UNIVERSIDAD NACIONAL DE LA MATANZA

DEPARTAMENTO DE MATEMÁTICA

Tesis: ALGORITMO DE PUNTO INTERIOR PARA LA PROGRAMACIÓN LINEAL CON AVANCE EN EL CONJUNTO CONVEXO POLIÉDRICO

Tesis presentada por Carlos Cavallero para la Licenciatura de Matemática Aplicada

---

Supervisada por Mariano De Leo

a vos que me comprendes,
a vos que me respetas,
a vos que me esperas...

Agradecimientos

Quiero agradecer a toda mi familia, a mi hijo Jeremías por estar presente en esta dialectica de la matemática, a Cecilia por todo lo que compartimos, a Mariano por hacernos disfrutar de la Matemática, a Jorge Barreto por su paciencia y comprensión.

Índice

1. Abstract

Esta investigación desarrolla un algoritmo de programación lineal de punto interior, cuya finalidad es avanzar en la región factible del problema, convergiendo hacia la solución óptima a partir de una estructura matemático computacional. La generación del mismo intenta establecer un espacio inicial para que los alumnos comprendan y desarrollen herramientas de investigación operativa.

This paper develops an interior point linear programming algorithm,with the purpose of advancing towards the feasible region of the problem, converging toward the optimal solution from a computational mathematical structure. This development aims at establishing a starting point for students to understand and develop operative research tools.

2. Introducción

Al comenzar esta investigación estabamos enfocados en profundizar nuestros conocimientos respecto a los procesos que dieron lugar al algoritmo de punto interior de Narendra Karmarkar[1]; con el objetivo de acercar a los alumnos de la carrera de Analistas de Sistemas, contenidos matemáticos complejos y desarrollos lógicos que intervienen en el algoritmo.

La elección del algoritmo de Narendra Karmarkar estuvo radicada, no solo por sus contenidos matemáticos, sino también por el contexto que llevo a su autor a elaborar el mismo. Esto funcionó como disparador de la presente investigación, en donde podrán encontrar un conjunto de paralelismos entre los conceptos aplicados por Karmarkar y el producto de esta tesis.

Para cumplir con los objetivos propuestos nos hemos planteado el desarrollo de un algoritmo sencillo, que permitiera realizar un abordaje didáctico, con el fín de presentarle a los alumnos una plataforma inicial para la comprensión de los métodos que dan solución a los problemas de programación lineal.

[1]Narendra Karmarkar, A new polynomial-time algorithm for linear programing. Combinatorica, Estados Unidos, Primera edición,1984

3. La investigación operativa

La investigación operativa hace uso de modelos matemáticos con el objetivo de que la información obtenida permita tomar decisiones significativamente mejores en comparación con aquellas que se toman con una base cualitativa.

La resolución de un modelo analítico en este área, avanzó en los últimos años apoyándose en distintas teorías matemáticas tales como: teoría de los grafos, teoría de control, programación lineal, probabilidad y estadística, teoría de juegos, teoría de colas de espera, teoría de la decisión y programación dinámica entre otras.

El aporte de esta tesis se basa por un lado en el desafío de establecer una nueva propuesta de resolución en la programación lineal y por el otro ampliar el desarrollo de la didáctica en los problemas de programación lineal, dentro de los métodos de resolución que parten de un punto interior y recorren la región factible.

4. Avance en la región factible

El Algoritmo que proponemos en esta tesis comienza en un punto interior en la región factible y a través de dos movimientos. El primero avanzando en dirección al vector normal generado por los coeficientes de la función objetivo y el segundo perpendicular al anterior. Cada uno de estos vectores avanza con un módulo menor a la distancia del punto actual a la frontera de la región factible. Para realizar el movimiento perpendicular primero debemos elegir la dirección del mismo, esto lo realizamos considerando tomar aquel punto que esta a mayor distancia del punto actual. Esta elección permite que los desplazamientos se realicen donde se genera el mayor espacio angular. Si eligieramos la otra opción los siguientes desplazamientos se encontrarían abarrotados contra la frontera aumentando significativamente la cantidad de iteraciones para su convergencia.

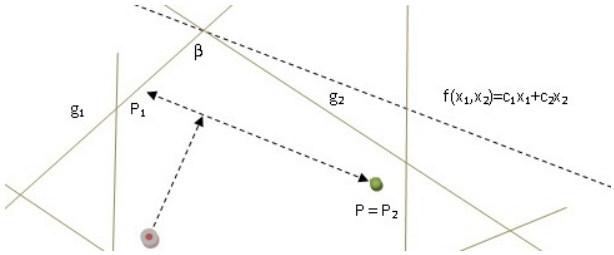


Figura 1: Avance en la región factible

Luego demostraremos que el algoritmo a partir de esta decisión converge a la solución del problema para todo conjunto convexo.

5. Principios básicos para los métodos de punto interior

Los procedimientos numéricos de punto interior para resolver problemas de programación lineal basan su estrategia en solucionar las tres cuestiones siguientes:

1. Encontrar un punto inicial de partida en el interior de la región factible del problema.

2. Definir una dirección de movimiento tal que se avance hasta un punto en el que se maximice el valor de la función objetivo.

3. Establecer una cota de crecimiento mínimo para continuar iterando.

Si proponemos un criterio o estrategia para que de un punto interior de la región factible nos acerquemos al óptimo del problema, deberíamos tener en cuenta cual es la dirección en la que tendríamos que movernos para acercarnos al valor óptimo. Esta dirección debe ser la de máxima pendiente, luego podremos concluir que se trata de un vector perpendicular a la función objetivo del problema. Teniendo en cuenta esto debemos realizar el movimiento dentro de la región factible.

6. Definiciones para la Programación Lineal

Comenzaremos definiendo todos los conceptos que debemos tener en cuenta en un problema de programación lineal.

6.1. Desigualdad lineal

La desigualdad lineal es una inecuación que puede escribirse de cuatro formas:

$$a_1x_1 + a_2x_2 + ... + a_nx_n < b$$
$$a_1x_1 + a_2x_2 + ... + a_nx_n \leqslant b$$
$$a_1x_1 + a_2x_2 + ... + a_nx_n > b$$
$$a_1x_1 + a_2x_2 + ... + a_nx_n \geqslant b$$

6.2. Semiespacios abiertos y cerrados

Los semiespacios abiertos y cerrados son los conjunto de puntos que satisface una desigualdad lineal del tipo:

6.2.1. Semiespacios cerrados

$$A = \{x = (x_1; x_2, ..., x_n) : a_1x_1 + a_2x_2 + ... + a_nx_n \leqslant b\}$$
$$A = \{x = (x_1; x_2, ..., x_n) : a_1x_1 + a_2x_2 + ... + a_nx_n \geqslant b\}$$

6.2.2. Semiespacios abiertos

$$A = \{x = (x_1; x_2, ..., x_n) : a_1x_1 + a_2x_2 + ... + a_nx_n < b\}$$
$$A = \{x = (x_1; x_2, ..., x_n) : a_1x_1 + a_2x_2 + ... + a_nx_n > b\}$$

6.3. Variables de holgura

La variable de holgura convierte a una inecuación en una igualdad

$$a_1x_1 + a_2x_2 + ... + a_nx_n \leqslant b$$
$$a_1x_1 + a_2x_2 + ... + a_nx_n + s_1 = b$$
$$a_1x_1 + a_2x_2 + ... + a_nx_n \geqslant b$$
$$a_1x_1 + a_2x_2 + ... + a_nx_n + s_2 = b$$

6.4. Hiperplano

El Hiperplano en R^n esta dado por el conjunto formado por

$$H = \{x = (x_1; x_2....x_n) : a_1x_1 + a_2x_2 + ... + a_nx_n = b\}$$

6.4.1. Hiperplano frontera

Es el Hiperplano H de los Semiespacios previos.

6.5. Punto esquina

EL Punto esquina del conjunto convexo poliédrico cumple con la siguiente proposición: un punto x en R^n se lo llama punto esquina del conjunto convexo poliédrico C, si $x \in C$ y si x es el punto de intersección de n de los hiperplanos fronteras que determinan C.

6.6. Conjunto solución de un sistema de desigualdades lineales

Es el conjunto de vectores $x = (x_1; x_2....x_n)$ en R^n cuyas componentes satisfacen las m desigualdades lineales, esta colección es un conjunto convexo poliédrico en R^n

6.7. Problema estandar en la programación lineal

Todo problema que es resuelto a partir de la programación lineal se basa en determinar el valor de $x = (x_1; x_2....x_n) \in R^n$ que maximice o minimice la función objetivo del problema y satisfaga las m desigualdades lineales correspondientes.

6.7.1. Formato básico de un P.P.L.

Determinar el vector $x = (x_1; x_2, ..., x_n)$ que maximice o minimice una función lineal
Maximice o minimice la función

$$f(x) = c_1x_1 + c_2x_2 + c3_x3..... + c_nx_n$$

sujeto a las m + n desigualdades lineales, esta función es llamada función objetivo.

$$\begin{gathered}
a_{11}x_1 + a_{12} + ... + a_{1n}x_n \leqslant b_1 \\
a_{21}x_2 + a_{22} + ... + a_{2n}x_n \leqslant b_2 \\
a_{31}x_3 + a_{32} + ... + a_{3n}x_n \leqslant b_3 \\
..... \\
a_{n1}x_n + a_{n2} + ... + a_{mn}x_n \leqslant b_n \\
x_1 \geqslant 0, x_2 \geqslant 0, x_3 \geqslant 0....x_n \geqslant 0
\end{gathered}$$

f recibe el nombre de función objetivo del problema de P.L.

6.8. Conjunto convexo poliédrico

Un conjunto S de R^n es convexo si siendo a y b elementos de S, entonces $(1-t)a+tb \in S para 0 \leqslant t \leqslant 1$.A la suma $(1-t)a+tb$ se la llama combinación convexa de a y b.

6.9. Solución factible

Cualquier punto de R^n que pertenezca al conjunto convexo poliédrico será una solución factible del problema.

6.10. Solución óptima

La solución óptima del problema de programación lineal siempre se encuentra en un punto esquina del conjunto convexo poliédrico.

6.11. Simplex

El conjunto simplex es un tipo especial de conjunto convexo poliédrico, si $P_1, P_2, P_3...., P_{n+1}$ son n+1 puntos o vectores en R^{n+1} se dice que los vectores tienen independencia afín si los n vectores $P_1P_2, P1_P3, P_1P_4.....P_1P_{n+1}$son linealmente independientes.
Si los puntos tienen independencia afín, entonces el conjunto convexo más pequeño que contiene los n+1 puntos se llama n-simplex.
Tres puntos tienen independencia afín si no son coliniales.
El conjunto convexo más pequeño que contiene tres puntos no coliniales es un triángulo con estos puntos como vértice.
Por lo tanto, un 2-simplex es un triángulo, en R^4 cuatro puntos tienen independencia afín si no son coplanares.
El conjunto convexo más pequeño que contiene cuatro de tales puntos es un tetraedro y este es un 3-simplex.

7. La función lineal

Teorema **1** *Si f es una función lineal definida sobre R^n y donde a y b son dos vectores de R^n , entonces la función lineal f toma valores entre f(a) y f(b) en el segmento de recta que une a y b.*[2]

Demostración:

Puede suponerse que $f(a) < f(b)$, entonces $(1-t)a+tb$ es un segmento de recta que une a y b, al evaluar f en $(1-t)a+tb$ se obtiene: $f((1-t)a+tb) = c_1((1-t)a_1+tb_1)+c_2((1-t)a_2+tb_2)+....+c((1-t)a_n+tb_n)$
$f((1-t)a+tb) = (1-t)(c_1a_1+c_2a_2+c_3a_3+....c_na_n)+t(c_1b_1+.....+c_nb_n)$
$f((1-t)a+tb) = (1-t)f(a)+f(b)$ *como* $f(a) \leqslant f(b)$
entonces
$f((1-t)a+tb) = (1-t)f(a)+tf(b) \leqslant (1-t)f(b)+tf(b) = f(b)$

De manera similar
$f((1-t)a+tb) = (1-t)f(a)+tf(b) \geqslant (1-t)f(a)+tf(a) = f(a)$
$f(a) \leqslant f((1-t)a+tb) \leqslant f(b)$

con lo que queda completa la demostración.

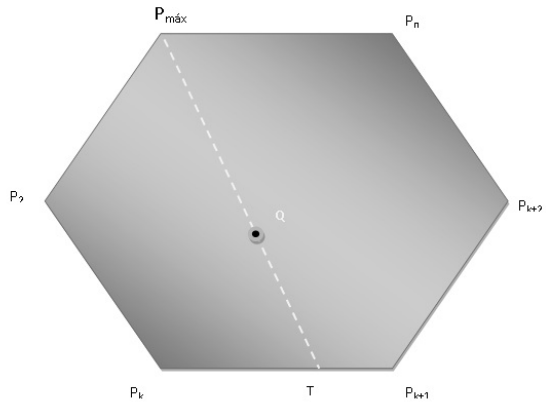
Ahora bien, como *corolario* se tiene que si f es una función lineal sobre R^n y si a y b son dos vectores de R^n tales que $f(a) = f(b)$, entonces f es constante en el segmento de recta que une a y b.

Este *corolario* nos permite observar que cuando f toma valores sobre la recta cuyo vector director es perpendicular al normal de la función objetivo, estos valores se mantienen constantes es decir $f(x) = k$ y en particular en el segmento de recta contenido entre las dos funciones lineales cuya intersección es el punto de esquina donde la función alcanza máximo, es decir donde el valor es el óptimo del P.P.L. Este último *corolario* nos será de mucha utilidad en la demostración del algoritmo que propongo en esta tesis.
El próximo teorema nos permite afirmar que la función f alcanza el máximo en un punto esquina.

Teorema **2** *La función lineal $f(X) = C.X$ definida sobre un conjunto convexo poliédrico acotado S toma sus valores máximo y mínimo en los puntos esquinas de S.*
Demostración: ésta se dará en R^2 mediante un argumento geométrico sencillo, las ideas de la demostración son similares en R^n . Un conjunto convexo

[2]Stanley I. Grossman, Aplicaciones de álgebra lineal, Mexico, Cuarta Edición, McGRAW-HILL,1992

poliédrico acotado S está representado en la figura que se muestra a continuación:



La función $f(x)$ se evalúa en estos n puntos de esquina, suponiendo que el máximo de $f(x)$ ocurre en P_1 , tenemos que $f(P_k) \leqslant f(P_{max})$ para $k = 2, 3, \ldots, n$.
Supongamos ahora que Q es cualquier punto del conjunto S, debemos demostrar que $f(Q) \leqslant f(P_{max})$ y para esto trazamos un segmento de recta $P_{max}QT$ siendo el punto de intersección de la recta que une P_{max} y Q con la frontera de S.
Supongamos que T se encuentra en el segmento de recta que une P con P_{k+1} entonces por el teorema anterior $f(T)$ se encuentra entre $f(P_k)$ y $f(P_{k+1})$ y por lo tanto $f(T) \leqslant f(P_{max})$ pero Q se encuentra en el segmento de recta que une P_{max} con T; por lo tanto $f(T) \leqslant f(Q) \leqslant f(P_{max})$

Por lo tanto $f(T) \leqslant f(Q) \leqslant f(P_{max})$ sin embargo Q puede ser cualquier punto de S, se concluye que el máximo de $f(X) = C.X$ en todo S, ocurre en un punto esquina.
La demostración para el mínimo es exactamente igual.

8. Problema dual en la programación lineal

TEOREMA **3** *Teorema fundamental de la programación lineal*
Sea f la función objetivo de un problema de máximo de programación lineal y sea g la función objetivo del correspondiente problema de mínimo dual. Entonces el problema de máximo de f tiene solución sí y solo sí el problema de mínimo de g tiene solución. Además $(x_1, x_2, x_3....., x_n)$ y $(y_1, y_2, y_3....., y_n)$ son soluciones óptimas de los dos problemas sí y solo sì f evaluada en $(x_1, x_2, x_3.....x_n)$ es igual a g evaluada en $(y_1, y_2, y_3....., y_n)$.[3]

Problemas duales expresados en forma matricial
Maximice $f(x) = c.x$ sujeto a: $A.X \leqslant b$ donde $X \geqslant 0$
$g(Y) = b.Y$ sujeto a: $A^t \geqslant c$ donde $Y \geqslant 0$

Siendo: $x : (x_1, x_2.....x_n)^T$ $Y : (y_1, y_2.....y_n)^T$ $b : (b_1, b_2.....b_n)^T$ $c : (c_1, c_2.....c_n)^T$

$$A(i,j) = \begin{pmatrix} a_{11} & a_{12} & .. & .. & a_{1n} \\ a_{21} & a_{22} & .. & .. & a_{2n} \\ a_{31} & a_{32} & .. & .. & a_{3n} \\ .. & .. & .. & .. & .. \\ a_{m1} & a_{m2} & .. & .. & a_{mn} \end{pmatrix}.$$

$$A^T = \begin{pmatrix} a_{11} & a_{21} & .. & .. & a_{m1} \\ a_{12} & a_{22} & .. & .. & a_{m2} \\ a_{13} & a_{23} & .. & .. & a_{m3} \\ .. & .. & .. & .. & .. \\ a_{1n} & a_{2n} & .. & .. & a_{mn} \end{pmatrix}.$$

Todo problema de programación lineal puede definirse a través de su dual.

8.1. Reglas para la construcción de problemas duales

Primal del problema de programación lineal

$$\begin{aligned} Minimizar : Z = c.x \\ Sujeto\ a : A.x \geqslant b \\ con\ x \geqslant 0 \end{aligned}$$

[3]Wayne L. Winston, Investigación de Operaciones, Aplicacionesv y algoritmos, Cuarta edición, Thomson, 2005

8.2. El Dual del Primal

$Maximizar : g = b^T.y$
$Sujeto\ a : A^T.c^T \leqslant c^T$
$con\ y \geqslant 0$

8.3. Minimizar f es maximizar -f

Transformamos de la siguiente forma:
$Minimizar : Z = c.x$
$Sujeto\ a : A.x \geqslant b$
$con\ x \geqslant 0$

en:
$Maximizar : -Z = -c.x$
$Sujeto\ a : -A.x \leqslant -b$
$con\ x \geqslant 0$

Este problema se ajusta a la definición sobre dualidad dada y su dual es:
$Minimizar : -g = -b^T.y$
$Sujeto\ a : -A^T.y \geqslant -c^T$
$con\ y \geqslant 0$

por lo cual nos queda:

$Maximizar : g = b^T.y$
$Sujeto\ a : A^T.y \leqslant c^T$
$con\ y \geqslant 0$

8.4. Propiedad de la función objetivo en el Primal-Dual

Teorema **4** *Si x e y son soluciones factibles de un par de problemas primal y dual, entonces se verifica:*
$z(x) = c.x \leqslant b^T y = g(y)$ *Es decir, para cualquier par de soluciones factibles de un problema primal y su dual el valor de la función objetivo del problema de máximo es menor o igual que el valor de la función objetivo del problema de mínimo.*[4]

Teorema **5** *: (Criterio de Optimalidad) Si existen soluciones factibles para los problemas primal y dual P y D tales que los valores correspondientes*

[4]Frederick S.Hillier - Gerald J.Lieberman, Investigacion de Operaciones, Séptima edición, México, McGRAW-HILL, 2001

de las funciones objetivo coinciden, entonces dichas soluciones factibles son óptimas para sus respectivos problemas.

Demostración: Sean x^* *e* y^* *factibles para los problemas Primal y Dual respectivamente, y tal que* $c.x^* = b^T y*$ *Sea* x *cualquier solución factible del problema primal* P*, se tiene que* $c.x \leqslant b^t.y^* = c.x^*$ *y por lo tanto* x^* *es una solución óptima del problema primal* P*. Un argumento análogo prueba la optimalidad de* y^* *para el problema dual.*

9. Demostración del Algoritmo

En el espacio métrico de R^2 la solución óptima de un problema de programación lineal se encuentra en unos de los vértices del conjunto convexo poliédrico, es decir en un punto esquina que se genera en la intersección de dos funciones lineales g_1 y g_2 las cuales forman un ángulo β menor a Pi por ser la región un conjunto convexo, por lo tanto a medida que nos acercamos al vértice los puntos se encuentran a menor distancia entre ellos, en este contexto podremos aproximar la solución sin sobrepasar la frontera y acercarnos cada vez más a la solución óptima como se observa en la figura 1

Veamos como describir este primer paso:
Si $f(x) = f(x_1; x_2)$ cada vez que especializamos en un punto obtenemos $f(x_1^0, x_2^0) = c_1x_1^0 + c_1x_2^0$ una constante, si los valores de c_1 y c_2 se mantienen constantes, entonces existe una combinación lineal entre los valores de $c_1x_1^i + c_2x_2^i = k_0$, ahora bien si conocemos k queda determinada una recta para las cuales se verifican los valores de $c_1x_1 + c_2x_2 = k_0$
Encontramos primeramente el vector tangente a la función f, este vector es el vector director de la recta $c_1x_1 + c_2x_2 = k_0$ *que es* $\left(1, -\dfrac{c_1}{c_2}\right)$ *o bien* $(c_2, -c_1)$ ahora bien un vector perpendicular a este debe cumplir que el producto escalar se anule verificándose: $(c_2, -c_1)(x, y) = 0$, luego $c_2x + -c_2y = 0$ de aquí que (c_1, c_2) es solución de la ecuación, luego (c_1, c_2) es perpendicular a $f(x_1, x_2) = k_0$
Ahora nos proponemos trazar rectas perpendiculares al vector normal generado por f, recordemos que el vector normal se forma a partir de los coeficientes de la función objetivo y por lo tanto tenemos determinados los valores de c_1 y c_2.
Para cada una de estas rectas y en cada una de ellas $f(x_1^k; x_2^k) = c_1x_1 + c_2x_2$ se mantiene constante es decir toma los mismos valores a lo largo del segmento de recta contenido por esta.
Como el algoritmo genera puntos a través de un desplazamiento en dirección perpendicular al vector normal de f, observamos que f se mantiene constante en este desplazamiento.
Ahora nos preguntamos por qué crece f, si el punto avanzó en dirección al vector normal de $f(X_0) = k_0$; veamos que para cada constante k tenemos una recta $j : c_1x_1 + c_2x_2 = k_J$ cuya ordenada al origen será $\frac{k_j}{c_2}$ y su vector director es $\left(1, -\dfrac{c_1}{c_2}\right)$ por lo anteriormente mencionado, luego cada una de las rectas son paralelas y como el vector (c_1, c_2) es perpendicu-

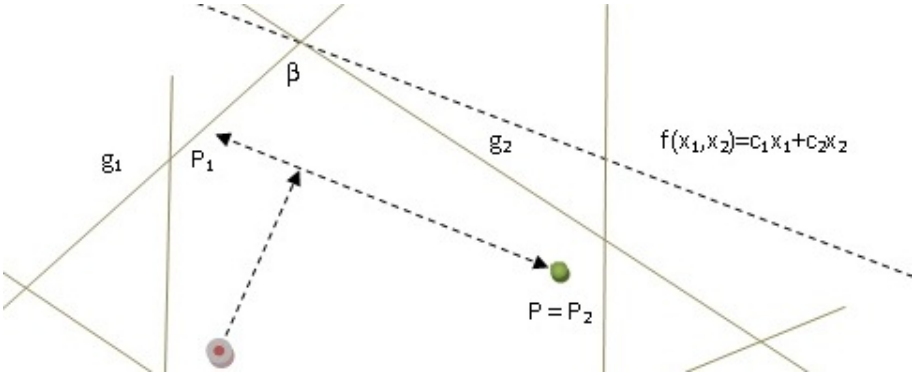


Figura 2: Avance en la región factible

lar al vector $\left(1, -\frac{c_1}{c_2}\right)$ cada vez que avanzamos con la dirección del vector (c_1, c_2) varia el valor de la ordenada al origen $\frac{k_j}{c_2}$; como c_2 es constante entonces k aumenta a medida que avanzamos en esta dirección, recordemos que se debe mantener la combinación lineal $c_1x_1 + c_2x_2 = k$ por lo tanto $f(x_1, x_2) = k_1 \leqslant f(x_1, x_2) = k_2 \leqslant ..... \leqslant f(x_1, x_2) = k_n$

El desplazamiento perpendicular al vector normal de f se realiza teniendo en cuenta la mayor distancia posible entre el punto Q alcanzado y el nuevo punto P.
Este desplazamiento se efectúa comparando las distancias desde el punto Q a los puntos encontrados a izquierda P_1 y a derecha P_2, para que el desplazamiento siempre tenga lugar en el espacio y no se obstruya, generando demasiadas iteraciones para poder desplazarse, luego avanzamos desde este nuevo punto lo máximo posible en dirección al vector normal de f para mantener el crecimiento a lo largo del algoritmo.

Estos avances se producen a partir de un factor $u = 2^n$ que multiplica escalarmente al vector unitario formado por el vector normal a f y al vector perpendicular al mismo, donde n tomará valores enteros sin sobrepasar la frontera de la región factible.
Existen dos motivos por el cual elegimos este escalar: el primero es porque la exponencial 2^n crece rápidamente cuando $n > 1$ alcanzando de este modo la máxima de las distancias posibles al punto P factible en la región evaluando de este modo la menor cantidad de puntos posibles, cuando las distancias de movimientos son menores a una unidad $n < 0$ nos asegura encontrar la distancia máxima que podemos avanzar sin sobrepasarnos de la región factible y el segundo motivo es que para cuando la distancia al punto de

convergencia es menor a la unidad la serie geométrica

$$S = \sum_{i=1}^{n} 2^{-n} \tag{1}$$

converge a 1 es decir

$$\sum_{i=1}^{n} 2^{-n} = 1 \tag{2}$$

también converge

$$\sum_{i=k}^{n} 2^{-n} = 1 - \sum_{i=1}^{k-1} 2^{-n} \tag{3}$$

y tambien lo hara toda serie formada por una subsucesión de la misma, que en nuestro caso es la suma de los módulos de los distintos desplazamientos hacia el punto óptimo.
Para cada desplazamiento perpendicular al vector normal de f , construímos cuatro sucesiones, dos cuyas imágenes están incluídas en las imágenes de g_1 y g_2 respetivamente, llamamos a estas a_k y b_k las otras dos formadas por los avances que genera el algoritmo u_k y w_k para cada punto como muestra la gráfica, la sucesión a_k se mantiene menor que u_k , y por otro lado b_k se mantiene mayor que w_k , pero como a_k y b_k son las proyecciones de los puntos que pertenecen a las dos funciones lineales que forman el vértice donde se encuentra el punto óptimo, resulta que a medida que crece f las sucesiones formadas por las proyecciones de los puntos que intervienen, tienden a acercarse cada vez más y también lo harán u_k y w_k convergiendo por estricción a la primer coordenada del punto óptimo, luego la segunda coordenada convergerá ya que quedará contenida entre $g_1(a_k)$ y $g_2(b_k)$, de este modo el algoritmo converge a P siendo P el punto óptimo del problema de programación lineal en cuestión.

Ahora bien desde el punto de vista topológico, en el espacio métrico $R \times R$ podemos encontrar siempre un abierto que incluya a P_k (el punto al cual se avanzó) tal que para cada nuevo punto P_j siempre se podrá avanzar en una de las dos direcciones propuestas para converger hacia el punto P que optimiza la función objetivo del problema.
Las longitudes de las proyecciones de los distintos puntos sobre el eje de abscisa tiende a cero a partir de que el punto esta contenido por esas dos rectas que forman el ángulo β ,ya que la máxima longitud está dada por los puntos que pertenecen a una y otra recta, pero P_k y P_{k+1} son puntos interiores al recinto formado por esas dos rectas y el segmento de recta que une el vértice anterior y posterior al óptimo, luego la longitud entre $[u_k, w_k]$ se mantiene

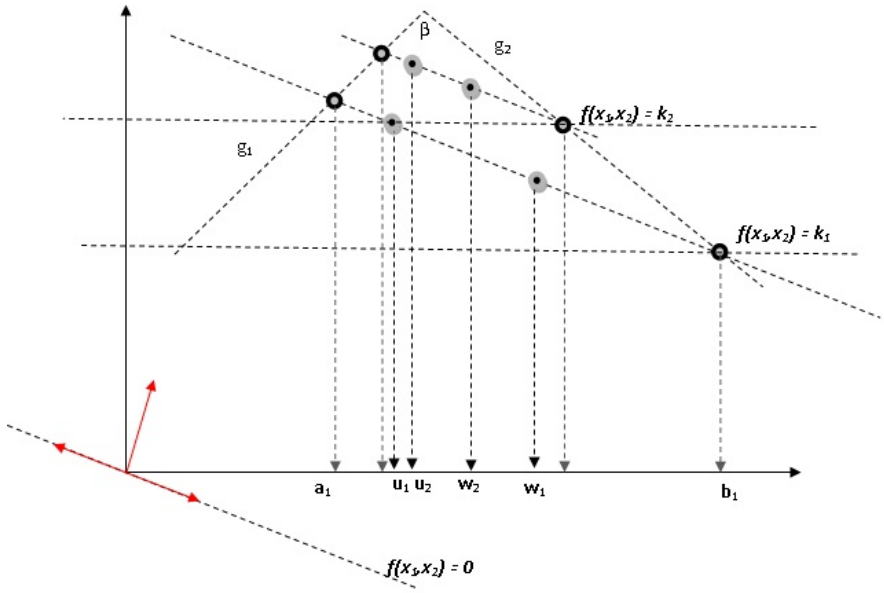


Figura 3: Construcción de sucesiones

menor que la longitud de $[a_k, b_k]$ pero $[a_k, b_k]$ cumple con la condición de encaje de intervalos cerrados donde existe $x \in [a_n, b_n] \subset [a_k, b_k] \subset [a_1, b_1]$ para todo $n \in N$.
Por lo tanto por el teorema de Bolzano – Weierstrass tenemos que la $\bigcap [a_n, b_n] = x$, pero sabemos que por construcción de los puntos que genera el algoritmo $[u_k, w_k] \subset [a_k, b_k]$ para todo valor de $k \in N$, por lo tanto x es el valor de la primera coordenada del punto óptimo al cual converge el algoritmo.
Nos queda por último ver que también converge Y_n que son las proyecciones de los puntos sobre el eje de ordenadas, si bien podríamos valernos del mismo criterio que usamos anteriormente en algunos casos las sucesiones w_k y u_k su crecimiento no es monotono, implementaremos entonces una transformación de rotación W que nos permitirá demostrar que la sucesión formada por las imágenes de las sucesiones a través de $W(u_k)$ *y* $W(w_k)$ es creciente y convergente por ende lo será la sucesión en el espacio original encontrando la segunda coordenada del punto óptimo buscado.

Para probar la convergencia de la sucesión Y_n realizaremos un cambio de variable que permita generar una transformación a partir de una rotación del punto P_k con un ángulo α *siendo* $\alpha = \dfrac{\pi}{2} - \arctan\left(-\frac{c_2}{c_1}\right)$; *donde* c_1 *y* c_2 son los coeficientes de la función objetivo, por lo tanto utilizando la matriz de transformación W obtenemos el espacio transformado que permite obser-

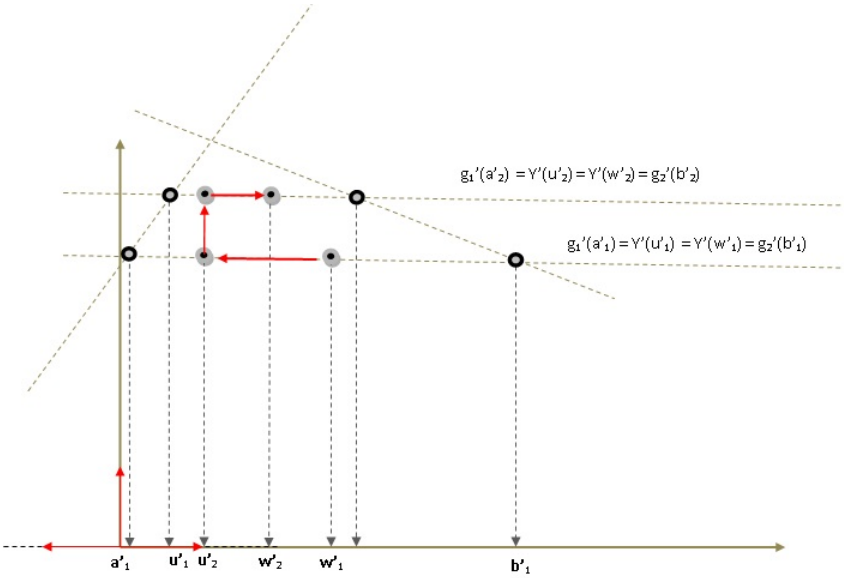


Figura 4: Avance en la región transformada

var que la sucesión es creciente en el nuevo espacio transformado, pero la transformación llevada a cabo por una rotación es una transformación lineal y mapea uno a uno los elementos de un espacio al otro, por lo tanto existe la inversa de W que nos permite volver al espacio original.

$$W(\alpha) = \begin{pmatrix} cos(\alpha) & -sin(\alpha) \\ sin(\alpha) & cos(\alpha) \end{pmatrix}.$$

$$W^{-1}(\alpha) = \begin{pmatrix} cos(\alpha) & sin(\alpha) \\ -sin(\alpha) & cos(\alpha) \end{pmatrix}.$$

En este espacio la sucesión $W(Y_n) = Y'_n$ es acotada por un valor M', ya que Y'_n se mantiene dentro del espacio transformado, debemos probar entonces que Y'_n es creciente y por lo tanto tiene límite.
Veamos entonces cuál es su límite:

Comenzaremos demostrando que las proyecciones sobre el eje de ordenadas Y'_n de cada punto transformado coinciden con las proyecciones de $g'_1(a'_k)$ y $g'_2(b'_k)$.
Demostración:
Sea W: la transformación que rota la región un ángulo $\alpha = \dfrac{\pi}{2} - \arctan\left(-\frac{c_2}{c_1}\right)$ observamos que cada punto (x_1^0, x_2^0) es transformado por W en

$$W(x_1^0; x_2^0) = \begin{pmatrix} x_1^0 cos(\alpha) - x_2^0 sin(\alpha) \\ x_1^0 sin(\alpha) + x_2^0 cos(\alpha) \end{pmatrix}.$$

Ahora nos interesa la ordenada en esta transformación, veamos que x_2^i se transforma en $x_2^i sin(\alpha) + x_2^i cos(\alpha)$

Si tenemos dos puntos P y Q que pertenecen al segmento de recta formado por el desplazamiento perpendicular a la normal, al evaluarlos: $f(P) = f(Q)$ por la citado anteriormente, vamos a demostrar que estos dos puntos transformados tienen la misma proyección en el eje de ordenadas.
Demostración:

Sea $P' = W[p_1, p_2]$ y $Q' = W[q_1, q_2]$, el punto Q podemos escribirlo en función de P y un desplazamiento de la siguiente forma: $q_1 = p_1 + \delta_x$ y $q_2 = p_2 + \delta_y$
esto nos permite ver que si transformamos estos puntos obtenemos que las ordenadas sean iguales es decir: $p_2' = q_2'$
observemos que: $W(q_1, q_2) = W(p_1 + \delta_x; p_2 - \delta_y)$ dependiendo del ángulo que forma el desplazamiento. Para demostrar la afirmación anterior observamos que al realizar la transformación

$$W(q_1, q_2) = \begin{pmatrix} (p_1 + \delta_x) cos(\alpha) - (p_2 - \delta_y) sin(\alpha) \\ (p1_+ \delta_x) sin(\alpha) + (p_2 - \delta_y) cos(\alpha) \end{pmatrix}.$$

$$W(q_1, q_2) = \begin{pmatrix} (p_1 + \delta_x) cos(\alpha) - (p_2 - \delta_y) sin(\alpha) \\ p_1 sin(\alpha) + \delta_x sin(\alpha) + p_2 cos(\alpha) - \delta_y cos(\alpha) \end{pmatrix}.$$

pero como $\alpha = \dfrac{\pi}{2} - \beta$ obtenemos que:

$$W(q_1, q_2) = \begin{pmatrix} (p_1 + \delta_x) cos(\frac{\pi}{2} - \beta) - (p_2 - \delta_y) sin(\frac{\pi}{2} - \beta) \\ p_1 sin(\frac{\pi}{2} - \beta) + \delta_x sin(\frac{\pi}{2} - \beta) + p_2 cos(\frac{\pi}{2} - \beta) - \delta_y cos(\frac{\pi}{2} - \beta) \end{pmatrix}.$$

$$W(q_1, q_2) = \begin{pmatrix} (p_1 + \delta_x) sin(\beta) - (p_2 - \delta_y) cos(\beta) \\ p_1 Cos(\beta) + \delta_x cos(\beta) + p_2 sin(\beta) - \delta_y sin(\beta) \end{pmatrix}.$$

Observamos también que $\dfrac{\delta_y}{\delta_x} = \dfrac{cos\beta}{sin(\beta)}$
de donde:

$$\delta_y = \delta_x \frac{cos(\beta}{sin(\beta)}$$

reemplazando

$$W(q_1, q_2) = \begin{pmatrix} (p_1 + \delta_x)sin(\beta) - (p_2 - \delta_y)cos(\beta) \\ p_1 cos(\beta) + \delta_x cos(\beta) + p_2 sin(\beta) - \delta_x \dfrac{cos(\beta)}{sin(\beta)} sin(\beta) \end{pmatrix}.$$

simplificando y cancelando obtenemos:

$$W(q_1, q_2) = \begin{pmatrix} (p_1 + \delta_x)sin(\beta) - (p_2 - \delta_y)cos(\beta) \\ p_1 cos(\beta) + p_2 sin(\beta) \end{pmatrix}.$$

pero como

$$W(p_1, p_2) = \begin{pmatrix} p_1 sin(\beta) - p_2 cos(\beta) \\ p_1 cos(\beta) + p_2 sin(\beta) \end{pmatrix}.$$

las proyecciones de las ordenadas son iguales luego de la transformación por W que es lo que queriamos demostrar.

Pero como P y Q pertenecen al segmento de recta entre $(a_i; g_1(a_i))y(b_i; g_2(b_i))$ aplicando la transformación W, todos los puntos pertenecen al mismo segmento de recta transformado, pero como $p'_2 = q'_2$ también será $g'_1(a'_i) = g'_2(b'_i)$.

La transformación por rotación mapea uno a uno cada uno de los puntos de la región, por lo tanto $W(g_1) = g'_1, W(g_2) = g'_2$, $g'_1 \cap g'_2 = (x_1^{'k}, x_2^{'k})$ pero como $g'_1(a'_n) = g'_2(b'_n)$ y $p'_2 n = q'_2 n$; $\forall n \in N$ en particular $n \to \infty$ las sucesiones $g'_1(a'_n)$ y $g'_2(b'_n)$ tiende al valor de la segunda ordenada en la intersección de g'_1 y g'_2 pero en la intersección $Y'(u'_n) = p_2 i = x_2^{'k} = q_2 i = Y'(w'_n)$ luego la sucesión Y'_n formada por $p'_2 i$ o $q'_2 i$ tiene límite en la intersección de g'_1 y g'_2

Cuando $n \to \infty$: $g'_1(a'_n) \to g'_2(b'_n)$ luego $|g'_1(a'_n) - g'_2(b'_n)| < \epsilon$ para todo $n > n_0$, por lo anteriormente citado $|Y'(u'_n) - Y'(w'_n)| < |g'_1(a'_n) - g'_2(b'_n)| < \epsilon$ luego la sucesión converge y tiene límite y'_n que es el límite de la sucesión transformada que genera el algoritmo, solo nos queda aplicar la inversa de la transformación y encontrar el valor de la segunda coordenada del punto óptimo buscado.
A continuación mostraremos una secuencia del algoritmo en forma gráfica.

10. Desarrollo gráfico del algoritmo

Maximizar la función objetivo $f(x_1, x_2) = x_1 + 4x_2$ sobre el conjunto de restricciones definido por:

$$\begin{gathered}
x_1 + 1{,}5x_2 \leqslant 2 \\
x_2 + 2x_2 \leqslant 3 \\
6x_2 + 2x_2 \leqslant 8 \\
-1x_2 + 4x_2 \leqslant 3 \\
0{,}25x_2 + 1x_2 \geqslant +0{,}5 \\
1x_2 + 1x_2 \geqslant +1 \\
2x_2 - 1x_2 \geqslant 0 \\
1x_1 \geqslant 0 \\
1x_2 \geqslant 0
\end{gathered}$$

Expresamos las restricciones uniformemente

$$\begin{gathered}
x_1 + 1{,}5x_2 \leqslant 2 \\
x_2 + 2x_2 \leqslant 3 \\
6x_2 + 2x_2 \leqslant 8 \\
-1x_2 + 4x_2 \leqslant 3 \\
-0{,}25x_2 - 1x_2 \leqslant -0{,}5 \\
-1x_2 - 1x_2 \leqslant -1 \\
-2x_2 + 1x_2 \leqslant 0 \\
-1x_1 \leqslant 0 \\
-1x_2 \leqslant 0
\end{gathered}$$

Determinamos las matrices de coeficientes de restricciones y costes

$$A = \begin{pmatrix} 1 & 1 \\ 1 & 1{,}5 \\ 1 & 2 \\ 6 & 2 \\ -1 & 4 \\ -0{,}25 & -1 \\ -1 & -1 \\ -2 & 1 \\ -1 & 0 \\ 0 & -1 \end{pmatrix} \; B = \begin{pmatrix} 2 \\ 3 \\ 8 \\ 3 \\ -0{,}5 \\ -1 \\ 0 \\ 0 \\ 0 \end{pmatrix} \; C = \begin{pmatrix} 1 \\ 4 \end{pmatrix} \; Error = 0,001$$

El algoritmo nos devuelve:

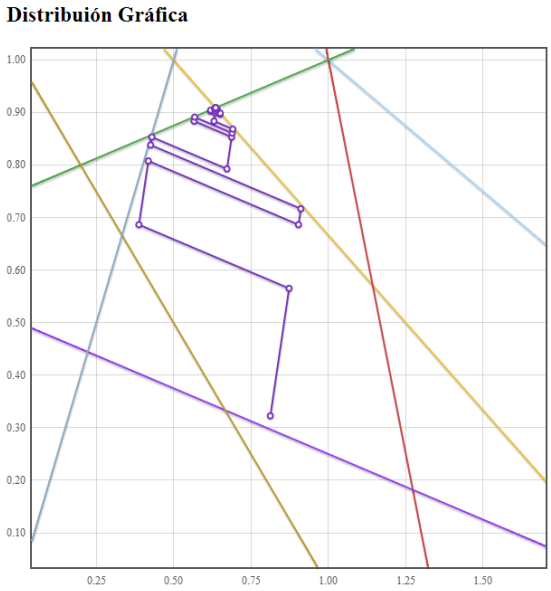


Figura 5: Convergencia con un error de 0,001

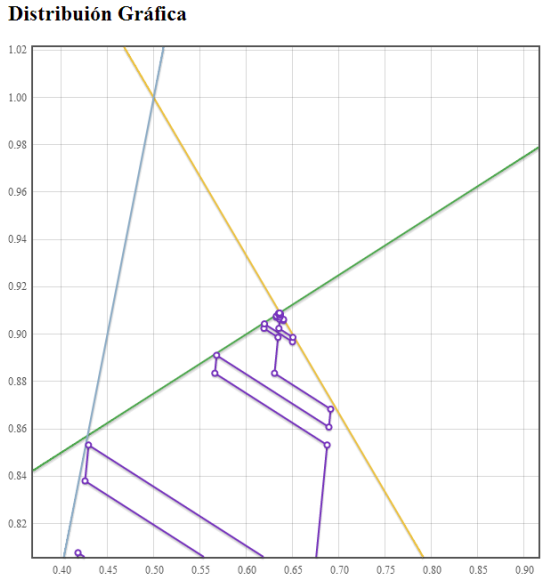


Figura 6: Desarrollo gráfico del algoritmo

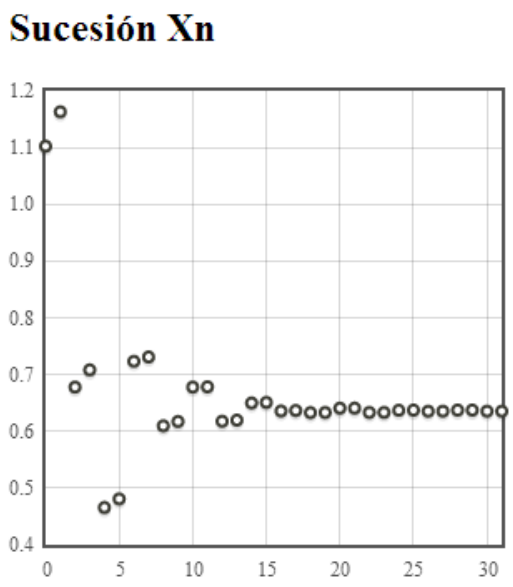


Figura 7: Sucesión sobre los valores de X

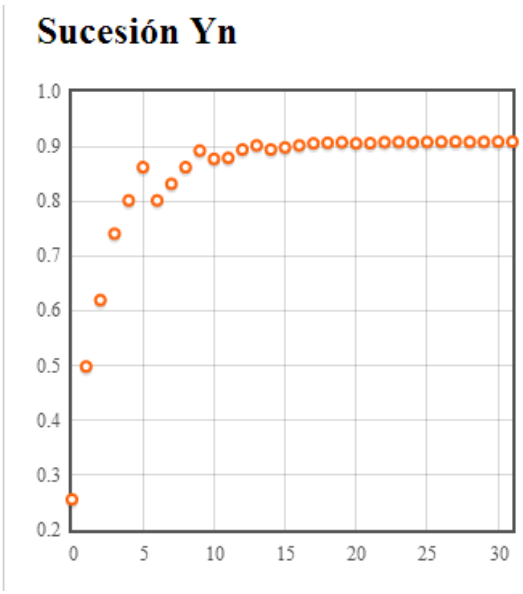


Figura 8: Sucesión sobre los valores de Y

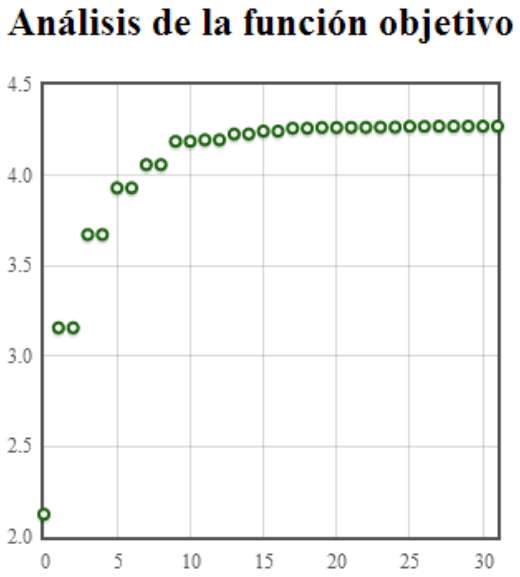


Figura 9: Sucesión sobre los valores de la Función Objetivo

11. Pseudocodigo del algoritmo

Algoritmo 1 Avance a partir de un punto Interior

Entrada: {Variables principales}
P0 : Punto inicial en la región factible
A: Matriz de coeficientes de las restricciones de cada una de las variables
B: Matriz de alcance de la restricción
CN: Vector Normal a la función objetivo
λ: Variación de la variable en cada iteración
Salida: P_k Punto Óptimo
mientras $\lambda < error$ **hacer**
  $AnteriorP_k \leftarrow Pk$
  mientras $Pk \in \{x/A.x \leqslant B\}$ **hacer**
    $AnteriorP_k \leftarrow P_k$
    $P_k \leftarrow AvanzaHaciaFuncionObjetivo\,(Pk, CN)$
    $P_k \leftarrow AvanzaTangencial(P_k)$
  fin mientras
  $\lambda \leftarrow P_k - P_kAnterior$
fin mientras
devolver P_k

Algoritmo 2 Avanza hacia funcion objetivo

Entrada: {Parametros de entrada}
P_k: Punto interior alcanzado
A: matriz de coeficientes
B: vector de limites de restriciones
VecDeAvance: vector unitario de avance
{Variables locales}
u: variable escalar que actúa sobre el vector normal a la función objetivo
n: potencia enésima
P_v: punto interior al cual se avanzó.
$AnteriorP_v, AuxP_v$: variable auxilares
Salida: P_V {Punto factible}
mientras $P_v \in \{x/A.x \leqslant B\}$ **hacer**
  AuxPv = P_v
  $u \leftarrow 2^n$
  $P_v \leftarrow P_k + u.VecDeAvance$
  $n \leftarrow n + 1$
fin mientras
si $AnteriorP_v = AuxP_v$ **entonces**
  mientras $P_v \notin \{x/A.x \leqslant B\}$ **hacer**
    $u \leftarrow 2^n$
    $P_v = P_k + u.VecDeAvance$
    $n \leftarrow n - 1$
  fin mientras
si no
  $AuxP_v \leftarrow P_v$
fin si
devolver P_v

Algoritmo 3 Avance tangencial

Entrada: {Parametros de entrada}
P_k: Punto interior alcanzado
A: matriz de coeficientes
B: vector de limites de restriciones
CN: vector normal unitario
{Variables locales}
Nulo: vector nulo
alfa: angulo $\frac{Pi}{2}$
VecTangIzq: Vector unitario a izquierda
VecTangDer: Vector unitario a derecha
Q1,P1,D1,D2: Variables auxiliares
Salida: P_k
$Q1 \leftarrow Avanza(P_k, A, B, VecTangDer)$
$P1 \leftarrow Avanza(P_k, A, B, VecTangIzq)$
$D1 \leftarrow Norma(Q1 - P_k)$
$D2 \leftarrow Norma(P1 - P_k)$
si D1 ¿D2 **entonces**
    $P_k \leftarrow Q1$
si no
    $P_k \leftarrow P1$
fin si
devolver P_k

12. Algoritmo en JavaScrip

```
<html>
<head>
<link type='text/css' rel='stylesheet' href='style.css'>
<script language='javascript' src='jquery-1.8.2.min.js'></script>
<script language='javascript' src='sylvester.js'></script>
<script language='javascript' src='flot/jquery.flot.js'></script>
<script language='javascript'
    src='flot/jquery.flot.selection.js'></script>
<script language="javascript" type="text/javascript"
    src='flot/jquery.flot.symbol.js'></script>
<script language='javascript'>

function Pertenece(Pk, A, B){
   var CantDeRestric =B.dimensions();
   var i=1;
   var EvalRest;

   while(i<=CantDeRestric)
      {EvalRest = Pk.dot(A.row(i));
      if (EvalRest > B.e(i)){i = CantDeRestric + 1}
      i++;}
   i--;
   if (i == CantDeRestric){return true} else {return false}
}
function Avanza(P, A, B, VecPerp){
   var nn = 1;
   var PV = P;
   var AnteriorPV = P;
   var AuxPV;
   var u;
   while (Pertenece(PV, A, B)){AuxPV = PV; u = Math.pow(2,nn);
      PV = P.add(VecPerp.multiply(u));nn++;}
   if(AnteriorPV == AuxPV){nn = -1;
      while(!Pertenece(PV, A, B))
         {u=Math.pow(2,nn); PV = P.add(VecPerp.multiply(u)); nn--;}
      AuxPV=PV;}
   PV=AuxPV;
   return PV;
};

function AvanceTangencial(P, A, B, CNUnit){
```

```
   var Nulo = Vector.Zero(CNUnit.dimensions());
   var alfa = Math.PI/2;
   var VecTangIzq = CNUnit.rotate(alfa,Nulo);
   var VecTangDer = CNUnit.rotate(-alfa,Nulo);
   var Q1 = Avanza(P, A, B, VecTangDer);
    var P1 = Avanza(P, A, B, VecTangIzq);
    var D1 = Q1.subtract(P).modulus();
    var D2 = P1.subtract(P).modulus();
    if ( D1 > D2 ){ P = Q1 } else { P = P1 };
    return P;
};

function AlgoritmoDeCavallero_R2(A,B,C,P0,MaxError){
   var CNUnit = C.multiply(1/C.modulus());
   var Pk = P0;
   var AnteriorPk = Vector.Zero(P0.dimensions());
   var PuntosFactibles=[];
   while( Pk.subtract(AnteriorPk).modulus() > MaxError ){
      AnteriorPk=Pk;
      Pk = Avanza(Pk, A, B, CNUnit);
      PuntosFactibles.push({x:Pk.e(1),y:Pk.e(2)});
      Pk = AvanceTangencial(Pk, A, B, CNUnit);
      PuntosFactibles.push({x:Pk.e(1),y:Pk.e(2)});
   };
   return PuntosFactibles;
}

function evaluacion_restriccion(A,B,i,x) { return -A.e(i,1) /
    A.e(i,2) * x + B.e(i) / A.e(i,2); }
function empezar() {
   var A = $M(eval($('#matriz_a').val()));
   var B = $V(eval($('#vector_b').val()));
   var C = $V(eval($('#vector_c').val()));
   var P0 = $V(eval($('#vector_P0').val()));
   var MaxError = $('#error').val();
   var PuntosFactibles;
   var PuntoOptimo;
   var y ;
   if (Pertenece(P0, A, B)){
      PuntosFactibles = AlgoritmoDeCavallero_R2(A,B,C,P0,MaxError);
   }
   else {
```

```
   alert('EL PUNTO INICIAL NO ESTA EN EL INTERIOR DEL CONJUNTO
      CONVEXO ');
}
PuntoOptimo = PuntosFactibles.pop();
$('#resultado').html('P0:
    ('+PuntoOptimo.x+','+PuntoOptimo.y+')');
var PuntosShow = '';
var convergencia = new Array();
var SucesionX = new Array();
var SucesionY = new Array();
var FuncionObjetivo = new Array();
convergencia.push([P0.e(1),P0.e(2)]);
SucesionX.push([0,P0.e(1)]);
SucesionY.push([0,P0.e(2)]);
FuncionObjetivo.push([0,P0.dot(C)]);
console.log(FuncionObjetivo);
for(var i=0;i<PuntosFactibles.length;i++) {
   var Punto = PuntosFactibles[i];
   PuntosShow += 'P'+ (i+1) + ': ('+Punto.x+','+Punto.y+'),<br>';
   convergencia.push([Punto.x,Punto.y]);
   SucesionX.push([i+1,Punto.x]);
   SucesionY.push([i+1,Punto.y]);
   FuncionObjetivo.push([i+1 ,Punto.x * C.e(1) + Punto.y *
      C.e(2)]);
}
$('#puntos_factibles').html(PuntosShow);
 var xmin=0; var xmax=2;
 Restricciones = new Array();
for(var i=1;i<=B.dimensions();i++) {
   ymin = evaluacion_restriccion(A,B,i,xmin);
   ymax = evaluacion_restriccion(A,B,i,xmax);
   Restricciones[i-1] = {data:[],fill:true};
   Restricciones[i-1].data.push([xmin,ymin]);
   Restricciones[i-1].data.push([xmax,ymax]);
}
Restricciones[i-1] = {data:convergencia};
var opciones = $.extend(true, {}, options, rangessaved);
plot = $.plot($("#grafica"), Restricciones, opciones);
overview = $.plot($("#minigrafico"), Restricciones,
    optionschico);
var OpcionSerieX = {series:
   { points: { show: true, radius: 3 } ,color: "rgb(60, 60, 52)"
      },
```

```
      grid: { hoverable: true }}
   var OpcionSerieY = {series:
      { points: { show: true, radius: 3 } ,color: "rgb(255, 100,
         12)" },
      grid: { hoverable: true }}
   var OpcionSerieF = {series:
      { points: { show: true, radius: 3 } ,color: "rgb(25, 100,
         12)" },
      grid: { hoverable: true }}
   var datosSucesionX = [{ data: SucesionX, points: { symbol:
       "circle" } }];
    $.plot($("#sucesiongraficoX"), datosSucesionX,OpcionSerieX);
   var datosSucesionY = [{ data: SucesionY, points: { symbol:
       "circle" } }];
    $.plot($("#sucesiongraficoY"), datosSucesionY, OpcionSerieY);
   var datosFuncionObjetivo = [{ data: FuncionObjetivo, points: {
       symbol: "circle" } }];
    $.plot($("#fobjetivografico"), datosFuncionObjetivo,
        OpcionSerieF);
};

var Restricciones = new Array();
var options = {
        legend: { show: false },
        grid: { hoverable: true, clickable: true },
        series: {
            lines: { show: true},
            points: { show: true }
        },
        yaxis: { ticks: 10 },
        selection: { mode: "xy" }
    };
var optionschico = {
        series: {
            lines: { show: true, lineWidth: 1 },
            shadowSize: 0
        },
        xaxis: { ticks: 4 },
        yaxis: { ticks: 3, min: -2, max: 2 },
        grid: { color: "#999" },
        selection: { mode: "xy" }
    };
```

```
var overview=null;
var plot=null;
var rangessaved=null;
var seleccion=false;

$(document).ready(function() {
   empezar();
   $("#grafica").bind("plotselected", function (event, ranges) {
        // clamp the zooming to prevent eternal zoom
        if (ranges.xaxis.to - ranges.xaxis.from < 0.000000001)
            ranges.xaxis.to = ranges.xaxis.from + 0.000000001;
        if (ranges.yaxis.to - ranges.yaxis.from < 0.000000001)
            ranges.yaxis.to = ranges.yaxis.from + 0.000000001;

        rangessaved = {
                        xaxis: { min: ranges.xaxis.from, max:
                            ranges.xaxis.to },
                        yaxis: { min: ranges.yaxis.from, max:
                            ranges.yaxis.to }
                    };
        plot = $.plot($("#grafica"), Restricciones, $.extend(true,
            {}, options, rangessaved));
        overview.setSelection(ranges, true);
    });
    $("#minigrafico").bind("plotselected", function (event, ranges)
        {
        plot.setSelection(ranges);

    });
    $("#grafica").bind("plotclick", function (event, pos, item) {
           $('#vector_P0').val('['+pos.x1+','+pos.y1+']');
           empezar();
    });
});

</script>
</head>
<body>
   <meta http-equiv="Content-Type" content="text/html;
      charset=utf-8">
   <h1>Algoritmo Cavallero</h1>
   <div id='div_parametros'>
      <title>ALGORITMO DE CAVALLERO</title>
```

```
      <h2>Par&aacute;metros</h2>
      <form>
         <b>A</b><br>
         <textarea id='matriz_a' cols='20'
            rows='10'>[[1,1.5],[1,2],[6,2],[-1,4],[-0.25,-1],[-1,-1],
            [-2,1], [-1,0],[0,-1]]
</textarea><br>
         <b>B</b><br>
         <textarea id='vector_b' cols='25'
            rows='2'>[2,3,8,3,-0.5,-1,0,0,0]
</textarea><br>
         <b>C</b><br>
         <textarea id='vector_c' cols='5' rows='5'>[1,4]
</textarea><br>
         <b>Punto interior</b><br>
         <textarea id='vector_P0' cols='38' rows='5'>[1,0.5]
</textarea><br>
         <b>Error</b><br>
         <input type='text' value='0.001' id='error'><br>
         <br>
         <input type='button' value='Ejecutar' onClick='empezar();'>
      </form>
   </div>
   <div id="div_resultado">
      <div id='minigrafico'
         style="float:right;width:300px;height:300px"></div>
      <h2> Distribui&oacute;n Gr&aacute;fica</h2>
      <div id='grafica' style="width:600px;height:600px"></div>
      <h2> Sucesi&oacute;n Xn</h2>
      <div id='sucesiongraficoX'
         style="width:300px;height:300px"></div>
      <h2> Sucesi&oacute;n Yn</h2>
      <div id='sucesiongraficoY'
         style="width:300px;height:300px"></div>
      <h2> An&aacute;lisis de la funci&oacute;n objetivo</h2>
      <div id='fobjetivografico'
         style="width:300px;height:300px"></div>
      <h2>Converge a</h2>
      <div id='resultado'><small>(Sin resultados
         calculados)</small></div>
      <h2>Sucesi&oacute;n de convergencia</h2>
      <div id='puntos_factibles'><small>(Sin resultados
         calculados)</small></div>
```

```
   </div>
</body>
</html>
```

13. Algoritmo en Mathematica

```
<< Graphics`InequalityGraphics`
 AlgoritmoCavalleroPuntoInterior[] := (
(* Prepara Ingreso de Carpetas con Problemas *)
SetDirectory["C:\Users\CharleeStone\Google
   Drive\TESIS\Algoritmo\Cavallero Mathematica\\demos\\"];
Problemas = StringReplace[FileNames["*", "."], ".\\" -> ""];
iProblemas = 1; NumGraph = 1;

(* Para cada problema *)
While [iProblemas \leqslant Length[Problemas],problema =
   Problemas[[iProblemas]]
Print["Problema: ", problema];
(* Lectura de archivos *)
A = Import[StringJoin[problema, "\\a.dat"], "Table"];
B = Import[StringJoin[problema, "\\b.dat"], "Table"];
P0 = First[Import[StringJoin[problema, "\\p0.dat"], "Table"]];
CV = First[Import[StringJoin[problema, "\\cv.dat"], "Table"]];
ESP = Import[StringJoin[problema, "\\espacio.dat"], "Table"];
ERROR = First[
First[Import[StringJoin[problema, "\\error.dat"], "Table"]]];

(* Ejecucion del algoritmo generado para encontrar la solucion
   optima *)
ListaDePuntos = AlgoritmoCavallero[A, B, CV, P0, ERROR];

MuestraResultados[A, B, CV, P0, ListaDePuntos, Inecuaciones[A, B]];

(* Preparacion de funciones e inecuaciones para mostrar en el
   grafico *)

GraphFuncionObjetivoEvaluada[ListaDePuntos, CV];
ListaDePuntosTransformados =
   TransformacionRotacion[ListaDePuntos,CV];
AnalisisDeConvergenciaGraph[ListaDePuntos];
GraficaTransformacion[ListaDePuntosTransformados];
Show[GraficaFuncionObjetivo[ListaDePuntos, CV, ESP],
   DisplayFunction -> \$DisplayFunction];
GraficaRegionFactible[GraphXX[FuncionGx[A, B]],
   Graphyy[FuncionGy[A, B]],
   GraficaInecuaciones[A,B,ESP],TRecorrido[ListaDePuntos],
GraficaFuncionObjetivo[ListaDePuntos, CV, ESP], ESP];
```

```
AnimacionListaDePuntos[ListaDePuntos,
GraficaInecuaciones[A, B, ESP],
GraphXX[FuncionGx[A, B]], ESP, iProblemas, CV];
iProblemas++;
];
);
```

```
<<Graphics`InequalityGraphics`

FuncionGx[A_,B_]:=(
i=1;Gx={};
While [i<=Length[A],
   If[A[[i,2]]\[Equal]0,Gx=Insert[Gx,B[[i]]/A[[i,1]],-1],
   If[A[[i,1]]\[Equal]0,,
   x=Insert[Gx,(-A[[i,1]]/A[[i,2]])x+(B[[i]]/A[[i,2]]),-1]]
   ];
   i++
];
Gx
);

GraficaFuncionObjetivo[ListaDePuntos_,CV_,ESP_]:=(xmin=ESP[[1,1]];
xmax=ESP[[2,1]];ymin=ESP[[1,2]];ymax=ESP[[2,2]];
x0=Last[ListaDePuntos][[1]];
y0=Last[ListaDePuntos][[2]];
Gx=-(CV[[1]]/CV[[2]])x+(CV[[1]]/CV[[2]])x0+y0;
Plot[Evaluate[Gx],{x,xmin,xmax},PlotStyle\[Rule]Hue[0.9],PlotPoints
\[Rule]25,DisplayFunction\[Rule]Identity])

FuncionGy[A_,B_]:=
(i=1;Gy={};
While [i<=Length[A],
   If[A[[i,1]]\[Equal]0,Gy=Insert[Gy,B[[i]]/A[[i,2]],-1]];
   i++;
];
Gy
);

GraficaInecuaciones[A_,B_,ESP_]:=(
xmin=ESP[[1,1]];xmax=ESP[[2,1]];
ymin=ESP[[1,2]];ymax=ESP[[2,2]];

InequalityPlot[Apply[And,Inecuaciones[A,B]],{x,xmin,xmax},
{y,ymin,ymax},Fills\[Rule]{GrayLevel[0.6]},DisplayFunction\[Rule]Identity]
);

Inecuaciones[A_,B_]:=(
i=1;GIneq ={};
While [i<=Length[A],
   If[A[[i,2]]\[Equal]0,
```

```
   GIneq = Insert[GIneq,A[[i,1]]x\leqslant First[B[[i]]],-1],
   If[A[[i,1]]\[Equal]0,GIneq=Insert[GIneq,A[[i,2]]y\leqslant
      First[B[[i]]],-1],
   GIneq = Insert[GIneq,A[[i,1]]x+A[[i,2]]y\leqslant
      First[B[[i]]],-1]]];
   i++;
];
GIneq
);

GraficaTransformacion[ListaDePuntosTransformados]:= (
Print[ " Transformaci\acute{o}n de la lista de puntos: "];
ListPlot[ListaDePuntosTransformados,PlotStyle\[Rule]PointSize[0.8],
PlotJoined\[Rule]True,ImageSize\[Rule]{500,500}];
);

GraficaRegionFactible[GraphX_,Graphy_,GraphIne_,Recorrido_,GraphFuncObj_,ESP_]:=(
Print["REGION FACTIBLE"];
xmin=ESP[[1,1]];xmax=ESP[[2,1]];ymin=ESP[[1,2]];ymax=ESP[[2,2]];
Show[{GraphX,Graphy,GraphIne,Recorrido,GraphFuncObj},
ImageSize\[Rule] {1600,900},AxesLabel\[Rule]{"Y1"},
AspectRatio\[Rule]Automatic,PlotRange\[Rule]{{xmin,xmax},{
    ymin,ymax}},
DisplayFunction\[Rule]$DisplayFunction];
);


GraphFuncionObjetivoEvaluada[ListaDePuntos_,VectNormal_]:=(
ListPlot[EvaluaFuncionObjetivo[ListaDePuntos,VectNormal],
PlotStyle\[Rule]{PointSize[0.01],Hue[0.7]},ImageSize\[Rule]{500,500}];
);

GraphXX [Gx_]:=(Plot[Evaluate[Gx],{x,xmin,xmax},
PlotStyle\[Rule]{{Hue[0.01]},{Hue[0.2]},{Hue[0.3]},{Hue[0.4]},
{Hue[0.5]},{Hue[0.65]},{Hue[0.76]},{Hue[0.8]},{Hue[0.55]}},
DisplayFunction\[Rule]Identity]
);

Graphyy[Gy_]:=(
    Plot[Evaluate[Gy],{y,ymin,ymax},PlotStyle\[Rule]{{Hue[0.1]},{Hue[0.2]},
{Hue[0.3]},{Hue[0.4]},{Hue[0.5]},{Hue[0.6]},{Hue[0.7]},{Hue[0.8]},{Hue[0.9]}},
DisplayFunction\[Rule]Identity]
);
```

```
TRecorrido[ListaDePuntos_]:=
(ListPlot[ListaDePuntos,PlotStyle\[Rule]PointSize[0.6],
PlotJoined\[Rule]True,PlotStyle\[Rule]Hue[0.9],DisplayFunction\[Rule]
    Identity]
);

EvaluaFuncionObjetivo[ListaDePuntos_,VectNormal_]:=(
EvaluaFuncion=Table[ListaDePuntos[[i,{1,2}]].
VectNormal,{i,Length[ListaDePuntos]}]
);

MuestraResultados[A_,B_,VectNormal_,PuntoInicial_,ListaDePuntos_,Inec_]:=
(Print["Restricciones: ",TableForm[Inec]];
FuncObj=VectNormal[[1]]x+VectNormal[[2]]y;
Print["Funcion Objetvio: ",FuncObj];
Print["Matriz de Coeficientes A" ,MatrixForm[A]];
Print["Matriz de Coeficientes B ",MatrixForm[B]];
Print["Matriz de Coeficientes C" ,MatrixForm[VectNormal]];
Print["Puntos Factibles (x;y): "];
Print[GridBox[N[ListaDePuntos],RowLines\[Rule]True,
ColumnLines\[Rule]True]//DisplayForm];
Print["Comportamiento de la Fumncion Objetivo :"];
Print[EvaluaFuncionObjetivo[ListaDePuntos,VectNormal]];
Print["Aproxima al Optimo: ",Last[N[ListaDePuntos]]," Con un error
    menor a: 0,001"];
Print["Con un Costo de: ",VectNormal.Last[ListaDePuntos]];
);


SucesionXn[ListaDePuntos_]:=
(    ListaYYY={};
For [i=1,i< Length[ListaDePuntos],
   ListaYYY=Insert[ListaYYY,ListaDePuntos[[i,1]],-1];
   i++
];
ListaYYY
);
SucesionYn[ListaDePuntos_]:=(
ListaYYY={};
For [i=1,i< Length[ListaDePuntos],
   ListaYYY=Insert[ListaYYY,ListaDePuntos[[i,2]],-1];
   i++
```

```
];
ListaYYY
);

AnalisisDeConvergenciaGraph[ListaDePuntos_]:=(
Print["Sucesion Xn"];
ListPlot[SucesionXn[ListaDePuntos],PlotStyle\[Rule]{PointSize[0.01],Hue[0.7]},
ImageSize\[Rule]{500,500}];
Print["Sucesion Yn"];
ListPlot[SucesionYn[ListaDePuntos],PlotStyle\[Rule]PointSize[0.01],\\
ImageSize\[Rule]{500,500}];
);

ExportaGraficaGif[ListaShowGrafica_,iProblemas_]:=(
NFileAnimation="";
NFileAnimation=StringJoin["C:\Users\CharleeStone\Desktop\GraphAnimation",
ToString[iProblemas],".gif"];
Export[NFileAnimation,ListaShowGrafica,ConversionOptions
\[Rule]{"AnimationDisplayTime"\[Rule]0.6,"Loop"\[Rule]True},\\
ImageSize\[Rule]{1600,900}];
);

AnimacionListaDePuntos[ListaDePuntos_,GraphIne_,GraphX_,Espacio_,iProblemas_,CV_]:=
(ListaShowGrafica={};xmin=Espacio[[1,1]];xmax=Espacio[[2,1]];
ymin=Espacio[[1,2]];ymax=Espacio[[2,2]];
LisIncremento={};
i=1;
GFO=GraficaFuncionObjetivo[ListaDePuntos,CV,Espacio];
Print[GFO];
LisIncremento=Insert[LisIncremento,ListaDePuntos[[i]],i];
Print[ " ANIMACI\acute{O}N DEL ALGORITMO "];
While[i<Length[ListaDePuntos],i++;
   LP=ListPlot[LisIncremento,PlotStyle\[Rule] PointSize[0.3],
   PlotJoined\[Rule]True,PlotStyle\[Rule]Hue[0.6],DisplayFunction\[Rule]
      Identity];
   LisIncremento=Insert[LisIncremento,ListaDePuntos[[i]],i];
   ListaShowGrafica=Insert[ListaShowGrafica,Show[{GraphIne,GraphX,GFO,LP},
   ImageSize\[Rule]{1600,900},PlotRange\[Rule]{{xmin,xmax},{ymin,ymax}},Axes\\\
   [Rule]True,Background
   \[Rule]RGBColor[0.10,0.2,0.10],DisplayFunction\[Rule]$DisplayFunction],-1]];
   Exporta=Input["Exporta al Browser? s/n : "];
   If [Exporta==s,ExportaGraficaGif[ListaShowGrafica,iProblemas]];
);
```

```
Transformaci\acute{o}nRotacion[ListaDePuntos_,VectNormal_]:=(
alfa=ArcTan[VectNormal[[2]]/VectNormal[[1]]];
If [alfa>0And alfa <Pi/2,alfa=-alfa+Pi/2,alfa=-alfa-Pi/2];
MT={{Cos[alfa] ,Sin[-alfa]},{Sin[alfa],Cos[alfa]}};
ListaDePuntosTransformados={};
i=1;
While[i<=Length[ListaDePuntos],
   zzx=ListaDePuntos[[i,{1,2}]].MT[[1,{1,2}]];
   zzy=ListaDePuntos[[i,{1,2}]].MT[[2,{1,2}]];
   PuntoTransformado={zzx,zzy};
   ListaDePuntosTransformados=Insert[ListaDePuntosTransformados,PuntoTransformado,i];
   i++
];
ListaDePuntosTransformados
);


Pertenece[PP_,A_,B_,CantDeRestric_]:=(
i=1;VC=PP.A[[i,{1,2}]];b={1}.B[[i]];
While[VC<=b &&
    i<=CantDeRestric,VC=PP.A[[i,{1,2}]];b={1}.B[[i]];i++];
   If[i==CantDeRestric +1 ,Bandera=True,Bandera=False];
Bandera
);

AvanzaPerpendicular[P_,A_,B_,CantDeRestric_,VectorPerpNormalDer_,VectorPerpNormalIzq_]
   Q1=AvanzaDerPerp[P,A,B,CantDeRestric,VectorPerpNormalDer];
   P1=AvanzaIzqPerp[P,A,B,CantDeRestric,VectorPerpNormalIzq];
   D1=Norm[Q1-P];
   D2= Norm[P1-P];
   If[N[D1]> N[D2],W=Q1,W=P1];
   W
);


AvanzaY[PP_,nyy_,VectNormalUnit_]:=(u=2^nyy;PV=PP +u
    VectNormalUnit ;PV);

AvanzaHaciaObjetivo[P_,A_,B_,CantDeRestric_,nyy_,VectNormalUnit_]:=(
Pk=P;
AnteriorPk=Pk;ny=nyy;
While [Pertenece[Pk,A,B,CantDeRestric] ,
```

```
   AuxPk=Pk;
   Pk=AvanzaY[AuxPk,ny,VectNormalUnit];
   ny++
];

If[AnteriorPk\[Equal]AuxPk,
ny=-1;
While [Not[Pertenece[Pk,A,B,CantDeRestric]]
    ,Pk=AvanzaY[AuxPk,ny,VectNormalUnit];ny--];
AuxPk=Pk,AuxPk=Pk];
Pk=AuxPk;
Pk
);

AlgoritmoCavallero[A_,B_,VectNormal_,PuntoInicial_,Error_]:=(
CantDeRestric=Length[B];
VectNormalUnit=(1/Norm[VectNormal])VectNormal;
alfa=Pi/2;Pendiente=VectNormal[[2]]/VectNormal[[1]];
VectorPerpNormalDer={Cos[ArcTan [Pendiente]-alfa],Sin[ArcTan
    [Pendiente]-alfa]};
VectorPerpNormalIzq={Cos[ArcTan [Pendiente]+alfa],Sin[ArcTan
    [Pendiente]+alfa]};

Pk=PuntoInicial;ListaDePuntos={Pk};
(* Comienza Iteraciones *)
ny=0;
While [Norm[AnteriorPk-Pk]>Error|| ny==0 ,ny=1;
   Pk=AvanzaHaciaObjetivo[Pk,A,B,CantDeRestric,ny,VectNormalUnit];
   ListaDePuntos=Insert[ListaDePuntos,Pk,-1];
   Pk=AvanzaPerpendicular[Pk,A,B,CantDeRestric,VectorPerpNormalDer,VectorPerpNormalIzq
   ListaDePuntos=Insert[ListaDePuntos,Pk,-1];
];
ListaDePuntos
);
```

TEOREMA **6** *Teorema general de Encaje de Intervalos*

Sea $\{C_m\}_{m\in N}$ *una sucesión de conjuntos cerrados y acotados en* R^n *, no vacios, tal que* $C_{m+1} \subseteq C_m$ *, para todo m. Entonces* $C = \bigcap C_m$ *es no vacio; que es cerrado por las propiedades de conjuntos cerrados. Si algún* C_m *es finito, la sucesión es estacionaria y la demostración es trivial. Se supondrá que todos los* C_m *contienen infinitos puntos, en cuyo caso, se puede construir un conjunto de puntos distintos* $A = \{x_1, x_2, ...x_n, ...\}$ *tomando a* x_k *de cada* C_k
Por la suposición anterior, en cada C_k *hay infinitos puntos de A. Como A es infinito y acotado, pues* $A \subseteq C_1$*, el teorema de Bolzano-Weierstrass asegura la existencia de un punto x de acumulación de A.*
Todo entorno de A incluye infinitos puntos de y por tanto incluye infinitos puntos de C_k*, lo que implica que x es también punto de acumulación de* C_k *como es cerrado y como esto ocurre para cada k,* $x \in C_k$*.*

TEOREMA **7** *Teorema de Bolzano-Weierstrass*
Todo conjunto en R^n *infinito y acotado tiene al menos un punto de acumulación.*
Demostración:
Como A es acotado podemos incluirlo en un intervalo
$I_1 = [-a, a] \times [-a, a] \times [-a, a] \times ..... \times [-a, a] = I_1 \times I_2 \times I_3 \times ....I_n$
Siendo $I_k = [-a, a]$ *para cada* $k : 1, 2, 3, 4...n$*, con* $(a > 0)$
Cada I_k *puede dividirse en dos intervalos iguales* $I_{kp1} \times I_{kp2}, \times I_{kp3}... \times I_{kpm}$ *con* $P_i = 1$ *o* 2 *para cada i.*
Se pueden construir exactamente 2^n *intervalos de este tipo. Su unión es* I^1 *que contiene infinitos puntos de A y, por lo tanto, al menos uno de estos* 2^n *intervalos contiene infinitos puntos de A. Se denomina a este intervalo* I^2 *y se renombra por* $I^2 = I_1^2 \times I_2^2 \times I_3^2 \times ..... \times I_n^2$*, donde cada* I_k^2 *es uno de los dos subintervalos de* I_k^1 *de longitud a. Realizamos sobre* I^2 *el mismo proceso de división en* 2^n *intervalos y extraemos uno de los que poseen infinitos puntos de A, al que se le llamará* $I^3 = I_1^3 \times I_2^3 \times .... \times I_n^3$
Siendo cada I_k^3 *de longitud* $\dfrac{a}{2}$
Continuando este proceso sucesivamente se obtiene una colección numerable de intervalos de $R^n, \{I^m\}_{m\in N}$ *, con la propiedad común de contener infinitos puntos de A.*
Escribiendo $I^m = I_1^m \times I_1^m \times ..... \times I_1^m$ *cada* $I_1^k = [a_k^m, b_k^m]$ *tiene longitud* $a_k^m - b_k^m = \dfrac{a}{2^{m-2}}$
La sucesión $\{I^m\}_{m+1}$ *construida constituye un encaje de intervalos cerrados*

y como

$$\lim_{n\to\infty}(a_k^m - b_k^m \in) = 0 \tag{4}$$

el principio de encaje asegura que existe un único punto x que pertenece a todos los intervalos I^m
Para probar que x es un punto de acumulación de A bastará ver que cada entorno $\varepsilon(x,\delta)$ con $\delta > 0$ contiene algún intervalo I^m, pues de este modo $\varepsilon(x,\delta) \cap A \neq \emptyset$ ya que I^m contiene infinitos puntos de A. Para que $I^m \subseteq \varepsilon(x,\delta)$ es suficiente que la diagonal d del intervalo sea menor que el radio del entorno. Como la longitud de la diagonal es

$$d = \sqrt{\sum_{i=1}^{n}(a^{k,m} - b^{k,m})} = \sqrt{\frac{a}{2^{m-2}}} \tag{5}$$

tomando m suficientemente grande, se tiene que $d < \delta$.

Referencias

[1] Narendra Karmarkar, A new polynomial-time algorithm for linear programing. Combinatoria, 4:373-395, 1990.

[2] Robert Faure, Jean-Paul Boss, Andre Le Garff, La Investigación Operativa, Quinta edición, EUDEBA S.E.M. U.B.A., Buenos Aires, 1978.

[3] Stanley I. Grossman, Aplicaciones de álgebra lineal, México, Cuarta Edición, McGRAW-HILL, 1992.

[4] Frederick S.Hillier - Gerald J.Lieberman, Investigación de Operaciones, Séptima edición, México, McGRAW-HILL, 2001.

[5] R.J. Vanderbei, M. S. Meketon, and B.A.Freedman, A modification of Karmarkar's linear programming algorithm. Primera edición, Algorithmica, 1986.

[6] Y. Ye, Karmarkar's algorithm and the elipsoid metho, Operatios Research Letters, Primera edición, 1987.

[7] Sylvain E Cappell, Michael E.Taylor, Contemporary Mathematics, Mathematical Developments Arising from Linear Programing, The AMS-IMS-SIAM Joint Summer Research Conference on Mathematical, 1988.

[8] Barry Render, Michael E. Hanna, Métodos cuantitativos para los negocios, Novena edición, Pearson Pretice Hall, 2002.

[9] Marcel F. Goic, Algoritmo de Karmarkar, Universidad de Chile, Facultad de Ciencias Físicas y Matemáticas, Departamento de Ingeniería Industrial, Dirección URL: http://www.contrib.andrew.cmu.edu/ mgoic/-files/documents/optimization/karmarkar.pdf.

[10] Algoritmo de Karmarkar y matrices ralas, Revista de Matemática: Teoría y Aplicaciones 2(2):35-48,1995, Dirección URL: http://revista.emate.ucr.ac.cr/index.php/revista/article/viewFile/24/20.