UNIVERSIDAD CENTRAL "MARTA ABREU" DE LAS VILLAS
FACULTAD DE MATEMÁTICA, FÍSICA Y COMPUTACIÓN
Departamento de Matemática

Aplicación de los Métodos de Punto Interior para resolver Problemas de Optimización Lineal y Cuadráticos Convexos

Tesis presentada en opción al título académico de:

Master en Matemática Aplicada

Autor: **Lic. Gonzalo J. Palencia Fernández**

Tutora: **Dra. Rosina Hing Cortón**

2008

Pensamiento

"Ningún problema puede resolverse con la misma conciencia con que se creo"

Albert Einstein (1879-1955)

Agradecimientos

A mi tutora Rosina Hing Cortón por su ayuda y por su dedicación en todo momento

A Marisela Mainegra Hing por su solidaridad

A Miriam Nicado y al Consejo Científico de la Facultad

A mis Padres

A Iris por su dedicación

A todos los que me han ayudado en estos años

A todos los que han contribuido a la conclusión de este trabajo

Resumen

En este trabajo se hace una breve descripción de las ideas esenciales que sirven de base a los Métodos de Punto Interior para resolver problemas de Optimización Lineal. Se implementa una variante de un algoritmo Predictor-Corrector sobre Delphi 6.0.

Se obtiene una modificación recursiva de Cholesky que permite la factorización de matrices Semidefinidas Positivas, aun cuando estas no sean Definidas Positivas, sin incremento de costo computacional. Con ayuda de esta factorización se transforman los problemas de Programación Cuadrática Convexa en uno de Programación Cónica de segundo orden, los cuales se resuelven con la ayuda de la generalización del algoritmo Predictor-Corrector de Mehrotra para dichos problemas. Nuestros software GALOIS 1.0 (para el problema de la Programación lineal) y LARX 2.0 (para el problema de Programación Cuadrática Convexa) corren sobre una plataforma Windows. Para ambos tipos de problemas se realizan experimentos numéricos para validar los resultados obtenidos.

Abstract

This research work provides a brief description of the essential elements that serve as basis for the Interior Point Methods in solving linear optimization. It implements a variant of an Predictor-Corrector algorithm on Delphi 6.0. This produces a change of recursive Cholesky factorization on allowing parent Semidefinite Positive, even though these are not Positive Definite without increasing computational cost. Using this factoring come to the problems of a Convex Quadratic Programming one of Second Order Conic Programming, which are resolved with the help of widespread Mehrotra Predictor-Corrector algorithm to those problems. Our software GALOIS 1.0 (for the problem of Linear Programming) and LARX 2.0 (for the problem of Convex Quadratic Programming) run on a Windows platform. For both types of problems are performed numerical experiments to validate the results.

Tabla de Contenidos

Capítulo 1. Introducción

La Optimización Lineal (LO) es una de las técnicas más ampliamente divulgadas de la Matemática Aplicada. Aunque la LO fue utilizada por G. Monge en 1776, se considera a L. V. Kantorovich uno de sus creadores, ya que éste la formalizó en su libro Métodos Matemáticos para la Organización de la Producción en 1939 y la desarrolló en su trabajo sobre la transferencia de masas (1942), además recibió el premio Nobel de Economía en 1975 por sus aportes al Problema de la Asignación Óptima de Recursos Humanos. Uno de los momentos más importantes en el desarrollo de la LO fue la aparición del Método Simplex (MS). Este método desarrollado por G. B. Dantzig en 1947, tiene como idea fundamental el paso de una solución factible básica a otra de forma tal que se obtenga un mejor valor de la función objetivo. Esta estrategia reduce la tarea de resolver un problema de LO a la búsqueda en el conjunto de soluciones factibles básicas, las que tienen al menos $n-m$ variables nulas. Cómo para un problema con n variables y m restricciones hay a lo sumo $\frac{n!}{m!(n-m)!}$ soluciones básicas, por lo que podemos aplicar una técnica de búsqueda finita. El MS ha demostrado robustez y eficiencia en la práctica, pero los esfuerzos por demostrar, utilizando la teoría de complejidad que es un algoritmo de tiempo de corrida polinomial han fracasado, ya que por el contrario se ha podido probar que para cada $d>1$, hay un Programa Lineal con $2d$ ecuaciones, $3d$ variables y con coeficientes enteros con valores absolutos acotados por 4, tal que el MS puede realizar 2^d-1 iteraciones antes de encontrar el óptimo[PS82]. Los años 50 han sido la década del descubrimiento de las interioridades teóricas de la OL, de la extensión de su aplicación al escenario industrial, a problemas definitivamente combinatorios y de la elaboración de los primeros códigos de propósito general. Los 60 son llamados los años del surgimiento de la OL en gran escala y de la extensión el MS a la Optimización Cuadrática (QO). Finalizando la década del 70 se abre una nueva etapa en el desarrollo de la OL, caracterizada por la aparición de nuevos algoritmos. El primero de ellos fue desarrollado por Jachiyán en el año 1979 y está basado en el método elipsoidal. Con este resultado se demuestra teóricamente que los problemas lineales son polinomialmente resolubles. Este resultado tuvo gran impacto teórico pero poca eficiencia práctica, no obstante sirvió de

estímulo al surgimiento de nuevos algoritmos para estos problemas.

En el año 1984 Karmarkar publica su algoritmo proyectivo, que muestra mejor cota polinomial que el de Jachiyán y logra mejores resultados en la práctica, ya que supera la eficiencia del Simplex en problemas grandes y con matrices poco densas. En el período de 1984 a 1989 fueron publicados más de 1300 artículos en esta materia y en este marco aparecen los denominados Método de Punto Interior (MPI) para la OL, los que a diferencia del Simplex, buscan la solución en el interior del octante positivo. Existen en la literatura diferentes variantes de MPI, siendo los más divulgados los que se basan en la aplicación de las técnicas de barrera logarítmica y el desplazamiento en una vecindad del camino central.

Podemos mencionar diferentes autores que han obtenido execelentes resultados en el desarrollo de este nuevo enfoque de la OL, por ejemplo, Dikins, Frish, Huard, Nesterov, Nemirovski, Breitfield, Shanno, Jarre y Zowe. Con el desarrollo de esta novedosa teoría no solamente se logra una mejor comprensión de estos problemas, sino también una mayor efectividad en la solución práctica de los mismos.

La aparición de estos nuevos algoritmos, conjuntamente con el impetuoso desarrollo tanto de la teoría como de la tecnología de cómputo, se calcula que en los últimos 10 años la velocidad para obtener la solución de estos problemas se ha multiplicado por cantidades con seis órdenes de magnitud. Esto significa que problemas que no podían ser resueltos 10 años atrás, con menos de 1 año de cálculos continuos, ahora puede obtenerse su solución en sólo algunos minutos [AT02].

La programación cuadrática es muy importante, en parte, porque muchas formulaciones de este tipo surgen de manera natural en muchas aplicaciones. En este trabajo consideramos el siguiente Problema de Programación Cuadrática Convexa (QCP):

$$
\begin{gathered}
\tfrac{1}{2} x^t Q x + c^t x \rightarrow \min \\
\text{sujeto a :} \\
Ax = b \\
x \geq 0 \\
c, x \in R^n, \quad Q \in M(n,n), \quad A \in M(m,n), \quad b \in R^n, \\
rank(A) = m, \quad Q^t = Q, \quad Q \geq 0
\end{gathered}
\qquad (1)
$$

Este es uno de los modelos más utilizados para describir problemas relacionados con la ingeniería, el control, las finanzas, la optimización robusta y la optimización combinatoria. Referencias a algunas de estas aplicaciones aparecen en [BN01], [GT01] y [AT02]. Aunque se trata de un problema no lineal, el hecho de que tanto las restricciones, como las condiciones de Karush-Kuhn-Tucker(KKT) se describen mediante sistemas lineales, permite elaborar métodos de solución más simples que los generales de la Programación No-Lineal. De hecho en el caso en que Q sea definida positiva, estos problemas no son más difíciles de resolver que los de Programación Lineal.

En 1959 apareció el primer algoritmo Simplex-Cuadrático formulado por Wolfe P., que fue el más ventajoso en esa época, pero que resultaba realmente engorroso por el incremento del número de variables que producía y por la necesidad de aplicar un procedimiento de Fase I para determinar una solución inicial factible.

En la actualidad para resolver el problema (1) se utilizan métodos basados en la condición necesaria de primer orden que parten de un estimado inicial de la solución [JN99] y que requieren la resolución del sistema de KKT:

$$\begin{bmatrix} -p \\ \lambda \end{bmatrix} = \begin{bmatrix} Q & A^t \\ A & 0 \end{bmatrix} \begin{bmatrix} -p \\ \lambda \end{bmatrix} = \begin{bmatrix} g \\ 0 \end{bmatrix}, \tag{2}$$

El paso de mayor complejidad computacional de este algoritmo es la solución del sistema (2). Si la matriz A es de rango completo y Q es definida positiva, es decir $Z^t Q Z > 0,$ donde las columnas de Z constituyen una base del subespacio nulo de $A,$ existen métodos para resolverlo que son eficientes para problemas pequeños o de mediana escala. Usualmente esto se hace factorizando directamente la matriz del sistema utilizando métodos de factorización de matrices indefinidas como el de Bunch-Parlett, invirtiendo la matriz Q y luego factorizando $A_k^t Q^{-1} A_k$, o utilizando el método del subespacio nulo que requiere el cálculo de Z y la factorización de la matriz $Z^t Q Z$ [JN99]. Si $Z^t Q Z$ no es definida positiva, el problema de determinar cuando un punto factible es minimizador global es un problema NP-hard [MK87]. Aunque el algoritmo anterior ha sido descrito a partir de un punto inicial factible, el mismo

permite comenzar con un estimado inicial de la solución del problema, que no sea necesariamente factible [JN99].

La importancia de buscar métodos cada vez más eficientes para resolver los QCP se ha incrementado en las últimas décadas debido a que uno de las técnicas más usadas para resolver problemas generales de programación no lineal es la Programación Cuadrática Secuencial, que requiere resolver en cada iteración un problema cuadrático. Un gran numero de paquetes como el NPSOL, NOPQL, OPSYC, OPTIMA, MATLAB y SQP están basados en este enfoque. En su forma más pura este algoritmo remplaza la función objetivo en el punto actual, por su aproximación cuadrática dada por:

$$q(x_k) = \nabla f(x_k)^t d + \frac{1}{2} d^t \nabla^2_{xx} L(x_k, \lambda_k) d,$$

y reemplaza las restricciones por su aproximación lineal. Por tanto para determinar el paso desde el punto actual debe resolverse el siguiente problema cuadrático:

$$\begin{aligned} &\min q(x_k) \\ s.a \quad &c_i(x_k) + \nabla c_i(x_k)^t d = 0, i \in W_k \end{aligned}$$

Existen diferentes variantes de este algoritmo, por ejemplo, se sustituye la matriz Hessiana del Lagrangiano por una aproximación BFGS de éste, se utiliza búsqueda lineal para mejorar la convergencia o se utiliza el Lagrangiano Aumentado con una función de Mérito o una función de Penalidad para resolver el compromiso entre el decrecimiento de la función objetivo y la verificación de las restricciones.

Debido al fabuloso desarrollo alcanzado por la Programación Lineal (LP) a partir de la aplicación de algoritmos basados en los MPI, que han permitido resolver problemas con millones de variables y millones de restricciones en pocos minutos [AT02] se comienzan a aplicar estas técnicas a los Problemas Cuadráticos(QP) convexos. Efectivamente si escribimos las condiciones de KKT al problema QCP, incluyendo las restricciones de igualdad obtenemos que el minimizador debe satisfacer el sistema:

$$Qx + c - A^t\lambda - s = 0$$
$$Ax - b = 0$$
$$s_i x_i = 0, i = 1,\ldots, n$$
$$(s, x) \geq 0$$

muy parecido al que se obtiene en la LP, por tanto podemos aplicar las técnicas de los MPI y desarrollar algoritmos similares por ejemplo al Predictor-Corrector de Mehrotra. La única diferencia es que en este caso los incrementos de las variables primales y duales no son independientes, ya que están relacionados por la primera ecuación. En este caso, al igual que en los algoritmos que usan el conjunto activo, puede apreciarse que pueden surgir dificultades computacionales si la matriz Q no es definida positiva. La implementación es muy eficiente para los problemas estrictamente convexos y aunque las iteraciones son más costosas que en los Métodos del conjunto Activo, se logra la convergencia con un número menor de iteraciones.

El desarrollo de la teoría de los MPI han dominado el campo de la Optimización en los últimos 15 años y debido fundamentalmente a la influencia de los trabajos de [NT97] y [NS96] ha dado lugar al surgimiento de la Programación Cónica (CP) y en particular de la Programación Cónica de Segundo Orden $(QCQP)$, donde se trata el problema de la minimización de una función lineal sobre la intersección de una variedad lineal afín, con el producto cartesiano de conos de segundo orden. Estos problemas se caracterizan por ser muy tratables computacionalmente ya que en [NS96] se prueba que los algoritmos de MPI para estos problemas alcanzan una complejidad de $\sqrt{r}$, donde r es el número de conos de segundo orden que aparecen en el problema y [NS96] prueban que el Método Primal Dual de Punto interior de la Programación Lineal puede ser aplicado palabra por palabra a los *QCQPs*, el desarrollo de estos algoritmos comienza con los trabajos de [NT97] y [NT98].

Un problema QCQP se describe de la forma siguiente:

$$\begin{aligned} \min &\rightarrow \hat{c}^t \hat{x} \\ s.a.\ & \hat{A}\hat{x} = \hat{b} \\ & \hat{x} \in K \end{aligned} \tag{3}$$

donde K, es el producto de conos convexos K_i, cerrados de segundo orden, $K_i = \left\{ \left(x_i^0, \hat{x}_i \right) \middle| \left\| \hat{x}_i \right\| \le x_i^0, x_i^0 \ge 0, \hat{x}_i \in R^{n_i} \right\}, i = 1,...N$. En particular, el octante no negativo puede ser considerado como un producto de conos de segundo orden unidimensionales [Ren01].

En los trabajos de [AG03] demuestran que los QCP pueden ser expresados como QCQP, de la siguiente forma:

$$\begin{aligned} &\min u_0 \\ s.a\ \ & Q^{½} x - \hat{u} = \tfrac{1}{2} Q^{-½} c \\ & Ax = b \\ & x \ge 0, \left(u_0, \hat{u} \right) \in K \\ & K = \left\{ \left(u_0, \hat{u} \right) \middle| u_0 \ge 0, \left\| \hat{u} \right\| \le u_0 \right\} \end{aligned} \tag{4}$$

la obtención de está transformación desde el punto de vista práctico implica el cálculo de la matriz $Q^{½} = \Lambda^{½} T^t$, donde $\Lambda^{½} = diag\left(\sqrt{\lambda_1}, ... \sqrt{\lambda_n} \right)$ y T es una matriz ortogonal cuyas columnas son los vectores propios de la matriz Q. Lo que a su vez requiere el cálculo de los valores propios, cálculo de vectores propios y ortonormalización de éstos. También se requiere invertir la matriz $Q^{½}$, lo cual no es posible si la matriz Q no es definida positiva. Nuestro trabajo consiste en la implementación de un algoritmo de Punto Interior para resolver Problemas Lineales, dicho algoritmo puede ser utilizado con fines docentes e investigativos en el Grupo de Optimización, así como también buscar un algoritmo eficiente para transformar el QCP a un QCQP, que sea eficiente aún cuando la matriz Q no sea definida positiva y aplicar para su solución un algoritmo Primal-Dual de Punto Interior.

Problema de Investigación

El método más conocido para resolver problemas de Programación Lineal, el Método Simplex (MS), es debido a Dantzig, quien lo introdujo en 1947. Hasta la década de los años 70, el MS en sus diversas variantes resultaba el procedimiento más efectivo para dar solución a los Programas Lineales. Fue utilizado con éxito en los Centros de Cómputos de todo el mundo, independientemente que desde el punto de vista teórico presentaba el inconveniente de no ser un algoritmo de tiempo polinomial, aunque en la práctica los resultados eran satisfactorios, aún con problemas de miles de variables y miles de restricciones. A partir de la aparición de los trabajos de Kantorovich miles de investigadores se dieron a la tarea de encontrar algoritmos más eficientes que los basados en el Simplex y estos esfuerzos se vieron coronados ya en la década de los 90 con un rotundo éxito, al aparecer toda una familia de métodos, conocidos como Métodos de Punto Interior, que permiten obtener la solución de problemas lineales con un número de iteraciones que es de orden polinomial con respecto al tamaño de los problemas. Se reporta en la literatura la solución de problemas con millones de variables y millones de restricciones en solamente varios minutos, con estos algoritmos. Mientras que los algoritmos basados en el Simplex buscan la solución en el conjunto finito de vértices del poliedro determinado por las restricciones del problema, los métodos de Punto Interior se mueven por el interior de este conjunto, aproximándose a la solución optima del problema.

Las técnicas de Punto Interior, primeramente en la forma de métodos de barrera, fueron extensamente usadas desde los años 1960 para problemas con restricciones no lineales. Después de la publicación del artículo sobre MPI en 1984 de Narendra Karmarkar ha habido un resurgimiento en la extensión de tales métodos hacia la Programación Lineal, Programación Cuadrática, en específico, el caso convexo. Toda esta investigación ha traído como resultado el desarrollo de un algoritmo primal-dual de Punto Interior, en el cual es altamente eficiente ambos en teoría y en la práctica. Por tanto, varios autores han estudiado como generalizar este algoritmo a otros problemas. Un importante trabajo en esta dirección es el artículo de Nesterov y Todd en el cual muestra que el algoritmo primal-dual mantiene su eficiencia teórica cuando las restricciones no negativas son reemplazadas por un cono

convexo. Los programas lineales, los programas cuadráticos convexos se pueden formular como un problema de Programación Cónica de Segundo Orden (QCQP).

Schmieta y Alizadeh han demostrado que muchos de los algoritmos polinomiales desarrollados para la optimización semidefinida inmediatamente pueden ser transformados en algoritmos polinomiales para la CP.

En nuestro Grupo de Optimización no existen experiencias anteriores en la implementación de los algoritmos de MPI. En las tesis de doctorado de esta rama de la Matemática solo hemos encontrado referencia a estos en el trabajo de [Kakes02]. En el caso de problemas cuadráticos, aunque los códigos que se usan actualmente son muy eficientes, se presentan dificultades si Q no es Definida Positiva (DP).

Hipótesis de investigación

Es posible implementar algoritmos de Punto Interior que permitan resolver problemas de Programación Lineal y de Programación Cuadrática Convexa con matriz SDP.

Objetivos Generales

1. Dominar los fundamentos teóricos y aspectos computacionales de los Métodos de Punto Interior relacionados con la Programación Lineal y la Programación Cuadrática Convexa con matriz SDP.
2. Diseñar y elaborar nuestros sistemas que permitan resolver problemas de Optimización Lineal que puedan ser utilizados por el Grupo de Optimización con fines docentes e investigativos.
3. Diseñar e implementar un algoritmo que permita resolver eficientemente QCP (Problema Cuadrático Convexo) aunque la matiz de la forma cuadrática no sea DP.

Objetivos específicos:

1. Implementar el Método primal-dual de Barrera Logarítmica, en la versión de Mehrotra, para resolver Problemas Lineales.
2. Implementar un algoritmo que permita factorizar matrices simétricas SDP, aún cuando éstas no sean DP.

3. Programar la transformación de un problema de Programación Cuadrática Convexa con matriz SDP en un problema de Programación Cónica de Segundo Orden.
4. Implementar la generalización del algoritmo Predictor-Corrector de Mehrotra propuesto por Alizadeh para resolver problemas de Programación Cuadrática Convexa con matriz SDP.
5. Realizar experimentos numéricos para verificar la efectividad de los sistemas.

Tareas de investigación

1. Análisis de los diferentes algoritmos de Punto Interior que permitan resolver problemas de LP y QCP con matriz SDP, teniendo en cuenta su fundamentación teórica y complejidad computacional.
2. Implementación del Método Primal-Dual de Barrera Logarítmica, en la versión de Mehrotra para resolver problemas de LP.
3. Análisis y estudio profundo de la caracterización de las matrices SDP.
4. Implementación de una modificación del algoritmo recursivo de Cholesky que permite la factorización de matrices SDP, aun cuando estas no sean DP.
5. Justificación teórica de la transformación del QCP con matriz SDP a un QCQP.
6. Programación de la transformación del QCP con matriz SDP a un QCQP.
7. Implementación de la generalización del algoritmo Predictor-Corrector de Mehrotra propuesto por Alizadeh para resolver problemas QCP con matriz SDP.
8. Realización de los experimentos numéricos para comprobar la efectividad de los sistemas.

Novedad Científica

A. La implementación de una modificación del algoritmo recursivo de Cholesky que permite la factorización de matrices SDP, aun cuando estas no sean DP.
B. Transformación del QCP con matriz SDP a un QCQP y su programación.
C. Construcción de nuestro software propio capaz de resolver problemas LP y QCP con matriz SDP.

La tesis está estructurada en: cuatro Capítulos, Conclusiones, Recomendaciones, Referencias Bibliográficas y Anexos.

El resto de la tesis esta estructurado como sigue:

En el Capítulo 2, titulado Método de Punto Interior para los LP, se señalan las principales definiciones y notaciones que serán utilizadas, se presenta una síntesis de las ideas esenciales en las que se basan los algoritmos de tipo Primal-Dual de Punto Interior que se utiliza en nuestra implementación y por ultimo se describe los pasos de nuestro algoritmo.

En el Capítulo 3, titulado Método de Punto Interior para los QCP, se aclaran las notaciones utilizadas, se detalla el algoritmo de factorización de Cholesky para matrices SDP y ejemplos, se introduce el desarrollo teórico de la transformación del problema QCP con matriz SDP a un QCQP.

El Capítulo 4 se dedica a la realización de los Experimentos Numéricos, comparando los resultados obtenidos con el Galois, el Storm y el Mathematica para el caso LP y para el caso de QCP (LARX 2.0) comparando con las funciones de optimización del Mathematica 5.5 y el MatLab 6.0, teniendo en cuenta diferentes dimensiones para ambos tipos de problemas, así como también se exponen todos los detalles del algoritmo de Alizadeh para el caso cuadrático.

Capítulo 2. Método de Punto Interior para el caso LP

En este capítulo primeramente se introducen las definiciones y notaciones que serán utilizadas, después se hace una breve descripción de las ideas esenciales que sirven de base a los Métodos de Punto Interior para resolver Problemas de Optimización Lineal y por último se presenta una variante de un algoritmo de MPI que está basado en el método primal-dual de Barrera logarítmica, con la variante del paso Predictor- Corrector.

2.1 Notaciones y Definiciones Utilizadas

Comenzamos definiendo el problema Primal (P) de la siguiente manera:

$$\min \mapsto \left\{ c^t x \ : \ Ax = b, x \geq 0 \right\}$$
$$A \in M(m,n), \quad rank(A) = m, \ c, x \in \Re^n, b \in \Re^m$$

El Dual (D) del problema anterior es:

$$\max \mapsto \left\{ b^t y \ : \ A^t y + s = c, s \geq 0 \right\}$$
$$A \in M(m,n), \quad rank(A) = m, \ c, s \in \Re^n, \ b, y \in \Re^m$$

Las regiones factibles del Primal y el Dual son denotadas P y D respectivamente..

Las regiones estrictamente factibles del Primal y el Dual son denotados por P^+ y D^+ :

$$P^+ \ : \ \min \to \left\{ c^t x \ : \ Ax = b, x > 0 \right\}, \quad D^+ \ : \ \max \to \left\{ b^t y \ : \ A^t y + s = c, s > 0 \right\}$$

Usaremos las siguientes notaciones para vectores y matrices relacionadas:

$$x = (x_1, x_2, ..., x_n)^t, \quad X = diag(x_1, ..., x_n). \quad s^{-1} = \left(\frac{1}{s_1}, \frac{1}{s_2}, ..., \frac{1}{s_n} \right)^t$$

Definiremos el producto de vectores por coordenadas por:

$$xs^t = (x_1 s_1, x_2 s_2, ..., x_n s_n)^t$$

Esta operación es conmutativa, asociativa y su vector unitario será: $e = (1,1,...,1)^t$

De las notaciones y definiciones anteriores es evidente que:

$$x = Xe, \quad x^{-1} = X^{-1} e, \quad xs = XSe$$

Introduciremos ahora la llamada función Primal de Barrera Logarítmica para el problema Primal (P).

Esta es la función $\tilde{g}_\mu(x)$ definida por:

$$\tilde{g}_\mu(x) = c^t x - \mu \sum_{j=1}^{n} Log(x_j)$$

Donde μ es un número positivo llamado parámetro de barrera, x corre sobre todos los vectores factibles para el problema primal, que son positivos. El dominio de $\tilde{g}_\mu(x)$ es el conjunto P^+. Con la optimización de $\tilde{g}_\mu(x)$, tratamos de resolver dos ideas a la vez: Encontrar un vector x factible para P^+, para el cual $c^t x$ sea pequeña y a la vez $\sum_{j=1}^{n} Log(x_j)$ sea grande. Con el método de Barrera logarítmica solucionamos ambos problemas en tiempo polinomial.

2.2 Fundamentos Teóricos de los MPI

El punto de partida es la aplicación de las condiciones de Karush-Kunhn-Tucker (KKT) al problema Primal. Usando notaciones apropiadas para los multiplicadores de Lagrange, estas condiciones consisten en:

$$\begin{aligned} A^t y + s - c &= 0 \\ Ax - b &= 0 \\ XSe &= 0 \\ (x, s) &\geq 0 \end{aligned} \qquad (5)$$

Del análisis de este sistema podemos ver que en la 1a., 2a, y 4ta. Línea están contenidas las condiciones de factibilidad del Primal y del Dual. Además que la 3a. ecuación es la condición de holgura complementaria, la cual es una condición necesaria y suficiente para la optimalidad en estos problemas. De esta manera podemos apreciar la relación entre el Método Simplex y los MPI. La clase de los algoritmos que se basan en la solución de este sistema se denomina Primal-Dual.

Si (x, y, s) es una solución de (5) entonces x es solución óptima de y (y, s) es solución

óptima del Dual. Los (x, y, s) que satisfacen las tres primeras ecuaciones de (5), pero no las condiciones de no negatividad de la cuarta línea, determinan soluciones espurias del Primal y del Dual.

La complejidad del sistema (5) se debe fundamentalmente a la tercera ecuación que es no lineal y a la inecuación de la cuarta fila. Prescindiendo de la inecuación podemos considerar la aplicación de un método iterativo para resolver sistemas no lineales, que requieren un punto inicial y un procedimiento para determinar la dirección de búsqueda y el tamaño del paso en está dirección. Sustituyendo en el sistema (5) (x, y, s) por $(x + \Delta x, y + \Delta y, s + \Delta s)$, se obtiene que el desplazamiento óptimo a partir del punto inicial debe satisfacer el sistema siguiente:

$$\begin{aligned} A^t \Delta y + \Delta s &= c - A^t y - s = r_c \\ A\Delta x &= b - Ax = r_b \\ S\Delta x + X\Delta s + \Delta x \Delta s &= -XSe \end{aligned} \tag{6}$$

Una de las técnicas más usadas por los MPI es el método de Newton, en el que se determinan la dirección y la longitud del paso resolviendo el sistema lineal que se obtiene eliminando la parte no lineal y haciendo algunos artificios algebraicos en (6), es decir:

$$\begin{aligned} A S^{-1} X A^t \Delta y &= -r_b + A\left(S^{-1} X r_c + Ax - \mu S^{-1} e\right) \\ \Delta s &= -r_c - A^t \Delta y \\ \Delta x &= -x - S^{-1} X \Delta x \end{aligned} \tag{7}$$

El paso determinado por el sistema (7) se denomina paso completo de Newton o paso de escala afín. Como no se han tenido en cuenta las restricciones de no negatividad el vector $(x + \Delta x, s + \Delta s)$ puede tener componentes no positivas, para evitar este inconveniente se introduce un factor de amortiguamiento que reduce el tamaño del paso, para garantizar la no violación de estas restricciones. Por otro lado este factor de amortiguamiento puede reducir demasiado el tamaño del paso, cuando el punto actual está muy próximo a la frontera del octante positivo $(x, s) \geq 0$, lo que ocasiona que las iteraciones avancen muy lentamente hacia la solución del sistema (5).

Una de las formas más utilizadas en los MPI para resolver esta contradicción esta basada en la introducción del concepto de camino central, que es un arco parametrizado C, compuesto de puntos estrictamente factibles definido por:

$$C = \{(x, y, s) \mid A^t y + s = c, Ax = b, xs = \mu e, (x, s) > 0, \mu > 0\}$$

La idea es tomar un punto inicial que sea estrictamente factible y acercarnos en cada iteración, no a la solución del sistema (5), sino a un punto del camino central correspondiente a un valor de μ. Con el parámetro $\mu > 0$, se evita un acercamiento demasiado temprano a la frontera del octante $(x, s) > 0$,. evitando así la aproximación a soluciones espurias. Por otro lado haciendo tender de manera apropiada el parámetro μ a cero, estamos reduciendo el déficit de dualidad y por tanto aproximándonos a la solución primal-dual óptima.

El sistema para la búsqueda del paso para la solución del problema que define un punto del camino central para un valor dado de μ tiene la forma:

$$\begin{aligned} AS^{-1}XA^t\Delta y &= -r_b + A\left(S^{-1}Xr_c + Ax - \mu S^{-1}e\right) \\ \Delta s &= -r_c - A^t\Delta y \\ \Delta x &= -x + \mu S^{-1}e - S^{-1}X\Delta x \end{aligned} \tag{8}$$

Para lograr un acuerdo entre estos dos objetivos se utiliza un parámetro de centrado $\sigma \in [0,1]$ para la reducción de μ según la fórmula: $\mu = (1-\sigma)\mu$.

Tomando valores intermedios $\sigma \in (0,1)$ se lograrán en cierta medida los dos objetivos, es mantenerse cerca del camino central para evitar soluciones espurias y disminuir el déficit de dualidad. Una fundamentación detallada y rigurosa de esta teoría puede encontrarse en [JN99] y en forma aún más exhaustiva en [RT97].

2.3 Implementación del Método Primal - Dual de Punto Interior

Se implementa una variante del Método Primal Dual de Punto Interior [RT97], que tiene las siguientes características:

- Nuestro algoritmo asume que el problema de optimización lineal debe estar en forma estándar.
- Los valores iniciales que nuestro algoritmo usa por defecto son: $x^0 = ne$; $s^0 = ne$; $y^0 = 0$. Estos valores iniciales son sugeridos en [AS97]. Usamos además $\mu_0 = 4$.

- Las longitudes del paso para el problema primal α_P , y para el problema dual α_D se calculan de la siguiente manera:

$$\alpha_P = \min\left\{1, \min\left\{\frac{x_i}{-\Delta x_i} : \Delta x_i < 0,\ i = 1,2,...,n\right\}\right\}$$
$$\alpha_D = \min\left\{1, \min\left\{\frac{y_i}{-\Delta y_i} : \Delta y_i < 0,\ i = 1,2,...,n\right\}\right\}$$

Estas longitudes del paso son reducidas por $\alpha_0 = 0.8,$ para evitar que el nuevo punto se encuentre en la frontera del octante $(x, s) \geq 0,$ este valor se ajustó mediante los experimentos numéricos.

Así x, y y s se calculan para cada iteración de la manera siguiente:

$$\begin{aligned} x^{k+1} &= x^k + \alpha_0 \alpha_P \Delta x \\ y^{k+1} &= y^k + \alpha_0 \alpha_D \Delta y \\ s^{k+1} &= s^k + \alpha_0 \alpha_D \Delta s \end{aligned} \tag{9}$$

- Para actualizar μ usamos el siguiente criterio $\mu := (1-\sigma)\mu$, donde $\sigma = \frac{1}{4}$, este valor del parámetro de centrado también fue ajustado a partir de los resultados experimentales.
- La condición de parada fue determinada por la proximidad a cero del déficit de dualidad y la proximidad del punto actual las regiones factibles P y D. Para verificar estas condiciones usamos los criterios siguientes:

$$\frac{\left|c^t x - b^t y\right|}{1 + \left|c^t x\right|} \leq \varepsilon \text{ , donde } \varepsilon = 10^{-8},\ \ \|r_P\| \leq \varepsilon, \|r_D\| \leq \varepsilon.$$

Para concluir que el problema es no acotado usamos los siguientes criterios [AS97]:

$$\|x\| \geq \frac{1}{\varepsilon} \quad \text{ó} \quad \|s\| \geq \frac{1}{\varepsilon}$$

2.4 Algoritmo Predictor - Corrector

***Entrada*:**

Un parámetro de precisión $\varepsilon > 0$; $(x^0, s^0, y^0) \in P^+ \times D^+$; $\mu^0 > 0$; $0 < \alpha^0 < 1$;

begin

$x := x^0$; $s := s^0$; $y := y^0$; $\mu := \mu^0$;

while not *Criterio De Parada* **do**

begin

Reducir μ;

Resolver el sistema (8) para obtener $(\Delta x, \Delta y, \Delta s)$;

Determinar α_P y α_D;

Actualizar (x, y, s) usando (9);

end;

end;

En este Capítulo fueron expuestas las ideas fundamentales en las que se basan los MPI para los LP y se hace una descripción de las principales características de los algoritmos que se diseñan en este marco teórico.

Capítulo 3. Método de Punto Interior para el caso QCP

En este capítulo el algoritmo que se propone para transformar un QCP con matriz SDP a un QCQP está basado en la aplicación de la teoría de las matrices semidefinidas. Para hacer esa transformación se reduce la forma cuadrática a la forma canónica y se introduce una variable más, "y". La fundamentación teórica esta basada en las propiedades de las matrices SDP, específicamente en la existencia de una base ortonormal de vectores propios. Todas estas propiedades están perfectamente demostradas en [Gant59]. Pero estos resultados son teóricos, sirven para la fundamentación pero no para la construcción práctica de un algoritmo que permita la reducción del problema. Para la construcción del algoritmo tratamos de utilizar la Factorización de Cholesky $Q = LL^t$, donde L es una matriz triangular inferior. La Factorización de Cholesky aparece perfectamente fundamentada para matrices DP y para SDP en el caso en que los r primeros menores seccionales sean distintos de cero (r es el rango de Q). En los comentarios sobre la implementación, en la bibliografía actualizada, se dice que el algoritmo de Cholesky puede funcionar para SDP, pero que en algunos casos es inestable y provoca divisiones por cero. Esto lo verificamos a través de ejemplos con el paquete Mathematica. En las notas de implementación del Mathematica garantiza su algoritmo para matrices DP y comprobamos que para SDP a veces no funciona, a veces ocurre divisiones por cero. Por lo que para facilitar la exposición se aclaran algunas notaciones que serán utilizadas, se diseña e implementa un algoritmo de factorización de Cholesky para matrices SDP. Por último se exponen la fundamentación teórica de transformar un QCP con matriz SDP a un QCQP, los pasos del algoritmo de la transformación y un ejemplo de la misma.

3.1 Notaciones Utilizadas

Notación 1: Dada la matriz $A \in M(m,n)$, o sea, $A = [a_{ij}] \quad i = \overline{1,m},\ j = \overline{1,n}$ se introduce una forma compacta de denotar los elementos de dicha matriz.

Sea

$$A\begin{pmatrix} i_1 & i_2 & \dots & i_p \\ k_1 & k_2 & \dots & k_p \end{pmatrix} = \begin{vmatrix} a_{i_1k_1} & a_{i_1k_2} & \dots & a_{i_1k_p} \\ a_{i_2k_1} & a_{i_2k_2} & \dots & a_{i_2k_p} \\ \vdots & \vdots & \vdots & \vdots \\ a_{i_pk_1} & a_{i_pk_2} & \dots & a_{i_pk_p} \end{vmatrix}$$

Notación 2: Menores de orden p de la matriz Q

$$Q\begin{pmatrix} i_1,...,i_p \\ k_1,...,k_p \end{pmatrix}, \quad \begin{matrix} 1 \le i_1 < i_2 < ... < i_p \le n \\ 1 \le j_1 < j_2 < ... < j_p \le n \end{matrix}, p \le n$$

donde los índices i representan las filas y los índices j las columnas, cuya intersección determina el menor.

Notación 3: Menores principales de orden p de la matriz Q.

$$Q\begin{pmatrix} i_1,...,i_p \\ i_1,...,i_p \end{pmatrix}, \quad 1 \le i_1 < i_2 < ... < i_p \le n, \ p \le n$$

Notación 4: Menores principales diagonales de orden p de la matriz Q.

$$Q\begin{pmatrix} 1,..., p \\ 1,..., p \end{pmatrix}, p \le n$$

3.2 Factorización de Cholesky para matrices SDP. Algoritmo Recursivo

Un argumento importante que será utilizado en nuestro trabajo es la caracterización de las matrices semidefinidas positivas, expresadas en el teorema siguiente [Gant59].

Teorema: Una condición necesaria y suficiente para que la matriz Q sea semidefinida positiva es que todos sus menores principales sean no negativos, es decir:

$$Q\begin{pmatrix} i_1,...,i_p \\ i_1,...,i_p \end{pmatrix} \ge 0, \quad 1 \le i_1 < i_2 < ... < i_p \le n, p = 1,...,n \tag{10}$$

Nuestro primer paso será llevar la forma cuadrática contenida en la función objetivo a su forma canónica sin utilizar las matrices que aparecen en (4). En su lugar utilizaremos la factorizacíon de Cholesky de la matriz Q.

Para las matrices definidas positivas se usa el método de factorización de Cholesky, que descompone la matriz Q en el producto siguiente:

$$Q = L\,L^t,$$

donde L - es una matriz triangular inferior. El algoritmo para calcular los elementos de L no requiere pivotación, requiere solamente $\frac{n^3}{3}$ operaciones y en su forma recursiva, para matrices Q definidas positivas [Van02] utiliza las fórmulas siguientes:

$$Q = \begin{bmatrix} q_{11} & Q_{21}^t \\ Q_{21} & Q_{22} \end{bmatrix} = \begin{bmatrix} l_{11} & 0 \\ L_{21} & L_{22} \end{bmatrix} \begin{bmatrix} l_{11} & L_{21}^t \\ 0 & L_{22}^t \end{bmatrix}$$

$$Q = \begin{bmatrix} l_{11}^2 & l_{11}L_{21}^t \\ l_{11}L_{21} & L_{21}L_{21}^t + L_{22}L_{22}^t \end{bmatrix}$$

para determinar la primera columna de L se usan las igualdades:

$$l_{11} = \sqrt{q_{11}}, \quad l_{11}L_{21} = Q_{21} \tag{11}$$

De las fórmulas anteriores es fácil ver que $L_{22}L_{22}^t,$ que es también definida positiva [Van02], es la factorización de la matriz ya conocida $Q_{22} - L_{21}L_{21}^t$. Por tanto usando fórmulas similares a la anteriores se halla la primera columna de L_{22} y consecuentemente la segunda columna de L. Repitiendo recursivamente el proceso, al cabo de *n* iteraciones se obtiene la matriz L buscada.

Es evidente que si tratamos de aplicar el algoritmo anterior a una matriz que sea semi-definida positiva pero no definida positiva, falla en algún paso en el que $\hat{l}_{11} = 0,$ por lo que es necesario hacer algunas modificaciones a éste.

Se han propuesto algunos algoritmos que primeramente determinan el rango de la matriz y seleccionan una submatriz DP a la cual se le aplica el algoritmo anterior, pero estos cálculos son considerados inestables [Hig96]. Se proponen también algoritmos basados en el conocimiento de la estructura del subespacio nulo de la matriz Q para una computación muy precisa de la factorización de Cholesky, muy eficientes para el caso en que tanto la matriz Q, como la base del subespacio nulo conocida, sean suficientemente poco densas [AD00]. En nuestro trabajo proponemos un algoritmo que no requiere ninguna información previa sobre la matriz semidefinida positiva, para lograrlo demostraremos la siguiente propiedad:

Proposición: Si Q es una matriz semidefinida positiva y $q_{ii} = 0$ entonces $q_{ij} = 0, \quad j = 1,...,n$.

Demostración: Sea $q_{ii} = 0,$ para todo $j = 1,...,n$ se tiene que los menores de segundo orden que se obtienen de la intersección de las filas i y j con las columnas i y j son iguales a: $q_{ii}q_{jj} - q_{ij}^2 = -q_{ij}^2,$ y aplicando (10) se obtiene la igualdad a cero de todos los elementos de la fila y la columna j-ésimas.

Con este resultado podemos analizar las igualdades (11) del algoritmo recursivo de Cholesky y obtener que si $q_{11} = 0$, entonces $l_{11} = 0$ pero como también la columna $Q_{21} = Q_{12} = 0$, la segunda igualdad se cumple para cualquier valor de $L_{21},$ por lo que la descomposición no es única. Para nuestro algoritmo tomamos la que simplifica más los cálculos, esto es $L_{21} = 0,$ lo que implica que $L_{22}L_{22}^t = Q_{22}$. Esto significa que en nuestro algoritmo modificado no se incrementan los cálculos, por el contrario pueden disminuir cuando disminuye el rango Q y el número de operaciones en el caso peor sigue siendo $\frac{n^3}{3}$.

3.3 Ejemplos de la Factorización para matrices SDP

Por ejemplo apliquemos la modificación propuesta a la matriz

$$\begin{bmatrix} 1 & 0 & 1 \\ 0 & 0 & 0 \\ 1 & 0 & 3 \end{bmatrix}, \text{ con valores propios : } \sqrt{2}+2, 2-\sqrt{2}, 0.$$

La matriz es semidefinida positiva ya que sus valores propios son no negativos, pero no es

definida positiva ya que uno de sus valores propios es nulo. En el primer paso del algoritmo de Cholesky se obtiene:

$$l_{11} = \sqrt{1} = 1,\ l_{11}L_{21}^t = 1\begin{bmatrix} l_{21} & l_{31} \end{bmatrix} = \begin{bmatrix} 0 & 1 \end{bmatrix},$$

$$L_{22}L_{22}^t = \begin{bmatrix} l_{22}^2 & l_{22}l_{32} \\ l_{22}l_{32} & l_{32}^2 + l_{33}^2 \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 0 & 2 \end{bmatrix} = \hat{Q}$$

En el segundo paso debemos aplicar la modificación propuesta:

$$l_{22} = \sqrt{0} = 0\,,\ \ l_{22}L_{32} = 0\ ,\ l_{32} = 0$$

La solución es indeterminada, si hacemos $l_{32} = 0,$ entonces:

$$\begin{bmatrix} l_{22} & l_{32} \end{bmatrix} = \begin{bmatrix} 0 & 0 \end{bmatrix}\ ,\ \ l_{33}^2 = \hat{q}_{33} = 2\,,\ \ l_{32} = \sqrt{2}$$

Por tanto la factorización obtenida será:

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & \sqrt{2} \end{bmatrix}\begin{bmatrix} 1 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & \sqrt{2} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 0 & 0 \\ 1 & 0 & 3 \end{bmatrix}$$

Para comprobar que la descomposición no es única podemos hacer $l_{32} = 1,$ entonces se obtiene que:

$$\begin{bmatrix} l_{22} & l_{32} \end{bmatrix} = \begin{bmatrix} 0 & 1 \end{bmatrix},\ \ l_{33}^2 = \hat{q}_{33} - l_{32}^2 = 2 - 1 = 1,\ l_{33} = 1$$

Por tanto la factorización obtenida será:

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}\begin{bmatrix} 1 & 0 & 1 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 0 & 0 \\ 1 & 0 & 3 \end{bmatrix}$$

Para un segundo ejemplo trabajamos con la matriz factorizada en [AD00], utilizando la información sobre su subespacio nulo. Con la modificación propuesta al algoritmo recursivo de Cholesky, sin ninguna información adicional, obtuvimos el mismo resultado.

$$\begin{bmatrix} 1 & 0 & 1 & 1 & 3 \\ 0 & 9 & 3 & 9 & 9 \\ 1 & 3 & 3 & 6 & 8 \\ 1 & 9 & 6 & 14 & 16 \\ 3 & 9 & 8 & 16 & 22 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 1 & 3 & 2 & 0 & 0 \\ 3 & 3 & 2 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 1 & 1 & 3 \\ 0 & 3 & 1 & 3 & 3 \\ 0 & 0 & 1 & 2 & 2 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

La factorización de Cholesky de esta matriz no fue hallada con los paquetes de cálculo disponible, por no ser definida positiva.

3.4 Transformación del QCP a un QCQP

Como resultado de aplicar nuestro algoritmo modificado de Cholesky a la matriz Q de rango r, se obtiene la matriz triangular inferior $L = [l_1 \quad l_2 \quad .... \quad l_n]$, donde exactamente $n-r$ vectores columnas son nulos. Si $r < n$ esta matriz es singular, por lo que construimos una nueva matriz de la forma siguiente:

Sea $I = \{i \mid l_i \neq 0\}$ y $J = \{j \mid l_j = 0\}$, entonces hacemos

$$\tilde{L} = [\tilde{l}_1 \quad \tilde{l}_2 \quad \ldots \quad \tilde{l}_n], \qquad \tilde{l}_k = \begin{cases} l_k, & k \in I \\ e_k & k \in J \end{cases},$$

donde e_k - es el k-ésimo vector canónico.

Utilizamos ahora la transformación lineal $\xi = \tilde{L}^t x,$ es decir $\xi_k = \tilde{l}_k^{\,t} x, k \in I$ y $\xi_k = x_k, k \in J$, con la que la forma cuadrática definida por Q se transforma en:

$$x^t Q x = x^t L L^t x = \left(L^t x\right)^t L^t x = \sum_{k \in I} \xi_k^2$$

Usando la matriz de permutación:

$$P = [e_{i_1} \quad \ldots \quad e_{i_{rr}} \quad e_{j_1} \quad \ldots \quad e_{j_{n-r}}],$$

donde los índices i pertenecen a I y los j a J y la transformación

$\zeta = P\xi = P\tilde{L}^t x,$ logramos que las primeras r componentes de ζ sean iguales a las componentes de ξ cuyos índices se encuentran en I.

Denotando por $\left(P\tilde{L}^t\right)^{-1} = H,$ y $G = \left[h_{ik}\right]_{i=1,..,r;.k=1...n}$ el problema QCP se puede reescribir como:

$$\begin{gathered} \sum_{i=1}^{r} \zeta_i^2 + c^t H\zeta \to \min \\ s.a \;\; AH\zeta = b \\ G\zeta \geq 0 \\ \zeta_j \geq 0,\; j = r+1,\ldots,n \end{gathered}$$

Introduciendo variables superplus y las notaciones: $\tilde{c} \in R^{n+r}$, $\tilde{c} = \left[c^t H \quad 0\right]$

$$\tilde{A} = \begin{bmatrix} AH_{m,n} & 0_{m,r} \\ G_{r,n} & -I_r \end{bmatrix} \quad \tilde{b} = \begin{bmatrix} b_{m,1} \\ 0_{r,1} \end{bmatrix}$$

el problema se transforma en:

$$\begin{gathered} \sum_{i=1}^{r} \zeta_i^2 + \tilde{c}^t\zeta \to \min \\ s.a \;\; \tilde{A}\zeta = \tilde{b} \\ \zeta_j \geq 0.\, j = r+1, n+r \end{gathered} \tag{12}$$

Se introduce ahora una nueva variable unidimensional $y,$ y se considera a ζ^1 como un vector r - dimensional que contiene las primeras r componentes de ζ, el problema anterior puede reescribirse como:

$$\begin{gathered} y_1 - y_2 \to \min \\ s.a. \quad \tilde{A}\zeta = \tilde{b} \\ \sum_{i=1}^{r} \zeta_i^2 + \tilde{c}^t\zeta \leq y_1 - y_2 \\ \zeta_j \geq 0.\, j = r+1, n+r \\ y_i \geq 0, i = 1,2 \end{gathered} \tag{13}$$

Se ha tenido en cuenta que la parte lineal de la restricción cuadrática puede ser negativa, por lo que se acota con la diferencia de dos variables no negativas.

Para transformar la restricción cuadrática de (13) en un cono de segundo orden usaremos una técnica similar a la utilizada en [AT02] para transformar una restricción cuadrática convexa en la intersección de dos restricciones lineales afines y un cono de segundo orden. Primeramente hacemos $y_1 - y_2 - \tilde{c}^t\zeta = \theta$ y añadimos la restricción lineal $t = 1$. Con lo que la restricción cuadrática se transforma en $\sum_{i=1}^{r}\zeta_i^2 \le \theta * t, \quad y_1 - y_2 - \tilde{c}^t\zeta = \theta$, y $t = 1.$ Haciendo ahora $\theta = u + v$ y $t = u - v,$ se obtiene finalmente el QCQP siguiente:

$$\begin{gathered} y_1 - y_2 \to \min \\ s.a. \qquad \tilde{A}\zeta = \tilde{b} \\ u + v - y_1 + y_2 + \tilde{c}^t\zeta = 0 \\ u - v = 1 \\ \sum_{i=1}^{r}\zeta_i^2 + v^2 \le u^2, u \ge 0 \\ \zeta_j \ge 0. j = \overline{r+1, n+r}, y_1, y_2 \ge 0 \end{gathered} \tag{14}$$

Es evidente que (14) es un problema de QCQP, ya que la función objetivo es lineal y las filas 2a., 3a. y 4a. contienen las restricciones lineales. La penúltima restricción representa un cono de segundo orden de dimensión $r+2$ y la última el producto cartesiano de $n+2$ conos de segundo orden unidimensionales.

El cálculo de mayor complejidad que debemos hacer es la inversión de la matriz $\tilde{L}^t$, que por ser triangular sólo requiere un número de operaciones que es equivalente también a $\frac{n^3}{3}$, por lo que incluyendo la factorización de Q la transformación requiere un número de operaciones que es aproximadamente $\frac{2n^3}{3}$.

3.4.1 Pasos para la transformación del QCP con matriz SDP a un QCQP

Se expondrán los pasos para la transformación del QCP con matriz SDP a un QCQP, luego de haberse justificado teóricamente este problema.

Paso 1:

Si $r < n$ donde $r = rank(Q)$ y n es el orden de la matriz Q, entonces la matriz Q es singular por lo que hay que calcular la matriz $\tilde{L}^t$ que no es más que la matriz L^t obtenida en la factorización de Cholesky [MZ03] para matrices SDP y DP ($Q = LL^t$) pero en las filas nulas se le agrega un número uno en la posición de la diagonal. Si la matriz no es singular simplemente $\tilde{L}^t = L^t$.

Paso 2:

Se reorganiza la matriz $\tilde{L}^t$ para que luego sea más fácil obtener la matriz de permutación. Esta matriz se organiza comenzando desde la primera fila, cuando se encuentra una fila nula se inserta al final de la matriz y se elimina de la posición que anteriormente tenía.

Paso 3:

Se calcula P que es una matriz de permutación la cual se obtiene a partir de la matriz identidad. Las filas que fueron cambiadas en la matriz $\tilde{L}^t$ también son cambiadas en la matriz identidad y el resultado se le asigna a la matriz P.

Paso 4:

Se calcula ζ de la siguiente forma $\zeta = P\tilde{L}^t x$.

Paso 5:

Se calcula la matriz H que no es más que $H = \left(P\tilde{L}^t\right)^{-1}$.

Paso 6:

Se calcula la matriz G que no es más que la matriz H pero sin las filas nulas. G es una matriz de r filas y n columnas.

Paso 7:

Calcular $c^t H$.

Paso 8:

Se calcula AH.

3.4.2 Forma matricial de la transformación

El problema de QCQP se puede representar en forma matricial como se muestra a continuación:

$$(y_1 - y_2) \rightarrow Min,$$

sa:

$$\begin{bmatrix} AH_{mxn} & 0_{mxr} & 0_{mx1} & 0_{mx1} & 0_{mx1} \\ G_{rxn} & -I_{rxr} & 0_{rx1} & 0_{rx1} & 0_{rx1} \\ \hat{c}_{1xn} & 0_{1xr} & 1_{1x1} & 1_{1x1} & -1_{1x1} \\ 0_{1xn} & 0_{1xr} & 1_{1x1} & -1_{1x1} & 0_{1x1} \end{bmatrix} * \begin{bmatrix} \zeta_{nx1} \\ s_{rx1} \\ u_{1x1} \\ v_{1x1} \\ y_{1(1x1)} \\ y_{2(1x1)} \end{bmatrix} = \begin{bmatrix} b_{mx1} \\ 0_{rx1} \\ 0_{1x1} \\ 1_{1x1} \end{bmatrix}$$

$$\sum_{i=1}^{r} \zeta_i^2 + v^2 \leq u^2$$

$$s, u \geq 0$$

$$\zeta_j \geq 0 \quad para \quad j = \overline{r+1, n+r}$$

La forma matricial del QCQP tiene una forma tal que permite que la matriz se divida en dos matrices M_1 y M_2, en la matriz M_1 estarían solamente las columnas que pertenecen a las variables de la restricción del cono cuadrático, teniendo a *u,* que es el radio del cono, en la primera posición de la matriz, y en M_2 el resto de las variables. En la transformación que se plantea en este trabajo la restricción del cono cuadrático contiene las variables *u*, *v* y ζ_i para $i = \overline{1, r}$ entonces su forma matricial quedaría como se muestra en (15).

$$(y_1 - y_2) \rightarrow Min,$$

sa :

$$\begin{bmatrix} 0_{mx1} & AH_{mxr} & 0_{mx1} & AH_{mx(n-r)} & 0_{mxr} & 0_{mx1} & 0_{mx1} \\ 0_{rx1} & G_{rxr} & 0_{rx1} & 0_{rx(n-r)} & -I_{rxr} & 0_{rx1} & 0_{rx1} \\ 1_{1x1} & \hat{c}_{1xr} & 1_{1x1} & \hat{c}_{1x(n-r)} & 0_{1xr} & -1_{1x1} & 1_{1x1} \\ 1_{1x1} & 0_{1xr} & -1_{1x1} & 0_{1x(n-r)} & 0_{1xr} & 0_{1x1} & 0_{1x1} \end{bmatrix} * \begin{bmatrix} u_{1x1} \\ \zeta_{rx1} \\ v_{1x1} \\ \zeta_{(n-r)x1} \\ s_{rx1} \\ y_{1(1x1)} \\ y_{2(1x1)} \end{bmatrix} = \begin{bmatrix} b_{mx1} \\ 0_{rx1} \\ 0_{1x1} \\ 1_{1x1} \end{bmatrix}$$

$$\sum_{i=1}^{r} \zeta_i^2 + v^2 \leq u^2 \qquad (15)$$

$$s, u \geq 0$$

$$\zeta_j \geq 0 \quad para \quad j = \overline{r+1, n+r}$$

donde:

r es el rango de la matriz original del problema cuadrático.

m es la cantidad de restricciones del problema cuadrático original.

n es la cantidad de variables del problema cuadrático original.

3.4.3 Ejemplo de la transformación de un problema QCP con matriz SDP a un QCQP

Problema de Programación Cuadrática Convexa:

$$x_1^2 + x_2^2 + x_3^2 - 2x_1 + x_2 \rightarrow Min$$

$$sa: x_1 + 2x_2 + 3x_3 + x_4 = 12$$

$$2x_1 + x_2 + x_3 + x_5 = 6$$

$$x_1 \geq 0, \quad x_2 \geq 0, \quad x_3 \geq 0, \quad x_4 \geq 0, \quad x_5 \geq 0$$

donde:

$$Q=\begin{bmatrix}1&0&0&0&0\\0&1&0&0&0\\0&0&1&0&0\\0&0&0&0&0\\0&0&0&0&0\end{bmatrix} \quad c=\begin{bmatrix}-2&1&0&0&0\end{bmatrix} \quad A=\begin{bmatrix}1&2&3&1&0\\2&1&1&0&1\end{bmatrix} \quad b=\begin{bmatrix}12\\6\end{bmatrix}$$

Primeramente se realiza la factorización de Cholesky de la matriz *Q*.

$$Q = LL^t$$

$$L = L^t = \begin{bmatrix}1&0&0&0&0\\0&1&0&0&0\\0&0&1&0&0\\0&0&0&0&0\\0&0&0&0&0\end{bmatrix}$$

Se busca la matriz $\tilde{L}^t$ que en este caso coincide con la matriz identidad.

$$\tilde{L}^t = \begin{bmatrix}1&0&0&0&0\\0&1&0&0&0\\0&0&1&0&0\\0&0&0&1&0\\0&0&0&0&1\end{bmatrix}$$

Se reorganiza la matriz $\tilde{L}^t$, en este caso no hace falta reorganizarla pues las filas nulas en la matriz L^t son las últimas.

Se construye la matriz de permutación *P*

$$P=\begin{bmatrix}1&0&0&0&0\\0&1&0&0&0\\0&0&1&0&0\\0&0&0&1&0\\0&0&0&0&1\end{bmatrix}$$

Se busca ζ de tal forma que $\zeta = P\tilde{L}^t x$

$$\zeta = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix}$$

esto significa que cada ζ_i coincide con x_j donde $i = j$.

Se busca H de tal forma que $H = \left(P\tilde{L}^t\right)^{-1}$

$$H = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}^{-1} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Construir la matriz *G*

$$G = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix}$$

Se calcula $c^t H$

$$\tilde{c} = \begin{bmatrix} -2 & 1 & 0 & 0 & 0 \end{bmatrix}$$

Calcular la matriz *AH*

$$AH = \begin{bmatrix} 1 & 2 & 3 & 1 & 0 \\ 2 & 1 & 1 & 0 & 1 \end{bmatrix}$$

Problema de Programación Cónica de Segundo Orden en forma matricial:

$$\hat{c} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 \end{bmatrix}$$

$$\hat{b}^t = \begin{bmatrix} 12 & 6 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\hat{x} = \begin{bmatrix} u & \zeta_1 & \zeta_2 & \zeta_3 & v & \zeta_4 & \zeta_5 & S_1 & S_2 & S_3 & y_1 & y_2 \end{bmatrix}$$

$$\hat{A} = \begin{array}{c} \begin{matrix} u & \zeta_1 & \zeta_2 & \zeta_3 & v & \zeta_4 & \zeta_5 & S_1 & S_2 & S_3 & y_1 & y_2 \end{matrix} \\ \begin{bmatrix} 0 & 1 & 2 & 3 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 2 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 \\ 1 & -2 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & -1 & 1 \\ 1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{array}$$

En este Capítulo se obtienen los resultados de mayor novedad científica de nuestro trabajo, los que se concretan en: la modificación del algoritmo de Factorización de Cholesky para que este sea eficiente aún siendo Q no Definida Positiva (DP) y la metodología para la transformación de un problema de QCP en un QCQP. Se muestra un ejemplo de la aplicación de este resultado.

Capítulo 4. Detalles de la implementación. Resultados Experimentales

En este Capítulo se exponen los resultados de la implementación computacional para la resolución de problemas del tipo LP y QCQP. Para el caso de LP se presenta una variante de un algoritmo de MPI que está basado en el método primal-dual de Barrera Logarítmica, con la variante del paso Predictor-Corrector y se implementa el algoritmo de [AS97] que es una generalización del método Predictor-Corrector de Mehrotra [Meh92] para problemas del tipo QCQP. Para la implementación de cada tipo de problema se utilizan diferentes parámetros, por ejemplo, en el caso de los problemas LP, los mismos se encuentran reflejados en el epígrafe 2.3 del Capítulo II, para el caso de QCQP se exponen en el epígrafe 4.4 los detalles de este algoritmo.

Las pruebas de mayor peso son los experimentos numéricos y las comparaciones entre software validados, sobre todo si se realizan las pruebas usando problemas reconocidos. Se pudieran citar muchos de ellos, accesibles en Internet, pero estos problemas se encuentran en formatos desconocidos para nosotros. Para resolver este problema, nos dimos a la tarea de generar aleatoriamente los problemas para nuestras pruebas con cuatro formatos distintos: Galois, Larx (software creados por nosotros), Storm y Mathematica. Se presentan resultados de los experimentos numéricos a diferentes dimensiones para cada tipo de problema. También se usaron para realizar los experimentos, ejercicios extraídos de libros de texto [JN99] [ZS84] y del artículo publicado en Internet [AD00]. En el presente capítulo se ofrece una comparación entre nuestros sistemas implementados y software reconocidos.

4.1 Experimentos Numéricos para el caso LP. Problemas pequeños

En estos problemas se muestra la superioridad de los métodos Simplex sobre los de Punto Interior.

Numero del Problema	*Cantidad de iteraciones*		*Solución*	*Cant. Variables*	*Cant. Restricciones*
	Galois	**Storm**			
1	19	2	43.42084	3	2
2	16	3	7.026881	4	2
3	16	3	3.947281	4	2
4	17	3	8.58185	4	2
5	16	2	13.5575	4	2
6	17	2	3.625953	3	2
7	18	2	37.10345	4	2
8	19	2	1.159617	4	2
9	17	2	6.330596	3	2
10	16	2	1.757143	3	2
Promedio	17.1	2.3	12.65111	3.6	2

Tabla 1 Comparación de los resultados con pequeños problemas.

Podemos ver que el Storm realizó entre 2 y 3 iteraciones para cada uno de estos problemas, mientras que el Galois necesitó entre 16 y 19, con un promedio de 17,1.

4.2 Experimentos Numéricos para el caso LP. Problemas medianos

En este caso analizamos problemas donde el Galois logra alcanzar la solución con un número menor de iteraciones que el Storm.

Problema	Cantidad de iteraciones		Solución	Cantidad de Variables	Cantidad de Restricciones
	Galois	Storm			
1	26	76	129.4433	28	28
2	28	95	134.8838	27	28
3	26	126	138.2372	28	28
4	28	106	115.411	27	28
5	27	108	106.1115	28	28

6	29	116	171.0478	28	27
7	27	140	53.04239	27	28
8	25	87	104.6487	27	27
9	27	108	103.2879	28	27
10	27	110	125.0589	27	28
11	27	96	236.1853	28	27
12	26	114	70.31507	28	28
13	26	147	74.12237	28	27
14	28	99	150.7636	28	28
15	42	113	18.86368	27	28
16	29	86	37.14934	27	27
17	27	88	46.62728	27	27
18	28	107	102.2874	28	27
19	27	82	127.8674	27	27
20	31	80	54.52178	28	27
21	26	98	48.69728	27	27
22	25	123	101.4366	28	28
23	27	108	40.34757	28	28
24	28	107	176.7126	27	27
25	26	103	122.0463	28	27
26	26	125	102.1747	28	27
27	33	122	54.4656	28	28
28	27	106	105.7706	27	28
29	26	71	90.43713	28	27
30	31	94	76.15268	28	27
Promedio	27.9	104.7	100.6039	27.6	27.5

Tabla 2. Comparación de los resultados con problemas medianos.

Podemos observar que para problemas entre 27 y 28 variables y con esta misma cantidad de restricciones el método de punto interior necesita un número mucho menor de iteraciones que el Storm para llegar a la solución óptima

Comparando los resultados de las tablas 1 y 2 vemos que los métodos de punto interior, a medida que aumenta la dimensión del problema, van tomando ventaja sobre el método Simplex.

4.3 Otros resultados

- Se probaron otros diez problemas pequeños de [ZS84] y nuestro programa devolvió idénticos resultados a los del libro [ZS84] con cantidad de iteraciones parecidas a las de la tabla 1.
- Se resolvieron diez problemas cercanos a tener 100 variables y 50 restricciones, tres problemas que generamos de alrededor de 200 variables y 100 restricciones, y finalmente se generó un problema de 1000 variables y 50 restricciones. En todos los casos nuestro programa obtuvo los mismos resultados que el paquete Mathematica. Todos estos problemas se crearon con nuestro generador.
- Todos estos resultados se pueden ver en Palencia y autores [GP01].
- En el Anexo 1 se muestra el Manuel de Usuario del software Galois que permite resolver problemas de LP usando una variante de un algoritmo de MPI.

4.4 Detalles del algoritmo de Alizadeh para el caso QCQP

Para resolver el problema QCQP se aplica el algoritmo de Alizadeh [AS97] y para su implementación también se utilizan las heurísticas y los parámetros de configuración definidos en l. En el Anexo 2 se muestran los pasos del mismo.

Se introducen las siguientes definiciones:

$$gap \overset{def}{=} x^t z$$

$$totalerr \overset{def}{=} gap + \|r_p\| + \|r_d\|$$

4.4.1 Solución del Sistema de Ecuaciones en el Algoritmo de Alizadeh

En el algoritmo de Alizadeh, que resuelve un problema QCQP para moverse de un punto solución a otro, se resuelve un sistema de ecuaciones. Para resolver dicho sistema formado por el complemento de Schur se puede usar la Factorización de Cholesky o la Factorización de Matrices Simétricas Indefinidas (FMSI), pero esto es posible solo si la matriz asociada a dicho sistema de ecuaciones es simétrica. Esta es una de las causas por la que se escogió el método de

Eliminación Gaussiana para resolver el sistema y evitarnos calcular la inversa del complemento de Schur añadiéndole a esto los errores de cálculo que esto trae consigo. La implementación del algoritmo FMSI se encuentra en [GP01] y los pasos del mismo se pueden ver en el Anexo 3.
En el algoritmo de Alizadeh [AS97] es necesario el cálculo del alfa (α) para el problema primal y beta (β) para el problema dual, a los cuales se les llama tamaño del paso.

4.4.2 Selección del Punto Inicial. Obtención del tamaño del paso

Selección del Punto inicial

Este algoritmo comienza con un punto no factible. Realizando un estudio del artículo [AS97] se toma $x_i = z_i = \sigma u_i$ y $y = 0$. El factor de escala σ, es el parámetro para esta heurística y se toma por defecto como $\sigma = n_Q$, donde n_Q es la cantidad de bloques cuadráticos.

Obtención del tamaño del paso.

Bloque Cuadrático: El tamaño del paso se calcula a partir de la restricción cónica, la cual se obtiene después de transformar el problema de Programación Cuadrática Convexa en uno de Programación Cónica de Segundo orden. Para obtener el tamaño del paso tanto en el problema primal como en el dual partimos de la restricción del cono cuadrático: $\sum_{i=1}^{r} \zeta_i^2 + v^2 \le u^2$

Hacemos las siguientes sustituciones en esta restricción:

- v por $(v + \alpha \Delta x_r)$
- u por $(u + \alpha \Delta x_0)$
- ζ_i por $(\zeta_i + \alpha \Delta x_i)$

$$\sum_{i=1}^{r} (\zeta_i + \alpha \Delta x_i)^2 + (v + \alpha \Delta x_r)^2 - (u + \alpha \Delta x_0)^2 \le 0$$

Realizando transformaciones algebraicas obtenemos la siguiente ecuación:

$$\alpha^2 \left[\sum_{i=1}^{r} \Delta x_i^2 + \Delta x_r^2 - \Delta x_0^2 \right] + 2\alpha \left[\sum_{i=1}^{r} \zeta_i \Delta x_i + v \Delta x_r - u \Delta x_0 \right] + \left[\sum_{i=1}^{r} \zeta_i^2 + v^2 - u^2 \right] \le 0$$

donde:

$$A=\sum_{i=1}^{r}\Delta x_i^2+\Delta x_r^2-\Delta x_0^2 \qquad B=\sum_{i=1}^{r}\zeta_i\Delta x_i+v\Delta x_r-u\Delta x_0 \qquad C=\sum_{i=1}^{r}\zeta_i^2+v^2-u^2$$

Por lo que la ecuación quedaría de la forma siguiente:

$\alpha^2 A+2\alpha B+C\le 0$, donde se tiene que cumplir que $C<0$ y alfa se calcula como:

$$\alpha_{1,2}=\frac{-B\pm\sqrt{B^2-4*A*C}}{2*A}$$

Pero aquí se presentan 5 casos que se tienen en consideración para dar el verdadero resultado de α , estos casos se explican a continuación:

Caso 1:

Hay una sola raíz en la ecuación y es positiva, $\alpha_1>0$ entonces se toma alfa como $\alpha=\alpha_1$.

Caso 2:

Hay una sola raíz en la ecuación y es negativa, $\alpha_1<0$ entonces se toma alfa como $\alpha=1$.

Caso 3:

Hay dos raíces en la ecuación, α_1 y α_2 donde se cumple que : $\alpha_1<0<\alpha_2$ entonces se escoge a alfa como : $\alpha=\min(\alpha_2,1)$.

Caso 4:

Hay dos raíces en la ecuación, α_1 y α_2 donde se cumple que: $\alpha_1<\alpha_2<0$ entonces se escoge a alfa como: $\alpha=1$.

Caso 5:

Hay dos raíces en la ecuación, α_1 y α_2 donde se cumple que: $0<\alpha_1<\alpha_2$ entonces se escoge a alfa como: $\alpha=\alpha_1$.

Octante Positivo:

Para resolver la ecuación $x_0+\alpha\Delta x_0\ge 0$

$\tilde{\alpha}=-\dfrac{x_0}{\Delta x_0}$ si $\Delta x_0<0$, sino $\tilde{\alpha}=1$

Para resolver la ecuación $x_i+\alpha\Delta x_i\ge 0$

$\hat{\alpha}=\min\left(-\dfrac{x_i}{\Delta x_i}\right)$, para todas las *i* donde se cumple que $\Delta x_i<0$ y además $i=\overline{1,n}$

Para calcular el *alfa*

$\bar{\alpha} = \min(\alpha, \tilde{\alpha}, \hat{\alpha})$

$alfa = \min(\bar{\alpha} * \tau, 1)$

Este es el alfa final que se obtiene para ser usado en la resolución del problema QCQP, donde τ es un número tomado en [AS97] a partir de experimentos numéricos. En este trabajo se toma a $\tau = 0.999$ aunque hay otros autores que proponen otros valores menos agresivos [DL84] [SW97], esto garantiza que *alfa* esté dentro del cono. De esta misma forma se calcula *beta* para el problema dual.

4.4.3 Heurísticas

Para la predicción de μ en el paso corrector se utiliza la heurística Mehrotra, por lo que se toma

$$\mu = \left(\frac{\hat{x}^t \hat{z}}{gap} \right)^3 \frac{gap}{N_S + n_Q}$$

Las demás heurísticas tratan sobre la terminación del algoritmo. Estas se dividen en dos clases: cuando se rechaza el nuevo punto porque no se puede progresar más y cuando se detiene el proceso porque el punto actual es suficientemente bueno.

Un nuevo punto es rechazado si:

1. α o β son menores que *steptol* . En este caso el tamaño del paso es muy pequeño para que el algoritmo progrese en un número razonable de iteraciones.
2. Uno de los bloques x o z no están en el cono factible. Dado el cálculo del largo del paso, debería ser imposible, esto indica que la precisión numérica no es suficiente para procesar una iteración correctamente.

Si el nuevo punto es aceptado, entonces las heurísticas de terminación tratan de determinar si el algoritmo debería detenerse. Existen cuatro razones para que el algoritmo termine.

1. La cantidad de iteraciones alcanza el máximo. El máximo número de iteraciones que el usuario puede gastar.
2. $\|x\|$ o $\|z\|$ es mayor que *bndtol* . Esto es tomado como indicación de que el problema primal o dual no está acotado.

3. El error total es pequeño en término absoluto y norma relativa de la solución.

$$totalerr < abstol$$
$$totalerr < reltol(1 + \|x\| + \|z\|)$$

4. El error total no puede ser mejorado en un número determinado de iteraciones. Esto sucede frecuentemente cuando el algoritmo no puede alcanzar la exactitud deseada.

4.4.4 Otros Parámetros

A continuación se exponen otros parámetros que influyen en el comportamiento del algoritmo:

1. $\tau = 0.999$: La fracción del paso limitado por el cono factible del algoritmo produce una rápida convergencia si la iteración es suficientemente cercana a la solución, pero puede causar que el algoritmo falle porque el paso se haga muy pequeño.
2. $\sigma = N_S + n_Q$: El factor de escala para el punto inicial.
3. $reltol = 10^{-11}$: El error total debe ser más pequeño que $(1 + \|x\| + \|z\|)$ multiplicado por un factor de *reltol* para que la solución pueda ser considerada satisfactoria.
4. $abstol = 10^{-8}$: Una solución es aceptada solamente si el error total es más pequeño que este valor.
5. $steptol = 10^{-8}$: Si α o β es más pequeño que este valor, el algoritmo se detiene porque el tamaño del paso es muy chiquito.
6. $bndtol = 10^{-8}$: La norma de la solución es menor que este valor. Si cualquiera de $\|x\|$ o $\|z\|$ excede este, entonces el problema es declarado no acotado.

4.5 Experimentos Numéricos para el caso QCQP

Los QCQP son analizados detalladamente en [AS97], donde se formulan condiciones de no degeneración y de complementariedad estricta y además se elabora para ellos un algoritmo primal-dual de punto interior, eficiente y numéricamente estable bajo condiciones de no degeneración de los problemas primal y dual y condiciones de complementariedad estricta. En este algoritmo se implementa una generalización del método precdictor-corrector de Mehrotra, adaptado para ser aplicado a CP definidos con conos cuadráticos de segundo orden (QCQP). En nuestro trabajo se elabora e implementa un algoritmo para QCP con dos fases, en la primera se transforma el problema en un QCQP utilizando el método propuesto en el epígrafe 3.4 y en la

segunda se resuelve el problema transformado usando el algoritmo propuesto en [AS97]. En el epígrafe anterior se exponen los parámetros, las heurísticas utilizadas, la selección del punto inicial y la obtención del tamaño del paso que se utilizaron en la implementación del algoritmo para validar los resultados numéricos. Se usan los vectores r_p y r_d para representar las violaciones de la factibilidad en el primal y el dual respectivamente. Por último se multiplica por la matriz H el vector de las primeras n componentes de la solución del QCQP y se tiene la solución del QCP. Este código fue denominado Larx.

4.5.1 Validación de los resultados

Para validar las dos fases de nuestro algoritmo, comparamos los resultados que se obtienen al aplicarlo 14 QCP de diferentes dimensiones, generados aleatoriamente, con los resultados que se obtienen al resolver estos mismos problemas con las funciones de optimización del Mathematica 5.5. Los resultados aparecen en la tabla siguiente:

TABLA No. 3

Problema			Larx		Mathematica	
m	n	Tipo	Objetivo	Seg.	Objetivo	Seg.
3	7	NDP	31.394	0.2	31.394	1.375
		NDP	**36.963**	0.2	**36.953**	0.547
		NDP	31.969	0.0	31.969	1.187
8	13	NDP	56.034	0.4	56.034	4.782
		NDP	**225.680**	0.8	**230.598**	4.422
4	8	DP	**37.718**	0.4	**37.716**	1.266
		DP	51.166	0.2	51.166	1.625
12	17	DP	**538.030**	2.7	**541.314**	4.907
			537.420	1.4	**538.189**	4.953
			424.130	1.9	**424.133**	4.187
10	20	DP	**345.110**	3.2	**345.037**	13.641
		DP	**729.280**	4.0	**729.270**	18.938

15	25	NDP	**730.840**	11	**730.838**	28.703
		NDP	**927.200**	12	**927.193**	34.86
		NDP	792.880	11	792.880	23.688

Donde m es el número de restricciones y n es el número de variables. Todos los problemas tienen matrices semidefinidas positivas, en la tabla DP significa que dicha matriz es además definida positiva y NDP que no lo es. Para cada uno de los códigos aparece el valor de la función objetivo obtenido y el tiempo de ejecución expresado en segundos. .

Como puede apreciar los valores de las soluciones obtenida son bastante similares, hay 5 pequeñas diferencias que favorecen al Mathematica y 4 diferencias, no todas tan pequeñas, que favorecen al Larx. Casi todas debidas a errores de aproximación al calcular el valor final de la función objetivo, pues los vectores soluciones son iguales.

Los resultados obtenidos al resolver con el Larx otros 45 QCP generados aleatoriamente se resumen en la tabla siguiente:

TABLA No. 4

m	n	Tipo	N	**R.E**.	**M.E**.	**M.I.**	**M.T.**
3	7	NDP	3	E-6,E-9	7.5E-7	27	0.03
4	8	DP	3	E-7,E-12	2.7E-7	73.6	0.2
8	9	DP	4	E-7,E-6	1.7E-7	28.5	0.225
8	12	DP	3	E-8,E-13	3.3E-8	132.3	0.67
8	13	NDP	3	E-10,E-12	2.3E-10	142	1.43
9	13	DP	5	E-7,E-10	8.4E-8	186.6	1.08
9	14	DP	5	E-6,E-11	6,5E-7	156,6	1.14
9	14	NDP	3	E-9,E-11	2.9E-9	151,7	1.06
10	15	NDP	5	E-7,E-12	4.3E-8	318.2	9.9
10	20	DP	5	E-8,E-11	5.1E-9	246.6	5.46
12	19	NDP	5	E-7,E-12	1.2E-7	253	4.38
15	25	NDP	1	E-5,	4.9E-5	312	33

Las tres primeras columnas tienen el mismo significado que en la tabla anterior y caracterizan a los problemas cuyos resultados aparecen en cada fila. N es el número de instancias del problema analizado en la fila. En R.E. aparecen los lugares decimales de la primera cifra no nula en el

mayor y el menor error obtenido en los problemas de la fila. M.E. es el error promedio, M.I. es el número promedio de iteraciones y MT el tiempo promedio de ejecución de los problemas de la fila. Las columnas ME y MI reflejan la calidad de las soluciones y la eficiencia del Larx. Estos resultados obtenidos se pueden ver en Palencia y autores[GP07][GP08]. En el Anexo 4 se muestra el Manual de Usuario del software Larx que permite resolver problemas QCP.

Conclusiones

Como resultado de nuestro trabajo se pueden obtener las siguientes conclusiones:

1. Se dominan los fundamentos teóricos para la implementación de algoritmos de punto interior, al igual que los aspectos computacionales, ya que se elaboró un software capaz de resolver problemas lineales y cuadráticos convexos usando métodos de punto interior.
2. También se obtiene una modificación del algoritmo recursivo de Cholesky, que permite factorizar matrices semidefinidas positivas sin incrementar el costo computacional en las no definidas positivas. Con ayuda de esta factorización se logra transformar cualquier problema de Programación Cuadrática Convexa en un Problema de Programación Cónica de Segundo Orden, aún cuando la matriz de la forma cuadrática no sea definida positiva. Lo anterior permite aplicar una generalización del algoritmo predictor-corrector de Mehrotra de Punto Interior, para resolver el Problema Cuadrático Convexo. Estos resultados fueron validados por las pruebas numéricas realizadas.
3. Aunque los experimentos fueron realizados con problemas a pequeña escala para los QCP con matriz SDP, los resultados obtenidos nos permiten conjeturar que este procedimiento de transformación nos ofrece una alternativa para poder aplicar los eficientes Métodos de Punto Interior a la solución de Problemas Cuadráticos Convexos a gran escala, incluyendo los casos en que estos problemas no sean estrictamente convexos.

Recomendaciones

1. Realizar pruebas numéricas con ejemplos de la práctica social.
2. Realizar pruebas numéricas con problemas de mayor dimensión, pues en los problemas de mayor tamaño es donde mejor se pueden apreciar las ventajas de los métodos de Punto Interior sobre los métodos clásicos de programación matemática.
3. Analizar los procesos para la búsqueda de puntos iniciales u otra variante para buscar el punto inicial del QCQP.

Referencias bibliográficas

[AS97] F. Alizadeh and S. H. Schemieta. Optimization whit semidefinite, quadratic and linear constraints. Rutgers Center for Operations Research (RUTCOR). 1997.

[AG03] Alizadeh F., Goldfarb D.(2003): Second-Order Cone Programming, RUTCOR

[AT02] Andersen E.D., Ross C., Terlaky T.(2002): On implementing a primal-dual interior-point method for conic quadratic optimization, Springer Verlag.

[AD00] Arbenz P, Dramac Z.(2000): On Semi-Definite Positive Matrices whith null space known, ETH Zürich, Computer Science Department, Technical Report Research #352, November.

[BN01] Ben-Tal A., Nemirovski A.(2001): Lectures on Modern Convex Optimization: Analysis, Algorithms and Engineering Applications, MPS/SIAM, Series on Optimization, SIAM.

[DL84] Luenberger, David G.,"Programación Lineal y no Lineal";Stanford University, California, 1984, USA

[Gant59] Gantmacher F.R.(1959): The Theory of Matrices, Chelsea, New York.

[GT01] Gould N.I.M, Toint P., (2001): The state-of-the-art in numerical methods for quadratic programming. 19th Biennial Conference on Numerical Analysis, Dundee, Scotland, 28th June 2001.

[GP01] Palencia F. G., Hing C. R, Tapanes G. G.,(2001), "Galois – Un paquete para resolver problemas de Programación Lineal usando un Método de Punto Interior." (Registro de Software, Habana-2001).

[GP07] Palencia F. G., Hing C. R, Medina S. D., Rojas C. M.,(2007), "Larx2 – Software para solucionar Problemas de Programación Cuadrática Convexa". (Registro de Software, Habana-2007).

[GP08] Palencia F. G., Hing C. R, Medina S. D., Correa R. M., "Métodos de Punto Interior para la Optimización Cuadrática Convexa con Matrices no Definidas Positivas", Revista de Matemática: Teoría y Aplicaciones. 2008 15(1) , 1-10, ISSN: 1409-2433.

[Hig96] Higham N.(1996): The Symmetric Indefinite Factorization: Stability an Applications in Optimization.

[JN99] Nocedal J., Wright S.J.(1999):Numerical Optimization, Springer Series in Operations Research, Springer Verlag New York.

[Kakes02] Kakes C. Alibeith," Matrices Escalonadas y Métodos Primal-Dual de Punto Interior", Tesis Doctoral, Dpto. Matemática Aplicada. Facultad de Matemática y Computación. UH 2002.

[Meh92] Mehrotra S. On the implementation of a primal-dual interior point method. SIAM J. on Optimization 2 (4) 575 -- 601, 1992.

[MK87] Murty, Kabadi(1987): Some NP-Complete problems in Quadratic and Nonlinear Programming, Mathematical Programming 39.

[NS96] Nemirovski A., Sherinberg K.(1996): Extension of Karmarkar algoritm onto Convex Quadratically Constraints Quadratic Programming, Mathematical Programming 72.

[NS97] Nesterov Y.E., Todd. M.J.(1997): Self-Scaled Barriers and Interior-Point Methods for Convex Programming. Mathematics of Operations Research., 22(1):1-42, February.

[NT98] Nesterov Y.E., Todd. M.J.(1998): Primal-dual interior-point methods for self-scaled cones. SIAM J. Optimization., 8:324-364.

[PS82] Papadimitrou, C. H., and Steiglitz K.,"Combinatorial Optimization: Algorithm and Complexity". MIT, Prentice-Hall, USA, 1982

[Ren01] Renegar J. (2001): A Mathematical View of Interior-Point Methods in Convex Optimization, MPS-SIAM Series in Optimization.

[RT97] Ross C., Terlaky T., Vial J.(1997): Theory and Algorithms for Linear Optimization an interior point approach, John Wiley and Sons.

[SW97] Stephen J. Wright. Primal-Dual Interior-Point Methods. SIAM. 1997.

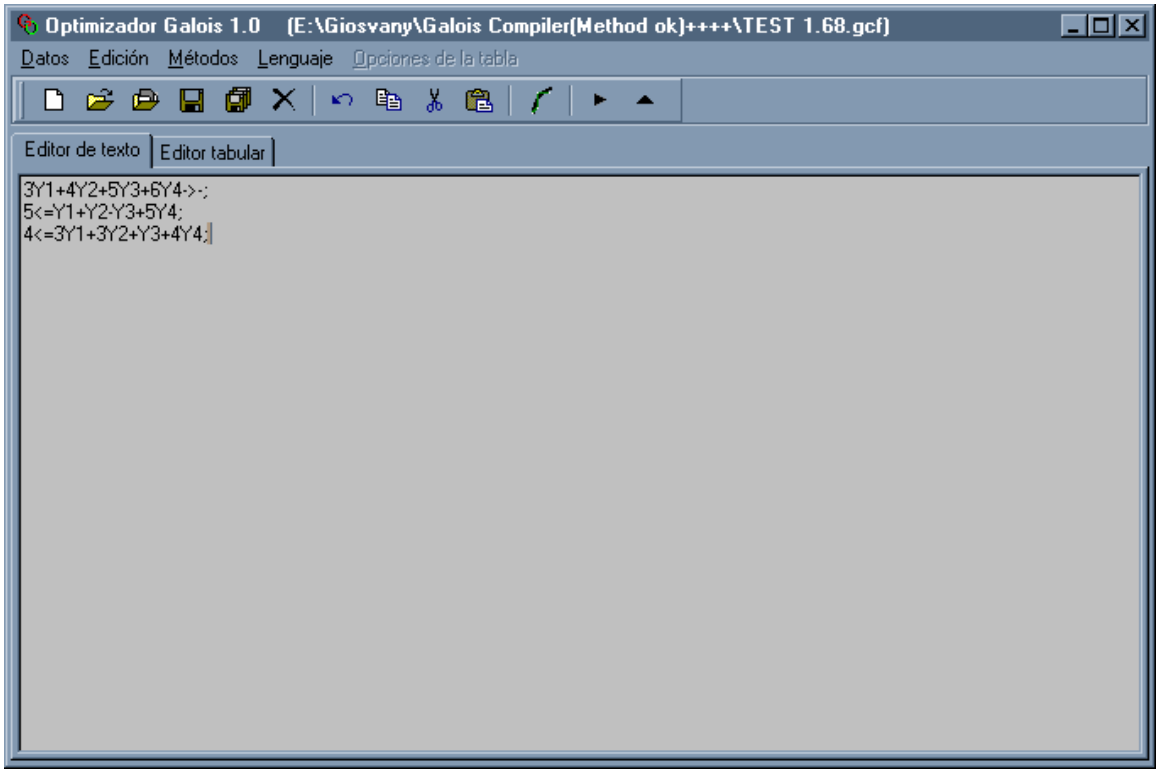
[Van02] Vandenberghe L.,(2002): The Cholesky factorization, EE103 Winter, Germany.

[ZS84] Zaychenko, Yu. P., Shumilova, C. A., "Investigación de Operaciones". Manual de Ejercicios, Kiev, 1984.

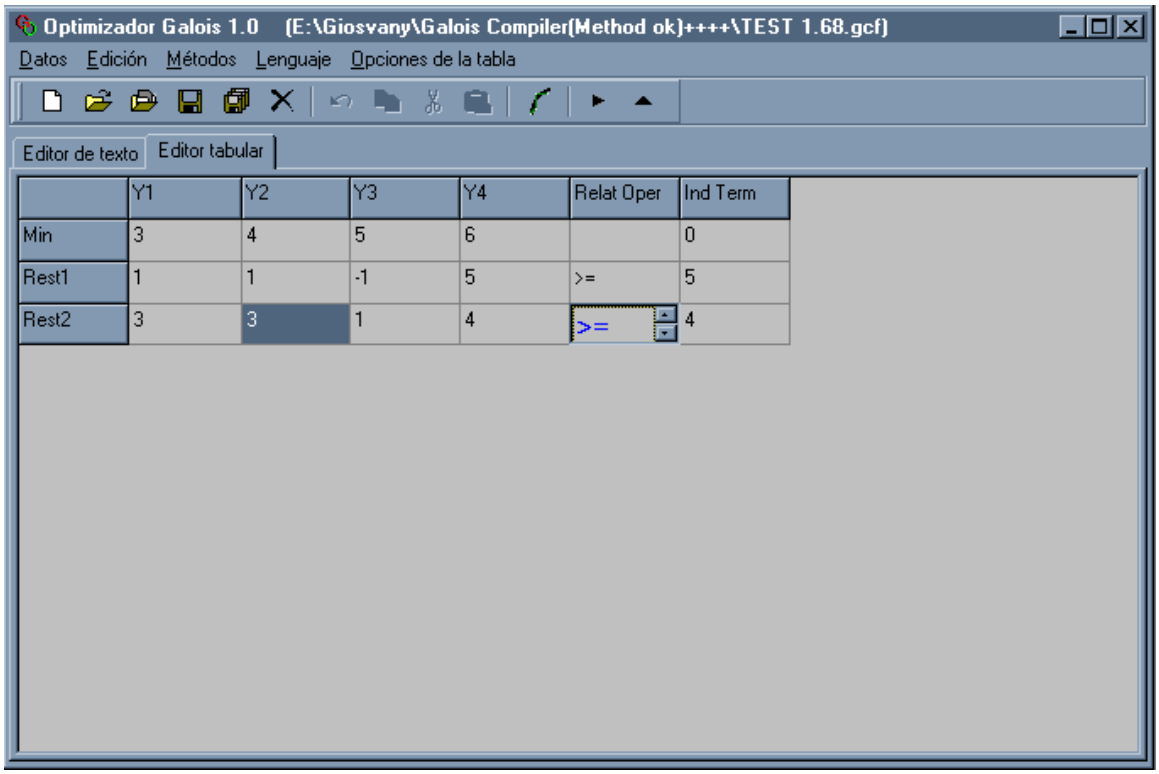
Anexo 1. Manual de Usuario. El Sistema Galois Optimizer

Con este material pretendemos facilitar aun más el trabajo con nuestro sistema. Se comienza mostrando el logotipo de nuestra aplicación el cual aparece durante cinco segundos antes de que aparezca el ambiente de trabajo de la aplicación.

Cuando aparece el ambiente, ya se está listo para editar problemas en modo texto.



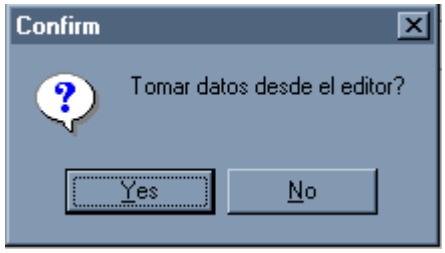
Y puede cambiarlo a el editor tabular si lo desea pulsando clic en el tab edición tabular.



.

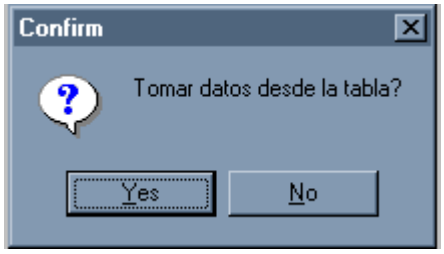
De donde puede retornar al modo texto si lo desea pulsando clic al tab Editor de texto.

Cuando se cambia del modo texto al modo tabular aparece el siguiente diálogo:



Si usted oprime Yes, y hay un problema correctamente escrito en el editor de texto, se traslada el problema del editor de texto al editor tabular, si está mal escrito, se le muestra un mensaje señalándole el error que contiene su problema. Si usted oprime No, aparece el editor tabular vacío.

Cuando se cambia del modo tabular al modo texto se muestra el siguiente diálogo:



Oprimiendo Yes, un problema editado correctamente en el editor tabular es llevado al editor de texto. De tener algún error el problema, entonces se muestra un mensaje con el error que se ha cometido en la tabla. Pulsando clic en No, aparece un problema vacío en el editor de texto.

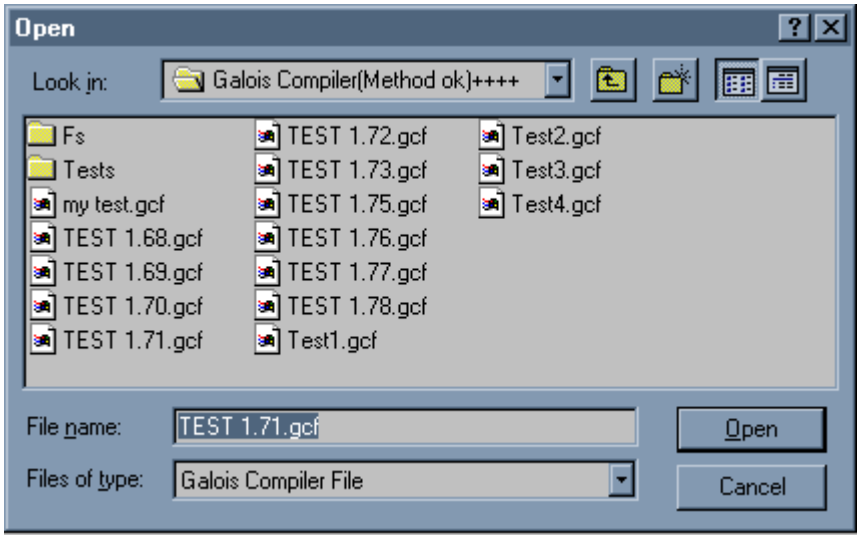
Ahora damos paso a las opciones que aparecen en el menú principal.

Comenzamos por la opción datos, al seleccionarla aparecen las siguientes opciones:

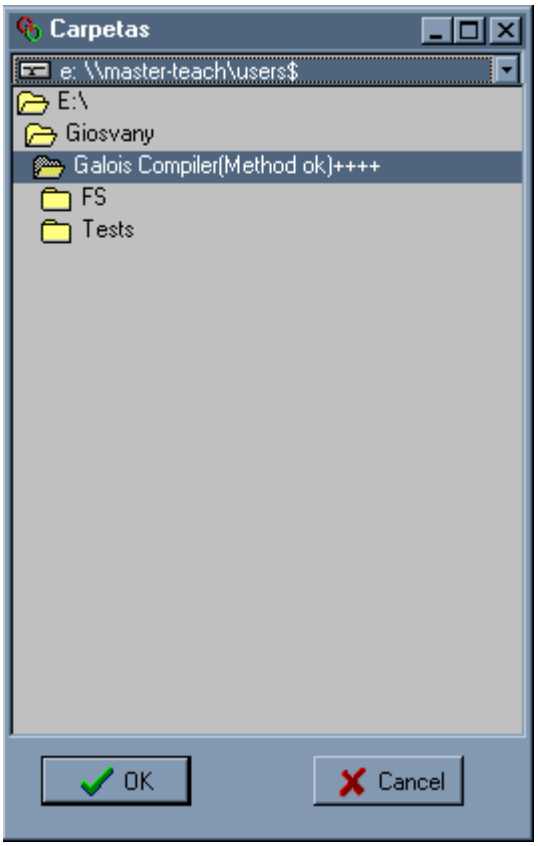
Este menú datos se concibió para darle facilidades al usuario en lo referente a la manipulación de los ficheros de entrada de datos al sistema.

Sus funciones son:

- Nuevo: Limpia ambos editores, el de texto y el tabular.
- Abrir: Carga un fichero de extensión gcf desde el disco usando el siguiente diálogo:



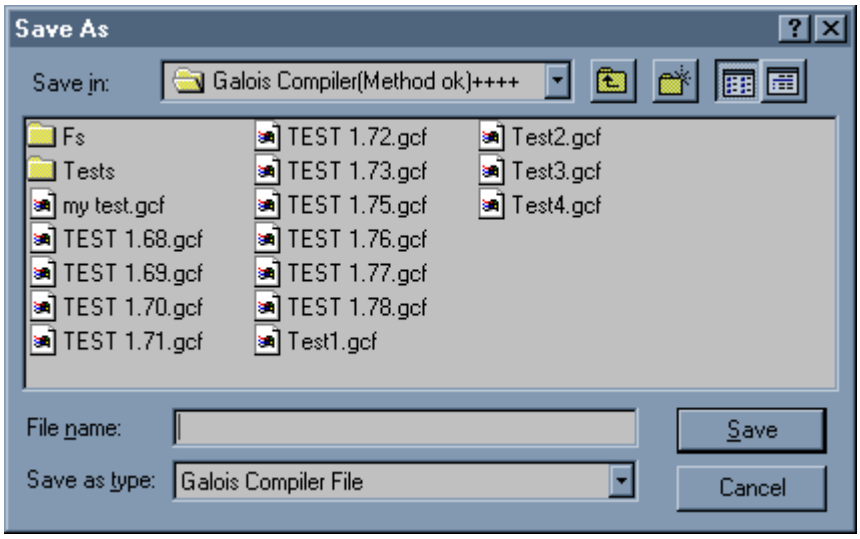
- Abrir carpeta: Resuelve todos los problemas con extensión gcf en el directorio especificado en el siguiente diálogo:



Esta opción nos brinda la posibilidad de resolver a la vez una gran cantidad de problemas escritos en el lenguaje Galois. El cuadro combinado en la parte superior del diálogo nos permite seleccionar la unidad en que se encuentran los datos. Si los datos están por la red, debe utilizar la opción del sistema: "Conectarse a unidad de red (Map network drive)". Debajo entonces usted puede seleccionar la carpeta que contiene los problemas, los cuales deben tener extensión gcf, si esta carpeta tiene subcarpetas con problemas, estos también se resolverán. Las soluciones a los problemas se pueden encontrar en la misma carpeta con el mismo nombre que el problema del que es solución y con extensión grf.

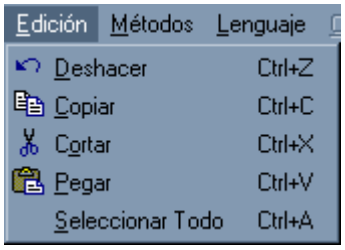
- Guardar: Si el problema que se está editando ha sido cargado de fichero o guardado anteriormente, entonces se guarda en la dirección donde se guardó o de donde ha sido

cargado. En otro caso, aparece el siguiente diálogo para que usted especifique la dirección donde guardarlo.



- Guardar como: Aparece el diálogo anterior para que usted especifique la dirección para guardar el problema que se está editando.
- Cerrar: Aborta la ejecución de la aplicación.

Ahora procedemos a mostrar las opciones del menú edición.



El menú edición está diseñado para facilitar el trabajo cuando se edita el problema en modo texto.

- Deshacer: Deshace la última acción que se realizara en el editor.
- Copiar: Almacena en el clipboard el texto seleccionado en el editor.
- Cortar: Borra el texto seleccionado en el editor y lo almacena en el clipboard.
- Pegar: Inserta en la posición del cursor en el editor el texto que se encuentra en el clipboard.
- Seleccionar todo: Selecciona todo lo que se encuentra en el editor

Menú lenguaje:

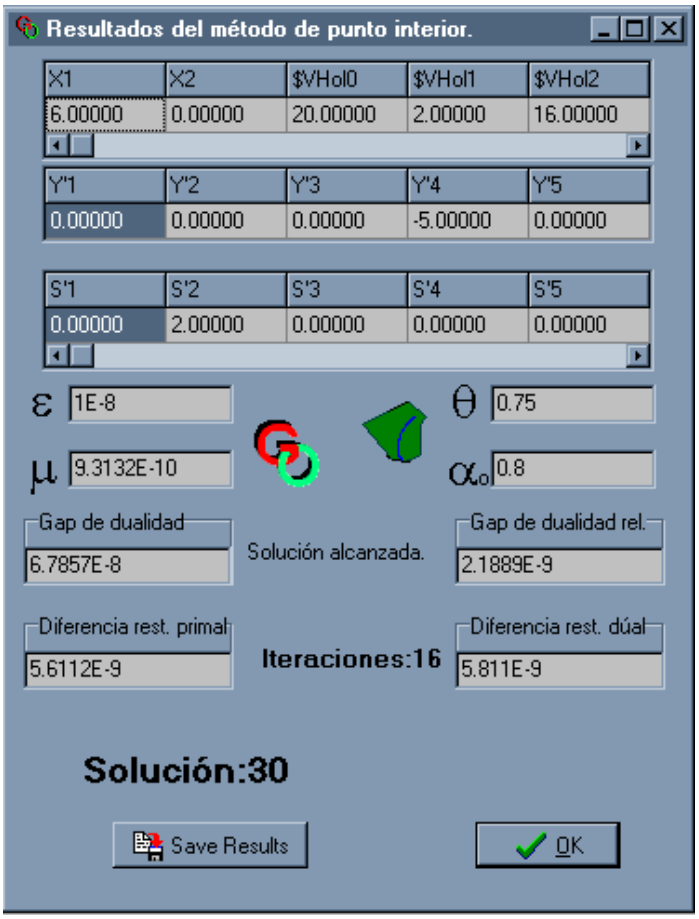
Esta herramienta fue diseñada con el objetivo de ayudar a los usuarios a comunicarse con el sistema en la lengua que mejor domine.

- Inglés: Los mensajes de la aplicación serán a partir de este momento en inglés.
- Español: Cambia la interfaz de la aplicación para el idioma español, a no ser los diálogos predefinidos por el sistema.

Sin dudas, el menú más importante de nuestra aplicación es el que mostramos a continuación:

Agrupa los métodos con que cuenta nuestro sistema.

- Punto Interior: Resuelve el problema que se encuentra en el editor de texto si no contiene errores, usando nuestro método de punto interior. El resultado se muestra en el siguiente formulario:



Donde de arriba hacia abajo la primera tabla es la solución del problema planteado. La segunda y la tercera son las soluciones del dual correspondiente al problema. El resultado de la función objetivo se muestra en el campo Solución y la cantidad de iteraciones en el campo Iteraciones El botón Save Results es para guardar el contenido de este formulario. Aparece un diálogo, donde los resultados se pueden salvar con extensión *grf*.

También se tiene que:

- ε: La cota de error que se usa para calcular resolver el problema.
- θ: Parámetro de actualización de barrera.
- μ: Parámetro de barrera.
- α_0: Factor de reducción del tamaño del paso de Newton.
- Diferencia rest. dual: Nos dice cuan cerca no encontramos de la región factible del dual.
- Diferencia rest. primal: Nos dice cuan cerca nos encontramos de la región factible del primal.
- Gap de dualidad: Nos indica cuan cerca se encuentran los resultados del problema y su dual.
- Gap de dualidad rel.: Nos indica la distancia entre las soluciones del problema y su dual en relación con la norma del resultado del problema.

El menú Opciones de la tabla solo está activo cuando se activa el editor tabular, y sus opciones son:

Diseñado para introducir las dimensiones del problema cuando se trabaja con el editor tabular.

- Cantidad de variables: Aparece un diálogo sencillo para cambiar la cantidad de variables en el editor tabular.
- Cantidad de rest…: Opción para cambiar la cantidad de restricciones en el editor tabular.

Finalmente les mostramos la barra de herramientas donde se accede a casi todas las opciones de nuestra aplicación. Sus iconos coinciden con los de su opción equivalente en el menú principal.

Anexo 2. Pasos del Algoritmo de Alizadeh

En este anexo se muestran los pasos del algoritmo [AS97] implementado en el software Larx 2.0.

Pasos del algoritmo.

1. Inicialización $y \leftarrow 0 \quad z_j = x_j \leftarrow \sigma\mu_j \quad j = 1\ldots, n$

2. Paso predictor

   a) Computar los residuales con $\mu = 0$

   $$r_d \overset{def}{=} c - A^t y - z$$
   $$r_p \overset{def}{=} b - Ax$$
   $$r_c \overset{def}{=} \mu u - XZu$$

   b) Forma y factor del complemento Schur $AZ^{-1}XA^T$

   c) Computar $\Delta x, \Delta y, \Delta z$

   $$\Delta y = (AZ^{-1}XA^t)^{-1}(r_p + AZ^{-1}(Xr_d - r_c))$$
   $$\Delta z = r_d - A^t \Delta y$$
   $$\Delta x = -Z^{-1}(X\Delta z) - r_c$$

   d) Tomar $\bar{\alpha}(\bar{\beta})$ el máximo tamaño del paso tal que todos los bloques $x + \bar{\alpha}\Delta x (z + \bar{\beta}\Delta z)$ estén dentro del cono respectivo.

   e) $\alpha \leftarrow \min(1, \tau\bar{\alpha}) \quad \beta \leftarrow \min(1, \tau\bar{\beta})$ y tomar

   $$\hat{x} \leftarrow x + \alpha\Delta x \quad \hat{y} \leftarrow y + \beta\Delta y \quad \hat{z} \leftarrow z + \beta\Delta z$$

3. Paso corrector

   a) Computar residuales

   $$r_d \overset{def}{=} c - A^t y - z$$
   $$r_p \overset{def}{=} b - Ax$$
   $$r_c \overset{def}{=} \mu u - XZu$$

   con

$$\mu = \left(\frac{\hat{x}^T \hat{z}}{x^T z} \right)^3 \frac{x^T z}{N_S + n_Q} \quad \text{y} \qquad r_c \overset{def}{=} \mu u - XZu - \Delta X \Delta Z u$$

b) Computar $\Delta x, \Delta y, \Delta z$

$$\Delta y = (AZ^{-1}XA^t)^{-1}(r_p + AZ^{-1}(Xr_d - r_c))$$
$$\Delta z = r_d - A^t \Delta y$$
$$\Delta x = -Z^{-1}(X\Delta z) - r_c$$

y la factorización del complemento Schur obtenida en 2b.

c) Tomar el máximo tamaño del paso $\bar{\alpha}(\bar{\beta})$ tal que todos los bloques $x + \bar{\alpha}\Delta x (z + \bar{\beta}\Delta z)$ estén dentro del cono respectivo.

d) $\alpha \leftarrow \min(1, \tau\bar{\alpha})$ $\beta \leftarrow \min(1, \tau\bar{\beta})$ y tomar

$$x \leftarrow x + \alpha\Delta x \quad y \leftarrow y + \beta\Delta y \quad z \leftarrow z + \beta\Delta z$$

4. Prueba de aceptación: el nuevo punto es rechazado si uno de los bloques no está en el cono. La mejor iteración es retornada como la solución y el algoritmo termina.

5. Prueba de terminación: el algoritmo termina con el nuevo punto como la solución si el error total ($gap + \|r_p\| + \|r_d\|$) es pequeño o uno de $\|x\|$ y $\|z\|$ es grande. En el segundo caso el problema es asumido como no acotado.

6. Regresar a 2.

Anexo 3. Pasos del Algoritmo FMSI

Partiendo de que se quiere factorizar una matriz simétrica $A \in \Re^{nxn}$ para lo cual se conoce que cualquier matriz simétrica puede expresarse de la siguiente manera:

$$PAP^t = LBL^t$$

donde *L* - es una matriz triangular inferior en cuya diagonal contiene el elemento unidad; *B* es una matriz diagonal de bloque cuyos bloques solo pueden ser de $(1x1)$ ó de $(2x2)$, con esta matriz se garantiza que siempre exista la factorización y podrá ser computada por procesos numéricos estables [JN99]; *P* es la matriz de permutación.

El algoritmo que usaremos para factorizar la matriz *A* es el siguiente:

Algoritmo

Paso 1

Hallar una matriz *E*, que será la matriz pivote de dimensión 1 ó 2. Esta matriz se halla en el algoritmo BBK (el cual se presentará más adelante) que permite hallar la matriz de permutación *P*.

Paso 2

Se obtiene la matriz *P* tal que se cumple que $PAP^T = D$, y *D* es *A* pero con filas y columnas intercambiadas debido a la multiplicación por la matriz de permutación *P* que no es más que la matriz identidad con filas y columnas intercambiadas. Entonces resulta que PAP^t se puede representar de la siguiente forma:

$$PAP^t = \begin{bmatrix} E_{sxs} & C^t{}_{(n-s)xs} \\ C_{sx(n-s)} & H_{(n-s)x(n-s)} \end{bmatrix}$$

Donde *E* es la matriz pivote, *C* y *H* son matrices.

Paso 3

Realizar la factorización de tal manera que las matrices *L*, *B* y L^t se calculan de la siguiente forma:

$$PAP^t = \begin{bmatrix} I_{sxs} & 0 \\ CE^{-1} & I_{(n-s)x(n-s)} \end{bmatrix} x \begin{bmatrix} E & 0 \\ 0 & H - CE^{-1}C^t \end{bmatrix} x \begin{bmatrix} I_{sxs} & E^{-1}C^t \\ 0 & I_{(n-s)x(n-s)} \end{bmatrix}$$

Donde se cumple que $E^{-1}C^t = \left(CE^{-1}\right)^t$.

Paso 4

Verificar que se cumple que $PAP^t = LBL^t$, donde *L* es una matriz triangular inferior y *B* es una matriz diagonal de bloques de $(1x1)$ ó $(2x2)$. Si esta condición no se cumple se regresa al paso 1, usando en este caso la matriz $H - CE^{-1}C^t$ para factorizar, de lo contrario se va al Paso 5.

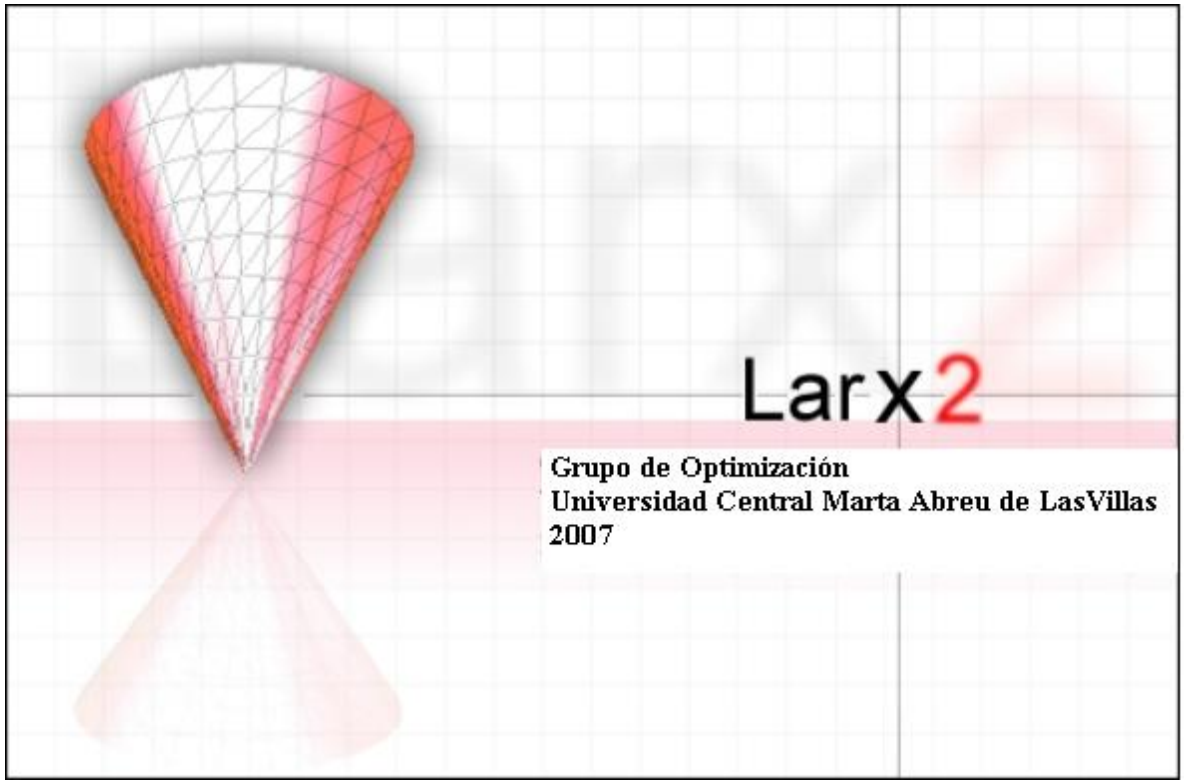
Paso 5

Integrar todas las factorizaciones de las submatrices que se realizaron en los pasos anteriores del algoritmo de forma tal que queden tres matrices de la forma deseada, que serán las matrices que conformarán la factorización de la matriz *A.*

Anexo 4. Manual de Usuario. El Sistema Larx 2.0

Manual de usuario.

Primeramente se mostrará el splash de nuestro software durante unos pocos segundos:



A continuación se presentarán todas las opciones del software comenzando por la ventana principal que se nuestra a continuación del splash:

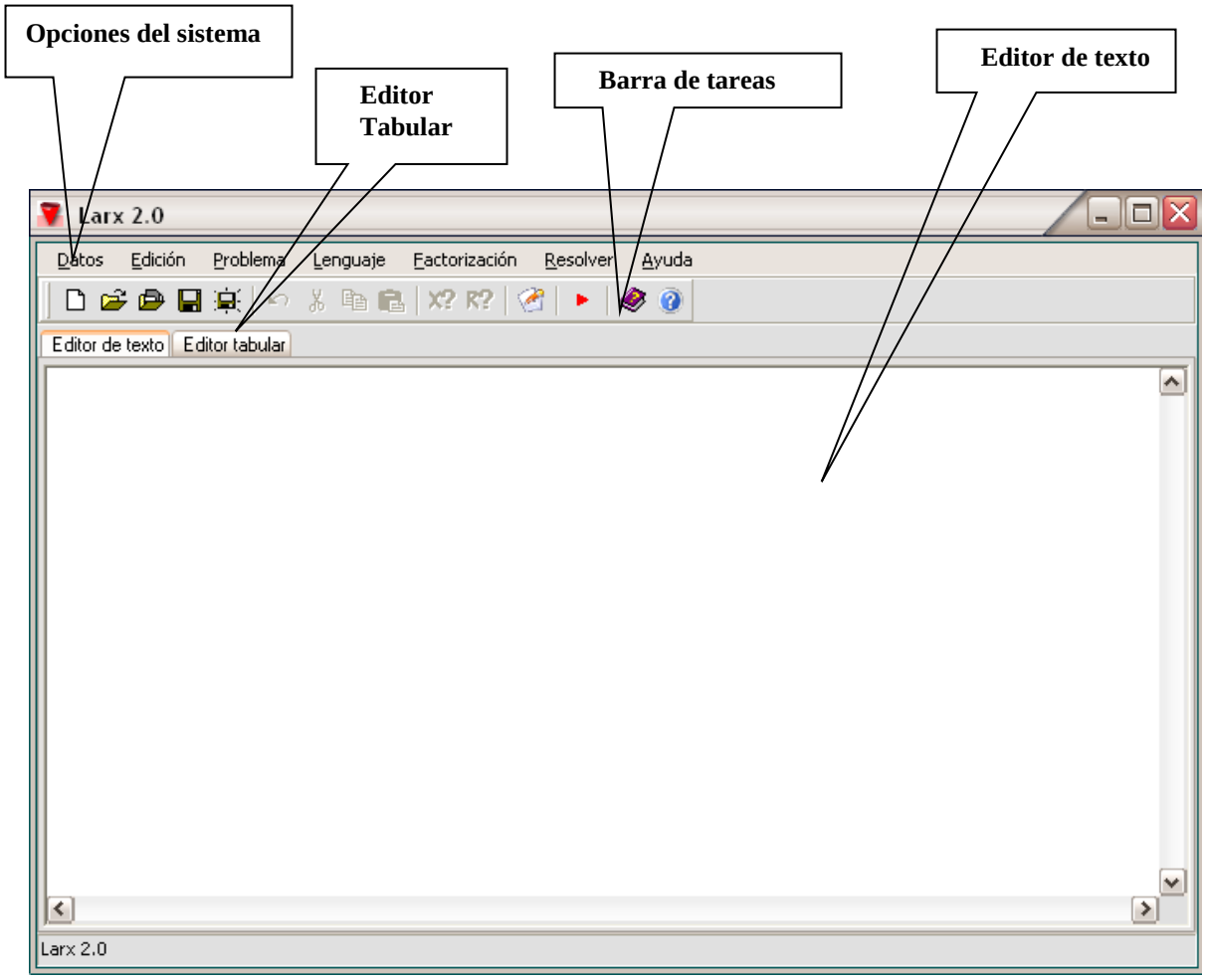


Fig. 1: Ventana principal. Editor de texto.

Esta ventana está conformada por los elementos siguientes: el menú de opciones del sistema, la barra de tareas, el editor de texto y el editor tabular. El menú de opciones del sistema y la barra de tareas están muy relacionados, siendo la barra de tareas un acceso directo a las acciones más importantes del menú opciones del sistema.

El Editor de texto es la parte donde se introduce el problema en forma de texto como se muestra en la Fig. 1.

El Editor tabular le brindar al usuario la opción de introducir el problema en forma matricial, este tiene la apariencia que se muestra en la Fig. 2.

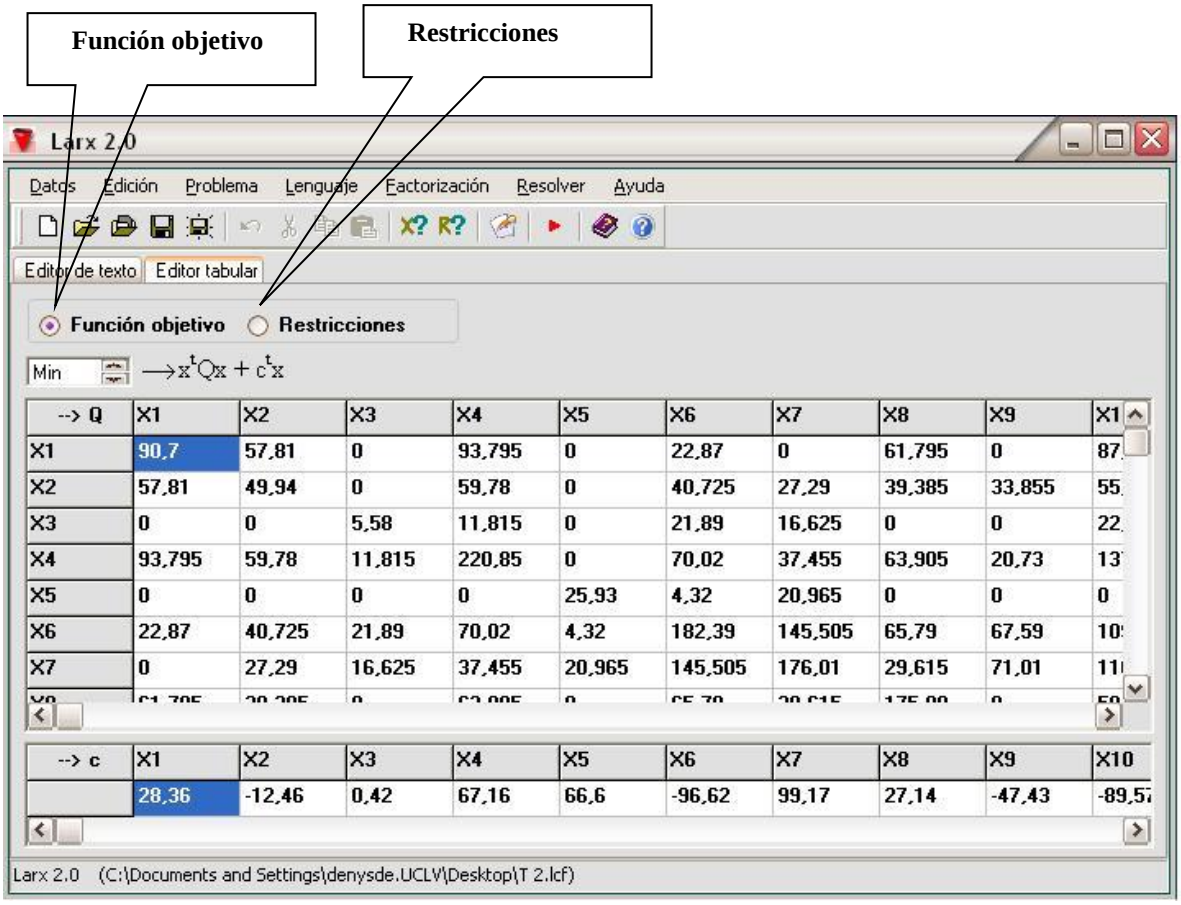
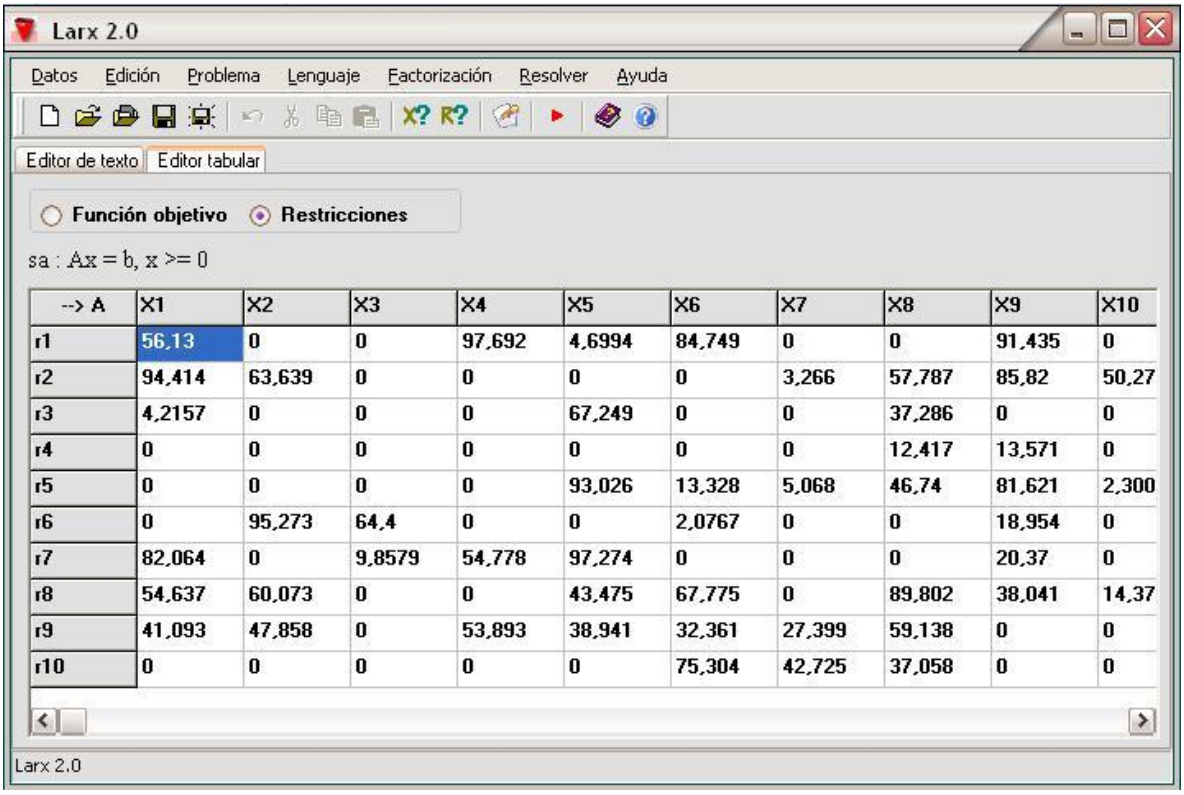


Fig. 2: Ventana principal. Función objetivo en el Editor tabular.

En este editor tabular podemos disfrutar de dos opciones:

Función objetivo: ver los datos de la función objetivo del problema dividida en dos partes , la matriz asociada a la parte cuadrática de la función objetivo que es la matriz *Q* y la parte lineal que es el vector *c*, lo cual lo podemos lograr marcando el radio button que se muestra en la Fig. 2.

Restricciones: ver la matriz de restricciones asociada al problema, lo cual lo podemos lograr marcando el radio button Restricciones. Ver Fig. 3.



--> A	X1	X2	X3	X4	X5	X6	X7	X8	X9	X10
r1	56,13	0	0	97,692	4,6994	84,749	0	0	91,435	0
r2	94,414	63,639	0	0	0	0	3,266	57,787	85,82	50,27
r3	4,2157	0	0	0	67,249	0	0	37,286	0	0
r4	0	0	0	0	0	0	0	12,417	13,571	0
r5	0	0	0	0	93,026	13,328	5,068	46,74	81,621	2,300
r6	0	95,273	64,4	0	0	2,0767	0	0	18,954	0
r7	82,064	0	9,8579	54,778	97,274	0	0	0	20,37	0
r8	54,637	60,073	0	0	43,475	67,775	0	89,802	38,041	14,37
r9	41,093	47,858	0	53,893	38,941	32,361	27,399	59,138	0	0
r10	0	0	0	0	0	75,304	42,725	37,058	0	0

Fig. 3: Ventana principal. Restricciones en el Editor tabular.

A continuación se realizará una breve explicación de cada uno de los menús que pertenecen a las Opciones del sistema.

Menú Datos:

Fig. 4: Menú Datos.

En este menú se realizan las operaciones tradicionales como son:

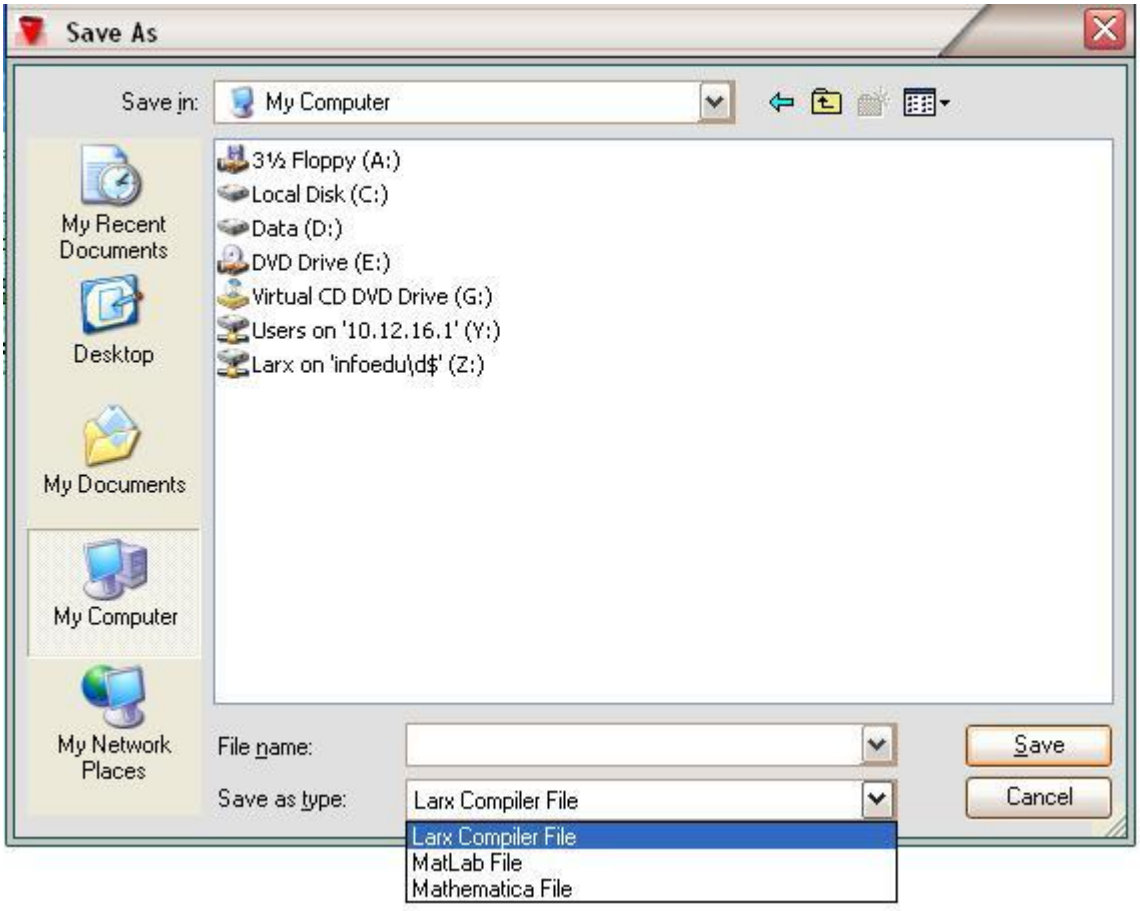
1- Crear un nuevo documento donde introducir un nuevo problema.

2- Abrir un problema existente.

3- Abrir una carpeta.

4- Guardar el problema.

5- Guardar el problema con otro nombre. Además puede guardar el problema cargado, en el formato de MatLab o en el formato del Mathematica, para facilitar el trabajo en caso de que se quiera comparar los resultados obtenidos en Larx 2.0 con los obtenidos en dichos software.



6- Salir del software.

Menú Edición:

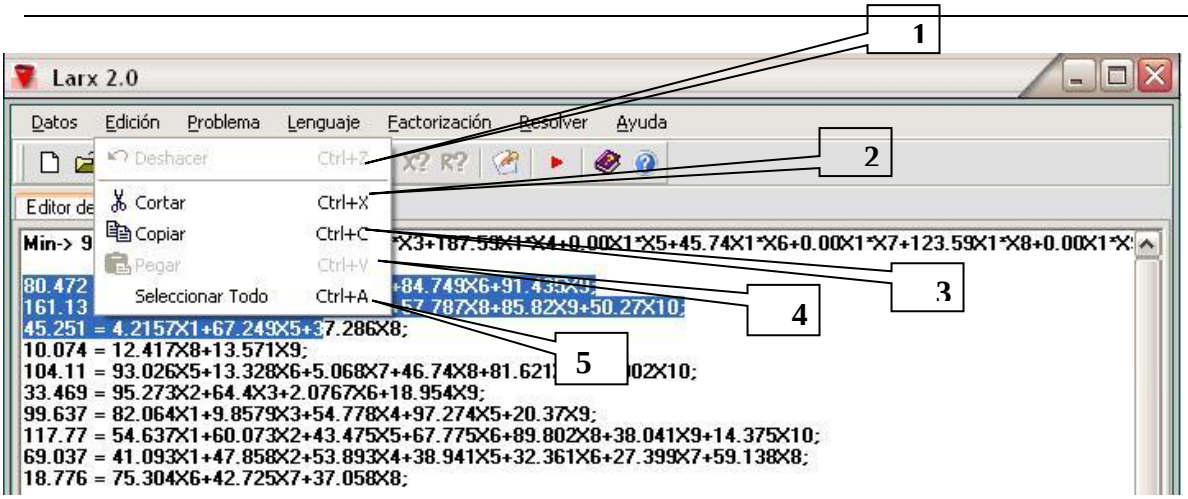


Fig. 5: Menú Edición.

En este menú se realizan las operaciones tradicionales que proporciona un editor de texto:

1- Deshacer algún cambio indeseado.

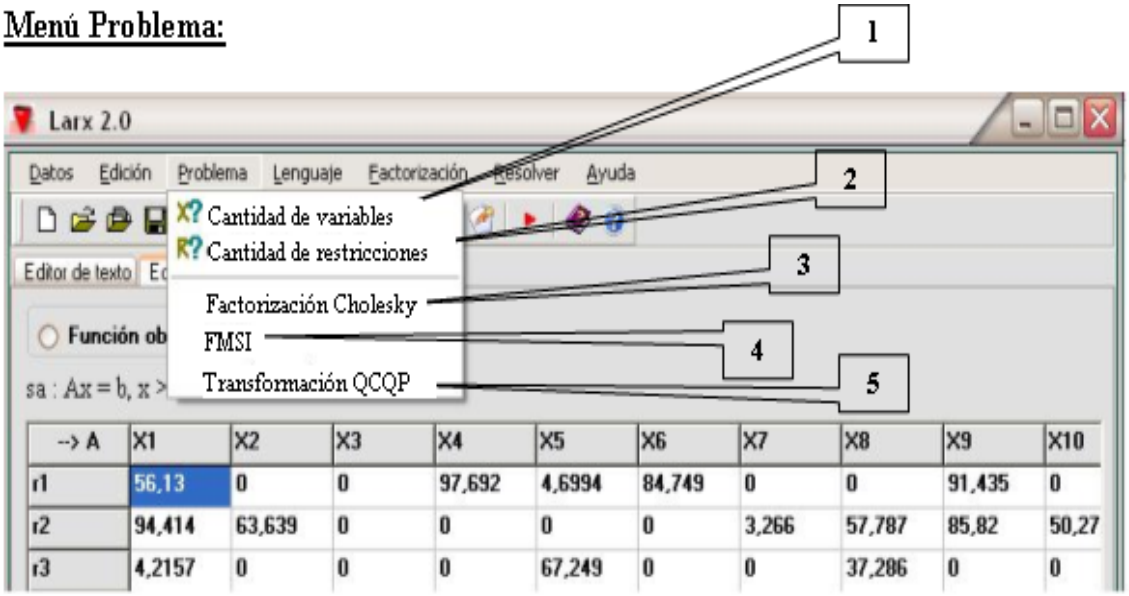
2- Cortar un fragmento del texto.

3- Copiar el texto seleccionado.

4- Pegar el texto copiado o cortado.

5- Seleccionar todo el texto

Menú Problema:



--> A	X1	X2	X3	X4	X5	X6	X7	X8	X9	X10
r1	56,13	0	0	97,692	4,6994	84,749	0	0	91,435	0
r2	94,414	63,639	0	0	0	0	3,266	57,787	85,82	50,27
r3	4,2157	0	0	0	67,249	0	0	37,286	0	0

Fig. 6: Menú Problema.

Este menú contiene las siguientes opciones:

1- Esta opción se usa cuando se está trabajando en el editor tabular, para fijar la cantidad de variables que va a tener el nuevo problema que se quiere introducir.

2- Esta opción se usa cuando se está trabajando en el editor tabular, para fijar la cantidad de restricciones que el nuevo problema va a tener.

3- Esta opción se usa cuando se quiere factorizar la matriz que está asociada a la función objetivo del problema que ha sido cargado, la factorización que se usa es la de Cholesky. El resultado se muestra en la Fig. 7.

4- Esta opción se usa cuando se quiere factorizar la matriz que está asociada a la función objetivo del problema que ha sido cargado, la factorización que se usa es la de matrices simétricas indefinidas (FMSI). El resultado se muestra en la Fig. 16.

5- Esta opción se usa cuando se quiere transformar el problema que ha sido cargado mediante la nueva transformación QCQP. El resultado se muestra en la Fig. 9.

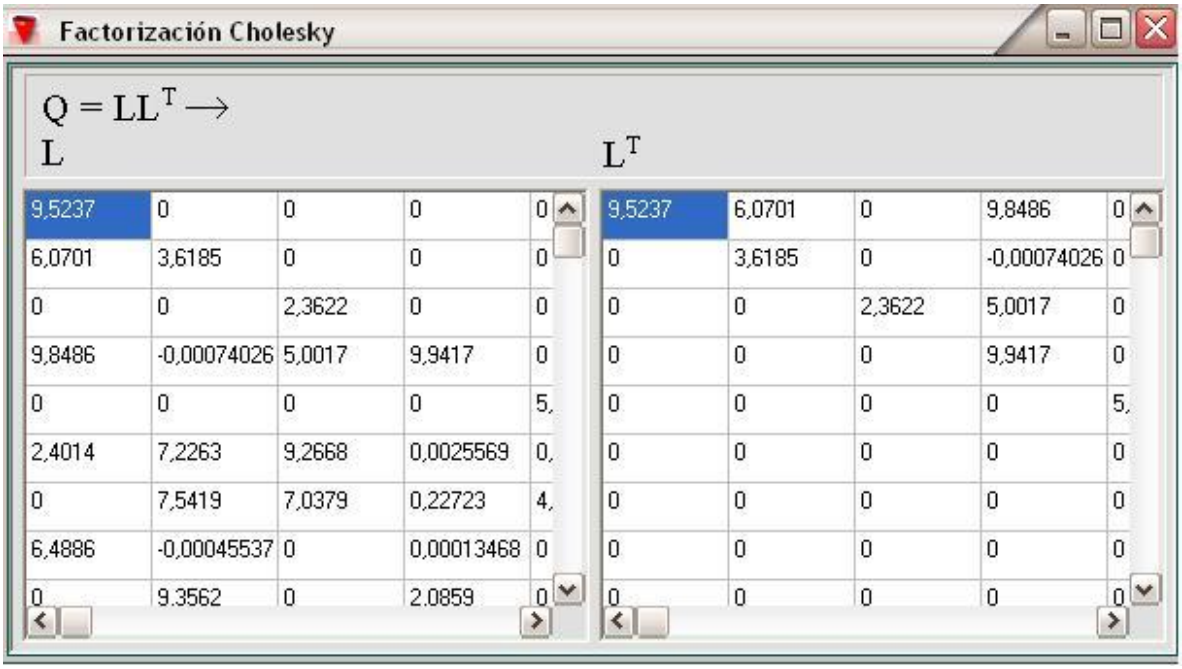
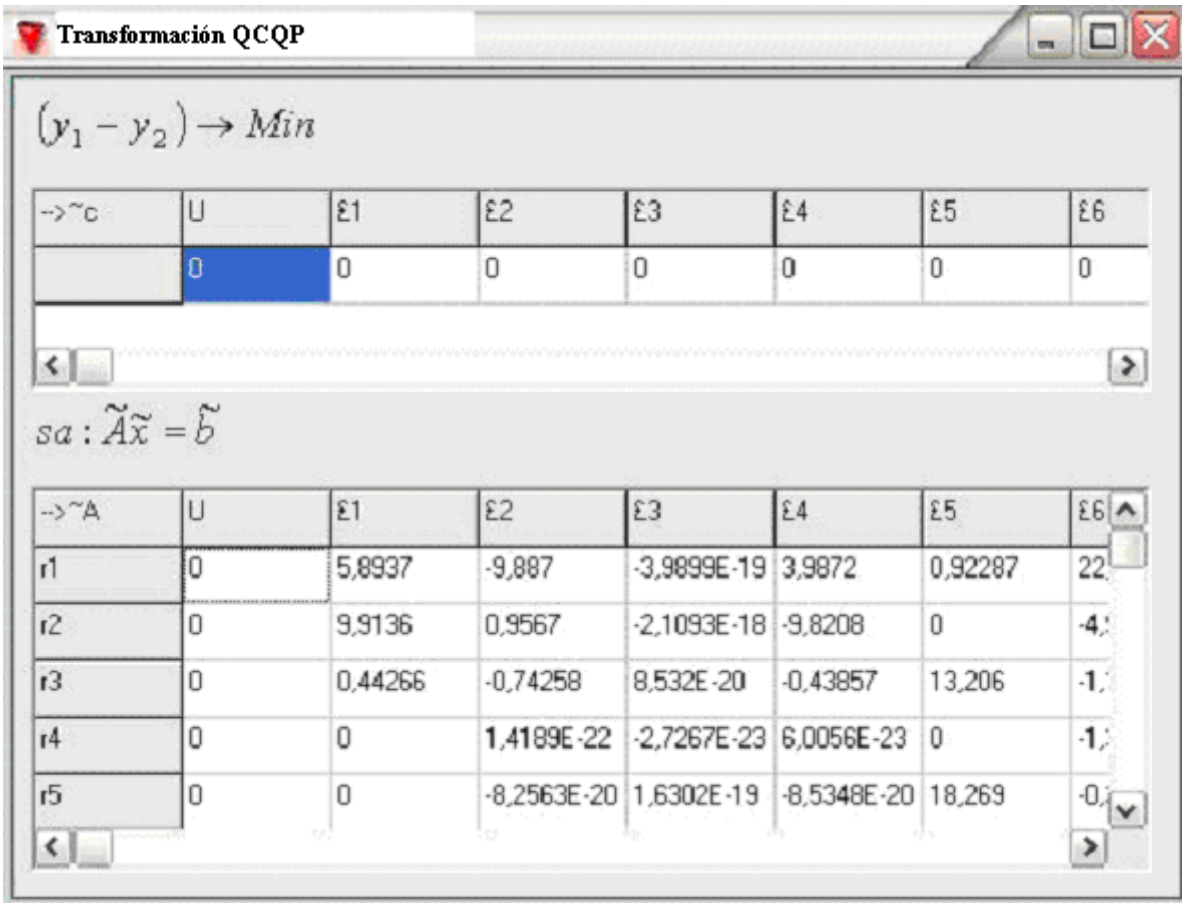


Fig. 7: Ventana factorización de Cholesky.

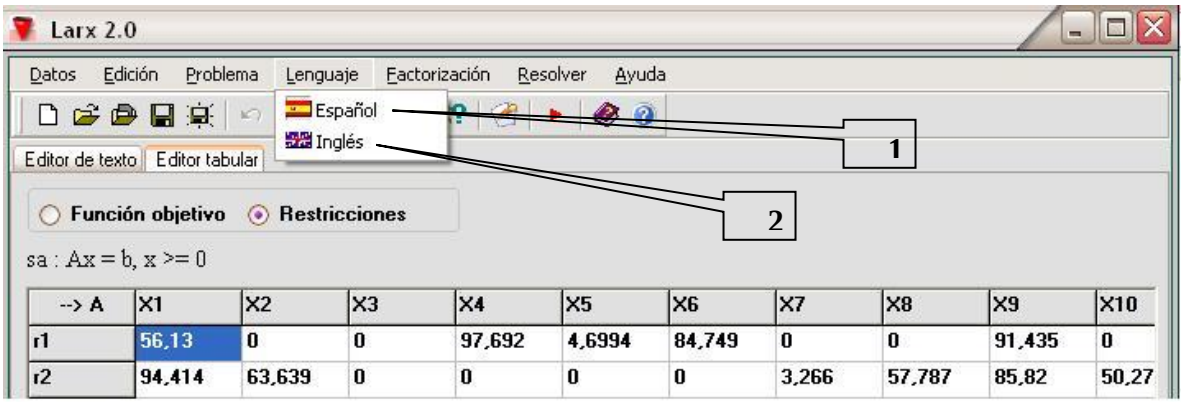


-->~c	U	£1	£2	£3	£4	£5	£6
	0	0	0	0	0	0	0

-->~A	U	£1	£2	£3	£4	£5	£6
r1	0	5,8937	-9,887	-3,9899E-19	3,9872	0,92287	22
r2	0	9,9136	0,9567	-2,1093E-18	-9,8208	0	-4,
r3	0	0,44266	-0,74258	8,532E-20	-0,43857	13,206	-1,
r4	0	0	1,4189E-22	-2,7267E-23	6,0056E-23	0	-1,
r5	0	0	-8,2563E-20	1,6302E-19	-8,5348E-20	18,269	-0,

Fig. 9: Ventana Transformación QCQP.

Menú Lenguaje:



--> A	X1	X2	X3	X4	X5	X6	X7	X8	X9	X10
r1	56,13	0	0	97,692	4,6994	84,749	0	0	91,435	0
r2	94,414	63,639	0	0	0	0	3,266	57,787	85,82	50,27

Fig. 10: Menú Lenguaje.

Este menú contiene las siguientes opciones:

1- Esta opción se usa si se quiere visualizar todo el texto del software en el idioma Español.

2- Esta opción se usa si se quiere visualizar todo el texto del software en el idioma Inglés.

Menú Factorización:

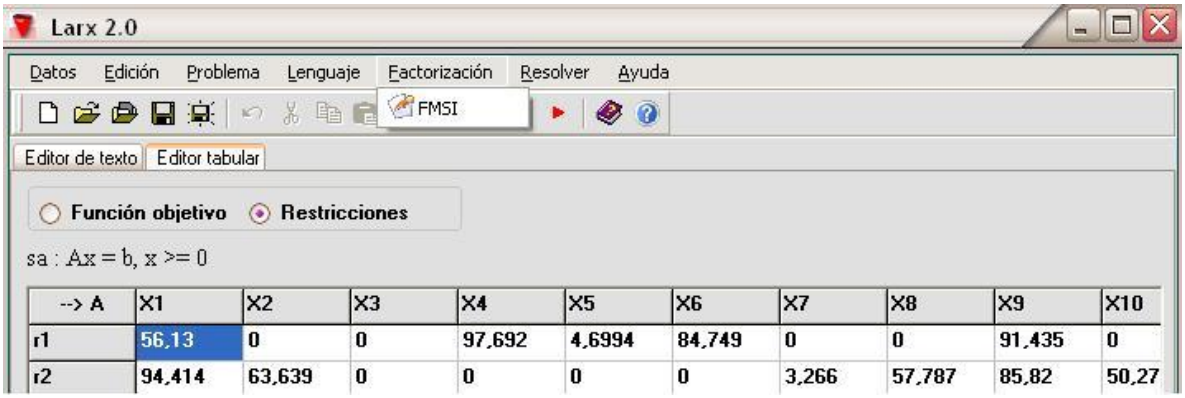


Fig. 11: Menú Factorización.

En este menú se brinda la posibilidad de factorizar una matriz de forma independiente, o sea, que no este asociada a un problema de Programación Cuadrática Convexa. Para esto se usa la factorización de matrices simétricas indefinidas. Al elegir esta opción se muestra en pantalla la Fig. 12.

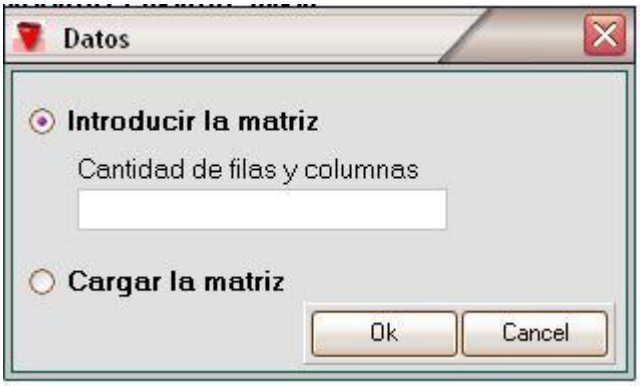


Fig. 12: Ventana Datos.

Si se marca introducir la matriz se tendrán que especificar sus dimensiones, o sea la cantidad de filas y columnas. Si elegimos esta opción al pulsar el botón Ok se muestra en pantalla la Fig. 13.

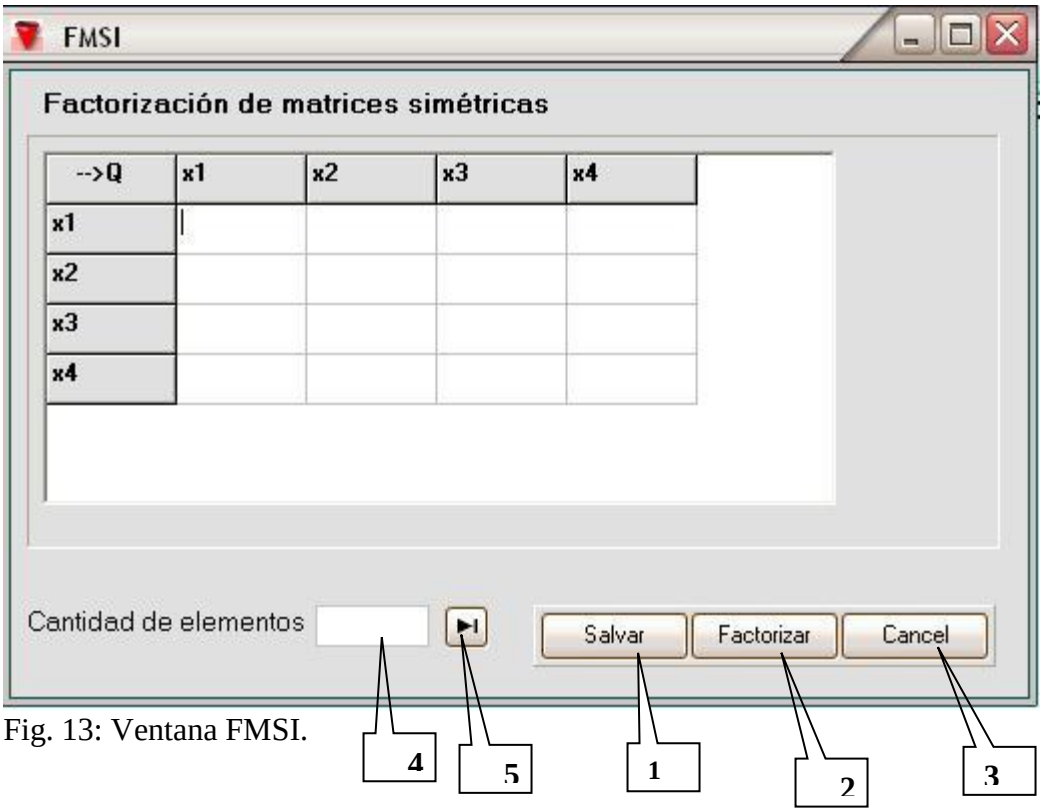


Fig. 13: Ventana FMSI.

En esta ventana usted podrá introducir los elementos de la matriz que desea factorizar donde:

1- Salva la matriz que se desea factorizar.

2- Al presionar este botón se obtiene la factorización de dicha matriz.

3- Si quiere cancelar la operación y cerrar la ventana.

4- Si usted se confundió en la ventana anterior en poner la dimensión de la matriz, le damos la posibilidad de corregir este error sin tener que volver a la otra ventana, para esto solo tendrá que poner la cantidad de filas o columnas que necesita y no se borran los datos introducidos.

5- Para redimensionar la matriz.

Si en vez de seleccionar Introducir la matriz selecciona Cargar la matriz la ventana se mostrará como aparece en la Fig. 14.

Fig. 14: Ventana Datos.

Donde:

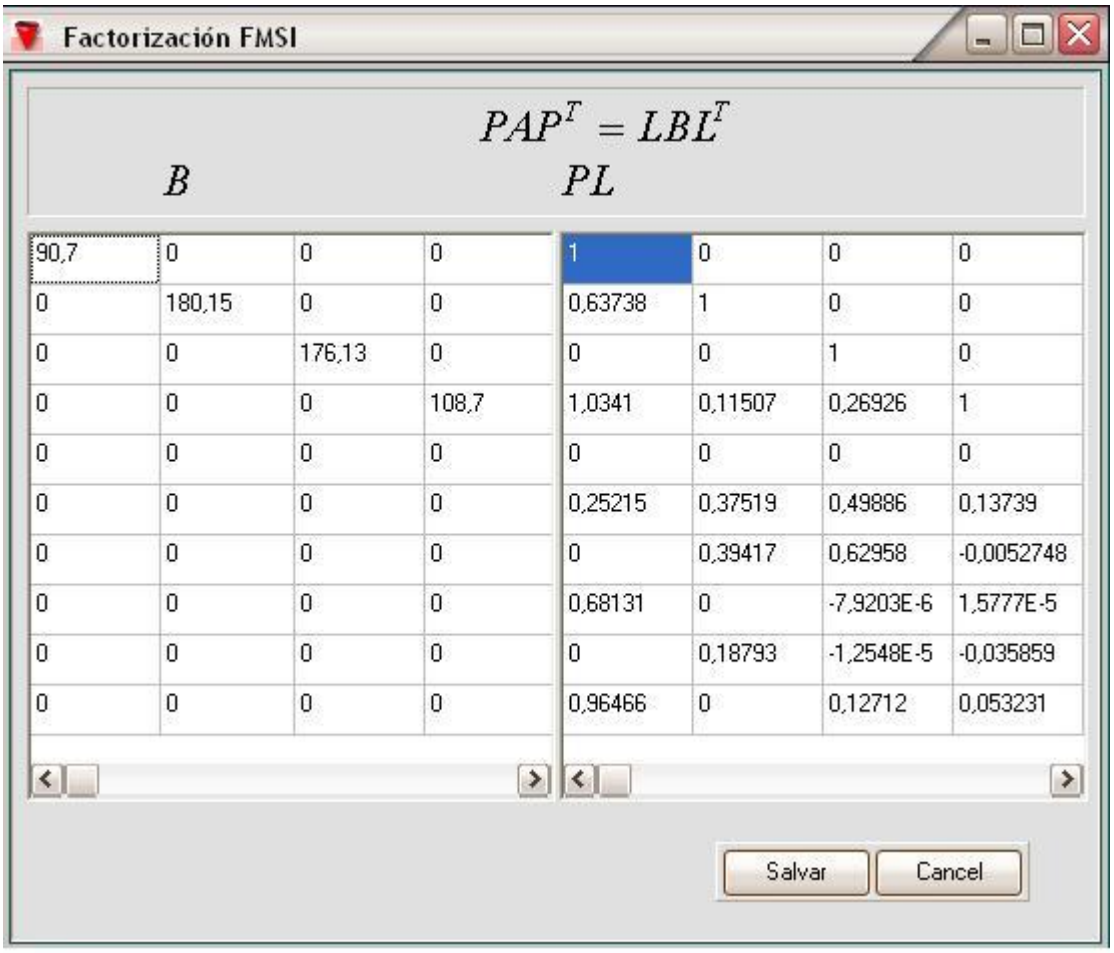
1- Se muestra un diálogo de Abrir para que seleccione el ejercicio que tiene asociada la matriz que usted desea factorizar, la cual se ilustra en la ventana de la Fig. 15.

2- Se cancela la operación de elegir la matriz a factorizar cerrándose la ventana.

Fig. 15: Ventana FMSI.

Si presiona el botón Salvar se salvará la matriz, si selecciona Factorizar obtendrá la factorización de la matriz cargada. Si presiona Cancel abortará la factorización.

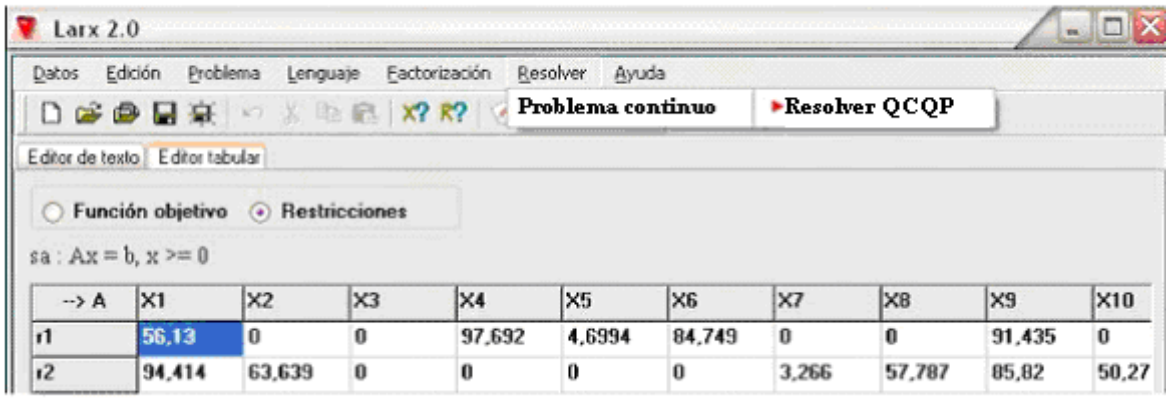
La factorización ya sea introduciendo la matriz como cargándola se mostrará en una ventana como a parece en la Fig. 16.



90,7	0	0	0	1	0	0	0
0	180,15	0	0	0,63738	1	0	0
0	0	176,13	0	0	0	1	0
0	0	0	108,7	1,0341	0,11507	0,26926	1
0	0	0	0	0	0	0	0
0	0	0	0	0,25215	0,37519	0,49886	0,13739
0	0	0	0	0	0,39417	0,62958	-0,0052748
0	0	0	0	0,68131	0	-7,9203E-6	1,5777E-5
0	0	0	0	0	0,18793	-1,2548E-5	-0,035859
0	0	0	0	0,96466	0	0,12712	0,053231

Fig. 16: Ventana Factorización FMSI.

Menú Resolver:



--> A	X1	X2	X3	X4	X5	X6	X7	X8	X9	X10
r1	56,13	0	0	97,692	4,6994	84,749	0	0	91,435	0
r2	94,414	63,639	0	0	0	0	3,266	57,787	85,82	50,27

Fig. 17: Menú Resolver.

En este menú se pueden resolver problemas continuos utilizando el algoritmo de Alizadeh, transformando dicho problema a un problema cónico.

La solución se mostrará en una ventana como la que se presenta en la Fig. 19.

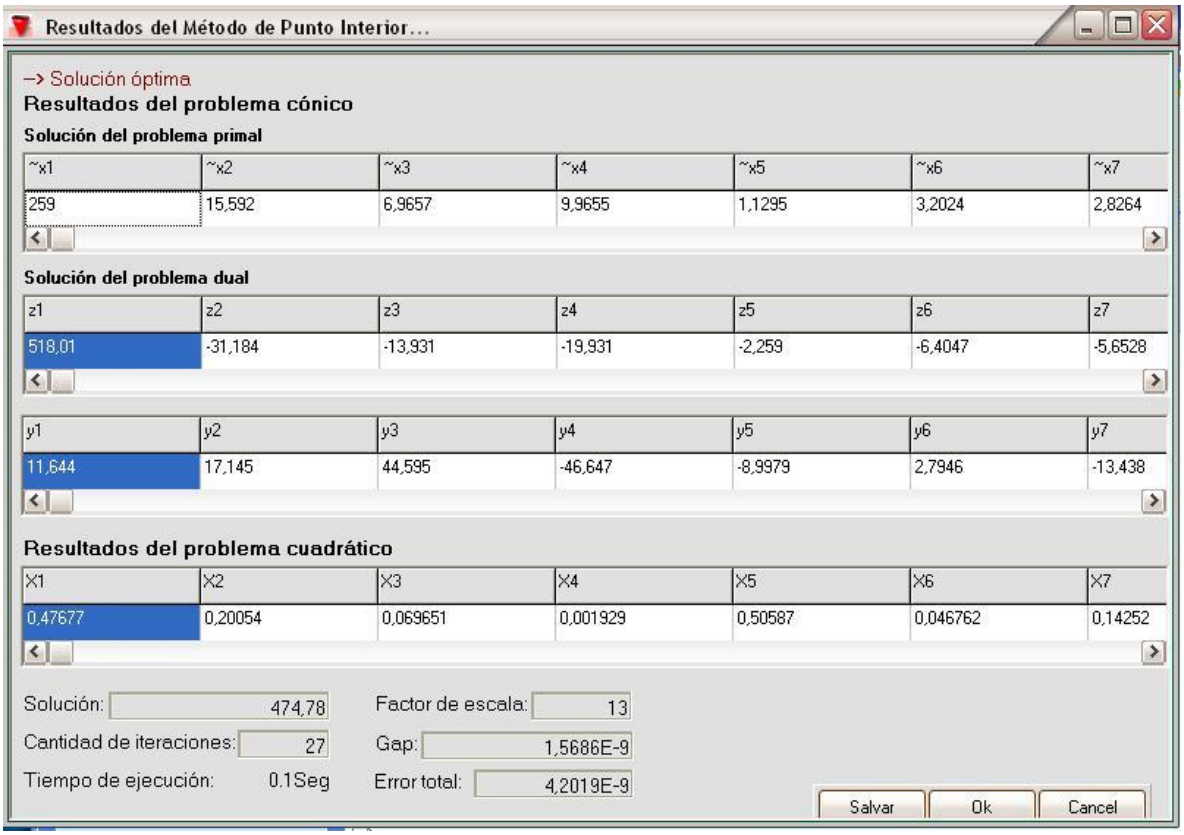


Fig. 19: Ventana Resultados del Método de Punto Interior.

Menú Ayuda:

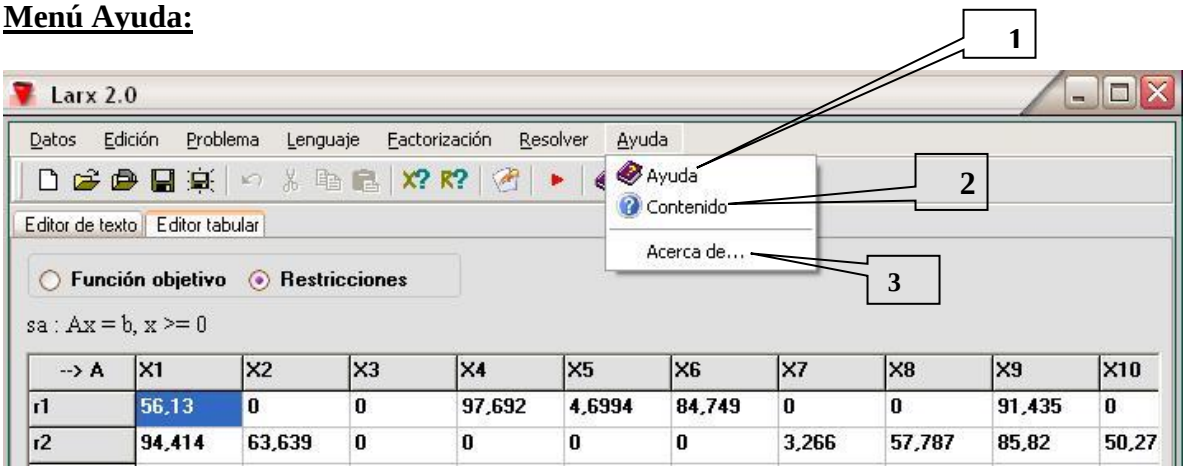


Fig. 24: Menú Ayuda.

Donde:

1- Brinda la ayuda sobre el software.

2- Brinda el contenido matemático para aquellos usuarios que no tengan un dominio completo de este.

3- Muestra algunos datos de interés como los autores del software y la versión.