose entonces $\hat{z}^t M\hat{z} = -\hat{z}^t e\hat{z_0} \leqslant 0$, verificando así la propiedad 5. y finalizando con esto la demostración.

Definición 2.3 *Se dice que una matriz M cuadrada de orden n es copositiva si $z^t Mz \geqslant 0 \quad \forall z \geqslant 0$. Asimismo, se dice que M es copositiva-plus si es copositiva y verifica que si $z \geqslant 0$ y $z^t Mz = 0$, entonces $(M + M^t)z = 0$.*

El siguiente teorema afirma que si la matriz M es copositiva-plus y si el LCP es consistente, entonces el algoritmo de Lemke genera una solución básica factible complementaria en un número finito de pasos.

Teorema 2.1 *Sea el sistema definido por 2.1, 2.2 y 2.3, supongamos que cada solución básica factible casi complementaria es no degenerada y además, supongamos que M es una matriz copositiva-plus. Entonces el algoritmo de pivotaje complementario finaliza en un número finito de pasos.*

Asimismo, si el sistema es consistente, el algoritmo termina con una solución básica factible para el sistema; por el contrario, si es inconsistente, el algoritmo se detiene con una terminación en rayo.

Demostración. Como ya habíamos visto anteriormente el algoritmo de pivotaje complementario finaliza en un número finito de pasos. Supongamos que acaba en una terminación en rayo, siendo la solución básica factible casi complementaria $(\bar{w}, \bar{z}, \bar{z_0})$ y $(\hat{w}, \hat{z}, \hat{z_0})$ la dirección extrema, en la tabla final. Por el lema 2.2:

$$\hat{z} \geqslant 0, \quad \hat{z} \neq 0, \quad \hat{z}^t M\hat{z} = -e^t \hat{z}\hat{z_0} \leqslant 0 \tag{2.7}$$

Al ser M una matriz copositiva-plus, entonces $\hat{z}^t M\hat{z} \geqslant 0$. Se deduce de 2.5 que $0 = \hat{z}^t M\hat{z} = -e^t \hat{z}\hat{z_0}$ y como $\hat{z} \neq 0$, $\hat{z_0} = 0$. Además, $\hat{w} - M\hat{z} - e\hat{z_0} = 0$, ya que $(\hat{w}, \hat{z}, \hat{z_0})$ es una dirección del conjunto definido por 2.1 y 2.2, por lo que nos queda:

$$\hat{w} = M\hat{z} \tag{2.8}$$

Veamos ahora que $q^t\hat{z} < 0$. Como $\hat{z}^t M\hat{z} = 0$y M es copositiva-plus, $(M + M^t)\hat{z} = 0$, por la propiedad 3. del lema 2.2 y que $\bar{w} = q + M\bar{z} + e\bar{z_0}$, tenemos lo siguiente:

$$0 = \bar{w}^t\hat{z} = (q + M\bar{z} + e\bar{z_0})^t\hat{z} = q^t\hat{z} - \bar{z}tM\hat{z} + \bar{z_0}e^t\hat{z} \tag{2.9}$$

Por 2.6 sabemos que, $M\hat{z} = \hat{w}$, y por la propiedad 3. del lema 2.2 se llega a que $\bar{z}^t M\hat{z} = 0$. Por otro lado, por 2.5 sabemos que $\bar{z_0} > 0$ y $e^t z^{>}0$; y al sustituir en 2.7, obtenemos que $q^t z^{<}0$.

Por lo que hemos demostrado así que $M\hat{z} = \hat{w} \geqslant 0$. Como $(M + M^t)\hat{z} = 0$,

entonces $M^t\hat{z} = -M\hat{z} \leqslant 0$, $-I\bar{z} \leqslant 0$, y $q^t\hat{z} < 0$. Nos queda pues , que el sistema $M^t y \leqslant 0$, $-ly \leqslant 0$, y $q^t y < 0$, tiene una solución, $y = \hat{z}$. Por lo que se deduce que $w - Mz = q$, $w \geqslant 0, z \geqslant 0$ no tiene solución.

Por otro lado, si el sistema definido es consistente, el algoritmo se debe detener con una solución factible básica complementario, ya que, de no ser así, este debe finalizar con una terminación en rayo y como demostramos anteriormente esto ocurre solo si el sistema es inconsistente.
Además si el problema es inconsistente obviamente no podría finalizar con una solución básica factible complementaria, por lo que debe finalizar con una terminación en rayo.

Corolario 2.1 *Si la matriz M tiene entradas no negativas, con elementos en la diagonal positivos, el algoritmo de Lemke se detiene en un número finito de pasos, y da como resultado una solución básica factible complementaria.*

Demostración. Por las condiciones de la matriz M, el sistema $w - Mz = q$, $(w, z) \geqslant 0$ tiene una solución. Un ejemplo de ello es elegir z lo suficientemente grande que verifique $w = Mz + q \geqslant 0$. Este resultado se deduce del teorema anterior, puesto que la matriz M es copositiva-plus.

3

Aplicación a problemas de Programación Cuadrática. Aplicación a Juegos Bimatriciales.

3.1. Programación Cuadrática

Los problemas de Programación Cuadrática son una clase especial de problemas de Programación no Lineal, donde la función a optimizar es cuadrática y las restricciones son lineales. A lo largo de esta sección veremos que las condiciones de Karush- Kuhn-Tucker (KKT) para este tipo de problemas pueden reducirse a resolver un LCP, y, por tanto, se pueden resolver aplicando el Método de Lemke.
Motivemos el estudio posterior, presentando dos ejemplos:

Ejemplo 3.1 ***Problema de la cartera***
Una empresa de inversión tiene un total de a euros y una lista de n acciones en las que invertir este dinero. Hay que determinar cuánto de este dinero se debe invertir en cada acción.
En este problema, no se puede invertir todo el dinero, por lo que el gerente de la empresa determina unos límites tanto superior (k_j) *como inferiormente* (I_j) *para la cantidad de dinero que invertirá en cada acción* j, $\forall j = 1, ..., n$. *Teniendo en cuenta que el rendimiento de cada capital varía aleatoriamente cada año, mediante el análisis de datos anteriores se ha estimado el rendimiento,* μ_j, *por euro invertido en la acción* j *anualmente. Los rendimientos de varias acciones no son mutuamente independientes, pero el análisis de los datos obtenidos en los años anteriores nos ha proporcionado la matriz de varianzas- covarianzas,* D, *para el rendimiento anual de cada acción por euro invertido.* D *es una matriz simétrica y definida positiva de orden* n. *Si la cantidad invertida en la acción* j, *es* x_j $\forall j = 1, ..., n$, *la solución, a la que se le llama "cartera", es* $x = (x_1, \ldots, x_n)^T$. *El rendimiento anual esperado es* $\sum_{j=1}^{n} \mu_j x_j$ *y la variación de ese rendimiento es* $x^T Dx$.

Esta variación es una medida de la fluctuación aleatoria del rendimiento anual, y por ello, hay que minimizarla. Como cabe esperar, a la empresa le gustaría que el rendimiento esperado fuera maximizado, una forma de conseguir que se cumplan ambos objetivos es especificando un límite inferior, μ, en el rendimiento esperado y minimizar la variación que se encuentra sujeta a esta restricción. De esta manera, se nos plantea el siguiente problema de programación cuadrática:

$$\begin{aligned} \text{Minimizar} \quad & x^T D x \\ \text{sujeto a} \quad & \sum_{j=1}^{n} \mu_j x_j \geqslant \mu \\ & \sum x_j \leqslant a \\ l_j \leqslant x_j \leqslant k_j, \quad & j = 1 \quad a \quad n \end{aligned}$$

Ejemplo 3.2 ***Regresión lineal restringida***
Un ejemplo puede ser el planteado por C. Marmolinero. En este ejemplo hablaremos de huevos y pollos. En la cría de pollos el primer paso es la eclosión que es realizada por criaderos especializados. Cuando nace un pollo este no necesita comida durante los primeros dos días, después este pollo pasa a llamarse pollito en crecimiento y es trasladado de la planta de incubación. Las pollitas deben crecer durante aproximadamente 19 semanas antes de empezar a producir huevos. Después de este periodo son llevadas a la parvada y pasan a llamarse gallinas.

Consideremos una región geográfica a la que llamaremos estado, en cada uno de ellos son registrados cada mes en las estadísticas publicadas por el gobierno estatal los pollos nacidos en criaderos. Además, los pollos recién eclosionados pueden comprarse o venderse a empresas fuera del estado, no estando disponibles estos datos. Definimos entonces los siguientes parámetros:
y_t será el número de pollitas (en millones) en crecimiento en el estado el primer día del mes t y d_t será el número (en millones) de pollitos recién eclosionados por criaderos en el estado en el mes t.
Al venir dado por las estadísticas del gobierno, d_t no es una variable sino un dato proporcionado. Las personas que se dedican a la producción de alimento para pollos están interesadas en saber las estimaciones de ambos parámetros, ya que esto les proporcionará información para la producción. Debemos tener en cuenta que no todos los pollos recién eclosionados tienen porque sobrevivir. Por otro lado, después de los 5 meses de edad pasan a ser gallinas y ya no se consideran parte de la población de pollos en crecimiento. Por lo que el modelo de regresión lineal más apropiado para y_t puede ser: $y_t = \beta_0 + \sum_{i=1}^{5} \beta_i d_{ti}$, siendo β_0 el número de pollitas en el censo que han sido importadas o exportadas desde

el estado, y β_i la proporción de pollos en el mes $t-i$ que están vivos en el mes t, con $i = 1$ a 5, como cabe esperar estos β_i deben verificar

$$0 \leqslant \beta_5 \leqslant \beta_4 \leqslant \beta_3 \leqslant \beta_2 \leqslant \beta_1 \leqslant 1. \tag{3.1}$$

Para lograr mejores estimaciones para los parámetros $\beta = (\beta_0, \beta_1, \beta_2, \beta_3, \beta_4, \beta_5)^T$ de datos anteriores, se puede utilizar el método de mínimos cuadrados. Para ello, teniendo los datos de los últimos 10 años de los valores de y_t, d_t, definimos $L_2(\beta) = \sum_t (y_t - \beta_0 - \sum_{i=1}^{5} \beta_i d_{ti})^2$. Una vez definido, según el método de mínimos cuadrados, los mejores valores de β son los que minimizan $L_2(\beta)$ sujeto a las restricciones 3.1. Este planteamiento es un problema de programación cuadrática.

$$\begin{aligned} \text{Minimizar} \quad & L_2(\beta) = \sum_t (y_t - \beta_0 - \sum_{i=1}^{5} \beta_i d_{ti})^2 \\ \text{sujeto a} \quad & 0 \leqslant \beta_5 \leqslant \beta_4 \leqslant \beta_3 \leqslant \beta_2 \leqslant \beta_1 \leqslant 1. \end{aligned}$$

El uso del método de mínimos cuadrados para la estimación de parámetro en regresión lineal es un problema bastante común en numerosas aplicaciones estadísticas y en prácticamente todas las ramas de la investigación científica. El problema de la estimación de parámetros en los problemas de regresión lineal restringida es un problema de programación cuadrática cuando las restricciones a las que se encuentra sujeto sean lineales.

Vamos a proceder durante este apartado a encontrar soluciones a los problemas cuadráticos usando para ello el Algoritmo de Lemke estudiado en el capítulo 2.

Consideramos el siguiente problema de Programación Cuadrática:

$$\begin{aligned} \text{Minimizar} \quad & Q(x) = c^T x + \frac{1}{2} x^T D x \\ \text{Sujeto a} \quad & Ax \geqslant b \\ & x \geqslant 0 \end{aligned} \tag{3.2}$$

siendo A una matriz de orden $m \times n$ y D una matriz cuadrada simétrica de orden n. Asumiremos sin pérdida de generalidad que D es una matriz simétrica, puesto que de no ser así esta podría ser reemplazada por $(D + D^T)/2$, que sería simétrica, sin alterar $Q(x)$.

Denotamos ahora a los multiplicadores de Lagrange de las restricciones $Ax \geqslant b$ y $x \geqslant 0$ como u y v, respectivamente, y al vector de variables de holgura como y. Por lo que las condiciones KKT son:

$$\begin{aligned} Ax + y &= b \\ -Dx - A^T u + v &= c \\ x^T v = 0, \quad u^T y &= 0 \\ x, y, u, v &\geqslant 0 \end{aligned}$$

Si tomamos:

$$M = \begin{pmatrix} 0 & -A \\ A^T & D \end{pmatrix} \quad q = \begin{pmatrix} b \\ c \end{pmatrix} \quad w = \begin{pmatrix} y \\ v \end{pmatrix} \quad z = \begin{pmatrix} u \\ x \end{pmatrix}$$

el anterior sistema puede reescribirse como un LCP $w - Mz = q$, con $w^t z = 0$, $(w, z) \geqslant 0$.

Entonces, podríamos aplicar el algoritmo de pivotaje complementario, introducido en el capítulo 2, para hallar un punto KKT del problema de programación cuadrática.

A continuación resolveremos un problema de programación cuadrática utilizando el Método de Lemke.

3.1.1. Ejemplo resuelto

Ejemplo 3.3 *Nos planteamos un problema de Programación Cuadrática de la forma 3.3, cuyos datos son los siguientes:*

$$D = \begin{pmatrix} -4 & 2 \\ -6 & 6 \end{pmatrix}, \quad A = (1, 0)$$
$$c = (10, 8) \quad \text{y} \quad b = 2.$$

Por lo que se forma un problema complementario lineal que podemos resolver utilizando el algoritmo de Lemke, cuyos datos son los siguientes:

$$w = \begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} u_1 \\ u_2 \\ v \end{pmatrix}, \quad z = \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} x_1 \\ x_2 \\ v \end{pmatrix}$$
$$M = \begin{pmatrix} D & -A^T \\ A & 0 \end{pmatrix} = \begin{pmatrix} -4 & 2 & -1 \\ -6 & 6 & 0 \\ 1 & 0 & 0 \end{pmatrix}, \quad q = \begin{pmatrix} c^T \\ -b \end{pmatrix} = \begin{pmatrix} 10 \\ 8 \\ -2 \end{pmatrix}.$$

La tabla que queda es la siguiente, que resolvemos con el algoritmo de Lemke:

V.B.	u_1	u_2	v	x_1	x_2	y	z_0	
u_1	1	0	0	4	−2	1	−1	10
u_2	0	1	0	6	−6	0	−1	8
v	0	0	1	−1	0	0	**−1**	−2

Entra z_0 en el lugar de v, por lo que la siguiente variable a entrar será la complemntaria de v, es decir, y.

V.B.	u_1	u_2	v	x_1	x_2	y	z_0	
u_1	1	0	−1	5	−2	**1**	0	12
u_2	0	1	−1	7	−6	0	0	10
z_0	0	0	−1	1	0	0	1	2

Al ser 1 el único valor positivo de la columna pivote, la variable u_1 será la que abandone el vector básico. Siendo, por ello, la próxima variable a entrar x_1.

V.B.	u_1	u_2	v	x_1	x_2	y	z_0		Min
y	1	0	−1	5	−2	1	0	12	12/5
u_2	0	1	−1	**7**	−6	0	0	10	10/7
z_0	0	0	−1	1	0	0	1	2	2/1

Aplicando la regla de la relación mínima, observamos que el candidato a salir es u_2 y por tanto, x_2 será la proxima variable de entrada.

V.B.	u_1	u_2	v	x_1	x_2	y	z_0		Min
y	1	−5/7	−2/7	0	16/7	1	0	34/7	34/16
x_1	0	1/7	−1/7	1	−6/7	0	0	10/7	
z_0	0	−1/7	−6/7	0	**6/7**	0	1	4/7	4/6

La variable a salir es z_0 teniendo como resultado la tabla siguiente:

V.B.	u_1	u_2	v	x_1	x_2	y	z_0	
y	1	14/42	2	0	0	1	−16/6	70/21
x_1	0	0	−1	1	0	0	7/6	14/7
x_2	0	−1/6	−1	0	1	0	7/6	4/6

Al haber salido z_0 de la base el algoritmo finaliza. Teniendo como resultado del algoritmo una solución básica factible complementaria del LCP: $w = (u_1, u_2, v) = 0$ y $z = (x_1, x_2, y) = (14/7, 4/6, 70/21) = (2, 2/3, 10/3)$. Por lo que, la solución para el problema cuadrático planteado, es $(x_1, x_2) = (2, 2/3)$.

3.1.2. Análisis de convergencia del algoritmo de pivotaje complementario para la programación cuadrática (ver [2]).

En el capítulo anterior observamos que para los problemas no degenerados el algoritmo se detenía en un número finito de pasos,finalizando con una solución básica factible complementaria o con la terminación en rayo. Además, si la matriz M era copositiva-plus y las restricciones lineales era consistentes, el algoritmo finalizaba con una solución básica factible complementaria.
El siguiente teorema nos garantiza que la matriz M es copositiva-plus bajo ciertas condiciones.

Teorema 3.1 *Sea A una matriz de orden $m \times n$, y sea D una matriz de orden $n \times n$ simétrica. Si para todo $y \geqslant 0$, $y^T Dy \geqslant 0$, entonces*

$$M = \begin{pmatrix} 0 & -A \\ A^T & D \end{pmatrix}$$

es copositiva. Si, además, $y \geqslant 0$ y $y^T Dy = 0$ implican que $Dy = 0$, entonces la matriz M es copositiva-plus.

Demostración. Veamos que M es copositiva. Sea $z^T = (x^T, y^T) \geqslant 0$, entonces:

$$z^T M z = (x^T, y^T) \begin{pmatrix} 0 & -A \\ A^T & D \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = y^T Dy \tag{3.3}$$

Como , $y^T Dy \geqslant 0$, para todo $y \geqslant 0$ entonces la matriz D es copositiva. Veamos ahora que M es copositiva- plus, para ello, suponemos $z \geqslant 0$ y $z^T Mz = 0$, y bastaría con ver que $(M + M^T)z = 0$. Pero,

$$M + M^T = \begin{pmatrix} 0 & 0 \\ 0 & 2D \end{pmatrix}$$

Por lo que,

$$(M + M^T)z = \begin{pmatrix} 0 \\ 2Dy \end{pmatrix}$$

Además tenemos que $z^t Mz = 0$, por lo que, por 3.3, $y^T Dy = 0$. Asimismo, como por hipótesis, $y \geqslant 0$ y $y^T Dy = 0$, entonces $Dy = 0$, y, por lo tanto, $(M + M^T)z = 0$, entonces M es copositiva-plus.

Corolario 3.1 *Si D es semidefinida positiva, $y^T Dy = 0$ implica que $Dy = 0$, por lo que M es copositiva-plus.*

Demostración. Bastaría con demostrar que, $y^T Dy = 0$ implica que $Dy = 0$. Tomemos $Dy = d$ y al ser D una matriz semidefinida positiva, tenemos,

$$0 \leqslant (y^T - \lambda d^T)D(y - \lambda d) = y^T Dy + \lambda^2 d^T Dd - 2\lambda \|d\|^2$$

Como , $y^T Dy = 0$, entonces basta con dividir en ambos lados de la inecuación por λ y tomando $\lambda \to 0^+$, obteniendo finalmente que $0 = d = Dy$.

Corolario 3.2 *Si la matriz D tiene entradas no negativas, entonces M es una matriz copositiva. Si, además, los elementos diagonales de D son positivos, entonces la matriz M es copositiva-plus.*

Demostración. Si $y \geqslant 0$ y $y^T Dy = 0$, entonces $y = 0$ y, por lo tanto, $Dy = 0$. Por el teorema M es copositiva-plus.

Teorema 3.2 *Sea $c^T x + \frac{1}{2}x^T Dx$ el problema a minimizar sujeto a $Ax \leqslant 0$, $x \geqslant 0$. Supongamos que la región factible es no vacía. Además, busquemos, usando el algoritmo de Lemke, una solución para el sistema KKT, $w - Mz = q$, $(w, z) \geqslant 0$, $w^T z = 0$, donde*

$$M = \begin{pmatrix} 0 & -A \\ A^T & D \end{pmatrix} \quad q = \begin{pmatrix} b \\ c \end{pmatrix} \quad w = \begin{pmatrix} y \\ v \end{pmatrix} \quad z = \begin{pmatrix} u \\ x \end{pmatrix}$$

Donde y es el vector de variables de holgura, y siendo u y v los multiplicadores de Lagrange asociados con las restricciones. Si no hay degeneración, el algoritmo finaliza en un número finito de pasos en cualquiera de las siguientes circunstancias:

1. *D es semidefinida positiva y $c = 0$.*
2. *D es semidefinida positiva .*
3. *D tiene elementos no negativos, y además los elementos de la diagonal son positivos.*

Además, si D es semidefinida positiva, entonces la terminación en rayo implica que la solución optima no tiene límites.

Demostración. Para empezar, asumiremos que $D = D^T$, ya que de no ser así D se podría reemplazar por $\frac{1}{2}(D + D^T)$. Como ya hemos visto en el lema 2.1, el algoritmo de Lemke acaba después de un número finito de iteraciones, ya sea con un punto KKT o terminación en rayo. Por los resultados anteriores si D es definida o semidefinida positiva o sus elementos son no negativos y la diagonal posee elementos positivos, entonces M es copositiva-plus.

Supongamos que ocurre la terminación en rayo. En este caso, por el teorema 2.1 sabemos que al ser M copositiva-plus, la terminación en rayo solo ocurre si el siguiente sistema carece de solución:

$$\begin{aligned} Ax + y &= b \\ -Dx - A^T u + v &= c \\ x^T v = 0, \quad u^T y &= 0 \\ x, y, u, v &\geqslant 0 \end{aligned}$$

Además sabemos que el siguiente sistema debe tener una solución (d, f)

$$Ad \leqslant 0 \tag{3.4}$$

$$A^T f - Dd \geqslant 0 \tag{3.5}$$

$$f \geqslant 0 \tag{3.6}$$

$$d \geqslant 0 \tag{3.7}$$

$$b^T f + c^T d < 0 \tag{3.8}$$

Multiplicando 3.5 por $d^T \geqslant 0$, y observando que $f \geqslant 0$ y $Ad \leqslant 0$, se deduce que

$$0 \leqslant d^T A^T f - d^T Dd \leqslant 0 - d^T Dd = -d^T Dd \tag{3.9}$$

Además, existen $\hat{x}$ y $\hat{y}$ de tal manera que $A\hat{x}+\hat{y} = b$, $(\hat{x}, \hat{y}) \geqslant 0$. Al sustituir b en 3.8, observando 3.5 y teniendo en cuenta que $(f, \hat{x}, \hat{y}) \geqslant 0$, obtenemos

$$0 > c^T d + b^T d + (\hat{y} + A\hat{x})^T f \geqslant c^T d + \hat{x} Dd \tag{3.10}$$

Supongamos que D es semidefinida positivo. Se deduce que $d^T Dd = 0$ por 3.9, además, por el corolario 1, se deduce que $Dd = 0$ y por 3.10 tenemos que $c^T d < 0$. Al ser $Ad \leqslant 0$ y $d \geqslant 0$, siendo d una dirección de la región factible, de tal manera que $\hat{x} + \lambda d$ es factible para todo $\lambda \geqslant 0$. Consideremos $f(\hat{x} + \lambda d)$, donde $f(x) = c^T x + \frac{1}{2} x^T Dx$. Por lo que, para $Dd = 0$ se tiene:

$$f(\hat{x} + \lambda d) = f(\hat{x}) + \lambda(c^T + \hat{x}^T D)d + \frac{1}{2}\lambda^2 d^T D d = f(\hat{x}) + \lambda c^T d$$

Como $c^T d < 0$, $f(\hat{x} + \lambda d)$, para λ suficientemente grande, se acerca a menos infinito, tenemos que la solución óptima no tiene límites.

Para finalizar la prueba, debemos demostrar que no es posible la terminación en rayo si ocurren las condiciones 1, 2 y 3 del teorema. Procedamos por reducción al absurdo, para ello supongamos que la terminación en rayo ocurre para cualquiera de esas tres condiciones. Tenemos, por 3.9 que $d^T D d \leqslant 0$. Si tomásemos como la condición 2 o 3, $d = 0$, esto es imposible por 3.10. Por lo que nos queda ver para la condición 1. Si se cumple para esta condición, tenemos que $Dd = 0$. Teniendo en cuenta esto, unido al supuesto de que $c = 0$, contradice 3.10.

Por lo que hemos demostrado que si la matriz D es semidefinida positiva y el algoritmo tiene una terminación en rayo la solución óptima no tiene límite. Asimismo, esta terminación en rayo es posible al cumplirse las condiciones 1., 2. o 3. del teorema, debiéndose producir en este caso un punto KKT.

3.2. Aplicación a Juegos Bimatriciales

Tenemos un juego de dos jugadores en el que cada partida el jugador 1 elige una opción entre sus m elecciones e independiente de la elección del otro jugador, el jugador 2 elige una opción entre sus N posibles elecciones.

En una jugada el jugador 1 ha tomado una opción i, mientras que el jugador 2 ha elegido la opción j, por lo que cada uno pierde una cantidad de dinero a'_{ij}, b'_{ij} respectivamente, siendo las matrices de pérdida $A' = (a'_{ij})$ y $B' = (b'_{ij})$.

Si para todo i y j, $a'_{ij} + b'_{ij} = 0$, el juego es llamado juego de suma nula. En este caso, se puede desarrollar una estrategia óptima utilizando el teorema Minimax de Von Neumann. Los juegos que no cumplen que $a'_{ij} + b'_{ij} = 0$ son conocidos como juegos bimatriciales o juegos de suma no nula. En estos es difícil definir una estrategia óptima, no obstante, se puede definir algunas estrategias de equilibro. El cálculo de estas estrategias puede transformarse en un LCP.

Supongamos que el primer jugador tiene una probabilidad x_i de elegir la opción i, por lo tanto la estrategia de este jugador vendrá definida por el vector columna $x = (x_i) \in \mathbb{R}^m$. De la misma forma, la estrategia del jugador 2 será representada por el vector $y = (y_j) \in \mathbb{R}^N$. Si cada uno de los jugadores toma estas estrategias, la pérdida esperada es $x^T A' y$ para el jugador 1 y la del segundo jugador es $x^T B' y$.

El par de estrategia $(\bar{x}, \bar{y})$ se dice que es de equilibrio si ninguno se beneficia al cambiar su propia estrategia, mientras que el otro jugador mantiene sin cambios su estrategia en el par $(\bar{x}, \bar{y})$, es decir, si

$$\bar{x}^T A' \bar{y} \leqslant x^T A' \bar{y} \text{ para cada vector de probabilidad } x \in \mathbb{R}^m$$

y

$$\bar{x}^T B' \bar{y} \leqslant \bar{x}^T B' y \text{ para cada vector de probabilidad } y \in \mathbb{R}^N.$$

Tomamos α, β números positivos arbitrarios, de tal manera que $a_{ij} = a'_{ij} + \alpha > 0$ y $b_{ij} = b'_{ij} + \beta > 0$ para todo i, j. Sean $A = (a_i j)$, $B = (b_i j)$, puesto que para todos los vectores $x \in \mathbb{R}^m$ y $y \in \mathbb{R}^N$, $x^T A' y = x^T A y - \alpha$ y $x^T B' y = x^T B y - \beta$, si $(\bar{x}, \bar{y})$ es un par de equilibrio para el juego con matrices de pérdida A',B', entonces ese par es de equilibrio para el juego con matrices de pérdida A,B y viceversa. Por lo que, sin pérdida de generalidad consideramos el juego cuyas matrices de pérdida sean A y B.

Al ser x un vector de probabilidad para todos los vectores x, la condición $\bar{x}^T A \bar{y} \leqslant x^T A \bar{y}$ es equivalente al siguiente sistema de restricciones:

$$\bar{x}^T A \bar{y} \leqslant A_{i.} \bar{y} \quad \forall i = 1, ..., m.$$

Sea $e_r \in \mathbb{R}^r$ el vector columna cuyos elementos sean 1, el sistema de restricciones anterior se puede denotar como $(\bar{x}^T A \bar{y}) e_m \leqslant A \bar{y}$.

De manera similar, para todos los vectores $y \in \mathbb{R}^N$ la condición condición $\bar{x}^T B \bar{y} \leqslant \bar{x}^T B y$ es equivalente a $(\bar{x}^T B \bar{y}) e_N \leqslant B^T \bar{x}$. Por lo que el par de estrategias $(\bar{x}, \bar{y})$ es de equilibrio para el juego con A y B como matrices de pérdida si y solo si,

$$\begin{aligned} A\bar{y} &\geqslant (\bar{x}^T A \bar{y}) e_m \\ B^T \bar{x} &\geqslant (\bar{x}^T B \bar{y}) e_N \end{aligned} \tag{3.11}$$

Al ser A y B estrictamente matrices positivas, $\bar{x}^T A \bar{y}$ y $\bar{x}^T B \bar{y}$ son números estrictamente positivos.

Tomando $\bar{\xi} = \bar{x}/(\bar{x}^T B \bar{y})$ y $\bar{\eta} = \bar{y}/(\bar{x}^T A \bar{y})$. Podemos decir entonces que la introducción de variables de holgura correspondientes a las restricciones de desigualdad, 3.11 es equivalente a

$$\begin{aligned} \begin{pmatrix} \bar{u} \\ \bar{v} \end{pmatrix} - \begin{pmatrix} 0 & A \\ B^T & 0 \end{pmatrix} \begin{pmatrix} \bar{\xi} \\ \bar{\eta} \end{pmatrix} &= \begin{pmatrix} -e_m \\ -e_N \end{pmatrix} \\ \begin{pmatrix} \bar{u} \\ \bar{v} \end{pmatrix} \geqslant 0, \quad \begin{pmatrix} \bar{\xi} \\ \bar{\eta} \end{pmatrix} \geqslant 0, \begin{pmatrix} \bar{u} \\ \bar{v} \end{pmatrix}^T \begin{pmatrix} \bar{\xi} \\ \bar{\eta} \end{pmatrix} &= 0 \end{aligned} \tag{3.12}$$

Inversamente podemos demostrar de manera sencilla que si $(\bar{u}, \bar{v}, \bar{\xi}, \bar{\eta})$ fuera solución del LCP 3.12, entonces $(\bar{x}, \bar{y})$ es un par de equilibrio, con $\bar{x} = \bar{\xi}/(\sum \bar{\xi}_i)$ e $\bar{y} = \bar{\eta}/(\sum \bar{\eta}_j)$. Por lo que verificamos que resolviendo el LCP se pueden calcular un par de estrategias de equilibrio.

Ejemplos:
El dilema del prisionero
Dos conocidos criminales son apresados. Durante la negociación con los reos, el juez los instó a confesar y declararse culpables.

Explicó que, si uno de ellos confiesa y el otro no, el primero será absuelto mientras que el segundo será condenado a 10 años de prisión. Si los dos confesaran, les esperaría a ambos una sentencia de 5 años. Por otro lado, ambos saben que no se les puede culpar del delito debido a la insuficiencia de pruebas y la poca solidez del caso. Aún así, el juez insiste en que si ambos deciden no confesar, entonces los condenará a ambos por infracciones menores a un año de cárcel para cada uno.
Tomemos entonces el número 1 como referencia para la elección de confesar y tomemos 2 para la opción de declararse inocente. Teniendo esto en cuenta, sus matrices de pérdida, tomando estas como los años a pasar en prisión, son las siguientes:

		A		B	
Elección jugador 2		1	2	1	2
	1	5	0	5	10
Elección jugador 1					
	2	10	1	0	1

En este juego los vectores de probabilidad $(\bar{x} = (1,0)^T, \bar{y} = (1,0)^T)$ proporcionan el único par de equilibrio existente en el juego, que serían los 5 años de prisión para cada uno. Si ambos jugadores coincidieran y usaran como vectores de probabilidad $(\tilde{x} = (0,1)^T, \tilde{y} = (0,1)^T)$, es decir que ninguno confesara, solo tendrían un año de prisión, que sería la mejor opción para ellos. Sin embargo, el problema de elegir $(\tilde{x}, \tilde{y})$ como opción es el hecho de que cada uno puede ganar engañando al otro.

La batalla de los sexos
Otro ejemplo de juego bimatricial es el siguiente. Una joven pareja de casados debe decidir que hacer un viernes por la tarde. El marido, que tomaremos como el jugador 2, propone ir a un combate de boxeo y su mujer, es decir, el jugador 1, plantea como opción ir a un concierto. Ambos califican en una escala del 0 al 5 el placer (ganancia) que obtienen al ir a ambas actividades: por un lado, el hombre concede 1 unidad para el concierto y 4 unidades para el combate de

boxeo; por el contrario, la mujer califica ambas actividades con 4 y 1 unidades respectivamente. Si no se ponen de acuerdo, pelearán y ninguno de los dos sale esa noche. Teniendo en cuenta que la pérdida lo tomaremos como placer negativo, las matrices de pérdida serán las siguientes:

		A		B	
Elección jugador 2		1	2	1	2
Elección jugador 1	1	-4	0	-1	0
	2	0	-1	0	-4

En este juego se puede verificar que los vectores de probabilidad ($\bar{x} = (1,0)^T, \bar{y} = (1,0)^T$) y ($\tilde{x} = (0,1)^T$, $\tilde{y} = (0,1)^T$), son ambos pares en equilibrio.

Las pérdidas son diferentes para cada uno de los pares de equilibrio, en el caso de $(\bar{x}, \bar{y})$ se verá beneficiado el jugador 1, mientras que con $(\tilde{x}, \tilde{y})$ el jugador 2 sale ganando. Por esta razón, estos pares de equilibrio son inestables. Incluso si la mujer sabe que su marido usará la estrategia $\tilde{y}$, ella puede decidir seguir con la estrategia $\bar{x}$ en vez de cambiar a la de $\tilde{x}$ esperando que esto incite a su pareja a cambiar a $\bar{y}$. Por lo que en este juego es muy difícil saber que pasará.

Otro par de equilibrio son los vectores de probabilidad ($\tilde{x} = (4/5, 1/5)^T, \tilde{y} = (1/5, 4/5)^T$), sin embargo en este problema a pesar de conocer estos pares de equilibrio esto no ha ayudado en la búsqueda de una estrategia optima.

A pesar de que la teoría de las estrategias de equilibrio posee algoritmos a través de la formulación LCP prácticamente eficientes, no se han encontrado un gran número de aplicaciones en el mundo real, debido a problemas como los que podemos ver en los ejemplos anteriores.

3.2.1. Resolución

A la hora de encontrar un par de estrategias de equilibrio en un juego bimatricial 3.12, con A y B^T matrices positivas, plantearíamos la siguiente tabla:

u	v	ξ	η	z_0	
I_m	0	0	$-A$	$-e_m$	$-e_m$
0	I_N	$-B^T$	0	$-e_N$	$-e_N$

$u \geqslant 0, v \geqslant 0, \xi \geqslant 0, \eta \geqslant 0$

Donde I_r denota la matriz identidad de orden r, para un r cualquiera. En este problema los pares complementarios son (u_i, ξ_i), $i = 1$ a m, y (v_j, η_j), $j = 1$ a N. Además, se puede verificar que, en este tipo de problemas al hacer el algoritmo, este termina en rayo después de conseguir el vector básico factible casi complementario inicial. Sin embargo, hay una variante del algoritmo que, al aplicarlo a este problema, funciona.

En el caso el algoritmo es el siguiente. Los vectores columna de las variables ξ_i, η_j son todos no positivos, pero no estrictamente negativos. Debido a su estructura, se puede construir un vector básico factible inicial de la siguiente forma. En primer lugar, tomamos ξ_1 como variable básica y $\xi_2, \ldots, \xi_m$ como no básicas. A continuación, igualamos ξ_1 a ξ_1^0, siendo este el número positivo más pequeño que cumple $0 = -e_N + (B^T)_{\cdot 1}\ xi_1^0 \geqslant 0$. En un v^0 tiene al menos una componente, v_r^0 igual a cero.

Convertimos ahora v_r, cuyo complemento es η_r, en una variable no básica y hacemos que el valor de η_r sea el valor más pequeño positivo, η_r^0, tal que cumpla $u^0 = A_{\cdot r}\eta_r^0 - e_m \geqslant 0$. Por otro lado, sabemos que u_s. tiene al menos un componente, u_s^0, igual a cero.
Si $s = 1$, el vector básico básico $(u_2, ..., u_m, v_1, ..., v_{r-1}, v_{r+1}, ..., v_N, \xi_1, \eta_r)$ sería factible complementario y la solución factible correspondiente es solución del LCP, por lo que finalizaría el método.
Si $s \neq 1$, entonces el vector $(u_1, ..., u_{s-1}, u_{s+1}, ..., u_m, v_1, ..., v_{r-1}, v_{r+1}, ..., v_N, \xi_1, \eta_r)$ es un vector básico factible que contiene ambas variables de par (u_1, ξ_1) como variables básicas. Además, las variables del par (u_s, ξ_s) son no básicas. Y por último este vector básico contiene una variable básica de cada par complementario, exceptuando a los pares (u_1, ξ_1), (u_s, ξ_s).
Al cumplir estas propiedades podemos decir que este vector básico inicial es casi complementario. Todos los vectores básicos que se obtienen en el algoritmo serán de este tipo (salvo el vector final que puede ser complementario) y contendrán a las variables del par (u_1, ξ_1) como básicas.

Cuando u_s se selecciona como variable a entrar en el vector básico, se genera un rayo. Por lo que el algoritmo empieza con ξ_s como variable entrante en el vector básico. Una vez hecho esto, el algoritmo continúa seleccionando la variable de entrada usando la regla de pivotaje complementaria.

Este algoritmo finaliza cuando alguna de las variables del par (u_1, ξ_1) entra en el vector básico. La base final es una base complementaria factible. Si se encuentra degeneración, se debe resolver mediante la regla léxico mínima.

Ejemplo 3.4 *Sea el vector de estrategia del jugador* 1, $x = (x_1, x_2)^T$ *e* $y = (y_1, y_2)^T$ *el vector estrategia del jugador* 2 *y partiendo de las siguientes matrices de juego:*

$$A = \begin{pmatrix} -8 & 0 \\ 0 & -1 \end{pmatrix} \quad \text{y} \quad B = \begin{pmatrix} -1 & 0 \\ 0 & -8 \end{pmatrix}$$

Tenemos la siguiente tabla:

V.B	u_1	u_2	v_1	v_2	ξ_1	ξ_2	η_1	η_2	z_0	
u_1	1	0	0	0	0	0	8	0	$\boxed{-1}$	-1
u_2	0	1	0	0	0	0	0	1	-1	-1
v_1	0	0	1	0	1	0	0	0	-1	-1
v_2	0	0	0	1	0	8	0	0	-1	-1

z_0 es la variable a entrar a la base, para ello, la variable que deja de ser básica es u_1. Por lo que nos queda al pivotar lo siguiente:

V.B	u_1	u_2	v_1	v_2	ξ_1	ξ_2	η_1	η_2	z_0	
z_0	-1	0	0	0	0	0	-8	0	1	1
u_2	-1	1	0	0	0	0	-8	1	0	0
v_1	-1	0	1	0	1	0	-8	0	0	0
v_2	-1	0	0	1	0	8	-8	0	0	0

Como la variable u_1 fue la que salió de la base, la variable que debe entrar ahora es xi_1

V.B	u_1	u_2	v_1	v_2	ξ_1	ξ_2	η_1	η_2	z_0	
z_0	-1	0	0	0	0	0	-8	0	1	1
u_2	-1	1	0	0	0	0	-8	1	0	0
v_1	-1	0	1	0	$\boxed{1}$	0	-8	0	0	0
v_2	-1	0	0	1	0	8	-8	0	0	0

La entra ξ_1 solo es candidata a salir la variable v_1 por lo que nos queda que:

Al salir la variable v_1 la variable que debe entrar es η_1 pero esto no es posible puesto que todos los elementos de esta columna son negativos. Por lo que

V.B	u_1	u_2	v_1	v_2	ξ_1	ξ_2	η_1	η_2	z_0	
z_0	-1	0	0	0	0	0	-8	0	1	1
u_2	-1	1	0	0	0	0	-8	1	0	0
ξ_1	-1	0	1	0	1	0	-8	0	0	0
v_2	-1	0	0	1	0	8	-8	0	0	0

el problema finaliza con una terminación en rayo. Por lo que, no se ha podido calcular para este problema ningún par de estrategias de equilibrio utilizando el algoritmo de Lemke.

4

Resolución computacional de ejemplos

4.1. Algoritmo de Lemke en Python

El algoritmo de Lemke puede implementarse en diversas plataformas y en varios lenguajes de programación. A continuación veremos algunos ejemplos resueltos con el algoritmo de resolución de Lemke implementado en phyton [7] realizado por AndyLamperski y que podemos encontrar en el Apéndice.

En esta implementación del algoritmo se necesita como parámetros de entrada la matriz cuadrada M y el vector q, tomando así el problema LCP (q, M). Una vez introducidos estos datos el código nos devuelve una línea de comando que puede ser dada de tres maneras.

- *(vector solución,* 0*, 'Solution Found')*, en este caso el algoritmo termina con una solución básica factible complementaria.

- *(None, 1, 'Secondary ray found')*, es decir, el algoritmo finaliza con una terminación en rayo.

- *(None, 2, 'Max Iterations Exceeded')*, en este caso el algoritmo no ha finalizado en el número máximo de iteraciones, 100, dado por defecto en la implementación.

Para ver como trabaja estudiaremos con el algoritmo problemas sencillos como son los ejemplos realizados a lo largo del trabajo, verificando así los resultados obtenidos, para luego pasar a resolver problemas más complejos.
Para empezar, tomamos el **Ejemplo 1.1** para que sea resuelto. Este problema se había añadido para ver como transformar un problema LP en uno LCP. Los datos de entrada al algoritmo son los siguiente:

$$q = \begin{pmatrix} 5 \\ 12 \\ 4 \\ 20 \\ 16 \end{pmatrix} \quad \text{y} \quad M = \begin{pmatrix} 0 & 0 & 0 & -4 & -2 \\ 0 & 0 & 0 & -6 & 4 \\ 0 & 0 & 0 & 2 & -4 \\ 1 & 6 & -2 & 0 & 0 \\ 2 & -4 & 4 & 0 & 0 \end{pmatrix}$$

Este problema no había sido resuelto, pero debido a que $q \geqslant 0$ la solución del problema es $(w, z) = (q, 0)$, por lo que al introducir los datos, debería devolvernos el vector $(0, 0, 0, 0, 0)$. Vemos que efectivamente no da el resultado esperado.

(array([0. , 0. , 0. , 0. , 0.]), 0 , 'Solution Found')

El **Ejemplo 2.1** tiene como matriz M y vector q:

$$q = \begin{pmatrix} 3 \\ 5 \\ -9 \end{pmatrix} \quad \text{y} \quad M = \begin{pmatrix} 1 & -1 & -1 \\ -1 & 1 & -1 \\ 1 & 1 & 2 \end{pmatrix}$$

Al introducir estos datos en el programa obtenemos:

(array([1. , 0. , 4.]), 0 , 'Solution Found')

Para el **Ejemplo 2.2** introducimos los siguientes datos:

$$q = \begin{pmatrix} -3 \\ 2 \\ -1 \end{pmatrix} \quad \text{y} \quad M = \begin{pmatrix} -1 & 2 & -1 \\ -1 & -2 & 3 \\ -2 & -1 & -1 \end{pmatrix}$$

Siendo devuelta por pantalla la siguiente sentencia:

(None, 1 , 'Secondary ray found')

Los siguientes datos son del **Ejercicio 3.3**:

$$M = \begin{pmatrix} -4 & 2 & -1 \\ -6 & 6 & 0 \\ 1 & 0 & 0 \end{pmatrix}, \quad q = \begin{pmatrix} 10 \\ 8 \\ -2 \end{pmatrix}.$$

Al ser introducidos en el algoritmo este nos devuelve:

(array([2. , 0.66666667, 3.33333333]), 0, 'Solution Found')

Observemos ahora la solución del Juego Bimatricial del **Ejercicio 3.4** cuyos datos son:

$$M = \begin{pmatrix} 0 & 0 & -8 & 0 \\ 0 & 0 & 0 & -1 \\ -1 & 0 & 0 & 0 \\ 0 & -8 & 0 & 0 \end{pmatrix}, \quad q = \begin{pmatrix} -1 \\ -1 \\ -1 \\ -1 \end{pmatrix}.$$

Al introducir estos datos en el programa obtenemos:

(None, 1 , 'Secondary ray found')

Como podemos observar todos los resultados proporcionados por la implementación coinciden con los obtenidos a lo largo del trabajo.

A continuación trabajaremos algunos problemas más complejos de carácter real.

Como primer ejemplo, veamos un caso real de Programación Lineal encontrado en [8] en el que se trata la inversión de una empresa en 7 proyectos distintos. Cada uno de los cuales tiene una estimación del VPN, la financiación necesaria en dólares y el personal que requiere (ver estos datos en [8]).
Siendo x_i la cantidad de dólares que se van a invertir en el proyecto i. Cumpliendo una serie de condiciones se obtiene el siguiente modelo de problema:

$$\begin{aligned} \text{maximizar} \quad & 8x_1 + 2x_2 + x_3 + 4x_4 + 0.5x_5 + 1.2x_6 + 3x_7 \\ \text{sujeto a} \quad & 120x_1 + 25x_2 + 15x_3 + 60x_4 + 8x_5 + 12x_6 + 20x_7 \leqslant 155 \\ & 15x_1 + 5x_2 + 6x_3 + 10x_4 + 3.2x_5 + 4x_6 + 12x_7 \leqslant 40 \\ & x_1 + x_2 + x_3 + x_4 + x_5 + x_6 + x_7 \geqslant 4 \\ & x_3 + x_6 \leqslant 1 \end{aligned}$$

Entonces tenemos que:

$$q = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 120 & 15 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 25 & 5 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 15 & 6 & -1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 60 & 10 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 8 & 3.2 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 12 & 4 & -1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 20 & 12 & -1 & 0 \\ -120 & -25 & -15 & -60 & -8 & -12 & -20 & 0 & 0 & 0 & 0 \\ -15 & -5 & -6 & -10 & -3.2 & -4 & -12 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \quad \text{y} \quad M = \begin{pmatrix} -8 \\ -2 \\ -1 \\ -4 \\ -0.5 \\ -1.2 \\ -3 \\ 155 \\ 40 \\ -4 \\ 1 \end{pmatrix}$$

Al ser introducidos en el algoritmo este nos devuelve:

(array([0. , 5.3 , 0. , 0. , 0. , 0. , 1.125, 0.045, 0.175, 0. , 0.]), 0, 'Solution Found')

Un problema interesante a resolver es la aplicación a los problemas cuadráticos reales. Uno de ellos podría ser el siguiente, planteado en [9].
Este es un problema de construcción de una cartera de acciones de Estados Unidos. Siguiendo los datos históricos [9] se obtiene el siguiente problema de Programación Cuadrática:

$$\begin{aligned} \text{minimizar} \quad & 0.02778x_1^2 + 2 \cdot 0.00387x_1x_2 + 2 \cdot 0.00021x_1x_3 \\ & +0.01112x_2^2 - 2 \cdot x_2x_3 + 0.00115x_3^2 \\ \text{sujeto a} \quad & 0.1073x_1 + 0.0737x_2 + 0.0627x_3 \geqslant R \\ & x_1 + x_2 + x_3 = 1 \\ & x_1, x_2, x_3 \geqslant 0. \end{aligned}$$

Por lo que nos queda lo siguiente:

$$M = \begin{pmatrix} 0 & 0 & 0 & 0.1073 & 0.0737 & 0.627 \\ 0 & 0 & 0 & -1 & -1 & -1 \\ 0 & 0 & 0 & 1 & 1 & 1 \\ -0.1073 & 1 & -1 & 2 \cdot 0.02778 & 2 \cdot 0.00387 & 2 \cdot 0.00021 \\ -0.0737 & 1 & -1 & 2 \cdot 0.00387 & 2 \cdot 0.01112 & -2 \cdot 0.0002 \\ -0.627 & 1 & -1 & 2 \cdot 0.00021 & -2 \cdot 0.0002 & 2 \cdot 0.00115 \end{pmatrix}, \quad q = \begin{pmatrix} -R \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}.$$

Si introducimos estos datos en el algoritmo para valores de $R = 6.5\,\%$ y $R = 10.5\,\%$ obtenemos los siguientes resultados:

Para $R = 6.5\,\%$

(array([0.00037697, 0. , 0. , 0. , 0.00310726, 0.10330649]), 0, 'Solution Found')

Para $R = 10.5\,\%$:

(array([0.00060896, 0. , 0. , 0. , 0.00501942, 0.16687972]), 0, 'Solution Found')

A

Apéndice

A.1. Algoritmo de Lemke implementado en Python

```
import numpy as np

class lemketableau:
    def __init__(self,M,q,maxIter = 100):
        n = len(q)
        self.T = np.hstack((np.eye(n),-M,-np.ones((n,1))
                                    ,q.reshape((n,1))))
        self.n = n
        self.wPos = range(n)
        self.zPos = range(n,2*n)
        self.W = 0
        self.Z = 1
        self.Y = 2
        self.Q = 3
        TbInd = np.vstack((self.W*np.ones(n,dtype=int),
                           np.arange(n,dtype=int)))
        TnbInd = np.vstack((self.Z*np.ones(n,dtype=int),
                            np.arange(n,dtype=int)))
        DriveInd = np.array([[self.Y],[0]])
        QInd = np.array([[self.Q],[0]])
        self.Tind = np.hstack((TbInd,TnbInd,DriveInd,QInd))
        self.maxIter = maxIter


    def lemkeAlgorithm(self):
        initVal = self.initialize()
```

```
        if not initVal:
            return np.zeros(self.n),0,'Solution Found'

        for k in range(self.maxIter):
            stepVal = self.step()
            if self.Tind[0,-2] == self.Y:
                # Solution Found
                z = self.extractSolution()
                return z,0,'Solution Found'
            elif not stepVal:
                return None,1,'Secondary ray found'

        return None,2,'Max Iterations Exceeded'

    def initialize(self):
        q = self.T[:,-1]
        minQ = np.min(q)
        if minQ < 0:
            ind = np.argmin(q)
            self.clearDriverColumn(ind)
            self.pivot(ind)

            return True
        else:
            return False

    def step(self):
        q = self.T[:,-1]
        a = self.T[:,-2]
        ind = np.nan
        minRatio = np.inf
        for i in range(self.n):
            if a[i] > 0:
                newRatio = q[i] / a[i]
                if newRatio < minRatio:
                    ind = i
                    minRatio = newRatio

        if minRatio < np.inf:
            self.clearDriverColumn(ind)
            self.pivot(ind)
            return True
        else:
```

```
        return False

def extractSolution(self):
    z = np.zeros(self.n)
    q = self.T[:,-1]
    for i in range(self.n):
        if self.Tind[0,i] == self.Z:
            z[self.Tind[1,i]] = q[i]
    return z
def partnerPos(self,pos):
    v,ind = self.Tind[:,pos]
    if v == self.W:
        ppos = self.zPos[ind]
    elif v == self.Z:
        ppos = self.wPos[ind]
    else:
        ppos = None
    return ppos

def pivot(self,pos):
    ppos = self.partnerPos(pos)
    if ppos is not None:
        self.swapColumns(pos,ppos)
        self.swapColumns(pos,-2)
        return True
    else:
        self.swapColumns(pos,-2)
        return False



def swapMatColumns(self,M,i,j):
    Mi = np.array(M[:,i],copy=True)
    Mj = np.array(M[:,j],copy=True)
    M[:,i] = Mj
    M[:,j] = Mi
    return M

def swapPos(self,v,ind,newPos):
    if v == self.W:
        self.wPos[ind] = newPos %(2*self.n+2)
    elif v == self.Z:
        self.zPos[ind] = newPos %(2*self.n+2)
```

```
    def swapColumns(self,i,j):
        iInd = self.Tind[:,i]
        jInd = self.Tind[:,j]

        v,ind = iInd
        self.swapPos(v,ind,j)
        v,ind = jInd
        self.swapPos(v,ind,i)

        self.Tind = self.swapMatColumns(self.Tind,i,j)
        self.T = self.swapMatColumns(self.T,i,j)


    def clearDriverColumn(self,ind):
        a = self.T[ind,-2]
        self.T[ind] /= a
        for i in range(self.n):
            if i != ind:
                b = self.T[i,-2]
                self.T[i] -= b * self.T[ind]


    def ind2str(self,indvec):
        v,pos = indvec
        if v == self.W:
            s = 'w%d' %pos
        elif v == self.Z:
            s = 'z%d' %pos
        elif v == self.Y:
            s = 'y'
        else:
            s = 'q'
        return s

    def indexStringArray(self):
        indstr = np.array([self.ind2str(indvec)
                           for indvec in self.Tind.T],dtype=object)
        return indstr

    def indexedTableau(self):
        indstr = self.indexStringArray()
```

```
        return np.vstack((indstr,self.T))
    def __repr__(self):
        IT = self.indexedTableau()
        return IT.__repr__()
    def __str__(self):
        IT = self.indexedTableau()
        return IT.__str__()

def lemkelcp(M,q,maxIter=100):
    """
    sol = lemkelcp(M,q,maxIter)
    Uses Lemke's algorithm to copute a solution to the
    linear complementarity problem:
    Mz + q >= 0
    z >= 0
    z'(Mz+q) = 0
    The inputs are given by:
    M - an nxn numpy array
    q - a length n numpy array
    maxIter - an optional number of pivot iterations.
    Set to 100 by default
    The solution is a tuple of the form:
    z,exit_code,exit_string = sol
    The entries are summaries in the table below:

    |z                | exit_code | exit_string
|
    ------------------------------------------------------------
    | solution to LCP |     0     | 'Solution Found'
|
    | None            |     1     | 'Secondary ray found'
|
    | None            |     2     | 'Max Iterations Exceeded' |
    """

    tableau = lemketableau(M,q,maxIter)
    return tableau.lemkeAlgorithm()
```

Bibliografía

[1] Adler, Verma *The Linear Complementarity Problem, Lemke Algorithm, Perturbation, and the Complexity Class*

[2] Bazaraa, Sherali, Shetty. Nonlinear Programming. *J. Wiley, 2006. PPAD.* Internet edition.

[3] Cottle, Dantzig *Complementary Pivot Theory of Mathematical Programming.* Internet edition.

[4] Liu, K. *Lemke and Howson Algorithm.* Internet edition.

[5] Murty, K.G. *Linear Complementarity problem, Linear and Nonlinear Programming.* Internet edition.

[6] Olsson, D. *The linear complementarity problem: Methods and applications.* Internet edition.

[7] https://github.com/AndyLamperski/lemkelcp.

[8] https://www.gestiondeoperaciones.net/proyectos/problema-de-seleccion-de-cartera-de-proyectos-a-traves-de-la-programacion-entera/.

[9] https://idus.us.es/xmlui/bitstream/handle/11441/40834/Ruiz

Lista de símbolos y abreviaciones

LCP	Problema complementario lineal.
LP	Problema lineal.
KKT	Condiciones de optimalidad de Karush-Kuhn-Tucker.
$(\boldsymbol{q}, \boldsymbol{M})$	Representa el LCP cuyos datos son el vector $\boldsymbol{q}$ y la matriz $\boldsymbol{M}$.
$()^{\boldsymbol{T}}$	Traspuesta, ya sea de una matriz o un vector.
$\geqslant$	Sean $\boldsymbol{x}$ e $\boldsymbol{y}$ dos vectores en $\boldsymbol{R^n}$, diremos que $\boldsymbol{x} \geqslant \boldsymbol{y}$ si $\boldsymbol{x_i} - \boldsymbol{y_i}$ es no negativo $\forall \boldsymbol{i}$, y además $\boldsymbol{x} \neq \boldsymbol{y}$.
$>$	Sean $\boldsymbol{x}$ e $\boldsymbol{y}$ dos vectores en $\boldsymbol{R^n}$, diremos que $\boldsymbol{x} > \boldsymbol{y}$ si $\boldsymbol{x_i} - \boldsymbol{y_i}$ es estrictamente positivo para todo $\boldsymbol{i}$.
$\boldsymbol{A_{i.}}$	Representa el vector formado por la fila $\boldsymbol{i}$-ésima de la matriz $\boldsymbol{A}$.
$\boldsymbol{A_{.j}}$	Representa el vector formado por la columna $\boldsymbol{j}$-ésima de la matriz $\boldsymbol{A}$.
$\boldsymbol{I}$, $\boldsymbol{I_r}$	La matriz identidad. El subíndice este indica el orden.
$\boldsymbol{e}, \boldsymbol{e_r}$	Representa un vector columna cuyos elementos son $\boldsymbol{1}$. El subíndice indica su dimensión.
$()^{\boldsymbol{r}}$	Representa el $\boldsymbol{r}$-ésimo vector de un conjunto de vectores.
$\boldsymbol{\phi}$	Simboliza el conjunto vacío, es decir sin elementos.
$\boldsymbol{Pos(A_{.1}, ..., A_{.n})}$	Sea $\boldsymbol{A_{.1}, ..., A_{.n}}$ vectores en $\boldsymbol{R^n}$entonces$\boldsymbol{Pos(A_{.1}, ..., A_{.n}) = \{y : y = \alpha_1 A_{.1} + ... + \alpha_n A_{.n}; \alpha_1 \geqslant 0, ...\alpha_n \geqslant 0\}}$ es un cono en $\boldsymbol{R^n}$.
$\boldsymbol{C(M)}$	Clases de los $\boldsymbol{2^n}$ conos complementarios de la matriz $\boldsymbol{M}$.
$\boldsymbol{minimo}\,\{\}$	Tomamos el número mínimo del conjunto de elementos que se encuentran entre las llaves.
$\boldsymbol{K(M)}$	Unión de todos los conos complementarios asociados a $\boldsymbol{M}$.

Problemas Complementarios lineales

Virginia Vargas Mesa
Facultad de Ciencias · Sección de Matemáticas
Universidad de La Laguna
alu0100893892@ull.edu.es

Abstract

IN THIS PROJECT the problems of Linear Complementary Programming are studied. The importance of this study is emphasized by the fact that, among others, Linear Programming, Quadratic Programming and Bimatrix Games problems can be modeled as problems of Linear Complementary Programming.

For its resolution, Lemke's complementary pivot algorithm is presented, studied in detail and applied to the resolution of different case studies.

1. Introduction

A LINEAR complementary problem (LCP) consists of solving a particular system of linear equations, with non-negative variables and fulfilling a condition of complementarity by specific pairs. The resolution of problems of Linear Programming, Quadratic Programming with linear restrictions and Bimatrix Games can be undertaken through a convenient LCP. Among the algorithms developed to solve LCPs we are interested in highlighting the complementary pivoting algorithm, or Lemke algorithm.
After this introduction, throughout the first chapter we talk about the general problem and some of the problems related to it, such as linear programming.
In chapter two we study Lemke's algorithm, using examples to show how it works and ending with the study of its convergence.
Then, in the next chapter, we study and apply this algorithm in the case of quadratic programming problems and bimatrix games, solving the corresponding examples. To finish the work, we use the programmed version of the algorithm in python for the resolution of examples presented in previous chapters and some practical examples of LCP.

2. Statement of the problem

THE Linear Complementary Programming problem is given in the following way: M is a square array of order n and q a vector column in $\mathbb{R}^n$, we must find $w = (w_1, ..., w_n)^T$ and $z = (z_1, ..., z_n)^T$ belonging to $\mathbb{R}^n$, that satisfy the following conditions:

$$\begin{gathered} w - Mz = q \\ w \geqslant 0, z \geqslant 0 \quad \text{y} \quad w_i z_i = 0, \quad \forall\, i \in \{1, ..., n\}. \end{gathered} \tag{1}$$

In this problem the only known data are the vector q and the array M. This serves to represent the LCP with the pair (q, M). If the dimension of the vectors and the matrix is n, then we say that the LCP is of order n and has $2n$ variables.

The problems of linear programming, quadratic programming and bimatricial games can be modeled as a LCP, transforming the fundamental problem into a system whose goal is to find w and z.

In this section, we will focus on linear programming problems and study quadratic programming problems and bimatrix games in chapter 3.

3. Lemke algorithm

THERE are several algorithms to solve the linear complementary problem, the best known method is the Lemke method, also known as complementary pivoting algorithm, since it chooses the input variable by the complementary pivoting rule.

We then proceed to describe this method for resolving LCPs, which converges under certain assumptions of non-degeneration and when M satisfies certain properties.

The method of solving the linear complementary problem, known as the complementary pivot algorithm or Lemke Algorithm, generates almost adjacent feasible basic solutions until a complementary feasible basic solution is obtained or it becomes impossible to pivot y, therefore, to the so-called ray termination.
Later it will be shown so that conditions of M this algorithm converges into a finite number of steps.

4. Quadratic programming and Bimatrix Games applications

QUADRATIC Programming problems are a special kind of non-Linear Programming problems, where the function to be optimized is quadratic and the constraints are linear. Throughout this section we will see that the Karush-Kuhn-Tucker (KKT) conditions for this type of problems can be reduced to solving an LCP, and, therefore, they can be solved by applying the Lemke Method.
On the other hand, for developing equilibrium strategies in a bimatrix game **??** having A and B^T positive matrixes, the following table would be set:

u	v	ξ	η	z_0	
I_m	0	0	$-A$	$-e_m$	$-e_m$
0	I_N	$-B^T$	0	$-e_N$	$-e_N$

$u \geqslant 0, v \geqslant 0, \xi \geqslant 0, \eta \gneqq 0$

I_r indicates the identity matrix of order r, for any r. In this problem the complementary pairs are constituted by (u_i, ξ_i), $i = 1$ in m, and (v_j, η_j), $\forall\, j \in \{1, ..., n\}$.

5. Computational Resolution

Lemke algorithm has been implemented in different ways and by using multiple programming languages: one them is Python language. We use the programmed version of the algorithm for the resolution of examples presented in previous chapters and some practical examples of LCP.

References

[1] ADLER, VERMA *The Linear Complementarity Problem, Lemke Algorithm, Perturbation, and the Complexity Class*

[2] BAZARAA, SHERALI, SHETTY. Nonlinear Programming. *J. Wiley, 2006. PPAD.* Internet edition.

[3] COTTLE, DANTZIG *Complementary Pivot Theory of Mathematical Programming.* Internet edition.

[4] LIU, K. *Lemke and Howson Algorithm.* Internet edition.

[5] MURTY, K.G. *Linear Complementarity problem, Linear and Nonlinear Programming.* Internet edition.

[6] OLSSON, D. *The linear complementarity problem: Methods and applications.* Internet edition.

[7] https://github.com/AndyLamperski/lemkelcp.

[8] https://www.gestiondeoperaciones.net/proyectos/problema-de-seleccion-de-cartera-de-proyectos-a-traves-de-la-programacion-entera/.

[9] https://idus.us.es/xmlui/bitstream/handle/11441/40834/Ruiz