UNIVERSIDAD CARLOS III DE MADRID
ESCUELA POLITÉCNICA SUPERIOR
DEPARTAMENTO DE
TEORÍA DE LA SEÑAL Y COMUNICACIONES

Ingeniería de Telecomunicación
Proyecto de Fin de Carrera

HERRAMIENTAS DE OPTIMIZACIÓN CONVEXA Y APLICACIONES EN TELECOMUNICACIONES

Autor: **Emilio Mejía Fernández de Velasco**
Director: **José Joaquín Escudero Garzás**
Tutora: **Ana García Armada**

Leganés, 2011

TÍTULO: *HERRAMIENTAS DE OPTIMIZACIÓN CONVEXA Y APLICACIONES EN TELECOMUNICACIONES.*

AUTOR: *EMILIO MEJÍA FERNÁNDEZ DE VELASCO*

DIRECTOR: *JOSÉ JOAQUÍN ESCUDERO GARZÁS*

TUTORA: *ANA GARCÍA ARMADA*

La defensa del presente Proyecto Fin de Carrera se realizó el día 6 de Septiembre de 2011; siendo calificada por el siguiente tribunal:

PRESIDENTE: CARLOS BOUSOÑO CALZÓN

SECRETARIO: VÍCTOR PEDRO GIL JIMÉNEZ

VOCAL: CARLOS GARCÍA RUBIO

habiendo obtenido la siguiente calificación:

CALIFICACIÓN: .....

Presidente **Secretario** **Vocal**

A mis padres.
Dedicado a mi madre, con cariño.
Dedicado en memoria a mi padre.
Gracias por estar orgullosos de mí
como yo lo estoy de vosotros.

Agradecimientos

Con este proyecto cierro un ciclo de mi vida que me planteé como un desafío personal fuera de la edad óptima, pero que me ha servido para satisfacer la necesidad de completar unas carencias de formación que llevaba tiempo echando en falta en mi etapa profesional como ingeniero técnico. He disfrutado aprendiendo de todas las áreas de conocimiento que he estudiado, aunque me ha costado un esfuerzo que sólo mi familia más cercana conoce verdaderamente. Por eso quiero agradecerles haber sido mi principal soporte y ánimo todos estos años, especialmente a mis padres: doy gracias a mi madre por su entrega incansable a su familia y su generosidad con todo el que la conoce, y sobre todo echo en falta no poder ver la expresión de satisfacción y felicidad que tendría mi padre si viviera este momento.

También agradezco la ayuda y el cariño de mis hermanos, José Luis, Alma y Paula, de mis cuñados Asun, Julián, y Raúl, y de todos mis sobrinos, Joselu, Carmen, Fran, Juli, Luis, Miki, Raulete y Fer (especialmente Joselu por haber compartido tantas horas de biblioteca).

Gracias a todos mis amigos, a mis compañeros desde los años de ingeniería técnica en Alcalá y de trabajo, por demostrarme su aprecio.

Por último, quiero agradecer el trabajo de revisión de este proyecto a Joaquín Escudero, el director del mismo, por ayudarme a mejorarlo.

Resumen

En las últimas décadas se han producido resultados fundamentales en teoría de optimización convexa y su utilización práctica. La comunidad de ingenieros no sólo se ha beneficiado de estos avances al encontrar aplicaciones prácticas, sino que también ha contribuido en agilizar el desarrollo matemático de la teoría y de algoritmos eficientes. Esto se ha traducido en la creación y comercialización de una gran cantidad de herramientas que permiten resolver problemas de grandes dimensiones con mucha precisión y rapidez. Frente a toda esta diversidad de opciones a la hora de escoger una u otra herramienta, este trabajo ofrece un recorrido por las técnicas más utilizadas y los programas que permiten la resolución de este tipo de problemas.

El motivo de estudiar en particular la optimización convexa es la altísima eficiencia que podemos obtener en su resolución, en comparación con problemas no convexos. Tradicionalmente se hacía una distinción entre problemas lineales y no lineales para delimitar la condición de fácil o difícil de resolver, pero en realidad esta frontera se encuentra en su naturaleza convexa o no convexa. Cuando un problema es convexo, se puede resolver de forma óptima bien con una solución cerrada por medio de las condiciones derivadas de la dualidad de Lagrange, o bien de forma numérica con algoritmos muy eficientes. Como consecuencia, una vez que hemos expresado un problema en forma convexa podemos decir que está resuelto.

El principal objetivo de este trabajo es la exposición de los recursos de optimización convexa más importantes que puede encontrar un ingeniero de telecomunicación para resolver diferentes problemas en su ámbito de trabajo. Estos recursos se materializan en los algoritmos existentes y los programas disponibles en el mercado que los utilizan. Para conocerlos en detalle comenzamos con los fundamentos teóricos básicos que se presentan en la primera parte de esta memoria, y a continuación se revisan los programas más relevantes que un usuario puede adquirir; por último se estudian dos casos de aplicación en los que se ponen en práctica todos principios aprendidos hasta aquí.

La primera parte de este trabajo contiene el marco teórico que nos permite transformar el planteamiento de diferentes problemas para desvelar su carácter convexo de forma que lo podamos resolver con los algoritmos y programas disponibles. Por desgracia, la mayoría de los problemas de ingeniería

no son directamente convexos tal como se formulan inicialmente, aunque muchos de ellos contienen una convexidad oculta que tendremos que descubrir para ser capaces de utilizar toda la maquinaria disponible de optimización convexa.

En la segunda parte se examinan los principales programas de optimización convexa. En ella se ponen de manifiesto las características principales de un conjunto de herramientas seleccionadas por su relevancia en el mercado. Analizaremos factores como su complejidad o facilidad de uso, los algoritmos y técnicas que implementan, o la forma en que se pueden adquirir. Dejaremos patente la diferencia entre herramientas de resolución o *solvers* y de modelado, y veremos cómo las primeras forman parte del núcleo de las segundas, siendo los *solvers* los encargados de proporcionar la ejecución de los algoritmos de minimización, mientras que los sistemas de modelado sirven como lenguaje con el que el usuario describe las funciones y restricciones del problema en un formato matemático más intuitivo que en los anteriores.

En la tercera parte veremos dos ejemplos de aplicación en telecomunicaciones en los que se utilizan herramientas de optimización convexa, y que nos sirven para consolidar las ideas y los datos de los capítulos anteriores. El primero es un conformador de haz robusto en el que se tienen en cuenta las variaciones del vector de apuntamiento para que estas no perjudiquen la calidad del receptor. En este ejemplo se consigue replicar los resultados de la bibliografía consultada y poner en relieve algunos aspectos de la naturaleza del problema que no había detectado antes de la implementación, por ejemplo la importancia del tamaño asignado a la región de incertidumbre y los valores que hacen inviable el problema. Los resultados muestran una comparativa en el comportamiento de distintas herramientas de optimización y en especial se comprueba la gran ventaja en rendimiento que ofrecen los programas de optimización convexa con respecto a los que están orientados a optimización global (no convexa).

La segunda aplicación consiste en optimizar el diseño del precodificador lineal en sistemas MIMO-OFDM en función de dos criterios: minimizar el error cuadrático medio de los flujos de datos transmitidos, o maximizar la información mutua del sistema. En ambos casos se aplica una restricción de control de potencia para reducir las variaciones de nivel de la señal OFDM. Se implementan dos modelos de optimización: un esquema no cooperativo en el que cada portadora se considera como un canal independiente, y un esquema cooperativo en el que el conjunto completo de portadoras constituye un solo canal que agrupa a todos y la optimización permite la distribución óptima de información entre distintas portadoras. En este segundo ejemplo únicamente he utilizado herramientas de optimización convexa debido a la diferencia de fiabilidad demostrada en el primer caso práctico.

Abstract

Several paramount results in convex optimization theory and its practical use have taken place over the last few decades. The engineering community not only has benefited from these recent advances by finding practical applications, but has also contributed to speeding up the mathematical development of both the theory and efficient algorithms. This has resulted in the creation and marketing of a great deal of tools that allow us to solve high dimensional problems very fast and accurately. Given all this diversity of options when it comes to choosing one tool or another, this work presents a survey of the most frequently used techniques and the programs that make possible the resolution of this kind of problems.

The reason of focusing on convex optimization in particular is the extremely high solving efficiency that can be achieved, compared to non convex problems. Traditionaly, a distinction between linear and non linear problems used to be made to mark the difference between easy and hard to solve problems, but the actual watershed is their convexity or non convexity. When a problem is found to be convex, it can be efficiently solved either with a closed solution by means of Lagrange duality and its derivative conditions, or numerically with very powerful algorithms. As a consequence, once a problem is stated in convex form we can consider it solved.

The main goal of this work is to present the most important resources in convex optimization that a telecommunication engineer may use to solve different problems within his/her professional field. These resources arise as existing algorithms and available programs in the market that use them. In order to get acquainted with them, we start with the basic theoretical foundations which are introduced in the first part of this report, and afterwards, the most relevant programs a user can get are examined; finally, two application cases are studied where all the principles that we have learned so far are put into practice.

The first part of this work comprises the theoretical framework that allows us to transform the approach to different problems in order to reveal their convexity and thus to be able to solve them with available algorithms and programs. Unfortunately, most engineering problems are not convex as they are initially stated; although many of them keep a hidden convexity that we will have to discover if we want to be able to use all the available

convex optimization machinery.

In the second part, the main convex optimization software packages are examined. The foremost features of a set of tools selected by their relevance in the market are highlighted therein. We will analyze factors like their complexity or simplicity in use, implemented algorithms and techniques, or the way they can be acquired. We will shed some light on the difference between solvers and modeling systems, and we will see how the former is part of the core of the latter, where solvers are in charge of providing the execution of the minimization algorithms, whereas modeling systems serve as a descriptive languaje for the user to formulate the problem functions and constraints in a more intuitive format than the one used by the solvers.

In the third part we will see two application cases in telecommunications where convex optimization tools are used. These will consolidate all the ideas and data from the previous chapters. The first one is a robust beamformer where variations in the steering vector are taken into account in order to keep them from damaging the receiver's quality. In this example the results of the bibliography I looked up are successfully replicated, and several features of the underlying nature of the problem which I had not detected before the implementation are highlighted, for instance, the importance of the size assigned to the uncertainty region, and the values that make the problem unfeasible. The results show a comparative in the behaviour of different optimization tools, and the big advantage in throughput offered by convex optimization packages with respect to those oriented to (nonconvex) global optimization is specially noticed.

The second application consists on the design optimization of a linear precoder in MIMO-OFDM systems as a function of two criteria: minimizing the mean square error of the transmitted data streams, or maximizing the system mutual information. Either way, a power control constraint is applied to reduce the OFDM signal level variations. Two optimization models are implemented: a non-cooperative scheme where each carrier is considered as an independent channel, and a cooperative strategy where the full set of carriers is considered as a single channel spanning all of them, and the optimization allows the optimal distribution of information among different carriers. In this second example I have only used convex optimization tools due to the difference in reliability shown in the first practical case.

Tabla de contenidos

Agradecimientos **iii**

Resumen **v**

Abstract **vii**

Tabla de contenidos **ix**

Lista de figuras **xvii**

Lista de tablas **xix**

Lista de algoritmos **xxi**

1 Introducción **1**
1.1 Motivación 1
1.2 Objetivos del proyecto 2
1.2.1 Identificación y evaluación de herramientas 3
1.2.2 Sistemas de modelado y de resolución 3
1.2.3 Casos de aplicación 3
1.2.4 Marco teórico 3
1.3 Método de trabajo 4
1.4 Medios utilizados 4
1.5 Casos prácticos desarrollados 5
1.5.1 Conformación de haz 6
1.5.2 Canales MIMO 6
1.6 Estructura de la memoria 7

I Fundamentos de optimización convexa. **9**

2 Conceptos teóricos **11**
2.1 Optimización convexa 11
2.1.0.1 Métodos de resolución 12
2.1.0.2 Optimización no lineal 12
2.1.1 Optimización convexa 13

2.2 Conjuntos convexos 14
2.2.1 Definiciones importantes 14
2.2.1.1 Conjuntos afines, convexos y conos 14
2.2.1.1.1 Cono asociado a una norma 15
2.2.1.1.2 Cono semidefinido positivo 15
2.2.1.2 Hiperplanos y semiespacios 16
2.2.1.2.1 Poliedros 16
2.2.1.3 Elipsoides 17
2.2.2 Operaciones que conservan la convexidad 17
2.2.2.1 Intersecciones 17
2.2.2.2 Funciones afines 17
2.2.3 Desigualdades generalizadas 18
2.2.3.1 Conos propios y desigualdades generalizadas 18
2.2.3.2 Elemento mínimo y minimal 18
2.2.4 Conos duales y desigualdades generalizadas 19
2.2.4.1 Elemento mínimo y minimal 19
2.3 Funciones convexas 19
2.3.1 Condiciones necesarias 19
2.3.1.1 Condición de primer orden 20
2.3.1.2 Condición de segundo orden 20
2.3.1.3 Ejemplos 20
2.3.1.3.1 Funciones de variable real, $f : \mathbb{R} \to \mathbb{R}$ 20
2.3.1.3.2 Funciones de variable vectorial, $f : \mathbb{R}^n \to \mathbb{R}$ 21
2.3.1.4 Condición del epígrafo 21
2.3.2 Operaciones que conservan la convexidad 22
2.3.2.1 Suma ponderada con pesos no negativos 22
2.3.2.2 Función compuesta con un mapa afín 22
2.3.2.3 Máximo y supremo puntual 22
2.3.2.4 Minimización de componentes 23
2.3.2.5 Perspectiva de una función 23
2.3.2.6 Composición 23
2.3.2.6.1 Composición escalar 23
2.3.2.6.2 Composición vectorial 24
2.3.3 Funciones cuasi-convexas 24
2.3.3.1 Definición 25
2.3.3.2 Condición de primer orden 25
2.3.3.3 Condición de segundo orden 25
2.3.3.4 Operaciones que conservan la cuasi-convexidad 26
2.3.4 Convexidad con desigualdades generalizadas 26
2.3.4.1 Funciones monótonas 26
2.3.4.2 Funciones convexas 26
2.4 Dualidad 27
2.4.1 Función dual de Lagrange 28
2.4.2 El problema dual de Lagrange 29

2.4.2.1 Dualidad débil 29
2.4.2.2 Condición de Slater 29
2.4.3 Condiciones de punto óptimo 30
2.4.3.1 Holgura complementaria 30
2.4.3.2 Condiciones KKT 31
2.4.4 Sensibilidad frente a perturbaciones 31
2.4.4.1 Funciones diferenciables 32

3 Planteamiento de problemas 33
3.1 Problemas equivalentes 33
3.1.1 Cambio de variables 34
3.1.2 Transformación de las funciones 34
3.1.3 Variables de holgura 34
3.1.4 Eliminación de restricciones de igualdad 35
3.1.4.1 Eliminación de las restricciones lineales de igualdad . 35
3.1.5 Introducción de restricciones de igualdad 36
3.1.6 Optimización sobre algunas variables 36
3.1.7 Problema en forma de epígrafo 36
3.1.8 Restricciones implícitas y explícitas 37
3.2 Optimización cuasi-convexa 37
3.2.1 Definición 38
3.2.2 Solución por medio de problemas de viabilidad 38
3.3 Clasificación de problemas convexos 40
3.3.1 Problemas de optimización lineal 40
3.3.1.1 LP en formato estándar y en formato desigualdad . . 40
3.3.1.2 Ejemplos 41
3.3.1.2.1 Función lineal por partes. 41
3.3.1.2.2 Programa de fracción lineal. 42
3.3.2 Problemas de optimización cuadráticos 42
3.3.3 Programación cónica de segundo orden 43
3.3.4 Programación con desigualdades generalizadas 43
3.3.4.1 Problemas en forma cónica 43
3.3.4.2 Programación semidefinida 44
3.3.5 Programación geométrica 44
3.3.5.1 Programación geométrica en forma convexa 45
3.4 Optimización vectorial 45
3.4.1 Óptimos de Pareto 46
3.4.2 Escalarización 46
3.4.2.1 Escalarización en problemas convexos 47
3.4.2.2 Optimización multicriterio 47

4 Algoritmos de minimización 49
4.1 Minimización sin restricciones 49
4.1.1 Métodos de descenso 50

4.1.1.1 Descenso de gradiente 51
4.1.1.2 Búsqueda lineal exacta 51
4.1.1.3 Búsqueda lineal con retroceso 52
4.1.2 Método de Newton 53
4.1.2.1 Independencia afín 54
4.1.2.2 Decremento de Newton 54
4.1.2.3 Algoritmo de Newton 55
4.2 Minimización con restricciones de igualdad 56
4.2.1 Minimización cuadrática con restricciones de igualdad . . 56
4.2.2 Método de Newton con restricciones de igualdad 57
4.2.3 Método de Newton desde un punto inicial no viable 58
4.2.3.1 Interpretación como método primal-dual 58
4.2.3.2 Reducción de la norma residual 59
4.2.3.3 Viabilidad con el paso completo 60
4.2.3.4 Algoritmo de Newton desde un punto no viable 60
4.2.3.5 Inicialización simplificada 61
4.3 Métodos de punto interior 61
4.3.1 Método de barrera 62
4.3.1.1 Camino central 63
4.3.1.2 Condiciones KKT 64
4.3.1.3 Algoritmo de barrera 64
4.3.2 Método de fase I 66
4.3.2.1 Método de Newton desde un punto inicial no viable . 66
4.3.3 Problemas con desigualdades generalizadas 67
4.3.3.1 Logaritmo generalizado 67
4.3.3.2 Funciones barrera logarítmica 68
4.3.3.3 Camino central 69
4.3.3.4 Puntos duales en el camino central 69
4.3.4 Métodos de punto interior primal-dual 70
4.3.4.1 Dirección de búsqueda primal-dual 70
4.3.4.2 Distancia alternativa de dualidad 71
4.3.4.3 Algoritmo de punto interior primal-dual 72
4.4 Algoritmo símplex 72
4.5 Problemas con variables enteras 73

II Herramientas de optimización convexa 77

5 Herramientas disponibles 79
5.1 Motivación 79
5.2 Herramientas de resolución 80
5.2.1 Efectividad de las herramientas de resolución 81
5.2.2 Problemas lineales 82
5.2.2.1 lp_solve 82

5.2.2.1.1 Licencias 83
5.2.2.2 GLPK 83
5.2.2.2.1 Licencias 83
5.2.2.3 BPMPD 84
5.2.2.3.1 Licencias 84
5.2.2.4 MINOS 85
5.2.2.4.1 Licencias 85
5.2.3 Problemas cuadráticos 85
5.2.3.1 CPLEX 85
5.2.3.1.1 Algoritmos 86
5.2.3.1.2 Utilización 87
5.2.3.1.3 Licencias 87
5.2.3.2 Gurobi 88
5.2.3.2.1 Licencias 88
5.2.3.3 Xpress-Optimizer 88
5.2.3.3.1 Licencias 89
5.2.3.4 OOQP 89
5.2.3.4.1 Licencias 89
5.2.4 Problemas cónicos 90
5.2.4.1 Mosek 90
5.2.4.1.1 Licencias 91
5.2.4.2 SeDuMi 91
5.2.4.2.1 Procesamiento previo 92
5.2.4.2.2 Matrices dispersas 92
5.2.4.2.3 Licencias 92
5.2.4.3 SDPT3 93
5.2.4.3.1 Licencias 93
5.2.4.4 PENSDP 93
5.2.4.4.1 Principales características 94
5.2.4.4.2 Utilización 94
5.2.4.4.3 Licencias 94
5.3 Sistemas de modelado 95
5.3.1 CVX 96
5.3.1.1 Programas de resolución 96
5.3.1.2 Licencias 96
5.3.2 AMPL 96
5.3.2.1 Características del lenguaje de modelado 96
5.3.2.2 Características del entorno de modelado 97
5.3.2.3 Herramientas de resolución utilizadas 97
5.3.2.4 Licencias 97
5.3.3 GAMS 98
5.3.3.1 Características del programa 98
5.3.3.2 Herramientas de resolución utilizadas 99
5.3.3.3 Licencias 99

5.3.4 YALIMP 99
5.3.4.1 Herramientas de resolución 100
5.3.4.2 Licencias 100
5.4 Programas de optimización global 103
5.4.1 Matlab Optimization Toolbox 103
5.4.1.1 Licencias 103
5.4.2 Mathematica 104
5.4.2.1 Licencias 105
5.4.3 MathOptimizer Professional 105
5.4.3.1 Licencias 106
5.5 Ejemplo 106
5.5.1 Resolución por *CVX* 107
5.5.2 Resolución por *Mosek* 109
5.5.3 Resolución por *SeDuMi* 110
5.5.4 Resolución por *Mathematica* 114
5.5.5 Resolución por *Matlab Optimization Toolbox* 115
5.5.6 Comparación entre los resultados obtenidos 116
5.6 Conclusiones 117

III Aplicaciones en telecomunicaciones 119

6 Conformación de Haz 121
6.1 Agrupaciones de antenas 122
6.2 Conformación de haz en recepción 123
6.2.1 Contexto 124
6.2.2 Extensiones de los diseños de caso peor 126
6.3 Implementación del optimizador 128
6.3.1 CVX 129
6.3.2 SeDuMi 130
6.3.3 Mosek 133
6.3.4 Matlab Optimization Toolbox 134
6.3.4.1 Descripción directa 134
6.3.4.2 Planteamiento estándar SOCP 138
6.3.5 Mathematica 140
6.3.5.1 Descripción directa 141
6.3.5.2 Planteamiento estándar SOCP 143
6.4 Simulaciones 144
6.4.1 Matriz de covarianza 144
6.4.2 Perturbación en el vector de apuntamiento 145
6.4.3 Selección de la región de incertidumbre 146
6.4.3.1 Gráfica de *CVX*, *SeDuMi* y *Mosek* 146
6.4.3.2 Gráficas de *Matlab Optimization Toolbox* 147
6.4.3.2.1 Descripción directa 147

6.4.3.2.2 Planteamiento estándar SOCP 148
6.4.3.3 Gráfica de *Mathematica* 149
6.4.4 SINR en función de SNR 150
6.4.4.1 Gráfica de *CVX*, *SeDuMi* y *Mosek* 151
6.4.4.2 Gráfica de *Matlab Optimization Toolbox* 151
6.4.4.2.1 Descripción directa 152
6.4.4.2.2 Planteamiento estándar SOCP 153
6.4.4.3 Gráfica de *Mathematica* 154
6.4.5 Tiempos de ejecución 155
6.4.5.1 Gráficas de *CVX*, *SeDuMi* y *Mosek* 155
6.4.5.2 Gráfica de *Matlab Optimization Toolbox* 157
6.4.5.3 Gráfica de *Mathematica* 158
6.4.6 Diagramas de radiación 159
6.4.6.1 Gráfica con 10 antenas 159
6.4.6.2 Gráfica con 100 antenas 160
6.5 Conclusiones 164

7 Canales MIMO **167**
7.1 Motivación 168
7.2 Modelo del canal 168
7.2.1 Ganancias en canales MIMO y sus propiedades 169
7.2.1.1 Subcanales y flujos 170
7.2.2 Compatibilidad entre ganancias 171
7.2.3 Técnicas de transmisión 171
7.2.3.1 Técnicas de transmisión MIMO sin CSIT 172
7.2.3.2 Técnicas de transmisión MIMO con CSIT exacto . . . 172
7.2.3.3 Técnicas de transmisión MIMO con CSIT parcial . . . 172
7.3 Procesamiento lineal 173
7.3.1 Parámetros de calidad del sistema 175
7.3.2 Receptor lineal 176
7.3.3 Diseño del transmisor 177
7.3.3.1 Canal MIMO simple 177
7.3.3.2 Múltiples canales MIMO 179
7.3.4 Criterios de optimización utilizados 180
7.3.4.1 Suma de MSE 180
7.3.4.2 Información mutua 182
7.3.5 Restricciones adicionales 184
7.4 Implementación del optimizador 185
7.4.1 CVX 185
7.4.1.1 Criterio de suma de MSE 185
7.4.1.2 Criterio de información mutua 187
7.4.2 Sedumi 187
7.4.3 Mosek 191
7.5 Arquitectura del sistema de simulación 192

7.6 Simulaciones 195
7.6.1 Comparativa de herramientas de optimización 195
7.6.2 Comparativa de criterios 199
7.6.3 Efecto de la restricción de PAR 201
7.7 Conclusiones 207

8 Conclusiones y líneas futuras **209**
8.1 Líneas futuras 212

A Resolución de ecuaciones lineales **213**
A.1 Factorización LU 213
A.2 Factorización de Cholesky 214
A.3 Complemento de Schur 214
A.4 El lema de inversión matricial 216

B Presupuesto del proyecto **219**
B.1 Fases del proyecto 219
B.2 Desglose presupuestario 220

Bibliografía **223**

Acrónimos **231**

Lista de figuras

3.1 Clasificación de problemas de optimización convexa. 41

4.1 Búsqueda lineal con retroceso 53
4.2 Estructura de árbol en un MIP 74

5.1 Ejemplo de problema QCQP. Curvas de nivel 107
5.2 Representación 3D del problema QCQP 108
5.3 Problema QCQP resuelto con *Mathematica* 115

6.1 Agrupación uniforme y sistema de coordenadas. 122
6.2 SINR vs. ε para *CVX*, *SeDuMi* y *Mosek* 146
6.3 SINR vs. ε para *Matlab Opt.* cod. directa 148
6.4 SINR vs. ε para *Matlab Opt* cod. SOCP 149
6.5 SINR vs. ε para *Mathematica* 150
6.6 SINR vs. SNR para *CVX*, *SeDuMi* y *Mosek*. 151
6.7 SINR vs. SNR para *Matlab Opt* directo 152
6.8 SINR vs. SNR para *Matlab Opt* SOCP 154
6.9 SINR vs. SNR para *Mathematica*. 155
6.10 T vs. N para *CVX*, *SeDuMi* y *Mosek* 156
6.11 T vs. N para *CVX*, *SeDuMi* y *Mosek* 157
6.12 T vs N para *Matlab Opt.* 158
6.13 T vs N para *Mathematica* 159
6.14 Diagrama de radiación de 10 antenas 160
6.15 Parámetros con 10 antenas 161
6.16 SINR vs. ε para N=100 antenas. 162
6.17 Diagrama de radiación para N=100 antenas. 162
6.18 SINR vs. SNR para N=100 antenas. 163

7.1 Modelo de múltiples canales MIMO 169
7.2 Esquema no cooperativo de canales MIMO 174
7.3 Esquema cooperativo entre canales MIMO. 175
7.4 Procesos realizados por cada función del código. 193
7.5 BER 2×2, $L = 1$. Herramientas 197
7.6 BER 4×2, $L = 1$. Herramientas 198
7.7 BER 4×4, $L = 2$. Herramientas 198
7.8 BER 4×4, $L = 4$. Herramientas 199

7.9 BER 2×2, $L = 1$. Criterios 200
7.10 BER 4×4, $L = 2$. Criterios 201
7.11 Efecto de μ 2×2, $L = 1$ sin recorte 203
7.12 P_{clip} 2×2, $L = 1$ sin recorte 203
7.13 BER 2×2, $L = 1$ sin recorte 204
7.14 Efecto de μ 2×2, $L = 1$ 204
7.15 P_{clip} 2×2, $L = 1$ con recorte 205
7.16 BER 2×2, $L = 1$ con recorte 205
7.17 Efecto de μ 4×4, $L = 2$ 206
7.18 BER Suma de MSE 4×4, $L = 2$ 206
7.19 BER Información mutua 4×4, $L = 2$ 207

Lista de tablas

5.1 Resumen de las herramientas de resolución lineales. 82
5.2 Resumen de las herramientas de resolución cuadráticas. . . 86
5.3 Resumen de las herramientas de resolución de problemas cónicos. 90
5.4 Resumen de las herramientas de modelado. 102
5.5 Resumen de las herramientas de resolución no convexa. . . . 104

7.1 Opciones de simulación 194
7.2 Perfil de potencia y retardo 196

B.1 Fases del Proyecto 220
B.2 Costes de personal 221
B.3 Costes de equipos 221
B.4 Otros costes directos del proyecto 222
B.5 Coste total del proyecto 222

Lista de Algoritmos

3.1 Método de bisección para optimización cuasi-convexa 39

4.1 Método general de descenso 50
4.2 Búsqueda lineal con retroceso 52
4.3 Método de Newton 55
4.4 Método de Newton con restricciones de igualdad 58
4.5 Método de Newton desde un punto inicial no viable 61
4.6 Método de barrera 65
4.7 Método de punto interior primal-dual 72

A.1 Ecuaciones lineales con factorización LU 214
A.2 Ecuaciones lineales por Cholesky 214
A.3 Ecuaciones lineales por eliminación en bloque 215

Capítulo 1

Introducción

1.1 Motivación

La optimización matemática es el estudio sobre cómo hacer la mejor elección cuando estamos limitados por un conjunto de requerimientos [Dat06]. Cuando hablamos en concreto de optimización convexa nos referimos a minimizar funciones convexas reales definidas para una variable contenida dentro de un subconjunto convexo de un espacio vectorial.

Las técnicas abarcadas por este campo de estudio son importantes en aplicaciones de ingeniería porque al plantear un problema de diseño en forma convexa se puede identificar la estructura de la solución óptima, que suele revelar aspectos importantes de dicho diseño. Esto se debe, entre otras cosas al hecho de que cualquier solución local es también una solución global, y a que existe una teoría de dualidad y unas condiciones de punto óptimo que permiten verificar la solución encontrada. Además, existen algoritmos numéricos muy potentes que resuelven este tipo de problemas de forma muy eficiente.

En los últimos años se han producido avances significativos en la utilización de técnicas de optimización convexa en telecomunicaciones y procesamiento de señales, con un impacto importante en problemas que antes se consideraban intratables. Así, existen muchas líneas de investigación abiertas, y el rango de aplicaciones en que se puede utilizar llega desde sistemas MIMO[1], localización de sensores, optimización de potencia en redes de tipo malla, tratamiento de imágenes, diseño de filtros, y muchas más. Por ejemplo, en [Chi09] se propone un método de localización de terminales móviles por medio de una aproximación convexa de un problema que originalmente no lo es; también es fácil encontrar una gran cantidad aplicaciones relacionadas con redes de sensores, como en [Mo09], donde se considera la selección de la medida de un conjunto de estos con el objetivo de maximizar el tiempo de vida de la red manteniendo las restricciones de calidad y precisión esta-

[1] *Multiple Input Multiple Output*

blecidas; la optimización de potencia en redes con distribución en malla es otra línea ampliamente extendida en la que se utiliza optimización convexa de forma generalizada, por ejemplo en [Pha09] se optimiza la transmisión en una red con nodos de retransmisión. Esta aplicación está relacionada con la conformación de haz de nuestro tema 6. En definitiva, el rango de aplicaciones de este tipo de soluciones está en continua expansión y es muy variado.

Aunque los fundamentos matemáticos en los que se basan estas técnicas no son novedosos, su aplicación práctica no ha despegado hasta hace algunos años debido a los adelantos en capacidad de procesamiento, porque las limitaciones de uso que se planteaban en los artículos publicados en las pasadas décadas han dejado de ser un obstáculo con las capacidades computacionales actuales.

Todo lo anterior ha originado la aparición de un número considerable de métodos y herramientas para optimización convexa, como por ejemplo *CVX*, *Mosek*, etc., de forma que a la hora de hacer uso de las mismas para resolver un problema concreto, no resulta fácil determinar cuál de ellas es la adecuada. Por este motivo, es conveniente detenerse a recopilar las técnicas existentes y sus posibles aplicaciones. Para ello, he intentado recoger en este trabajo, por un lado un resumen de la base teórica, a continuación una revisión del estado del arte en herramientas disponibles en el mercado que facilitan su utilización, y por último he seleccionado dos aplicaciones que muestren los beneficios de los recursos presentados y que sean representativas de los contenidos de nuestra carrera.

1.2 Objetivos del proyecto

La finalidad de este trabajo es realizar un estudio comparativo de las herramientas *software* de optimización convexa más relevantes disponibles en el mercado. Esto se concreta en los siguientes objetivos, que iremos cubriendo a lo largo del presente trabajo:

- Identificación y evaluación de herramientas de optimización convexa disponibles en el mercado. Comprobar su complejidad o facilidad de uso, y sus características principales.
- Destacar las diferencias entre herramientas de resolución (*solvers*) y sistemas de modelado.
- Definir casos de aplicación relevantes en telecomunicaciones. Para ello he seleccionado una aplicación de conformación de haz y otra de canales MIMO.
- Proporcionar un marco teórico para poder evaluar de forma objetiva las herramientas y estudiar su aplicación.

1.2.1 Identificación y evaluación de herramientas

El primer objetivo se materializa en el capítulo 5, donde se distinguen claramente los aspectos más importantes de los programas con más calado en el mercado. Además de conocer las bondades de cada paquete, es interesante saber de qué forma se puede adquirir, si su licencia es de libre distribución o si es de pago; pero sobre todo es necesario saber a qué tipo de problemas están enfocados, lo que nos permitirá no equivocarnos a la hora de comprar un producto u otro en función de nuestras necesidades.

1.2.2 Sistemas de modelado y de resolución

El segundo objetivo es consecuencia del primero. Al hacer la revisión de herramientas en el mercado es inmediato encontrarse con que unas se identifican como sistemas de modelado y otras como *solvers* o sistemas de resolución. En este trabajo, y en concreto en el tema 5 se especifica claramente qué significan estos dos términos de clasificación, y se muestran las diferencias entre ambos, no sólo por su finalidad sino también por la forma de trabajar con cada uno de ellos. Posteriormente, y en especial en el tema 6, las simulaciones están enfocadas a comparar varias de estas herramientas, entre las que hay casos de modelado y de resolución, y donde termina de aclararse cuáles son las diferencias entre unos sistemas y otros.

1.2.3 Casos de aplicación

Se han seleccionado dos ejemplos de aplicación relevantes en nuestra ingeniería, que sirven para demostrar de qué forma nos podemos beneficiar si utilizamos estas herramientas en nuestros problemas. En ambos casos se hace una comparativa entre varios programas y se muestran los procesos necesarios para adaptar nuestras especificaciones al formato de cada uno. Se han utilizado también algunos paquetes de optimización generalista que no son específicamente convexos, para que podamos encontrar el sentido de enfocar nuestro estudio a la optimización convexa en particular.

1.2.4 Marco teórico

Este objetivo está resuelto en los capítulos que forman la primera parte de este documento. He intentado reducir este bloque a los conceptos imprescindibles que se necesitan para asimilar de forma eficiente la información que nos podemos encontrar, por un lado en la descripción comercial de cualquier herramienta de optimización convexa, y por otro lado en cualquier artículo publicado sobre líneas de investigación sobre aplicaciones en telecomunicaciones o en cualquier otro campo.

1.3 Método de trabajo

Para demostrar los beneficios que puede aportar la optimización convexa a las telecomunicaciones, en el presente trabajo se ha decidido seguir la siguiente metodología: conocer las herramientas de optimización convexa disponibles, y después presentar ejemplos de su utilización a fin de evaluar el comportamiento de las mencionadas herramientas de optimización cuando se aplican en telecomunicaciones.

Para entender los términos que nos encontramos a la hora de buscar información sobre herramientas disponibles, así como en la utilización en aplicaciones reales, es necesario contar con una base teórica sólida, por lo que he añadido una recopilación imprescindible de conceptos que hay que conocer antes de abordar un problema de optimización convexa.

Esto ha supuesto una carga importante de trabajo, porque no partía con conocimientos suficientes para comprender el significado de muchos temas tratados en la bibliografía que encontraba. Por este motivo la parte teórica consta de varios capítulos en lugar de figurar como un breve apéndice al final de la memoria. No obstante, he eliminado una gran cantidad de información que no considero especialmente relevante a pesar de haber invertido un gran esfuerzo en asimilar.

Para conocer los recursos disponibles se presenta en el tema 5 una revisión de las herramientas de *software* más importantes del mercado, y también he incluido un ejemplo sencillo que permite visualizar la geometría de un problema de optimización de forma didáctica e intuitiva por tratarse de un caso en tres dimensiones. En las aplicaciones prácticas se trabaja con un número elevado de dimensiones y el esquema de funcionamiento puede quedar enmascarado en un nivel de abstracción elevado, por lo que es conveniente partir de un caso sencillo y desde ahí avanzar hacia otros ejemplos más complicados.

En esta revisión de programas disponibles, comienzo por distinguir entre herramientas de modelado y de resolución. Dentro de cada tipo he seleccionado un conjunto significativo de utilidades en función del tipo de problemas que resuelven.

Los casos prácticos presentados tienen cierta relación entre ellos y también reflejan aspectos importantes que hemos visto en varias asignaturas de nuestra carrera, en especial algunas impartidas por este departamento.

1.4 Medios utilizados

Para desarrollar las simulaciones he trabajado principalmente sobre *Matlab* en su versión *R2010a*. Para poder ejecutarlas es necesario tener instalados los siguientes complementos de optimización:

- *Matlab Optimization Toolbox*. Con licencia comercial de *Mathworks*.

- *CVX*. Con licencia libre *GNU*[2]*-GPL*[3].
- *SeDuMi*[4]. Con licencia libre *GNU-GPL*.
- *Mosek*. Con licencia comercial. He utilizado una versión de demostración para estudiantes.

Para las simulaciones del capítulo 7 es necesario tener el complemento *Matlab Communications Toolbox*, que por un lado nos permite utilizar la estructura de tipo `mimochan` para canales MIMO, y por otro lado incluye la consola de pruebas de tasa de error con la que he obtenido las curvas de BER[5].

También he utilizado el programa *Mathematica* de *Wolfram* en su versión 8, y para poder llamar a las funciones de *Mathematica* desde *Matlab* se necesita instalar el complemento *Mathematica Symbolic Toolbox for Matlab*.

La escritura de esta memoria está hecha en LaTeX, con *Miktex* y el editor *WinEdt* bajo entorno *Windows 7*.

El ordenador utilizado es un portátil *Dell Studio 1745* con procesador *Intel Dual-Core* a 2.1GHz.

1.5 Casos prácticos desarrollados

La optimización convexa nos permite resolver con gran eficiencia grandes problemas. La mayor complicación está en plantear las especificaciones de forma adecuada, pero una vez resuelta la definición, la ejecución y el escalado se convierten en un paso sencillo.

En este trabajo se estudian dos casos prácticos de líneas de investigación en telecomunicaciones, y las soluciones que he encontrado en la bibliografía publicada [Ger03, Vor03, Lor05, Pal03a, Pal05] pasan por la utilización de algoritmos de búsqueda adaptados a cada caso. En este proyecto se propone una resolución de ambos problemas utilizando distintos programas disponibles en el mercado. Las soluciones propuestas consisten en la adaptación de ambos casos prácticos a la sintaxis de cada herramienta, con el procesamiento previo y las modificaciones de planteamiento que esto implica.

Habitualmente es aconsejable utilizar recursos existentes en el mercado que nos ahorren un trabajo innecesario. Por eso veremos en este proyecto que las herramientas de modelado nos permiten una primera aproximación al problema en un lenguaje más próximo a nuestras especificaciones, y cuando hemos comprobado que tenemos una solución correcta podemos adaptarlo a herramientas de resolución más adecuadas para la realización práctica.

[2] *GNU's Not Unix*
[3] *GNU Public License / General Public License*
[4] *Self Dual Minimization*
[5] *Bit Error Rate*

1.5.1 Conformación de haz

El primer modelo es un método de conformación de haz en receptores con agrupaciones de antenas. El ejemplo consiste en simular un optimizador robusto capaz de ofrecer prestaciones aceptables frente a perturbaciones en el modelo teórico estimado de vector de apuntamiento. La finalidad de este ejercicio es demostrar el procedimiento de trabajo con varias herramientas de optimización, dejando patente la diferencia entre sistemas de resolución y sistemas de modelado en términos de complejidad en el procesamiento previo del problema y de velocidad de ejecución. Se espera que todos los programas utilizados den resultados similares con respecto a la aplicación, y para comprobarlo, he añadido dos procedimientos basados en inversión de matrices muestrales de covarianza que sirven de referencia para saber si nuestra simulación está funcionando correctamente y mejora la calidad de los métodos no convexos.

En este ejemplo he buscado replicar los resultados de las publicaciones que he considerado más representativas de esta aplicación, en especial [Vor03], aportando la utilización de un conjunto de herramientas de optimización, de forma que se pueda mostrar una comparativa de procedimientos y características entre ellas.

Esta comparativa se muestra en las simulaciones, donde se ponen en relieve las virtudes y desventajas que presenta cada una en diferentes aspectos. Gracias a haber utilizado programas de optimización, el paso de un diseño simple a un caso robusto se limita a ser cuidadosos en la forma de especificar el problema porque a partir de aquí, la implementación es inmediata.

1.5.2 Canales MIMO

En el ejemplo de canales MIMO también se plantea como finalidad conseguir resultados similares a la bibliografía que me ha servido de guía [Pal03a], pero en este caso aprovecho las conclusiones de la aplicación anterior y el objetivo principal se orienta a presentar las herramientas de optimización convexa como una alternativa que permite añadir restricciones que de otro modo sería excesivamente complicado.

En este segundo ejercicio se optimiza el diseño del precodificador lineal del transmisor en base a dos criterios: maximizar la información mutua, y minimizar a suma del error cuadrático medio en los flujos de datos transmitidos. Una vez más, el propósito de este ejemplo es la demostración de que las herramientas utilizadas aportan una gran flexibilidad en el diseño a la hora de añadir especificaciones y restricciones. Con este motivo, se ha añadido una técnica que permite mitigar el efecto de los picos de potencia en la señal OFDM[6] limitando la probabilidad de recorte.

[6] *Orthogonal Frequency Division Multiplexing*

Si no tuviésemos en cuenta las restricciones añadidas, se podrían utilizar algoritmos personalizados para resolver cada caso. Las simulaciones presentadas abren la puerta a la utilización de estos programas para estudiar otros criterios sin necesidad de preocuparnos por implementar algoritmos iterativos de búsqueda para encontrar soluciones óptimas.

1.6 Estructura de la memoria

La primera parte contiene la teoría básica de optimización convexa y comprende los capítulos 2, 3 y 4. El tema 2 trata la definición de conjuntos y funciones convexas, junto con la teoría de dualidad de Lagrange. El tema 3 explica algunas técnicas de procesamiento y conversión de problemas. Esto constituye la mayor carga de trabajo a la hora de tratar este tipo de ejercicios. Este capítulo también contiene la clasificación y las formas estandarizadas de los distintos tipos de problemas convexos, esquematizada en la figura 3.1. Para finalizar la parte teórica, en el capítulo 4 se repasa la forma de llegar a los algoritmos básicos que permiten resolver los problemas convexos.

La segunda parte contiene un solo capítulo en el que se revisan las características de las herramientas de optimización convexa disponibles en el mercado. En este tema 5 se incluye un ejemplo práctico representativo de la forma de implementar problemas con varias herramientas.

Las aplicaciones implementadas para las telecomunicaciones son una simulación de conformación de haz en receptores con agrupaciones de antenas (capítulo 6), y un diseño de procesadores lineales en transmisión y recepción para sistemas MIMO-OFDM (capítulo 7). Ambas aplicaciones están relacionadas entre sí. De hecho, cuando preparaba el tema 6 pensaba añadir también una aplicación de conformación de haz en transmisores, pero los planteamientos se aproximan en varios aspectos al diseño de sistemas MIMO, por lo que este capítulo refleja de forma más general las ideas que intentaba mostrar.

Por último, el tema 8 contiene las conclusiones del trabajo. No obstante, al final de los capítulos 6 y 7 también se extraen las conclusiones de cada ejemplo.

Parte I

Fundamentos de optimización convexa.

Capítulo 2

Conceptos teóricos

Este capítulo contiene los fundamentos básicos de teoría de optimización convexa que se necesitan para comprender el funcionamiento y para qué sirven las herramientas que se tratan en este proyecto, y también las ideas necesarias para desarrollar las simulaciones de los dos casos prácticos de los temas 6 y 7.

En primer lugar se revisan algunos conceptos sobre conjuntos y funciones convexas que nos encontramos continuamente al leer bibliografía sobre este tipo de aplicaciones. Todo esto nos servirá para fijar las bases de nomenclatura y la forma en que se plantean los problemas que vamos a ver, pero también nos servirá para poder disponer de recursos suficientes para razonar las transformaciones y combinaciones que nos permiten formular una gran variedad de problemas de forma convexa.

En §2.4 podemos estudiar la teoría de dualidad de Lagrange, que constituye el fundamento teórico más importante en el que se basa la optimización convexa. En este apartado se describen las condiciones KKT[1] que forman el núcleo de los sistemas de resolución de estos problemas.

La principal referencia que he utilizado para esta primera parte es el libro de Stephen Boyd [Boy04] y las clases de este profesor en la universidad de Stanford, que se pueden descargar en vídeo [Boy08].

2.1 Optimización convexa

Cuando hablamos de un problema de optimización en general, es decir, sin especificar que sea convexo, queremos minimizar una función $f_0(\boldsymbol{x}) : \mathbb{R}^n \to \mathbb{R}$ que será la *función objetivo*, con una serie de restricciones indicadas por las

[1] *Karush-Kuhn-Tucker*

funciones $f_i(\boldsymbol{x}) : \mathbb{R}^n \to \mathbb{R}$. Esto lo expresamos de la siguiente forma:

$$\begin{array}{ll} \underset{\boldsymbol{x}}{\text{minimizar}} & f_0(\boldsymbol{x}) \\ \text{sujeto a} & f_i(\boldsymbol{x}) \leq b_i, \quad i = 1, \ldots, m \\ & h_i(\boldsymbol{x}) = g_i, \quad i = 1, \ldots, p \end{array} \tag{2.1}$$

Por tanto, buscamos un vector óptimo $\boldsymbol{x}^*$ que cumpla las restricciones $f_1(\boldsymbol{x}^*) \leq b_1, \ldots, f_m(\boldsymbol{x}^*) \leq b_m$, $h_1(\boldsymbol{x}^*) \leq g_1, \ldots, h_p(\boldsymbol{x}^*) \leq g_p$ y cuya función objetivo $f_0(\boldsymbol{x}^*)$ sea el mínimo de los valores obtenidos con cualquier otro vector que cumpla las restricciones.

Podemos encontrar diferentes tipos de problemas de optimización, y en este trabajo nos fijamos en el caso específico de problemas de optimización convexa, en los que el objetivo y las restricciones son funciones convexas. Más adelante veremos en qué consiste esto.

La función objetivo $f_0(\boldsymbol{x})$ representa el coste de elegir un determinado valor para $\boldsymbol{x}$, y las funciones $f_i(\boldsymbol{x})$ y $h_i(\boldsymbol{x})$ son las especificaciones que limitan el conjunto de valores de $\boldsymbol{x}$ que podemos elegir. La solución nos dará la elección de $\boldsymbol{x}$ que supone un coste mínimo dentro de los requerimientos del problema.

2.1.0.1 Métodos de resolución de problemas de optimización

Para resolver un problema de optimización es necesario utilizar el método adecuado al tipo de problema concreto que queremos solucionar. La complejidad de la solución depende de las características de la función objetivo y de las funciones de restricción, así como de la cantidad de variables y restricciones del problema.

El motivo de centrarnos en optimización convexa es que cuando se trata de este caso, existen métodos muy eficientes para encontrar una solución. Sin embargo, en el caso general esto no sucede, y las soluciones disponibles dejan de ser eficientes cuando las dimensiones de las variables son elevadas. Sólo en algunos casos podemos encontrar métodos eficaces.

En el capítulo 5 veremos una comparativa de diferentes herramientas para resolver problemas de optimización. Algunas de ellas no necesitan que el problema sea convexo. Lo que caracteriza a los métodos de resolución convexa es que pueden abordar problemas de grandes dimensiones, especialmente cuando las relaciones entre las variables son poco densas o dispersas.

2.1.0.2 Optimización no lineal

Se suele llamar optimización no lineal no sólo a los problemas que no son lineales, sino en general a los problemas que no son convexos. De hecho, los problemas lineales son un caso particular de problemas convexos.

Por ejemplo, en la documentación sobre optimización del programa *Mathematica* [Wol10] o del complemento para *Matlab* sobre optimización, *Optimization Toolbox* [Mat10], se utiliza este término.

En estos casos, cuando los problemas exceden pocas centenas de variables se pueden complicar hasta convertirse en imposibles. Para resolverlos se recurre a diferentes técnicas, como la optimización local y algunos métodos de optimización global. Cuando buscamos una solución global tenemos que limitar el número de variables y restricciones, aunque este inconveniente se va superando con el avance de la tecnología. Por ejemplo, el complemento de *Mathematica*, *MathOptimizer* puede abordar hasta mil variables con mil restricciones según la información comercial [Pin10].

En la asignatura de tratamiento digital de señales hemos visto diferentes algoritmos de optimización, y no teníamos el requisito de que las funciones tuviesen que ser convexas, por lo que vimos que era fácil encontrarnos con soluciones que se convergían hacia mínimos locales.

Estos métodos necesitan un punto inicial que puede resultar crítico en la solución obtenida. También resultan de gran importancia los valores de los parámetros de los algoritmos, que tienen que ser ajustados en cada aplicación.

La principal dificultad en estos problemas no se encuentra en el planteamiento, sino en la resolución y en el ajuste de los parámetros mencionados. En cambio, cuando el problema es convexo, veremos que la dificultad se encuentra en encontrar la forma de plantearlo de forma que sea convexo.

2.1.1 Optimización convexa

Los problemas de optimización convexa se exponen de la siguiente forma:

$$\begin{array}{ll} \underset{\boldsymbol{x}}{\text{minimizar}} & f_0(\boldsymbol{x}) \\ \text{sujeto a} & f_i(\boldsymbol{x}) \leq 0, \quad i = 1, \ldots, m \\ & \boldsymbol{a}_i^T \boldsymbol{x} = b_i, \quad i = 1, \ldots, p \end{array} \tag{2.2}$$

Para que el problema sea de optimización convexa necesitamos que la función objetivo y las funciones de restricción sean convexas, es decir, deben satisfacer

$$f(\alpha \boldsymbol{x} + \beta \boldsymbol{y}) \leq \alpha f(\boldsymbol{x}) + \beta f(\boldsymbol{y})$$

Vemos que no sólo tenemos restricciones en forma de desigualdad, sino también en forma de igualdad. En el caso de las igualdades, se requiere que las funciones $h_i(\boldsymbol{x}) = \boldsymbol{a}_i^T \boldsymbol{x} - b_i$ sean afines.

Cuando se cumplen estas condiciones, existen métodos muy eficientes para resolver el problema. El principal objetivo de la teoría que se explica en este trabajo es adquirir suficientes recursos para poder formular nuestro problema en estos términos.

2.2 Conjuntos convexos

Las especificaciones de un problema de optimización planteado según el apartado §2.1 definen un conjunto de vectores desde el que tenemos que seleccionar el óptimo. Este capítulo es importante para conocer las condiciones que deben cumplir esos conjuntos en los problemas convexos.

2.2.1 Definiciones importantes

En este apartado vemos una serie de definiciones que nos van a resultar de gran utilidad en el resto de capítulos.

2.2.1.1 Conjuntos afines, convexos y conos

Si partimos de dos puntos distintos $\boldsymbol{x}_1$ y $\boldsymbol{x}_2$ de $\mathbb{R}^n$, la línea que pasa por estos puntos se define como

$$\boldsymbol{y} = \theta\boldsymbol{x}_1 + (1-\theta)\boldsymbol{x}_2$$

Generalizando a más de dos puntos, una combinación afín de puntos en $\mathbb{R}^n$ es un punto de forma

$$\boldsymbol{y} = \theta_1\boldsymbol{x}_1 + \cdots + \theta_k\boldsymbol{x}_k, \quad \text{donde} \quad \theta_1 + \cdots + \theta_k = 1$$

Un conjunto $\mathcal{C}$ es afín si la línea que une dos puntos cualquiera de $\mathcal{C}$ también pertenece a $\mathcal{C}$, es decir, $\mathcal{C}$ contiene todas las combinaciones afines de sus puntos.

Por ejemplo, el conjunto de soluciones de un sistema de ecuaciones lineales, $\mathcal{C} = \{\boldsymbol{x} \mid \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}\}$, donde $\boldsymbol{A} \in \mathbb{R}^{m\times n}$, y $\boldsymbol{b} \in \mathbb{R}^m$, es un conjunto afín. Se puede comprobar fácilmente partiendo de dos puntos y la definición anterior.

El conjunto de todas las combinaciones de puntos de un conjunto $\mathcal{C}$ se llama la envoltura afín de $\mathcal{C}$, y se escribe $\mathbf{aff}\,\mathcal{C}$.

$$\mathbf{aff}\,\mathcal{C} = \left\{ \theta_1\boldsymbol{x}_1 + \cdots + \theta_k\boldsymbol{x}_k \;\middle|\; \begin{array}{l} \boldsymbol{x}_1, \ldots, \boldsymbol{x}_k \in \mathcal{C}, \\ \theta_1 + \cdots + \theta_k = 1 \end{array} \right\}$$

Se define el interior relativo de un conjunto $\mathcal{C}$ en función de la envoltura afín de $\mathcal{C}$.

$$\mathbf{relint}\,\mathcal{C} = \{\boldsymbol{x} \in \mathcal{C} \mid \quad B(\boldsymbol{x}, r) \cap \mathbf{aff}\,\mathcal{C} \subseteq \mathcal{C} \text{ para algún } r > 0\}$$

Ya hemos visto en la asignatura de métodos matemáticos la definición de una bola de radio r en función de una determinada norma $\|\cdot\|$.

Un conjunto $\mathcal{C}$ es convexo si el segmento que une cualquier par de puntos en $\mathcal{C}$ también pertenece a $\mathcal{C}$.

$$\theta\boldsymbol{x}_1 + (1-\theta)\boldsymbol{x}_2 \in \mathcal{C}, \quad 0 \leq \theta \leq 1$$

Decimos que $\boldsymbol{y}$ es una combinación convexa de los puntos $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_k$ si cumple

$$\boldsymbol{y} = \theta_1 \boldsymbol{x}_1 + \cdots + \theta_k \boldsymbol{x}_k, \quad \theta_1 + \cdots + \theta_k = 1 \quad \theta_i \geq 0,\ i = 1, \ldots, k$$

Un conjunto es convexo si y sólo si contiene todas las combinaciones convexas de sus puntos.

La envoltura convexa de un conjunto $\mathcal{C}$ es el conjunto de todas las combinaciones convexas de los puntos de $\mathcal{C}$.

$$\mathbf{conv}\,\mathcal{C} = \left\{ \theta_1 \boldsymbol{x}_1 + \cdots + \theta_k \boldsymbol{x}_k \;\middle|\; \begin{array}{l} \boldsymbol{x}_i \in \mathcal{C}, \\ \theta_i \geq 0, \quad i = 1, \ldots, k \\ \theta_1 + \cdots + \theta_k = 1 \end{array} \right\}$$

La envoltura convexa es el conjunto convexo más pequeño que contiene a $\mathcal{C}$.

La idea de combinaciones convexas o de combinaciones afines que hemos visto se puede extender a sumas infinitas, a integrales, y a distribuciones de probabilidad.

Decimos que un conjunto $\mathcal{C}$ es un cono si para cada $\boldsymbol{x} \in \mathcal{C}$, con $\theta \geq 0$ se cumple que $\theta \boldsymbol{x} \in \mathcal{C}$. Si además de esto, el conjunto $\mathcal{C}$ es convexo, decimos que es un cono convexo, y se cumple que para cualquier par de puntos $\boldsymbol{x}_1,\ \boldsymbol{x}_2 \in \mathcal{C}$

$$\theta_1 \boldsymbol{x}_1 + \theta_2 \boldsymbol{x}_2 \in \mathcal{C}$$

Una combinación cónica de $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_k$ es un punto $\boldsymbol{y}$ de la forma

$$\boldsymbol{y} = \theta_1 \boldsymbol{x}_1 + \cdots + \theta_k \boldsymbol{x}_k \quad \theta_i \geq 0,\ i = 1, \ldots, k$$

La envoltura cónica de un conjunto $\mathcal{C}$ es el conjunto de todas las combinaciones cónicas en $\mathcal{C}$, y es el cono convexo más pequeño que contiene a $\mathcal{C}$.

2.2.1.1.1 Cono asociado a una norma

El cono asociado a una norma $\|\cdot\|$ es un conjunto convexo definido como

$$\mathcal{C} = \{(\boldsymbol{x}, t) \mid \quad \|\boldsymbol{x}\| \leq t\} \subseteq \mathbb{R}^{n+1} \tag{2.3}$$

Tal como su nombre sugiere, es un cono convexo.

2.2.1.1.2 Cono semidefinido positivo

Utilizaremos con bastante frecuencia la notación $\mathbb{S}^n$ para referirnos al conjunto de matrices simétricas de $n \times n$,

$$\mathbb{S}^n = \{\boldsymbol{X} \in \mathbb{R}^{n \times n} \mid \boldsymbol{X} = \boldsymbol{X}^T\}$$

El conjunto de matrices simétricas y semidefinidas positivas se define como $\mathbb{S}^n_+$, y el de matrices definidas positivas $\mathbb{S}^n_{++}$

$$\mathbb{S}^n_+ = \{\boldsymbol{X} \in \mathbb{S}^n \mid \boldsymbol{X} \succeq \boldsymbol{0}\}$$
$$\mathbb{S}^n_{++} = \{\boldsymbol{X} \in \mathbb{S}^n \mid \boldsymbol{X} \succ \boldsymbol{0}\}$$

Una matriz simétrica $\boldsymbol{A} \in \mathbb{S}^n_+$ es semidefinida positiva si para todo $\boldsymbol{x} \in \mathbb{R}^n$ distinto de cero, se cumple que $\boldsymbol{x}^T\boldsymbol{A}\boldsymbol{x} \geq 0$.

Si la desigualdad es estricta ($>$), entonces será definida positiva.

El símbolo $\succeq$ es una desigualdad generalizada. Cuando comparamos matrices simétricas, decimos que $\boldsymbol{A} \succeq \boldsymbol{B}$ si $\boldsymbol{x}^T\boldsymbol{A}\boldsymbol{x} \geq \boldsymbol{x}^T\boldsymbol{B}\boldsymbol{x}$ para todo $\boldsymbol{x} \in \mathbb{R}^n$. Lo mismo ocurre con las desigualdades estrictas.

El conjunto $\mathbb{S}^n$ es un cono convexo. Si $\theta_1, \theta_2 \geq 0$ y $\boldsymbol{A}, \boldsymbol{B} \in \mathbb{S}^n_+$, entonces $\theta_1\boldsymbol{A} + \theta_2\boldsymbol{B} \in \mathbb{S}^n_+$.

2.2.1.2 Hiperplanos y semiespacios

Un hiperplano es un conjunto de la siguiente forma

$$\{\boldsymbol{x} \mid \boldsymbol{a}^T\boldsymbol{x} = b\}, \quad \boldsymbol{a} \in \mathbb{R}^n,\ \boldsymbol{a} \neq \boldsymbol{0},\ b \in \mathbb{R}$$

A partir de un vector $\boldsymbol{a}$, todos los puntos del hiperplano tienen el mismo producto interno con dicho vector.

Un hiperplano divide a $\mathbb{R}^n$ en dos semiespacios. Un semiespacio cerrado es un conjunto de la forma

$$\{\boldsymbol{x} \mid \boldsymbol{a}^T\boldsymbol{x} \leq b\}$$

Los semiespacios son convexos, pero no son afines. Los puntos del semiespacio forman un ángulo obtuso o recto con el vector $\boldsymbol{a}$. La frontera del semiespacio es el hiperplano

$$\{\boldsymbol{x} \mid \boldsymbol{a}^T\boldsymbol{x} = b\}$$

2.2.1.2.1 Poliedros

Un conjunto de semiespacios hiperplanos representa un poliedro, que se define como la solución de un número finito de igualdades y desigualdades lineales. Siendo estrictos, este término sería un caso particular de politopo en $\mathbb{R}^3$, igual que un polígono lo es en $\mathbb{R}^2$, pero nos atenemos a la nomenclatura de [Boy04] considerando los poliedros como el caso general.

$$\mathcal{P} = \left\{ \boldsymbol{x} \;\middle|\; \begin{array}{ll} \boldsymbol{a}_i^T\boldsymbol{x} \leq b_i, & i = 1, \ldots, m, \\ \boldsymbol{c}_j^T\boldsymbol{x} = d_j, & j = 1, \ldots, p \end{array} \right\}$$

Los poliedros son conjuntos convexos. Se puede expresar también en forma compacta por medio de matrices que contienen los vectores $\boldsymbol{a}_i^T$ y $\boldsymbol{c}_j^T$

$$\mathcal{P} = \{\boldsymbol{x} \mid \quad \boldsymbol{A}\boldsymbol{x} \preceq \boldsymbol{b},\ \boldsymbol{C}\boldsymbol{x} = \boldsymbol{d}\}$$

En este caso, el símbolo $\preceq$ indica desigualdad vectorial o desigualdad por componentes. $\boldsymbol{u} \preceq \boldsymbol{v}$ significa que cada componente u_i es menor o igual que la componente v_i. Observamos que la utilización de éste símbolo comparando vectores no significa lo mismo que comparando matrices simétricas como veíamos en §2.2.1.1.2.

2.2.1.3 Elipsoides

Una elipsoide es una familia de conjuntos convexos que se define de las siguientes formas

$$\begin{aligned} \mathcal{E} &= \{\boldsymbol{x} \quad | \quad (\boldsymbol{x} - \boldsymbol{x}_c)^T \boldsymbol{P}^{-1} (\boldsymbol{x} - \boldsymbol{x}_c) \leq 1\} \\ \mathcal{E} &= \{\boldsymbol{x}_c + \boldsymbol{A}\boldsymbol{u} \quad | \quad \|\boldsymbol{u}\|_2 \leq 1\} \end{aligned} \tag{2.4}$$

El vector $\boldsymbol{x}_c \in \mathbb{R}^n$ es el centro de la elipsoide. La matriz $\boldsymbol{P} \in \mathbb{S}^n_{++}$ determina la extensión de la elipsoide; la longitud de los semiejes de $\mathcal{E}$ se calcula a partir de los autovalores de $\boldsymbol{P}$ como $\sqrt{\lambda_i}$, y su dirección viene determinada por los autovectores de $\boldsymbol{P}$. La equivalencia de las dos expresiones de (2.4) se cumple si $\boldsymbol{A} = \boldsymbol{P}^{1/2}$.

2.2.2 Operaciones que conservan la convexidad

Siguiendo con nuestro interés por mantener la convexidad en los conjuntos para que nuestro problema de optimización sea convexo, esta sección nos permitirá utilizar una serie de operaciones para conseguir hacer que nuestro problema se pueda resolver con eficiencia.

2.2.2.1 Intersecciones

Si los conjuntos $\mathcal{S}_1$ y $\mathcal{S}_2$ son convexos, entonces $\mathcal{S}_1 \cap \mathcal{S}_2$ es convexo. Esta propiedad se extiende a un número infinito de conjuntos.

Un ejemplo son los poliedros, que son la intersección de semiespacios e hiperplanos. Los semiespacios e hiperplanos son conjuntos convexos, por lo que los poliedros también lo son.

2.2.2.2 Funciones afines

Una función afín tiene la forma $f(\boldsymbol{x}) = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{b}$, donde $\boldsymbol{A} \in \mathbb{R}^{m \times n}$ y $\boldsymbol{b} \in \mathbb{R}^m$. Suponemos que el conjunto $\mathcal{S} \subseteq \mathbb{R}^n$ es convexo. La imagen de $\mathcal{S}$ bajo la función afín f es un conjunto convexo. Lo mismo ocurre con la imagen inversa f^{-1} de un conjunto convexo con una función afín.

Como ejemplos de esta propiedad tenemos el escalado y la traslación.

2.2.3 Desigualdades generalizadas

2.2.3.1 Conos propios y desigualdades generalizadas

Un cono $\mathcal{K} \subseteq \mathbb{R}^n$ es un cono propio si satisface lo siguiente

- $\mathcal{K}$ es convexo.
- $\mathcal{K}$ es cerrado.
- $\mathcal{K}$ es sólido, es decir, su interior no está vacío.
- $\mathcal{K}$ tiene punta. Esto significa que no contiene ninguna línea. Dicho de otro modo, si $\mathcal{K}$ contiene a $\boldsymbol{x}$ no puede contener a $-\boldsymbol{x}$ salvo que $\boldsymbol{x} = \mathbf{0}$.

Los conos propios se utilizan para definir una desigualdad generalizada del siguiente modo

$$\boldsymbol{x} \preceq_{\mathcal{K}} \boldsymbol{y} \iff \boldsymbol{y} - \boldsymbol{x} \in \mathcal{K}$$

Esto se conoce como una ordenación parcial. Si la desigualdad es estricta definimos la siguiente ordenación parcial estricta

$$\boldsymbol{x} \prec_{\mathcal{K}} \boldsymbol{y} \iff \boldsymbol{y} - \boldsymbol{x} \in \mathbf{int}\mathcal{K}$$

Cuando $\mathcal{K} = \mathbb{R}_+$, la ordenación parcial $\preceq_{\mathcal{K}}$ es la ordenación normal $\leq$ en $\mathbb{R}$. Si $\mathcal{K} = \mathbb{R}^n_+$ (el ortante no negativo), entonces la desigualdad será lo que hemos visto como la comparación por componentes $\preceq$ entre vectores, y en el caso de $\mathcal{K} = \mathbb{S}^n_+$ tenemos la comparación que hemos visto en §2.2.1.1.2 entre matrices simétricas.

2.2.3.2 Elemento mínimo y minimal

La principal diferencia entre una desigualdad en $\mathbb{R}$ y una ordenación generalizada es que en el caso de $\leq$ en $\mathbb{R}$ se puede comparar cualquier par de puntos, mientras que esto no ocurre con las desigualdades generalizadas. Esto hace que los conceptos de mínimo y máximo sean distintos.

Decimos que $\boldsymbol{x} \in \mathcal{S}$ es el elemento mínimo de $\mathcal{S}$ con respecto a la desigualdad generalizada $\preceq_{\mathcal{K}}$ si se cumple que $\boldsymbol{x} \preceq_{\mathcal{K}} \boldsymbol{y}$ para todo $\boldsymbol{y} \in \mathcal{S}$. De manera similar se define el elemento máximo.

Además de esto, introducimos otro concepto relacionado. Decimos que $\boldsymbol{x} \in \mathcal{S}$ es un elemento minimal de $\mathcal{S}$ con respecto a la desigualdad generalizada $\preceq_{\mathcal{K}}$ si se cumple que $\boldsymbol{y} \preceq_{\mathcal{K}} \boldsymbol{x}$ únicamente en el caso en que $\boldsymbol{y} = \boldsymbol{x}$. También se define un elemento maximal con el mismo razonamiento.

Si existe el elemento mínimo en el conjunto $\mathcal{S}$, éste tiene que ser único; en caso contrario, un conjunto puede tener diferentes minimales.

Un punto $\boldsymbol{x} \in \mathcal{S}$ es el mínimo elemento de $\mathcal{S}$ si y sólo si

$$\mathcal{S} \subseteq \boldsymbol{x} + \mathcal{K}$$

El conjunto $\boldsymbol{x} + \mathcal{K}$ contiene todos los puntos que se pueden comparar con $\boldsymbol{x}$ y son mayores o iguales que $\boldsymbol{x}$ conforme a $\preceq_{\mathcal{K}}$. Un punto $\boldsymbol{x} \in \mathcal{S}$ es minimal de $\mathcal{S}$ si y sólo si

$$(\boldsymbol{x} - \mathcal{K}) \cap \mathcal{S} = \{\boldsymbol{x}\}$$

Aquí tenemos el conjunto $\boldsymbol{x} - \mathcal{K}$ que contiene todos los puntos que se pueden comparar con $\boldsymbol{x}$ y son menores o iguales que $\boldsymbol{x}$.

2.2.4 Conos duales y desigualdades generalizadas

Si $\mathcal{K}$ es un cono, definimos el cono dual de $\mathcal{K}$ como

$$\mathcal{K}^* = \{\boldsymbol{y} \mid \boldsymbol{x}^T\boldsymbol{y} \geq 0 \quad \forall \boldsymbol{x} \in \mathcal{K}\}$$

El cono $\mathcal{K}^*$ siempre será convexo, aunque $\mathcal{K}$ no lo sea.

Si el cono $\mathcal{K}$ es propio, puede definir una desigualdad generalizada $\preceq_{\mathcal{K}}$. En este caso, el cono dual $\mathcal{K}^*$ también será propio y define una desigualdad generalizada $\preceq_{\mathcal{K}^*}$ que conocemos como la dual de la desigualdad $\preceq_{\mathcal{K}}$, y que cumple algunas propiedades importantes.

- $\boldsymbol{x} \preceq_{\mathcal{K}} \boldsymbol{y}$ si y sólo si $\boldsymbol{\lambda}^T\boldsymbol{x} \leq \boldsymbol{\lambda}^T\boldsymbol{y}$ para todo $\boldsymbol{\lambda} \preceq_{\mathcal{K}^*} \mathbf{0}$.
- $\boldsymbol{x} \prec_{\mathcal{K}} \boldsymbol{y}$ si y sólo si $\boldsymbol{\lambda}^T\boldsymbol{x} < \boldsymbol{\lambda}^T\boldsymbol{y}$ para todo $\boldsymbol{\lambda} \preceq_{\mathcal{K}^*} 0$, $\boldsymbol{\lambda} \neq \mathbf{0}$.

2.2.4.1 Elemento mínimo y minimal según desigualdades generalizadas

Decimos que $\boldsymbol{x}$ es el mínimo elemento de $\mathcal{S}$ con respecto a la desigualdad generalizada $\preceq_{\mathcal{K}}$ si y sólo si $\boldsymbol{x}$ es el minimizador único de $\boldsymbol{\lambda}^T\boldsymbol{z}$ con $\boldsymbol{\lambda} \prec_{\mathcal{K}^*} \mathbf{0}$, $\boldsymbol{z} \in \mathcal{S}$. Esto significa que el hiperplano $\{\boldsymbol{z} \mid \boldsymbol{\lambda}^T(\boldsymbol{z} - \boldsymbol{x}) = 0\}$ es un hiperplano soporte estricto para $\mathcal{S}$ en $\boldsymbol{x}$. El hiperplano de soporte estricto quiere decir que la intersección entre $\mathcal{S}$ y el hiperplano es únicamente el punto $\boldsymbol{x}$.

Si $\boldsymbol{\lambda} \prec_{\mathcal{K}^*} \mathbf{0}$ y $\boldsymbol{x}$ minimiza $\boldsymbol{\lambda}^T\boldsymbol{z}$ sobre $\boldsymbol{z} \in \mathcal{S}$, entonces $\boldsymbol{x}$ es minimal.

2.3 Funciones convexas

2.3.1 Condiciones necesarias

Decimos que una función $f : \mathbb{R}^n \to \mathbb{R}$ es convexa si su dominio es un conjunto convexo, y si para todo $\boldsymbol{x}, \boldsymbol{y} \in \mathbf{dom}\, f$ se cumple

$$f(\theta\boldsymbol{x} + (1-\theta)\boldsymbol{y}) \leq \theta f(\boldsymbol{x}) + (1-\theta)f(\boldsymbol{y}) \tag{2.5}$$

Esto se conoce como la desigualdad de Jensen, y se puede extender a combinaciones convexas de más de dos puntos: $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_k \in \mathbf{dom}\, f$, y $\theta_1, \ldots, \theta_k \geq 0$, con $\theta_1 + \cdots + \theta_k = 1$,

$$f(\theta_1\boldsymbol{x}_1 + \cdots + \theta_k\boldsymbol{x}_k) \leq \theta_1 f(\boldsymbol{x}_1) + \cdots + \theta_k f(\boldsymbol{x}_k)$$

Los sumatorios se pueden sustituir por integrales, por esperanza estadística o por sumas infinitas.

Decimos que f es cóncava si $-f$ es convexa.

Una propiedad importante es que una función es convexa si y sólo si lo es cuando nos restringimos a una línea que pasa por su dominio, es decir, si $g(t) = f(\boldsymbol{x} + t\boldsymbol{v})$ es convexa.

En los problemas de optimización convexa trabajaremos con la función de valor extendido

$$\tilde{f}(\boldsymbol{x}) = \begin{cases} f(\boldsymbol{x}) & \boldsymbol{x} \in \mathbf{dom}\, f \\ \infty & \boldsymbol{x} \notin \mathbf{dom}\, f \end{cases}$$

2.3.1.1 Condición de primer orden

Si existe el gradiente ∇f en todos los puntos de su dominio, entonces decimos que f es convexa si y sólo si su dominio es un conjunto convexo, y se cumple lo siguiente

$$f(\boldsymbol{y}) \geq f(\boldsymbol{x}) + \nabla f(\boldsymbol{x})^T(\boldsymbol{y} - \boldsymbol{x}) \tag{2.6}$$

La función afín $f(\boldsymbol{x}) + \nabla f(\boldsymbol{x})^T(\boldsymbol{y} - \boldsymbol{x})$ es la aproximación de Taylor de primer orden de f en torno a $\boldsymbol{x}$. Si $\nabla f(\boldsymbol{x}) = \mathbf{0}$, entonces $f(\boldsymbol{y}) \geq f(\boldsymbol{x})$ para todo $\boldsymbol{y} \in \mathbf{dom}\, f$, por lo que $\boldsymbol{x}$ es un mínimo global de la función f.

2.3.1.2 Condición de segundo orden

Si existe el hessiano $\nabla^2 f$ en todos los puntos del dominio de f, decimos que para que f sea convexa se tiene que cumplir que su dominio sea un conjunto convexo, y que el hessiano sea semidefinido positivo en todos los puntos del dominio.

$$\nabla^2 f(\boldsymbol{x}) \succeq \mathbf{0} \tag{2.7}$$

Para entender esta condición, en el caso de una función en $\mathbb{R}$ se tiene que cumplir que $f''(\boldsymbol{x}) \geq 0$, y esto significa que la derivada es creciente (para ser exacto, no decreciente). En general, $\nabla^2 f(\boldsymbol{x}) \succeq \mathbf{0}$ indica que la gráfica de la función tendrá curvatura positiva en $\boldsymbol{x}$.

2.3.1.3 Ejemplos

A continuación se presenta una lista de ejemplos de funciones convexas o cóncavas. Para comprobar que lo son podemos verificar alguna de las condiciones que acabamos de ver.

2.3.1.3.1 Funciones de variable real, $f : \mathbb{R} \to \mathbb{R}$

- *Funciones afines.* Todas las funciones afines son convexas y cóncavas.
- *Funciones cuadráticas.* $f(x) = (1/2)px^2 + qx + r$ es convexa si $p \geq 0$.

- *Exponencial.* $f(x) = e^{ax}$ es convexa en $\mathbb{R}$ para todo $a \in \mathbb{R}$.
- *Potencia.* $f(x) = x^a$ es convexa en $\mathbb{R}_{++}$ si $a \geq 1$ o $a \leq 0$, y cóncava si $0 \leq a \leq 1$.
- *Potencia del valor absoluto.* $f(x) = |x|^p$ es convexa en $\mathbb{R}$ si $p \geq 1$.
- *Logaritmo.* $f(x) = log(x)$ es cóncava en $\mathbb{R}_{++}$.
- *Entropía no negativa.* $f(x) = xlogx$ es convexa en $\mathbb{R}_{++}$, y si definimos que vale $f(0) = 0$ también es convexa en $\mathbb{R}_+$

2.3.1.3.2 Funciones de variable vectorial, $f : \mathbb{R}^n \to \mathbb{R}$

- *Funciones afines.* Todas las funciones afines son convexas y cóncavas.
- *Funciones cuadráticas.* $f(\boldsymbol{x}) = (1/2)\boldsymbol{x}^T\boldsymbol{P}\boldsymbol{x} + \boldsymbol{q}^T\boldsymbol{x} + r$ es convexa si $\boldsymbol{P} \succeq \boldsymbol{0}$.
- *Normas.* Cualquier norma es convexa, y se comprueba utilizando la desigualdad triangular.
- *Máximo.* $f(\boldsymbol{x}) = \text{máx}_i\, x_i$ es convexa en $\mathbb{R}^n$.
- *Función cuadrática entre lineal.* $f(x, y) = x^2/y$ es convexa en su dominio, que es $\mathbb{R} \times \mathbb{R}_{++}$. En realidad este es un caso particular de función matriz fraccional: $f : \mathbb{R}^n \times \mathbb{S}^n \to \mathbb{R}$, que se define como $f(\boldsymbol{x}, \boldsymbol{Y}) = \boldsymbol{x}^T\boldsymbol{Y}^{-1}\boldsymbol{x}$, y que es una función convexa en todo su dominio $\mathbb{R}^n \times \mathbb{S}^n$.
- *Función log-sum-exp.* $f(\boldsymbol{x}) = log(e^{x_1} + \cdots + e^{x_n})$ es convexa en $\mathbb{R}^n$, y además se puede utilizar como una aproximación diferenciable de la función *máximo*.
- *Media geométrica.* $f(\boldsymbol{x}) = (\prod_{i=1}^n x_i)^{1/n}$ es cóncava en $\mathbf{dom}\, f = \mathbb{R}^n_{++}$.
- *Función log-determinante.* $f(\boldsymbol{X}) = \log\det \boldsymbol{X}$ es cóncava en $\mathbf{dom}\, f = \mathbb{S}^n_{++}$.

2.3.1.4 Condición del epígrafo

El grafo de una función $f : \mathbb{R}^n \to \mathbb{R}$ se define como un subconjunto de $\mathbb{R}^{n+1}$

$$\{(\boldsymbol{x}, f(\boldsymbol{x})) \mid \boldsymbol{x} \in \mathbf{dom}\, f\}$$

El epígrafo de una función se define también como un subconjunto de $\mathbb{R}^{n+1}$

$$\mathbf{epi}\, f\{(\boldsymbol{x}, t) \mid \quad \boldsymbol{x} \in \mathbf{dom}\, f, \quad f(\boldsymbol{x}) \leq t\}$$

Una función es convexa si y sólo si su epígrafo es un conjunto convexo. En este momento encontramos la relación entre el capítulo de conjuntos convexos y las funciones convexas.

Una función es cóncava si y sólo si su hipógrafo es un conjunto convexo. Se define el hipógrafo como

$$\mathbf{hypo}\, f = \{(\boldsymbol{x}, t) \mid t \leq f(\boldsymbol{x})\}$$

2.3.2 Operaciones que conservan la convexidad

Con las siguientes operaciones podremos construir nuevas funciones convexas o cóncavas.

2.3.2.1 Suma ponderada con pesos no negativos

La suma ponderada de funciones convexas es una función convexa si los pesos son positivos o cero. Esto indica que el conjunto de funciones convexas es un cono convexo.

$$f = w_1 f_1 + \cdots + w_m f_m, \qquad w_i \geq 0$$

También se cumple que la suma ponderada de funciones cóncavas es una función cóncava si ningún peso es negativo. Se puede extender el resultado a integrales y sumas infinitas.

2.3.2.2 Función compuesta con un mapa afín

A continuación tenemos la función $g : \mathbb{R}^n \to \mathbb{R}$ compuesta por una función $f : \mathbb{R}^m \to \mathbb{R}$, en la que ponemos como argumento una función afín

$$g(\boldsymbol{x}) = f(\boldsymbol{A}\boldsymbol{x} + \boldsymbol{b}), \qquad \boldsymbol{A} \in \mathbb{R}^{m \times n},\ \boldsymbol{b} \in \mathbb{R}^m$$

Si f es convexa, g también lo es; y si f es cóncava, g también es cóncava.

2.3.2.3 Máximo y supremo puntual

La función $f(\boldsymbol{x}) = \max\{f_1(\boldsymbol{x}), \ldots, f_m(\boldsymbol{x})\}$, donde todas las $f_i(\boldsymbol{x})$ son convexas, es una función convexa. El dominio de f es la intersección de los dominios de las $f_i(\boldsymbol{x})$. Esta propiedad se puede extender al supremo puntual de un conjunto infinito de funciones convexas.

Además, el ínfimo puntual de un conjunto de funciones cóncavas es una función cóncava.

Visto de otro modo, en el supremo puntual de funciones convexas, el epígrafo corresponde a la intersección de los epígrafos de las f_i, y vimos en §2.2.2.1 que se trata de un conjunto convexo.

Esta propiedad es importante, porque casi todas las funciones convexas se pueden expresar como el supremo puntual de una familia de funciones afines. Si el epígrafo es un conjunto convexo, podemos encontrar un hiperplano soporte en cada punto $(\boldsymbol{x}, f(\boldsymbol{x}))$, y este hiperplano será una estimación de $f(\boldsymbol{x})$ que estará siempre por debajo, por lo que decimos que es un subestimador de esta función.

2.3.2.4 Minimización de componentes

Otra función que conserva la convexidad es la minimización con respecto a alguno de los componentes de la función original. Si $f(\boldsymbol{x}, \boldsymbol{y})$ es convexa en $(\boldsymbol{x}, \boldsymbol{y})$, y $\mathcal{C}$ es un conjunto convexo no vacío, entonces la función $g(\boldsymbol{x})$ definida a continuación, es convexa

$$g(\boldsymbol{x}) = \inf_{\boldsymbol{y} \in \mathcal{C}} f(\boldsymbol{x}, \boldsymbol{y})$$

2.3.2.5 Perspectiva de una función

Si $f : \mathbb{R}^n \to \mathbb{R}$, la perspectiva de f es la función $g : \mathbb{R}^{n+1} \to \mathbb{R}$ definida como

$$g(\boldsymbol{x}, t) = tf(\boldsymbol{x}/t)$$

El dominio son los vectores $(\boldsymbol{x}, t)$ tales que $\boldsymbol{x}/t$ está en el dominio de f, siendo $t > 0$. Esta operación mantiene la convexidad de f. Si f es convexa, g también; y si f es cóncava, g es cóncava.

2.3.2.6 Composición

Veremos aquí las condiciones que deben cumplir las funciones $h : \mathbb{R}^k \to \mathbb{R}$ y $\boldsymbol{g} : \mathbb{R}^n \to \mathbb{R}^k$ para que la función compuesta $f = h \circ \boldsymbol{g} : \mathbb{R}^n \to \mathbb{R}$ sea cóncava o convexa. Definimos f como

$$f(\boldsymbol{x}) = h(\boldsymbol{g}(\boldsymbol{x})), \qquad \mathbf{dom}\, f = \{\boldsymbol{x} \in \mathbf{dom}\, \boldsymbol{g} \,|\, \boldsymbol{g}(\boldsymbol{x}) \in \mathbf{dom}\, h\}$$

2.3.2.6.1 Composición escalar

En primer lugar nos fijamos en el caso en que k y n valen 1, por lo que $h : \mathbb{R} \to \mathbb{R}$ y $g : \mathbb{R} \to \mathbb{R}$, y suponemos que h y g tienen segunda derivada en sus dominios. La segunda derivada de una función compuesta es

$$f''(x) = h''(g(x))g'(x)^2 + h'(g(x))g''(x) \tag{2.8}$$

Si g es convexa ($g'' \geq 0$), y h es convexa y creciente ($h'' \geq 0$, y $h' \geq 0$), entonces f es convexa, porque $f'' \geq 0$ según (2.8). Con este tipo de

razonamiento llegamos a los siguientes resultados:

f es convexa	si h es convexa y no decreciente,	y g es convexa.
f es convexa	si h es convexa y no creciente,	y g es cóncava.
f es cóncava	si h es cóncava y no decreciente,	y g es cóncava.
f es cóncava	si h es cóncava y no creciente,	y g es convexa.

Esto también es aplicable cuando $n > 1$ y las funciones no son diferenciables, aunque en las comprobaciones de función no creciente o no decreciente hay que utilizar la función de valor extendido de h.

2.3.2.6.2 Composición vectorial

Ahora hacemos que $k \geq 1$.

$$f(\boldsymbol{x}) = h(g(\boldsymbol{x})) = h(g_1(\boldsymbol{x}), \ldots, g_k(\boldsymbol{x})), \qquad h : \mathbb{R}^k \to \mathbb{R},\ g_i : \mathbb{R}^n \to \mathbb{R}$$

Si volvemos a suponer que $n = 1$, la regla de la cadena en este caso queda

$$f''(x) = g'(x)^T \nabla^2 h(g(x)) g'(x) + \nabla h(g(x))^T g''(x)$$

Y volvemos a hacer el razonamiento para que $f(x)'' \geq 0$ llegando a las siguientes conclusiones

$$f \text{ es convexa si } \begin{cases} h \text{ es convexa,} \\ h \text{ es no decreciente en todos sus argumentos,} \\ \text{y } g_i \text{ son convexas.} \end{cases}$$

$$f \text{ es convexa si } \begin{cases} h \text{ es convexa,} \\ h \text{ es no creciente en todos sus argumentos,} \\ \text{y } g_i \text{ son cóncavas.} \end{cases}$$

$$f \text{ es cóncava si } \begin{cases} h \text{ es cóncava,} \\ h \text{ es no decreciente en todos sus argumentos,} \\ \text{y } g_i \text{ son cóncavas.} \end{cases}$$

$$f \text{ es cóncava si } \begin{cases} h \text{ es cóncava,} \\ h \text{ es no decreciente en todos sus argumentos,} \\ \text{y } g_i \text{ son convexas.} \end{cases}$$

Y en este caso también es aplicable para $n \geq 1$ sin necesidad de que h o g sean diferenciables.

2.3.3 Funciones cuasi-convexas

Este tipo de funciones aparecen en la resolución de problemas de optimización cuasi-convexos que veremos en §3.2.

2.3.3.1 Definición

Una función $f : \mathbb{R}^n \to \mathbb{R}$ es cuasi-convexa si su dominio y todos sus conjuntos subnivel son convexos. Los conjuntos subnivel son

$$\mathcal{S}_\alpha = \{\boldsymbol{x} \in \mathbf{dom}\, f \mid f(\boldsymbol{x}) \leq \alpha\}$$

Una función es cuasi cóncava si $-f$ es cuasi-convexa, o si sus conjuntos supernivel son convexos. Los conjuntos supernivel se definen igual que los subniveles pero con $f(\boldsymbol{x}) \geq \alpha$.

Si una función es cuasi cóncava y cuasi-convexa, entonces es cuasi lineal, y sus conjuntos de nivel ($f(\boldsymbol{x}) = \alpha$) son convexos.

Existe una variante de la desigualdad de Jensen para caracterizar la cuasi-convexidad. Una función f es cuasi-convexa si y sólo si su dominio es convexo y se cumple que

$$f(\theta\boldsymbol{x} + (1-\theta)\boldsymbol{y}) \leq \text{máx}\{f(\boldsymbol{x}), f(\boldsymbol{y})\} \qquad \forall \boldsymbol{x}, \boldsymbol{y} \in \mathbf{dom}\, f,\ 0 \leq \theta \leq 1$$

Esto quiere decir que el valor de una función en un segmento no sobrepasa el máximo de sus valores en los extremos.

Algunas propiedades de las funciones convexas tienen su equivalente en las funciones cuasi-convexas. Por ejemplo, una función es cuasi-convexa si y sólo si es convexa cuando nos restringimos a una línea que pase por su dominio.

2.3.3.2 Condición de primer orden

Si $f : \mathbb{R}^n \to \mathbb{R}$ es diferenciable, decimos que es cuasi-convexa si y sólo si su dominio es convexo y se cumple que

$$f(\boldsymbol{y}) \leq f(\boldsymbol{x}) \implies \nabla f(\boldsymbol{x})^T(\boldsymbol{y} - \boldsymbol{x}) \leq 0$$

Esto implica que $\nabla f(\boldsymbol{x})$ define un hiperplano soporte del conjunto subnivel $\{\boldsymbol{y} \mid f(\boldsymbol{y}) \leq f(\boldsymbol{x})\}$, en el punto $\boldsymbol{x}$. En este caso es posible que $\nabla f(\boldsymbol{x}) = \mathbf{0}$ pero en $\boldsymbol{x}$ no haya un mínimo global de f.

2.3.3.3 Condición de segundo orden

Si f es dos veces diferenciable, decimos que es cuasi-convexa si y sólo si

$$\boldsymbol{y}^T \nabla f(\boldsymbol{x}) = 0 \implies \boldsymbol{y}^T \nabla^2 f(\boldsymbol{x}) \boldsymbol{y} \geq 0$$

En el caso de una función de variable real, esto quiere decir que en cualquier punto con pendiente cero, la segunda derivada es no negativa. En dimensiones mayores implica que $\nabla^2 f(\boldsymbol{x})$ puede tener como mucho un autovalor negativo.

2.3.3.4 Operaciones que conservan la cuasi-convexidad

- *Máximo ponderado con pesos no negativos.* La función $f = \max\{w_1 f_1, \ldots, w_m f_m\}$ es cuasi-convexa si $w_i \geq 0$ y f_i son cuasi-convexas. Esta propiedad se extiende al supremo puntual

$$f(\boldsymbol{x}) = \sup_{\boldsymbol{y} \in \mathcal{C}} (w(\boldsymbol{y}) g(\boldsymbol{x}, \boldsymbol{y})), \qquad w(\boldsymbol{y}) \geq 0$$

  donde $g(\boldsymbol{x}, \boldsymbol{y})$ es cuasi-convexa en $\boldsymbol{x}$ para cualquier valor de $\boldsymbol{y}$.

- *Composición.* Si $g : \mathbb{R}^n \to \mathbb{R}$ es cuasi-convexa y $h : \mathbb{R} \to \mathbb{R}$ es no decreciente, entonces $f = h \circ g$ es cuasi-convexa.

- *Minimización.* Si $f(\boldsymbol{x}, \boldsymbol{y})$ es conjuntamente cuasi-convexa en $\boldsymbol{x}$ y en $\boldsymbol{y}$, y $\mathcal{C}$ es un conjunto convexo, entonces la función $g(\boldsymbol{x}) = \inf_{\boldsymbol{y} \in \mathcal{C}} f(\boldsymbol{x}, \boldsymbol{y})$ es cuasi-convexa.

2.3.4 Convexidad respecto a desigualdades generalizadas

2.3.4.1 Funciones monótonas respecto a una desigualdad generalizada

Una función $f : \mathbb{R}^n \to \mathbb{R}$ es $\mathcal{K}$-no decreciente siendo $\mathcal{K} \subseteq \mathbb{R}^n$ si se cumple que

$$\boldsymbol{x} \preceq_{\mathcal{K}} \boldsymbol{y} \implies f(\boldsymbol{x}) \leq f(\boldsymbol{y})$$

Si la función es diferenciable, la condición para que f sea $\mathcal{K}$-no decreciente es

$$\nabla f(\boldsymbol{x}) \succeq_{\mathcal{K}^*} \mathbf{0} \qquad \forall \boldsymbol{x} \in \mathbf{dom}\, f$$

Es importante tener en cuenta que en este caso el gradiente es no negativo en la desigualdad dual.

2.3.4.2 Funciones convexas respecto a una desigualdad generalizada

Si $\mathcal{K} \subseteq \mathbb{R}^m$ es un cono propio que define la desigualdad $\preceq_{\mathcal{K}}$, decimos que $\boldsymbol{f} : \mathbb{R}^n \to \mathbb{R}^m$ es $\mathcal{K}$-convexa si

$$\boldsymbol{f}(\theta \boldsymbol{x} + (1-\theta)\boldsymbol{y}) \preceq_{\mathcal{K}} \theta \boldsymbol{f}(\boldsymbol{x}) + (1-\theta)\boldsymbol{f}(\boldsymbol{y}), \qquad 0 \leq \theta \leq 1$$

Si la función es diferenciable, la condición de convexidad es

$$\boldsymbol{f}(\boldsymbol{y}) \succeq_{\mathcal{K}} \boldsymbol{f}(\boldsymbol{x}) + D\boldsymbol{f}(\boldsymbol{x})(\boldsymbol{y} - \boldsymbol{x})$$

El operador D es la matriz *Jacobiana* de $\boldsymbol{f}$ en $\boldsymbol{x}$.

2.4 Dualidad

En este apartado vamos a ver la formulación que permite resolver problemas convexos. Por este motivo me he referido a esta sección como la base teórica más importante de esta parte del proyecto. Comenzaremos con la definición de la función lagrangiana y los multiplicadores de Lagrange, que nos permiten manejar las restricciones incorporándolas a la función objetivo. A continuación veremos qué son la función y el problema dual. Esta teoría hace que terminemos produciendo un problema convexo a partir de cualquier problema de optimización. Si partimos de un problema no convexo obtenemos el dual de Lagrange que sí es convexo, y por tanto podremos resolverlo. Este problema (dual) nos dará un límite inferior para el valor óptimo del problema original. Si inicialmente era convexo, y bajo algunas condiciones, entonces el límite se ajusta al valor óptimo del problema original, y para que esto ocurra veremos las condiciones KKT que sirven de base a los algoritmos de resolución que veremos en el capítulo 4.

El estudio de este apartado nos permite comprender la aplicación directa de los conceptos utilizados en algunos problemas de telecomunicaciones. El desarrollo está planteado para abordar problemas de optimización convexa con restricciones definidos con el formato descrito en (2.2). De forma general, para resolverlos necesitaremos algoritmos iterativos como los del tema 4, pero existen algunos casos en los que podemos encontrar la solución de forma analítica, y es ahí donde nos conviene comprender esta teoría para simplificar considerablemente nuestro resultado.

El ejemplo de aplicación más cercano lo tenemos en §7.3.4.1 y en §7.3.4.2, donde planteamos directamente las ecuaciones KKT y llegamos a una solución sencilla por medio de un algoritmo conocido como *water-filling*. En [Esc10] podemos encontrar más referencias a aplicaciones en las que la solución puede venir de utilizar las ecuaciones KKT, y también otras en las que basta con minimizar la función lagrangiana, aunque se trata de problemas con restricciones de igualdad, que se pueden interpretar como un caso particular de las ecuaciones KKT en el que eliminamos las restricciones de desigualdad.

En algunos casos encontraremos aplicaciones en las que la solución se obtiene de forma más eficiente resolviendo el problema dual. Por ejemplo, en [Esc10] encontramos referencias a aplicaciones con este enfoque para gestión dinámica del espectro en sistemas multiportadora, o para llevar a cabo un control equilibrado entre la tasa de transmisión de bits que se puede obtener en un enlace y la tasa de transmisión que se debe proporcionar a cada usuario, o para asignar potencia y agrupaciones de subportadoras en sistemas WiMAX.

2.4.1 Función dual de Lagrange

Partimos de un problema de optimización en forma estándar

$$\begin{array}{ll} \underset{\boldsymbol{x}}{\text{minimizar}} & f_0(\boldsymbol{x}) \\ \text{sujeto a} & f_i(\boldsymbol{x}) \leq 0, \quad i = 1, \ldots, m \\ & h_i(\boldsymbol{x}) = 0, \quad i = 1, \ldots, p \end{array} \tag{2.9}$$

Nuestra variable $\boldsymbol{x} \in \mathbb{R}^n$ estará en el dominio $\mathcal{D}$ comprendido por la intersección de los dominios de las funciones implicadas. Se define la función lagrangiana $L : \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^p \to \mathbb{R}$ asociada al problema (2.9) como

$$L(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{\nu}) = f_0(\boldsymbol{x}) + \sum_{i=1}^{m} \lambda_i f_i(\boldsymbol{x}) + \sum_{i=1}^{p} \nu_i h_i(\boldsymbol{x})$$

Llamamos a λ_i el multiplicador de Lagrange asociado a la desigualdad $f_i(\boldsymbol{x}) \leq 0$, igual que los ν_i son los multiplicadores de Lagrange de sus igualdades correspondientes $h_i(\boldsymbol{x}) = 0$. De forma vectorial, consideramos a $\boldsymbol{\lambda}$ y $\boldsymbol{\nu}$ las variables duales o vectores multiplicadores de Lagrange de nuestro problema.

La función dual de Lagrange es el mínimo de la función lagrangiana con respecto a $\boldsymbol{x}$.

$$g(\boldsymbol{\lambda}, \boldsymbol{\nu}) = \inf_{\boldsymbol{x} \in \mathcal{D}} L(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{\nu})$$

Es importante observar que la función $L(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{\nu})$ es afín con respecto a $\boldsymbol{\lambda}$ y $\boldsymbol{\nu}$, y por tanto, la función dual será el ínfimo puntual de una familia de funciones afines de $(\boldsymbol{\lambda}, \boldsymbol{\nu})$, es decir, $g(\boldsymbol{\lambda}, \boldsymbol{\nu})$ es una función cóncava independientemente de si nuestro problema es o no convexo.

Si un determinado $\tilde{\boldsymbol{x}}$ es un punto viable de nuestro problema (2.9), es decir, $f_i(\tilde{\boldsymbol{x}}) \leq 0$ y $h_i(\tilde{\boldsymbol{x}}) = 0$, se cumple que

$$\sum_{i=1}^{m} \lambda_i f_i(\tilde{\boldsymbol{x}}) + \sum_{i=1}^{p} \nu_i h_i(\tilde{\boldsymbol{x}}) \leq 0, \qquad \text{si } \boldsymbol{\lambda} \succeq \mathbf{0}$$

Esto implica que

$$L(\tilde{\boldsymbol{x}}, \boldsymbol{\lambda}, \boldsymbol{\nu}) = f_0(\tilde{\boldsymbol{x}}) + \sum_{i=1}^{m} \lambda_i f_i(\tilde{\boldsymbol{x}}) + \sum_{i=1}^{p} \nu_i h_i(\tilde{\boldsymbol{x}}) \leq f_0(\tilde{\boldsymbol{x}})$$

y por tanto

$$g(\boldsymbol{\lambda}, \boldsymbol{\nu}) = \inf_{\boldsymbol{x} \in \mathcal{D}} L(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{\nu}) \leq L(\tilde{\boldsymbol{x}}, \boldsymbol{\lambda}, \boldsymbol{\nu}) \leq f_0(\tilde{\boldsymbol{x}})$$

Esto nos lleva a una conclusión importante. Como $g(\boldsymbol{\lambda}, \boldsymbol{\nu}) \leq f_0(\tilde{\boldsymbol{x}})$ se cumple para cualquier punto viable $\tilde{\boldsymbol{x}}$, también se cumplirá en el punto óptimo $f_0(\boldsymbol{x}^*) = p^*$, por lo que

$$g(\boldsymbol{\lambda}, \boldsymbol{\nu}) \leq p^*$$

Esto nos servirá como cota inferior para nuestro valor óptimo p^*. Este límite no nos servirá para nada si $g(\boldsymbol{\lambda}, \boldsymbol{\nu}) = -\infty$, por lo que hablaremos del par $(\boldsymbol{\lambda}, \boldsymbol{\nu})$ como el dual viable cuando $\boldsymbol{\lambda} \succeq \mathbf{0}$ y $(\boldsymbol{\lambda}, \boldsymbol{\nu}) \in \mathbf{dom}\, g$.

2.4.2 El problema dual de Lagrange

Hemos visto que si $\boldsymbol{\lambda} \succeq \mathbf{0}$, el par $(\boldsymbol{\lambda}, \boldsymbol{\nu})$ nos indica una cota inferior para el valor óptimo p^*. El Problema dual de Lagrange consiste en buscar la mejor cota inferior para nuestro problema, es decir

$$\begin{array}{ll} \underset{\boldsymbol{\lambda},\boldsymbol{\nu}}{\text{maximizar}} & g(\boldsymbol{\lambda}, \boldsymbol{\nu}) \\ \text{sujeto a} & \boldsymbol{\lambda} \succeq \mathbf{0} \end{array} \tag{2.10}$$

Por este motivo, el par $(\boldsymbol{\lambda}, \boldsymbol{\nu})$ que es viable en el problema dual se llama dual viable. Este problema es convexo sin depender de que el problema principal lo sea o no. Conocemos como dual óptimo al par $(\boldsymbol{\lambda}^*, \boldsymbol{\nu}^*)$ que nos da una solución óptima al problema (2.10). En muchos casos tenemos que incluir las condiciones del problema dual como restricciones explícitas.

2.4.2.1 Dualidad débil

La mejor cota inferior para el valor óptimo del problema principal p^*, es la solución del problema dual d^*. Conocemos como desigualdad débil a la siguiente expresión

$$d^* \leq p^*$$

Y se cumple sea o no convexo nuestro problema. La diferencia $p^* - d^*$ es la distancia óptima de dualidad. En algunos casos se utiliza este valor como cota inferior para el valor óptimo cuando el problema es difícil de resolver, ya que el problema dual siempre será convexo y permitirá resolver de forma eficiente el valor de d^*.

2.4.2.2 Condición de Slater

Cuando la distancia óptima de dualidad es cero decimos que hay dualidad fuerte, y significa que el problema dual de Lagrange nos da el mejor resultado posible $d^* = p^*$. Para que esto se cumpla se tienen que dar unas determinadas condiciones que se conocen como hipótesis de cualificación.

Uno de los posibles requisitos es la condición de Slater:

$$\exists \boldsymbol{x} \in \mathbf{relint}\, \mathcal{D} \quad \left| \quad \begin{array}{l} f_i(\boldsymbol{x}) < 0, \quad i = 1, \ldots, m \\ \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b} \end{array} \right.$$

Es decir, tiene que existir un punto $\boldsymbol{x}$ estrictamente viable, lo que quiere decir que no sólo cumple las restricciones, sino que cumple las desigualdades con un margen positivo, o de forma estricta ($<$).

Esto es únicamente necesario para las desigualdades que no sean afines. Para refinar la condición de Slater hay que añadir que si algunas de las restricciones de desigualdad son afines, pueden cumplirse como desigualdad no estricta ($\leq$).

La condición de Slater también nos garantiza que se si se cumple (y el problema es convexo) se puede encontrar una solución para el problema dual.

2.4.3 Condiciones de punto óptimo

Si podemos encontrar un punto dual viable $(\boldsymbol{\lambda}, \boldsymbol{\nu})$, nos servirá como prueba que certifica que $p^* \geq g(\boldsymbol{\lambda}, \boldsymbol{\nu})$. Estos puntos nos permiten conocer en qué medida es subóptimo un punto $\boldsymbol{x}$ que sea viable, sin necesidad de conocer el valor exacto de p^*

$$f_0(\boldsymbol{x}) - p^* \leq f_0(\boldsymbol{x}) - g(\boldsymbol{\lambda}, \boldsymbol{\nu})$$

Se conoce como distancia de dualidad asociada al punto viable principal $\boldsymbol{x}$ y al punto viable dual $(\boldsymbol{\lambda}, \boldsymbol{\nu})$, a la distancia entre los objetivos principal y dual $f_0(\boldsymbol{x}) - g(\boldsymbol{\lambda}, \boldsymbol{\nu})$.

2.4.3.1 Holgura complementaria

Suponemos que encontramos los valores óptimos del problema principal y del dual, y que ambos valores son iguales, es decir, que hay dualidad fuerte. Si $\boldsymbol{x}^*$ es el óptimo principal, y $(\boldsymbol{\lambda}^*, \boldsymbol{\nu}^*)$ el óptimo dual,

$$\begin{aligned} f_0(\boldsymbol{x}^*) &= g(\boldsymbol{\lambda}^*, \boldsymbol{\nu}^*) \\ &= \inf_{\boldsymbol{x}} \left(f_0(\boldsymbol{x}) + \sum_{i=1}^{m} \lambda_i^* f_i(\boldsymbol{x}) + \sum_{i=1}^{p} \nu_i^* h_i(\boldsymbol{x}) \right) \\ &\leq f_0(\boldsymbol{x}^*) + \sum_{i=1}^{m} \lambda_i^* f_i(\boldsymbol{x}^*) + \sum_{i=1}^{p} \nu_i^* h_i(\boldsymbol{x}^*) \\ &\leq f_0(\boldsymbol{x}^*) \end{aligned}$$

Esto indica que las desigualdades se tienen que cumplir como igualdades. Una conclusión importante es que

$$\sum_{i=1}^{m} \lambda_i^* f_i(\boldsymbol{x}^*) = 0$$

Como todos los términos de este sumatorio son negativos o cero, llegamos a la condición de Holgura complementaria

$$\lambda_i^* f_i(\boldsymbol{x}^*) = 0, \qquad i = 1, \ldots, m$$

Que se cumple siempre que hay dualidad fuerte. Otra forma de expresarlo es

$$\begin{aligned} \lambda_i^* > 0 &\implies f_i(\boldsymbol{x}^*) = 0 \\ f_i(\boldsymbol{x}^*) < 0 &\implies \lambda_i^* = 0 \end{aligned} \tag{2.11}$$

Cuando una restricción de desigualdad i se cumple como igualdad se dice que está activa. Esto ocurrirá cuando el multiplicador λ_i sea positivo. Cuando no está activa, el multiplicador es cero.

2.4.3.2 Condiciones KKT

Suponemos que las funciones f_0, f_i, h_i son diferenciables, y no ponemos ninguna condición en cuanto a convexidad. Suponemos de nuevo que en los puntos $\boldsymbol{x}^*$ y $(\boldsymbol{\lambda}^*, \boldsymbol{\nu}^*)$ la distancia de dualidad es cero. Como $\boldsymbol{x}^*$ minimiza la función dual de Lagrange $L(\boldsymbol{x}, \boldsymbol{\lambda}^*, \boldsymbol{\nu}^*)$, su gradiente en $\boldsymbol{x}^*$ se tiene que hacer cero

$$\nabla f_0(\boldsymbol{x}^*) + \sum_{i=1}^{m} \lambda_i^* \nabla f_i(\boldsymbol{x}) + \sum_{i=1}^{p} \nu_i^* \nabla h_i(\boldsymbol{x}) = \mathbf{0}$$

Por tanto, para que un problema con funciones diferenciables en el objetivo y en las restricciones tenga dualidad fuerte en los puntos principal y dual $\boldsymbol{x}^*$ y $(\boldsymbol{\lambda}^*, \boldsymbol{\nu}^*)$, se tienen que satisfacer las condiciones de *Karush-Kuhn-Tucker* (KKT):

$$\begin{aligned} f_i(\boldsymbol{x}^*) &\leq 0, \quad i = 1, \ldots, m \\ h_i(\boldsymbol{x}^*) &= 0, \quad i = 1, \ldots, m \\ \lambda_i^* &\geq 0, \quad i = 1, \ldots, m \\ \lambda_i^* f_i(\boldsymbol{x}^*) &= 0, \quad i = 1, \ldots, m \\ \nabla f_0(\boldsymbol{x}^*) + \sum_{i=1}^{m} \lambda_i^* \nabla f_i(\boldsymbol{x}) + \sum_{i=1}^{p} \nu_i^* \nabla h_i(\boldsymbol{x}) &= \mathbf{0} \end{aligned}$$

Si el problema principal es convexo, entonces las condiciones KKT son también suficientes para que los puntos sean óptimos. Por tanto, cualquier par de puntos $\boldsymbol{x}$ y $(\boldsymbol{\lambda}, \boldsymbol{\nu})$ que cumpla las condiciones KKT será óptimo y con distancia de dualidad cero.

2.4.4 Sensibilidad frente a perturbaciones

Cuando se produce la dualidad fuerte, las variables duales óptimas proporcionan una información muy útil acerca de la sensibilidad que tiene el valor óptimo frente a perturbaciones en las restricciones.

Consideramos un problema en el que hemos modificado las restricciones

$$\begin{array}{ll} \underset{\boldsymbol{x}}{\text{minimizar}} & f_0(\boldsymbol{x}) \\ \text{sujeto a} & f_i(\boldsymbol{x}) \leq u_i, \quad i = 1, \ldots, m \\ & h_i(\boldsymbol{x}) = v_i, \quad i = 1, \ldots, p \end{array}$$

Si u_i es positivo significa que hemos relajado la restricción de desigualdad i. Si es negativo hemos reforzado esta restricción. Llamaremos $p^*(\boldsymbol{u}, \boldsymbol{v})$ al

valor óptimo del problema perturbado en función de los valores que hemos añadido.

Suponemos que hay dualidad fuerte, y que $\boldsymbol{x}$ es un punto viable para el problema perturbado. Por tanto,

$$\begin{aligned} p^*(\mathbf{0},\mathbf{0}) = g(\boldsymbol{\lambda}^*,\boldsymbol{\nu}^*) &\leq f_0(\boldsymbol{x}) + \sum_{i=1}^{m} \lambda_i^* f_i(\boldsymbol{x}) + \sum_{i=1}^{p} \nu_i^* h_i(\boldsymbol{x}) \\ &\leq f_0(\boldsymbol{x}) + \boldsymbol{\lambda}^{*T}\boldsymbol{u} + \boldsymbol{\nu}^{*T}\boldsymbol{v} \end{aligned}$$

Esto nos lleva a la siguiente conclusión para cualquier $\boldsymbol{x}$ que sea viable en el problema perturbado

$$p^*(\boldsymbol{u},\boldsymbol{v}) \geq p^*(\mathbf{0},\mathbf{0}) - \boldsymbol{\lambda}^{*T}\boldsymbol{u} - \boldsymbol{\nu}^{*T}\boldsymbol{v}$$

- Si λ_i^* es grande y reforzamos la restricción i ($u_i < 0$), entonces el valor óptimo $p^*(\boldsymbol{u},\boldsymbol{v})$ aumentará mucho.
- Si ν_i^* es grande y positivo, y variamos $v_i < 0$, o si ν_i^* es grande y negativo y variamos $v_i > 0$, entonces el valor óptimo $p^*(\boldsymbol{u},\boldsymbol{v})$ aumentará mucho.
- Si λ_i^* es pequeño y relajamos la restricción i ($u_i > 0$) entonces el valor óptimo $p^*(\boldsymbol{u},\boldsymbol{v})$ no disminuirá mucho.
- Si ν_i^* es pequeño y positivo, y $v_i > 0$, o si ν_i^* es pequeño y negativo y $v_i < 0$, entonces el valor óptimo $p^*(\boldsymbol{u},\boldsymbol{v})$ no disminuirá mucho.

2.4.4.1 Funciones diferenciables

Si $p^*(\boldsymbol{u},\boldsymbol{v})$ es diferenciable en $\boldsymbol{u} = \mathbf{0}$, $\boldsymbol{v} = \mathbf{0}$, y se produce dualidad fuerte, entonces podemos relacionar las variables óptimas duales λ^* y ν^* con el gradiente de $p^*(\boldsymbol{u},\boldsymbol{v})$ en $\boldsymbol{u} = \mathbf{0}$, $\boldsymbol{v} = \mathbf{0}$

$$\lambda_i^* = -\frac{\partial p^*(\mathbf{0},\mathbf{0})}{\partial u_i}, \qquad \nu_i^* = -\frac{\partial p^*(\mathbf{0},\mathbf{0})}{\partial v_i}$$

Los multiplicadores óptimos de Lagrange son exactamente las sensibilidades del valor óptimo con respecto a las perturbaciones. Estas medidas son válidas en la proximidad del punto óptimo, y nos dan una idea cuantitativa de lo activa que es una restricción en el punto óptimo.

Si $f_i(\boldsymbol{x}^*) < 0$ entonces la restricción no está activa, y la podemos reforzar o relajar en pequeñas cantidades sin afectar al valor óptimo. La holgura complementaria nos dice que el multiplicador de Lagrange asociado será igual a cero. Si $f_i(\boldsymbol{x}^*) = 0$ la restricción está activa en el óptimo, y el multiplicador de Lagrange nos indica cuánto afectará la variación de esta restricción. Si λ_i^* es grande significa que al modificar esa restricción el valor óptimo cambiará mucho. Por tanto, si tenemos que modificar alguna restricción empezaremos por las que presenten multiplicadores más pequeños para no cambiar mucho el problema inicial.

Capítulo 3

Planteamiento de problemas convexos

Este capítulo comienza a buscar un enfoque práctico a la teoría que hemos visto hasta ahora. Veremos las técnicas de procesamiento que necesitamos para convertir las especificaciones iniciales de algunos problemas, de forma que se puedan tratar como convexos. Nos encontraremos con las técnicas descritas en §3.1 continuamente en las aplicaciones y en toda la literatura sobre este tema.

En §3.3 tenemos la clasificación de los principales tipos de problemas convexos que vamos a encontrar, y de forma esquemática se resume en la figura 3.1.

Aunque no he tratado ningún caso de problema cuasi-convexo, me ha parecido importante añadir un pequeño apartado sobre este asunto, en §3.2.

En §3.4 se explica cómo abordar problemas en los que las funciones a optimizar sean vectoriales. Aquí nos es especialmente útil la teoría de conos y desigualdades generalizadas estudiada en el tema 2. La utilidad de este apartado se trata de forma indirecta en el tema 6, en el que aparece un caso de escalarización conocido como carga diagonal.

3.1 Problemas equivalentes

Partimos de un problema genérico en formato estándar, no necesariamente convexo

$$\begin{array}{ll} \underset{\boldsymbol{x}}{\text{minimizar}} & f_0(\boldsymbol{x}) \\ \text{sujeto a} & f_i(\boldsymbol{x}) \leq 0, \quad i = 1, \ldots, m \\ & h_i(\boldsymbol{x}) = 0, \quad i = 1, \ldots, p \end{array} \tag{3.1}$$

Decimos que dos problemas son equivalentes si la solución de uno nos conduce directamente a la solución del otro. En esta sección veremos varias transformaciones que podemos aplicar a este problema para modificarlo y obtener un problema equivalente.

3.1.1 Cambio de variables

Si sustituimos nuestra variable $\boldsymbol{x}$ por la función $\phi : \mathbb{R}^n \to \mathbb{R}^n$, decimos que estamos haciendo un cambio de variable $\boldsymbol{x} = \phi(\boldsymbol{z})$, y podremos resolver el problema en función de una nueva variable $\boldsymbol{z}$ que tiene la misma dimensión que $\boldsymbol{x}$.

$$\begin{array}{lll} \underset{\boldsymbol{z}}{\text{minimizar}} & \tilde{f}_0(\boldsymbol{z}) = f_0(\phi(\boldsymbol{z})) & \\ \text{sujeto a} & \tilde{f}_i(\boldsymbol{z}) = f_i(\phi(\boldsymbol{z})) \leq 0, & i = 1, \dots, m \\ & \tilde{h}_i(\boldsymbol{z}) = h_i(\phi(\boldsymbol{z})) = 0, & i = 1, \dots, p \end{array} \tag{3.2}$$

Si encontramos $\boldsymbol{z}^*$ que resuelve el problema (3.2), entonces $\boldsymbol{x}^* = \phi(\boldsymbol{z}^*)$ resuelve el problema original (3.1).

3.1.2 Transformación de las funciones

Podemos transformar la función objetivo y las restricciones utilizando funciones que cumplan las siguientes condiciones:

- $\psi_0 : \mathbb{R} \to \mathbb{R}$ monótona creciente,
- $\psi_1, \dots, \psi_m : \mathbb{R} \to \mathbb{R}$ tales que $\psi_i(u) \leq 0$ si y sólo si $u \leq 0$,
- $\psi_{m+1}, \dots, \psi_{m+p} : \mathbb{R} \to \mathbb{R}$ tales que $\psi_i(u) = 0$ si y sólo si $u = 0$.

El problema equivalente es

$$\begin{array}{lll} \underset{\boldsymbol{x}}{\text{minimizar}} & \tilde{f}_0(\boldsymbol{x}) = \psi_0(f_0(\boldsymbol{x})) & \\ \text{sujeto a} & \tilde{f}_i(\boldsymbol{x}) = \psi_i(f_i(\boldsymbol{x})) \leq 0, & i = 1, \dots, m \\ & \tilde{h}_i(\boldsymbol{z}) = \psi_{m+i}(h_i(\boldsymbol{x})) = 0, & i = 1, \dots, p \end{array}$$

En este caso los valores óptimos estarán en los mismos puntos $\boldsymbol{x}^*$, y los conjuntos viables serán iguales que en el problema original.

Por ejemplo, podemos escalar la función objetivo y las desigualdades multiplicándolas por constantes positivas, y las igualdades multiplicándolas por constantes distintas de cero. Otro ejemplo bastante común lo utilizamos al minimizar la norma euclídea $\|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_2$, que transformamos por el cuadrado $\|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_2^2$. Esto es válido ya que la norma siempre será positiva o cero, por lo que el cuadrado será creciente. En el primer caso la función objetivo no es diferenciable, pero al elevarlo al cuadrado sí lo es y podemos aplicar las técnicas que hemos visto para encontrar el mínimo.

3.1.3 Variables de holgura

Las variables de holgura se utilizan para sustituir a las funciones $f_i(\boldsymbol{x})$ de forma que las desigualdades quedan reducidas a condiciones de no negatividad para variables sencillas, que se llaman variables de holgura, $s_i \in \mathbb{R}$.

La sustitución consiste en $s_i = -f_i(\boldsymbol{x}) \geq 0$, y el problema equivalente es

$$\begin{array}{lll} \underset{\boldsymbol{x},s_i}{\text{minimizar}} & f_0(\boldsymbol{x}) & \\ \text{sujeto a} & s_i \geq 0, & i = 1,\ldots,m \\ & f_i(\boldsymbol{x}) + s_i = 0, & i = 1,\ldots,m \\ & h_i(\boldsymbol{x}) = 0, & i = 1,\ldots,p \end{array}$$

A primera vista puede parecer que estamos complicando el problema porque estamos añadiendo más dimensiones al problema, pero en muchos casos esta simple transformación puede hacer que un problema inicialmente no convexo se convierta en uno que sí lo sea y por tanto se pueda resolver eficientemente.

3.1.4 Eliminación de restricciones de igualdad

En algunos problemas, al hacer un cambio de variable podemos hacer que las funciones h_i se hagan cero. Supongamos que la función $\phi : \mathbb{R}^k \to \mathbb{R}^n$ se hace cero para algún conjunto de parámetros $\boldsymbol{z} \in \mathbb{R}^k$, por lo que $h_i(\phi(\boldsymbol{z})) = 0$. En este caso, el problema equivalente queda

$$\begin{array}{lll} \underset{\boldsymbol{z}}{\text{minimizar}} & f_0(\phi(\boldsymbol{z})) & \\ \text{sujeto a} & f_i(\phi(\boldsymbol{z})) \leq 0, & i = 1,\ldots,m \end{array}$$

3.1.4.1 Eliminación de las restricciones lineales de igualdad

Cuando el problema es convexo, las restricciones de igualdad son afines, y las expresamos como $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$. Como recordatorio de álgebra lineal, llamaremos $\mathcal{R}(\boldsymbol{A})$ al conjunto imagen [1] de la matriz $\boldsymbol{A} \in \mathbb{R}^{p\times n}$,

$$\mathcal{R}(\boldsymbol{A}) = \{\boldsymbol{z} \in \mathbb{R}^p \,|\, \boldsymbol{z} = \boldsymbol{A}\boldsymbol{x}\}$$

con $\boldsymbol{x} \in \mathbb{R}^n$. Para que el problema sea viable $\boldsymbol{b} \in \mathcal{R}(\boldsymbol{A})$, y esta igualdad normalmente tendrá muchas soluciones, ya que $\boldsymbol{A} \in \mathbb{R}^{p\times n}$ con $p < n$. Esto tiene sentido, ya que si la restricción $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ es un sistema de ecuaciones con $p = n$ entonces la solución es $\boldsymbol{x} = \boldsymbol{A}^{-1}\boldsymbol{b}$ sin necesidad de minimizar, y si $p > n$ lo normal es que $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ no tenga solución o que algunas igualdades sean equivalentes entre sí.

Para eliminar las restricciones de igualdad recurrimos al espacio nulo o núcleo de $\boldsymbol{A}$:

$$\mathcal{N}(\boldsymbol{A}) = \{\boldsymbol{x} \,|\, \boldsymbol{A}\boldsymbol{x} = \boldsymbol{0}\}$$

Si elegimos una de las soluciones de $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ y lo llamamos punto $\boldsymbol{x}_0$, y una matriz $\boldsymbol{F} \in \mathbb{R}^{n\times k}$ cuyo conjunto imagen sea el núcleo de $\boldsymbol{A}$, $\mathcal{R}(\boldsymbol{F}) = \mathcal{N}(\boldsymbol{A})$, entonces la solución general del sistema $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ se puede expresar como

[1] La notación $\mathcal{R}$ se debe al término en inglés *range* que podemos confundir con el rango. Para nosotros el rango es el equivalente al término *rank*

$\boldsymbol{F}\boldsymbol{z} + \boldsymbol{x}_0$ con $\boldsymbol{z} \in \mathbb{R}^k$. Sustituimos $\boldsymbol{x} = \boldsymbol{F}\boldsymbol{z} + \boldsymbol{x}_0$ en el problema original y tenemos

$$\begin{array}{ll} \underset{\boldsymbol{z}}{\text{minimizar}} & f_0(\boldsymbol{F}\boldsymbol{z} + \boldsymbol{x}_0) \\ \text{sujeto a} & f_i(\boldsymbol{F}\boldsymbol{z} + \boldsymbol{x}_0) \leq 0, \quad i = 1, \ldots, m \end{array}$$

Con esta transformación conservamos la convexidad de las funciones, y reducimos el número de variables al rango de $\boldsymbol{A}$. Existen varios métodos para obtener $\boldsymbol{F}$ y $\boldsymbol{x}_0$ a partir de un sistema indeterminado de ecuaciones lineales basados en distintas factorizaciones de $\boldsymbol{A}$. Podemos encontrar algunos en el apéndice C.5 de [Boy04].

3.1.5 Introducción de restricciones de igualdad

También podemos añadir restricciones de igualdad y nuevas variables en un problema, por ejemplo en el siguiente caso

$$\begin{array}{lll} \underset{\boldsymbol{x}}{\text{minimizar}} & f_0(\boldsymbol{A}_0\boldsymbol{x} + \boldsymbol{b}_0) & \\ \text{sujeto a} & f_i(\boldsymbol{A}_i\boldsymbol{x} + \boldsymbol{b}_i) \leq 0, & i = 1, \ldots, m \\ & h_i(\boldsymbol{x}) = 0, & i = 1, \ldots, p \end{array}$$

El problema se puede transformar de la siguiente forma

$$\begin{array}{lll} \underset{\boldsymbol{x}, \boldsymbol{y}_0, \boldsymbol{y}_i}{\text{minimizar}} & f_0(\boldsymbol{y}_0) & \\ \text{sujeto a} & f_i(\boldsymbol{y}_i) \leq 0, & i = 1, \ldots, m \\ & \boldsymbol{y}_i = \boldsymbol{A}_i\boldsymbol{x} + \boldsymbol{b}_i, & i = 0, \ldots, m \\ & h_i(\boldsymbol{x}) = 0, & i = 1, \ldots, p \end{array}$$

Si el problema es convexo, debemos tener en cuenta que las igualdades que añadimos tienen que ser afines, como en el ejemplo.

3.1.6 Optimización sobre algunas variables

Siempre podremos escalonar el proceso de minimización, empezando por minimizar respecto a algunas variables

$$\inf_{\boldsymbol{x}, \boldsymbol{y}} f(\boldsymbol{x}, \boldsymbol{y}) = \inf_{\boldsymbol{x}} \tilde{f}(\boldsymbol{x}) = \inf_{\boldsymbol{x}} \left(\inf_{\boldsymbol{y}} f(\boldsymbol{x}, \boldsymbol{y}) \right)$$

3.1.7 Problema en forma de epígrafo

El problema (3.1) se puede expresar en forma de epígrafo del siguiente modo

$$\begin{array}{lll} \underset{\boldsymbol{x}, t}{\text{minimizar}} & t & \\ \text{sujeto a} & f_0(\boldsymbol{x}) - t \leq 0 & \\ & f_i(\boldsymbol{x}) \leq 0, & i = 1, \ldots, m \\ & h_i(\boldsymbol{x}) = 0, & i = 1, \ldots, p \end{array}$$

Con esto conseguimos que la función objetivo f_0 sea una función lineal de $(\boldsymbol{x}, t)$.

3.1.8 Restricciones implícitas y explícitas

Como trabajamos con los valores extendidos de las funciones, podemos incluir cualquier restricción de forma implícita en la función objetivo volviendo a definir su dominio. Por ejemplo, si queremos incluir todas las restricciones en el objetivo escribiremos

$$\underset{\boldsymbol{x}}{\text{minimizar}}\ \tilde{f}_0(\boldsymbol{x}) = \begin{cases} f_0(\boldsymbol{x}) & \text{si } f_i(\boldsymbol{x}) \leq 0, \text{y } h_i(\boldsymbol{x}) = 0 \\ \infty & \text{en otros casos} \end{cases}$$

Simplemente hemos hecho un cambio en la descripción del problema, pero esto no hace que sea más fácil de resolver. En otros casos nos encontraremos con problemas que tengan restricciones implícitas, y las podemos hacer explícitas, por ejemplo

$$\underset{\boldsymbol{x}}{\text{minimizar}}\ f(\boldsymbol{x}) = \begin{cases} \boldsymbol{x}^T\boldsymbol{x} & \text{si } \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b} \\ \infty & \text{en otros casos} \end{cases}$$

Al poner la restricción de forma explícita, la función objetivo es diferenciable

$$\begin{array}{ll} \underset{\boldsymbol{x}}{\text{minimizar}} & \boldsymbol{x}^T\boldsymbol{x} \\ \text{sujeto a} & \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b} \end{array}$$

3.2 Optimización cuasi-convexa

A pesar de no haber tratado con problemas de este tipo en las aplicaciones de este proyecto, la optimización cuasi-convexa es importante en el ámbito de las telecomunicaciones porque se utiliza en una gran cantidad de aplicaciones: por ejemplo, en [Ngo10] se presenta un caso de canal bidireccional con un nodo de retransmisión en el que la asignación óptima de potencias se resuelve con un problema cuasi-convexo; en [Wu96] se formula con este tipo de problemas varios ejemplos de diseño de filtros FIR[2] en forma de desigualdades matriciales lineales (LMI[3]); en [Gu08] podemos ver una aplicación de control de admisión y potencia para redes cognitivas de banda ultra ancha (UWB[4]), en la que se maximiza la velocidad mínima de transmisión resolviendo un problema de optimización cuasi-convexo; en [Shi01] se presenta una técnica para encontrar el umbral óptimo en el problema de detección con hipótesis binaria para un número arbitrario de sensores, en el que la probabilidad de error se demuestra que es una función cuasi-convexa y el problema se descompone en una serie de problemas de este tipo. En definitiva, la lista de ejemplos en los que esta variante de problemas es aplicable a nuestra ingeniería es abundante.

[2] *Finite Impulse Response*
[3] *Linear Matrix Inequality*
[4] *Ultra Wide Band*

3.2.1 Definición

Un problema cuasi-convexo se describe de forma estándar

$$\begin{array}{ll} \underset{\boldsymbol{x}}{\text{minimizar}} & f_0(\boldsymbol{x}) \\ \text{sujeto a} & f_i(\boldsymbol{x}) \leq 0 \quad i = 1, \ldots, m \\ & \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b} \end{array} \tag{3.3}$$

con función objetivo cuasi-convexa, y con funciones de desigualdad convexas. Si las restricciones son cuasi-convexas se pueden sustituir por restricciones convexas equivalentes por tener el mismo conjunto subnivel 0.

Los problemas cuasi-convexos se pueden resolver por medio de una secuencia de problemas convexos. La principal diferencia entre los problemas convexos y los cuasi-convexos es que estos últimos pueden tener óptimos locales que no lo sean globales. Sin embargo, cuando f_0 es derivable, podemos decir que $\boldsymbol{x}$ es óptimo si además de ser factible, cumple lo siguiente para todos los puntos $\boldsymbol{y}$ viables y distintos de $\boldsymbol{x}$

$$\nabla f_0(\boldsymbol{x})^T(\boldsymbol{y} - \boldsymbol{x}) > 0$$

Esta condición es sólo suficiente para ser óptimo. No es necesario que se cumplan para que un punto lo sea. Además, el gradiente no puede valer cero, y esto no pasa cuando la función es convexa.

3.2.2 Solución por medio de problemas de viabilidad

Partimos de una familia de funciones convexas $\phi_t : \mathbb{R}^n \to \mathbb{R}$ que cumplen lo siguiente

$$f_0(\boldsymbol{x}) \leq t \quad \Longleftrightarrow \quad \phi_t(\boldsymbol{x}) \leq 0$$

Además, para cada $\boldsymbol{x}$, $\phi_t(\boldsymbol{x})$ es una función no creciente de t, es decir, $\phi_s(\boldsymbol{x}) \leq \phi_t(\boldsymbol{x})$ cuando $s \geq t$.

Planteamos el siguiente problema de viabilidad, que es un problema convexo

$$\begin{array}{ll} \text{encontrar} & \boldsymbol{x} \\ \text{sujeto a} & \phi_t(\boldsymbol{x}) \leq 0 \\ & f_i(\boldsymbol{x}) \leq 0, \qquad i = 1, \ldots, m \\ & \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b} \end{array} \tag{3.4}$$

Si encontramos que es viable, entonces, el valor óptimo p^* del problema de optimización cuasi-convexo (3.3) será menor o igual que t. Si el problema (3.4) no es viable, entonces podemos deducir que $p^* \geq t$.

Esto sirve para el algoritmo 3.1 que resuelve los problemas cuasi-convexos por bisección, resolviendo un problema convexo de viabilidad en cada paso. Comenzamos con un intervalo $[l, u]$ si sabemos que contiene el valor óptimo p^*, y resolvemos el problema de viabilidad en el punto medio $t = (l + u)/2$ para determinar en qué mitad del intervalo está el óptimo. Esto produce un

Algoritmo 3.1: Método de bisección para optimización cuasi-convexa

Datos: $l \leq p^*$, $u \geq p^*$, tolerancia $\epsilon > 0$

repetir

$t := (l + u)/2$
Resolver el problema de viabilidad (3.4)
Si (3.4) es viable, $u := t$, **si no**, $l := t$

hasta que $u - l \leq \epsilon$

nuevo intervalo que contiene al óptimo. Se repite esta secuencia hasta que la anchura del intervalo es suficientemente pequeña.

3.3 Clasificación de problemas convexos

En las siguientes secciones veremos distintos tipos de problemas convexos, que se pueden agrupar de acuerdo con el esquema de la figura 3.1. Cuando veamos cada tipo de problema entenderemos la disposición de este esquema.

Como ya hemos visto, en un problema de optimización convexa la función objetivo y las restricciones de desigualdad son convexas, y las restricciones de igualdad son afines.

$$\begin{array}{lll} \underset{\boldsymbol{x}}{\text{minimizar}} & f_0(\boldsymbol{x}) & \\ \text{sujeto a} & f_i(\boldsymbol{x}) \leq 0, & i = 1, \ldots, m \\ & \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b} & \boldsymbol{A} \in \mathbb{R}^{p \times n} \end{array} \tag{3.5}$$

3.3.1 Problemas de optimización lineal

Cuando todas las funciones del problema son afines, tanto el objetivo como las restricciones, el problema se llama un problema lineal o LP[5]

$$\begin{array}{lll} \underset{\boldsymbol{x}}{\text{minimizar}} & \boldsymbol{c}^T\boldsymbol{x} + d & \\ \text{sujeto a} & \boldsymbol{G}\boldsymbol{x} \preceq \boldsymbol{h}, & \boldsymbol{G} \in \mathbb{R}^{m \times n} \\ & \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}, & \boldsymbol{A} \in \mathbb{R}^{p \times n} \end{array} \tag{3.6}$$

El conjunto viable es un poliedro. Normalmente se omite la constante d porque no afecta al resultado. Al ser las funciones afines convexas y cóncavas, podemos considerar también el problema de maximización como convexo.

3.3.1.1 LP en formato estándar y en formato desigualdad

En el formato estándar de un problema lineal las desigualdades son únicamente especificaciones de no negatividad para las componentes de $\boldsymbol{x}$

$$\begin{array}{ll} \underset{\boldsymbol{x}}{\text{minimizar}} & \boldsymbol{c}^T\boldsymbol{x} \\ \text{sujeto a} & \boldsymbol{x} \succeq \boldsymbol{0} \\ & \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b} \end{array} \tag{3.7}$$

Para pasar un LP a formato estándar empezamos utilizando variables de holgura

$$\begin{array}{ll} \underset{\boldsymbol{x},\boldsymbol{s}}{\text{minimizar}} & \boldsymbol{c}^T\boldsymbol{x} + d \\ \text{sujeto a} & \boldsymbol{G}\boldsymbol{x} + \boldsymbol{s} = \boldsymbol{h} \\ & \boldsymbol{s} \succeq \boldsymbol{0} \\ & \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b} \end{array}$$

[5] *Linear Program*

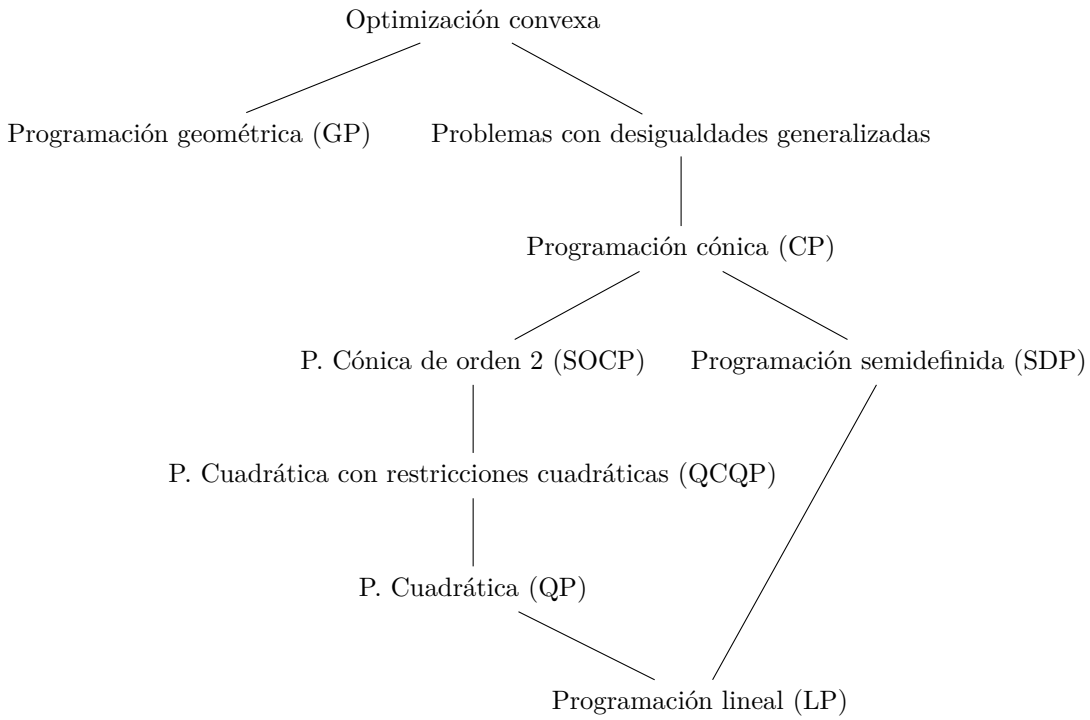

Figura 3.1: Clasificación de problemas de optimización convexa.

A continuación expresamos la variable $\boldsymbol{x}$ como la diferencia de dos variables $\boldsymbol{x}^+$ y $\boldsymbol{x}^-$ no negativas, $\boldsymbol{x} = \boldsymbol{x}^+ + \boldsymbol{x}^-$.

$$\begin{array}{ll} \underset{\boldsymbol{x}^+,\boldsymbol{x}^-,\boldsymbol{s}}{\text{minimizar}} & \boldsymbol{c}^T\boldsymbol{x}^+ + \boldsymbol{c}^T\boldsymbol{x}^- + d \\ \text{sujeto a} & \boldsymbol{G}\boldsymbol{x}^+ + \boldsymbol{G}\boldsymbol{x}^- + \boldsymbol{s} = \boldsymbol{h} \\ & \boldsymbol{A}\boldsymbol{x}^+ + \boldsymbol{A}\boldsymbol{x}^- = \boldsymbol{b} \\ & \boldsymbol{x}^+ \succeq \boldsymbol{0}, \quad \boldsymbol{x}^- \succeq \boldsymbol{0}, \quad \boldsymbol{s} \succeq \boldsymbol{0} \end{array} \tag{3.8}$$

Cuando el programa lineal no tiene igualdades, está en forma de desigualdad

$$\begin{array}{ll} \underset{\boldsymbol{x}}{\text{minimizar}} & \boldsymbol{c}^T\boldsymbol{x} \\ \text{sujeto a} & \boldsymbol{A}\boldsymbol{x} \preceq \boldsymbol{b} \end{array} \tag{3.9}$$

3.3.1.2 Ejemplos

3.3.1.2.1 Función lineal por partes.

Consideramos el problema de minimizar una función convexa lineal por partes

$$f(\boldsymbol{x}) = \max_i(\boldsymbol{a}_i^T\boldsymbol{x} + b)$$

Podemos transformar este problema en un LP equivalente con la técnica del epígrafo. En primer lugar minimizamos t sujeto a $\max_i(\boldsymbol{a}_i^T\boldsymbol{x}+b_i) \leq t$, y a continuación expresamos el máximo como m desigualdades separadas

$$\begin{array}{ll} \underset{\boldsymbol{x},t}{\text{minimizar}} & t \\ \text{sujeto a} & \boldsymbol{a}_i^T\boldsymbol{x}+b_i \leq t, \qquad i=1,\ldots,m \end{array}$$

3.3.1.2.2 Programa de fracción lineal.

Un programa de fracción lineal minimiza la división de funciones afines entre un poliedro

$$\begin{array}{ll} \underset{\boldsymbol{x}}{\text{minimizar}} & f_0(\boldsymbol{x}) = \dfrac{\boldsymbol{c}^T\boldsymbol{x}+d}{\boldsymbol{e}^T\boldsymbol{x}+f} \\ \text{sujeto a} & \boldsymbol{G}\boldsymbol{x} \preceq \boldsymbol{h} \\ & \boldsymbol{A}\boldsymbol{x} \preceq \boldsymbol{b} \end{array} \tag{3.10}$$

La función $f_0(\boldsymbol{x})$ es cuasi-convexa si su dominio son las $\boldsymbol{x}$ que producen un denominador positivo. Si el conjunto viable no está vacío, el problema (3.10) se puede transformar en un programa lineal equivalente. Para ello hacemos el cambio de variables

$$\boldsymbol{y} = \frac{\boldsymbol{x}}{\boldsymbol{e}^T\boldsymbol{x}+f}, \qquad z = \frac{1}{\boldsymbol{e}^T\boldsymbol{x}+f}$$

Y lo aplicamos en el problema (3.10) del siguiente modo

$$\begin{array}{ll} \underset{\boldsymbol{y},z}{\text{minimizar}} & \boldsymbol{c}^T\boldsymbol{y}+dz \\ \text{sujeto a} & \boldsymbol{G}\boldsymbol{y}+\boldsymbol{h}z \preceq \boldsymbol{0} \\ & \boldsymbol{A}\boldsymbol{y}-\boldsymbol{b}z = \boldsymbol{0} \\ & \boldsymbol{e}^T\boldsymbol{y}+fz = 1 \\ & z \geq 0 \end{array} \tag{3.11}$$

El programa de fracción lineal se puede generalizar como el máximo puntual de un conjunto de fracciones lineales

$$f_0(\boldsymbol{x}) = \max_i \frac{\boldsymbol{c}_i^T\boldsymbol{x}+d_i}{\boldsymbol{e}_i^T\boldsymbol{x}+f_i}$$

3.3.2 Problemas de optimización cuadráticos

Si la función objetivo es cuadrática y las restricciones son afines, nuestro problema es un programa cuadrático o QP[6]

$$\begin{array}{lll} \underset{\boldsymbol{x}}{\text{minimizar}} & (1/2)\boldsymbol{x}^T\boldsymbol{P}\boldsymbol{x}+\boldsymbol{q}^T\boldsymbol{x}+r, & \boldsymbol{P} \in \mathbb{S}_+^n \\ \text{sujeto a} & \boldsymbol{G}\boldsymbol{x} \preceq \boldsymbol{h}, & \boldsymbol{G} \in \mathbb{R}^{m\times n} \\ & \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}, & \boldsymbol{A} \in \mathbb{R}^{p\times n} \end{array} \tag{3.12}$$

[6] *Quadratic Program*

Si la función objetivo y las desigualdades son funciones cuadráticas, entonces tenemos un programa cuadrático con restricciones cuadráticas o QCQP[7]

$$\begin{array}{ll} \underset{\boldsymbol{x}}{\text{minimizar}} & (1/2)\boldsymbol{x}^T\boldsymbol{P}_0\boldsymbol{x} + \boldsymbol{q}_0^T\boldsymbol{x} + r_0 \\ \text{sujeto a} & (1/2)\boldsymbol{x}^T\boldsymbol{P}_i\boldsymbol{x} + \boldsymbol{q}_i^T\boldsymbol{x} + r_i \leq 0, \quad i = 1, \ldots, m \\ & \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b} \end{array} \tag{3.13}$$

Por supuesto, aquí también se utilizan matrices simétricas semidefinidas positivas para que las formas cuadráticas sean convexas.

3.3.3 Programación cónica de segundo orden

Conocemos como programa cónico de segundo orden a un tipo de problemas en el que cada desigualdad describe un cono asociado a la norma $\|\cdot\|_2$

$$\begin{array}{lll} \underset{\boldsymbol{x}}{\text{minimizar}} & \boldsymbol{f}^T\boldsymbol{x} & \\ \text{sujeto a} & \|\boldsymbol{A}_i\boldsymbol{x} + \boldsymbol{b}_i\|_2 \leq \boldsymbol{c}_i^T\boldsymbol{x} + d_i, & \boldsymbol{A}_i \in \mathbb{R}^{n_i \times n} \\ & \boldsymbol{F}\boldsymbol{x} = \boldsymbol{g}, & \boldsymbol{F} \in \mathbb{R}^{p \times n} \end{array} \tag{3.14}$$

Las desigualdades $\|\boldsymbol{A}_i\boldsymbol{x} + \boldsymbol{b}_i\|_2 \leq \boldsymbol{c}_i^T\boldsymbol{x} + d_i$ se llaman restricciones cónicas de segundo orden, porque equivalen a que los puntos $(\boldsymbol{A}_i\boldsymbol{x} + \boldsymbol{b}_i, \boldsymbol{c}_i^T\boldsymbol{x} + d_i)$ pertenezcan al cono definido por la norma euclídea ℓ^2, según vimos en la definición (2.3).

Las desigualdades en las que $\boldsymbol{c}_i = 0$ describen funciones cuadráticas, por lo que vemos la relación de la figura 3.1 en la que los problemas QCQP son un caso particular de programas SOCP[8] para los que todas las desigualdades cumplen que $\boldsymbol{c}_i = 0$.

3.3.4 Programación con desigualdades generalizadas

Este tipo de problemas es una generalización de los problemas convexos, y se describen del siguiente modo

$$\begin{array}{ll} \underset{\boldsymbol{x}}{\text{minimizar}} & f_0(\boldsymbol{x}) \\ \text{sujeto a} & \boldsymbol{f}_i(\boldsymbol{x}) \preceq_{\mathcal{K}_i} \boldsymbol{0} \quad i = 1, \ldots, m \\ & \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b} \end{array} \tag{3.15}$$

El problema (3.5) es un caso particular en el que se utiliza el cono $\mathcal{K}_i = \mathbb{R}_+$.

3.3.4.1 Problemas en forma cónica

Un caso sencillo de problema con desigualdades generalizadas son los programas cónicos o CP[9] en los que las funciones utilizadas son afines y las

[7] *Quadratically Constrained Quadratic Program*
[8] *Second Order Cone Program*
[9] *Conic Program*

comparaciones se hacen con respecto al mismo cono propio $\mathcal{K}$

$$\begin{array}{ll} \underset{\boldsymbol{x}}{\text{minimizar}} & \boldsymbol{c}^T\boldsymbol{x} \\ \text{sujeto a} & \boldsymbol{F}\boldsymbol{x}+\boldsymbol{g} \preceq_{\mathcal{K}} \boldsymbol{0} \\ & \boldsymbol{A}\boldsymbol{x}=\boldsymbol{b} \end{array} \tag{3.16}$$

Se trata de la generalización de los problemas lineales, y también tienen su equivalente en formato estándar y en formato de desigualdad.

3.3.4.2 Programación semidefinida

Cuando $\mathcal{K}$ es $\mathbb{S}^k_+$, los problemas cónicos se conocen como programas semidefinidos o SDP[10]

$$\begin{array}{ll} \underset{\boldsymbol{x}}{\text{minimizar}} & \boldsymbol{c}^T\boldsymbol{x} \\ \text{sujeto a} & x_1\boldsymbol{F}_1+\cdots+x_n\boldsymbol{F}_n+\boldsymbol{G} \preceq \boldsymbol{0} \\ & \boldsymbol{A}\boldsymbol{x}=\boldsymbol{b} \end{array} \tag{3.17}$$

Las desigualdades en este caso se conocen como desigualdades matriciales lineales (LMI) y están formadas por matrices simétricas, $\boldsymbol{G}, \boldsymbol{F}_1, \ldots, \boldsymbol{F}_n \in \mathbb{S}^k$.

Cuando todas estas matrices son diagonales, el problema se reduce a un LP, como vemos en la figura 3.1. También encontraremos estos problemas en formato estándar y en formato desigualdad.

3.3.5 Programación geométrica

Los programas geométricos no son convexos en su forma natural, pero se pueden convertir fácilmente con un cambio de variables y una transformación en las funciones que lo componen.

Llamamos monomio a la función $f : \mathbb{R}^n \to \mathbb{R}$ con $\mathbf{dom}\, f = \mathbb{R}^n_{++}$

$$f(\boldsymbol{x}) = cx_1^{a_1}x_2^{a_2}\cdots x_n^{a_n}, \qquad c>0,\ a_i \in \mathbb{R} \tag{3.18}$$

Una suma de monomios es un posinomio

$$f(\boldsymbol{x}) = \sum_{k=1}^{K} c_k x_1^{a_{1k}} x_2^{a_{2k}} \cdots x_n^{a_{nk}} \tag{3.19}$$

Un programa geométrico o GP se describe como:

$$\begin{array}{lll} \underset{\boldsymbol{x}}{\text{minimizar}} & f_0(\boldsymbol{x}) = \sum_{k=1}^{K_0} c_{0k} x_1^{a_{0k_1}} x_2^{a_{0k_2}} \cdots x_n^{a_{0k_n}} & \\ \text{sujeto a} & f_i(\boldsymbol{x}) = \sum_{k=1}^{K_i} c_{ik} x_1^{a_{ik_1}} x_2^{a_{ik_2}} \cdots x_n^{a_{ik_n}} \leq 1, & i=1,\ldots,m \\ & h_i(\boldsymbol{x}) = \mathrm{h}_i x_1^{q_{i1}} x_2^{q_{i2}} \cdots x_n^{q_{in}} = 1, & i=1,\ldots,p \end{array} \tag{3.20}$$

[10] *SemiDefinite Program*

La función objetivo y las restricciones de desigualdad son posinomios, y las igualdades son monomios. El dominio es $\mathcal{D} = \mathbb{R}^n_{++}$, y hay que tener en cuenta una restricción implícita: $\boldsymbol{x} \succ \mathbf{0}$.

3.3.5.1 Programación geométrica en forma convexa

Para convertir un problema geométrico en convexo, hacemos el cambio de variables $y_i = \log x_i$, por lo que $x_i = e^{y_i}$. Este cambio convierte cada monomio en la exponencial de una función afín, de forma que el problema (3.20) queda

$$\begin{array}{lll} \underset{\boldsymbol{y}}{\text{minimizar}} & \sum_{k=1}^{K_0} e^{\boldsymbol{a}_{0k}^T \boldsymbol{y} + b_{0k}} & \\ \text{sujeto a} & \sum_{k=1}^{K_i} e^{\boldsymbol{a}_{ik}^T \boldsymbol{y} + b_{ik}}, & i = 1, \ldots, m \\ & e^{\boldsymbol{g}_i^T \boldsymbol{y} + h_i} = 1, & i = 1, \ldots, p \end{array}$$

donde $b_{ik} = \log c_{ik}$ y $h_i = \log \mathrm{h}_i$. A continuación tomamos el logaritmo de todas las funciones y nos queda el problema geométrico en forma convexa

$$\begin{array}{lll} \underset{\boldsymbol{y}}{\text{minimizar}} & \log\left(\sum_{k=1}^{K_0} e^{\boldsymbol{a}_{0k}^T \boldsymbol{y} + b_{0k}}\right) & \\ \text{sujeto a} & \log\left(\sum_{k=1}^{K_i} e^{\boldsymbol{a}_{ik}^T \boldsymbol{y} + b_{ik}}\right), & i = 1, \ldots, m \\ & \boldsymbol{g}_i^T \boldsymbol{y} + h_i = 0, & i = 1, \ldots, p \end{array} \tag{3.21}$$

Como hemos visto, estos problemas se pueden expresar como problemas geométricos en forma posinomial, según (3.20), o en forma convexa, según (3.21).

3.4 Optimización vectorial

En el apartado §3.3.4 hemos visto problemas en los que las restricciones de desigualdad estaban formados por funciones vectoriales $f_i : \mathbb{R}^n \to \mathbb{R}^{k_i}$, pero ahora veremos qué ocurre cuando la función objetivo es vectorial $f_0 : \mathbb{R}^n \to \mathbb{R}^q$. En este caso tenemos que minimizar con respecto a un cono propio $\mathcal{K} \subseteq \mathbb{R}^q$.

$$\begin{array}{lll} \text{minimizar respecto a } \mathcal{K} & \underset{\boldsymbol{x}}{\boldsymbol{f}_0(\boldsymbol{x})} & \\ \text{sujeto a} & f_i(\boldsymbol{x}) \leq 0, & i = 1, \ldots, m \\ & h_i(\boldsymbol{x}) = 0, & i = 1, \ldots, p \end{array} \tag{3.22}$$

Si tenemos dos puntos viables $\boldsymbol{x}$ e $\boldsymbol{y}$, decimos que $\boldsymbol{x}$ es mejor que $\boldsymbol{y}$ cuando $f_0(\boldsymbol{x}) \preceq_{\mathcal{K}} f_0(\boldsymbol{y})$, pero puede haber casos en los que no podamos comparar estos valores, porque ninguno es mejor que el otro, es decir, ni $f_0(\boldsymbol{x}) \preceq_{\mathcal{K}} f_0(\boldsymbol{y})$ ni $f_0(\boldsymbol{y}) \preceq_{\mathcal{K}} f_0(\boldsymbol{x})$.

Consideramos el conjunto de valores objetivo para los puntos viables

$$\mathcal{O} = \{f_0(\boldsymbol{x}) \mid \exists \boldsymbol{x} \in \mathcal{D},\ f_i(\boldsymbol{x}) \leq 0,\ h_i(\boldsymbol{x}) = 0\} \subseteq \mathbb{R}^q$$

Si este conjunto tiene un mínimo tal como vimos en el apartado §2.2.3.2, entonces será el valor óptimo del problema, y además será un valor único. Si $\boldsymbol{x}^*$ es el punto óptimo, entonces $f_0(\boldsymbol{x}^*)$ se puede comparar con el objetivo de todos los demás puntos viables, y será mejor o igual que en ellos. Un punto $\boldsymbol{x}^*$ es óptimo si y sólo si es viable y cumple que

$$\mathcal{O} \subseteq f_0(\boldsymbol{x}^*) + \mathcal{K}$$

3.4.1 Óptimos de Pareto

En la mayoría de los problemas vectoriales no hay un valor óptimo por no existir el elemento mínimo de $\mathcal{O}$. En estos casos, lo que sí existe son elementos *minimales*. Decimos que un punto $\boldsymbol{x}$ es *eficiente*, o es óptimo de Pareto si $f_0(\boldsymbol{x})$ es un elemento minimal del conjunto de valores objetivo alcanzables. Esto quiere decir que ningún otro punto tendrá un valor objetivo mejor que el óptimo.

Decimos que $\boldsymbol{x}$ es óptimo de Pareto si y sólo si es viable y se cumple lo siguiente

$$(f_0(\boldsymbol{x}) - \mathcal{K}) \cap \mathcal{O} = \{f_0(\boldsymbol{x})\}$$

El conjunto $f_0(\boldsymbol{x}) - \mathcal{K}$ contiene los valores que son mejores o iguales que $f_0(\boldsymbol{x})$. Un problema de optimización vectorial puede tener muchos valores óptimos de Pareto, y estos estarán en el borde del conjunto $\mathcal{O}$.

3.4.2 Escalarización

Tal como vimos en el apartado §2.2.4.1, $f_0(\boldsymbol{x})$ es un punto minimal del conjunto $\mathcal{O}$ si para $\boldsymbol{\lambda} \succ_{\mathcal{K}^*} \mathbf{0}$, $f_0(\boldsymbol{x})$ minimiza $\boldsymbol{\lambda}^T f_0(\boldsymbol{z})$ en $f(\boldsymbol{z}) \in \mathcal{O}$. Entonces, en nuestro problema, elegimos cualquier $\boldsymbol{\lambda} \succ_{\mathcal{K}^*} \mathbf{0}$ y resolvemos el siguiente problema escalar

$$\begin{array}{lll} \underset{\boldsymbol{x}}{\text{minimizar}} & \boldsymbol{\lambda}^T f_0(\boldsymbol{x}) & \\ \text{sujeto a} & f_i(\boldsymbol{x}) \leq 0, & i = 1, \ldots, m \\ & h_i(\boldsymbol{x}) = 0, & i = 1, \ldots, p \end{array} \tag{3.23}$$

Se puede utilizar esta técnica para cualquier problema de optimización vectorial. A medida que modificamos el vector de pesos $\boldsymbol{\lambda}$ obtendremos diferentes soluciones óptimas de Pareto, aunque en algunos casos no encontraremos todas las soluciones posibles de Pareto por este método.

3.4.2.1 Escalarización en problemas convexos

Si el problema vectorial (3.22) es convexo, entonces el problema escalarizado (3.23) también lo es. En este caso, con el método de escalarización podemos encontrar todos los puntos óptimos de Pareto variando $\boldsymbol{\lambda} \succeq_{\mathcal{K}^*} \mathbf{0}$. Pero hay que tener cuidado porque esta desigualdad no es estricta y nos puede dar valores que no sean óptimos de Pareto. Si la desigualdad es estricta, $\boldsymbol{\lambda} \succ_{\mathcal{K}^*} \mathbf{0}$, todos los puntos serán minimales.

3.4.2.2 Optimización multicriterio

Cuando el cono utilizado es $\mathcal{K} = \mathbb{R}^q_+$, nuestro problema de optimización es *multicriterio* o *multiobjetivo*. Significa que cada componente de f_0 es un objetivo escalar que queremos minimizar. Nos referimos a cada objetivo del problema como F_i. Si todas las F_i y las restricciones f_i son convexas, y las h_i son afines, entonces nuestro problema de optimización multicriterio es convexo.

Cuando decimos que $\boldsymbol{x}$ es mejor que $\boldsymbol{y}$, siendo ambos viables, esto significa que $F_i(\boldsymbol{x}) \leq F_i(\boldsymbol{y})$ en todos los objetivos, y además, al menos en uno de ellos $F_j(\boldsymbol{x}) < F_j(\boldsymbol{y})$.

Un punto óptimo $\boldsymbol{x}^*$ cumple que $F_i(\boldsymbol{x}^*) \leq F_i(\boldsymbol{y})$ en todos los objetivos para cualquier $\boldsymbol{y}$ viable.

Capítulo 4

Algoritmos de minimización

Este capítulo explica el funcionamiento de los algoritmos más importantes utilizados en la mayoría de las herramientas de optimización convexa. Su exposición está orientada a comprender la base de los métodos de punto interior, comenzando con los casos sin restricciones que ya hemos visto en la asignatura de tratamiento digital de señales, para continuar de forma progresiva con la forma en que se añaden las restricciones y cómo afectan a dichos algoritmos. Veremos cómo el funcionamiento de estos métodos consiste en convertir el problema convexo en una secuencia de problemas con restricciones de igualdad, y cada uno de estos últimos se convierten en una secuencia de ecuaciones lineales gracias a las condiciones KKT que vimos en el tema 2.

Al final del tema se recogen también el método símplex y una introducción a los algoritmos utilizados para resolver problemas con variables enteras, ya que encontraremos múltiples referencias a estos en las herramientas de resolución del capítulo 5.

4.1 Métodos para minimizar sin restricciones

Empezamos viendo métodos para minimizar una función $f : \mathbb{R}^n \to \mathbb{R}$ convexa y diferenciable. La condición que tiene que cumplir un punto $\boldsymbol{x}^*$ para ser óptimo es que $\nabla f(\boldsymbol{x}^*) = \mathbf{0}$. Esto es un sistema de n ecuaciones con n variables $x_1, \ldots, x_n$. En algunos casos la solución puede ser analítica, pero lo normal es que tengamos que utilizar un algoritmo iterativo que calcule una secuencia de puntos $\boldsymbol{x}^{(0)}, \boldsymbol{x}^{(1)}, \ldots \in \mathbf{dom}\, f$ de forma que $f(\boldsymbol{x}^{(k)})$ tienda a p^* cuando $k \to \infty$. El algoritmo finaliza cuando $f(\boldsymbol{x}^{(k)}) - p^* \leq \epsilon$, donde ϵ es un escalar mayor que cero que representa la tolerancia permitida en la solución, o la precisión del resultado.

4.1.1 Métodos de descenso

La secuencia de minimización que producen estos algoritmos es la siguiente

$$\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} + t^{(k)}\Delta\boldsymbol{x}^{(k)}$$

con $t > 0$ excepto cuando $\boldsymbol{x}^{(k)}$ es óptimo. El símbolo $\Delta\boldsymbol{x}$ es un vector en $\mathbb{R}^n$ llamado paso o dirección de búsqueda. El escalar $t^{(k)} \geq 0$ es el tamaño del paso en la iteración k, aunque en realidad este nombre no es estrictamente correcto, ya que no vale $\|\boldsymbol{x}^{(k+1)} - \boldsymbol{x}^{(k)}\|$, salvo que $\|\Delta\boldsymbol{x}^{(k)}\|$ valga 1.

Decimos que son métodos de descenso porque $f(\boldsymbol{x}^{(k+1)}) < f(\boldsymbol{x}^{(k)})$ excepto cuando $\boldsymbol{x}^{(k)}$ es óptimo. Esto implica que todos los $\boldsymbol{x}^{(k)}$ están dentro del conjunto subnivel inicial. Este conjunto se define como

$$\mathcal{S} = \{\boldsymbol{x} \in \mathbf{dom}\, f \mid f(\boldsymbol{x}) \leq f(\boldsymbol{x}^{(0)})\}$$

Si la función es convexa, de acuerdo con la expresión (2.6) del apartado §2.3.1.1, se cumple que

$$\nabla f(\boldsymbol{x}^{(k)})^T(\boldsymbol{x}^{(k+1)} - \boldsymbol{x}^{(k)}) \geq 0 \quad \Longrightarrow \quad f(\boldsymbol{x}^{(k+1)}) \geq f(\boldsymbol{x}^{(k)})$$

Para buscar el mínimo no nos podemos permitir que esto ocurra. Por tanto, una dirección de búsqueda en un método de descenso tiene que cumplir que

$$\nabla f(\boldsymbol{x}^{(k)})^T\Delta\boldsymbol{x}^{(k)} < 0$$

es decir, $\Delta\boldsymbol{x}^{(k)}$ tiene que formar un ángulo agudo con el gradiente negativo. Si lo cumple, decimos que tenemos una dirección descendente.

El algoritmo 4.1 se basa en alternar dos fases principales: determinar la dirección de descenso $\Delta\boldsymbol{x}$, y seleccionar un tamaño de paso t.

Algoritmo 4.1: Método general de descenso

Datos: un punto inicial $\boldsymbol{x} \in \mathbf{dom}\, f$

repetir

Determinar una dirección de descenso $\Delta\boldsymbol{x}$
Seguimiento lineal. Elegir un tamaño de paso $t > 0$
Actualizar, $\boldsymbol{x} := \boldsymbol{x} + t\Delta\boldsymbol{x}$.

hasta que *se satisface el criterio de parada*

La diferencia entre los distintos métodos de descenso está en la dirección elegida. En este trabajo veremos como ejemplo el descenso de gradiente en §4.1.1.1. Otras opciones son el descenso por máxima pendiente [Boy04], el de gradientes conjugados [Hes52], los métodos de direcciones viables [Pol72], o el de gradiente reducido generalizado [Aba69].

El siguiente paso del algoritmo 4.1, el seguimiento lineal, se llama así porque la selección del tamaño de paso determina dónde estará la siguiente iteración dentro de la línea $\{\boldsymbol{x} + t\Delta\boldsymbol{x} \mid t \in \mathbb{R}_+\}$. Las opciones más comunes que encontraremos son las que tratamos en §4.1.1.2 y §4.1.1.3.

4.1.1.1 Descenso de gradiente

Una elección lógica de dirección de búsqueda es la opuesta al gradiente, $\Delta\boldsymbol{x} = -\nabla f(\boldsymbol{x})$, que corresponde al algoritmo de descenso de gradiente. De hecho, es el método más obvio que se nos puede ocurrir para minimizar una función, ya que en cada iteración buscamos el máximo progreso hacia el objetivo sin necesidad de considerar lo que pueda ocurrir en los siguientes pasos. Sin embargo, su utilización práctica no da buenos resultados. Para demostrarlo podemos recurrir al análisis de convergencia de [Boy04, Cap. 9] donde se llega a la conclusión de que la diferencia entre el valor en un punto $\boldsymbol{x}^{(k)}$ y el óptimo p^* después de k iteraciones tiene la siguiente expresión

$$f(\boldsymbol{x}^{(k)}) - p^* \leq c^k \left(f(\boldsymbol{x}^{(0)}) - p^*\right)$$

Esto quiere decir que el error converge a cero en una serie geométrica caracterizada por la constante $c < 1$. Por tanto, se trata de una convergencia lineal, según la nomenclatura habitual en el contexto de métodos numéricos, debido a su representación en un eje semilogarítmico. La constante c que marca la velocidad de convergencia depende en gran medida de la matriz hessiana, $\nabla^2 f(\boldsymbol{x})$, que define la forma de las curvas de nivel. La convergencia puede ser muy lenta si la relación entre sus autovalores máximo y mínimo es muy alta.

Una vez definida la dirección de descenso, se asigna el tamaño del paso con uno de los métodos de §4.1.1.2 o §4.1.1.3, y repetimos el algoritmo 4.1 hasta que se cumpla el criterio de parada, que suele expresarse como $\|\nabla f(\boldsymbol{x})\|_2 \leq \eta$, siendo η un valor pequeño y positivo.

A pesar del inconveniente de la velocidad de convergencia, su utilización se justifica porque la principal ventaja de este método es su simplicidad.

4.1.1.2 Búsqueda lineal exacta

Este método consiste en elegir t que minimice f en la línea $\{\boldsymbol{x}+t\Delta\boldsymbol{x} \mid t \geq 0\}$

$$t = \operatorname*{argmin}_{s \geq 0} f(\boldsymbol{x} + s\Delta\boldsymbol{x})$$

Se utiliza esta solución cuando el coste del problema de minimización es bajo en comparación con el coste de calcular la dirección de búsqueda. En algunos casos se puede calcular analíticamente el minimizador sobre una línea.

4.1.1.3 Búsqueda lineal con retroceso

La mayoría de las búsquedas lineales en la práctica no son exactas. La longitud de paso se elige para minimizar aproximadamente f sobre la línea $\{\boldsymbol{x} + t\Delta\boldsymbol{x} \mid t \geq 0\}$ o incluso simplemente para reducir f suficientemente.

Un método inexacto de búsqueda lineal es el algoritmo 4.2, conocido como método con retroceso, que depende de dos constantes α y β tales que $0 < \alpha < 0{,}5$ y $0 < \beta < 1$. El nombre de este método se debe a que comienza con un paso unitario y lo reduce por un factor β hasta que llega a cumplir la siguiente condición

$$f(\boldsymbol{x} + t\Delta\boldsymbol{x}) \leq f(\boldsymbol{x}) + \alpha t \nabla f(\boldsymbol{x})^T \Delta\boldsymbol{x} \tag{4.1}$$

Como $\Delta\boldsymbol{x}$ es una dirección descendente, se cumple que $\nabla f(\boldsymbol{x})^T \Delta\boldsymbol{x} < 0$, por lo que si t es suficientemente pequeño podemos aproximar

$$f(\boldsymbol{x} + t\Delta\boldsymbol{x}) \approx f(\boldsymbol{x}) + t\nabla f(\boldsymbol{x})^T \Delta\boldsymbol{x} < f(\boldsymbol{x}) + \alpha t \nabla f(\boldsymbol{x})^T \Delta\boldsymbol{x}$$

Algoritmo 4.2: Búsqueda lineal con retroceso

Datos: $\alpha \in (0, 0{,}5)$, $\beta \in (0, 1)$, y una dirección de descenso $\Delta\boldsymbol{x}$ para f en $\boldsymbol{x} \in \mathbf{dom} f$

$t := 1$
mientras que $f(\boldsymbol{x} + t\Delta\boldsymbol{x}) > f(\boldsymbol{x}) + \alpha t \nabla f(\boldsymbol{x})^T \Delta\boldsymbol{x}$
  $t := \beta t$

La condición de retroceso se muestra en la figura 4.1. Si suponemos que el punto inicial $t = 1$ está a la derecha de t_0, al reducir el valor de t llegaremos a un punto inferior a t_0. Si por el contrario, al empezar con $t = 1$ ya estamos a la izquierda de t_0, $(t_0 \geq 1)$, entonces no hace falta cambiarlo. Vemos en esta figura que la desigualdad (4.1) se mantiene para $t \geq 0$ en un intervalo $(0, t_0]$. La búsqueda termina con una longitud de paso t que satisface

$$t = 1, \qquad \text{o} \qquad t \in (\beta t_0, t_0]$$

Podemos decir que la longitud de paso que obtenemos con el retroceso satisface lo siguiente

$$t \geq \text{mín}\{1, \beta t_0\}$$

Cuando el dominio de f no es todo $\mathbb{R}^n$, la condición (4.1) se tiene que interpretar con cuidado. Según requeríamos, f tiene que ser infinito fuera de su dominio. La desigualdad implica que

$$\boldsymbol{x} + \Delta\boldsymbol{x} \in \mathbf{dom}\, f$$

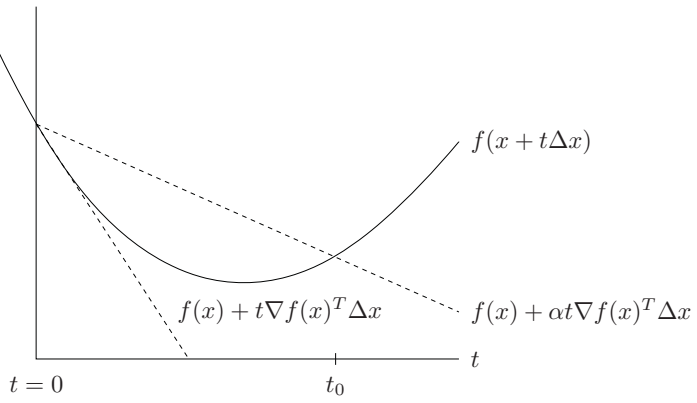

Figura 4.1: Búsqueda lineal con retroceso

En una implementación práctica, primero multiplicamos t por β hasta que $\boldsymbol{x} + t\Delta\boldsymbol{x}$ esté dentro del dominio de f, y entonces comenzamos a comprobar si se cumple dicha desigualdad (4.1).

El parámetro α se elige típicamente entre 0.01 y 0.3, por lo que aceptamos un decremento en f entre el 1 % y el 30 % de la predicción basada en la extrapolación lineal. El parámetro β se elige entre 0.1 y 0.8, donde el valor de 0.1 corresponde a una búsqueda muy fina, y 0.8 tendrá saltos de t más amplios.

4.1.2 Método de Newton

El método de Newton se caracteriza por utilizar la siguiente dirección de descenso

$$\Delta\boldsymbol{x}_{nt} = -\nabla^2 f(\boldsymbol{x})^{-1} \nabla f(\boldsymbol{x}) \tag{4.2}$$

Esta dirección se conoce como paso de Newton. Para justificarlo nos basamos en que $\nabla^2 f(\boldsymbol{x})$ es definida positiva, por lo que

$$\nabla f(\boldsymbol{x})^T \Delta\boldsymbol{x}_{nt} = -\nabla f(\boldsymbol{x})^T \nabla^2 f(\boldsymbol{x})^{-1} \nabla f(\boldsymbol{x}) < 0$$

Vemos que el paso de Newton es una dirección descendiente, salvo cuando $\nabla f(\boldsymbol{x}) = \boldsymbol{0}$ en el $\boldsymbol{x}$ óptimo.

Buscamos el mínimo de la aproximación de Taylor de segundo orden $\hat{f}$

$$\hat{f}(\boldsymbol{x} + \boldsymbol{v}) = f(\boldsymbol{x}) + \nabla f(\boldsymbol{x})^T \boldsymbol{v} + \frac{1}{2} \boldsymbol{v}^T \nabla^2 f(\boldsymbol{x}) \boldsymbol{v}$$

Se trata de una función convexa de $\boldsymbol{v}$ con mínimo en $\boldsymbol{v} = \Delta\boldsymbol{x}_{nt}$. El paso de Newton $\Delta\boldsymbol{x}_{nt}$ es el vector que hay que añadir al punto $\boldsymbol{x}$ para minimizar la aproximación de segundo orden de f en $\boldsymbol{x}$.

Si la función f es cuadrática, entonces $\boldsymbol{x}+\Delta\boldsymbol{x}_{nt}$ es el minimizador exacto de f. Si la función se parece a una cuadrática, entonces nuestra aproximación será muy buena. En los puntos cercanos a $\boldsymbol{x}$ la aproximación será muy precisa, por lo que a medida que nos acercamos al punto óptimo nuestra aproximación es mucho mejor. Esto se refleja en el comportamiento del algoritmo, que inicialmente puede tener una fase de convergencia lineal, y al final la convergencia se hace cuadrática.

4.1.2.1 Independencia afín

Una característica importante del paso de Newton es que permanece igual aunque hagamos un cambio de coordenadas. Si $\boldsymbol{T} \in \mathbb{R}^{n\times n}$ no es singular, y definimos $\bar{f}(\boldsymbol{y}) = f(\boldsymbol{T}\boldsymbol{y})$,

$$\nabla\bar{f}(\boldsymbol{y}) = \boldsymbol{T}^T\nabla f(\boldsymbol{x}), \qquad \nabla^2\bar{f}(\boldsymbol{y}) = \boldsymbol{T}^T\nabla^2 f(\boldsymbol{x})\boldsymbol{T}$$

Calculamos el paso de Newton para $\bar{f}$ en $\boldsymbol{y}$

$$\begin{aligned}\Delta\boldsymbol{y}_{nt} &= -\left(\boldsymbol{T}^T\nabla^2 f(\boldsymbol{x})\boldsymbol{T}\right)^{-1}\left(\boldsymbol{T}^T\nabla f(\boldsymbol{x})\right)\\ &= -\boldsymbol{T}^{-1}\nabla^2 f(\boldsymbol{x})^{-1}\nabla f(\boldsymbol{x})\\ &= \boldsymbol{T}^{-1}\Delta\boldsymbol{x}_{nt}\end{aligned}$$

Vemos que los pasos de Newton de f y $\bar{f}$ están relacionados por la misma transformación lineal, y

$$\boldsymbol{x} + \Delta\boldsymbol{x}_{nt} = \boldsymbol{T}\left(\boldsymbol{y} + \Delta\boldsymbol{y}_{nt}\right)$$

4.1.2.2 Decremento de Newton

Llamamos decremento de Newton a la siguiente cantidad

$$\lambda(\boldsymbol{x}) = \left(\nabla f(\boldsymbol{x})^T\nabla^2 f(\boldsymbol{x})^{-1}\nabla f(\boldsymbol{x})\right)^{1/2}$$

Nos servirá como criterio de parada. Podemos relacionar el decremento de Newton con la cantidad $f(\boldsymbol{x}) - \inf_{\boldsymbol{y}}\hat{f}(\boldsymbol{y})$, donde $\hat{f}$ es la aproximación de segundo orden de f en $\boldsymbol{x}$.

$$f(\boldsymbol{x}) - \inf_{\boldsymbol{y}}\hat{f}(\boldsymbol{y}) = f(\boldsymbol{x}) - \hat{f}(\boldsymbol{x}+\Delta\boldsymbol{x}_{nt}) = \frac{1}{2}\lambda(\boldsymbol{x})^2$$

Por este motivo, $\lambda^2/2$ es una estimación de $f(\boldsymbol{x}) - p^*$ basada en la aproximación cuadrática de f en $\boldsymbol{x}$. Podemos expresar el decremento de Newton como

$$\lambda(\boldsymbol{x}) = \left(\Delta\boldsymbol{x}_{nt}^T\nabla^2 f(\boldsymbol{x})\Delta\boldsymbol{x}_{nt}\right)^{1/2}$$

Esto muestra que λ es la norma cuadrática del paso de Newton definida por el hessiano

$$\|\boldsymbol{u}\|_{\nabla^2 f(\boldsymbol{x})} = \left(\boldsymbol{u}^T \nabla^2 f(\boldsymbol{x}) \boldsymbol{u}\right)^{1/2}$$

El decremento de Newton también aparece en la búsqueda lineal con retroceso, ya que tenemos

$$\nabla f(\boldsymbol{x})^T \Delta \boldsymbol{x}_{nt} = -\lambda(\boldsymbol{x})^2$$

Esta es la constante utilizada en la búsqueda lineal con retroceso, y se puede interpretar como la derivada direccional de f en $\boldsymbol{x}$ en la dirección del paso de Newton.

$$-\lambda(\boldsymbol{x})^2 = \nabla f(\boldsymbol{x})^T \Delta \boldsymbol{x}_{nt} = \frac{d}{dt} f(\boldsymbol{x} + \Delta \boldsymbol{x}_{nt} t)\Big|_{t=0}$$

En este caso, también vemos que el decremento de Newton es invariante afín, es decir, el decremento de $\bar{f}(\boldsymbol{y}) = f(\boldsymbol{T}\boldsymbol{y})$ en $\boldsymbol{y}$ es el mismo que el decremento de f en $\boldsymbol{x} = \boldsymbol{T}\boldsymbol{y}$.

4.1.2.3 Algoritmo de Newton

El algoritmo de Newton es semejante al método general de descenso, ya que básicamente se trata de repetir una secuencia de selecciones de dirección de descenso y tamaño de paso hasta llegar a un criterio de parada que nos indique que hemos encontrado el punto óptimo. El tamaño del paso también se hace por búsqueda lineal con retroceso, tal como vimos en §4.1.1.3, y la dirección de descenso se define con el paso de Newton definido en (4.2).

Comparando el algoritmo 4.3 con el 4.1 vemos que la principal diferencia es que el criterio de parada se calcula después de la dirección de búsqueda, y viene marcado por el decremento de Newton que hemos visto en §4.1.2.2.

Algoritmo 4.3: Método de Newton

Datos: un punto inicial $\boldsymbol{x} \in \mathbf{dom}\, f$, y una tolerancia $\epsilon > 0$

repetir

Calcular el paso de Newton y el decremento
$\Delta \boldsymbol{x}_{nt} := -\nabla^2 f(\boldsymbol{x})^{-1} \nabla f(\boldsymbol{x})$; $\lambda^2 := \nabla f(\boldsymbol{x})^T \nabla^2 f(\boldsymbol{x})^{-1} \nabla f(\boldsymbol{x})$
Criterio de parada. **Salir** si $\lambda^2/2 \leq \epsilon$
Búsqueda lineal. Elegir el tamaño de paso t por búsqueda lineal con retroceso.
Actualizar. $\boldsymbol{x} := \boldsymbol{x} + t\Delta \boldsymbol{x}_{nt}$

4.2 Algoritmos para minimizar con restricciones de igualdad

Añadimos ahora restricciones de igualdad a nuestro problema

$$\begin{array}{ll} \underset{\boldsymbol{x}}{\text{minimizar}} & f(\boldsymbol{x}) \\ \text{sujeto a} & \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b} \end{array} \tag{4.3}$$

La función f es convexa y dos veces derivable. $\boldsymbol{A} \in \mathbb{R}^{p\times n}$ es una matriz con menos igualdades que variables, $p < n$. Para que un punto $\boldsymbol{x}^* \in \mathbf{dom}\, f$ sea óptimo tiene que existir un $\boldsymbol{\nu}^*$ que cumpla las ecuaciones KKT

$$\boldsymbol{A}\boldsymbol{x}^* = \boldsymbol{b}, \qquad \nabla f(\boldsymbol{x}^*) + \boldsymbol{A}^T\boldsymbol{\nu}^* = \boldsymbol{0}$$

Tenemos $n+p$ ecuaciones con $n+p$ variables $\boldsymbol{x}^*$ y $\boldsymbol{\nu}^*$. Las primeras son las condiciones primales de viabilidad, y el segundo grupo son las condiciones duales de viabilidad. Normalmente no serán lineales. Únicamente en algunos casos podremos resolver este sistema de forma analítica. Lo habitual es tener que utilizar un algoritmo iterativo.

Aunque existen varias alternativas para buscar un problema equivalente que no tenga restricciones, en nuestro caso utilizaremos extensiones del método de Newton para que tenga en cuenta las igualdades añadidas.

4.2.1 Minimización cuadrática con restricciones de igualdad

En primer lugar vemos este caso que nos servirá de base para la extensión del método de Newton. El problema cuadrático es

$$\begin{array}{ll} \underset{\boldsymbol{x}}{\text{minimizar}} & (1/2)\boldsymbol{x}^T\boldsymbol{P}\boldsymbol{x} + \boldsymbol{q}^T\boldsymbol{x} + r \\ \text{sujeto a} & \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b} \end{array}$$

donde $\boldsymbol{P} \in \mathbb{S}^n_+$ y $\boldsymbol{A} \in \mathbb{R}^{p\times n}$. Las condiciones para los puntos óptimos son

$$\boldsymbol{A}\boldsymbol{x}^* = \boldsymbol{b}, \qquad \boldsymbol{P}\boldsymbol{x}^* + \boldsymbol{q} + \boldsymbol{A}^T\boldsymbol{\nu}^* = \boldsymbol{0}$$

Podemos escribirlo de forma matricial

$$\begin{bmatrix} \boldsymbol{P} & \boldsymbol{A}^T \\ \boldsymbol{A} & \boldsymbol{0} \end{bmatrix} \begin{bmatrix} \boldsymbol{x}^* \\ \boldsymbol{\nu}^* \end{bmatrix} = \begin{bmatrix} -\boldsymbol{q} \\ \boldsymbol{b} \end{bmatrix}$$

La matriz de coeficientes se llama matriz KKT. Cuando es una matriz no singular, existe un único par óptimo primal-dual $(\boldsymbol{x}^*, \boldsymbol{\nu}^*)$. Si la matriz KKT es singular pero el sistema KKT se puede resolver, cualquier solución dará un par óptimo $(\boldsymbol{x}^*, \boldsymbol{\nu}^*)$. Si no se puede resolver el sistema KKT, el problema de optimización no tiene límite inferior o no es viable.

4.2.2 Método de Newton con restricciones de igualdad

En este caso, partimos de un punto $\boldsymbol{x} \in \mathbf{dom}\, f$ viable, es decir, que satisface $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$. Nos tenemos que asegurar de que el paso de Newton $\Delta\boldsymbol{x}_{nt}$ es una dirección viable, es decir, $\boldsymbol{A}\Delta\boldsymbol{x}_{nt} = \boldsymbol{0}$ para que $\boldsymbol{A}(\boldsymbol{x} + \Delta\boldsymbol{x}_{nt}) = \boldsymbol{b}$.

Para obtener el paso de Newton $\Delta\boldsymbol{x}_{nt}$ de nuestro problema con restricciones de igualdad (4.3), sustituimos la función objetivo por su aproximación de Taylor de segundo orden cerca de $\boldsymbol{x}$

$$\begin{array}{ll} \underset{\boldsymbol{v}}{\text{minimizar}} & \hat{f}(\boldsymbol{x}+\boldsymbol{v}) = f(\boldsymbol{x}) + \nabla f(\boldsymbol{x})^T\boldsymbol{v} + (1/2)\boldsymbol{v}^T\nabla^2 f(\boldsymbol{x})\boldsymbol{v} \\ \text{sujeto a} & \boldsymbol{A}(\boldsymbol{x}+\boldsymbol{v}) = \boldsymbol{b} \end{array} \tag{4.4}$$

donde la variable es $\boldsymbol{v}$. Esto es un problema cuadrático de minimización con restricciones de igualdad, y se puede resolver analíticamente. Definimos el paso de Newton en $\boldsymbol{x}$ como la solución del problema (4.4), suponiendo que la matriz KKT no es singular. El paso de Newton $\Delta\boldsymbol{x}_{nt}$ es lo que hay que añadir a $\boldsymbol{x}$ para resolver el problema cuadrático. Sustituimos $\boldsymbol{v}$ por $\Delta\boldsymbol{x}_{nt}$ y las condiciones KKT quedan

$$\begin{aligned} &\boldsymbol{A}(\boldsymbol{x} + \Delta\boldsymbol{x}_{nt}) = \boldsymbol{b}, \\ &\nabla f(\boldsymbol{x} + \Delta\boldsymbol{x}_{nt}) + \boldsymbol{A}^T\boldsymbol{w} \approx \nabla f(\boldsymbol{x}) + \nabla^2 f(\boldsymbol{x})\Delta\boldsymbol{x}_{nt} + \boldsymbol{A}^T\boldsymbol{w} = \boldsymbol{0} \end{aligned}$$

Como $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ tenemos

$$\begin{aligned} &\boldsymbol{A}(\Delta\boldsymbol{x}_{nt}) = \boldsymbol{0}, \\ &\nabla^2 f(\boldsymbol{x})\Delta\boldsymbol{x}_{nt} + \boldsymbol{A}^T\boldsymbol{w} = -\nabla f(\boldsymbol{x}) \end{aligned}$$

En forma matricial es el sistema KKT de ecuaciones lineales

$$\begin{bmatrix} \nabla^2 f(\boldsymbol{x}) & \boldsymbol{A}^T \\ \boldsymbol{A} & \boldsymbol{0} \end{bmatrix} \begin{bmatrix} \Delta\boldsymbol{x}_{nt} \\ \boldsymbol{w} \end{bmatrix} = \begin{bmatrix} -\nabla f(\boldsymbol{x}) \\ \boldsymbol{0} \end{bmatrix}$$

A partir de aquí existen métodos de álgebra lineal para resolver este problema de forma muy eficiente: aprovechando la estructura de la matriz KKT es conveniente utilizar el complemento de Schur y aplicar la factorización de Cholesky. Para profundizar sobre estos recursos podemos revisar la bibliografía [Tre97, Zha05].

Cuando la función objetivo es cuadrática, la actualización de Newton $\boldsymbol{x} + \Delta\boldsymbol{x}_{nt}$ resuelve exactamente el problema original, y en este caso el vector $\boldsymbol{w}$ es la variable dual óptima. Cuando la función objetivo parece cuadrática tendremos una buena aproximación.

El decremento de Newton será igual que cuando no hay restricciones.

$$\lambda(\boldsymbol{x}) = \left(\Delta\boldsymbol{x}_{nt}^T\nabla^2 f(\boldsymbol{x})\Delta\boldsymbol{x}_{nt}\right)^{1/2}$$

Por último, resumimos el algoritmo 4.4, que se trata de un método viable de descenso, ya que todas las iteraciones son viables, con $f(\boldsymbol{x}^{(k+1)}) < f(\boldsymbol{x}^{(k)})$ salvo cuando $\boldsymbol{x}^{(k)}$ es óptimo.

Es necesario que la matriz KKT se pueda invertir en todo $\boldsymbol{x}$.

Algoritmo 4.4: Método de Newton con restricciones de igualdad

Datos: un punto inicial $\boldsymbol{x} \in \mathbf{dom}\, f$ viable $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$, y una tolerancia $\epsilon > 0$

repetir

Calcular el paso de Newton y el decremento $\Delta\boldsymbol{x}_{nt}$; $\lambda(\boldsymbol{x})$
Criterio de parada. **Salir** si $\lambda^2/2 \leq \epsilon$
Búsqueda lineal. Elegir el tamaño de paso t por búsqueda lineal con retroceso.
Actualizar. $\boldsymbol{x} := \boldsymbol{x} + t\Delta\boldsymbol{x}_{nt}$

4.2.3 Método de Newton desde un punto inicial no viable

Aunque empecemos por un punto $\boldsymbol{x} \in \mathbf{dom}\, f$ no viable, queremos encontrar un paso $\Delta\boldsymbol{x}$ de forma que $\boldsymbol{x} + \Delta\boldsymbol{x}$ cumpla las condiciones de punto óptimo aunque sea aproximadamente, $\boldsymbol{x} + \Delta\boldsymbol{x} \approx \boldsymbol{x}^*$

$$\boldsymbol{A}\boldsymbol{x}^* = \boldsymbol{b}, \qquad \nabla f(\boldsymbol{x}^*) + \boldsymbol{A}^T\boldsymbol{\nu}^* = \boldsymbol{0}$$

Utilizaremos la aproximación de primer orden para el gradiente

$$\nabla f(\boldsymbol{x} + \Delta\boldsymbol{x}) \approx \nabla f(\boldsymbol{x}) + \nabla^2 f(\boldsymbol{x})\Delta\boldsymbol{x}$$

Y obtenemos las condiciones

$$\boldsymbol{A}(\boldsymbol{x} + \Delta\boldsymbol{x}) = \boldsymbol{b}, \qquad \nabla f(\boldsymbol{x}) + \nabla^2 f(\boldsymbol{x})\Delta\boldsymbol{x} + \boldsymbol{A}^T\boldsymbol{w} = \boldsymbol{0}$$

Esto es un conjunto de ecuaciones lineales de $\Delta\boldsymbol{x}$ y $\boldsymbol{w}$ que expresamos en forma matricial

$$\begin{bmatrix} \nabla^2 f(\boldsymbol{x}) & \boldsymbol{A}^T \\ \boldsymbol{A} & \boldsymbol{0} \end{bmatrix} \begin{bmatrix} \Delta\boldsymbol{x} \\ \boldsymbol{w} \end{bmatrix} = - \begin{bmatrix} \nabla f(\boldsymbol{x}) \\ \boldsymbol{A}\boldsymbol{x} - \boldsymbol{b} \end{bmatrix} \tag{4.5}$$

En este sistema aparece $\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}$ que es el vector residual de las restricciones de igualdad. Cuando $\boldsymbol{x}$ es viable este residuo desaparece y tenemos el método estándar de Newton desde un punto viable.

4.2.3.1 Interpretación como método primal-dual

Un método *primal-dual* actualiza la variable primal $\boldsymbol{x}$ y la variable dual $\boldsymbol{\nu}$ hasta que satisfacen las condiciones para ser óptimos. Estas condiciones se expresan como $\boldsymbol{r}(\boldsymbol{x}^*, \boldsymbol{\nu}^*) = \boldsymbol{0}$, y definimos $\boldsymbol{r} : \mathbb{R}^n \times \mathbb{R}^p \to \mathbb{R}^n \times \mathbb{R}^p$ como

$$\boldsymbol{r}(\boldsymbol{x}, \boldsymbol{\nu}) = (\boldsymbol{r}_{dual}(\boldsymbol{x}, \boldsymbol{\nu}), \boldsymbol{r}_{pri}(\boldsymbol{x}, \boldsymbol{\nu}))$$

En esta ecuación aparecen dos términos llamados residuo dual y residuo primal que se definen como

$$\boldsymbol{r}_{dual}(\boldsymbol{x},\boldsymbol{\nu}) = \nabla f(\boldsymbol{x}) + \boldsymbol{A}^T\boldsymbol{\nu}, \qquad \boldsymbol{r}_{pri}(\boldsymbol{x},\boldsymbol{\nu}) = \boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}$$

Si agrupamos las variables $\boldsymbol{x}$ y $\boldsymbol{\nu}$ en $\boldsymbol{y} = (\boldsymbol{x},\boldsymbol{\nu})$, la aproximación de Taylor de primer orden de $\boldsymbol{r}$ en torno a nuestra estimación $\boldsymbol{y}$ es

$$\boldsymbol{r}(\boldsymbol{y}+\boldsymbol{z}) \approx \hat{\boldsymbol{r}}(\boldsymbol{y}+\boldsymbol{z}) = \boldsymbol{r}(\boldsymbol{y}) + D\boldsymbol{r}(\boldsymbol{y})\boldsymbol{z}$$

Vemos que aparece la derivada de $\boldsymbol{r}$ en $\boldsymbol{y}$, $D\boldsymbol{r}(\boldsymbol{y}) \in \mathbb{R}^{(n+p)\times(n+p)}$. Definimos el paso primal-dual de Newton $\Delta\boldsymbol{y}_{pd}$ como el paso $\boldsymbol{z}$ para el que la aproximación de Taylor $\hat{\boldsymbol{r}}(\boldsymbol{y}+\boldsymbol{z})$ se hace cero

$$D\boldsymbol{r}(\boldsymbol{y})\Delta\boldsymbol{y}_{pd} = -\boldsymbol{r}(\boldsymbol{y})$$

Hemos considerado las dos variables $\boldsymbol{x}$ y $\boldsymbol{\nu}$. $\Delta\boldsymbol{y}_{pd} = (\Delta\boldsymbol{x}_{pd}, \Delta\boldsymbol{\nu}_{pd})$ nos da un paso primal y un paso dual.

También podemos plantearlo de forma matricial

$$\begin{bmatrix} \nabla^2 f(\boldsymbol{x}) & \boldsymbol{A}^T \\ \boldsymbol{A} & \boldsymbol{0} \end{bmatrix} \begin{bmatrix} \Delta\boldsymbol{x}_{pd} \\ \Delta\boldsymbol{\nu}_{pd} \end{bmatrix} = - \begin{bmatrix} \boldsymbol{r}_{dual} \\ \boldsymbol{r}_{pri} \end{bmatrix} = - \begin{bmatrix} \nabla f(\boldsymbol{x}) + \boldsymbol{A}^T\boldsymbol{\nu} \\ \boldsymbol{A}\boldsymbol{x} - \boldsymbol{b} \end{bmatrix}$$

Si escribimos $\boldsymbol{\nu} + \Delta\boldsymbol{\nu}_{pd}$ como $\boldsymbol{\nu}^+$ obtenemos el mismo sistema de ecuaciones que en (4.5), y además vemos que no necesitamos conocer el valor actual de la variable dual.

$$\begin{bmatrix} \nabla^2 f(\boldsymbol{x}) & \boldsymbol{A}^T \\ \boldsymbol{A} & \boldsymbol{0} \end{bmatrix} \begin{bmatrix} \Delta\boldsymbol{x}_{pd} \\ \boldsymbol{\nu}^+ \end{bmatrix} = - \begin{bmatrix} \nabla f(\boldsymbol{x}) \\ \boldsymbol{A}\boldsymbol{x} - \boldsymbol{b} \end{bmatrix}$$

Vemos que el paso de Newton con desde un punto no viable es igual que el paso primal-dual, y el vector dual asociado $\boldsymbol{w}$ es la variable primal-dual actualizada $\boldsymbol{\nu}^+ = \boldsymbol{\nu} + \Delta\boldsymbol{\nu}_{pd}$.

4.2.3.2 Reducción de la norma residual

La dirección de Newton en un punto no viable no tiene por qué ser necesariamente una dirección descendente para f

$$\begin{aligned} \frac{d}{dt} f(\boldsymbol{x} + t\Delta\boldsymbol{x})\Big|_{t=0} &= \nabla f(\boldsymbol{x})^T \Delta\boldsymbol{x} \\ &= -\Delta\boldsymbol{x}^T \left(\nabla^2 f(\boldsymbol{x})\Delta\boldsymbol{x} + \boldsymbol{A}^T\boldsymbol{w} \right) \\ &= -\Delta\boldsymbol{x}^T \nabla^2 f(\boldsymbol{x})\Delta\boldsymbol{x} + (\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b})^T \boldsymbol{w} \end{aligned}$$

De hecho, no es necesario, salvo que $\boldsymbol{x}$ sea viable. Ahora vemos que la interpretación primal-dual nos sirve para ver que lo que disminuye en la dirección de Newton es la norma del residuo

$$\frac{d}{dt}\left\|\boldsymbol{r}\left(\boldsymbol{y}+t\Delta\boldsymbol{y}_{pd}\right)\right\|_2^2\Bigg|_{t=0} = 2\boldsymbol{r}(\boldsymbol{y})^T D\boldsymbol{r}(\boldsymbol{y})\Delta\boldsymbol{y}_{pd} = -2\boldsymbol{r}(\boldsymbol{y})^T\boldsymbol{r}(\boldsymbol{y})$$

Esto nos permite utilizar $\|\boldsymbol{r}\|_2$ para medir el progreso del algoritmo, por ejemplo en la búsqueda lineal.

4.2.3.3 Viabilidad con el paso completo

Si volvemos a comprobar el sistema de ecuaciones (4.5) para calcular el paso de Newton desde un punto no viable, tenemos que

$$\boldsymbol{A}(\boldsymbol{x}+\Delta\boldsymbol{x}_{nt}) = \boldsymbol{b}$$

Si tomamos una longitud de paso igual a uno, entonces la siguiente iteración será viable, y el paso de Newton será una dirección viable, por lo que las siguientes iteraciones también lo serán.

Intentaremos ver el efecto que tiene amortiguar el paso desde un punto no viable. Si $t \in (0,1)$ la siguiente iteración será $\boldsymbol{x}^+ = \boldsymbol{x} + t\Delta\boldsymbol{x}_{nt}$, por lo que el residuo primal en la siguiente iteración será

$$\boldsymbol{r}_{pri}^+ = \boldsymbol{A}(\boldsymbol{x}+\Delta\boldsymbol{x}_{nt}t) - \boldsymbol{b} = (1-t)(\boldsymbol{A}\boldsymbol{x}-\boldsymbol{b}) = (1-t)\boldsymbol{r}_{pri}$$

Vemos que el residuo se reduce por un factor $(1-t)$. Esta secuencia nos permite relacionar cada residuo en función de la primera iteración

$$\boldsymbol{r}_{pri}^{(k)} = \left(\prod_{i=0}^{k-1}\left(1-t^{(i)}\right)\right)\boldsymbol{r}_{pri}^{(0)}$$

Esto quiere decir que conservarán la misma dirección que en el primer paso, y si en alguna iteración $t^{(i)} = 1$ el producto se hace cero y las siguientes iteraciones serán viables.

4.2.3.4 Algoritmo de Newton desde un punto no viable

En el algoritmo 4.5 describimos la extensión del método de Newton cuando el punto inicial $\boldsymbol{x}^{(0)} \in \mathbf{dom}\, f$ no es viable. Encontramos algunas diferencias con respecto al método viable. Las direcciones de búsqueda incluyen nuevos términos de corrección que dependen del residuo primal. La búsqueda lineal se hace con la norma del residuo. El algoritmo termina cuando llegamos a un $\boldsymbol{x}$ viable y la norma del residuo dual es pequeña.

La utilización de la norma del residuo para la búsqueda lineal supone un pequeño aumento del coste computacional, comparándolo con la búsqueda lineal basada en el valor de la función, pero este aumento normalmente no se nota.

Algoritmo 4.5: Método de Newton desde un punto inicial no viable

Datos: un punto inicial $(\boldsymbol{x}, \boldsymbol{\nu})$, $\alpha \in (0, 1/2)$, $\beta \in (0, 1)$, y tolerancia $\epsilon > 0$

repetir

Calcular los pasos primal y dual de Newton, $\Delta\boldsymbol{x}_{nt}$, $\Delta\boldsymbol{\nu}_{nt}$

Búsqueda lineal en $\|\boldsymbol{r}\|_2$

t:=1

mientras que $\|\boldsymbol{r}(\boldsymbol{x}+t\Delta\boldsymbol{x}_{nt}, \boldsymbol{\nu}+t\Delta\boldsymbol{\nu}_{nt})\|_2 > (1+\alpha t)\|\boldsymbol{r}(\boldsymbol{x}, \boldsymbol{\nu})\|_2$

$t := \beta t$

Actualizar. $\boldsymbol{x} := \boldsymbol{x} + t\Delta\boldsymbol{x}_{nt}$, $\boldsymbol{\nu} := \boldsymbol{\nu} + t\Delta\boldsymbol{\nu}_{nt}$

hasta que $\boldsymbol{Ax} = \boldsymbol{b}$, *y* $\|\boldsymbol{r}(\boldsymbol{x}, \boldsymbol{\nu})\|_2 \leq \epsilon$

4.2.3.5 Inicialización simplificada

La principal ventaja de comenzar en un punto no viable es la inicialización. Si el dominio de f es todo $\mathbb{R}^n$, entonces no ganamos mucho por empezar en un punto no viable, porque únicamente hay que buscar una solución a $\boldsymbol{Ax} = \boldsymbol{b}$.

Cuando no disponemos de un dominio tan generalizado, puede resultar complicado encontrar un punto que cumpla la restricción, y ni siquiera sabemos si **dom** f tiene intersección con el conjunto $\{\boldsymbol{z} \mid \boldsymbol{Az} = \boldsymbol{b}\}$. En este caso podemos utilizar un método de fase I para calcular ese punto, como veremos en §4.3.2. Cuando el dominio de f es sencillo y sabemos que contiene algún punto viable, entonces es útil utilizar el método desde un punto no viable.

4.3 Algoritmos para minimizar con restricciones en desigualdades: Métodos de punto interior

Pasamos ahora a resolver problemas que incluyen desigualdades

$$\begin{array}{lll} \underset{\boldsymbol{x}}{\text{minimizar}} & f_0(\boldsymbol{x}) & \\ \text{sujeto a} & f_i(\boldsymbol{x}) \leq 0, & i = 1, \ldots, m \\ & \boldsymbol{Ax} = \boldsymbol{b}, & \boldsymbol{A} \in \mathbb{R}^{p \times n} \end{array} \tag{4.6}$$

Los algoritmos de punto interior deben su nombre a que resuelven el problema (4.6) pasando por puntos de un camino central dentro del conjunto viable. En realidad, los métodos modernos de punto interior no funcionan exactamente de esta manera, pero han conservado el nombre porque su origen está en el método de barrera que veremos en §4.3.1.

Suponemos que se cumple la condición de Slater, explicada en §2.4.2.2. Recordemos de que si se cumple esta condición y el problema es convexo, entonces el problema dual y el primal tienen la misma solución (dualidad fuerte). Esto implica que existe un óptimo dual $\boldsymbol{\lambda}^* \in \mathbb{R}^m$, $\boldsymbol{\nu}^* \in \mathbb{R}^p$, y se tienen que cumplir las condiciones KKT

$$\begin{aligned} f_i(\boldsymbol{x}^*) &\leq 0, \qquad i = 1, \ldots, m \\ \boldsymbol{A}\boldsymbol{x}^* &= \boldsymbol{b} \\ \boldsymbol{\lambda}^* &\succeq \boldsymbol{0} \\ \lambda_i^* f_i(\boldsymbol{x}^*) &= 0, \qquad i = 1, \ldots, m \\ \nabla f_0(\boldsymbol{x}^*) + \sum_{i=1}^{m} \lambda_i^* \nabla f_i(\boldsymbol{x}^*) + \boldsymbol{A}^T \boldsymbol{\nu}^* &= \boldsymbol{0} \end{aligned} \tag{4.7}$$

Los métodos de punto interior resuelven este problema transformándolo en una secuencia de problemas con restricciones de igualdad. En cada paso, se aplica el método de Newton convirtiendo el problema en una secuencia de sistemas de ecuaciones lineales.

4.3.1 Método de barrera

Se trata de un tipo de algoritmos de punto interior. Intentaremos incluir de forma aproximada las desigualdades para que el problema tenga únicamente restricciones de igualdad. Empezamos utilizando la función indicador $I_- : \mathbb{R} \to \mathbb{R}$ que se define como

$$I_-(u) = \begin{cases} 0, & \text{si } u \leq 0 \\ \infty, & \text{si } u > 0 \end{cases}$$

Con esta función podemos construir un problema equivalente que incluye las desigualdades en el objetivo

$$\begin{array}{ll} \underset{\boldsymbol{x}}{\text{minimizar}} & f_0(\boldsymbol{x}) + \sum_{i=1}^{m} I_- \left(f_i(\boldsymbol{x})\right) \\ \text{sujeto a} & \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b} \end{array} \tag{4.8}$$

El problema es que la función que nos queda no es diferenciable y no podemos aplicar el método de Newton. Para solucionarlo aproximamos la función indicador I_- por lo siguiente

$$\hat{I}_-(u) = -(1/t)\log(-u), \qquad \mathbf{dom}\, \hat{I}_- = -\mathbb{R}_{++}$$

El parámetro $t > 0$ marca la precisión de la aproximación. La función $\hat{I}_-$ es convexa y no decreciente, por lo que $\hat{I}_- \left(f_i(\boldsymbol{x})\right)$ conservará la convexidad de $f_i(\boldsymbol{x})$. Como venimos haciendo hasta ahora, cuando $u > 0$ valdrá ∞. Con esto ya tenemos una función diferenciable.

$$\begin{array}{ll} \underset{\boldsymbol{x}}{\text{minimizar}} & f_0(\boldsymbol{x}) + \sum_{i=1}^{m} -(1/t)\log\left(f_i(\boldsymbol{x})\right) \\ \text{sujeto a} & \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b} \end{array} \tag{4.9}$$

Se conoce como función barrera logarítmica a la siguiente función

$$\phi(\boldsymbol{x}) = -\sum_{i=1}^{m} \log(-f_i(\boldsymbol{x}))$$

Su dominio es el conjunto de puntos que satisfacen las desigualdades de forma estricta, $f_i(\boldsymbol{x}) < 0$.

El nuevo problema (4.9) es una aproximación del problema original (4.8), y la diferencia entre sus soluciones depende del parámetro t.

Nos resultará útil conocer el gradiente y el hessiano de la función de barrera logarítmica ϕ

$$\nabla\phi(\boldsymbol{x}) = \sum_{i=1}^{m} \frac{1}{-f_i(\boldsymbol{x})}\nabla f_i(\boldsymbol{x})$$
$$\nabla^2\phi(\boldsymbol{x}) = \sum_{i=1}^{m} \frac{1}{f_i(\boldsymbol{x})^2}\nabla f_i(\boldsymbol{x})\nabla f_i(\boldsymbol{x})^T + \sum_{i=1}^{m} \frac{1}{-f_i(\boldsymbol{x})}\nabla^2 f_i(\boldsymbol{x})$$

4.3.1.1 Camino central

Para simplificar presentamos un problema equivalente a (4.9)

$$\begin{array}{ll} \underset{\boldsymbol{x}}{\text{minimizar}} & tf_0(\boldsymbol{x}) + \phi(\boldsymbol{x}) \\ \text{sujeto a} & \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b} \end{array} \tag{4.10}$$

Suponemos que podemos resolverlo por Newton y que tiene una solución única para cada $t > 0$. En este caso, definimos $\boldsymbol{x}^*(t)$ como la solución de (4.10), y el camino central asociado al problema (4.10) será el conjunto de puntos $\boldsymbol{x}^*(t)$ tales que $t > 0$ que conocemos como puntos centrales. Estos puntos serán estrictamente viables, es decir, $f_i(\boldsymbol{x}^*(t)) < 0$, y $\boldsymbol{A}\boldsymbol{x}^*(t) = \boldsymbol{b}$, y además, existe un $\hat{\boldsymbol{\nu}} \in \mathbb{R}^p$ para el que se cumple lo siguiente

$$\begin{aligned} \boldsymbol{0} &= t\nabla f_0(\boldsymbol{x}^*(t)) + \nabla\phi(\boldsymbol{x}^*(t)) + \boldsymbol{A}^T\hat{\boldsymbol{\nu}} \\ &= t\nabla f_0(\boldsymbol{x}^*(t)) + \sum_{i=1}^{m} \frac{1}{-f_i(\boldsymbol{x})}\nabla f_i(\boldsymbol{x}) + \boldsymbol{A}^T\hat{\boldsymbol{\nu}} \end{aligned} \tag{4.11}$$

Cada punto del camino central da un punto dual viable que será una cota inferior para el valor óptimo p^*. Además, definimos los siguientes puntos

$$\begin{aligned} \lambda_i^*(t) &= -\frac{1}{tf_i(\boldsymbol{x}^*(t))}, \qquad i = 1, \dots, m \\ \boldsymbol{\nu}^*(t) &= \hat{\boldsymbol{\nu}}/t \end{aligned} \tag{4.12}$$

con lo que la condición de los puntos óptimos (4.11) queda así

$$\nabla f_0(\boldsymbol{x}^*(t)) + \sum_{i=1}^{m} \lambda_i^*(t)\nabla f_i(\boldsymbol{x}) + \boldsymbol{A}^T\boldsymbol{\nu}^*(t) = \boldsymbol{0}$$

Este par de puntos $\boldsymbol{\lambda}^*(t)$ y $\boldsymbol{\nu}^*(t)$ es viable en el problema dual, y $\boldsymbol{x}^*(t)$ minimiza el lagrangiano

$$L(\boldsymbol{x}, \boldsymbol{\lambda}^*(t), \boldsymbol{\nu}^*(t)) = f_0(\boldsymbol{x}) + \sum_{i=1}^{m} \lambda_i^*(t) f_i(\boldsymbol{x}) + \boldsymbol{\nu}^*(t)^T(\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b})$$

por tanto, la función dual $g(\boldsymbol{\lambda}^*(t), \boldsymbol{\nu}^*(t))$ será finita

$$\begin{aligned} g(\boldsymbol{\lambda}^*(t), \boldsymbol{\nu}^*(t)) &= f_0(\boldsymbol{x}^*(t)) + \sum_{i=1}^{m} \lambda_i^*(t) f_i(\boldsymbol{x}^*(t)) + \boldsymbol{\nu}^*(t)^T(\boldsymbol{A}\boldsymbol{x}^*(t) - \boldsymbol{b}) \\ &= f_0(\boldsymbol{x}^*(t)) - m/t \end{aligned}$$

La distancia de dualidad asociada con $\boldsymbol{x}^*(t)$ es m/t, por lo que vemos que $\boldsymbol{x}^*(t)$ converge hacia el punto óptimo cuando $t \to \infty$

$$f_0(\boldsymbol{x}^*(t)) - p^* \leq m/t$$

4.3.1.2 Condiciones KKT

La condición (4.11) se puede ver como una deformación de las condiciones KKT. Un punto $\boldsymbol{x}$ es igual a $\boldsymbol{x}^*(t)$ si y sólo si existe $\boldsymbol{\lambda}, \boldsymbol{\nu}$ tal que

$$\begin{aligned} f_i(\boldsymbol{x}) &\leq 0, \qquad i = 1, \ldots, m \\ \boldsymbol{A}\boldsymbol{x} &= \boldsymbol{b} \\ \boldsymbol{\lambda} &\succeq \boldsymbol{0} \\ -\lambda_i f_i(\boldsymbol{x}) &= 1/t, \qquad i = 1, \ldots, m \\ \nabla f_0(\boldsymbol{x}) + \sum_{i=1}^{m} \lambda_i \nabla f_i(\boldsymbol{x}) + \boldsymbol{A}^T\boldsymbol{\nu} &= \boldsymbol{0} \end{aligned} \tag{4.13}$$

La única diferencia entre (4.7) y (4.13) es la condición de holgura complementaria $-\lambda_i f_i(\boldsymbol{x}) = 1/t$, por lo que $(t, \boldsymbol{x}^*(t))$ y su punto dual asociado $(\boldsymbol{\lambda}^*(t), \boldsymbol{\nu}^*(t))$ se acercarán a cumplir las condiciones KKT cuando t sea elevado.

4.3.1.3 Algoritmo de barrera

Hemos visto que el punto $\boldsymbol{x}^*(t)$ es m/t-subóptimo, y el par $(\boldsymbol{\lambda}^*(t), \boldsymbol{\nu}^*(t))$ constituye un certificado de esta exactitud. Si queremos resolver el problema (4.6) con una exactitud ϵ podemos utilizar $t \geq m/\epsilon$ y aplicar Newton, pero cuando el problema es grande y necesitamos mucha precisión esto no funcionará bien. En lugar de esto, llegaremos a $t \geq m/\epsilon$ aumentando el valor de t en una secuencia de problemas de minimización con restricciones de igualdad. En esto consiste el algoritmo 4.6.

Algoritmo 4.6: Método de barrera

Datos: un punto estrictamente viable $\boldsymbol{x}$, un parámetro inicial $t^{(0)}$, $\mu > 1$, y $\epsilon > 0$

repetir

Paso de centrado
Calcular $\boldsymbol{x}^*(t)$ minimizando $tf_0 + \phi$ sujeto a $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$, comenzando en $\boldsymbol{x}$.
Actualizar. $\boldsymbol{x} := \boldsymbol{x}^*(t)$
Criterio de parada. **salir** si $m/t < \epsilon$
$t := \mu t$

Cada minimización con restricciones de igualdad en la que se calcula $\boldsymbol{x}^*(t^{(i)})$ se conoce como un paso de centrado o iteración externa, y las iteraciones del método de Newton que se ejecutan en cada paso de centrado se llaman iteraciones internas.

No tiene sentido buscar mucha precisión en la secuencia de puntos $\boldsymbol{x}^*(t)$ y sus duales $(\boldsymbol{\lambda}^*(t), \boldsymbol{\nu}^*(t))$ hasta que lleguemos al punto óptimo. Esto se puede hacer añadiendo un término de corrección a las expresiones de (4.12).

Por otro lado, la elección del parámetro μ supone un compromiso entre la cantidad de iteraciones internas y externas que se necesiten. Si μ es pequeño, cercano a 1, entonces los puntos $\boldsymbol{x}^*(t)$ del camino central serán buenas inicializaciones para el método de Newton, por lo que tendremos pocas repeticiones internas por cada iteración externa, pero habrá muchas iteraciones externas. Si μ es grande ocurre lo contrario, con pocas iteraciones externas y muchos ciclos de Newton para cada una. En la práctica se utilizan valores de μ entre 3 y 100 más o menos, aunque tampoco es un parámetro especialmente crítico, por lo que si utilizamos valores en torno a 10 o 20 funcionará bien.

El valor inicial de $t^{(0)}$ marcará el comportamiento de las primeras repeticiones externas. Si es demasiado grande se necesitan muchos ciclos de Newton en la primera iteración externa, pero si es muy pequeño se necesitarán más iteraciones externas. Una elección razonable es hacer que $m/t^{(0)}$ sea del orden de $f_0(\boldsymbol{x}^{(0)}) - p^*$ si es que tenemos una idea de lo lejos que estamos del óptimo. Podemos considerar una medida de la desviación de $\boldsymbol{x}^{(0)}$ respecto al punto $\boldsymbol{x}^*(t)$ como

$$\inf_{\boldsymbol{\nu}} \left\| t\nabla f_0(\boldsymbol{x}^{(0)}) + \nabla\phi(\boldsymbol{x}^{(0)}) + \boldsymbol{A}^T\boldsymbol{\nu} \right\|_2 \tag{4.14}$$

y elegimos $\boldsymbol{x}^{(0)}$ como el valor que minimiza (4.14).

4.3.2 Método de fase I

El método de barrera necesita un punto inicial $\boldsymbol{x}^{(0)}$ estrictamente viable. Cuando no conocemos ese punto, el método de barrera comienza calculándolo en una etapa previa llamada fase I.

En primer lugar planteamos lo que se conoce como problema de optimización de fase I:

$$\begin{array}{ll} \underset{\boldsymbol{x},s}{\text{minimizar}} & s \\ \text{sujeto a} & f_i(\boldsymbol{x}) \leq s, \qquad i = 1, \ldots, m \\ & \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b} \end{array} \tag{4.15}$$

La variable s se puede ver como un límite superior para puntos no viables en las desigualdades. El objetivo es que el máximo punto no factible esté por debajo de cero.

Este problema siempre será estrictamente viable, ya que podemos elegir cualquier punto inicial $\boldsymbol{x}^{(0)}$ junto con un s que sea mayor que $\max_i f_i(\boldsymbol{x}^{(0)})$. Por tanto, podemos utilizar el método de barrera para resolver el problema (4.15), y con el valor óptimo obtenido $\bar{p}^*$ distinguimos tres casos posibles:

- Si $\bar{p}^* < 0$ el problema tiene solución estrictamente viable. Además, no necesitamos resolver (4.15) con mucha precisión, ya que podemos parar cuando encontremos un $(\boldsymbol{x}, s)$ con $s < 0$.

- Si $\bar{p}^* > 0$ el problema no es viable. Tampoco necesitamos mucha precisión, y podemos terminar cuando encontremos un punto dual viable con objetivo dual positivo.

- Si $\bar{p}^* = 0$ el problema es viable, pero no estrictamente. También puede ocurrir que $\bar{p}^* = 0$ pero que no se pueda alcanzar el mínimo, con lo que las desigualdades serían inviables. En la práctica no podemos determinar que $\bar{p}^* = 0$, sino que $|\bar{p}^*| < \epsilon$, por lo que comprobaremos que las desigualdades con $f_i(\boldsymbol{x}) \leq -\epsilon$ no son viables, y las $f_i(\boldsymbol{x}) \leq \epsilon$ sí lo son.

4.3.2.1 Fase I por el método de Newton desde un punto inicial no viable

Podemos aplicar el método de Newton para el problema de la fase I utilizando una versión equivalente del problema original:

$$\begin{array}{ll} \underset{\boldsymbol{x},s}{\text{minimizar}} & f_0(\boldsymbol{x}) \\ \text{sujeto a} & f_i(\boldsymbol{x}) \leq s, \qquad i = 1, \ldots, m \\ & \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b} \\ & s = 0 \end{array}$$

En este caso, la fase I consistirá en resolver el siguiente problema por el método de Newton desde un punto inicial no factible

$$\begin{array}{ll} \underset{\boldsymbol{x},s}{\text{minimizar}} & t^{(0)} f_0(\boldsymbol{x}) - \sum_{i=1}^m \log(s - f_i(\boldsymbol{x})) \\ \text{sujeto a} & \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b} \\ & s = 0 \end{array}$$

Podemos inicializar desde cualquier $(\boldsymbol{x}, s)$ con $s > \max_i f_i(\boldsymbol{x})$. Al ser un problema estrictamente viable, el método de Newton terminará por tomar un paso no amortiguado, y a continuación tendremos $s = 0$.

4.3.3 Problemas con desigualdades generalizadas

Ahora extendemos el método de barrera para problemas con desigualdades generalizadas

$$\begin{array}{lll} \underset{\boldsymbol{x}}{\text{minimizar}} & f_0(\boldsymbol{x}) & \\ \text{sujeto a} & \boldsymbol{f}_i(\boldsymbol{x}) \preceq_{\mathcal{K}_i} \boldsymbol{0}, & i = 1, \ldots, m \\ & \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b} & \boldsymbol{A} \in \mathbb{R}^{p \times n} \end{array} \tag{4.16}$$

Suponemos que las funciones $\boldsymbol{f}_i$ son dos veces diferenciables, y que el problema tiene solución. Planteamos las condiciones KKT

$$\begin{aligned} \boldsymbol{A}\boldsymbol{x}^* &= \boldsymbol{b} \\ \boldsymbol{f}_i(\boldsymbol{x}^*) &\preceq_{\mathcal{K}_i} \boldsymbol{0}, & i &= 1, \ldots, m \\ \boldsymbol{\lambda}_i^* &\succeq_{\mathcal{K}_i^*} \boldsymbol{0}, & i &= 1, \ldots, m \\ \boldsymbol{\lambda}_i^{*T} \boldsymbol{f}_i(\boldsymbol{x}^*) &= 0, & i &= 1, \ldots, m \\ \nabla f_0(\boldsymbol{x}^*) + \sum_{i=1}^m D\boldsymbol{f}_i(\boldsymbol{x}^*)^T \boldsymbol{\lambda}_i^* + \boldsymbol{A}^T \boldsymbol{\nu}^* &= \boldsymbol{0} \end{aligned}$$

Si el problema (4.16) es estrictamente viable, las condiciones KKT son necesarias y suficientes para que $\boldsymbol{x}^*$ sea óptimo.

4.3.3.1 Logaritmo generalizado

Al ampliar las dimensiones de las funciones en las restricciones de desigualdad, necesitamos utilizar una versión diferente de las barreras definidas en §4.3.1, y para ello se define la función logaritmo generalizado que se utilizará en las funciones de barrera logarítmica de §4.3.3.2.

Decimos que $\psi : \mathbb{R}^q \to \mathbb{R}$ es un logaritmo generalizado para $\mathcal{K}$ si se cumple que

- ψ es cóncava, cerrada, dos veces diferenciable, $\mathbf{dom}\,\psi = \mathbf{int}\,\mathcal{K}$, y $\nabla^2\psi(\boldsymbol{y}) \prec \boldsymbol{0}$ para $\boldsymbol{y} \in \mathbf{int}\,\mathcal{K}$.

- Existe una constante $\theta > 0$ tal que para todo $\boldsymbol{y} \succ_{\mathcal{K}} \mathbf{0}$, y para todo $s > 0$ se cumple que $\psi(s\boldsymbol{y}) = \psi(\boldsymbol{y}) + \theta \log s$. Es decir, ψ se comporta como un logaritmo en un rayo dentro del cono $\mathcal{K}$.

La constante θ es lo que llamamos el grado de ψ, ya que $\exp \psi$ es una función homogénea de grado θ. Utilizaremos dos propiedades de los logaritmos generalizados:

$$\nabla\psi(\boldsymbol{y}) \succ_{\mathcal{K}^*} \mathbf{0} \tag{4.17}$$

por lo que ψ es $\mathcal{K}$-creciente, y

$$\boldsymbol{y}^T \nabla\psi(\boldsymbol{y}) = \theta$$

Por ejemplo, para el ortante no negativo, $\mathcal{K} = \mathbb{R}^n_+$, la siguiente función es un logaritmo generalizado de grado n.

$$\psi(\boldsymbol{x}) = \sum_{i=1}^{n} \log x_i$$

Para el cono de segundo orden, $\mathcal{K} = \{(\boldsymbol{x}, t) \in \mathbb{R}^{n+1} \mid \|\boldsymbol{x}\|_2 \leq t\}$, un logaritmo generalizado es

$$\psi(\boldsymbol{x}, t) = \log\left(t^2 - \|\boldsymbol{x}\|_2^2\right)$$

y su gradiente en $\boldsymbol{x} \in \mathbf{int}\,\mathcal{K}$ se calcula así

$$\frac{\partial\psi(\boldsymbol{x}, t)}{\partial x_i} = \frac{-2x_i}{(t^2 - \|\boldsymbol{x}\|_2^2)}, \qquad i = 1, \ldots, n$$
$$\frac{\partial\psi(\boldsymbol{x}, t)}{\partial t} = \frac{2t}{(t^2 - \|\boldsymbol{x}\|_2^2)}$$

Para el cono semidefinido positivo $\mathbb{S}^p_+$, un logaritmo generalizado es

$$\psi(\boldsymbol{X}) = \log\det \boldsymbol{X}$$

y el gradiente de ψ en $\boldsymbol{X} \in \mathbb{S}^p_{++}$ es

$$\nabla\psi(\boldsymbol{X}) = \boldsymbol{X}^{-1}$$

por lo que el producto interior de $\boldsymbol{X}$ y $\nabla\psi(\boldsymbol{X})$ es igual a $\mathrm{tr}(\boldsymbol{X}\boldsymbol{X}^{-1}) = p$, donde tr() representa la traza de una matriz.

4.3.3.2 Funciones barrera logarítmica

Volviendo al problema (4.16), si las funciones $\psi_1, \ldots, \psi_m$ son logaritmos generalizados para los conos $\mathcal{K}_1, \ldots, \mathcal{K}_m$, con grados $\theta_1, \ldots, \theta_m$, definimos la función barrera logarítmica de ese problema como

$$\phi(\boldsymbol{x}) = -\sum_{i=1}^{m} \psi_i(-\boldsymbol{f}_i(\boldsymbol{x})), \qquad \mathbf{dom}\,\phi = \{\boldsymbol{x} \mid \boldsymbol{f}_i(\boldsymbol{x}) \prec 0,\ i = 1, \ldots, m\}$$

Se trata de una función convexa porque las funciones ψ_i son $\mathcal{K}$-crecientes, y las funciones $\boldsymbol{f}_i$ son $\mathcal{K}_i$-convexas.

4.3.3.3 Camino central

Definimos un punto central $\boldsymbol{x}^*(t)$ para $t \geq 0$ como el que minimiza $tf_0 + \phi$, sujeto a $\boldsymbol{Ax} = \boldsymbol{b}$, es decir, la solución de

$$\begin{array}{ll} \underset{\boldsymbol{x}}{\text{minimizar}} & tf_0(\boldsymbol{x}) - \sum_{i=1}^m \psi_i(-\boldsymbol{f}_i(\boldsymbol{x})) \\ \text{sujeto a} & \boldsymbol{Ax} = \boldsymbol{b} \end{array}$$

Los puntos centrales se caracterizan por la condición de punto óptimo

$$t\nabla f_0(\boldsymbol{x}) + \nabla\phi(\boldsymbol{x}) + \boldsymbol{A}^T\boldsymbol{\nu} = t\nabla f_0(\boldsymbol{x}) + \sum_{i=1}^m D\boldsymbol{f}_i(\boldsymbol{x})^T\nabla\psi_i(-\boldsymbol{f}_i(\boldsymbol{x})) + \boldsymbol{A}^T\boldsymbol{\nu} = \boldsymbol{0} \tag{4.18}$$

4.3.3.4 Puntos duales en el camino central

Igual que en el caso escalar, los puntos del camino central producen puntos duales viables para el problema (4.16). Definimos

$$\begin{aligned} \boldsymbol{\lambda}_i^*(t) &= \frac{1}{t}\nabla\psi_i(-\boldsymbol{f}_i(\boldsymbol{x}^*(t))), \qquad i = 1, \ldots, m \\ \boldsymbol{\nu}^*(t) &= \frac{\boldsymbol{\nu}}{t} \end{aligned}$$

donde $\boldsymbol{\nu}$ es la variable dual óptima de (4.18). Según la propiedad (4.17) de los logaritmos generalizados, $\boldsymbol{\lambda}_i^*(t) \succ_{\mathcal{K}_i^*} \boldsymbol{0}$, y según (4.18) el lagrangiano se minimiza en $\boldsymbol{x}^*(t)$

$$L(\boldsymbol{x}, \boldsymbol{\lambda}^*(t), \boldsymbol{\nu}^*(t)) = f_0(\boldsymbol{x}) + \sum_{i=1}^m \boldsymbol{\lambda}_i^*(t)^T\boldsymbol{f}_i(\boldsymbol{x}) + \boldsymbol{\nu}^*(t)^T(\boldsymbol{Ax} - \boldsymbol{b})$$

La función dual en $(\boldsymbol{\lambda}^*(t), \boldsymbol{\nu}^*(t))$ es

$$\begin{aligned} g\left(\boldsymbol{\lambda}^*\left(t\right), \boldsymbol{\nu}^*\left(t\right)\right) &= f_0\left(\boldsymbol{x}^*\left(t\right)\right) + \sum_{i=1}^m \boldsymbol{\lambda}_i^*\left(t\right)^T \boldsymbol{f}_i\left(\boldsymbol{x}^*\left(t\right)\right) + \boldsymbol{\nu}^*\left(t\right)^T\left(\boldsymbol{A}\boldsymbol{x}^*\left(t\right) - \boldsymbol{b}\right) \\ &= f_0\left(\boldsymbol{x}^*\left(t\right)\right) + (1/t)\sum_{i=1}^m \nabla\psi_i\left(-\boldsymbol{f}_i\left(\boldsymbol{x}^*\left(t\right)\right)\right)^T \boldsymbol{f}_i\left(\boldsymbol{x}^*\left(t\right)\right) \\ &= f_0\left(\boldsymbol{x}^*\left(t\right)\right) - (1/t)\sum_{i=1}^m \theta_i \end{aligned}$$

El grado de ψ_i es $\theta_i = \boldsymbol{y}^T\nabla\psi_i(\boldsymbol{y})$. Por tanto,

$$\boldsymbol{\lambda}_i^*(t)^T\boldsymbol{f}_i(\boldsymbol{x}^*(t)) = -\theta_i/t, \qquad i = 1, \ldots, m$$

Entonces, si definimos

$$\bar{\theta} = \sum_{i=1}^m \theta_i$$

el punto primal viable $\boldsymbol{x}^*(t)$ y el punto dual viable $(\boldsymbol{\lambda}^*(t), \boldsymbol{\nu}^*(t))$ tienen una distancia de dualidad $\bar{\theta}/t$.

Como las propiedades del camino central se pueden generalizar para este tipo de problemas, podemos aplicar el método de barrera descrito en el apartado §4.3.1 utilizando la fase I del apartado §4.3.2.

4.3.4 Métodos de punto interior primal-dual

Este tipo de métodos se parece bastante al método de barrera, con algunas diferencias.

- Sólo hay una iteración, en la que se actualizan las variables primales y duales.
- Las direcciones de búsqueda se obtienen por el método de Newton aplicado a las ecuaciones KKT para el problema de centrado con barrera logarítmica. Las direcciones de búsqueda son parecidas, pero no exactamente las mismas.
- Las iteraciones primal y dual no son necesariamente viables.

En muchos casos los métodos de punto interior primal-dual son más eficientes que el método de barrera, sobre todo cuando se necesita mucha precisión. Otra ventaja es que pueden funcionar cuando el problema es viable pero no estrictamente viable.

En definitiva, este tipo de algoritmos constituye una evolución del caso anterior, eliminando la ineficiencia que suponían las iteraciones internas y externas, y haciendo que el método de Newton se enfoque directamente a la búsqueda del óptimo en base a las características del camino central. Para más información podemos consultar la bibliografía, [Wri97], [Nes95], [Boy04].

4.3.4.1 Dirección de búsqueda primal-dual

Para comprender el funcionamiento de este algoritmo, conviene revisar el apartado §4.2.3.1, donde empezábamos a ver el planteamiento de las condiciones KKT en función de los residuos primal y dual. En este caso añadimos la versión subóptima de la holgura complementaria, con lo que las condiciones de (4.13) quedan en forma de $\boldsymbol{r}_t(x, \lambda, \nu) = \mathbf{0}$

$$\boldsymbol{r}_t(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{\nu}) = \begin{bmatrix} \nabla f_0(\boldsymbol{x}) + D\boldsymbol{f}(\boldsymbol{x})^T\boldsymbol{\lambda} + \boldsymbol{A}^T\boldsymbol{\nu} \\ -\mathrm{diag}(\boldsymbol{\lambda})\boldsymbol{f}(\boldsymbol{x}) - (1/t)\mathbf{1} \\ \boldsymbol{A}\boldsymbol{x} - \boldsymbol{b} \end{bmatrix} = \mathbf{0}$$

donde $t > 0$, y hemos definido la función $\boldsymbol{f} : \mathbb{R}^n \to \mathbb{R}^m$ como la composición vectorial de todas las restricciones de desigualdad

$$\boldsymbol{f}(\boldsymbol{x}) = [f_1(\boldsymbol{x}), \ldots, f_m(\boldsymbol{x})]^T$$

y $D\boldsymbol{f}(\boldsymbol{x})$ es su matriz jacobiana. Si $\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{\nu}$ satisfacen que $\boldsymbol{r}_t(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{\nu}) = \mathbf{0}$, y $f_i(\boldsymbol{x}) < 0$, entonces $\boldsymbol{x} = \boldsymbol{x}^*(t)$, $\boldsymbol{\lambda} = \boldsymbol{\lambda}^*(t)$, y $\boldsymbol{\nu} = \boldsymbol{\nu}^*(t)$. La distancia de dualidad entre los puntos $\boldsymbol{x}$ y $(\boldsymbol{\lambda}, \boldsymbol{\nu})$ será m/t. El primer componente de $\boldsymbol{r}_t$ es el residuo dual

$$\boldsymbol{r}_{dual} = \nabla f_0(\boldsymbol{x}) + D\boldsymbol{f}(\boldsymbol{x})^T\boldsymbol{\lambda} + \boldsymbol{A}^T\boldsymbol{\nu}$$

y el último componente es el residuo primal

$$\boldsymbol{r}_{pri} = \boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}$$

El componente central es el residuo que corresponde a la condición de holgura complementaria.

$$\boldsymbol{r}_{cent} = -\text{diag}(\boldsymbol{\lambda})\boldsymbol{f}(\boldsymbol{x}) - (1/t)\mathbf{1}$$

Ahora consideramos el paso de Newton para resolver las ecuaciones lineales $\boldsymbol{r}_t(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{\nu}) = \mathbf{0}$, para un t fijo, en un punto $(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{\nu})$ que satisfaga $\boldsymbol{f}(\boldsymbol{x}) \prec \mathbf{0}$, $\boldsymbol{\lambda} \succ \mathbf{0}$. Tenemos en cuenta la nueva variable vectorial que agrupa nuestro problema

$$\boldsymbol{y} = (\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{\nu}), \qquad \Delta\boldsymbol{y} = (\Delta\boldsymbol{x}, \Delta\boldsymbol{\lambda}, \Delta\boldsymbol{\nu})$$

El paso de Newton se caracteriza por las siguientes ecuaciones lineales

$$\boldsymbol{r}_t(\boldsymbol{y} + \Delta\boldsymbol{y}) \approx \boldsymbol{r}_t(\boldsymbol{y}) + D\boldsymbol{r}_t(\boldsymbol{y})\Delta\boldsymbol{y} = \mathbf{0}$$

Expresado en términos de $\boldsymbol{x}$, $\boldsymbol{\lambda}$ y $\boldsymbol{\nu}$

$$\begin{bmatrix} \nabla^2 f_0(\boldsymbol{x}) + \sum_{i=1}^m \lambda_i \nabla^2 f_i(\boldsymbol{x}) & D\boldsymbol{f}(\boldsymbol{x})^T & \boldsymbol{A}^T \\ -\text{diag}(\boldsymbol{\lambda})D\boldsymbol{f}(\boldsymbol{x}) & -\text{diag}\left(\boldsymbol{f}(\boldsymbol{x})\right) & \mathbf{0} \\ \boldsymbol{A} & \mathbf{0} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \Delta\boldsymbol{x} \\ \Delta\boldsymbol{\lambda} \\ \Delta\boldsymbol{\nu} \end{bmatrix} = - \begin{bmatrix} \boldsymbol{r}_{dual} \\ \boldsymbol{r}_{cent} \\ \boldsymbol{r}_{pri} \end{bmatrix} \tag{4.19}$$

La solución de este sistema de ecuaciones proporciona la dirección de búsqueda primal-dual $\Delta\boldsymbol{y}_{pd} = (\Delta\boldsymbol{x}_{pd}, \Delta\boldsymbol{\lambda}_{pd}, \Delta\boldsymbol{\nu}_{pd})$.

Las direcciones de búsqueda primal y dual están acopladas por la matriz de coeficientes y por los residuos. Por ejemplo, la dirección primal de búsqueda $\Delta\boldsymbol{x}_{pd}$ depende del valor actual de las variables $\boldsymbol{\lambda}$ y $\boldsymbol{\nu}$, así como de $\boldsymbol{x}$. Observamos también que si $\boldsymbol{x}$ satisface que $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$, es decir, el residuo de viabilidad primal $\boldsymbol{r}_{pri}$ es cero, entonces $\boldsymbol{A}\Delta\boldsymbol{x}_{pd} = \mathbf{0}$, por tanto $\Delta\boldsymbol{x}_{pd}$ define una dirección primal viable: para todo s, $\boldsymbol{x} + \Delta\boldsymbol{x}_{pd}$ cumplirá que $\boldsymbol{A}\left(\boldsymbol{x} + s\Delta\boldsymbol{x}_{pd}\right) = \boldsymbol{b}$.

4.3.4.2 Distancia alternativa de dualidad

En el método de punto interior primal-dual las iteraciones $\boldsymbol{x}^{(k)}$, $\boldsymbol{\lambda}^{(k)}$ y $\boldsymbol{\nu}^{(k)}$ no son necesariamente viables, excepto en el límite, cuando el algoritmo converge. Esto significa que no podemos evaluar fácilmente una distancia de

dualidad $\eta^{(k)}$ asociada a cada paso k del algoritmo, tal como hacíamos en los pasos externos del método de barrera. En lugar de esto, definimos la nueva distancia de dualidad, para todo $\boldsymbol{x}$ que satisfaga $\boldsymbol{f} \prec \mathbf{0}$ y $\boldsymbol{\lambda} \succ \mathbf{0}$, como

$$\hat{\eta}(\boldsymbol{x}, \boldsymbol{\lambda}) = -\boldsymbol{f}(\boldsymbol{x})^T \boldsymbol{\lambda} \tag{4.20}$$

La distancia alternativa $\hat{\eta}$ sería la distancia original de dualidad si $\boldsymbol{x}$ fuese viable primal y $\boldsymbol{\lambda}$ y $\boldsymbol{\nu}$ viables duales, es decir, si $\boldsymbol{r}_{pri} = \mathbf{0}$ y $\boldsymbol{r}_{dual} = \mathbf{0}$. Observemos que el valor del parámetro t correspondiente a la nueva distancia de dualidad $\hat{\eta}$ es $m/\hat{\eta}$.

4.3.4.3 Algoritmo de punto interior primal-dual

El algoritmo 4.7 describe el método de punto interior primal-dual.

Algoritmo 4.7: Método de punto interior primal-dual

Datos: un punto $\boldsymbol{x}$ que satisfaga $\boldsymbol{f}(\boldsymbol{x}) \prec \mathbf{0}$, un dual $\boldsymbol{\lambda} \succ \mathbf{0}$, y los parámetros $\mu > 1$, $\epsilon_{\text{viable}} > 0$ y $\epsilon > 0$

repetir

- Determinar $t := \mu m/\hat{\eta}$
- Calcular la dirección de búsqueda primal-dual $\Delta \boldsymbol{y}_{pd}$
- Búsqueda lineal y actualización
  - Calcular la longitud del paso $s > 0$ y actualizar $\boldsymbol{y} := \boldsymbol{y} + s\Delta \boldsymbol{y}_{pd}$.

hasta que $\|\boldsymbol{r}_{pri}\|_2 \leq \epsilon_{viable}$, $\|\boldsymbol{r}_{dual}\|_2 \leq \epsilon_{viable}$ y $\hat{\eta} \leq \epsilon$

En el primer paso, el parámetro t se fija a un factor μ veces $m/\hat{\eta}$, que es el valor de t asociado a la distancia alternativa de dualidad $\hat{\eta}$. Si $\boldsymbol{x}$, $\boldsymbol{\lambda}$ y $\boldsymbol{\nu}$ fuesen centrales, con el parámetro t, y por tanto, con distancia de dualidad m/t, entonces en el paso 1 aumentaríamos t por un factor μ, que es exactamente la actualización utilizada en el método de barrera. Generalmente se obtienen buenos resultados dando valores a μ del orden de 10.

El algoritmo de punto interior primal-dual termina cuando $\boldsymbol{x}$ es primal viable y $\boldsymbol{\lambda}$ y $\boldsymbol{\nu}$ son duales viables dentro del margen de tolerancia ϵ_{viable}, y la nueva distancia de dualidad es menor que la tolerancia ϵ. Como este método suele tener una convergencia más rápida que lineal, es normal elegir ϵ_{viable} y ϵ pequeños.

La búsqueda lineal suele ser con retroceso, como veíamos en §4.1.1.3.

4.4 Resolución de problemas lineales con el algoritmo símplex

Cuando el problema a resolver es lineal podemos utilizar el algoritmo símplex, cuya aparición es anterior a los que hemos visto en §4.3. Su velocidad

de ejecución puede ser mayor que los anteriores. En términos de cálculo numérico se dice que el método símplex puede converger en tiempo exponencial en los casos peores, aunque en muchos ejemplos se resuelve en tiempo polinomial comparable a los algoritmos de punto interior. La complejidad temporal exponencial significa que se resuelve en un tiempo del orden de 2^n, donde n es el tamaño de la entrada al algoritmo; si la complejidad es polinomial su tiempo de resolución está en el orden de $2^{\log n}$. Los datos de este apartado los he obtenido de [Cor09].

Para resolver un problema LP primero tiene que estar planteado en formato estándar tal como veíamos en §3.3.1.1, de forma que únicamente tendremos desigualdades de no negatividad en las variables utilizadas. En estas condiciones, y teniendo en cuenta que el conjunto viable de un LP es un poliedro, el valor mínimo de la función objetivo se obtiene en al menos uno de sus vértices, tal como se demuestra en [Mur83b]. Los poliedros tienen un número finito de puntos extremos, por lo que el problema se reduce a un número limitado de iteraciones, aunque en algunos casos puede ser muy elevado. Otra propiedad importante es que si en un vértice no encontramos un mínimo, entonces en alguna de las aristas que pasan por ese vértice la función objetivo será estrictamente decreciente, por lo que en su extremo opuesto nos conectará con otro vértice en el que esta función es menor. En caso contrario, no hay cota inferior en esa arista y el problema no tiene solución.

El algoritmo se resuelve en dos fases. En la primera se encuentra un punto extremo de inicio, y en la segunda fase se recorren los vértices conectados con aristas decrecientes hasta encontrar el punto óptimo si el problema tiene solución.

Si revisamos la teoría de §2.4 es sencillo entender que el problema dual de un LP es también otro LP. En algunos casos la resolución del problema dual con el método símplex puede ser más rápida que la del problema primal, y por eso veremos en §5.2 que algunas herramientas como *GLPK*, *Gurobi*, etc., implementan la resolución alternativa del primal o el dual con el fin de reducir el tiempo de ejecución.

4.5 Métodos para resolver problemas con variables enteras

En este apartado tratamos un tipo de problemas que no es convexo, pero lo necesitamos como referencia para cuando veamos en el capítulo 5 que muchas herramientas de optimización incluyen algoritmos para resolver problemas en los que algunas componentes de la variable de optimización pueden ser enteras. Esto es lo que se conoce como problemas lineales con mezcla de

enteros (MILP[1] o MIP[2]). El contenido de esta sección está basado en [Chi10].

Una primera aproximación consiste en enumerar todas las soluciones posibles combinando los valores enteros que cada componente puede tomar con los de las otras, y entre todas estas soluciones elegimos la mejor. Esto puede funcionar con problemas muy pequeños, pero el número de combinaciones posibles aumenta excesivamente con el tamaño de las variables y sus rangos de valores. Sin embargo, podemos plantear la exploración de todas las combinaciones viables de las componentes enteras como una estructura de árbol. Por ejemplo, si la componente x_1 puede valer 1, 2, o 3, y las componentes x_2 y x_3 son binarias (0 o 1) el árbol de combinaciones será el de la figura 4.2.

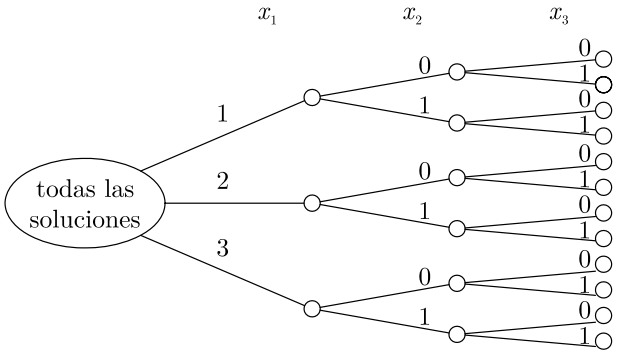

Figura 4.2: Ejemplo de estructura de árbol en un problema con variables enteras.

Los métodos de ramificar y acotar (*Branch-and-Bound*), se basan en construir las ramificaciones de este árbol haciendo que crezcan únicamente los nodos más prometedores en cada instante, y eliminando las ramas que vayan a dar resultados peores. Esto se asemeja a la poda en términos de jardinería, ya que descartamos permanentemente los nodos en los que podemos demostrar que su crecimiento no puede ser viable ni óptimo. También se habla de brotes para referirse a nodos que pueden crecer con las combinaciones de las siguientes variables.

Para este método es importante utilizar una función de acotación que nos permita conocer el mejor valor de la función objetivo que se puede obtener a partir de un determinado brote.

Otra aproximación a los problemas MIP son los algoritmos de plano cortante (*Cutting-Plane*), que plantean una versión del problema inicial sustituyendo las componentes enteras por reales, y a partir de ahí refinan la solución añadiendo restricciones que reducen el conjunto viable hasta obtener una solución óptima entera.

[1] *Mixed Integer Linear Program*
[2] *Mixed Integer Program*

También veremos herramientas que resuelven estos problemas con algoritmos de ramificar y cortar (*Branch-and-Cut*). Estos métodos son una combinación de los dos anteriores.

En [Esc10] y en las referencias que contiene podemos ampliar el conocimiento de este tipo de problemas y encontramos aplicaciones de los algoritmos de esta sección en el ámbito de las telecomunicaciones.

Parte II

Herramientas de optimización convexa

Capítulo 5

Revisión de herramientas disponibles de optimización convexa

En este capítulo se hace una revisión de las herramientas más importantes de optimización convexa que podemos encontrar en el mercado. Se estudian las características de cada programa de forma que podamos conocer cuál es más adecuado para el problema que necesitemos resolver, en función de diferentes propiedades como su complejidad o facilidad de uso, condiciones comerciales de distribución, y la variedad de problemas y formatos de datos que admita.

Los principales apartados que encontraremos corresponden a los tipos de programas que existen, que pueden ser de modelado o de resolución. Veremos la diferencia entre ambos. También he reservado un espacio para herramientas de optimización global orientadas a problemas no necesariamente convexos.

Al final del capítulo se incluye un ejemplo de problema cuadrático que nos permite comparar el procedimiento de utilización de varias herramientas, y que también sirve para visualizar de forma intuitiva la estructura de los problemas que estamos tratando.

5.1 Motivación

En estos momentos es posible para un usuario adquirir una gran variedad de aplicaciones que resuelven problemas de optimización, pero estos pueden ser de una naturaleza muy diversa, por lo que cada herramienta se centra en la resolución de un tipo o de un grupo más o menos amplio de problemas. Por ejemplo, si buscamos herramientas de optimización lineal, la cantidad de programas que encontramos en el mercado es innumerable; incluso en las hojas de cálculo de los paquetes de ofimática más populares se pueden

resolver LPs.

Cuando los problemas son convexos, las técnicas disponibles de resolución permiten una altísima eficiencia en cuanto a tiempo de convergencia, dimensiones de las variables, y cantidad de funciones de restricción que podemos indicar. Por este motivo es importante explotar las posibilidades que se nos ofrecen en este tipo de problemas, y aquí es donde nos hemos centrado en este trabajo.

Podemos distinguir entre los programas específicos de resolución (*solvers*), que suelen estar especializados en algún tipo de problema (lineales, cuadráticos, cónicos, etc.); y por otro lado los sistemas de modelado, que se encargan de automatizar las transformaciones necesarias para convertir un problema en su formato estándar. Los primeros se ven en §5.2, y los sistemas de modelado en §5.3.

Los *solvers* corresponden a un nivel bajo de programación, en el sentido de que son más complicados para el usuario, y están enfocados a aplicar un determinado algoritmo para un tipo de problema en concreto. Normalmente forman parte del "núcleo" de los programas de modelado, que a su vez, pueden utilizar una gran cantidad de *solvers* para resolver en cada caso un conjunto variado de problemas.

Los sistemas de modelado proporcionan un nivel alto de programación más cercano al lenguaje utilizado en la teoría de optimización que hemos estudiado en la primera parte de este trabajo. No están especializados en un tipo de problema concreto, porque para eso cuentan con varios *solvers* una vez que han transformado el problema genérico para el que se utilizan.

5.2 Herramientas de resolución

La cantidad de herramientas disponibles en el mercado para resolución de problemas de optimización es muy grande. En esta sección se analizan algunas de ellas agrupándolas por el tipo de problemas que resuelven. Para no extenderme excesivamente con un número elevado de ejemplos, he seleccionado en cada clase los cuatro programas incluidos en el mayor número de sistemas de modelado de los estudiados en §5.3. Veremos que la selección por el tipo de problemas abarcado no es un criterio exacto, porque muchos *solvers* pueden tratar más de un tipo de problemas. Por ejemplo, las herramientas de resolución cuadrática resuelven también problemas lineales, ya que estos se pueden ver como un caso particular de los anteriores, tal como vimos en §3.3. En definitiva, para asignar cada herramienta a un tipo de problema me he basado en la clasificación que de ellas hacen los programas de modelado que las incluyen.

Hemos visto en el capítulo 4 algunos algoritmos que se aplican para resolver problemas de optimización convexa, y en general son estos los que se emplean en las aplicaciones comerciales. La utilización de los métodos de

punto interior es común en la gran mayoría de las aplicaciones, y como hemos visto, se resumen en descomponer el problema original en una secuencia de problemas cuadráticos que a su vez se descomponen en una serie de sistemas lineales de ecuaciones. Por este motivo, empezaré por aclarar en §5.2.1 algunos aspectos que motivan la eficiencia de estas herramientas.

5.2.1 Efectividad de las herramientas de resolución

Con la teoría del capítulo 4 seríamos capaces de desarrollar un programa de resolución adaptado a un determinado problema, y nos ahorraríamos los recursos que un programa comercial dedica a otros aspectos que no necesitásemos. Sin embargo, en algunos casos nos podemos encontrar con que no somos capaces de superar la velocidad de resolución de un programa comercial. En las clases de [Boy08] se ilustra lo que quiero decir con un ejemplo bastante clarificante: Supongamos que hacemos un programa para multiplicar dos matrices. En C o cualquier otro lenguaje no requiere más de 5 o 6 líneas de código y no hay mucho más que podamos refinar. Sin embargo, al ejecutar esta multiplicación en Matlab, Octave, Mathematica, etc. siempre dará el resultado en menos tiempo. Esto se debe en gran parte a que los cálculos de álgebra en estos programas se hacen desde bibliotecas[1] como LAPACK[2], que utiliza como capa inferior las rutinas BLAS[3]. Éstas ejecutan las operaciones por bloques del tamaño óptimo para los registros de la CPU[4], teniendo en cuenta los accesos a memoria *caché*, y otros aspectos de este tipo, de forma que el programa que las utiliza ejecuta las operaciones algebraicas con el mínimo de carga computacional.

Esto nos indica en qué se basa la efectividad de los programas comerciales, pero otro factor importante es aprovechar la estructura de las matrices que forman parte de los problemas de optimización. Por ejemplo, en el caso de *Mosek*, se resuelven grandes problemas aprovechando los patrones de dispersión de los sistemas lineales, porque cuando las matrices no son densas, se puede trabajar con dimensiones muy elevadas. Es evidente que resulta poco eficiente almacenar y operar con matrices dispersas si guardamos todos sus elementos. De forma intuitiva utilizaríamos únicamente los índices y valores de los elementos distintos de cero. La clave para manipular matrices dispersas está en aprovechar la estructura que muestran sus elementos para almacenarlo por ejemplo en lo que se conoce como formato comprimido de columnas, o de filas, etc. Existe una gran variedad de estructuras de datos

[1] Es frecuente encontrar en la bibliografía el término "librería" como traducción del inglés *library*. En este trabajo prefiero utilizar "biblioteca" que corresponde a la traducción exacta a pesar de que el término "librería" es ampliamente reconocido y aceptado en la comunidad de desarrolladores.

[2] *Linear Algebra Package*

[3] *Basic Linear Algebra Subprograms*

[4] *Central Processing Unit*

Nombre	Algoritmos	Problemas	Licencia	Limitaciones en demo
lp_solve	símplex ramificar y acotar	LP MILP	LGPL[7]	
GLPK	símplex primal símplex dual p. interior primal-dual ramificar y cortar	LP MILP	GPL	
BPMPD	p. interior primal-dual	LP QP	libre	
MINOS	símplex gradiente reducido lagrangiano proyectado	LP no lineales	comercial	3 meses no gratuíta

Tabla 5.1: Resumen de las herramientas de resolución lineales.

para almacenar matrices dispersas. Podemos ampliar información sobre esta materia en [Eij92].

Si por el contrario, las matrices son densas, también necesitamos aprovechar su estructura que nos puede facilitar la factorización en formas más sencillas, como por ejemplo, utilizando la descomposición de Cholesky, o el complemento de Schur. Estas técnicas son las que, por ejemplo, hacen que en Matlab sea más rápido[5] resolver el sistema de ecuaciones $\boldsymbol{Ax} = \boldsymbol{b}$ con el código `x=A\b` que escribiendo `x=inv(A)*b`. Me ha parecido interesante añadir el apéndice A dedicado a estos recursos, pues en definitiva, vuelvo a incidir en que una gran parte de los algoritmos de optimización terminan basándose en la resolución de muchas series de ecuaciones lineales.

5.2.2 Herramientas de resolución de problemas lineales

Este grupo de programas es el más extenso. En este apartado analizamos *lp_solve*, *GLPK*[6], *BPMPD* y *MINOS*, aunque en el entorno académico puede resultar más accesible utilizar en *Matlab* el comando `linprog` del complemento *Matlab Optimization Toolbox* o incluso los *solvers* de *Microsoft Excel* y *OpenOffice*.

En la tabla 5.1 tenemos un resumen de las principales características de los programas estudiados en este apartado.

5.2.2.1 lp_solve

Es un programa de resolución de problemas lineales que utiliza el método símplex que hemos visto en §4.4, y el de ramificar y acotar (*Branch-and-*

[5] Si $\boldsymbol{A} \succeq \boldsymbol{0}$

[6] *GNU Linear Programming Kit*

Bound) para problemas de variables enteras y conjuntos especiales ordenados. También hemos visto este método en §4.5. La información sobre esta herramienta procede de [Ber11].

El tamaño de modelo admitido no tiene límite, y la entrada de datos y especificaciones se hace básicamente de tres formas:

- Por medio de las bibliotecas de interfaz de programación (*API*[8])
- Con ficheros de entrada
- A través del entorno integrado de desarrollo (*IDE*[9])

Para la entrada por archivo acepta formatos estándar *lp* o *mps*. Se puede enlazar como biblioteca desde C, Visual Basic, .NET, Delphi, Excel, Java,... También se puede utilizar desde *AMPL*[10], *Matlab*, *O-Matrix*, *Scilab*, y *Octave*.

5.2.2.1.1 Licencias

Se distribuye como *software* libre con licencia LGPL.

5.2.2.2 GLPK

Es una biblioteca GNU para resolver problemas lineales utilizando métodos de punto interior, que también puede trabajar con problemas MIP. Ofrece interfaces para *Matlab* y para el lenguaje de modelado *AMPL*. Está compuesto por un conjunto de rutinas escritas en ANSI[11] C, y el paquete incluye componentes que implementan los siguientes algoritmos:

- Símplex dual y primal
- Punto interior primal-dual
- Ramificar y cortar

También se incluye el interfaz de programación o *API* junto con el *solver*. La información de esta herramienta procede de [Mak11].

5.2.2.2.1 Licencias

GLPK es parte del proyecto GNU, y se ofrece bajo la licencia pública general GNU-GPL.

[8] *Application Programming Interface*
[9] *Integrated Development Environment*
[10] *A Mathematical Programming Language*
[11] *American National Standards Institute*

5.2.2.3 BPMPD

Se trata de una implementación del método de punto interior primal-dual escrita en C. Es una herramienta para programas lineales, aunque en la última versión se ha añadido también soporte para programas cuadráticos.

La información sobre *BPMPD* procede de [Més98] y de la dirección *web* en la que esta cita se encuentra.

La entrada es por archivos *mps* a los que ya nos hemos referido antes. Su eficiencia se basa en un tratamiento previo muy depurado de los datos antes del algoritmo:

- Se eliminan filas y columnas vacías, y filas o columnas con un solo elemento.
- Se eliminan columnas libres con un solo elemento.
- Se quitan restricciones redundantes, tanto en el problema primal como en el dual.
- Se sustituyen variables duplicadas
- Se quitan cotas redundantes para las variables
- Se eliminan variables libres
- Se construye una matriz de restricciones dispersa.

Este programa utiliza todos los conceptos explicados en §5.2.1. En concreto, si revisamos la teoría de los algoritmos de punto interior primal-dual de §4.3.4.3, para resolver el sistema de ecuaciones (4.19) utiliza dos aproximaciones: en la primera trata de resolver directamente el sistema original detectando su estructura de forma inteligente; y en la segunda utiliza una versión aumentada del sistema de acuerdo con el lema de inversión matricial explicado en §A.4. Para determinar cuál de ellas es más beneficiosa en un determinado problema utiliza un analizador simbólico con métodos heurísticos. También aplica un analizador de dispersión para favorecer la velocidad de procesamiento.

5.2.2.3.1 Licencias

Este programa fue desarrollado por Csaba Mészáros en el MTA SZTAKI[12] de Hungría (Instituto de investigación de informática y automática) [Més96], y su autor permite descargarlo libremente como ejecutable para *Windows* o como *DLL*[13].

[12] *Magyar Tudományos Akadémia Számítástechnikai és Automatizálási Kutatóintézete*
[13] *Dynamic Link Library*

5.2.2.4 MINOS

Es un paquete escrito en Fortran que sirve para resolver problemas de optimización a gran escala. Además de LP también resuelve problemas no lineales, aunque es especialmente eficiente cuando se trata de restricciones dispersas y lineales.

Utiliza una implementación dispersa del método símplex para problemas lineales. Para funciones no lineales utiliza un método de gradiente reducido, con aproximaciones casi-Newton para el hessiano reducido. Si se incluyen restricciones no lineales, se utiliza un algoritmo de lagrangiano proyectado relacionado con el método de Robinson. Este método resuelve una secuencia de subproblemas en los que las restricciones se linealizan y el objetivo es un lagrangiano aumentado que incluye las funciones lineales y no lineales. Con este método la convergencia aproximada es rápida.

La información de esta herramienta procede de [Mur83a] y de la *web* en la que se encuentra esta referencia.

5.2.2.4.1 Licencias

La licencia de este programa es comercial. El propietario es *Stanford Business Software, Inc.*, y ofrece precios para clientes particulares o empresas y para organismos académicos en formato individual, para departamento o para corporaciones completas. También hay una versión de evaluación para tres meses, pero no es gratuita.

5.2.3 Herramientas de resolución de problemas cuadráticos

Para resolver problemas cuadráticos de grandes dimensiones podemos utilizar las herramientas que se estudian en este apartado, *CPLEX*, *Gurobi*, *Xpress-Optimizer*, y *OOQP*[14], y muchas otras que no he abordado para poner un límite razonable de tamaño en esta revisión. También se pueden resolver problemas cuadráticos utilizando programas orientados a problemas cónicos, ya que son un caso particular de estos según vimos en §3.3.3. Estas herramientas sirven también para resolver problemas lineales, ya que los LPs son un subconjunto de los problemas cuadráticos, siguiendo el esquema de la figura 3.1. Por otro lado, el complemento *Matlab Optimization Toolbox* nos permite utilizar el comando `quadprog` para resolver problemas cuadráticos con restricciones lineales.

El contenido de este apartado se resume en la tabla 5.2.

5.2.3.1 CPLEX

IBM ILOG CPLEX es una herramienta de optimización ampliamente utilizada en la industria para estudios de alternativas, cuellos de botella o

[14] *Object Oriented Quadratic Program*

Nombre	Algoritmos	Problemas	Licencia	Limitaciones en demo
CPLEX	símplex primal símplex dual p.interior barrera ramificar y cortar	LP QP redes QCQP MILP MIQP[15]	comercial	90 días limitada en tamaño
Gurobi	símplex primal símplex dual barrera paralelo plano cortante heurísticas	LP QP MILP MIQP	comercial	500 variables y 500 restricciones. Demo académica: sin limitaciones
Xpress-Optimizer	símplex primal símplex dual p.interior barrera ramificar y acotar	LP QP MILP MIQP	comercial	30 días Demo académica: 400 filas, 800 col. 5000 coef. de mat. y 400 var. enteras
OOQP	p. interior primal-dual	QP	libre	

Tabla 5.2: Resumen de las herramientas de resolución cuadráticas.

inconsistencias en las tomas de decisiones, así como para desarrollar planes y horarios que se adapten al curso de las operaciones. Se utilizan en diseño de líneas aéreas, planes de producción, gestión de carteras, o planificación temporal de tareas.

Los problemas que resuelve *CPLEX Optimizer* pueden ser lineales, cuadráticos, con variables enteras, con restricciones cuadráticas, y también cónicos de segundo orden. Esto último hace que podría haber puesto esta herramienta en el siguiente apartado, pero en muchos programas se hace referencia a *CPLEX* como un *solver* para programación cuadrática, probablemente por su capacidad en versiones más antiguas. También indica en sus especificaciones que se puede utilizar para problemas de restricciones, o de viabilidad. Los datos sobre esta herramienta proceden de [CPL10].

5.2.3.1.1 Algoritmos

Todos los algoritmos vienen integrados con métodos de procesamiento previo para reducir el tamaño de los problemas y con ello el tiempo de resolución sin necesidad de intervención por parte del usuario.

Los algoritmos símplex primal y símplex dual resuelven problemas lineales y cuadráticos con restricciones lineales, y en *CPLEX* se incluye una versión creada específicamente para problemas de redes.

El algoritmo de punto interior con barrera resuelve problemas lineales,

y cuadráticos con restricciones cuadráticas. Las soluciones por este método también se pueden adaptar a los algoritmos símplex para reinicios rápidos y análisis de sensibilidad. Los reinicios rápidos son una técnica relacionada con el análisis de sensibilidad que vimos en §2.4.4 y que sirve para ver cómo se comporta nuestra solución al modificar algunos parámetros del problema sin tener que volver a ejecutar la optimización completamente.

Para los problemas con variables enteras se utiliza el algoritmo de ramificar y cortar (*Branch-and-Cut*), donde el usuario puede modificar las estrategias heurísticas y los planos de corte, definiendo si es más importante encontrar una solución óptima o determinar rápidamente una buena solución viable. El usuario también puede pedir que el optimizador devuelva múltiples soluciones para comparar los efectos de cada preferencia.

También se incluyen versiones en paralelo que aprovechan los sistemas con CPUs múltiples para resolver problemas de dificultad extrema. También se pueden utilizar optimizadores concurrentes con diferentes algoritmos para comparar sus velocidades.

5.2.3.1.2 Utilización

CPLEX Optimizer es un componente del paquete de modelado *CPLEX Optimization Studio*, pero también tiene conectores en otros programas como *AIMSS*, *AMPL*, *GAMS*[16], *MPL* y *Microsoft Solver Foundation*. Existe también un conector para *Microsoft Excel*.

Para incluir este *solver* en una aplicación se proporcionan bibliotecas de componentes para C, C++, .NET, Java y Python. Estas bibliotecas incluyen rutinas para definir, resolver, analizar y crear informes de los problemas de optimización y sus soluciones, y también permiten a los desarrolladores depurar sus aplicaciones

Además, el optimizador interactivo es una utilidad de línea de comandos que permite al usuario leer y escribir ficheros de programación matemática y ajustar el funcionamiento del optimizador a los requerimientos de cada problema específico.

CPLEX Optimizer se puede utilizar con una orientación matricial para representar los modelos matemáticos, y también con un sistema de alto nivel que utiliza objetos de modelado para ayudar al desarrollador a aprovechar las aproximaciones orientadas a objetos.

5.2.3.1.3 Licencias

IBM ILOG CPLEX Optimizer se vende con una licencia comercial para una gran variedad de plataformas. Es posible conseguir una versión de evaluación para 90 días, que viene limitada en tamaño.

[16] *General Algebraic Modeling System*

5.2.3.2 Gurobi

Este optimizador sirve para programas lineales, cuadráticos y sus versiones de variable entera (MILP, MIQP). Su diseño se planteó para explotar los procesadores modernos de núcleo múltiple.

Para resolver LPs y QPs incluye implementaciones de alto rendimiento de los métodos símplex primal y dual, y un sistema de barrera paralelo. Para los modelos MILP y MIQP incorpora métodos de plano cortante y heurísticas.

En todos los casos se aprovechan técnicas de preprocesado para simplificar los modelos y reducir el tiempo de resolución.

Este programa está escrito en C y se puede acceder a él desde varios lenguajes: además de un interfaz interactivo con Python y con C orientado a matrices, también se incluyen adaptaciones para C++, Java y .NET. Las adaptaciones requieren muy poca carga de memoria o de procesador, por lo que funcionan de forma muy eficiente.

5.2.3.2.1 Licencias

Se pueden comprar licencias en diferentes escenarios: Para sistemas simples, para usuarios en red con licencias flotantes, y para incluir *Gurobi* como parte de otro producto.

También hay una versión de evaluación gratuita, que está limitada a 500 variables y 500 restricciones. Además hay licencias gratuitas para uso académico que no tienen restricciones de tamaño.

Otra variante que se ofrece es la "nube" de *Gurobi*, que se puede utilizar por horas por medio de la *nube elástica de computación de Amazon* (*Amazon Elastic Computing Cloud -EC2-*).

5.2.3.3 Xpress-Optimizer

La herramienta de optimización cuadrática de *Xpress* se llama *Xpress-Optimizer*, aunque con la evolución de versiones, en este momento también existe un paquete de modelado (*Xpress-Mosel*) y motores de resolución para abarcar más tipos de problemas (*Xpress-SLP*).

Este optimizador trata las matrices dispersas de forma eficiente comprimiendo datos para resolver problemas de grandes dimensiones que se encuentran fácilmente en la industria. Los algoritmos que incluye permiten resolver problemas lineales, cuadráticos y sus variantes con variables enteras.

Se puede utilizar desde la línea de comandos de distintos sistemas operativos, o como biblioteca enlazable desde C, C++, Java, Fortran, Visual Basic y .NET. Es compatible con los formatos estándar de ficheros de entrada *lp* y *mps*. También se puede utilizar desde un entorno visual de desarrollo que se llama *Xpress-IVE*.

Para los problemas lineales implementa el algoritmo símplex en versiones primal y dual, con procesamiento previo para reducir el tamaño y el tiempo de resolución. El usuario puede configurar una gran variedad de parámetros para tener un control avanzado del proceso de optimización. Además cuenta con recursos de diagnóstico y detección de inviabilidades.

Para los problemas cuadráticos y también para los lineales se utiliza el método de barrera que hemos visto en §4.3.1. Para favorecer la implementación del algoritmo se utilizan procesos de factorización de Cholesky que aprovechan la estructura de las matrices que forman parte de la resolución. El método trabaja de forma paralela en caso de máquinas con múltiples procesadores o núcleos.

Para los problemas enteros se utiliza el algoritmo de ramificar y acotar (*Branch-and-Bound*) que ya hemos visto en otros productos.

5.2.3.3.1 Licencias

Los productos de optimización *Xpress* pertenecen a la empresa *FICO*, y se distribuyen con licencia comercial.

Es posible obtener una versión de evaluación durante 30 días sin limitaciones en capacidad ni funcionalidades, incluyendo soporte técnico y documentación. Esta opción está disponible para empresas.

También hay una versión gratuita para estudiantes, limitada a 400 restricciones (filas), 800 variables (columnas), 5000 coeficientes de matrices (elementos) y 400 variables globales enteras y binarias.

5.2.3.4 OOQP

Es un paquete programado en C++ orientado a objetos, que se basa en un método de punto interior de tipo primal-dual. Sirve para resolver problemas cuadráticos convexos. Contiene código que se puede utilizar en aplicaciones externas a este paquete, para resolver una gran variedad de problemas cuadráticos, que pueden ser dispersos, o problemas de regresión de Huber, y QPs con restricciones de acotación.

También se puede utilizar para diseñar otros *solvers* para otras clases de QPs estructurados. Su diseño permite la sustitución de los módulos de álgebra lineal, por lo que se pueden probar diferentes paquetes estándar de álgebra lineal.

Existe un interfaz para *Matlab*.

5.2.3.4.1 Licencias

Los derechos de copia están a nombre de E. Michael Gertz, que permite la libre copia, uso, redistribución y modificaciones para trabajos derivados, si estos cambios se documentan adecuadamente. El autor solicita que se

Nombre	Algoritmos	Problemas	Licencia	Limitaciones en demo
Mosek	símplex primal símplex dual p. interior id. base ramificar, acotar y cortar	LP QP convexos MILP MIQP	comercial	30 días
				Demo académica: 60 días sin limitaciones
SeDuMi	p. interior	LP QP SDP cónicos	GPL	
SDPT3	camino no viable	LP QP SDP	GPL	
PENSDP	p. interior	SDP	comercial	1 més no renovable
				Demo académica: 3 meses renovable

Tabla 5.3: Resumen de las herramientas de resolución de problemas cónicos.

incluya una referencia a los derechos de copia como comentario en el código fuente que utilice este *software*.

5.2.4 Herramientas de resolución de problemas cónicos

Este apartado contiene la descripción de cuatro herramientas de resolución de problemas cónicos: *Mosek*, *SeDuMi*, *SDPT3* y *PENSDP*. Sus características se resumen en la tabla 5.3.

5.2.4.1 Mosek

De acuerdo con [Mos11], *Mosek* es un programa de optimización diseñado para resolver problemas matemáticos a gran escala. Incluye rutinas de resolución para distintos tipos de problemas: lineales, cuadráticos, cónicos, convexos en general y problemas con mezcla de enteros.

Sus características más destacables son:

- El tamaño de los problemas está limitado únicamente por la memoria disponible.
- Un optimizador de punto interior con identificación de base.
- Optimizadores símplex primal y dual para programación lineal.

- Procesamiento previo para reducir el tamaño del problema antes de la optimización.
- Incluye un optimizador especial para problemas de redes y estructuras de flujos.
- Para problemas con mezcla de variables enteras se implementa un algoritmo de ramificar, acotar y cortar.

Incluye interfaces para lenguajes como C/C++, .NET, Java y Python, además de un complemento para *Matlab*. El optimizador de punto interior es capaz de explotar múltiples CPUs o núcleos. También dispone de un optimizador concurrente para resolver un mismo problema con diferentes optimizadores simultáneamente. La entrada de datos se puede hacer con ficheros en formatos estándar como *mps*, *lp* y *xml*. Incluye herramientas para diagnóstico y reparación de inviabilidad. También se puede hacer análisis de sensibilidad en problemas lineales. Existen conectores para utilizar *Mosek* desde muchos lenguajes de programación como C, C++, Java, *Matlab*, .NET y Python. También se puede utilizar desde herramientas de modelado como *AIMMS*, *COIN-OR*, *GAMS*, *Microsoft Solver Foundation* y *OptimJ*.

5.2.4.1.1 Licencias

Mosek está sujeto a una licencia comercial en la que se especifica el número de copias que se pueden utilizar simultáneamente y las características permitidas. También se pueden utilizar licencias flotantes para varios usuarios en forma de testigo que se devuelve al servidor después de utilizarlo.

Se pueden conseguir licencias de prueba en dos modalidades:

- Licencia de prueba para 30 días
- Licencia libre para uso académico, que está limitada a 90 días y contiene todas las características del producto.

5.2.4.2 SeDuMi

Es un paquete que se utiliza para resolver problemas de optimización sobre conos simétricos. Esto incluye problemas lineales, cuadráticos, semidefinidos, cónicos, y cualquier combinación de los anteriores. La información sobre esta herramienta la he obtenido de [Stu98], y según se indica en esta cita, está inspirado en una técnica conocida como empotrado autodual publicada en [Ye94], que consiste en optimizar sobre conos homogéneos autoduales[17]. Esto permite resolver algunos problemas en una sola fase, que lleva a la solución óptima o a una certificación de inviabilidad.

Sus características más destacables son las siguientes:

[17] Un cono cuyo dual es él mismo (en §2.2.4 se define qué es esto)

- Permite utilizar valores complejos.
- Genera soluciones duales en problemas no viables.
- Aprovecha las matrices dispersas para acelerar la ejecución.
- Gestiona las columnas densas de forma separada, y con ello favorece la dispersión.
- Puede importar problemas lineales en formato *mps* y problemas semidefinidos en formato *sdpa*.

5.2.4.2.1 Procesamiento previo

El procesamiento previo es algo limitado en las versiones actuales. Las técnicas utilizadas son:

- Para problemas SOCP/SDP se hace una separación de columnas densas y dispersas.
- El programa detecta bloques diagonales de SDPs y los convierte en variables no negativas.
- Las variables libres se incluyen en un cono cuadrático en lugar de separarlas.
- Si hay variables libres separadas, se agrupan.

Está previsto implementar varias técnicas para explotar la estructura especial de algunos problemas a gran escala.

5.2.4.2.2 Matrices dispersas

Por defecto, todas las matrices se almacenan como dispersas. Sin embargo, especialmente para problemas SDP la matriz del problema puede ser densa, por lo que su tratamiento en forma dispersa necesita demasiada información de índices. Está previsto utilizar la versión *ATLAS*[18] del paquete *BLAS* que hemos visto en §5.2.1 para manipular matrices densas, incorporando este paquete y distinguiendo los casos denso y disperso, como se hace en la mayor parte del resto de *solvers*.

5.2.4.2.3 Licencias

SeDuMi se distribuye de forma gratuita bajo los términos de código libre GPL.

[18] *Automatically Tuned Linear Algebra Software*

5.2.4.3 SDPT3

Esta herramienta está diseñada para resolver problemas cónicos en los que el cono de las restricciones puede ser el producto de conos semidefinidos, conos de segundo orden, ortantes no negativos, o espacios euclídeos. La función objetivo es la suma de funciones lineales y términos de barrera logarítmica asociados con los conos de las restricciones. La información de esta herramienta procede del manual [Toh06].

El código está escrito en *Matlab*, aunque también contiene rutinas escritas en C. Su funcionamiento se basa en un algoritmo de seguimiento de camino primal-dual no viable. Esto es una variante del algoritmo primal-dual que estudiamos en §4.3.4 en el que se utilizan dos versiones del paso de Newton definido en (4.19). Una de ellas está relacionada con este mismo sistema de ecuaciones, y se conoce como dirección NT[19], y la otra intenta linealizar una versión simétrica del mismo (llamada dirección HKM[20]). Si revisamos §4.2.3 y adaptamos las ecuaciones (4.19) nos podemos hacer una idea de la base de este algoritmo.

Sus características más destacables son:

- Permite utilizar variables libres.
- Se pueden resolver problemas de maximización de determinantes.
- Puede resolver SDPs con datos complejos.
- Se basa en un modelo de 3 parámetros (en referencia a la ecuación (4.19)) con restricciones en conos homogéneos autoduales, de forma semejante a *SeDuMi*.

5.2.4.3.1 Licencias

SDPT3 se distribuye bajo los términos de licencia pública general de GNU GPL 2.0, por lo que no es compatible para aplicaciones comerciales, aunque en este último caso las empresas se pueden poner en contacto con los autores para personalizar alternativas.

5.2.4.4 PENSDP

Es un programa para resolver problemas de tipo semidefinido, que tienen una función de objetivo lineal, y desigualdades matriciales lineales como restricciones, tal como vimos en §3.3.4.2.

Está orientado a problemas SDP a gran escala con matrices densas o dispersas.

[19] *Nesterov-Todd*

[20] *Helmberg-Rendl-Vanderbei-Wolkowick (1996), Kojima-Shindoh-Hara (1997), Monteiro (1995)*

5.2.4.4.1 Principales características

- Ahorro de utilización de memoria por liberación del hessiano.
- Solución iterativa para el sistema de Newton.
- Criterios de parada basados en la medida de error *DIMACS*[21], que da como resultado unos cálculos más fiables.
- Modo híbrido para la solución del sistema de Newton.
- Orientado a problemas a gran escala.
- Tratamiento eficiente de distintos patrones de dispersión en los datos del problema.
- Detección de no viabilidad.

5.2.4.4.2 Utilización

Se puede utilizar como programa autónomo, con ficheros de entrada en formato de *SDPA*[22], y también se puede utilizar desde *Matlab*, como función desde las herramientas de modelado *TOMLAB* o *YALIMP*. Otra opción es utilizarlo como rutina que se puede llamar desde los lenguajes de programación C o Fortran. En este último caso los datos se comunican en forma de parámetros de dichas rutinas.

5.2.4.4.3 Licencias

PENOPT se distribuye bajo una licencia comercial, que se puede comprar en diferentes formatos, dependiendo de su utilización para una empresa, un departamento, o un usuario único. También hay otras modalidades para uso académico, a nivel de universidad, departamento o usuario único, con precios más reducidos.

Por otro lado, también existe una licencia gratuita para desarrolladores, orientada a usuarios en entorno académico, y está limitada a tres meses, pudiendo renovarse cuando se necesite. Esta licencia se ofrece a investigadores que desarrollen y prueben *software* de optimización y utilicen *PENOPT* para pruebas, o también si lo necesitan para desarrollar *software* en aplicaciones que requieran resolver problemas de optimización.

Además, existe una licencia de evaluación para usuarios comerciales, que expira en un mes y no se puede renovar. Esta licencia no es gratuita, aunque su precio se descuenta si finalmente se compra una licencia definitiva antes de un plazo determinado.

[21] *Center for Discrete Mathematics and Theoretical Computer Science*
[22] *SemiDefinite Programming Algorithm*

5.3 Sistemas de modelado

De acuerdo con la nomenclatura utilizada en la información que he podido obtener sobre distintos programas de optimización, un modelo es una descripción compacta de un problema que consiste en unas líneas de código con la sintaxis específica de cada herramienta, y está compuesto por variables de optimización, conjuntos, y ecuaciones que describen la función objetivo y las restricciones además de otros parámetros como datos de precisión, informes de estado, etc., que varían de un producto a otro. A grandes rasgos, son todos los elementos que aparecen en un problema genérico como (2.1) y algunos más. Por ejemplo, en *CVX* el modelo corresponde a las líneas de código comprendidas entre `cvx_begin` y `cvx_end`. Cuando lleguemos a §5.5.1 veremos el primer caso de modelo.

Los sistemas de modelado proporcionan un lenguaje comprensible que permite al usuario expresar su problema centrando su esfuerzo en definirlo correctamente, dejando en manos del programa la conversión a otro formato compatible con las herramientas de resolución que forman parte del núcleo del sistema de modelado. Todas estas transformaciones del modelo descrito por el usuario producen otro problema o conjunto de problemas en un formato estándar semejantes a los que hemos visto en §3.3, y su mecanismo interno consiste básicamente en las técnicas que hemos estudiado en el capítulo 3, como pueden ser los cambios de variables, eliminar o añadir restricciones, utilizar variables de holgura, y otros recursos que podemos ampliar en [Boy04]. En el ejemplo de §5.5 veremos claramente que la definición del modelo en *CVX* es inmediata a partir del problema matemático, mientras que para utilizar directamente el *solver SeDuMi*, necesitamos modificar la formulación. Después de todas estas transformaciones, el sistema de modelado se encarga de elegir el *solver* más adecuado para el tipo de problema tratado. A partir de ahí, interpreta el resultado, indicando si es o no viable, y nos devuelve la solución óptima si es posible, adaptándola de nuevo al formato original.

En esta sección se describen las cuatro herramientas sobre las que he encontrado mayor número de referencias en la bibliografía que he consultado sobre optimización convexa y aplicaciones en telecomunicaciones: *CVX*, *AMPL*, *GAMS* y *YALIMP*. Si bien este criterio de selección no es demasiado riguroso, ya que quedan sin tratar herramientas de gran calado en el mercado como *AIMMS*, *TOMLAB*, *LINGO*, etc., lo que persigue es presentar una muestra significativa de las características de este tipo de programas, aunque limitada para no extender excesivamente el tamaño de esta memoria. En la tabla 5.4 encontramos un esquema de las herramientas tratadas en esta sección.

5.3.1 CVX

CVX se utiliza integrada como complemento de *Matlab*. El trabajo con esta herramienta se basa en lo que llaman *programación convexa disciplinada*, que consiste en un conjunto de reglas que aseguran que expresamos nuestro problema de forma convexa.

El programa aporta una biblioteca de funciones convexas que sirven de base para que el usuario construya las funciones y conjuntos del problema aplicando operaciones que conserven la convexidad, tal como vimos en §2.2.2 y §2.3.2.

Las restricciones y funciones objetivo que se construyen con estas reglas se transforman automáticamente en forma canónica, y a partir de ahí se resuelve con los programas de resolución.

También se pueden resolver problemas geométricos, ya que *CVX* los transforma a un formato convexo de acuerdo con el apartado §3.3.5, y una vez resuelto se devuelve el resultado en su formato original.

La información sobre *CVX* de este apartado procede del manual [Gra11].

5.3.1.1 Programas de resolución

Los programas de resolución que utiliza son *SDPT3* y *SeDuMi*.

5.3.1.2 Licencias

CVX se distribuye como *software* libre bajo los términos de GNU-GPL. Los derechos de copia están a nombre de Michael Grant y Stephen Boyd.

5.3.2 AMPL

AMPL es un lenguaje de programación para modelado algebraico en problemas de optimización lineales y no lineales, con variables discretas y continuas, y fue desarrollado en los laboratorios *Bell*.

Su utilización está orientada al desarrollo de prototipos y para trabajar en producción repetitiva. La información de esta herramienta procede de [AMP11, Fou02].

5.3.2.1 Características del lenguaje de modelado

Soporta la definición de conjuntos y operaciones de conjuntos. También se caracteriza por un tipo de sintaxis muy general y natural para describir expresiones aritméticas, lógicas y condicionales, así como para sumatorios y otros operadores de iteración.

Incluye características de programación no lineal como la posibilidad de especificar valores iniciales para los problemas primal y dual, funciones definidas por el usuario, diferenciación automática, y eliminación automática de variables definidas.

Se puede utilizar para problemas de redes, para lo que incluye indicaciones específicas para declarar nodos y arcos, y una sintaxis especial para funciones lineales por tramos.

5.3.2.2 Características del entorno de modelado

Se opera por medio de un entorno de comandos interactivo con opciones para procesos en segundo plano. Hay comandos que permiten visualizar cualquier componente o expresión del modelo, a través de la pantalla o escribiéndolo en un fichero que utiliza un formato automático o según las preferencias del usuario.

En *AMPL* los datos del problema están separados del modelo, por lo que éste contiene un sistema abstracto de variables, objetivos y restricciones que representa la forma general del problema a resolver, mientras que los datos se recopilan después de haber definido el modelo, creando un caso específico del problema. De esta forma, los modelos se mantienen de forma concisa incluso cuando los conjuntos y las tablas de datos aumentan.

Los modelos pueden incorporar muchos tipos de condiciones para validar los datos, y se incluyen interfaces con las herramientas de resolución.

5.3.2.3 Herramientas de resolución utilizadas

El entorno de *AMPL* permite la conexión con una gran cantidad de herramientas de resolución, y en muchos casos este *software* de enlace no supone un coste adicional..

- Para programación lineal
  - Variable continua: *BPMPD*, *Mosek*, y *LOQO*, entre otros.
  - Variable entera: *MINTO*, *LAMPS*, *lp_solve*, etc.
  - Redes: *CPLEX*, *OSL*
- Para programación no lineal
  - Cuadrática: *CPLEX*, *Mosek*, *OSL*
  - Convexa: *Mosek*, *SOPT*
  - Continua: *CONOPT*, *KNITRO*, *LANCELOT*, etc.
  - Entera: *MINLP*

5.3.2.4 Licencias

AMPL está sujeto a una licencia comercial, que se puede comprar para máquinas individuales o en forma de licencias flotantes, y se ofrecen para una gran cantidad de plataformas. El modo flotante se puede utilizar cuando

el número de usuarios es mayor que el de licencias, y se consigue con un servidor de autorizaciones en una red.

Se puede descargar una versión de evaluación en la que podemos utilizar hasta 300 variables y 300 restricciones y objetivos.

También se incluye la posibilidad de comprar a la misma empresa algunas de las herramientas de resolución como *Gurobi*, *CONOPT*, *KNITRO*, *SNOPT* y *MINOS*.

5.3.3 GAMS

Se trata de un sistema de modelado para programación matemática y optimización que consiste en un compilador de lenguaje y una serie de herramientas de resolución de alto rendimiento. La información de esta herramienta procede de [GAM11].

Este sistema es especialmente útil cuando el problema a resolver es de grandes dimensiones, y que pueden requerir muchas revisiones para establecer un modelo adecuado. Se puede encontrar en versiones para ordenadores personales y para grandes servidores.

GAMS facilita la descripción del problema haciendo que el modelado se haga de forma compacta y natural, permitiendo que el usuario pueda cambiar fácil y rápidamente la formulación, o la selección de uno u otro *solver*, o incluso se puede pasar de un modelo lineal a no lineal sin mayor problema.

5.3.3.1 Características del programa

El usuario se libera de tareas específicas del ordenador o servidor en que se instala, como por ejemplo calcular direcciones, asignaciones de almacenamiento, enlazado de subrutinas, o control de flujo. *GAMS* aumenta el tiempo disponible para conceptualizar y ejecutar el modelo, y analizar el resultado. Este sistema requiere especificaciones concisas y exactas sobre las entidades y sus relaciones. El lenguaje *GAMS* es formalmente similar a otros lenguajes de programación comunes, por lo que es familiar para cualquiera con experiencia en programación.

Los datos se introducen en forma de lista y de tabla. Los modelos se describen con sentencias algebraicas concisas fáciles de leer tanto para los usuarios como para las máquinas. Se pueden describir conjuntos completos de restricciones relacionadas en una sola orden, para que después *GAMS* genere automáticamente cada ecuación de restricción, permitiendo que el usuario añada excepciones cuando la generalidad no sea aplicable. Las órdenes se pueden reutilizar sin tener que cambiar el álgebra en otros problemas semejantes.

Se facilita especialmente el análisis de sensibilidad que vimos en §2.4.4. El usuario puede programar fácilmente un modelo para que se resuelva con

diferentes valores en algún elemento, y a continuación generar un listado de soluciones para cada caso.

Los modelos se desarrollan y documentan simultáneamente porque el usuario puede incluir textos explicativos como parte de la definición de cualquier símbolo o ecuación.

5.3.3.2 Herramientas de resolución utilizadas

GAMS incluye una gran cantidad de herramientas de resolución, como *CPLEX*, *Gurobi*, *Mosek*, entre otras. Estos *solvers* permiten resolver problemas de tipo LP, MIP, MIQCP[25], NLP[26], MINLP, CNS[27], MCP[28] o MPEC[29].

5.3.3.3 Licencias

GAMS se compra con licencia comercial para empresas, y hay precios diferentes para uso académico. Se puede obtener para un solo usuario o para múltiples máquinas, y el precio depende de la arquitectura y de los *solvers* que se deseen. Las licencias de estos últimos se tienen que comprar con el paquete *GAMS*.

Existe también una versión de demostración limitada a 300 restricciones y variables, con 2000 elementos distintos de cero, y con 50 variables discretas. En el caso de utilizar un *solver* de optimización global (no convexo) el número de variables y restricciones está limitado a 10.

5.3.4 YALIMP

YALIMP se utiliza como un complemento (*toolbox*) gratuito para *Matlab*, y sirve para modelar problemas de optimización convexos y no convexos.

El lenguaje es consistente con la sintaxis de *Matlab*, por lo que es muy fácil de aprender para usuarios familiarizados con este entorno. Implementa una gran cantidad de recursos de modelado, permitiendo que el usuario se concentre en el modelo a alto nivel, mientras que *YALIMP* se ocupa del modelado a bajo nivel para obtener modelos eficientes y numéricamente satisfactorios.

Soporta todos los tipos de problemas que hemos visto, lineales, cuadráticos, cónicos de orden 2, semidefinidos, geométricos, y otros más como por ejemplo los problemas cónicos con mezcla de variables enteras.

[25] *Mixed Integer Quadratically Constrained Program*
[26] *Non-Linear Program*
[27] *Constrained Non-Linear System*
[28] *Mixed Complementarity Problem*
[29] *Mathematical Program with Equilibrium Constraints*

5.3.4.1 Herramientas de resolución

Una de las ideas centrales de *YALIMP* es concentrarse en el lenguaje y en los algoritmos de alto nivel, dejando los verdaderos cálculos de optimización para los *solvers* externos. Sin embargo, también implementa algoritmos internos de optimización global no convexa, programación con mezcla de enteros, programación multiparamétrica, programación de suma de cuadrados y optimización robusta. Estos algoritmos se basan típicamente en el lenguaje de bajo nivel disponible en *YALIMP*, y resuelven subproblemas utilizando herramientas externas de resolución.

- Para programación lineal:
  - Con licencia libre: *CDD*, *GLPK*, *lp_solve*, *QSOPT*
  - Con licencia comercial o libre en versiones académicas: *BINTPROG*, *CPLEX*, *Gurobi*, `linprog` (de *Matlab Optimization Toolbox*), *Mosek*
- Para programación cuadrática:
  - Con licencia libre: *BPMPD*, *CLP*, *OOQP*, *QPC*
  - Con licencia comercial o libre en versiones académicas: *CPLEX*, *Mosek*, *NAG*, `quadprog` (de *Matlab Optimization Toolbox*), *Xpress*
- Para programación cónica de segundo orden:
  - Con licencia libre: *SDPT3*, *SeDuMi*
  - Con licencia comercial o libre en versiones académicas: *CPLEX*, *Mosek*
- Para programación semidefinida:
  - Con licencia libre: *CSDP*, *DSDP*, *SDPA*, *SDPLR*, *SDPT3*, *SeDuMi*
  - Con licencia comercial o libre en versiones académicas: *LMILAB*, *PENBMI*, *PENSDP*
- Para programación no lineal en general y otros *solvers*: `fmincon` (de *Matlab Optimization Toolbox*), *GPPOSY*, *IPOPT*, *KYPD*, *LMIRANK*, *MPT*, *PENNON*, *SNOPT*, *VSDP*

5.3.4.2 Licencias

La versión actual de *YALIMP* es libre de cargo y se distribuye de forma abierta. Sus autores únicamente piden que se mencione en una referencia si

se utiliza en un trabajo publicado. Por otro lado, no se puede redistribuir como parte de un producto comercial.

Esta disponibilidad implica que no cubre ningún servicio de garantía.

Nombre	Solvers	Licencia	Limitaciones en demo
CVX	*SeDuMi*, *SDPT3*	GPL	
AMPL	*BPMD*, *CPLEX*, *LAMPS*, *LOQO*, *lp_solve*, *MINOS*, *Mosek*, *OSL*, *SOPT*, *XA*, *Xpress-MP*, *MINTO*, *CONOPT*, *DONLP2*, *FILTER*, *FSQP*, *IPOPT*, *KNITRO*, *LANCELOT*, *NPSOL*, *PENNON*, *SNOPT*, *MINLP*[23], *Gurobi*	comercial	300 Variables y 300 restricciones y objetivos
GAMS	*ALPHAECP*, *BARON*, *BDMLP*, *COIN-OR*, *CPLEX*, *DECIS*, *DICOPT*, *Gurobi*, *KNITRO*, *LGO*[24], *LINDOGLOBAL*, *LOGMIP*, *MILES*, *MINOS*, *Mosek*, *MSNLP*, *NLPEC*, *OQNLP*, *PATH*, *SBB*, *SCIP*, *SNOPT*, *XA*, *Xpress*	comercial	300 restricciones y variables, con 2000 elementos distintos de 0, 50 variables discretas. En caso de utilizar un solver global está limitado a 10 restricciones y variables
YALIMP	*CDD*, *GLPK*, *lp_solve*, *QSOPT*, *BINTPROG*, *CPLEX*, *Gurobi*, `linprog`, *Mosek*, *BPMPD*, *CLP*, *OOQP*, *QPC*, *NAG*, `quadprog`, *Xpress*, *CSDP*, *SDPT3*, *SeDuMi*, *DSDP*, *SDPA*, *SDPLR*, *LMILAB*, *PENBMI*, *PENSDP*, `fmincon`, *GPPOSY*, *IPOPT*, *KYPD*, *LMIRANK*, *MPT*, *PENNON*, *SNOPT*, *VSDP*,	libre	

Tabla 5.4: Resumen de las herramientas de modelado.

5.4 Herramientas para resolver problemas de optimización global (convexos y no convexos)

En esta sección se estudian tres herramientas orientadas a problemas de optimización global que no necesariamente tienen que ser convexos. Por su grado de aceptación en el entorno universitario, he seleccionado *Matlab* y *Mathematica*. En el primer caso, para resolver este tipo de problemas puede incluir el complemento *Matlab Optimization Toolbox*. En el segundo, *Mathematica* incluye funciones de optimización, aunque también he encontrado un complemento especializado, *MathOptimizer Professional* que analizaremos en §5.4.3. De forma esquemática, podemos ver las características de cada paquete en la tabla 5.5.

5.4.1 Matlab Optimization Toolbox

Las características generales de *Matlab* son ampliamente conocidas en el ámbito de estudio de ingeniería de telecomunicaciones, por haberlo utilizado en una gran cantidad de asignaturas y prácticas de laboratorio. He incluido aquí la revisión del complemento de optimización que da título a este apartado, que como ya he indicado, no requiere que el problema a resolver sea convexo.

De acuerdo con [Mat10], en esta herramienta se distinguen cuatro grupos de funciones de optimización:

- *Minimización escalar*. Encontramos aquí las herramientas de resolución de problemas lineales, cuadráticos, y no lineales en general.
- *Minimización vectorial o multiobjetivo*. Estos comandos utilizan escalarización para resolver problemas con varios objetivos.
- *Resolución de ecuaciones*. Este grupo de comandos intenta resolver ecuaciones y sistemas de ecuaciones lineales y no lineales en torno a un punto inicial. Es equivalente a encontrar la norma de la función a minimizar en torno al punto inicial. Se trata de un problema convexo.
- *Mínimos cuadrados y ajuste de curvas*. Se incluyen comandos para resolver este tipo de problemas que en realidad también son convexos.

En los problemas convexos nos interesa utilizar el comando `fmincon`, en el que podemos especificar el uso de algoritmos de punto interior.

5.4.1.1 Licencias

Este complemento se distribuye bajo licencia comercial en formatos individual, de grupo, concurrente y para aulas. El precio de la licencia hay que añadirlo a la adquisición de *Matlab*, igual que en el resto de las herramientas que hemos visto funcionando como complemento o *toolbox*.

Nombre	Algoritmos	Problemas	Licencia	Limitaciones en demo
Mathematica	símplex p. interior Nelder-Mead algoritmos genéticos temple simulado	LP op. local op. global	comercial	15 días
MathOptimizer	Lipschitz Global Opt.	op. global	comercial	por solicitud
Matlab Optimization Toolbox	p. interior Nelder-Mead Quasi Newton Región de confianza	op. local op. global	comercial	para empresas

Tabla 5.5: Resumen de las herramientas de resolución no convexa.

También es posible conseguir una versión de prueba, aunque esta opción no está disponible para estudiantes.

5.4.2 Mathematica

Mathematica es una plataforma de cálculo simbólico y numérico cuyo rango de aplicaciones es mucho más amplio del que se utiliza en este proyecto. Está compuesta por un núcleo que interpreta expresiones y devuelve los resultados, y un sistema de gestión de entrada de datos que permite la creación y edición de documentos que contienen código en un formato que visualmente se asemeja a la formulación matemática de un editor de texto. En cuanto a sus capacidades de optimización, en esta herramienta no se requiere que el problema a optimizar sea convexo. Simplemente se distingue entre problemas lineales y no lineales. En el primer caso se utiliza el comando `LinearProgramming`. Cuando el problema no es lineal, podemos buscar una solución global con el comando `Minimize` o una solución local con `FindMinimum`.

Las restricciones se describen como una combinación booleana de ecuaciones que pueden ser igualdades, desigualdades estrictas y no estrictas, y también declaraciones de variables enteras $x \in \mathbb{Z}$

Los algoritmos de punto interior se encuentran en lo que este *software* clasifica como optimización local, porque efectivamente, si aplicamos las técnicas que hemos visto en el tema 4 en funciones no convexas, la solución puede converger hacia un mínimo local.

La información sobre esta herramienta procede de [Wol10].

5.4.2.1 Licencias

Wolfram Mathematica se puede conseguir bajo licencia comercial. La versión actual es 8. Su venta se clasifica según vaya a ser su utilización, bien para industria, gobierno o educación. En el primer caso encontramos licencias individuales o de grupo y de empresa; igual que en el caso gubernamental; y en el caso de uso académico, además de las licencias individuales, para departamentos y entidades docentes, también hay licencias para estudiantes a precios más asequibles. Se ofrecen diferentes precios para investigadores y docentes de educación superior, para escuelas técnicas y para educación primaria y secundaria.

Existe una versión de evaluación completamente funcional durante 15 días.

5.4.3 MathOptimizer Professional

Esta herramienta funciona como complemento para *Mathematica*, y utiliza como herramienta de resolución el optimizador global de Lipschitz (*LGO*), que permite la solución global y local de problemas de optimización continuos en general. Los datos sobre esta herramienta proceden de [Pin10] y de información directa enviada por su autor, según la cual, este *software* puede resolver modelos con algunos miles de variables y funciones objetivo y restricciones, y este límite depende de la infraestructura de *hardware* utilizada y del tiempo de ejecución que podamos tolerar. En pruebas numéricas y en algunas aplicaciones para clientes se han resuelto algunos problemas complejos con estas dimensiones. Sin embargo, dependiendo de cada tipo concreto de problema a resolver, la variación del tiempo de resolución puede abarcar desde minutos hasta algunos días. En algunos casos simplemente la evaluación de las funciones ya puede requerir bastante tiempo, por lo que la optimización global aplicada tiene que adaptarse para manejar esta situación.

A partir de las expresiones para *Mathematica*, esta herramienta es capaz de generar automáticamente código en C o en Fortran.

La implementación del solver *LGO* integra las siguientes estrategias:

- Búsqueda global basada en el método de ramificar y acotar, con partición adaptativa y muestreo.
- Búsqueda global estocástica adaptativa.
- Búsqueda global estocástica y adaptativa basada en inicio múltiple.
- Búsqueda local restringida por acotación, basada en el uso de una función exacta de penalización.
- Búsqueda local con restricciones, basada en una aproximación de gradiente reducido generalizado.

- Entrega inmediata de informes de resultados en el documento de *Mathematica*.

5.4.3.1 Licencias

Se puede comprar bajo licencia comercial desde la *web* de Wolfram, en modalidades estándar, gubernamental, académica y para estudiantes. También es posible una versión de evaluación después de contactar directamente con su autor.

5.5 Ejemplo

El objetivo de este apartado es profundizar más en el modo de utilización de algunas de las herramientas que hemos visto. Como sistema de modelado he elegido *CVX* y como *solvers Mosek* y *SeDuMi*. He seleccionado estas herramientas por ser las más citadas en la bibliografía que he consultado. También he utilizado dos herramientas de optimización globales *Mathematica* y *Matlab Optimization Toolbox* por su accesibilidad desde el entorno académico. En la exposición de esta sección hay que tener en cuenta que todas las aplicaciones utilizadas exceptuando *Mathematica* se ejecutan desde *Matlab*, y que el código descrito corresponde únicamente a la parte específica de optimización. Para comprobar el resto de comandos hay que consultar el CD[30] adjunto a este documento.

El ejemplo consiste en un problema cuadrático sobre $\mathbb{R}^2$, con dos restricciones cuadráticas, fácil de visualizar gráficamente, que consiste en resolver lo siguiente:

$$\begin{aligned} \underset{\boldsymbol{x}}{\text{minimizar}} \quad & \tfrac{1}{2}\boldsymbol{x}^T\boldsymbol{P}_0\boldsymbol{x} + \boldsymbol{q}_0^T\boldsymbol{x} + r_0 \\ \text{sujeto a} \quad & \tfrac{1}{2}\boldsymbol{x}^T\boldsymbol{P}_1\boldsymbol{x} + \boldsymbol{q}_1^T\boldsymbol{x} + r_1 \leq 0 \\ & \tfrac{1}{2}\boldsymbol{x}^T\boldsymbol{P}_2\boldsymbol{x} + \boldsymbol{q}_2^T\boldsymbol{x} + r_2 \leq 0 \end{aligned} \tag{5.1}$$

Asignamos valores arbitrarios a los parámetros de las ecuaciones en *Matlab*, de forma que las curvas de nivel de la función principal y las elipsoides de las restricciones correspondientes a las tres ecuaciones de (5.1) quedan dispuestas tal como aparecen en la figura 5.1. Para calcular los parámetros de las elipsoides partimos de la descripción del apartado §2.2.1.3. Elegimos un ángulo de inclinación para orientar la curva, con lo que se definen los autovectores de la matriz $\boldsymbol{P}$; asignamos los autovalores para dar el tamaño a los ejes, que serán proporcionales a la raíz cuadrada de estos autovalores; y situamos el punto central.

También he marcado en esta figura la solución óptima de nuestro problema con un punto rojo, y en la figura 5.2 tenemos una representación tridimensional de las formas cuadráticas del problema.

[30] *Compact Disc*

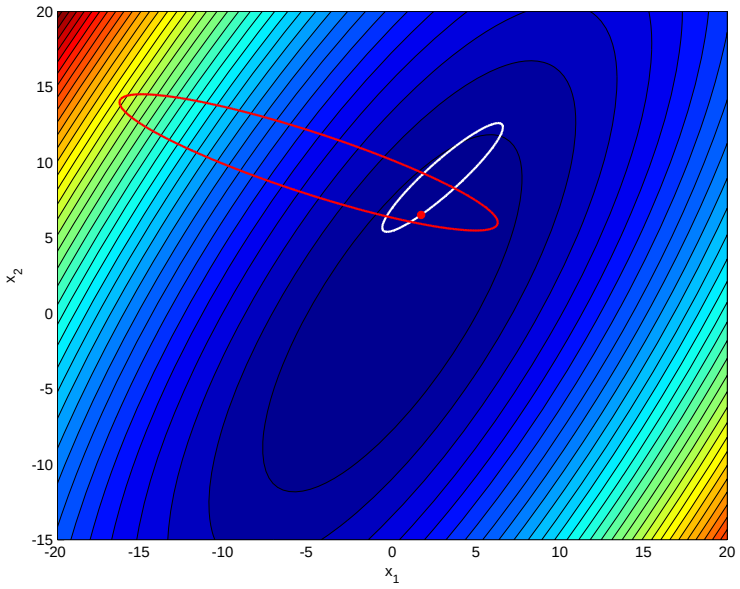

Figura 5.1: Ejemplo de problema QCQP. Curvas de nivel

5.5.1 Resolución por *CVX*

Una vez asignados los valores de $\boldsymbol{P}_0$, $\boldsymbol{q}_0$, r_0, $\boldsymbol{P}_1$, $\boldsymbol{q}_1$, r_1, $\boldsymbol{P}_2$, $\boldsymbol{q}_2$ y r_2, que son los parámetros de nuestro problema, el código del modelo más sencillo que necesitamos construir utilizando *CVX* es el siguiente:

```
cvx_begin
variable x(2)
minimize(0.5*x'*P0*x+q0'*x+r0)
subject to
    0.5*x'*P1*x+q1'*x+r1<=0;
    0.5*x'*P2*x+q2'*x+r2<=0;
cvx_end
```

Prácticamente es una transcripción directa del enunciado del problema. El resultado obtenido es el siguiente:

```
x_cvx =
    1.7155
    6.5185
```

También podemos guardar las variables duales correspondientes a cada desigualdad con una ligera modificación en el código anterior

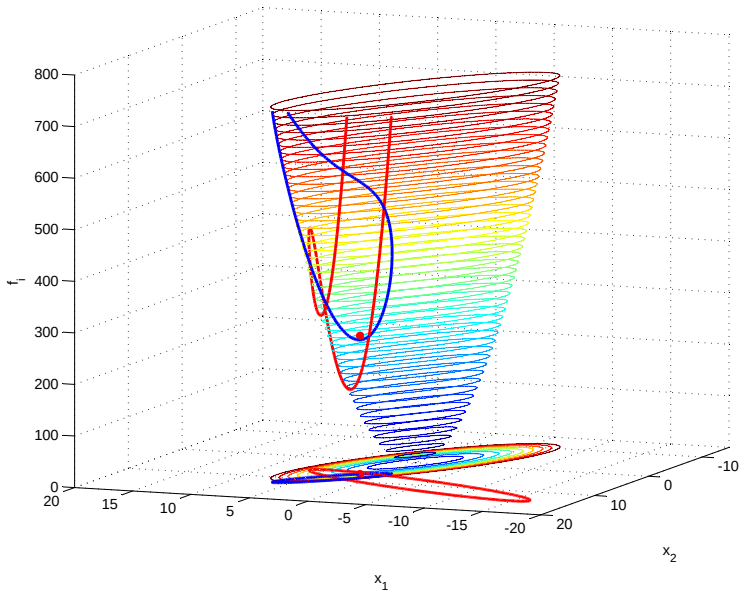

Figura 5.2: Representación tridimensional de la superficie optimizada y las elipses que delimitan la región viable

```
1      cvx_begin
2          variable x(2);
3          dual variable y1;
4          dual variable y2;
5          minimize(0.5*x'*P0*x+q0'*x+r0)
6          subject to
7              y1: 0.5*x'*P1*x+q1'*x+r1<=0;
8              y2: 0.5*x'*P2*x+q2'*x+r2<=0;
9      cvx_end
```

En la figura 5.1, la elipse blanca corresponde a la primera restricción, y la roja a la segunda. Revisamos la teoría de §2.4.3.1, y en este ejemplo confirmamos que al estar activa la primera restricción $f_1(\boldsymbol{x}^*) = 0$ en la solución $\boldsymbol{x}^*$, su variable dual es positiva $y_1 = 3{,}2$. Por otro lado, la elipse roja no toca al punto óptimo, por lo que $f_2(\boldsymbol{x}^*) < 0$ y su variable dual es igual a cero.

```
1  >> [y1; y2]
2  ans =
3      3.2317
4      0.0000
```

Este ejemplo también nos sirve para revisar la teoría de sensibilidad

frente a perturbaciones de §2.4.4, porque podemos ver claramente que una pequeña variación de la segunda restricción no altera nuestro resultado, es decir, la elipse roja puede hacerse un poco más pequeña sin variar la solución. Sin embargo, al variar la primera restricción, el punto óptimo será distinto, y la sensibilidad frente a pequeños cambios en esta elipse se mide por el valor de y_1.

5.5.2 Resolución por *Mosek*

Para utilizar *Mosek* tenemos que complicarnos un poco. La formulación del problema debe seguir el siguiente formato, para mantener la nomenclatura utilizada en este programa:

$$\begin{array}{ll} \underset{\boldsymbol{x}}{\text{minimizar}} & 1/2\boldsymbol{x}^T\boldsymbol{Q}_0\boldsymbol{x} + \boldsymbol{c}^T\boldsymbol{x} \\ \text{sujeto a} & 1/2\boldsymbol{x}^T\boldsymbol{Q}_{c1}\boldsymbol{x} + \boldsymbol{a}_1^T\boldsymbol{x} \leq u_1^c \\ & 1/2\boldsymbol{x}^T\boldsymbol{Q}_{c2}\boldsymbol{x} + \boldsymbol{a}_2^T\boldsymbol{x} \leq u_2^c \end{array}$$

Por tanto, $\boldsymbol{Q}_0$ es equivalente a $\boldsymbol{P}_0$, y las matrices $\boldsymbol{Q}_{c1}$ y $\boldsymbol{Q}_{c2}$ se introducen en formato de tres dimensiones con índices `i`,`j` y `k`, correspondiendo el índice `k` a cada restricción, y los otros dos a las filas y columnas de $\boldsymbol{P}_1$ y $\boldsymbol{P}_2$. Para introducir los valores de estas matrices tenemos que hacerlo de forma dispersa, es decir, por medio de listas de índices y valores. Además, como tienen que ser simétricas sólo hace falta indicar los valores de la submatriz triangular de cada una.

Los vectores $\boldsymbol{a}_1$ y $\boldsymbol{a}_2$ se tienen que incluir como filas de una sola matriz que contiene los parámetros $\boldsymbol{q}_1$ y $\boldsymbol{q}_2$ del problema (5.1) en el siguiente formato:

$$\boldsymbol{a} = \begin{bmatrix} \boldsymbol{q}_1^T \\ \boldsymbol{q}_2^T \end{bmatrix}$$

Del mismo modo los parámetros r_1 y r_2 tienen que formar el vector $\boldsymbol{u}_c$

$$\boldsymbol{u}_c = \begin{bmatrix} -r_1 \\ -r_2 \end{bmatrix}$$

Por último, r_0 es una constante de la función de optimización que no afecta al resultado, por lo que *Mosek* no la tiene en cuenta. Todos los parámetros que hemos identificado se agrupan como campos de la estructura `prob` tal como vemos en el siguiente código

```
clear prob;
 % Función objetivo
prob.qosubi    = [   1       1       2   ]';
prob.qosubj    = [     1       2       2 ]';
prob.qoval     = [P0(1,1) P0(1,2) P0(2,2)]';
prob.c = q0;
 % Restricciones cuadráticas
prob.qcsubi = [   1        1       2       1       1       2   ]';
prob.qcsubj = [     1        2       2       1       2       2 ]';
prob.qcsubk = [ 1        1       1       2       2       2     ]';
prob.qcval  = [P1(1,1)  P1(1,2) P1(2,2) P2(1,1) P2(1,2) P2(2,2)]';
prob.a = sparse([q1';q2']);
prob.buc=[-r1; -r2];     % Límite superior
[r,res] = mosekopt ('minimize', prob );
x_mosek = res.sol.itr.xx
```

Para facilitar la comprensión del código, he alineado los índices correspondientes de cada matriz guardada en formato disperso, así, por ejemplo, el primer elemento de `prob.qcsubi` está alineado verticalmente con el primer índice de `P1(1,1)`, el primer elemento de `prob.qcsubj` con el segundo índice de `P1(1,1)`, y el primer elemento de `prob.qcsubk` con el `1` de `P1(1,1)`. Al ejecutarlo obtenemos prácticamente el mismo resultado:

```
>> res.sol.itr.xx
ans =
    1.7160
    6.5190
```

5.5.3 Resolución por *SeDuMi*

Para resolver el problema (5.1) con *SeDuMi* tenemos que transformarlo en un problema cónico. Para entender mejor las transformaciones vuelvo a escribir el problema original:

$$\begin{aligned} \underset{\boldsymbol{x}}{\text{minimizar}} \quad & \tfrac{1}{2}\,\boldsymbol{x}^T\boldsymbol{P}_0\boldsymbol{x} + \boldsymbol{q}_0^T\boldsymbol{x} + r_0 \\ \text{sujeto a} \quad & \tfrac{1}{2}\,\boldsymbol{x}^T\boldsymbol{P}_1\boldsymbol{x} + \boldsymbol{q}_1^T\boldsymbol{x} + r_1 \leq 0 \\ & \tfrac{1}{2}\,\boldsymbol{x}^T\boldsymbol{P}_2\boldsymbol{x} + \boldsymbol{q}_2^T\boldsymbol{x} + r_2 \leq 0 \end{aligned}$$

El primer paso es convertir las ecuaciones cuadráticas en cónicas. Para ello, basándome en un ejemplo de [Lob98] utilizamos la ecuación (5.2) que se obtiene al desarrollar en forma polinomial los términos de la norma cuadrática

en $\mathbb{R}^2$.

$$\begin{array}{c} \frac{1}{2}\,\|\boldsymbol{P}_i^{1/2}\boldsymbol{x}+\boldsymbol{P}_i^{-1/2}\boldsymbol{q}_i\|_2^2=\frac{1}{2}\,\boldsymbol{x}^T\boldsymbol{P}_i\boldsymbol{x}+\boldsymbol{q}_i^T\boldsymbol{x}+\frac{1}{2}\,\boldsymbol{q}_i^T\boldsymbol{P}_i^{-1}\boldsymbol{q}_i \\ \Updownarrow \\ \frac{1}{2}\,\boldsymbol{x}^T\boldsymbol{P}_i\boldsymbol{x}+\boldsymbol{q}_i^T\boldsymbol{x}+r_i=\frac{1}{2}\,\|\boldsymbol{P}_i^{1/2}\boldsymbol{x}+\boldsymbol{P}_i^{-1/2}\boldsymbol{q}_i\|_2^2+r_i-\frac{1}{2}\,\boldsymbol{q}_i^T\boldsymbol{P}_i^{-1}\boldsymbol{q}_i \end{array} \tag{5.2}$$

Ahora modificamos el problema utilizando la nueva expresión

$$\begin{array}{ll} \underset{\boldsymbol{x}}{\text{minimizar}} & \|\boldsymbol{P}_0^{1/2}\boldsymbol{x}+\boldsymbol{P}_0^{-1/2}\boldsymbol{q}_0\|_2 \\ \text{sujeto a} & \frac{1}{2}\,\|\boldsymbol{P}_1^{1/2}\boldsymbol{x}+\boldsymbol{P}_1^{-1/2}\boldsymbol{q}_1\|_2^2+r_1-\frac{1}{2}\,\boldsymbol{q}_1^T\boldsymbol{P}_1\boldsymbol{q}_1\leq 0 \\ & \frac{1}{2}\,\|\boldsymbol{P}_2^{1/2}\boldsymbol{x}+\boldsymbol{P}_2^{-1/2}\boldsymbol{q}_2\|_2^2+r_2-\frac{1}{2}\,\boldsymbol{q}_2^T\boldsymbol{P}_2\boldsymbol{q}_2\leq 0 \end{array}$$

El siguiente paso es utilizar el planteamiento por epígrafo, tal como veíamos en §3.1.7:

$$\begin{array}{ll} \underset{\boldsymbol{x},t}{\text{minimizar}} & t \\ \text{sujeto a} & \|\boldsymbol{P}_0^{1/2}\boldsymbol{x}+\boldsymbol{P}_0^{-1/2}\boldsymbol{q}_0\|_2\leq t \\ & \|\boldsymbol{P}_1^{1/2}\boldsymbol{x}+\boldsymbol{P}_1^{-1/2}\boldsymbol{q}_1\|_2\leq\left(\boldsymbol{q}_1^T\boldsymbol{P}_1\boldsymbol{q}_1-2r_1\right)^{1/2} \\ & \|\boldsymbol{P}_2^{1/2}\boldsymbol{x}+\boldsymbol{P}_2^{-1/2}\boldsymbol{q}_2\|_2\leq\left(\boldsymbol{q}_2^T\boldsymbol{P}_2\boldsymbol{q}_2-2r_2\right)^{1/2} \end{array} \tag{5.3}$$

Ahora que ya tenemos un problema cónico de segundo orden, tenemos que intentar convertirlo al formato estándar que necesita *SeDuMi*, y que, de acuerdo con el manual [Stu98], consiste en lo siguiente:

$$\begin{array}{ll} \underset{\boldsymbol{z}}{\text{minimizar}} & \boldsymbol{c}_s^T\boldsymbol{z} \\ \text{sujeto a} & \boldsymbol{A}_s\boldsymbol{z}=\boldsymbol{b}_s \\ & \boldsymbol{z}\in\mathcal{K}_s \end{array} \tag{5.4}$$

El comando a utilizar es `sedumi(As,bs,cs,Ks);`, donde la matriz `As` tiene que ser dispersa. El parámetro `Ks` es una estructura que utilizaremos para describir el cono de segundo orden como veremos más adelante. En nuestro caso únicamente nos interesan dos campos de esta estructura:

- `Ks.f` es el número de variables libres. Esto quiere decir que si por ejemplo `Ks.f=2` las dos primeras componentes de $\boldsymbol{z}$ pueden tener cualquier valor en $\mathbb{R}$.

- `Ks.q` es una lista de dimensiones de conos de segundo orden. En nuestro caso, teniendo en cuenta que las dos primeras componentes son libres, `Ks.q=[3 3 3]` quiere decir que

```
z(3)>=norm(z(4:5))
z(6)>=norm(z(7:8))
z(9)>=norm(z(10:11))
```

Ahora construimos nuestro vector $\boldsymbol{z}$ utilizando variables auxiliares que conserven el orden que acabamos de ver, de forma que el problema (5.3) se transforma en el siguiente:

$$\begin{array}{lrl} \underset{\boldsymbol{z},t}{\text{minimizar}} & t & \\ \text{sujeto a} & \boldsymbol{A}_s\boldsymbol{z} & = \boldsymbol{b}_s \\ & t & \geq \|\boldsymbol{a}_0\|_2 \\ & b_1 & \geq \|\boldsymbol{a}_1\|_2 \\ & b_2 & \geq \|\boldsymbol{a}_2\|_2 \end{array} \tag{5.5}$$

Las variables que hemos añadido en (5.5) sustituyen a las expresiones de (5.3) de acuerdo con lo siguiente:

$$\begin{aligned} \boldsymbol{a}_0 &= \boldsymbol{P}_0^{1/2}\boldsymbol{x} + \boldsymbol{P}_0^{-1/2}\boldsymbol{q}_0 \\ \boldsymbol{a}_1 &= \boldsymbol{P}_1^{1/2}\boldsymbol{x} + \boldsymbol{P}_1^{-1/2}\boldsymbol{q}_1 \\ \boldsymbol{a}_2 &= \boldsymbol{P}_2^{1/2}\boldsymbol{x} + \boldsymbol{P}_2^{-1/2}\boldsymbol{q}_2 \\ b_1 &= \left(\boldsymbol{q}_1^T\boldsymbol{P}_1^{-1}\boldsymbol{q}_1 - 2r_1\right)^{1/2} \\ b_2 &= \left(\boldsymbol{q}_2^T\boldsymbol{P}_2^{-1}\boldsymbol{q}_2 - 2r_2\right)^{1/2} \end{aligned} \tag{5.6}$$

Ya podemos construir nuestra variable de optimización en el orden que entiende *SeDuMi* según (5.5):

$$\boldsymbol{z} = \begin{bmatrix} \boldsymbol{x}^T & t & \boldsymbol{a}_0^T & b_1 & \boldsymbol{a}_1^T & b_2 & \boldsymbol{a}_2^T \end{bmatrix}^T$$

donde los conos de segundo orden son los siguientes

$$\begin{aligned} \mathcal{K}_0 &= \{(t, \boldsymbol{a}_0), \mid t \geq \|\boldsymbol{a}_0\|\} & t \in \mathbb{R}, \boldsymbol{a}_0 \in \mathbb{R}^2, \mathcal{K}_0 \subset \mathbb{R}^3 \\ \mathcal{K}_1 &= \{(b_1, \boldsymbol{a}_1), \mid b_1 \geq \|\boldsymbol{a}_1\|\} & b_1 \in \mathbb{R}, \boldsymbol{a}_1 \in \mathbb{R}^2, \mathcal{K}_1 \subset \mathbb{R}^3 \\ \mathcal{K}_2 &= \{(b_2, \boldsymbol{a}_2), \mid b_2 \geq \|\boldsymbol{a}_2\|\} & b_2 \in \mathbb{R}, \boldsymbol{a}_2 \in \mathbb{R}^2, \mathcal{K}_2 \subset \mathbb{R}^3 \end{aligned}$$

Ya sólo queda construir $\boldsymbol{A}_s$ y $\boldsymbol{b}_s$ de forma que las ecuaciones de (5.6) se puedan expresar como $\boldsymbol{A}_s\boldsymbol{z} = \boldsymbol{b}_s$, y también construir $\boldsymbol{c}_s$ de forma que $\boldsymbol{c}_s^T\boldsymbol{z} = t$ para que el problema (5.5) quede expresado de forma estándar

(5.4).

$$\underbrace{\begin{bmatrix} P_{0\,11}^{\frac{1}{2}} & P_{0\,12}^{\frac{1}{2}} & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ P_{0\,21}^{\frac{1}{2}} & P_{0\,22}^{\frac{1}{2}} & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ P_{1\,11}^{\frac{1}{2}} & P_{1\,12}^{\frac{1}{2}} & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\ P_{1\,21}^{\frac{1}{2}} & P_{1\,22}^{\frac{1}{2}} & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\ P_{2\,11}^{\frac{1}{2}} & P_{2\,12}^{\frac{1}{2}} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 \\ P_{2\,21}^{\frac{1}{2}} & P_{2\,22}^{\frac{1}{2}} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}}_{\boldsymbol{A}_s} \underbrace{\begin{bmatrix} x_{11} \\ x_{12} \\ t \\ a_{01} \\ a_{02} \\ b_1 \\ a_{11} \\ a_{12} \\ b_2 \\ a_{21} \\ a_{22} \end{bmatrix}}_{\boldsymbol{z}} = \underbrace{\begin{bmatrix} (-P_0^{-\frac{1}{2}} q_0)_1 \\ (-P_0^{-\frac{1}{2}} q_0)_2 \\ (-P_1^{-\frac{1}{2}} q_1)_1 \\ (-P_1^{-\frac{1}{2}} q_1)_2 \\ (-P_2^{-\frac{1}{2}} q_2)_1 \\ (-P_2^{-\frac{1}{2}} q_2)_2 \\ \left(q_1^T P_1 q_1 - 2r_1\right)^{\frac{1}{2}} \\ \left(q_2^T P_2 q_2 - 2r_2\right)^{\frac{1}{2}} \end{bmatrix}}_{\boldsymbol{b}_s}$$

$$\boldsymbol{c}_s^T \boldsymbol{z} = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ t \\ a_{01} \\ a_{02} \\ b_1 \\ a_{11} \\ a_{12} \\ b_2 \\ a_{21} \\ a_{22} \end{bmatrix}$$

El código en *Matlab* queda como sigue:

```
clear As; clear Ks;
As=[[P0^.5;P1^.5;P2^.5;zeros(2,2)],zeros(8,9)];
As(1,4)=-1;     As(2,5)=-1;     As(3,7)=-1;     As(4,8)=-1;
As(5,10)=-1;    As(6,11)=-1;    As(7,6)=1;      As(8,9)=1;
bs=[-P0^-.5*q0;-P1^-.5*q1;-P2^-.5*q2;...
    (q1'*inv(P1)*q1-2*r1)^.5;(q2'*inv(P2)*q2-2*r2)^.5];
cs=[0;0;1;zeros(8,1)];
Ks.f=2; Ks.q=[3 3 3];
z=sedumi(sparse(As),bs,cs,Ks);
x_sdm=full(z(1:2))
```

Observemos que el primer argumento de `sedumi` tiene que ser una matriz dispersa, por lo que hay que convertirla con el comando `sparse(As)`. Al ejecutarlo, como era de esperar, obtenemos el mismo resultado que con las otras herramientas.

```
>> x_sdm
ans =
    1.7162
    6.5191
```

5.5.4 Resolución por *Mathematica*

La introducción de datos en *Mathematica* es más intuitiva, ya que es una herramienta con un enfoque muy orientado a la documentación, tal como veíamos en §5.4.2. Con esto me refiero por ejemplo a la visualización de subíndices y superíndices en las variables, a la representación de fracciones y matrices tal como las escribimos en cualquier apunte de matemáticas, o a la posibilidad de utilizar caracteres griegos.

El comando que emplearemos es `FindMinimum`. En las primeras versiones de este ejemplo, me encontré con que este optimizador devolvía un error indicando que no admitía funciones de variable vectorial. Más tarde, buscando en la ayuda del programa, llegué a un ejemplo en el que sí se podía utilizar. Para ello era necesario crear un vector con componentes simbólicas antes de llamar a la función de optimización. De este modo he podido aplicar esta herramienta a los ejemplos del tema 6. No obstante, en este capítulo he mantenido la ejecución inicial con la definición individual de cada componente, `f[x_,y_]` para poder representar la gráfica de la figura 5.3.

Teniendo esto en cuenta, la función objetivo y las restricciones del problema (5.1) se escriben en *Mathematica* del siguiente modo:

$$\texttt{f}_0\texttt{[x\_,y\_]:=}\frac{1}{2}\begin{pmatrix}\texttt{x} & \texttt{y}\end{pmatrix}\texttt{.P}_0\texttt{.}\begin{pmatrix}\texttt{x}\\ \texttt{y}\end{pmatrix}\texttt{ + q}_0^{\texttt{T}}\texttt{.}\begin{pmatrix}\texttt{x}\\ \texttt{y}\end{pmatrix}\texttt{ + r}_0\texttt{;}$$

$$\texttt{f}_1\texttt{[x\_,y\_]:=}\frac{1}{2}\begin{pmatrix}\texttt{x} & \texttt{y}\end{pmatrix}\texttt{.P}_1\texttt{.}\begin{pmatrix}\texttt{x}\\ \texttt{y}\end{pmatrix}\texttt{ + q}_1^{\texttt{T}}\texttt{.}\begin{pmatrix}\texttt{x}\\ \texttt{y}\end{pmatrix}\texttt{ + r}_1\texttt{;}$$

$$\texttt{f}_2\texttt{[x\_,y\_]:=}\frac{1}{2}\begin{pmatrix}\texttt{x} & \texttt{y}\end{pmatrix}\texttt{.P}_2\texttt{.}\begin{pmatrix}\texttt{x}\\ \texttt{y}\end{pmatrix}\texttt{ + q}_2^{\texttt{T}}\texttt{.}\begin{pmatrix}\texttt{x}\\ \texttt{y}\end{pmatrix}\texttt{ + r}_2\texttt{;}$$

Ahora podemos llamar a la función de optimización, y podemos especificar el algoritmo a utilizar:

```
FindMinimum[{f0[x,y][[1,1]], f1[x,y][[1,1]]≤0 &&
f2[x,y][[1,1]]≤0}, {x,y}, Method→‘InteriorPoint"]
```

Y obtendremos como respuesta

```
{282.399, {x→1.71604, y→6.51897}}
```

También podemos utilizar otras opciones para inicializar el algoritmo en el punto que queramos y visualizar el camino seguido en cada iteración del

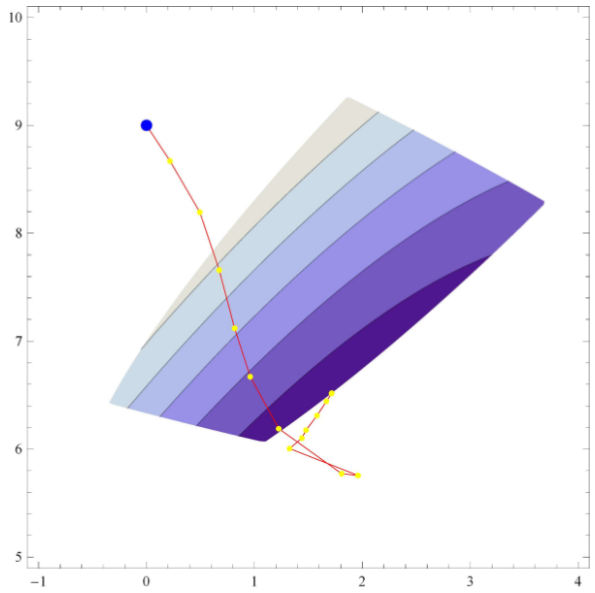

Figura 5.3: Ejemplo de problema QCQP resuelto con *Mathematica*

proceso de optimización. En la figura 5.3 inicializamos el problema desde un punto no viable. El comando `ContourPlot` nos permite utilizar la opción `RegionFunction` para dibujar las curvas de nivel de acuerdo con las restricciones, de modo que únicamente veremos la zona viable.

```
ContourPlot[f0[x,y][[1,1]], {x,-5,5}, {y,0,5},
RegionFunction→Function[{x,y},f1[x,y][[1,1]]≤0 &&
f2[x,y][[1,1]]≤0]]
```

5.5.5 Resolución por *Matlab Optimization Toolbox*

Tal como indicábamos en §5.4.1, utilizaremos el comando `fmincon` para resolver problemas no lineales con restricciones no lineales. El código es tan sencillo como lo siguiente:

```
1    options = optimset('Algorithm','interior-point');
2    [x_mot, fval]=fmincon(@f0,[5; 5],[],[],[],[],[],[],...
3                          @restricciones,options);
```

Claro que tenemos que definir en otro fichero las funciones `f0` y `restricciones`. Ambas funciones tienen que tener como único argumento de entrada el vector `x`. Por tanto, para pasarle los parámetros de las ecuaciones cuadráticas, he utilizado una variable global, `elipsoides`, que contiene como campos las matrices $\boldsymbol{P}_0$, $\boldsymbol{P}_1$ y $\boldsymbol{P}_2$, y también los vectores $\boldsymbol{q}_0$, $\boldsymbol{q}_1$ y $\boldsymbol{q}_2$ junto con los escalares r_0, r_1 y r_2.

La función objetivo estará contenida en el fichero `f0.m`

```
function [f] = f0(x)
 % Función objetivo
global elipsoides;  % Parámetros de las formas cuadráticas
P0 = elipsoides.P0; q0 = elipsoides.q0; r0 = elipsoides.r0;

 % Devolvemos el valor de la función
f = 0.5*x'*P0*x+q0'*x+r0;
```

El fichero *restricciones.m* sirve, como su nombre indica, para llamar a las funciones de restricción. En el parámetro `c` se devuelven las desigualdades no lineales ($f_1(\boldsymbol{x}) \leq 0$ y $f_2(\boldsymbol{x}) \leq 0$), y en el parámetro `ceq` las igualdades no lineales (en nuestro ejemplo no tenemos).

```
function [c ceq] = restricciones(x)
global elipsoides;  % Parámetros de las formas cuadráticas
P1 = elipsoides.P1; q1 = elipsoides.q1; r1 = elipsoides.r1;
P2 = elipsoides.P2; q2 = elipsoides.q2; r2 = elipsoides.r2;

f1 = 0.5*x'*P1*x+q1'*x+r1;
f2 = 0.5*x'*P2*x+q2'*x+r2;
c = [f1 f2];
ceq = [];
```

5.5.6 Comparación entre los resultados obtenidos

Después de haber resuelto el mismo problema con una herramienta de modelado (*CVX*), dos *solvers* de optimización convexa (*SeDuMi* y *Mosek*) y dos programas de optimización global (*Mathematica* y *Matlab Optimization Toolbox*), llegamos a resultados numéricamente similares, con una desviación típica de $3{,}3 \times 10^{-4}$ y $2{,}6 \times 10^{-4}$ en cada componente respecto a la variación entre cada herramienta, aunque no he modificado ningún parámetro de configuración para controlar la precisión, sino que he dejado los valores por defecto. Por tanto, esta medida no es significativa para este ejemplo, ya que la podremos ajustar tanto como necesitemos.

En cuanto a la forma de utilización y la sintaxis de cada herramienta, hemos comprobado que la adaptación del problema para resolverlo con *SeDuMi* presenta mayor dificultad por tener que convertirlo a problema cónico. En general, hemos visto que los *solvers* requieren mayor esfuerzo por parte del usuario, aunque en este caso, *Mosek* no resulta excesivamente complicado por tratarse de un problema cuadrático que esta herramienta puede resolver directamente, teniendo en cuenta el indexado de las matrices en forma dispersa. Por otro lado, la sintaxis de *CVX* resulta totalmente intuitiva; el código que hemos visto corresponde a lo que se define como modelo en este tipo de programas. En el caso de las herramientas orientadas a optimización

global, encontramos más flexibilidad en la definición de funciones, aunque la representación del problema de optimización resulta menos evidente que en los sistemas de modelado.

5.6 Conclusiones

El primer objetivo con el que se plantea este proyecto consiste en identificar y evaluar las herramientas de optimización convexa disponibles en el mercado. En este capítulo se hace efectiva esta revisión, y al comprobar las características de cada producto hemos podido comprender las diferencias entre programas de resolución y de modelado.

El criterio de selección de las distintas herramientas estudiadas en este capítulo ha consistido en elegir cuatro sistemas de modelado en función de la cantidad de referencias que he encontrado sobre ellos en la bibliografía consultada tanto para la teoría de optimización convexa como en los artículos de sus aplicaciones. A continuación se han escogido los cuatro *solvers* más utilizados en estos sistemas de modelado para los distintos tipos de problema que pueden resolver. Por último, he añadido la descripción de tres paquetes de optimización global para poder comparar sus características frente a los programas específicos de optimización convexa. Su selección se ha basado en la facilidad de acceso a ellos desde el entorno académico.

La revisión de las herramientas de resolución en §5.2 nos lleva a la conclusión de que se caracterizan por su especialización en un tipo de problemas o un conjunto limitado de estos, y en muchos casos emplean alguna variante de algoritmo de punto interior para resolver problemas de optimización convexa. También vemos que suelen incluir un procesamiento previo para aprovechar de forma eficiente la estructura de las matrices implicadas o su naturaleza de baja densidad.

Como resultado del estudio de los sistemas de modelado en §5.3 hemos aprendido que cuando queremos resolver un problema de optimización, primero necesitamos conocer suficientes técnicas para convertirlo en convexo, y después tenemos que convertirlo en el formato estándar correspondiente a alguno de los tipos de problema descritos en §3.3. En el caso de los programas de modelado nos ahorramos este último paso y únicamente nos tendremos que preocupar por describir nuestro modelo matemático convexo con la sintaxis característica de la herramienta que vayamos a utilizar.

La idea general que podemos extraer sobre los programas de optimización global en §5.4 es que incluyen un conjunto más amplio de funcionalidades que las otras herramientas de este capítulo ya que intentan comprender un rango generalizado de problemas, sin considerar el caso convexo en particular. Por este motivo no incluyen el procesamiento especializado de los programas estudiados en las secciones anteriores, mientras que añaden una gran cantidad de técnicas orientadas a no caer en mínimos locales que no se

necesitan en problemas convexos.

Existe un compromiso entre poder resolver una gran variedad de problemas diferentes y ser capaces de poder atacarlos cuando contienen una gran cantidad de variables y funciones de restricción. Cuando nuestro problema no sea excesivamente grande o no necesitemos una velocidad de ejecución muy exigente, probablemente no merezca la pena el esfuerzo para transformar un problema genérico en uno convexo, ya que las herramientas de resolución global podrán resolverlo directamente. Pero una vez adquirida la práctica suficiente, podremos abordar grandes problemas con técnicas de optimización convexa que permiten altas prestaciones en cuanto a precisión y velocidad de convergencia.

Para consolidar la comprensión acerca del funcionamiento de algunas de las herramientas, hemos resuelto un ejemplo sencillo pero bastante representativo del tipo de problemas que nos podemos encontrar. Al implementarlo para distintos *solvers* y programas de modelado ha quedado patente la diferencia entre ambos en cuanto a la forma de proceder para traducir nuestro problema al lenguaje de cada aplicación. La versión que más fácilmente hemos podido adaptar es para *CVX*, que es un sistema de modelado. Ha quedado de manifiesto que la mayor carga de trabajo está en reformular el problema que queremos resolver para adaptarlo al formato adecuado a la herramienta de *software* que vayamos a utilizar, especialmente, en §5.5.3, donde hemos tenido que utilizar recursos teóricos estudiados en la primera parte de este proyecto, como por ejemplo el planteamiento por epígrafo. Esto justifica el esfuerzo que hemos dedicado a comenzar este trabajo con una base teórica sólida y nos lleva a intuir que en los casos prácticos vamos a necesitar todos los recursos disponibles que hemos aprendido hasta ahora.

Parte III

Aplicaciones en telecomunicaciones

Capítulo 6

Conformación de haz

Después de haber analizado diferentes herramientas de optimización convexa disponibles en el mercado, pasamos a su utilización en aplicaciones prácticas. En este capítulo estudiamos el caso de conformación de haz, que consiste en combinar la respuesta de una agrupación de antenas para conseguir una comunicación selectiva en espacio, en presencia de interferencias y ruido.

La implementación práctica de este capítulo emplea *CVX* como sistema de modelado, *SeDuMi* y *Mosek* como herramientas de resolución, y *Matlab Optimization Toolbox* y *Mathematica* como programas de optimización global. El criterio de selección es el mismo que para el ejemplo de §5.5, es decir, en función de las referencias encontradas en la bibliografía y de su accesibilidad desde el entorno académico.

Dado que la resolución del problema de conformación de haz mediante las herramientas de optimización seleccionadas no es trivial, una parte muy importante del capitulo está dedicada a describir la adaptación de dicho problema a los formatos específicos requeridos para cada herramienta. Una vez hecho esto, los resultados obtenidos en términos de región de incertidumbre, relación señal a ruido, tiempos de ejecución y diagramas de radiación permiten establecer una comparación entre las herramientas empleadas.

El capítulo comienza con una introducción a las agrupaciones en §6.1 que está basada en [Car02] y en la teoría vista en la asignatura de antenas, y en ella se fijan las referencias básicas del ejemplo tratado en este tema. En §6.2.1 se determina la formulación y la opción de conformación robusta en recepción que se implementa en este trabajo, basada en la aplicación publicada en [Vor03]. Las ampliaciones revisadas en §6.2.2 tienen su origen en [Vor04, Vor08]. En §6.3 se modifica el problema robusto obtenido hasta aquí para convertirlo en un SOCP, y el proceso se basa en [Vor03]. Para adaptar el problema a cada herramienta tal como se describe en §6.3.1, §6.3.2, §6.3.3, §6.3.4 y §6.3.5 me he apoyado en el estudio de las ayudas y los manuales de cada programa, [Gra11, Stu98, Mos11, Mat10, Wol10]. En §6.4 veremos los resultados de las simulaciones, donde se compara el

funcionamiento de las herramientas utilizadas. La simulación de la matriz de covarianza de §6.4.1 está basada en [Lor05], y las perturbaciones del vector de apuntamiento son similares a uno de los casos presentados en [Vor03]. Por último, se extraen las conclusiones del capítulo en §6.5.

6.1 Agrupaciones de antenas

Tal como estudiábamos en la asignatura de transmisión y propagación, podemos obtener en teoría cualquier diagrama de radiación en una antena diseñando una distribución de corriente cuya transformada de Fourier sea el diagrama deseado [Car02]. Para conseguir esto en la práctica se recurre a las agrupaciones, haciendo que las alimentaciones de cada elemento sean una versión discreta de dicha distribución de corriente.

En este trabajo veremos únicamente distribuciones lineales sobre el eje z, tal como aparece en la figura 6.1, donde además se incluye el sistema de coordenadas en el que nos vamos a basar. De acuerdo con esto, dispondremos de N antenas equiespaciadas a una distancia d.

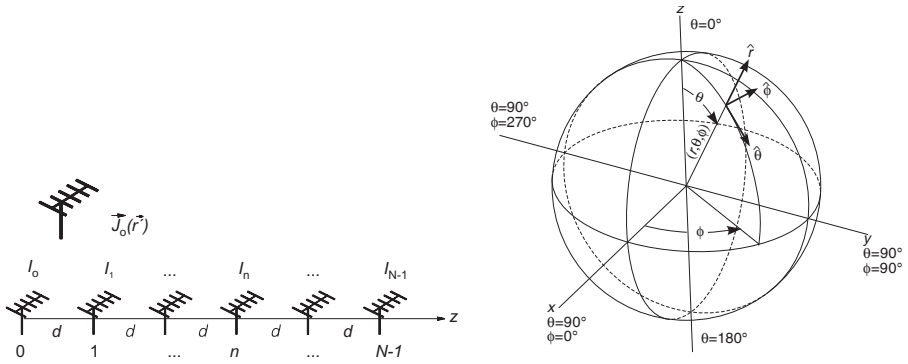

Figura 6.1: Agrupación uniforme y sistema de coordenadas.

Con estos datos vimos que el campo radiado tiene la siguiente expresión

$$\vec{\boldsymbol{E}}(\vec{\boldsymbol{r}}) = \vec{\boldsymbol{E}}_0(\vec{\boldsymbol{r}}) \sum_{n=0}^{N-1} I_n e^{jnkd\cos\theta} \tag{6.1}$$

es decir, el campo radiado es igual al de una antena multiplicado por un sumatorio que constituye el factor de agrupación. Desde el punto de vista práctico [Lor05], encontramos que la señal $y(t)$ obtenida de una agrupación cuando se transmite una señal $s(t)$ orientada desde θ_s será

$$\boldsymbol{y}(t) = \boldsymbol{a}_s s(t) + \boldsymbol{i}(t) + \boldsymbol{n}(t)$$

donde $\boldsymbol{i}(t)$ incluye los efectos de las señales interferentes, y $\boldsymbol{n}(t)$ es la componente de ruido. El vector $\boldsymbol{a}_s$ está relacionado con los componentes exponenciales de la ecuación (6.1), y depende del ángulo θ_s y de las posiciones de

cada antena de la agrupación, pero además incluye efectos como el acoplamiento entre elementos y la atenuación o amplificación. Es lo que se conoce como vector de apuntamiento o firma espacial. Su expresión teórica más sencilla en una distribución uniforme de N antenas separadas a una distancia d es la siguiente

$$\boldsymbol{a}_s = \left[1, e^{j\frac{2\pi}{\lambda}d\cos\theta_s}, e^{j\frac{2\pi}{\lambda}2d\cos\theta_s}, \ldots, e^{j\frac{2\pi}{\lambda}(N-1)d\cos\theta_s}\right]^T$$

El conformador de haz dispondrá de un vector de pesos $\boldsymbol{w} \in \mathbb{C}^N$ cuyo objetivo es conseguir la distribución de fasores I_n del planteamiento teórico.

Con esta disposición, el factor de agrupación resultante tendrá una simetría radial con respecto al eje z.

6.2 Conformación de haz en recepción

La conformación de haz en recepción es un campo clásico aunque en continuo desarrollo que tiene una historia rica de investigación teórica y aplicaciones prácticas en radar, sonar, comunicaciones, procesamiento de agrupaciones de micrófonos, biomedicina, radioastronomía, sismología y otras áreas. En los últimos años ha habido un interés renovado hacia la conformación de haz motivado por las comunicaciones inalámbricas, donde las tecnologías multiantena se han impuesto como una de las tecnologías clave para dar cabida al crecimiento masivo del número de usuarios y al rápido incremento de demanda por nuevos servicios a altas velocidades. Recientemente se ha producido un progreso significativo en el campo de la conformación de haz en recepción gracias a la optimización convexa. Motivado por el hecho de que las técnicas tradicionales de conformación adaptativa, como la conformación de varianza mínima, no son suficientemente robustas frente a pequeños desajustes en el vector de apuntamiento de la señal deseada. Varios autores propusieron técnicas robustas que se basan en el concepto de optimización de funcionamiento en el peor caso. Una característica distintiva en esta línea de trabajo es que utilizando teoría de optimización convexa, los problemas aparentemente robustos se pueden reformular en forma convexa, por lo que se pueden resolver de forma eficiente con los algoritmos que hemos visto en este trabajo.

Más allá de los diseños robustos deterministas de peor caso, ha aparecido recientemente una tendencia a utilizar alternativamente diseños probabilísticamente menos conservativos que emplean optimización convexa para resolver los problemas de riesgo resultantes. Además, ambos casos, el de conformación de haz con restricciones probabilísticas y de caso peor, se han extendido hasta el caso de diseñar receptores multiusuario para sistemas de comunicación MIMO codificadas en espacio y tiempo.

6.2.1 Contexto

Volviendo con el planteamiento de §6.1, la señal de salida de un conformador de haz de banda estrecha se puede escribir como

$$x(t) = \boldsymbol{w}^H \boldsymbol{y}(t)$$

donde $\boldsymbol{w} \in \mathbb{C}^N$ es el vector complejo de coeficientes de conformación de haz, $\boldsymbol{y}(t) \in \mathbb{C}^N$ es la muestra temporal del vector complejo de observaciones de la agrupación, y N es el número de sensores de la agrupación de antenas.

El vector de observaciones de agrupación se puede modelar como

$$\boldsymbol{y}(t) = \boldsymbol{s}(t) + \boldsymbol{i}(t) + \boldsymbol{n}(t)$$

donde $\boldsymbol{s}(t)$, $\boldsymbol{i}(t)$ y $\boldsymbol{n}(t)$ son la señal deseada y las componentes de interferencia y ruido de $\boldsymbol{y}(t)$, respectivamente. Cuando la fuente de señal es puntual, $\boldsymbol{s}(t) = s(t)\boldsymbol{a}_s$, donde $s(t)$ y $\boldsymbol{a}_s$ son la forma de onda deseada y su vector de apuntamiento o firma espacial, respectivamente.

El conformador de haz tiene que encontrar el conjunto de pesos que maximice la relación señal a ruido e interferencia, SINR[1], cuya expresión viene dada por

$$\begin{aligned} \text{SINR} &= \frac{\mathbb{E}\{|\boldsymbol{w}^H \boldsymbol{s}|^2\}}{\mathbb{E}\{|\boldsymbol{w}^H(\boldsymbol{i}+\boldsymbol{n})|^2\}} \\ &= \frac{\boldsymbol{w}^H \mathbb{E}\{\boldsymbol{s}\boldsymbol{s}^H\}\boldsymbol{w}}{\boldsymbol{w}^H \mathbb{E}\{(\boldsymbol{i}+\boldsymbol{n})(\boldsymbol{i}+\boldsymbol{n})^H\}\boldsymbol{w}} \\ &= \frac{\boldsymbol{w}^H \boldsymbol{R}_s \boldsymbol{w}}{\boldsymbol{w}^H \boldsymbol{R}_{i+n} \boldsymbol{w}} \\ &= \frac{\sigma_s^2 |\boldsymbol{w}^H \boldsymbol{a}_s|^2}{\boldsymbol{w}^H \boldsymbol{R}_{i+n} \boldsymbol{w}} \end{aligned} \tag{6.2}$$

donde $\boldsymbol{R}_s$ es la matriz de covarianza de la señal $\boldsymbol{s}(t)$, $\boldsymbol{R}_{i+n}$ es la matriz de covarianza de la suma de las señales $\boldsymbol{i}(t) + \boldsymbol{n}(t)$, y σ_s^2 es la varianza de la forma de onda deseada $s(t)$. Para maximizar esta expresión, hay que minimizar el denominador manteniendo la respuesta hacia la dirección de la señal deseada.

$$\begin{aligned} \underset{\boldsymbol{w}}{\text{minimizar}} \quad & \boldsymbol{w}^H \boldsymbol{R}_{i+n} \boldsymbol{w} \\ \text{sujeto a} \quad & \boldsymbol{w}^H \boldsymbol{a}_s = 1 \end{aligned} \tag{6.3}$$

En las aplicaciones prácticas no disponemos de la matriz de covarianzas de interferencias y ruido, $\boldsymbol{R}_{i+n}$, por lo que se utiliza la estimación muestral de $\boldsymbol{R}_y$ en lugar de $\boldsymbol{R}_{i+n}$

$$\hat{\boldsymbol{R}} = \frac{1}{J} \sum_{t=1}^{J} \boldsymbol{y}(t)\boldsymbol{y}^H(t) = \frac{1}{J} \boldsymbol{Y}\boldsymbol{Y}^H \tag{6.4}$$

[1] *Signal to Interference and Noise Ratio*

donde $\boldsymbol{Y} \triangleq [\boldsymbol{y}(1), \ldots, \boldsymbol{y}(J)]$ es la matriz de datos de entrenamiento del conformador de haz y J es el número de muestras disponibles. Por tanto, el vector óptimo de pesos se puede calcular aproximadamente resolviendo el siguiente problema convexo

$$\begin{array}{ll} \underset{\boldsymbol{w}}{\text{minimizar}} & \boldsymbol{w}^H \hat{\boldsymbol{R}} \boldsymbol{w} \\ \text{sujeto a} & \boldsymbol{w}^H \boldsymbol{a}_s = 1 \end{array} \tag{6.5}$$

Este problema tiene una solución cerrada que se puede expresar de la siguiente forma

$$\boldsymbol{w} = \beta \hat{\boldsymbol{R}}^{-1} \boldsymbol{a}_s \tag{6.6}$$

donde el escalar $\beta = (\boldsymbol{a}_s^H \hat{\boldsymbol{R}}^{-1} \boldsymbol{a}_s)^{-1}$ no afecta en la relación SINR de salida. El conformador de haz de (6.6) se conoce habitualmente como la técnica de mínima varianza basada en inversión de matriz de muestras, y por sus siglas en inglés sería SMI-MV[2].

El hecho de utilizar la matriz de covarianza de las muestras $\hat{\boldsymbol{R}}$ en lugar de $\boldsymbol{R}_{i+n}$ afecta de manera dramática al funcionamiento en comparación con el conformador óptimo en el caso en que la componente deseada de señal esté presente en las muestras de entrenamiento. Tengamos en cuenta que este último caso es típico en comunicaciones inalámbricas con múltiples antenas y en localización pasiva de fuente. La degradación mencionada causada por la cancelación de señales es conocida como autoanulación. Se hace especialmente fuerte en escenarios prácticos, cuando el conocimiento de $\boldsymbol{a}_s$ es también imperfecto.

Una de las aproximaciones más populares para mejorar la robustez de la técnica SMI-MV y para evitar la autoanulación es el método de carga diagonal (DL[3]), cuya idea clave es regularizar la solución de (6.5) añadiendo un término de penalización cuadrática $\gamma \boldsymbol{w}^H \boldsymbol{w}$ a la función objetivo, donde γ es el factor de carga diagonal preseleccionado (o factor DL). El conformador resultante se conoce como SMI[4] cargado o LSMI[5], y se calcula sustituyendo la matriz muestral de covarianza $\hat{\boldsymbol{R}}$ de (6.6) por su equivalente con carga diagonal, $\gamma \boldsymbol{I} + \hat{\boldsymbol{R}}$.

El único defecto de la aproximación por carga diagonal es que no existe una forma fiable para calcular el factor DL, γ. Cualquier elección fija de γ puede ser únicamente subóptima, porque la elección óptima de γ dependerá de cada escenario. Para evitar este inconveniente se han propuesto recientemente nuevos algoritmos robustos basados en diseños de caso peor [Vor03, Sha03, Vor04, Lor05, Li06].

La idea clave del conformador desarrollado en [Vor03] e independientemente en [Lor05] es modelar la incertidumbre del vector de apuntamiento

[2] *Sample Matrix Inversion based Minimum Variance*
[3] *Diagonal Loading*
[4] *Sample Matrix Inversion*
[5] *Loaded Sample Matrix Inversion*

como $\boldsymbol{\delta} \triangleq \tilde{\boldsymbol{a}}_s - \boldsymbol{a}_s$, donde $\tilde{\boldsymbol{a}}_s$ y $\boldsymbol{a}_s$ son el vector real de apuntamiento y el estimado, respectivamente; y la siguiente idea de este conformador es suponer que la norma euclídea de $\boldsymbol{\delta}$ está acotada como máximo a un valor conocido y constante ε. Esto corresponde al caso de incertidumbre esférica; en [Lor05, Li06] se considera un modelo elíptico más general de incertidumbre.

La esencia de la aproximación de [Vor03] es añadir robustez al problema (6.5) utilizando una restricción que tengan que satisfacer todos los vectores de apuntamiento desajustados siempre que estén dentro de un conjunto de incertidumbre específico. Con esta restricción, el conformador de mínima varianza robusto se formula como el siguiente problema de optimización:

$$\begin{array}{ll} \underset{\boldsymbol{w}}{\text{minimizar}} & \boldsymbol{w}^H \hat{\boldsymbol{R}} \boldsymbol{w} \\ \text{sujeto a} & |\boldsymbol{w}^H(\boldsymbol{a}_s + \boldsymbol{\delta})| \geq 1 \quad \forall \|\boldsymbol{\delta}\| \leq \varepsilon \end{array} \tag{6.7}$$

La restricción de (6.7) garantiza que la respuesta se mantenga sin distorsión en el peor caso, es decir, para la elección particular de $\boldsymbol{\delta}$ que corresponda al valor más pequeño de $|\boldsymbol{w}^H(\boldsymbol{a}_s+\boldsymbol{\delta})|$ si $\|\boldsymbol{\delta}\| \leq \varepsilon$. Para convertir (6.7) en forma convexa, encontramos en [Vor03] que si nuestra región de incertidumbre es razonablemente pequeña, como $\varepsilon \leq |\boldsymbol{w}^H \boldsymbol{a}_s|/\|\boldsymbol{w}\|$, entonces podemos acotar la restricción de (6.7) de acuerdo con (6.8)

$$\underset{\boldsymbol{w}}{\text{mín}} |\boldsymbol{w}^H(\boldsymbol{a}_s + \boldsymbol{\delta})| = |\boldsymbol{w}^H \boldsymbol{a}_s| - \varepsilon \|\boldsymbol{w}\|, \quad \forall \|\boldsymbol{\delta}\| \leq \varepsilon \tag{6.8}$$

Usando esta ecuación, y teniendo en cuenta que la función objetivo de (6.7) no varía cuando $\boldsymbol{w}$ sufre una rotación arbitraria en fase, se demuestra en [Vor03] que (6.7) se puede convertir en la siguiente forma convexa:

$$\begin{array}{ll} \underset{\boldsymbol{w}}{\text{minimizar}} & \boldsymbol{w}^H \hat{\boldsymbol{R}} \boldsymbol{w} \\ \text{sujeto a} & \boldsymbol{w}^H \boldsymbol{a}_s \geq \varepsilon \|\boldsymbol{w}\| + 1 \\ & \text{Im}\{\boldsymbol{w}^H \boldsymbol{a}_s\} = 0 \end{array} \tag{6.9}$$

Las restricciones de (6.9) implican que $\boldsymbol{w}^H \boldsymbol{a}_s$ sea positiva y real. Este problema es de tipo cónico de segundo orden (SOCP), y lo podemos resolver de forma eficiente con las herramientas que hemos visto en el capítulo 5. En [Vor03] se demuestra que la restricción de (6.9) es activa en la solución, es decir, se cumple con igualdad.

6.2.2 Extensiones de los diseños de caso peor

Una extensión útil del conformador robusto (6.9) se encuentra en [Vor04]. En este caso se considera un caso más general, donde además del desajuste en el vector de apuntamiento, los datos de entrenamiento no son estacionarios. Esto se caracteriza por medio de la matriz $\boldsymbol{\Delta}$ que modela el desajuste de la matriz de datos $\boldsymbol{Y}$ y se propone para combinar la robustez frente a no estacionariedad en interferencias y errores en los vectores de apuntamiento

utilizando las ideas de forma similar a [Vor03, Lor05]. De este modo, la función objetivo de (6.7) se puede modificar como

$$\max_{\|\boldsymbol{\delta}\| \leq \eta} \|(\boldsymbol{Y} + \boldsymbol{\Delta})^H \boldsymbol{w}\|^2$$

donde η es una cota superior conocida de la norma de la matriz $\boldsymbol{\Delta}$.

Se demuestra en [Vor04] que el problema resultante modificado se puede convertir a la siguiente forma convexa:

$$\begin{array}{ll} \underset{\boldsymbol{w}}{\text{minimizar}} & \|\boldsymbol{Y}^H \boldsymbol{w}\| + \eta \|\boldsymbol{w}\| \\ \text{sujeto a} & \boldsymbol{w}^H \boldsymbol{a}_s \geq \varepsilon \|\boldsymbol{w}\| + 1 \end{array} \tag{6.10}$$

Vemos que igual que en (6.9), el problema (6.10) también es de tipo SOCP, por lo que es fácil de resolver. Otra extensión útil de [Vor03, Lor05] se ha desarrollado en [Vor08]. Los autores argumentan que aunque los diseños de caso peor que hemos visto son bastante robustos, pueden ser extremadamente conservativos porque las condiciones de caso peor pueden ocurrir con muy baja probabilidad en la práctica. Esto motivó a los autores de [Vor08] a desarrollar una aproximación alternativa que proporciona robustez únicamente contra los errores más verosímiles en los vectores de apuntamiento.

Usando esta filosofía, el problema restringido probabilísticamente se escribe del siguiente modo

$$\begin{array}{ll} \underset{\boldsymbol{w}}{\text{minimizar}} & \boldsymbol{w}^H \hat{\boldsymbol{R}} \boldsymbol{w} \\ \text{sujeto a} & \Pr\{|\boldsymbol{w}^H (\boldsymbol{a}_s + \boldsymbol{\delta})| \geq 1\} > p \end{array} \tag{6.11}$$

donde $\boldsymbol{\delta}$ es un vector aleatorio de desajuste extraído de una distribución conocida; $\Pr\{\cdot\}$ es el operador de probabilidad, cuya forma específica se obtiene de las suposiciones estadísticas de los errores en los vectores de apuntamiento; y p es el umbral de probabilidad preseleccionado. En contraste con la restricción determinista usada en (6.7), que requiere mantener la respuesta sin distorsión en todos los vectores dentro de la esfera de incertidumbre, la restricción débil de (6.11) mantiene la respuesta sin distorsión únicamente en los vectores con desajuste $\boldsymbol{\delta}$ cuya probabilidad sea suficientemente alta, mientras que ignora los valores de $\boldsymbol{\delta}$ que pueden ocurrir con poca probabilidad. Por tanto, la restricción de (6.11) se puede interpretar como una restricción en la probabilidad de desconexión manteniendo baja esta probabilidad ($p_{desc} = 1 - p$).

En [Vor08] se muestra que para la distribución del desajuste del vector de apuntamiento gaussiana circularmente simétrica, y para la distribución de caso peor, el problema de riesgo en (6.11) se puede aproximar por medio de problemas SOCP. La distribución de caso peor para una varianza dada resulta ser discreta. Este resultado implica un reajuste que hace la restricción de desconexión sea más estricta. Curiosamente, los problemas resultantes son bastante parecidos a (6.9). Sin embargo, una ventaja importante de

los conformadores de haz de [Vor08] con respecto a los de caso peor de [Vor03] y [Lor05] es que los primeros permiten cuantificar explícitamente los parámetros de la región de incertidumbre en términos de la probabilidad de desconexión del conformador de haz. Esto hace más eficiente la elección de ε.

Una de las mayores dudas que me aparecían al comenzar a estudiar este tema era el planteamiento de la perturbación exclusivamente en el vector de apuntamiento de la dirección principal, pero en [Kim08, Oh05] se presenta un caso general en el que el vector a optimizar incluye el vector de apuntamiento y la matriz de covarianza, de forma que la perturbación que no he tenido en cuenta en este trabajo se refleja en la matriz. En cualquier caso, mi objetivo era dedicar una parte importante de las pruebas a comparar el funcionamiento de diferentes herramientas, y el caso presentado sirve de base para todas estas ampliaciones, en las que la base del funcionamiento adecuado está en la región para la que nuestro conformador es robusto.

6.3 Implementación del optimizador

Una vez revisadas las posibles ampliaciones que podríamos hacer en el diseño de caso peor, nos centramos en el caso implementado en este capítulo, que se basa en resolver el problema (6.9) con las herramientas presentadas en el capítulo 5. Para ello hay que hacer algunas transformaciones con el fin de trabajar con variables reales en lugar de complejas. Para ser más precisos, el motivo principal no es evitar que las señales sean complejas, ya que algunas herramientas lo permiten, sino la implementación de la restricción de igualdad en (6.9)

$$\mathrm{Im}\{\boldsymbol{w}^H \boldsymbol{a}_s\} = 0$$

El primer paso se dirige a convertir la función objetivo de (6.9), que es una forma cuadrática, en una restricción cónica. Hay que transformarla factorizando la matriz de covarianza por Cholesky, de forma que $\hat{\boldsymbol{R}} = \boldsymbol{U}^H \boldsymbol{U}$, por lo que se convierte en

$$\boldsymbol{w}^H \hat{\boldsymbol{R}} \boldsymbol{w} = \|\boldsymbol{U}\boldsymbol{w}\|^2$$

Como la minimización de $\|\boldsymbol{U}\boldsymbol{w}\|^2$ es equivalente a minimizar $\|\boldsymbol{U}\boldsymbol{w}\|$, podemos utilizar el planteamiento en forma de epígrafo que vimos en §3.1.7 añadiendo una nueva variable t

$$\begin{array}{ll} \underset{t,\boldsymbol{w}}{\text{minimizar}} & t \\ \text{sujeto a} & \|\boldsymbol{U}\boldsymbol{w}\| \leq t \\ & \varepsilon\|\boldsymbol{w}\| \leq \boldsymbol{w}^H \boldsymbol{a}_s - 1 \\ & \mathrm{Im}\{\boldsymbol{w}^H \boldsymbol{a}_s\} = 0 \end{array} \tag{6.12}$$

Por otro lado, hay que añadir nuevas variables reales del siguiente modo

$$\begin{aligned} \boldsymbol{w}_r &\triangleq \left[\mathrm{Re}\{\boldsymbol{w}\}^T, \mathrm{Im}\{\boldsymbol{w}\}^T\right]^T \\ \boldsymbol{a}_1 &\triangleq \left[\mathrm{Re}\{\boldsymbol{a}_s\}^T, \mathrm{Im}\{\boldsymbol{a}_s\}^T\right]^T \\ \boldsymbol{a}_2 &\triangleq \left[\mathrm{Im}\{\boldsymbol{a}_s\}^T, -\mathrm{Re}\{\boldsymbol{a}_s\}^T\right]^T \\ \boldsymbol{U}_r &\triangleq \begin{bmatrix} \mathrm{Re}\{\boldsymbol{U}\} & -\mathrm{Im}\{\boldsymbol{U}\} \\ \mathrm{Im}\{\boldsymbol{U}\} & \mathrm{Re}\{\boldsymbol{U}\} \end{bmatrix} \end{aligned} \tag{6.13}$$

Ahora podemos escribir (6.12) con variables reales

$$\begin{aligned} \underset{t,\boldsymbol{w}_r}{\text{minimizar}} \quad & t \\ \text{sujeto a} \quad & \|\boldsymbol{U}_r\boldsymbol{w}_r\| \leq t \\ & \varepsilon\|\boldsymbol{w}_r\| \leq \boldsymbol{w}_r^T\boldsymbol{a}_1 - 1 \\ & \boldsymbol{w}_r^T\boldsymbol{a}_2 = 0 \end{aligned} \tag{6.14}$$

Esta ecuación es un SOCP con variables reales que se puede utilizar en cualquiera de las herramientas que hemos visto con las adaptaciones que veremos a continuación para cada caso.

6.3.1 CVX

El problema (6.14) se escribe en formato de *CVX* casi igual que se plantea en dicha ecuación, ya que he utilizado los mismos nombres en las variables, salvo por la escritura con subíndices de las fórmulas anteriores, que no queda idéntica en *Matlab*.

```
1      a1 = [real(as); imag(as)];
2      a2 = [imag(as); -real(as)];
3      U = chol(R);
4      Ur = [real(U) -imag(U); imag(U) real(U)];
5      % optimización
6      cvx_begin
7          % Parámetros
8          cvx_quiet(true)
9          variable t;
10         variable wr(2*N);
11     % Problema
12     minimize( t )
13     subject to
14         norm(Ur*wr) <= t;
15         eps*norm(wr)<=wr'*a1-1;
16         wr'*a2==0;
17     cvx_end
```

Las primeras cuatro líneas corresponden a las definiciones de (6.13), incluyendo la factorización por Cholesky. Estas líneas son iguales en la implementación de las cinco herramientas utilizadas en este proyecto. El resto del

listado es el SOCP de (6.14). Hay que añadir que la orden `cvx_quiet(true)` sirve para eliminar toda la información de ejecución de *CVX*. Para comparar opciones, he añadido la posibilidad de cambiar el *solver* que utiliza *CVX*. Esto se hace añadiendo la línea `cvx_solver sdpt3` o `cvx_solver sedumi` dentro del problema de optimización.

Por último, para saber si el problema se ha resuelto correctamente hay que comprobar la variable `cvx_status`, que tiene contener la cadena de caracteres `'Solved'`. Otras posibilidades son `'Unbounded'` si el problema no tiene límite inferior, o `'Infeasible'` para problemas no viables.

6.3.2 SeDuMi

Para resolver el problema (6.14) con *SeDuMi* podemos revisar el procedimiento del apartado §5.5.3. En este caso tenemos la ventaja de que el problema ya es cónico.

El formato estándar en *SeDuMi*, utilizando la misma nomenclatura que en el manual de referencia [Stu98] que viene incluido en la descarga del *software*, es como en (5.4), el siguiente

$$\begin{array}{ll} \underset{\boldsymbol{z}}{\text{minimizar}} & \boldsymbol{c}^T\boldsymbol{z} \\ \text{sujeto a} & \boldsymbol{A}\boldsymbol{z} = \boldsymbol{b} \\ & \boldsymbol{z} \in \mathcal{K} \end{array} \tag{6.15}$$

En primer lugar hay que definir la variable del problema, $\boldsymbol{z}$, y esto viene condicionado por la definición del cono $\mathcal{K}$ con componentes consecutivas de dicha variable. Por ahora no conocemos su dimensión, pero los siguientes pasos nos llevarán a completar sus componentes. Para entenderlo mejor, comenzamos por la primera restricción del problema (6.14):

$$\|\boldsymbol{U}_r\boldsymbol{w}_r\| \leq t \tag{6.16}$$

Si llamamos $\boldsymbol{z}_1 \in \mathbb{R}^{2N}$ a un subconjunto de componentes de la variable $\boldsymbol{z}$, siendo N el número de antenas de la agrupación, y asignamos el interior de la norma (6.16) a este vector $\boldsymbol{z}_1$,

$$\boldsymbol{z}_1 \triangleq \boldsymbol{U}_r\boldsymbol{w}_r \tag{6.17}$$

entonces la restricción (6.16) define el siguiente cono de segundo orden para el vector $[t, \boldsymbol{z}_0^T]^T \in \mathbb{R}^{2N+1}$:

$$t \geq \|\boldsymbol{z}_1\|$$

Para la segunda restricción de (6.14), empezamos por reorganizarla del siguiente modo

$$\frac{\boldsymbol{w}_r^T\boldsymbol{a}_1 - 1}{\varepsilon} \geq \|\boldsymbol{w}_r\|$$

y asignamos otra componente $z_2 \in \mathbb{R}$ de la variable $\boldsymbol{z}$ a los términos de esta restricción que no están dentro del operador norma $\|\cdot\|$:

$$z_2 \triangleq \frac{\boldsymbol{w}_r^T \boldsymbol{a}_1 - 1}{\varepsilon} \tag{6.18}$$

por tanto, la segunda restricción de (6.14) define el siguiente cono de segundo orden para el vector $[z_2, \boldsymbol{w}_r^T]^T \in \mathbb{R}^{2N+1}$:

$$z_2 \geq \|\boldsymbol{w}_r\|$$

Ahora ya podemos definir la variable $\boldsymbol{z}$ del problema como

$$\boldsymbol{z} \triangleq \left[t, \boldsymbol{z}_1^T, z_2, \boldsymbol{w}_r^T\right]^T$$

Las primeras $2N+1$ componentes de $\boldsymbol{z}$ definen un cono de segundo orden, y las siguientes $2N+1$ componentes definen otro cono de segundo orden, por tanto $\boldsymbol{z} \in \mathbb{R}^{4N+2}$. Esto es lo único que necesita saber *SeDuMi* acerca de la variable del problema, y se indica en el campo `q` de la estructura `K` que pasaremos al optimizador: `K.q = [2*N+1 2*N+1];`. Además hay que decir que no hay variables libres, es decir, que $\boldsymbol{z}$ no tiene ninguna componente fuera de los conos anteriores. Esto se indica haciendo cero el campo `f` de dicha estructura: `K.f=0`. Ahora hay que indicar las definiciones (6.17) y (6.18), junto con la restricción de igualdad del problema (6.14) en forma de ecuación matricial $\boldsymbol{A}\boldsymbol{z} = \boldsymbol{b}$.

La igualdad (6.17) se puede expresar como

$$\begin{gathered} \boldsymbol{z}_1 = \boldsymbol{U}_r \boldsymbol{w}_r \\ \Updownarrow \\ [\boldsymbol{0}_{2N}, -\boldsymbol{I}_{2N}, \boldsymbol{0}_{2N}, \boldsymbol{U}_r] \begin{bmatrix} t \\ \boldsymbol{z}_1 \\ z_2 \\ \boldsymbol{w}_r \end{bmatrix} = \left[\boldsymbol{0}_{2N}\right] \end{gathered} \tag{6.19}$$

siendo $\boldsymbol{0}_{2N}$ un vector con $2N$ filas, e $\boldsymbol{I}_{2N}$ la matriz identidad de $\mathbb{R}^{2N \times 2N}$. La igualdad (6.18) se puede expresar como

$$\begin{gathered} z_2 = \frac{\boldsymbol{w}_r^T \boldsymbol{a}_1 - 1}{\varepsilon} \\ \Updownarrow \\ \left[\boldsymbol{0}_{2N+1}^T, \varepsilon, -\boldsymbol{a}_1^T\right] \begin{bmatrix} t \\ \boldsymbol{z}_1 \\ z_2 \\ \boldsymbol{w}_r \end{bmatrix} = -1 \end{gathered} \tag{6.20}$$

La restricción de igualdad de (6.14) se puede presentar del siguiente modo

$$\begin{gathered} \boldsymbol{w}_r^T \boldsymbol{a}_2 = 0 \\ \Updownarrow \\ \left[\mathbf{0}_{2N+2}^T, \boldsymbol{a}_2^T\right] \begin{bmatrix} t \\ \boldsymbol{z}_1 \\ z_2 \\ \boldsymbol{w}_r \end{bmatrix} = 0 \end{gathered} \tag{6.21}$$

Ya tenemos datos suficientes para construir la igualdad $\boldsymbol{Az} = \boldsymbol{b}$ agrupando (6.19), (6.20) y (6.21)

$$\begin{gathered} \boldsymbol{Az} = \boldsymbol{b} \\ \Updownarrow \\ \begin{bmatrix} \mathbf{0}_{2N} & -\boldsymbol{I}_{2N} & \mathbf{0}_{2N} & \boldsymbol{U}_r \\ \mathbf{0}_{2N+1}^T & \cdots\cdots & \varepsilon, & -\boldsymbol{a}_1^T \\ \mathbf{0}_{2N+2}^T & \cdots\cdots\cdots & & \boldsymbol{a}_2^T \end{bmatrix} \begin{bmatrix} t \\ \boldsymbol{z}_1 \\ z_2 \\ \boldsymbol{w}_r \end{bmatrix} = \begin{bmatrix} \mathbf{0}_{2N} \\ -1 \\ 0 \end{bmatrix} \end{gathered} \tag{6.22}$$

La función objetivo expresada como $\boldsymbol{cz}$ queda así

$$\begin{gathered} \boldsymbol{cz} \equiv t \\ \Updownarrow \\ \left[1, \mathbf{0}_{4N+1}^T\right] \begin{bmatrix} t \\ \boldsymbol{z}_1 \\ z_2 \\ \boldsymbol{w}_r \end{bmatrix} \end{gathered} \tag{6.23}$$

El código relevante de la implementación del problema con *SeDuMi* es el siguiente

```
1  A = [zeros(2*N,1) -eye(2*N) zeros(2*N,1) Ur; ...% -z1 + Ur*wr = 0
2       zeros(1,2*N+1) eps -a1';...              % eps*z2 - a1'*wr = -1
3       zeros(1,2*N+2) a2'];                     %           a2'*wr = 0
4  b = [zeros(2*N,1); -1; 0];
5  c = [1; zeros(4*N+1,1)];
6
7  K.f = 0; % No hacen falta variables libres
8  K.q = [2*N+1 2*N+1]; % 2 conos de segundo orden
9  pars.fid = 0; % Para ejecutar sin mostrar mensajes
10 [z y info] = sedumi(sparse(A),b,c,K,pars);
11 wr = full(z(2*N+3:end)); % convertir wr de dispersa a matriz completa
12 w = wr(1:N,1)+j*wr(N+1:end,1);
```

Las cuatro primeras líneas salen de identificar $\boldsymbol{A}$ y $\boldsymbol{b}$ de (6.22); la quinta viene de identificar $\boldsymbol{c}$ en (6.23); las líneas 7 y 8 indican las dimensiones de

los conos de segundo orden, como ya he comentado; y una vez añadido el parámetro de la línea 9 para que *SeDuMi* no vuelque toda la información durante la ejecución, la llamada al optimizador la tenemos en la línea 10. Después hay que extraer el vector de pesos $\boldsymbol{w}_r$ de la variable $\boldsymbol{z}$ y convertirlo en el vector complejo $\boldsymbol{w}$ de acuerdo con (6.13).

Para comprobar si el resultado es correcto, hay que mirar el valor del campo `info.feasratio` devuelto por *SeDuMi*. Dependiendo de la viabilidad del problema, convergerá a 1 si el problema tiene una solución completamente viable, y a -1 en problemas fuertemente inviables. Si el problema no es completamente lineal dará soluciones intermedias. En nuestro caso aceptamos valores superiores a 0.9.

6.3.3 Mosek

La forma de indicar los datos de un problema SOCP en *Mosek* es similar a *SeDuMi*, planteando el problema en el siguiente formato:

$$\begin{array}{ll} \underset{\boldsymbol{z}}{\text{minimizar}} & \boldsymbol{c}^T\boldsymbol{z} \\ \text{sujeto a} & \boldsymbol{l}^c \leq \boldsymbol{A}\boldsymbol{z} \leq \boldsymbol{u}^c \\ & \boldsymbol{l}^z \leq \boldsymbol{z} \leq \boldsymbol{u}^z \\ & \boldsymbol{z} \in \mathcal{K} \end{array} \tag{6.24}$$

Empezamos asignando valores a $\boldsymbol{A}$, $\boldsymbol{b}$ y $\boldsymbol{c}$ exactamente igual que en *SeDuMi*. Después, para que (6.15) y (6.24) sean equivalentes hay que asignar $\boldsymbol{l}^c = \boldsymbol{u}^c = \boldsymbol{b}$, con lo que la primera restricción de (6.24) se convierte en $\boldsymbol{A}\boldsymbol{z} = \boldsymbol{b}$. Como no tenemos ninguna restricción que acote la variable $\boldsymbol{z}$ haremos que $\boldsymbol{l}^z = -\infty$ y $\boldsymbol{u}^z = \infty$, convirtiendo la segunda restricción de (6.24) en $-\infty \leq \boldsymbol{x} \leq \infty$.

Con estos datos, y repitiendo el código de *SeDuMi* para asignar valores a $\boldsymbol{A}$, $\boldsymbol{b}$ y $\boldsymbol{c}$, la parte relevante de código para utilizar *Mosek* es la siguiente:

```
1  prob.a = sparse(A);
2  prob.c = c;
3  prob.blc = b;
4  prob.buc = b;
5  prob.blx = -inf*ones(4*N+2,1);
6  prob.bux = inf*ones(4*N+2,1);
7  prob.cones = cell(2,1);
8  prob.cones{1}.type = 'MSK_CT_QUAD';
9  prob.cones{1}.sub = [1:2*N+1];
10 prob.cones{2}.type = 'MSK_CT_QUAD';
11 prob.cones{2}.sub = [2*N+2:4*N+2];
12 param.MSK_IPAR_LOG=0;
13
14 [r,res] = mosekopt('minimize',prob,param);
15 wr = full(res.sol.itr.xx(2*N+3:end));
16 w = wr(1:N,1)+j*wr(N+1:end,1);
```

Observamos que los datos pasan al optimizador como campos de una estructura. Hasta la línea 6 asignamos los valores de $\boldsymbol{A}$, $\boldsymbol{b}$ y $\boldsymbol{c}$; después vemos que los conos se describen dentro del campo `prob.cones`, que es una celda, en nuestro problema con dos elementos. Las líneas 9 y 11 indican los índices de la variable $\boldsymbol{z}$ que definen cada cono, y las líneas 8 y 10 indican que se trata de conos de segundo orden.

En la línea 12 se elimina la mayor parte de la información que *Mosek* vuelca en la línea de comandos, pero no he conseguido hacer que se ejecute sin producir el mensaje de identificación con el número de versión.

Igual que en los casos anteriores, al final hay que extraer el vector de pesos $\boldsymbol{w}_r$ de la variable $\boldsymbol{z}$ y convertirlo en el vector complejo $\boldsymbol{w}$ de acuerdo con (6.13).

Para comprobar si el resultado es válido, el campo de la variable devuelta `res.sol.itr.solsta` tiene que contener la cadena de caracteres `'OPTIMAL'`.

6.3.4 Matlab Optimization Toolbox

6.3.4.1 Descripción directa

Como vimos en el capítulo 5, el complemento de optimización de *Matlab* es una herramienta que no necesita que el problema a optimizar sea convexo. Por este motivo, mi primera opción fue utilizar directamente el problema (6.14) para describir las funciones necesarias para `fmincon`.

En primer lugar hay que definir una variable de optimización, y en mi caso consiste en

$$\boldsymbol{x} \triangleq \begin{bmatrix} \boldsymbol{w}_r \\ t \end{bmatrix}$$

Después hay que crear la función objetivo y las restricciones tal como se crean funciones en *Matlab*. La función objetivo queda del siguiente modo

```
1 function [t] = f0(x)
2 t = x(end);
```

Las restricciones no lineales se incluyen en otra función

```
function[c, ceq] = restricciones(x)
    % Parámetros globales. Esta función sólo puede depender de x
    global PARAMETROS;
    a1 = PARAMETROS{1};
    a2 = PARAMETROS{2};
    Ur = PARAMETROS{3};
    eps = PARAMETROS{4};
    % Parámetros incluidos en la variable x
    t = x(end);
    wr = x(1:end-1,1);
    % Descripción de las restricciones
    f1 = norm(Ur*wr) - t;
    f2 = eps*norm(wr) - wr'*a1 + 1;
    c = [f1 f2];
    ceq = [];
```

Vemos que se necesita utilizar variables globales para que las restricciones puedan leer los parámetros del problema, ya que esta función únicamente puede admitir como argumento la variable de optimización $\boldsymbol{x}$. Por lo demás, el código es tal como se describen las dos primeras restricciones de (6.14) en formato

$$\|\boldsymbol{U}_r\boldsymbol{w}_r\| - t \leq 0$$
$$\varepsilon\|\boldsymbol{w}_r\| - \boldsymbol{w}_r^T\boldsymbol{a}_1 + 1 \leq 0$$

Por otro lado, las restricciones lineales se pasan como parámetros de `fmincon`. Una vez creadas las funciones anteriores, la optimización queda del siguiente modo

```
% Opciones del optimizador
opciones = optimset('Algorithm','interior-point');
x0 = rand(2*N+1,1); % Punto inicial (aleatorio)
Aeq = [a2' 0]; % Restricción lineal de igualdad
beq = 0;
[wr, fval, exf] = fmincon(@f0,x0,[],[],Aeq,beq,[],[],...
                          @restricciones,opciones);
w = wr(1:N,1)+j*wr(N+1:end-1,1);
```

El primer argumento de `fmincon` apunta a la función objetivo `@f0`; el segundo argumento es el punto inicial `x0`; los argumentos tercero y cuarto (`[],[]`) definen restricciones lineales de desigualdad $\boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{b}$ (en nuestro problema no hay); los argumentos quinto y sexto (`Aeq` y `beq`) definen las restricciones de igualdad $\boldsymbol{A}_{eq}\boldsymbol{x} = \boldsymbol{b}_{eq}$. En nuestro caso, teniendo en cuenta que $\boldsymbol{x} = \left[\boldsymbol{w}_r^T, t\right]^T$, la tercera restricción de (6.14) se añade del siguiente

modo

$$\begin{gathered}\boldsymbol{w}_r^T\boldsymbol{a}_2 = 0\\ \Updownarrow\\ \left[\boldsymbol{a}_2^T, 0\right]\begin{bmatrix}\boldsymbol{w}_r\\ t\end{bmatrix} = 0\end{gathered}$$

De aquí extraemos los valores asignados a $\boldsymbol{A}_{eq}$ y a $\boldsymbol{b}_{eq}$. El séptimo argumento de `fmincon` apunta a las restricciones; y el último argumento es la estructura en la que se definen las opciones del optimizador.

Al ejecutar las simulaciones tal como he descrito hasta aquí, el optimizador no converge a resultados correctos. Para solucionarlo la solución que he encontrado es añadir información sobre los gradientes de las funciones no lineales y de la función objetivo.

La función objetivo es lineal, por lo que es fácil ver que su gradiente es el siguiente

$$\begin{gathered}f_0(\boldsymbol{x}) = \left[\mathbf{0}_{2N}^T\, 1\right]\begin{bmatrix}\boldsymbol{w}_r\\ t\end{bmatrix} = \left[\mathbf{0}_{2N}^T\, 1\right]\boldsymbol{x}\\ \Updownarrow\\ \nabla f_1(\boldsymbol{x}) = \left[\mathbf{0}_{2N}^T\, 1\right]\end{gathered}$$

El gradiente de la primera función de restricción de (6.14) tiene la siguiente expresión:

$$\begin{gathered}f_1(\boldsymbol{x}) = \|\boldsymbol{U}_r\boldsymbol{w}_r\| - t\\ \Updownarrow\\ \nabla f_1(\boldsymbol{x}) = \begin{bmatrix}\dfrac{\boldsymbol{U}_r^T\boldsymbol{U}_r\boldsymbol{w}}{\|\boldsymbol{U}_r\boldsymbol{w}\|}\\ -1\end{bmatrix}\end{gathered}$$

El gradiente de la segunda función de restricción de (6.14) tiene la siguiente expresión:

$$\begin{gathered}f_2(\boldsymbol{x}) = \varepsilon\|\boldsymbol{w}_r\| - \boldsymbol{w}_r^T\boldsymbol{a}_1 + 1\\ \Updownarrow\\ \nabla f_2(\boldsymbol{x}) = \begin{bmatrix}\dfrac{\boldsymbol{w}_r}{\|\boldsymbol{w}_r\|} - \boldsymbol{a}_1\\ 0\end{bmatrix}\end{gathered}$$

Con estos datos, podemos modificar las funciones que hemos visto para que *Matlab* utilice estas expresiones analíticas en lugar de hacer aproximaciones numéricas. En el código de la función objetivo hay que añadir un segundo argumento de salida para que devuelva el gradiente:

```
function [t gradf] = f0(x)
t = x(end);
 % Gradiente
if nargout > 1
    gradf = [zeros(length(x)-1,1); 1];
end
```

También hay que hacer que la función de restricciones devuelva los gradientes de las funciones no lineales. No es necesario añadir el gradiente de la restricción de igualdad porque es lineal (si no lo fuese, nuestro problema no sería convexo).

```
function[c, ceq, gradc, gradceq] = restricciones(x)

    % Parámetros globales. Esta función sólo puede depender de x
    global PARAMETROS;
    a1 = PARAMETROS{1};
    a2 = PARAMETROS{2};
    Ur = PARAMETROS{3};
    eps = PARAMETROS{4};
    % Parámetros incluidos en la variable x
    t = x(end);
    wr = x(1:end-1,1);
    % Descripción de las restricciones
    f1 = norm(Ur*wr) - t;
    f2 = eps*norm(wr) - wr'*a1 + 1;
    c = [f1 f2];
    ceq = [];

    % Cálculo de gradientes
    gradf1 = [Ur'*Ur*wr/norm(Ur'*wr); -1];
    gradf2 = [wr/norm(wr); 0] - [a1; 0];
    if nargout > 2
        gradc = [gradf1 gradf2];
        gradceq = [];
    end
```

Para que `fmincon` utilice estas expresiones, hay que modificar la estructura de opciones que se pasa como último argumento, añadiendo lo siguiente

```
opciones = optimset('Algorithm','interior-point',...
                    'GradObj','on',...
                    'GradConstr','on');
```

También se puede añadir en esta estructura el campo `'Display'` con valor `'off'` para que se ejecute en modo silencioso, sin volcar mensajes en la línea de comandos.

Existe la posibilidad de añadir además el hessiano de la función lagran-

giana para que lo utilice el algoritmo de punto interior, tal como vimos en el capítulo 4. En este caso no he podido encontrar una solución analítica en forma vectorial para un número genérico de antenas N. No obstante, con añadir los gradientes, los resultados mejoran notablemente.

Con estas modificaciones, `fmincon` empieza a dar resultados comparables con las herramientas anteriores, aunque sigue fallando en algunos casos que veremos más adelante en (6.4.4).

6.3.4.2 Planteamiento estándar SOCP

Al no obtener resultados completamente satisfactorios con el planteamiento anterior, la siguiente opción es utilizar el problema en formato estándar SOCP como en *SeDuMi* y *Mosek* (6.25), con las mismas definiciones exactamente que en ambos casos anteriores.

$$\begin{array}{ll} \underset{\boldsymbol{z}}{\text{minimizar}} & \boldsymbol{c}^T\boldsymbol{z} \\ \text{sujeto a} & \boldsymbol{A}\boldsymbol{z} = \boldsymbol{b} \\ & \boldsymbol{z} \in \mathcal{K} \end{array} \tag{6.25}$$

Una vez asignados los valores de $\boldsymbol{A}$, $\boldsymbol{b}$ y $\boldsymbol{c}$, nos queda modificar la función objetivo y las restricciones no lineales. Hay que tener en cuenta que nuestra variable es ahora $\boldsymbol{z} \in \mathbb{R}^{4N+2}$

$$\boldsymbol{z} \triangleq \left[t, \boldsymbol{z}_1^T, z_2, \boldsymbol{w}_r^T\right]^T$$

por lo que el gradiente de la función objetivo es

$$\nabla f_0(\boldsymbol{z}) = \boldsymbol{c} = \begin{bmatrix} 1 \\ \boldsymbol{0}_{4N+1} \end{bmatrix}$$

El código que incluye la función objetivo y su gradiente es

```
function [t gradf] = f0_s(z)
    % z = [t; z1; z2; wr]
    % f0(z) = c*z
    c = [1; zeros(length(z)-1,1)];
    t = z(1);
    % Gradiente
    if nargout > 1
        gradf = c;
    end
```

Ahora calculamos el gradiente de la primera restricción

$$f_1(\boldsymbol{z}) = -t + \|\boldsymbol{z}_1\|$$
$$\Updownarrow$$
$$\nabla f_1(\boldsymbol{z}) = \begin{bmatrix} -1 \\ \dfrac{\boldsymbol{z}_1}{\|\boldsymbol{z}_1\|} \\ \boldsymbol{0}_{2N+1} \end{bmatrix}$$

Y del mismo modo calculamos el gradiente de la segunda restricción

$$f_2(\boldsymbol{z}) = -z_2 + \|\boldsymbol{w}_r\|$$
$$\Updownarrow$$
$$\nabla f_1(\boldsymbol{z}) = \begin{bmatrix} \mathbf{0}_{2N+1} \\ -1 \\ \dfrac{\boldsymbol{w}_r}{\|\boldsymbol{w}_r\|} \end{bmatrix}$$

El código de las restricciones devuelve también los gradientes calculados

```
function[c, ceq, gradc, gradceq] = restricciones_s(z)

    M = length(z);   % 4N+2
    N = (M-2)/4;     % Número de antenas
    r1 = norm(z(2:2*N+1)); % Norma de z1
    f1 = -z(1) + r1;     % Primera restricción cónica

    r2 = norm(z(2*N+3:M));   % Norma de wr
    f2 = -z(2*N+2) + r2;     % Segunda restricción cónica

    c = [f1 f2];     % Restricciones no lineales de desigualdad
    ceq = [];        % No hay restricciones no lineales de igualdad

    % Cálculo de gradientes
    gradf1 = [-1; z(2:2*N+1)/r1; zeros(2*N+1,1)];
    gradf2 = [zeros(2*N+1,1); -1; z(2*N+3:M)/r2];
    if nargout > 2
        gradc = [gradf1 gradf2];
        gradceq = [];
    end
```

Vemos que aquí no hacen falta variables globales para pasar parámetros del problema a las restricciones, puesto que únicamente hace falta conocer las dimensiones de la variable $\boldsymbol{z}$.

En este caso también podemos calcular el hessiano de la función lagrangiana

$$\nabla^2 L(\boldsymbol{z}, \boldsymbol{\lambda}, \boldsymbol{\nu}) = \nabla^2 f_0(\boldsymbol{z}) + \lambda_1 \nabla^2 f_1(\boldsymbol{z}) + \lambda_2 \nabla^2 f_2(\boldsymbol{z})$$

El hessiano de la función objetivo es cero, por ser una función lineal, igual que el de las funciones de restricción de igualdad. El hessiano de la primera función de restricción tiene la siguiente expresión:

$$\nabla^2 f_1(\boldsymbol{z}) = \begin{bmatrix} 0 & \cdots & 0 & \cdots & 0 \\ \vdots & \dfrac{\boldsymbol{I}_{2N}}{\|\boldsymbol{z}_1\|} - \dfrac{\boldsymbol{z}_1 \boldsymbol{z}_1^T}{\|\boldsymbol{z}_1\|^3} & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & \cdots & 0 \end{bmatrix}$$

El hessiano de la segunda función de restricción es

$$\nabla^2 f_2(\boldsymbol{z}) = \begin{bmatrix} 0 & \dots & 0 & \dots & 0 \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ 0 & \dots & 0 & \dots & 0 \\ \vdots & \ddots & \vdots & \dfrac{\boldsymbol{I}_{2N}}{\|\boldsymbol{w}_r\|} - \dfrac{\boldsymbol{w}_r \boldsymbol{w}_r^T}{\|\boldsymbol{w}_r\|^3} & \\ 0 & \dots & 0 & & \end{bmatrix}$$

Con estos datos, hay que añadir una función para pasársela a `fmincon` con el siguiente código

```
function H = hessiano(z,lambda)
    M = length(z);   % 4N+2
    N = (M-2)/4;     % Número de antenas
    H0 = zeros(M,M); % hessiano de la función objetivo

    r1 = norm(z(2:2*N+1)); % Norma de z1
    h1 = 1/r1*eye(2*N) - z(2:2*N+1)*z(2:2*N+1)'/r1^3; % versión 2N*2N
    H1 = zeros(M,M);
    H1(2:2*N+1,2:2*N+1) = h1; % versión M*M

    r2 = norm(z(2*N+3:M));  % Norma de wr
    h2 = 1/r2*eye(2*N) - z(2*N+3:M)*z(2*N+3:M)'/r2^3; % versión 2N*2N
    H2 = zeros(M,M);
    H2(2*N+3:M,2*N+3:M) = h2; % versión M*M

    % hessiano de la función lagrangiana
    H = H0 + lambda.ineqnonlin(1)*H1 + lambda.ineqnonlin(2)*H2;
```

El optimizador `fmincon` se encarga de llamar a esta función y le pasa los multiplicadores de Lagrange λ_1 y λ_2, pero para que lo haga, hay que indicarlo en la estructura de opciones con los siguientes campos:

```
opciones = optimset('Algorithm','interior-point',...
                    'GradObj','on',...
                    'GradConstr','on',...
                    'hessian','user-supplied',...
                    'HessFcn',@hessiano,...
                    'Display','off');
```

6.3.5 Mathematica

Igual que en el caso anterior, los comandos de optimización de *Mathematica* no necesitan que el problema a optimizar sea convexo. Para poder hacer una comparación más objetiva de los resultados con las herramientas vistas hasta ahora, he utilizado el complemento *Mathematica Symbolic Toolbox*

for MATLAB [Bar04], que permite abrir un *kernel* de *Mathematica* desde *Matlab*, y desde aquí se envían comandos y matrices, y se reciben sus respuestas y valores. De este modo, la perturbación aleatoria del vector de apuntamiento es exactamente la misma en todos los casos. Por ejemplo, para pasar la matriz $\boldsymbol{U}_r$ después de haber factorizado en *Matlab* la matriz de covarianza $\boldsymbol{R}$ y haber expandido el resultado para conseguir una matriz real, la transferimos a *Mathematica* con la siguiente orden

```
1      math('matlab2math','Ur',Ur);
```

No se puede hacer lo mismo con los vectores y los valores escalares, porque en la sintaxis de *Mathematica* no es lo mismo un escalar que una matriz en $\mathbb{R}^{1\times 1}$. Por eso, para pasar el valor de ε y N los comandos son los siguientes

```
1      math(sprintf('eps = %d;',eps));
2      math(sprintf('M = %d;',N));
```

Hay que tener en cuenta que no podemos utilizar la variable N en *Mathematica*, porque es un comando reservado, y por eso he utilizado M en su lugar.

En el caso de los vectores, después de pasarlos hay que utilizar el comando `Flatten` para que no sean tratados como matrices.

```
1      math('matlab2math','a1',a1);
2      math('a1=Flatten[a1]');
3      math('matlab2math','a2',a2);
4      math('a2=Flatten[a2]');
```

Observamos que después de pasar la variable $\boldsymbol{a}_1$ se pueden ejecutar comandos de *Mathematica* con esta variable escribiendo la orden como una cadena de caracteres dentro de la función `math()`. Lo mismo hacemos con $\boldsymbol{a}_2$.

También he añadido dos implementaciones del problema de conformación, para comparar el funcionamiento codificando directamente el problema (6.14) o utilizando el problema en formato estándar (6.25).

6.3.5.1 Descripción directa

El primer paso es definir una variable de optimización, y en mi caso consiste en

$$\boldsymbol{x} \triangleq \begin{bmatrix} t \\ \boldsymbol{w}_r \end{bmatrix}$$

Para describir la función objetivo necesitamos la constante $\boldsymbol{c} \in \mathbb{R}^{2N+1}$ que multiplica a $\boldsymbol{x}$ para dar t, es decir,

$$\boldsymbol{c} \triangleq \begin{bmatrix} 1 \\ \mathbf{0}_{2N} \end{bmatrix}$$

Para poner la primera restricción de (6.14) en función de $\boldsymbol{x}$, hay que añadir una columna de ceros a $\boldsymbol{U}_r$, de forma que

$$\begin{gathered} \|\boldsymbol{U}_r \boldsymbol{w}_r\| \leq t \\ \Updownarrow \\ \left\| [\mathbf{0}_{2N}\ \boldsymbol{U}_r] \begin{bmatrix} t \\ \boldsymbol{w}_r \end{bmatrix} \right\| \leq \begin{bmatrix} 1\ \mathbf{0}_{2N}^T \end{bmatrix} \begin{bmatrix} t \\ \boldsymbol{w}_r \end{bmatrix} \end{gathered}$$

En el término derecho de la ecuación podemos utilizar el vector $\boldsymbol{c}$ de la función objetivo para obtener t. Para adjuntar la columna de ceros a $\boldsymbol{U}_r$, y para asignar el valor de $\boldsymbol{c}$ he utilizado los siguientes comandos de *Mathematica*

```
U0 = Map[Prepend[#, 0] &, Ur];
c = Join[{1}, ConstantArray[0, 2*M]];
```

Estos comandos se ejecutan desde *Matlab* encerrándolos como cadenas de caracteres en la función `math()`.

Igualmente tenemos que escribir la segunda restricción de (6.14) en función de $\boldsymbol{x}$. Para esto podemos añadir un cero al principio del vector $\boldsymbol{a}_1$, y necesitamos otra constante $\boldsymbol{c}_w \in \mathbb{R}^{2N}$ que multiplique a $\boldsymbol{x}$ para dar $\boldsymbol{w}_r$.

$$\begin{gathered} \varepsilon \|\boldsymbol{w}_r\| \leq \boldsymbol{w}_r^T \boldsymbol{a}_1 - 1 \\ \Updownarrow \\ \varepsilon \left\| \begin{bmatrix} 0\ \mathbf{1}_{2N}^T \end{bmatrix} \begin{bmatrix} t \\ \boldsymbol{w}_r \end{bmatrix} \right\| \leq \begin{bmatrix} 0\ \boldsymbol{a}_1^T \end{bmatrix} \begin{bmatrix} t \\ \boldsymbol{w}_r \end{bmatrix} - 1 \end{gathered}$$

De aquí extraemos los valores de $\boldsymbol{c}_w$ y del nuevo vector $\boldsymbol{a}_{10}$ que es $\boldsymbol{a}_1$ con un cero añadido

```
a10 = Join[{0}, a1];
cw = Join[{0}, ConstantArray[1, 2*M]];
```

Nos queda ahora codificar la restricción de igualdad de (6.14) en función de $\boldsymbol{x}$, y del mismo modo, hay que añadir un cero al vector $\boldsymbol{a}_2$ creando un nuevo $\boldsymbol{a}_{20}$ extendido, con lo que la restricción queda

$$\begin{gathered} \boldsymbol{w}_r^T \boldsymbol{a}_2 = 0 \\ \Updownarrow \\ \begin{bmatrix} 0\ \boldsymbol{a}_2^T \end{bmatrix} \begin{bmatrix} t \\ \boldsymbol{w}_r \end{bmatrix} = 0 \end{gathered}$$

De aquí se justifica el siguiente comando de *Mathematica*

```
a10 = Join[{0}, a1];
```

Antes de llamar al optimizador, necesitamos asignar variables simbólicas a las componentes de $\boldsymbol{x}$, por ejemplo como

$$\boldsymbol{x} = [x_0 \; x_1 \; \ldots x_{2M}]$$

y esto en *Mathematica* se hace con el siguiente comando

```
x = Map[Subscript["x", #] &, Range[0, 2*M]];
```

De este modo, podemos usar la función `FindMinimum` con una variable vectorial. Ahora nos queda únicamente llamar al optimizador con la siguiente línea

```
Sol = FindMinimum[{c.x,
    Norm[U0.x] <= c.x &&
    eps*Norm[cw*x] <= a10.x - 1 &&
    a20.x == 0},
    x, Method -> "InteriorPoint"];
```

Observamos que en *Mathematica* no hay que transponer los vectores para el producto escalar (véase `c.x`, `a10.x` o `a20.x`). En el caso de `cw*x` se hace un producto término a término entre ambos vectores.

La solución viene en una variable con forma parecida a una celda de *Matlab*. Para extraer el resultado hacen falta las siguientes órdenes

```
Rw = Sol[[2]][[#]][[2]] & /@ Range[2, M + 1];
Iw = Sol[[2]][[#]][[2]] & /@ Range[M + 2, 2*M + 1];
w = Rw + I*Iw;
```

La transferencia de *Mathematica* a *Matlab* se hace del siguiente modo

```
1    w = math('math2matlab','w');
2    w = transpose(w);
```

6.3.5.2 Planteamiento estándar SOCP

Por el mismo motivo que en *Matlab Optimization Toolbox*, he añadido el formato estándar de nuestro problema para resolverlo con *Mathematica*. El planteamiento es exactamente igual que en los casos anteriores, por lo que no voy a repetir el código con el que se generan los parámetros $\boldsymbol{A}$, $\boldsymbol{b}$ y $\boldsymbol{c}$ desde *Matlab*, y después se transfieren a *Mathematica*.

Después de este proceso, la minimización se obtiene con el siguiente código

```
Sol = FindMinimum[{c.x,
     x[[1]] >= Norm[x[[2 ;; 2*M+1]]] &&
     x[[2*M+2]] >= Norm[x[[2*M+3 ;; 4*M+2]]] &&
     A.x == b},
     x, Method -> "InteriorPoint"];
```

6.4 Simulaciones

Las pruebas incluidas en este capítulo reproducen el conformador descrito en §6.2.1, con el problema (6.7) expresado en forma cónica (6.9).

En la mayor parte de las simulaciones consideramos una agrupación lineal de 10 antenas separadas en $0{,}45\lambda$, con el objetivo de reproducir resultados similares a [Vor03, Ger03, Lor05]. La señal útil viene desde la dirección $\theta_{obj} = 93°$, y se añaden dos señales interferentes desde las direcciones $\theta_{int1} = 120°$ y $\theta_{int2} = 140°$, con una potencia superior al ruido en 30 dB (INR[6]).

6.4.1 Matriz de covarianza

En lugar de generar un conjunto de muestras para obtener la matriz de covarianza descrita en (6.4), me he basado directamente en la potencia de las señales utilizadas. Si tenemos dos señales interferentes, la agrupación dará la siguiente salida

$$\boldsymbol{y}(t) = \tilde{\boldsymbol{a}}_s(\theta_{obj})s(t) + \boldsymbol{a}_s(\theta_{int1})s_{int1}(t) + \boldsymbol{a}_s(\theta_{int2}) + \boldsymbol{n}(t)$$

La notación de $\tilde{\boldsymbol{a}}_s(\theta_{obj})$ indica que se trata del verdadero vector de apuntamiento después de sufrir la distorsión descrita más adelante en §6.4.2. La matriz de covarianza de la señal tendrá la siguiente expresión

$$\begin{aligned}\boldsymbol{R} &= \mathbb{E}\{\boldsymbol{y}\boldsymbol{y}^H\} \\ &= \sigma^2_{obj}\tilde{\boldsymbol{a}}_s(\theta_{obj})\tilde{\boldsymbol{a}}^H_s(\theta_{obj}) \\ &+ \sigma^2_{int1}\boldsymbol{a}_s(\theta_{int1})\boldsymbol{a}^H_s(\theta_{int1}) \\ &+ \sigma^2_{int2}\boldsymbol{a}_s(\theta_{int2})\boldsymbol{a}^H_s(\theta_{int2}) \\ &+ \sigma^2_n\boldsymbol{I}\end{aligned} \tag{6.26}$$

donde la potencia de la señal útil es $\mathbb{E}\{ss^*\} = \sigma^2_{obj}$, y la de las señales interferentes $\mathbb{E}\{s_{int}s^*_{int}\} = \sigma^2_{int}$. La covarianza del ruido es $\sigma^2_n\boldsymbol{I}$, siendo σ^2_n la potencia de ruido. La matriz de covarianza de las interferencias y ruido será la siguiente

$$\boldsymbol{R}_{i+n} = \sigma^2_{int1}\boldsymbol{a}_s(\theta_{int1})\boldsymbol{a}^H_s(\theta_{int1}) + \sigma^2_{int2}\boldsymbol{a}_s(\theta_{int2})\boldsymbol{a}^H_s(\theta_{int2}) + \sigma^2_n\boldsymbol{I}$$

[6] *Interference to Noise Ratio*

Esta expresión nos servirá para calcular el conformador óptimo de acuerdo con (6.6) y para dar el valor de SINR de acuerdo con (6.2).

Recordemos que la solución al problema (6.27)

$$\begin{array}{ll} \underset{\boldsymbol{w}}{\text{minimizar}} & \boldsymbol{w}^H \boldsymbol{R}_0 \boldsymbol{w} \\ \text{sujeto a} & \boldsymbol{w}^H \boldsymbol{a}_s = 1 \end{array} \tag{6.27}$$

para una matriz $\boldsymbol{R}_0$ genérica tiene la siguiente solución

$$\begin{aligned} \boldsymbol{w}_0 &= \beta \boldsymbol{R}_0^{-1} \boldsymbol{a}_s \\ \beta &= (\boldsymbol{a}_s^H \boldsymbol{R}_0^{-1} \boldsymbol{a}_s)^{-1} \end{aligned}$$

En el caso del conformador ideal, si utilizamos $\boldsymbol{R}_{i+n}$ este problema maximiza la relación SINR como vimos en (6.3), por tanto, el conformador ideal tiene la siguiente expresión

$$\begin{aligned} \boldsymbol{w}_{opt} &= \beta \boldsymbol{R}_{i+n}^{-1} \boldsymbol{a}_s \\ \beta &= (\boldsymbol{a}_s^H \boldsymbol{R}_{i+n}^{-1} \boldsymbol{a}_s)^{-1} \end{aligned} \tag{6.28}$$

El conformador de inversión de matriz muestral, SMI, utiliza la matriz de covarianza basada en las muestras de la señal recibida, $\hat{\boldsymbol{R}}$, como vimos en (6.5). En nuestra simulación utilizo la expresión de $\boldsymbol{R}$ calculada en (6.26), por lo que

$$\begin{aligned} \boldsymbol{w}_{\text{SMI}} &= \beta \boldsymbol{R}^{-1} \boldsymbol{a}_s \\ \beta &= (\boldsymbol{a}_s^H \boldsymbol{R}^{-1} \boldsymbol{a}_s)^{-1} \end{aligned} \tag{6.29}$$

Hemos visto en §6.2.1 que el conformador con carga diagonal utiliza como matriz de covarianza $\gamma \boldsymbol{I} + \boldsymbol{R}$, por lo que su vector de pesos tiene la siguiente expresión

$$\begin{aligned} \boldsymbol{w}_{\text{LSMI}} &= \beta \left(\gamma \boldsymbol{I} + \boldsymbol{R}\right)^{-1} \boldsymbol{a}_s \\ \beta &= \left(\boldsymbol{a}_s^H \left(\gamma \boldsymbol{I} + \boldsymbol{R}\right)^{-1} \boldsymbol{a}_s\right)^{-1} \end{aligned} \tag{6.30}$$

6.4.2 Perturbación en el vector de apuntamiento

El modelo de variación en el vector de apuntamiento corresponde a una distorsión causada por dispersión local coherente de la señal deseada [Gol98], con cinco caminos diferentes de acuerdo con la siguiente expresión

$$\tilde{\boldsymbol{a}}_s = \boldsymbol{a}_s + \sum_{i=1}^{4} e^{j\psi_i} \boldsymbol{b}(\theta_i)$$

donde $\boldsymbol{a}_s$ corresponde al camino directo, y $\boldsymbol{b}(\theta_i)$, $(i = 1, 2, 3, 4)$ corresponde a los caminos que han sufrido dispersión coherente. Cada uno de estos dispersores llega a la agrupación desde la dirección θ_i, que se genera de forma aleatoria con distribución uniforme centrada en el ángulo nominal apuntado (93°) y con una desviación de 1°. Los parámetros ψ_i representan las fases de cada camino y se obtienen de una distribución uniforme en el intervalo $[0, 2\pi]$.

6.4.3 Selección de la región de incertidumbre

En este apartado vamos a realizar una prueba comparativa en la que modificamos el radio de la región de incertidumbre ε y medimos la relación SINR obtenida con cada herramienta. Se mostrarán los resultados para diferentes valores de SNR[7]. Recordamos que la relación SINR se calcula con la expresión (6.2), pero ahora tenemos en cuenta la distorsión descrita en §6.4.2.

$$\text{SINR} = \frac{\boldsymbol{w}^H \boldsymbol{R}_s \boldsymbol{w}}{\boldsymbol{w}^H \boldsymbol{R}_{i+n} \boldsymbol{w}} = \frac{\sigma_s^2 |\boldsymbol{w}^H \tilde{\boldsymbol{a}}_s|^2}{\boldsymbol{w}^H \boldsymbol{R}_{i+n} \boldsymbol{w}}$$

En las gráficas, aparecen como referencia los niveles de SINR para el conformador óptimo, sustituyendo en esta expresión el valor de $\boldsymbol{w}$ calculado en (6.28). También se incluye el valor del LSMI, basado en (6.30), y para el SMI (6.29).

6.4.3.1 Gráfica de *CVX*, *SeDuMi* y *Mosek*

Se incluyen aquí en el mismo bloque las simulaciones de estas tres herramientas porque producen el mismo resultado. Podemos comprobar en la figura 6.2 que las gráficas resultantes de los tres optimizadores se solapan y no se pueden distinguir unas de otras. Se incluyen dos versiones del optimizador

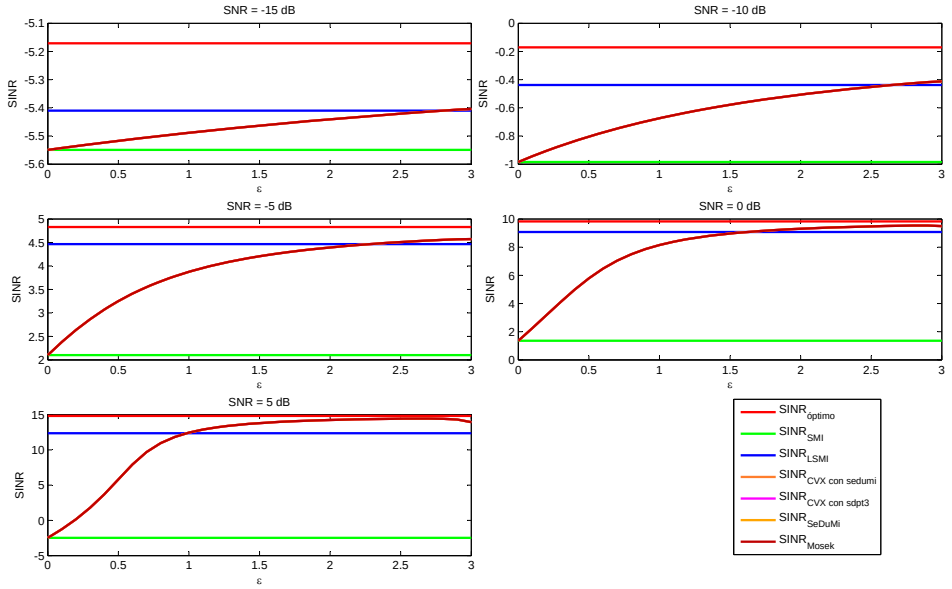

Figura 6.2: SINR vs. ε para las herramientas de optimización convexa.

con *CVX* porque en una se emplea *SeDuMi* como *solver* y en la otra *SDPT3*.

[7] *Signal to Noise Ratio*

Cuando $\varepsilon = 0$ nuestro problema es igual al SMI, ya que (6.9) se hace equivalente a (6.5). Aumentando ε el optimizador robusto supera al SMI hasta llegar a superar la calidad del LSMI. Cuando SNR= -15dB nuestro resultado supera al LSMI para $\varepsilon \geq 2{,}8$; si SNR= -10dB lo supera con $\varepsilon \geq 2{,}7$; si SNR= -5dB lo supera con $\varepsilon \geq 2{,}3$; si SNR= 0dB lo supera con $\varepsilon \geq 1{,}7$; y si SNR= 5dB lo supera con $\varepsilon \geq 1$;

Para valores superiores a $\varepsilon = 3$ el problema es inviable con todas las herramientas utilizadas.

El máximo valor de SINR para SNR≤ -5dB está en $\varepsilon = 3$; con SNR= 0dB está en $\varepsilon = 2{,}9$; y con SNR= 0dB está en $\varepsilon = 2{,}6$. Como conclusión de estas gráficas, para este margen de valores de SNR, nos conviene utilizar un valor de ε en torno a 2,9, ya que con los valores bajos de SNR superamos al LSMI para $\varepsilon \geq 2{,}8$, y en los valores altos de SNR los ε óptimos son 2,9 y 2,6 (descartamos el último dato por no superar al LSMI en los valores bajos de SNR).

6.4.3.2 Gráficas de *Matlab Optimization Toolbox*

Vemos ahora el comportamiento de *Matlab Optimization Toolbox*. Por la cantidad de errores que se producen, no podemos llegar a las mismas conclusiones con esta herramienta como las del apartado anterior. Este inconveniente hace que tengamos que buscar un valor muy bajo de SNR ($= -10$dB) suficientemente alejado del efecto de autoanulación descrito en §6.2.1, que nos permita ver una gráfica de esta simulación con la menor cantidad posible de puntos no convergentes. Recordemos que ya hemos podido estudiar el funcionamiento del conformador robusto con las herramientas de optimización convexa en §6.4.3.1, y en este apartado buscamos más comprobar cómo se comporta *Matlab Optimization Toolbox*, aunque no nos permita valores útiles de SNR en un sentido práctico.

6.4.3.2.1 Descripción directa

En la figura 6.3 tenemos los datos de la codificación directa descrita en §6.3.4.1. La figura de la izquierda es una ampliación de la figura que hay a la derecha. Esto es porque los datos con error producen valores muy bajos de SINR y el margen visible de la figura no nos permite apreciar la zona de resultados entre el SMI y el LSMI.

- ($\cdot$,$\times$) Opción directa sin gradiente.
  Cuando no se utiliza el gradiente, observamos una tendencia creciente con ε exceptuando los valores para los que el optimizador no converge a una solución. Los valores correctos están marcados con un punto ($\cdot$) y son los que forman la tendencia creciente comentada. Los valores erróneos con una ($\times$) y dan resultados erráticos.

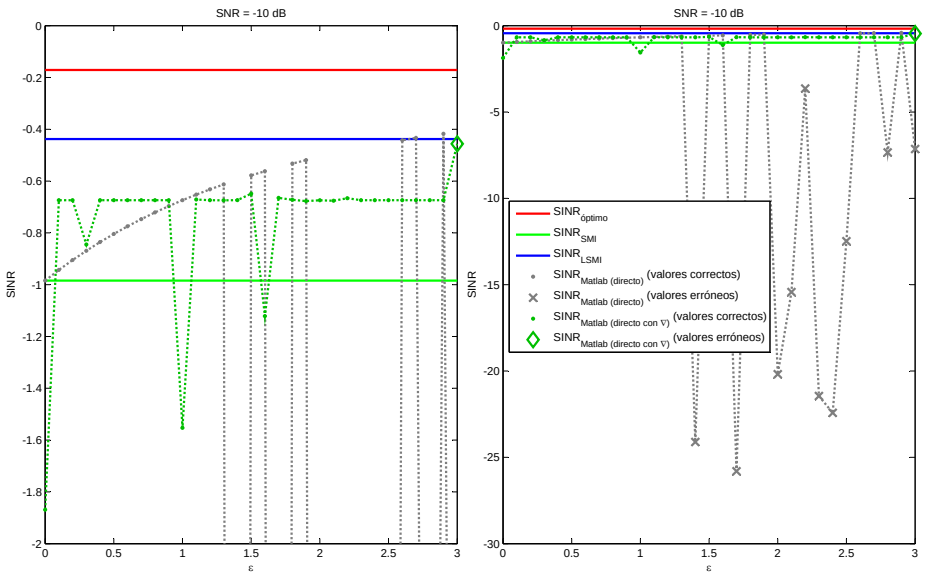

Figura 6.3: SINR vs. ε para *Matlab Optimization Toolbox, codificado directamente.*

- ($\cdot$, $\diamond$) Opción directa con gradiente.
  Cuando sí se utiliza el gradiente, no observamos esta tendencia creciente en los valores correctos. El optimizador devuelve valores por encima del SMI, pero no supera al LSMI. Los valores erróneos ($\diamond$) no se desvían demasiado de los correctos ($\cdot$) como ocurría en el caso anterior.

6.4.3.2.2 Planteamiento estándar SOCP

En la figura 6.4 están los datos de la codificación estándar descrita en §6.3.4.2.

- ($\cdot$, $\triangledown$) Opción estándar sin gradiente
  Si omitimos la información de los gradientes, únicamente se produce un error de convergencia cuando $\varepsilon = 0$, aunque la solución devuelta en este caso también produce una SINR prácticamente igual a la solución correcta.

- ($-$) Opción estándar con gradiente
  Cuando nos limitamos a proporcionar los gradientes, la gráfica obtenida no devuelve ningún error. Únicamente el valor inicial con $\varepsilon = 0$ se separa un poco del resultado esperado.

- ($\cdot$, $\circ$) Opción estándar con hessiano
  La gráfica que presenta más valores erróneos es la correspondiente

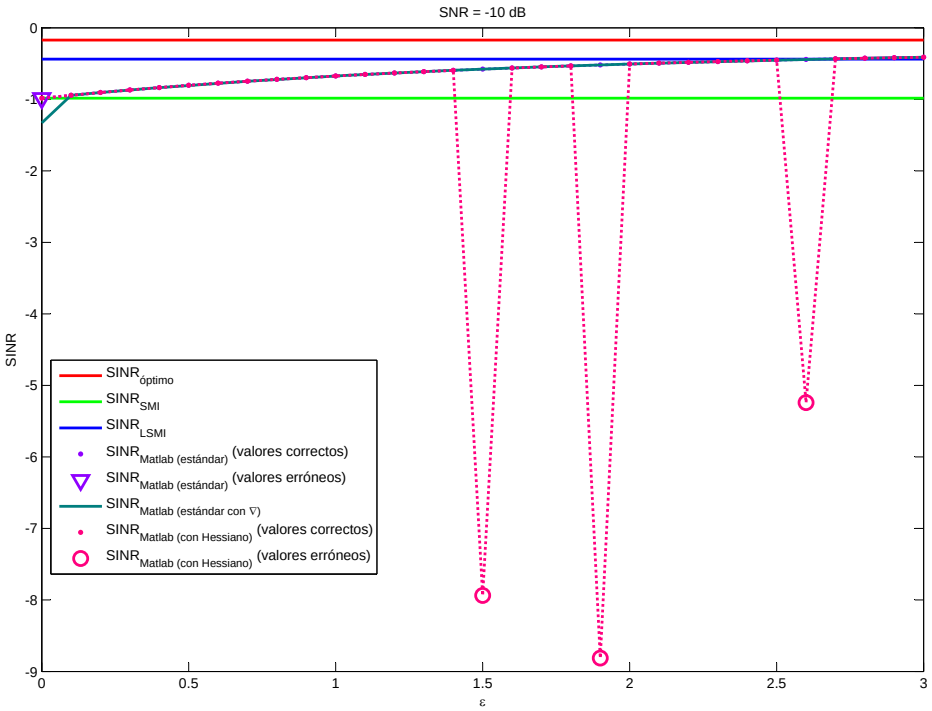

Figura 6.4: SINR vs. ε para *Matlab Optimization Toolbox, codificado en formato estándar SOCP.*

a la codificación en la que damos la versión analítica del hessiano de la función lagrangiana. Además, cuando se producen errores, los resultados se alejan excesivamente de su valor correcto.

Esto nos lleva a una conclusión desconcertante: al proporcionar la expresión analítica del hessiano, `fmincon` puede llegar a errores con más probabilidad que si no lo facilitamos. Al principio pensé que podría haber calculado mal esta expresión, pero si así fuese no tendríamos resultados correctos para ningún valor de ε. Después de revisar [Mat10], una posible explicación podría ser que en algunos casos la expresión analítica del hessiano no fuese definida positiva, mientras que la aproximación *cuasi-Newton* implementada por defecto sí lo es, por lo que existiría una gran diferencia en el cálculo de factorización de dicha matriz. En cualquier caso, esta situación es característica de nuestro problema, y no se puede generalizar para el optimizador. La opción que mejor resultado da es utilizar la información de los gradientes.

6.4.3.3 Gráfica de *Mathematica*

Con esta herramienta también se producen muchos errores, por lo que mostraré únicamente los resultados con SNR= 0dB, que es un valor poco prác-

tico, elegido con los mismos argumentos que he explicado en §6.4.3.2.

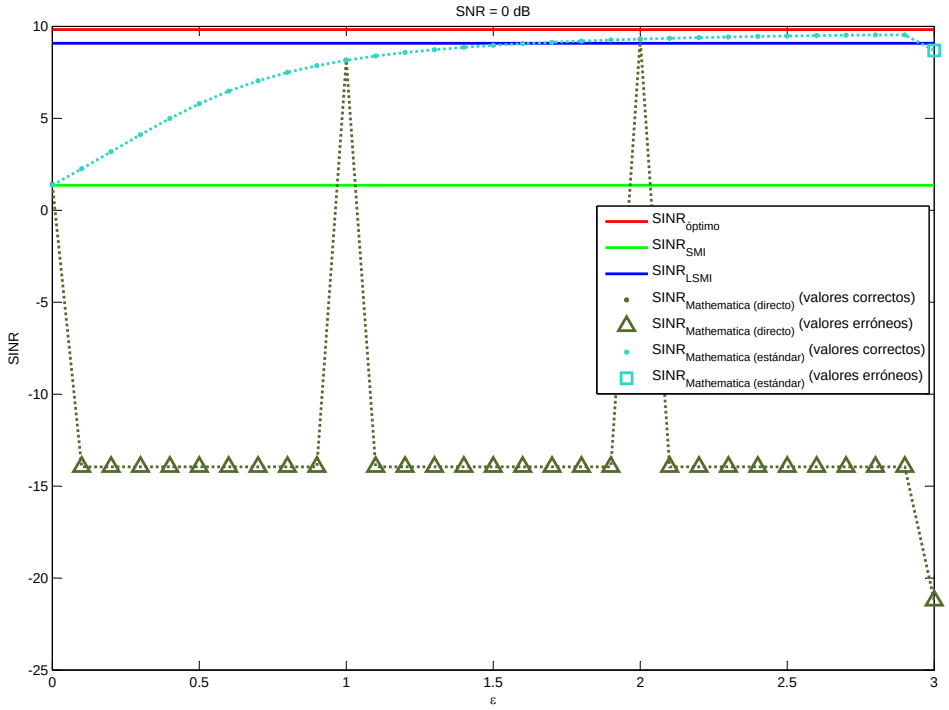

Figura 6.5: SINR vs. ε para *Mathematica*

La gráfica marcada con ($\cdot$, $\triangle$) corresponde a la implementación directa descrita en §6.3.5.1. En la figura 6.5 vemos que obtenemos valores correctos únicamente para $\varepsilon = 1$ y para $\varepsilon = 2$. El resto produce errores de convergencia.

Si describimos el problema con el planteamiento estándar de §6.3.5.2 llegamos a soluciones correctas ($\cdot$) excepto para $\varepsilon = 3$ que produce un error ($\square$).

La principal utilidad de esta gráfica está en que si queremos hacer otra comparativa de herramientas de optimización con nuestro problema, en la que queramos incluir la implementación directa con *Mathematica*, tendremos que asignar $\varepsilon = 2$, que es el punto en el que nuestro optimizador supera al LSMI y no produce error en el *solver* de *Mathematica*. Si queremos añadir la implementación estándar evitaremos el valor $\varepsilon = 3$.

6.4.4 SINR en función de SNR

Esta simulación muestra la evolución de la medida de calidad de nuestro conformador de haz en función de la SNR, es decir, del nivel de nuestra señal por encima del ruido. De acuerdo con el apartado anterior, §6.4.3.1 las pruebas se hacen con $\varepsilon = 2{,}9$, salvo en el caso de *Mathematica*.

6.4.4.1 Gráfica de *CVX*, *SeDuMi* y *Mosek*

En los datos de la figura 6.6 no podemos distinguir entre las líneas de las tres herramientas comparadas, porque producen prácticamente los mismos resultados.

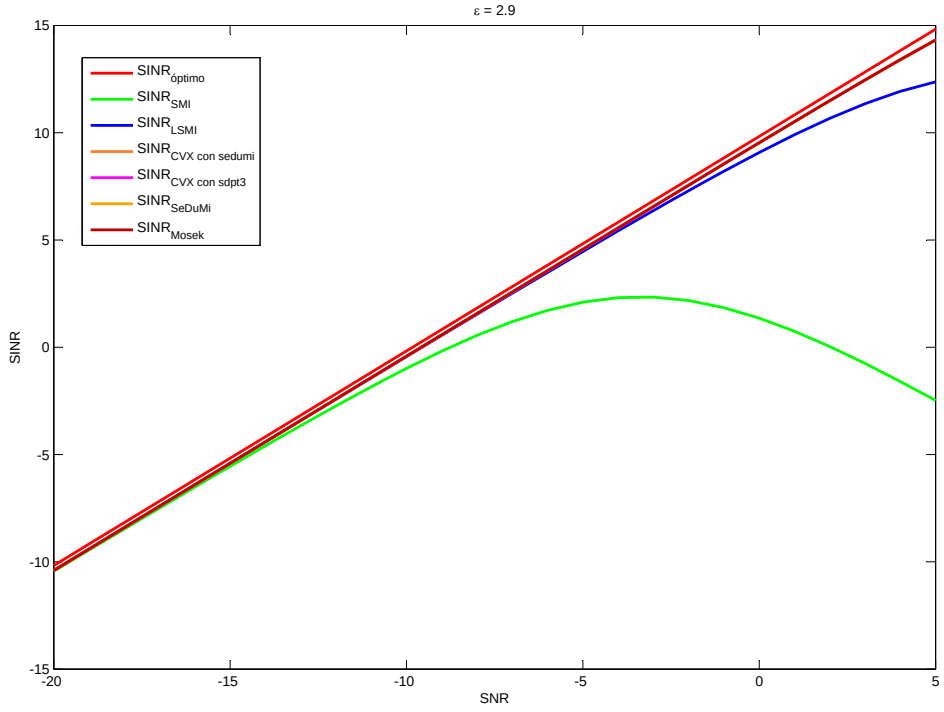

Figura 6.6: SINR vs. SNR para *CVX*, *SeDuMi* y *Mosek*.

Comprobamos que para valores bajos de SNR, tanto nuestro conformador robusto como el SMI y el LSMI producen resultados muy próximos entre sí y cercanos al valor óptimo (6.28). Cuando la SNR es alta, el conformador SMI se degrada por incluir muestras de la señal en la matriz de covarianzas [Ger03, Fel94]. Esto es debido al efecto de la autoanulación comentada en §6.2.1. Cuando la potencia de la señal es alta, la perturbación en su vector de apuntamiento influye en mayor medida en la degradación de la SINR. La regularización que aporta el método de carga diagonal permite paliar este problema, y lo vemos en la figura 6.6.

Aquí es donde comprobamos que nuestro conformador robusto se mantiene en valores próximos al óptimo cuando aumenta la potencia de la señal.

6.4.4.2 Gráfica de *Matlab Optimization Toolbox*

Esta simulación incluye las diferentes opciones con las que he probado el optimizador `fmincon` de *Matlab Optimization Toolbox*. He incluido en estas

gráficas la respuesta con *SeDuMi* para saber cuándo tenemos soluciones correctas de nuestro conformador robusto.

6.4.4.2.1 Descripción directa

La figura 6.7 muestra el comportamiento de la implementación descrita en §6.3.4.1.

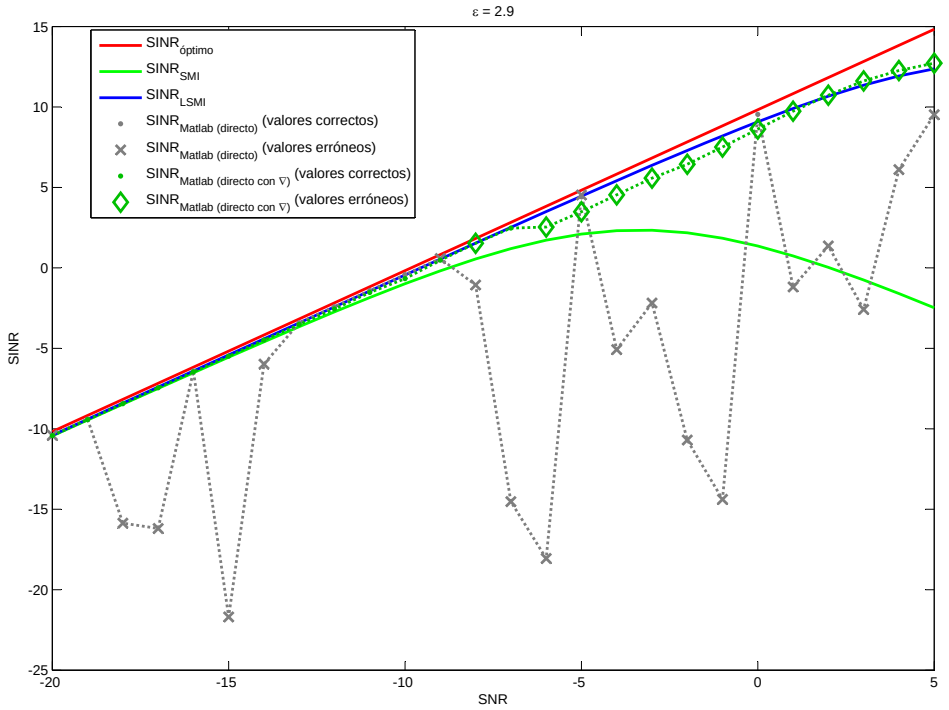

Figura 6.7: SINR vs. SNR para *Matlab Optimization Toolbox*, con implementación directa.

- (·,×) Opción directa sin gradiente
  En la gráfica que no utiliza el gradiente, vemos una solución correcta (·) por encima del optimizador con gradiente, con SNR= 0dB, pero se dan menos casos de soluciones buenas que en el siguiente caso. Sin embargo, los resultados correctos son similares a los obtenidas con *SeDuMi*. El principal inconveniente es que cuando `fmincon` no llega a una solución viable los resultados (×) se alejan demasiado de los valores correctos.

- (·, ◇) Opción directa con gradiente
  Aunque ambas gráficas dan errores para bastantes valores de SNR, cuando se incluye la información de los gradientes se produce un aumento del número de soluciones convergentes (·). En la realización de

la figura se pasa de 7 a 13 soluciones buenas. Además, los valores en los que `fmincon` devuelve error de viabilidad (◇) no se desvían demasiado de la trayectoria ascendente, al contrario que ocurre en el caso anterior. Las soluciones correctas (·) se dan sobre todo para valores bajos de SNR, y en muchos casos no son iguales a las obtenidas con *SeDuMi*. De hecho, en la realización de la figura 6.7 están por debajo del conformador LSMI (entre el LSMI y el SMI, como en la figura 6.3 de SINR vs. ε). El valor más alto de SNR para el que obtenemos una respuesta viable es para SNR$= -7$.

Dentro de la opción directa para trabajar con `fmincon` llegamos a la conclusión de que añadir el gradiente nos da más seguridad en la respuesta porque aunque devuelva resultado erróneo no se alejará demasiado del valor correcto. Sin embargo, perdemos precisión. Cuando no incluimos el gradiente se dan menos casos con éxito pero cuando se consiguen son mejores.

Si revisamos §6.4.3.2.1 confirmamos esta conclusión, ya que los puntos correctos de la gráfica sin gradiente en la figura 6.3 son los que marcan una curva similar a la de las herramientas convexas del apartado §6.4.3.1. Igual que en este caso, el inconveniente es que las soluciones malas son más y peores.

6.4.4.2.2 Planteamiento estándar SOCP

En este apartado se muestra el comportamiento de *Matlab Optimization Toolbox* para resolver el problema de acuerdo con la descripción de §6.3.4.2. El resultado está en la figura 6.8.

- (·,▽) Opción estándar sin gradiente
  El hecho de presentar este planteamiento, sin necesidad de especificar los gradientes ni el hessiano, ya produce unos resultados bastante precisos. Aunque en la mayoría de los puntos el valor devuelto es correcto (·), para los valores más altos de SNR el algoritmo se sale antes de llegar a la solución definitiva (▽). Sin embargo, los valores devueltos son prácticamente los mismos que la respuesta de *SeDuMi*, por lo que podemos considerarlos correctos.

- (−) Opción estándar con gradiente
  Cuando añadimos información sobre gradientes no encontramos ningún error y la respuesta es igual a la de *SeDuMi*. Esta es la mejor opción para utilizar `fmincon`.

- (·,○) Opción estándar con hessiano
  Si usamos la versión analítica del hessiano, el resultado es la peor opción dentro de este planteamiento. Aparecen bastantes errores (○).

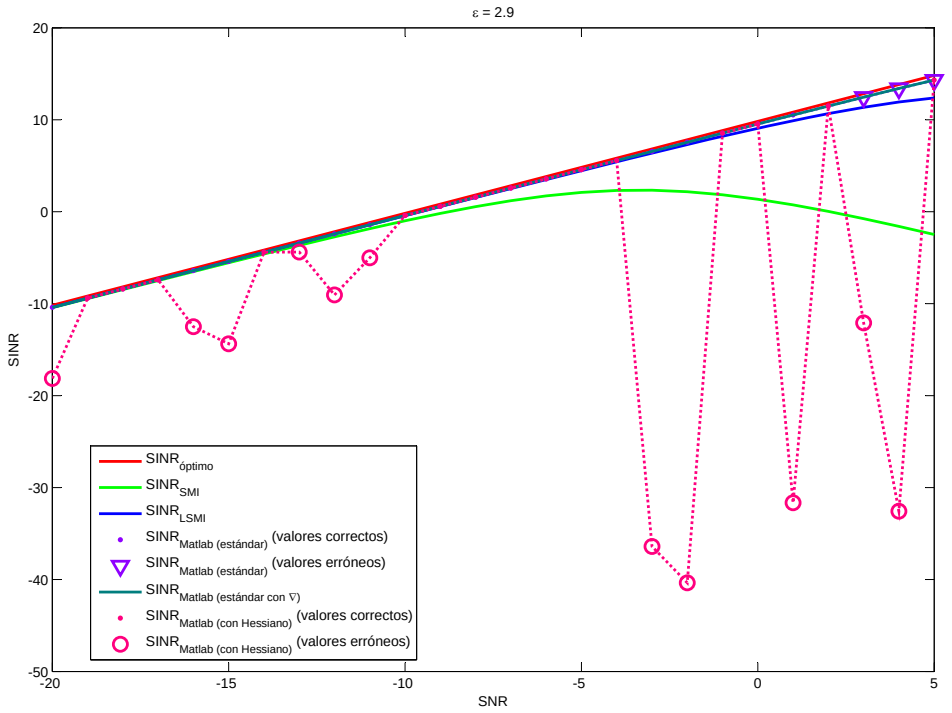

Figura 6.8: SINR vs. SNR para *Matlab Optimization Toolbox*, con planteamiento estándar SOCP.

De nuevo llegamos a la misma conclusión que en §6.4.3.2 con este problema, y de nuevo comprobamos que la expresión del hessiano está bien calculada, porque produce soluciones correctas. Sin embargo, es preferible no añadir esta segunda derivación.

El planteamiento estándar SOCP da unos resultados mucho mejores que el planteamiento directo, y dentro de esta solución hay que elegir incorporar únicamente la expresión de los gradientes.

6.4.4.3 Gráfica de *Mathematica*

En la figura 6.9 vemos que los valores correctos de *Mathematica* son iguales que los obtenidos con *SeDuMi*, aunque el problema es que cuando el optimizador no converge, las soluciones son muy malas. En este aspecto, la codificación directa se comporta igual que la codificación estándar SOCP, aunque en el primer caso el número de errores es bastante mayor.

Antes de ejecutar esta simulación hay que tener en cuenta el resultado de la figura 6.5, por la que hemos visto que es aconsejable utilizar un $\varepsilon = 2$.

Como conclusión de esta prueba, la opción de plantear el problema como un SOCP en lugar de codificarlo directamente produce una mejora significativa, aunque el optimizador no converge a valores correctos para cualquier

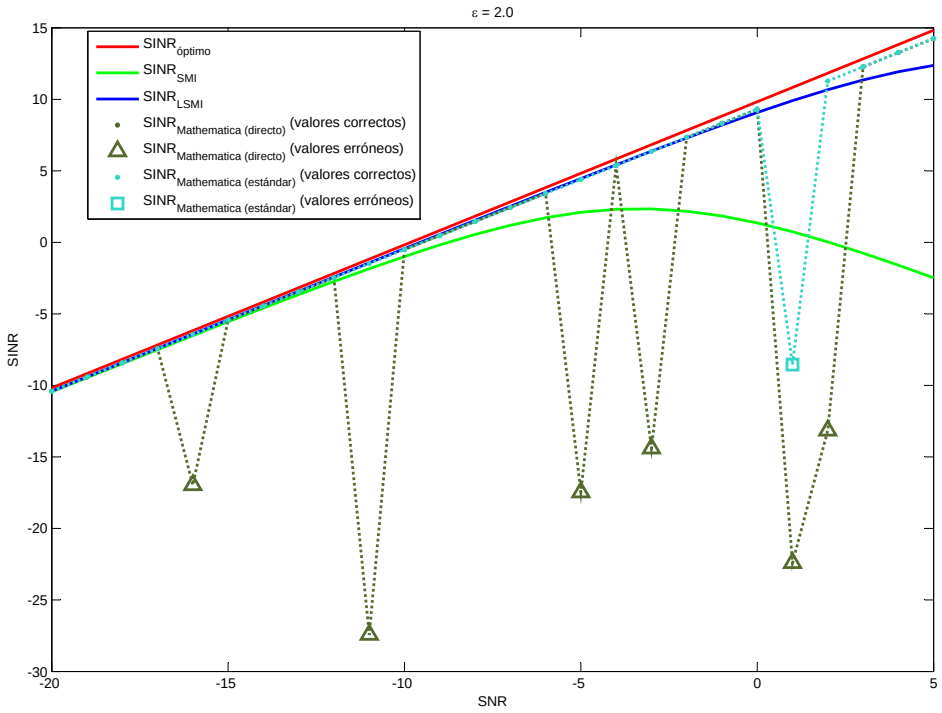

Figura 6.9: SINR vs. SNR para *Mathematica*.

SNR.

6.4.5 Tiempos de ejecución

En este ensayo se aumenta el número de antenas para comparar el tiempo de ejecución de cada herramienta.

6.4.5.1 Gráficas de *CVX*, *SeDuMi* y *Mosek*

Antes de realizar esta prueba, para fijar los valores de ε y SNR, revisamos las conclusiones de §6.4.3.1, y utilizamos $\varepsilon = 2{,}9$. Para SNR elegimos 0dB seleccionado de la figura 6.6, en este caso sin requerir ninguna especificación en concreto, ya que estas herramientas no dan errores para nuestro problema. Estos valores son sólo una referencia para el caso en que $N = 10$ antenas. Al aumentar N la dimensión del vector de apuntamiento es mayor, y la región de incertidumbre también lo será. Pero como no queremos que los optimizadores devuelvan errores utilizaremos estos valores. En la figura 6.10 se muestra la comparativa.

La primera conclusión es que la opción más lenta es *CVX* con el *solver SDPT3*. Las otras tres opciones dan resultados parecidos entre sí, y en los valores más altos la opción más rápida es *CVX* con el *solver SeDuMi*. Cu-

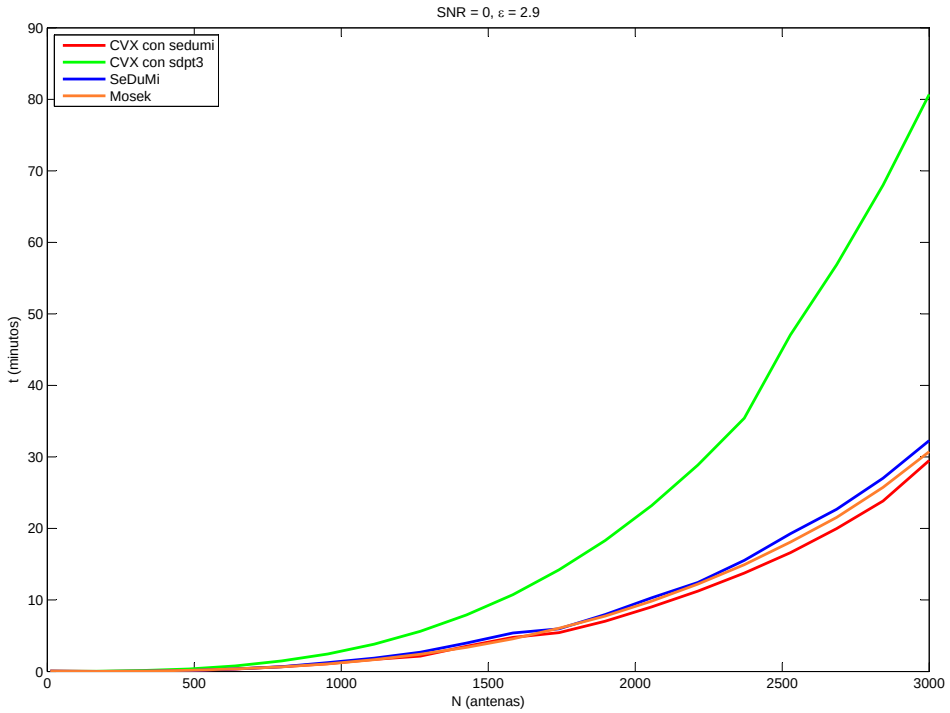

Figura 6.10: Tiempo de ejecución vs. Número de antenas para *CVX*, *SeDuMi* y *Mosek*.

riosamente, hay valores para los que es más rápido que utilizar directamente *SeDuMi*.

Esta gráfica tiene un valor comparativo entre las herramientas, pero su utilidad práctica no es mucha, por mostrar tiempos de ejecución de varios minutos. La prueba se ha hecho desde un ordenador portátil con procesador Intel Pentium Dual Core a 2.1 GHz.

Probando valores más altos de N, *CVX* termina con errores de memoria para $N = 6000$ antenas, aunque el motivo está en el ordenador utilizado y no en una limitación en el *software*.

Con *Mosek* he intentado hacer la prueba de carga (aumentar el número de antenas hasta que falle) y he obtenido valores correctos hasta $N = 6000$ antenas, con aproximadamente 4 horas de ejecución. Con 10000 antenas el programa termina con error de memoria.

Si nos preocupamos de estudiar las herramientas con vistas a aplicaciones prácticas, conviene fijarnos en la zona más baja de N por presentar velocidades de ejecución utilizables. En la figura 6.11 el número máximo de antenas es 200.

La herramienta más rápida en esta zona es *Mosek*, con 61ms para $N = 10$ antenas. Después tenemos *SeDuMi*, con 110ms. En este caso, el tiempo más

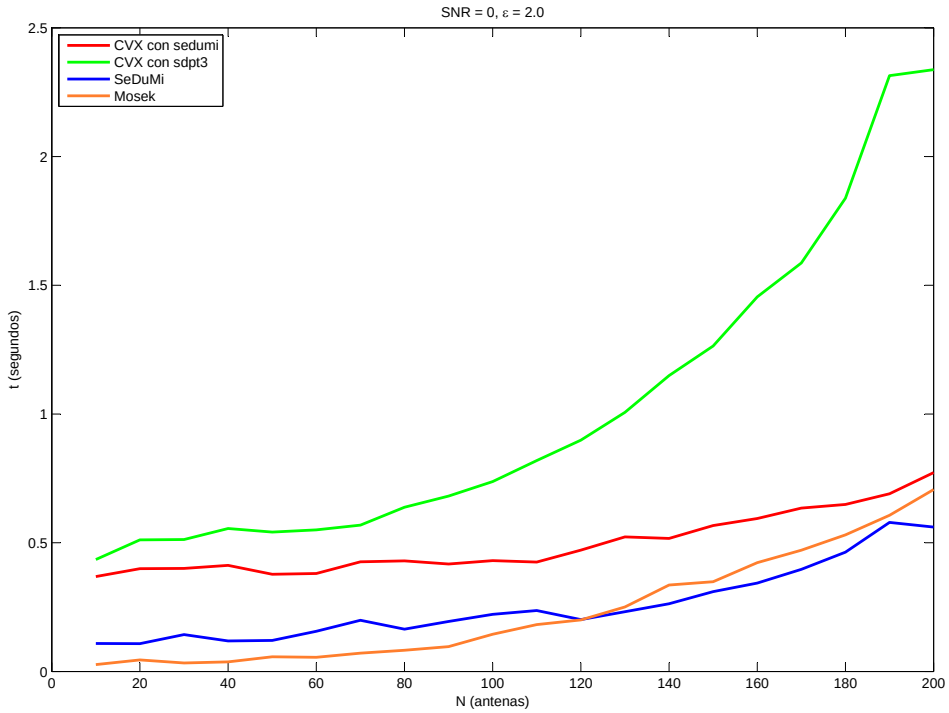

Figura 6.11: Tiempo de ejecución vs. Número de antenas para *CVX*, *SeDuMi* y *Mosek*.

rápido de *CVX* con *SeDuMi* como *solver* sí es más lento que la marca de *SeDuMi* sin modelar, con 348ms. La opción más lenta es *CVX* con *SDPT3*, cuyo mejor tiempo es 419ms.

6.4.5.2 Gráfica de *Matlab Optimization Toolbox*

Antes de realizar esta simulación hay que revisar las gráficas anteriores para asignar el valor de SNR y sobre todo, de ε a utilizar. De la figura 6.7 deducimos que nos conviene utilizar una SNR entre -13dB y -10dB para evitar errores, y de la figura 6.8 los márgenes libres de errores están entre -10dB y -4dB. Por tanto, el valor elegido es de -10dB.

Para el valor de ε, la figura 6.4 únicamente nos aconseja evitar los valores $\varepsilon = 1{,}5$, $\varepsilon = 1{,}9$ y $\varepsilon = 2{,}6$ en la codificación directa como SOCP. Teniendo esto en cuenta, buscamos valores sin errores en la figura 6.3, y elegimos el valor más alto disponible, que es $\varepsilon = 2{,}9$.

Las medidas de la figura 6.12 muestran que este *software* es mucho más lento que las herramientas específicas de optimización convexa del apartado anterior. Éste es el principal inconveniente de las herramientas generales de optimización, comentado en el capítulo 5. Este tipo de programas no aprovechan las ventajas que ofrecen los problemas convexos y nos encontramos con

una situación similar a la que tendríamos resolviendo sistemas de ecuaciones lineales con técnicas de resolución de sistemas no lineales de ecuaciones.

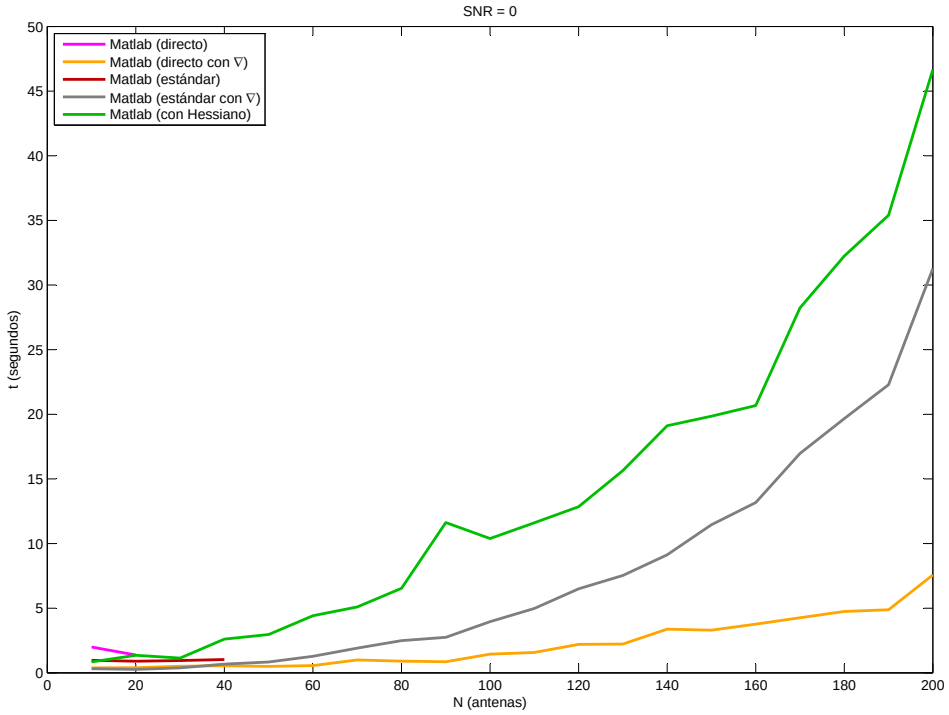

Figura 6.12: Tiempo de ejecución vs. Número de antenas para *Matlab Optimization Toolbox*.

Si tenemos que utilizar esta herramienta para el conformador de este tema, nos conviene la opción del planteamiento estándar SOCP con la descripción analítica de los gradientes. En caso contrario nos arriesgamos demasiado a no llegar a soluciones convergentes. Vemos que la opción del hessiano devuelve un error para $N = 80$, y en el resto de valore sí converge, aunque en $N = 160$ y $N = 190$ le cuesta varios minutos.

He podido hacer pruebas de carga y el programa termina con error de memoria a partir de 353 antenas. Vemos que bastante lejos de las 6000 antenas con *Mosek*.

6.4.5.3 Gráfica de *Mathematica*

Los resultados de la figura 6.13 muestran que esta herramienta es la más lenta de las que he probado. La principal desventaja viene de que se trata de un *software* con un enfoque más orientado a su capacidad de tratar con lenguaje simbólico en matemáticas que a producir el máximo rendimiento numérico. Por otro lado, añadimos la desventaja de utilizar el complemento *Mathematica Symbolic Toolbox for MATLAB*, aunque el retardo que pueda

añadir esta función es despreciable frente a los tiempos mostrados en la figura 6.13.

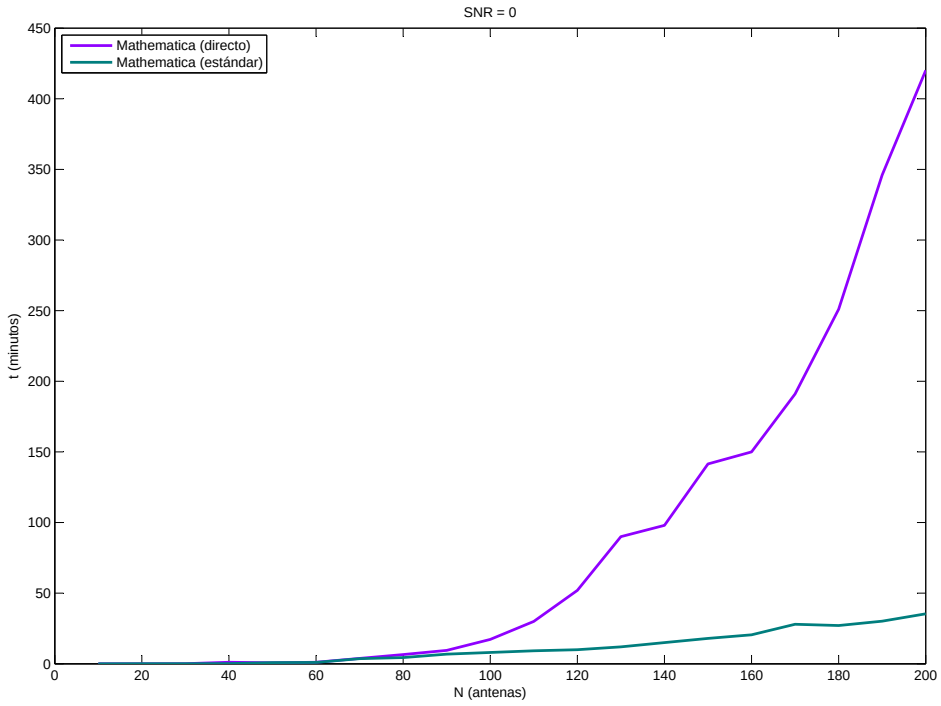

Figura 6.13: Tiempo de ejecución vs. Número de antenas para *Mathematica*.

Por otro lado, hemos visto hasta ahora que la selección del SNR y ε nos pueden llevar a resultados con errores. Revisando la figura 6.9 vemos que con SNR=0dB la ejecución con 10 antenas no da errores. Con este valor, vamos a la figura 6.5 y en consecuencia utilizamos $\varepsilon = 2$.

6.4.6 Diagramas de radiación

La última prueba tiene un objetivo más intuitivo que cuantificable, y consiste en mostrar el diagrama de radiación de algunas de las soluciones vistas hasta ahora.

6.4.6.1 Gráfica con 10 antenas

En la figura 6.14 se muestra el diagrama de radiación comparando todas las herramientas que hemos visto con SNR=0dB y $\varepsilon = 2$.

Todas las gráficas son prácticamente iguales, exceptuando la codificación directa de *Matlab Optimization Toolbox* y la codificación estándar SOCP con hessiano. Para este ejemplo no nos valen las figuras 6.7 ni 6.8 de SINR vs. SNR, porque éstas se han tomado con $\varepsilon = 2{,}9$ y en este ejemplo el valor es

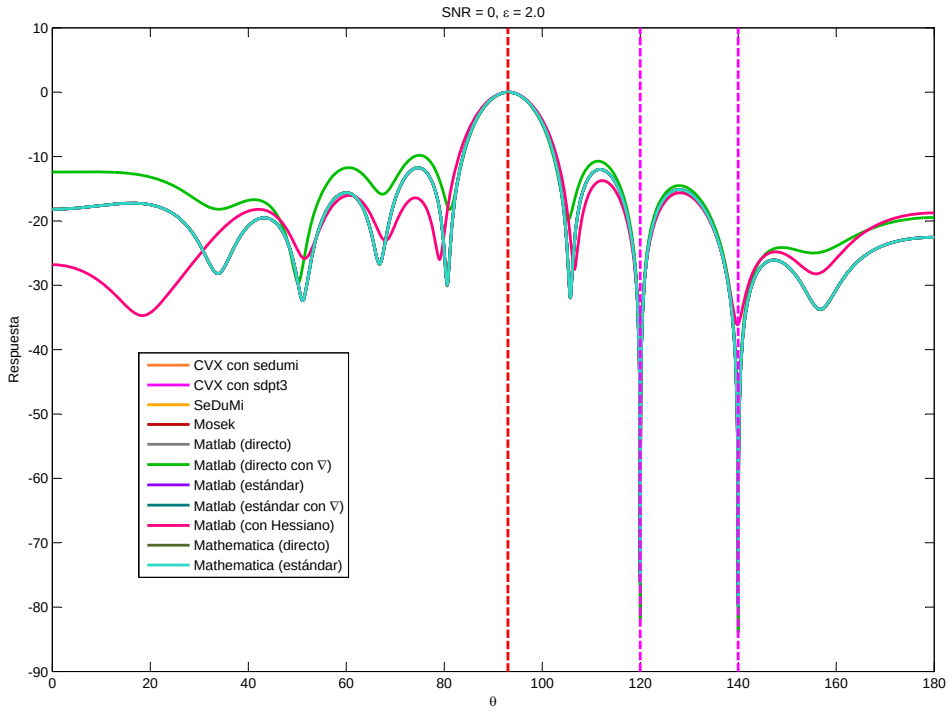

Figura 6.14: Diagrama de radiación de 10 antenas con SNR=0dB y $\varepsilon = 2$.

2. Tampoco valen las figuras 6.3 ni 6.4 de SINR vs. ε porque se hicieron con SNR=-10dB y en este caso es 0dB. Por eso, las repetimos en la figura 6.15 y comprobamos que cuando $\varepsilon = 2$ la gráfica de SINR vs. SNR muestra errores en las opciones que hemos visto, y para SNR=0dB la gráfica de SINR vs. ε da errores en $\varepsilon = 2$.

Sin embargo, a pesar de dar errores, el diagrama de radiación no se diferencia demasiado de la solución correcta, como vemos en la figura 6.14. Las líneas verticales marcan la dirección de la señal útil y las señales interferentes. Comprobamos que el diagrama es coherente con las especificaciones, ya que apunta hacia la señal indicada y atenúa significativamente las direcciones de las interferencias.

6.4.6.2 Gráfica con 100 antenas

En este caso, para ahorrar tiempo y evitar errores, mostraré únicamente el resultado con *Mosek*, que es la herramienta más rápida.

La relación señal-ruido elegida es SNR=0dB. Con este dato, la elección óptima de ε se obtiene de la gráfica 6.16, y es $\varepsilon = 9{,}7$. Para valores mayores de 10 el problema es inviable.

El diagrama de radiación resultante lo tenemos en la figura 6.17, donde se incluyen el conformador óptimo, el SMI y el LSMI. Se comprueba que las

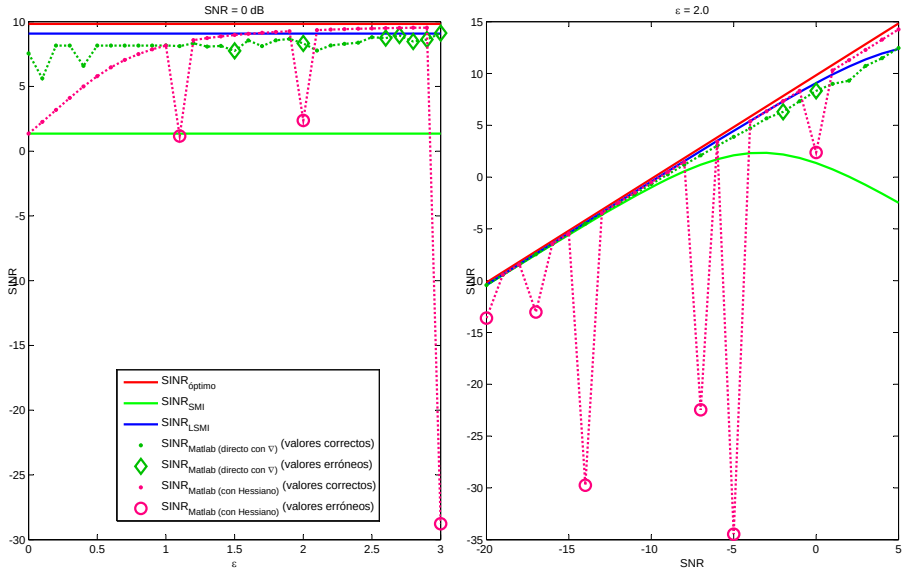

Figura 6.15: Parámetros de decisión para el diagrama de radiación con 10 antenas.

opciones no robustas tienen una forma muy diferente de la óptima, mientras que la solución robusta se aproxima al óptimo con mucha precisión. La principal diferencia es un factor de escala que está corregido en la figura, ya que nuestro problema no impone que el diagrama sea igual a 1 en la dirección apuntada. El motivo es que las restricciones no lineales de nuestro problema tienen que ser de desigualdad para que sea convexo. Este factor no afecta al funcionamiento de la agrupación, pues ya hemos estudiado en la asignatura de transmisión y propagación que habitualmente los diagramas de radiación se representan normalizados con respecto la dirección apuntada.

Pero además, podemos comprobar nuestra medida de calidad en la figura 6.18, donde apreciamos que el conformador robusto se diferencia notablemente de las soluciones SMI y LSMI especialmente para valores altos de SNR, resultando más eficiente frente a la autoanulación.

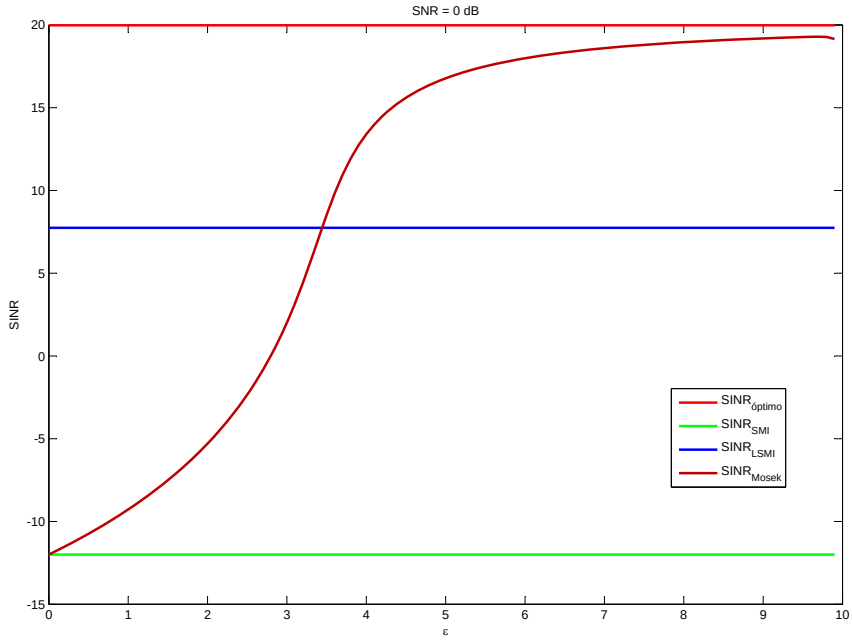

Figura 6.16: SINR vs. ε para N=100 antenas, utilizando *Mosek*.

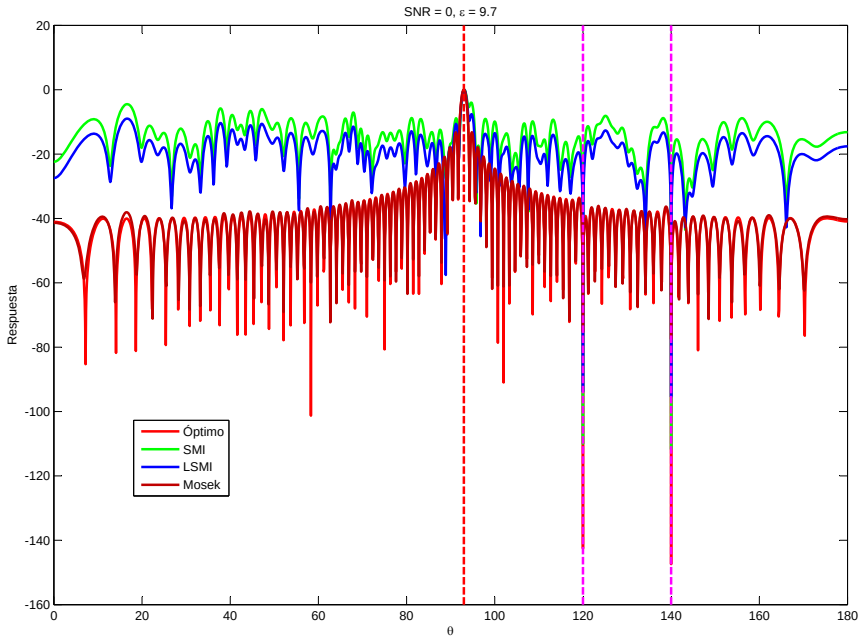

Figura 6.17: Diagrama de radiación para N=100 antenas, utilizando *Mosek*.

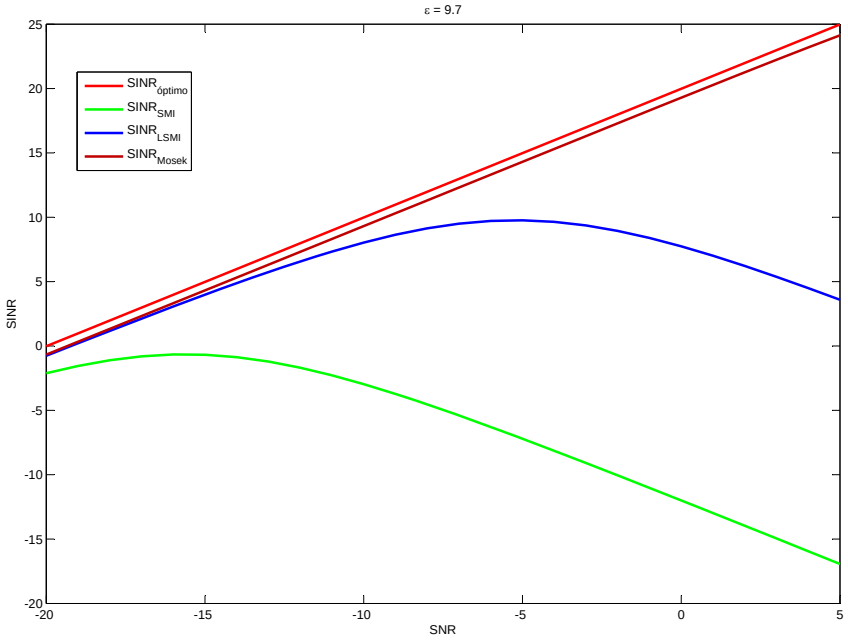

Figura 6.18: SINR vs. SNR para N=100 antenas, utilizando *Mosek*.

6.5 Conclusiones

En este capítulo hemos aplicado un conjunto representativo de las herramientas descritas en el capítulo Capítulo 5 al problema de conformación de haz. Hemos comenzado por revisar las diferentes tendencias de investigación que hay publicadas sobre este tema, con diferentes aplicaciones en recepción y en transmisión. La presentación de las diferentes opciones me ha servido para comprender la conexión existente entre este capítulo y el siguiente, que también consiste en una aplicación de las agrupaciones de antenas.

La comparativa de herramientas se ha reflejado en todos los pasos del problema. Respecto a la implementación, la conclusión es que disponemos de diferentes enfoques dependiendo de los datos que tengamos sobre el problema. Desde §2.1 he comentado que la parte más laboriosa de la optimización convexa está en la definición del problema y la manera de convertirlo en un formato convexo. Si no somos capaces de encontrar un problema convexo equivalente, tenemos la opción de utilizar *Matlab Optimization Toolbox* o *Mathematica*, pero hemos visto que nos podemos encontrar con soluciones erróneas. Las probabilidades de éxito aumentarán si las funciones utilizadas son derivables y conocemos su expresión analítica, aunque en este caso tampoco está asegurado encontrar solución. Si conseguimos plantear nuestro problema en forma convexa, pero no encontramos exactamente la forma de expresarlo en alguno de los formatos estándar descritos en la figura 3.1 y en general, en el tema 3, entonces nuestra mejor opción será buscar la forma de cumplir las normas de programación convexa disciplinada [Gra06, Gra11], o buscar otra herramienta de modelado como las descritas en §5.3. En este caso tenemos que conocer las funciones convexas con las que puede trabajar *CVX*, y la forma de combinarlas teniendo en cuenta la teoría de los capítulos 2.3 y 2.2, para conseguir describir nuestro problema. Por último, y avanzando un paso más en orden de complejidad, si somos capaces de plantear nuestro problema en un formato estándar, entonces podemos conseguir mayor velocidad de procesamiento con las herramientas de resolución como *SeDuMi* o *Mosek*. En este último caso, además de ser el más rápido con nuestro ejemplo, podríamos sacarlo de *Matlab* y utilizarlo en un programa hecho a medida por ejemplo en C, para aplicaciones prácticas.

Hemos visto que *Matlab Optimization Toolbox* y *Mathematica* son más intuitivas, y nos permiten una descripción de las funciones prácticamente como las planteamos inicialmente, aunque los resultados han demostrado que esta facilidad tiene un coste elevado en términos de fiabilidad. El planteamiento con *CVX* también es bastante intuitivo, aunque requiere un grado mayor de rigidez respecto a las normas de definición. La utilización directa de los *solvers SeDuMi* y *Mosek* necesitan trabajo para conseguir el problema en formato estándar.

Respecto a la calidad de los resultados, Las herramientas convexas dan valores prácticamente iguales. El problema lo encontramos en *Matlab Op-*

timization Toolbox y en *Mathematica*. El primero aporta soluciones menos precisas, pero cuando da errores, estos no se alejan demasiado del valor correcto. El segundo da soluciones iguales a las herramientas convexas, pero cuando no converge los resultados son inaceptables.

Capítulo 7

Canales MIMO

Este capítulo contiene otra aplicación de las herramientas de optimización convexa: el diseño de las matrices de precodificación y detección lineal en sistemas con múltiples antenas en transmisión y en recepción. Para el desarrollo de este tema me he basado en la tesis doctoral de Daniel Pérez Palomar [Pal03a], que unifica diferentes criterios de calidad para el diseño de la estructura del transmisor y el receptor. En mi caso he seleccionado dos de los criterios utilizados en [Pal03a] y he procurado que mi trabajo aportara un beneficio sobre las soluciones de dicha tesis en lugar de intentar simplemente reproducir sus resultados. La mayor parte de los casos de [Pal03a] se resuelven con algoritmos de *water-filling* adaptados a cada ejemplo y se pueden desarrollar sin necesidad de utilizar herramientas comerciales de optimización, por lo que la aportación de estas se tiene que centrar en aquellos ejemplos en los que [Pal03a] no da una solución cerrada. Por este motivo me pareció que era interesante implementar una técnica que se estudia en la citada tesis para reducir los picos de señal que se producen en la modulación OFDM, que conocemos como PAR[1] o PAPR[2]. En este caso las herramientas de optimización convexa nos permiten añadir nuevas restricciones sin necesidad de volver a diseñar el modelo inicial por completo.

Para revisar el esquema de este capítulo, comenzaremos con el contexto del problema tratado describiendo el modelo del canal y la formulación. Se incluye un repaso teórico de temas que hemos visto en la asignatura de comunicaciones móviles, aunque con la perspectiva particular de nuestro problema en concreto. Estos aspectos se tratan en §7.2 y §7.3 siguiendo la línea argumental de [Pal03a].

Del mismo modo que en el capítulo anterior, en §7.4 se describen las características del procedimiento de implementación para varias herramientas de optimización seleccionadas con los mismos criterios que en el capítulo anterior, salvo que en este tema, en vista de que en la aplicación de confor-

[1] *Peak to Average Ratio*
[2] *Peak to Average Power Ratio*

mación de haz los programas de optimización global no siempre convergían a resultados correctos, no los he utilizado porque las conclusiones del tema 6 son suficientes para comprender su funcionamiento. En definitiva, en este capítulo he utilizado un sistema de modelado (*CVX*) y dos *solvers* (*SeDuMi* y *Mosek*). En §7.4 se detallan las operaciones necesarias para convertir la formulación de [Pal03a] en un formato de problema SOCP, para lo que ha sido necesaria toda la teoría estudiada en la primera parte de este trabajo. El apartado §7.5 contiene un esquema de los bloques de código que componen la simulación, y la sección §7.6 proporciona los resultados obtenidos de las simulaciones. Por último, se extraen las conclusiones más importantes de este tema en §7.7.

7.1 Motivación

En general, se puede hablar de sistemas MIMO no sólo en el caso de múltiples antenas, sino para una gran variedad de sistemas que se pueden modelar como esquemas con múltiples entradas y salidas. Por ejemplo, se pueden estudiar con este planteamiento sistemas de transmisión DSL[3] con múltiples cables entre los que hay acoplamiento, o incluso sistemas selectivos en frecuencia con una sola antena en transmisión y en recepción, tomando como entradas y salidas vectores en la dimensión de los retardos.

Este capítulo se hace especialmente interesante porque contiene un problema de optimización convexa tratado en la asignatura de comunicaciones móviles. También está relacionado con el capítulo anterior, ya que podemos considerar algunas aplicaciones de conformación de haz como un caso particular de sistema MIMO.

7.2 Modelo del canal

La transmisión con sistemas MIMO aprovecha características del canal como la propagación multitrayecto para aumentar la eficiencia espectral por medio del aumento de dimensiones en el dominio espacial, de manera semejante al avance que hasta su utilización había supuesto el aprovechamiento de las dimensiones en los dominios del tiempo y la frecuencia. Su utilización permite establecer varios flujos simultáneos de datos multiplexados en el espacio. Una de las características de este tipo de canales es su naturaleza matricial, y esto es lo que permite representar diferentes canales físicos de forma compacta y unificada. Si tenemos n_t antenas en transmisión y n_r antenas en recepción, el modelo es

$$\boldsymbol{y} = \boldsymbol{H}\boldsymbol{s} + \boldsymbol{n} \tag{7.1}$$

[3] *Digital Subscriber Line*

donde $\boldsymbol{s} \in \mathbb{C}^{n_t}$ es el vector transmitido, $\boldsymbol{H} \in \mathbb{C}^{n_r \times n_t}$ es la matriz del canal, $\boldsymbol{y} \in \mathbb{C}^{n_r}$ es el vector recibido, y $\boldsymbol{n} \in \mathbb{C}^{n_r}$ es el vector de ruido e interferencias. En nuestro caso únicamente he utilizado ruido blanco gaussiano circularmente simétrico, con matriz de covarianzas $\boldsymbol{R}_n \in \mathbb{C}^{n_r \times n_r}$. En estas condiciones, esta matriz es igual a la potencia de ruido multiplicada por una matriz identidad, pero se mantiene la utilización de $\boldsymbol{R}_n$ para que la formulación sea válida en el caso en que $\boldsymbol{n}$ pueda incluir interferencias.

Hasta que lleguemos a las simulaciones, las dimensiones de los vectores corresponden a una transmisión simple, es decir, de un sólo símbolo por cada componente vectorial. La transmisión de un conjunto de N_{sym} símbolos es inmediata cambiando los vectores por matrices, es decir $\boldsymbol{s} \in \mathbb{C}^{n_t} \rightarrow \boldsymbol{s} \in \mathbb{C}^{n_t \times N_{sym}}$ en transmisión, y $\boldsymbol{y} \in \mathbb{C}^{n_r} \rightarrow \boldsymbol{y} \in \mathbb{C}^{n_r \times N_{sym}}$ en recepción.

Para hacer que este modelo multiplicativo pueda reflejar canales selectivos en frecuencia, podemos ampliar el número de dimensiones de los vectores, incluyendo en cada dimensión una secuencia de símbolos consecutivos. Otra opción es hacer esto en el dominio de la frecuencia, como ocurre en nuestras simulaciones, que consisten en un sistema MIMO-OFDM en el que cada componente de la transformada de Fourier del canal se considera también como un canal en paralelo con el resto de componentes en frecuencia, como en la figura 7.1

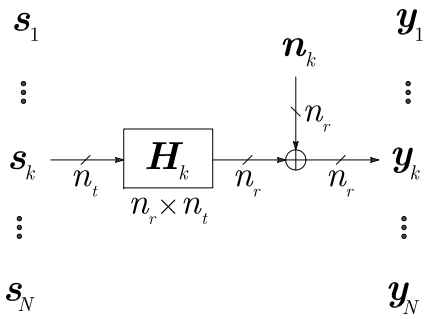

Figura 7.1: Modelo de múltiples canales MIMO

Si tenemos N portadoras, y el subíndice k indica la componente en frecuencia del canal, el modelo es el siguiente

$$\boldsymbol{y}_k = \boldsymbol{H}_k \boldsymbol{s}_k + \boldsymbol{n}_k$$

7.2.1 Ganancias en canales MIMO y sus propiedades

Las dimensiones múltiples en ambos extremos del enlace ofrecen importantes mejoras en términos de eficiencia espectral y fiabilidad del enlace. Estas mejoras se reflejan en parámetros que se pueden medir como ganancia de

conformación, de diversidad y de multiplexado. Las ganancias de conformación y de diversidad no son exclusivas de los canales MIMO, y existen también en canales SIMO[4] y MISO[5]. La ganancia de multiplexado es una característica única de los canales MIMO.

- La ganancia de conformación es el aumento de SINR obtenido al combinar de forma coherente las señales al utilizar dimensiones múltiples en transmisión o en recepción. En las curvas de BER respecto a la potencia transmitida o recibida, la ganancia de conformación se caracteriza por un desplazamiento de la gráfica debido a la ganancia en SINR.

- La ganancia de diversidad es la mejora en la fiabilidad del enlace por recibir réplicas de la señal de información a través de diferentes enlaces con características independientes respecto al desvanecimiento. La diversidad está relacionada con la naturaleza aleatoria del canal. En las gráficas de BER respecto a la potencia transmitida, esta ganancia se refleja como un aumento en la pendiente de la curva en la zona de BER baja. La idea, tal como vimos en la asignatura de comunicaciones móviles, es que, si hay varios enlaces, es más probable que en cada instante haya al menos uno libre de desvanecimiento, por lo que las dimensiones múltiples reducen las fluctuaciones de la señal recibida eliminando los desvanecimientos profundos.

- La ganancia de multiplexado es el incremento en velocidad obtenido por tener dimensiones múltiples en ambos extremos del enlace. Esto se consigue utilizando varios subcanales paralelos dentro del canal MIMO. Estos subcanales se conocen también como automodos, y producen un incremento lineal en la capacidad del canal, porque permiten la transmisión simultánea de varios símbolos.

7.2.1.1 Subcanales y flujos

Es importante diferenciar entre los conceptos de subcanales paralelos y flujos establecidos.

Los subcanales paralelos se obtienen al descomponer la matriz $\boldsymbol{H}$ en valores singulares, o la matriz cuadrada $\boldsymbol{H}^H\boldsymbol{H}$ en autovalores, y por eso se llaman también automodos del canal. Los valores singulares indican la ganancia de amplitud de cada subcanal, y los autovalores de la matriz cuadrada reflejan las ganancias de potencia. El número de subcanales paralelos disponibles es el rango de $\boldsymbol{H}$.

Cuando hablamos de flujos establecidos en un canal nos referimos a los símbolos que transmitimos simultáneamente. Si queremos establecer más

[4] *Single Input Multiple Output*
[5] *Multiple Input, Single Output*

flujos que el número de subcanales disponibles, entonces los símbolos transmitidos serán interferentes entre sí. Por tanto, si nuestras secuencias de símbolos tienen que ser ortogonales, será conveniente utilizar un número de flujos inferior o igual al de subcanales disponibles.

7.2.2 Compatibilidad entre ganancias

La ganancia de conformación necesita la combinación de múltiples copias de la misma señal, sin tener en cuenta la estadística del canal, pero la ganancia en diversidad está relacionada con el comportamiento aleatorio. Cuando hay dimensiones múltiples en recepción se pueden conseguir ambas ganancias simultáneamente combinado coherentemente las señales recibidas. Cuando hay dimensiones múltiples en transmisión, para que haya ganancia de conformación se necesita conocer el estado del canal en el transmisor (CSIT[6]), mientras que la ganancia de diversidad se puede conseguir aunque no se conozca el canal.

Para conseguir la ganancia máxima de conformación se necesita utilizar únicamente el subcanal correspondiente al autovalor máximo. Esto impide conseguir ganancia de multiplexado.

La ganancia de diversidad y la de multiplexado se pueden conseguir simultáneamente, pero existe un compromiso entre ambas. La ganancia de diversidad está relacionada con la BER, y la de multiplexado con la velocidad de transmisión. Por tanto el compromiso entre ambas está relacionado con el compromiso de capacidad de Shannon.

7.2.3 Técnicas de transmisión

Las técnicas de comunicación existentes para transmitir en canales MIMO dependen del grado de conocimiento del estado del canal disponible en el transmisor y en el receptor, que se conoce como CSIT o CSIR[7] por sus siglas en inglés.

Para conseguir la CSIR, se transmite una secuencia conocida de entrenamiento que permita una estimación del canal. Algunos casos los hemos estudiado y simulado en el laboratorio de procesamiento digital de señales. Para conseguir la CSIT, una posibilidad es establecer un canal de realimentación desde el receptor hacia el transmisor. Esta técnica necesita emplear una parte del ancho de banda para este canal de retorno. Si la comunicación es bidireccional, otra posibilidad es estimar la información del canal de transmisión a partir de la información del canal de recepción. En un sistema dúplex por división en tiempo, TDD[8], el tiempo de coherencia nos permite conocer la separación máxima que podemos tener para que esta información

[6] *Channel State Information at the Transmitter*
[7] *Channel State Information at the Receiver*
[8] *Time Division Duplex*

sea válida. En un sistema dúplex por división en frecuencia, la separación viene marcada por el ancho de banda de coherencia [Her04].

7.2.3.1 Técnicas de transmisión MIMO sin CSIT

Las técnicas que no requieren CSIT se pueden clasificar en dos categorías: codificación espacio-tiempo y arquitecturas por capas.

- *Codificación espacio-tiempo (STC).* Aunque el transmisor no tiene información del canal, añade redundancia en la señal transmitida, no sólo en el espacio, sino también en tiempo. Esto permite al receptor recuperar la señal incluso en condiciones difíciles de propagación. Los símbolos se codifican y después se dividen en flujos que se transmiten simultáneamente. La señal recibida se decodifica con un descodificador de máxima verosimilitud. La codificación STC[9] combina los beneficios de la codificación FEC[10] y la diversidad en transmisión. Hay dos tipos principales de técnicas de codificación espacio-tiempo: en formato de rejilla, o STTC[11] y en formato de bloque, o STBC[12].

- *Arquitecturas por capas (LST).* Los códigos espacio-tiempo por capas, o LST[13], se construyen uniendo varios códigos de una dimensión. La secuencia de datos original se demultiplexa en flujos de datos y cada flujo se codifica de forma independiente sin compartir información con los demás. En el receptor se utiliza supresión y cancelación de interferencias, con lo que se pueden separar los flujos tal como se transmitieron. La complejidad en este caso es menor que el decodificador de máxima verosimilitud. En este tipo se encuentra la arquitectura BLAST[14] de los laboratorios Bell. También se pueden utilizar esquemas híbridos que combinan ambas arquitecturas.

7.2.3.2 Técnicas de transmisión MIMO con CSIT exacto

Cuando se dispone del canal en el transmisor, se puede aplicar la técnica de precodificación lineal que he simulado en este capítulo. Nos basaremos en la descomposición de la matriz del canal en sus valores singulares.

7.2.3.3 Técnicas de transmisión MIMO con CSIT parcial

Cuando conocemos datos del canal, pero el modelo no es exacto, podemos distinguir dos técnicas para tratar esta situación.

[9] *Space-Time Coding*
[10] *Forward Error Correction*
[11] *Space-Time Trellis Coding*
[12] *Space-Time Block Coding*
[13] *Layered Space-Time*
[14] *Bell Laboratories Layered Space-Time*

- *Esquemas híbridos.* La idea básica es mejorar el funcionamiento de los códigos espacio-tiempo, combinándolos con algún tipo de conformador de haz o precodificador lineal.

- *Conformación robusta.* De forma parecida a lo que hemos visto en el tema anterior, los diseños robustos se basan en hacer que el sistema funcione dentro de un determinado margen de error en la estimación del canal.

7.3 Procesamiento lineal

El receptor óptimo de un sistema de comunicaciones viene dado por el detector de máxima verosimilitud, ML[15] [Art07]. Si la estructura del canal es adecuada, el detector ML se puede implementar de forma eficiente con el algoritmo de Viterbi. Una solución práctica es el uso de técnicas de procesamiento lineal, que aunque funcionan peor que el óptimo, reducen la complejidad. Este es el caso simulado en este capítulo.

Consideramos que vamos a transmitir L símbolos simultáneamente por el canal MIMO. L es por tanto el número de flujos establecidos. Para que el funcionamiento sea aceptable sin otra codificación, consideramos que L será menor o igual que el número de subcanales del sistema MIMO. El vector transmitido utiliza una matriz de procesamiento lineal $\boldsymbol{B} \in \mathbb{C}^{n_t \times L}$ que nos permite transformar la dimensión de la señal transmitida de L a n_t

$$\boldsymbol{s} = \boldsymbol{B}\boldsymbol{x} = \sum_{i=1}^{L} \boldsymbol{b}_i x_i \tag{7.2}$$

El vector $\boldsymbol{x} \in \mathbb{C}^L$ está compuesto por símbolos de información incorrelados y normalizados en energía. Cada columna $\boldsymbol{b}_i$ de la matriz $\boldsymbol{B}$ se puede considerar un vector de conformación asociado al símbolo x_i. Cuando $L = 1$ tendremos el caso de conformación de haz en transmisión.

El receptor utiliza la matriz de ecualización lineal $\boldsymbol{A} \in \mathbb{C}^{n_r \times L}$ con lo que el vector estimado es

$$\hat{\boldsymbol{x}} = \boldsymbol{A}^H \boldsymbol{y} \tag{7.3}$$

Para el caso de MIMO-OFDM con N canales MIMO paralelos e independientes, se transmiten L_k símbolos por cada portadora, por lo que el número total de flujos establecidos es

$$L_T = \sum_{k=1}^{N} L_k$$

En este caso podemos utilizar dos opciones [Pal03b, Pal03a]:

[15] *Maximum Likelihood*

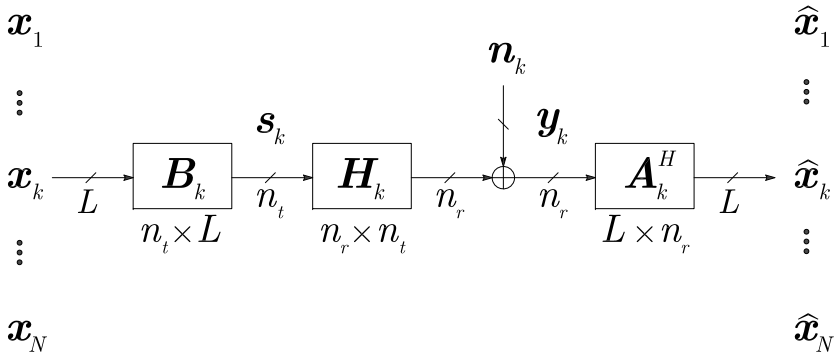

Figura 7.2: Esquema no cooperativo de canales MIMO independientes.

- Esquema no cooperativo. Los canales de cada portadora se procesan de forma independiente, tal como aparece en la figura 7.2. En cada portadora k tendremos el siguiente modelo

$$\boldsymbol{s}_k = \boldsymbol{B}_k \boldsymbol{x}_k$$
$$\hat{\boldsymbol{x}}_k = \boldsymbol{A}_k^H \boldsymbol{y}_k$$

  La potencia transmitida por cada símbolo OFDM es

$$P_T = \sum_{k=1}^{N} \mathrm{Tr}\left(\boldsymbol{B}_k \boldsymbol{B}_k^H\right)$$

- Esquema cooperativo. Corresponde a un caso general que permite el procesamiento conjunto entre distintas portadoras. El modelo de señal se consigue uniendo los vectores transmitidos por todas las portadoras, $\boldsymbol{x}^T = \left[\boldsymbol{x}_1^T, \cdots, \boldsymbol{x}_N^T\right]$, y utilizando matrices globales $\boldsymbol{B} \in \mathbb{C}^{(n_t N) \times L_T}$ y $\boldsymbol{A} \in \mathbb{C}^{(n_r N) \times L_T}$, tal como vemos en la figura 7.3. El modelo de canal es

$$\boldsymbol{H} = \mathrm{diag}\left(\{\boldsymbol{H}_k\}\right) \in \mathbb{C}^{(n_r N) \times (n_t N)}$$

El modelo no cooperativo se puede obtener como un caso particular del modelo cooperativo, en el que $\boldsymbol{B} = \mathrm{diag}\left(\{\boldsymbol{B}_k\}\right)$ y $\boldsymbol{A} = \mathrm{diag}\left(\{\boldsymbol{A}_k\}\right)$. Por tanto, el caso cooperativo es un caso general en el que no existen estas restricciones en la estructura de $\boldsymbol{A}$ y $\boldsymbol{B}$, por lo que tiene capacidad para reubicar símbolos entre distintas portadoras si algunas frecuencias dan mejores resultados que otras.

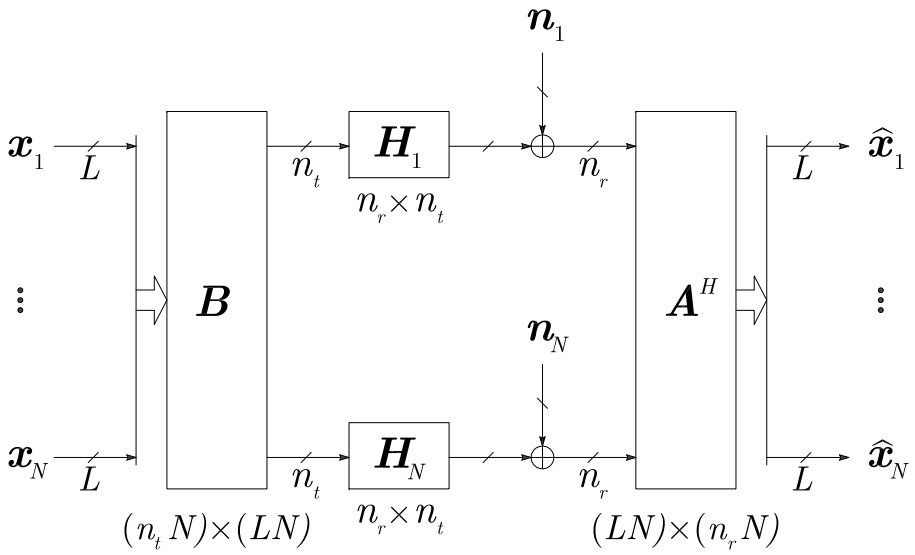

Figura 7.3: Esquema cooperativo entre canales MIMO.

7.3.1 Parámetros de calidad del sistema

Los parámetros que miden el funcionamiento de un sistema de comunicaciones son el error cuadrático medio (MSE[16]), la relación señal a ruido e interferencia (SINR), la tasa de error (BER) y la capacidad del canal.

Cuando el canal es determinista o cambia lentamente, las medidas de calidad se pueden dar con un valor instantáneo, pero si el canal es aleatorio, como suele suceder en las comunicaciones inalámbricas, las medidas se tienen que dar en forma estadística. Para caracterizar completamente una medida, se necesita la función de densidad de probabilidad, pero a efectos prácticos, hemos visto en repetidas ocasiones que basta con conocer algún parámetro de la *pdf*. Las dos formas más comunes de hacer esto son el valor medio y el valor de desconexión (*outage*). La elección de uno u otro depende de si el escenario es o no ergódico.

Cuando la transmisión es suficientemente larga como para calcular las propiedades ergódicas del canal variante, entonces conviene utilizar los valores medios de las medidas de calidad.

Cuando no se satisface la condición de ergodicidad, es mejor utilizar los valores de desconexión porque dan un resultado más realista. El valor de desconexión viene dado como el mejor valor que se puede conseguir con una probabilidad alta $(1 - P_{out})$. Por ejemplo, si medimos 100 curvas de BER, y queremos dar los valores para una probabilidad de desconexión del 5 %, entonces para cada valor de SNR tomaremos el punto en el que haya 95 valores por debajo y 5 por encima, es decir, el mejor de los 5 peores valores.

[16] *Mean Square Error*

El mismo ejemplo es válido para dar medidas de MSE. Si lo que medimos es la SINR o la capacidad, entonces daremos el punto en el que haya 95 valores por encima y 5 por debajo, de forma que el 5 % de los ensayos son peores que el valor dado. Por tanto, si el valor elegido se establece como el umbral de desconexión, entonces su probabilidad será del 5 %. Esto explica la utilización de este término.

En las simulaciones de este capítulo se muestran valores con una sola realización del canal invariante en tiempo.

7.3.2 Receptor lineal

Teniendo en cuenta el modelo definido en (7.1), (7.2) y (7.3) tenemos que

$$\hat{\boldsymbol{x}} = \boldsymbol{A}^H \left(\boldsymbol{H}\boldsymbol{B}\boldsymbol{x} + \boldsymbol{n} \right)$$

Con estos datos, podemos definir la matriz de covarianzas del error cuadrático medio

$$\begin{aligned} \boldsymbol{E}\left(\boldsymbol{B}, \boldsymbol{A}\right) &\triangleq \mathbb{E}\left\{ \left(\hat{\boldsymbol{x}} - \boldsymbol{x}\right) \left(\hat{\boldsymbol{x}} - \boldsymbol{x}\right)^H \right\} \\ &= \left(\boldsymbol{A}^H \boldsymbol{H}\boldsymbol{B} - \boldsymbol{I} \right) \left(\boldsymbol{B}^H \boldsymbol{H}^H \boldsymbol{A} - \boldsymbol{I} \right) + \boldsymbol{A}^H \boldsymbol{R}_n \boldsymbol{A} \\ &= \boldsymbol{A}^H \left(\boldsymbol{H}\boldsymbol{B}\boldsymbol{B}^H \boldsymbol{H}^H + \boldsymbol{R}_n \right) \boldsymbol{A} + \boldsymbol{I} - \boldsymbol{A}^H \boldsymbol{H}\boldsymbol{B} - \boldsymbol{B}^H \boldsymbol{H}^H \boldsymbol{A} \end{aligned} \tag{7.4}$$

Los elementos de la diagonal de $\boldsymbol{E}$ son los errores cuadráticos medios de cada flujo de datos. De la expresión (7.4) obtenemos que la matriz óptima en el receptor es [Pal03a, Pal03b, Pal05]

$$\begin{aligned} \boldsymbol{A}^* &= \left(\boldsymbol{H}\boldsymbol{B}\boldsymbol{B}^H \boldsymbol{H}^H + \boldsymbol{R}_n \right)^{-1} \boldsymbol{H}\boldsymbol{B} \\ &= \boldsymbol{R}_n^{-1} \boldsymbol{H}\boldsymbol{B} \left(\boldsymbol{I} + \boldsymbol{B}^H \boldsymbol{H}^H \boldsymbol{R}_n^{-1} \boldsymbol{H}\boldsymbol{B} \right)^{-1} \end{aligned} \tag{7.5}$$

Este resultado es el filtro lineal de mínimo error cuadrático medio, LMMSE[17] o filtro de Wiener, que minimiza simultáneamente todos los elementos de la diagonal de la matriz $\boldsymbol{E}$, con lo que dicha matriz queda

$$\begin{aligned} \boldsymbol{E}(\boldsymbol{B}) &\triangleq \boldsymbol{E}\left(\boldsymbol{B}, \boldsymbol{A}^*\right) \\ &= \boldsymbol{I} - \boldsymbol{B}^H \boldsymbol{H}^H \left(\boldsymbol{H}\boldsymbol{B}\boldsymbol{B}^H \boldsymbol{H}^H + \boldsymbol{R}_n \right)^{-1} \boldsymbol{H}\boldsymbol{B} \\ &= \boldsymbol{I} - \boldsymbol{B}^H \boldsymbol{H}^H \boldsymbol{R}_n^{-1} \boldsymbol{H}\boldsymbol{B} \left(\boldsymbol{I} + \boldsymbol{B}^H \boldsymbol{H}^H \boldsymbol{R}_n^{-1} \boldsymbol{H}\boldsymbol{B} \right)^{-1} \\ &= \left(\boldsymbol{I} + \boldsymbol{B}^H \boldsymbol{H}^H \boldsymbol{R}_n^{-1} \boldsymbol{H}\boldsymbol{B} \right)^{-1} \end{aligned} \tag{7.6}$$

Si nos fijamos en el caso en que $L = 1$, es decir, de conformación de haz, donde las matrices $\boldsymbol{A}$ y $\boldsymbol{B}$ pasan a ser vectores $\boldsymbol{a}$ y $\boldsymbol{b}$, la relación SINR se

[17] *Linear Minimum Mean Square Error*

define como

$$\text{SINR}(\boldsymbol{a}, \boldsymbol{b}) \triangleq \frac{|\boldsymbol{a}^H \boldsymbol{H} \boldsymbol{b}|^2}{\boldsymbol{a}^H \boldsymbol{R}_n \boldsymbol{a}} = \frac{\boldsymbol{a}^H \left(\boldsymbol{H} \boldsymbol{b} \boldsymbol{b}^H \boldsymbol{H}^H \right) \boldsymbol{a}}{\boldsymbol{a}^H \boldsymbol{R}_n \boldsymbol{a}}$$

De aquí obtenemos el vector óptimo que maximiza la SINR

$$\boldsymbol{a}^* = \beta \boldsymbol{R}_n^{-1} \boldsymbol{H} \boldsymbol{b}$$

Cuando tenemos varios flujos, $L > 1$, los vectores $\boldsymbol{a}$ y $\boldsymbol{b}$ pasan a ser las columnas de $\boldsymbol{A}$ y $\boldsymbol{B}$, y la solución óptima vuelve a ser el filtro de Wiener, en el que se puede identificar fácilmente el factor de escala entre (7.5) y (7.7)

$$\boldsymbol{a}_i = \beta \boldsymbol{R}_{n_i}^{-1} \boldsymbol{H} \boldsymbol{b}_i \tag{7.7}$$

donde $\boldsymbol{R}_{n_i}$ es la matriz de covarianza vista por el flujo i. También es posible incluir en el ecualizador una restricción de forzado a cero [Art07], pero no he incluido ninguna simulación con esta condición.

7.3.3 Diseño del transmisor

Para asignar la matriz de transmisión, lo más razonable es seleccionar un criterio específico a optimizar, y de acuerdo con éste se diseña el sistema. Con esto me refiero a optimizar una función objetivo que dependa de las medidas de calidad estudiadas en §7.3.1.

En [Pal03a, Pal03b, Pal05] encontramos un estudio que unifica los diferentes criterios que se pueden elegir, y su clasificación depende de la utilización de funciones convexas o cóncavas de Schur. Esto viene de la teoría de mayorización, en la que no vamos a entrar y sobre la que podemos ver una introducción en [Pal03a, Pal03b, Pal05]. Para nuestro propósito, únicamente nos hace falta saber que las funciones cóncavas de Schur nos permiten utilizar una estructura diagonal del canal, tal como vimos en la asignatura de comunicaciones móviles. Esta estructura la conseguimos con la descomposición en valores singulares. En este proyecto no utilizamos ningún criterio con funciones convexas de Schur, por lo que no voy a entrar en su solución.

Una función objetivo es un indicador del funcionamiento de un sistema, y el diseño óptimo la hace mínima. Si depende del MSE o del BER tiene que ser creciente en cada variable, y si depende de la relación SINR o de la capacidad será decreciente. En §7.3.4 veremos las expresiones de estas funciones objetivo para los criterios implementados en este capítulo.

7.3.3.1 Canal MIMO simple

En este apartado se describe el diseño del transmisor cuando hay un solo canal MIMO. Este modelo nos sirve para los cálculos en el caso del esquema cooperativo descrito al principio de §7.3.

En todos los casos suponemos que el receptor corresponde al filtro de Wiener (7.5).

$$\boldsymbol{A} = \left(\boldsymbol{H}\boldsymbol{B}\boldsymbol{B}^H\boldsymbol{H}^H + \boldsymbol{R}_n\right)^{-1}\boldsymbol{H}\boldsymbol{B}$$

Para facilitar la notación, se define la matriz cuadrada del canal del siguiente modo

$$\boldsymbol{R}_H \triangleq \boldsymbol{H}^H\boldsymbol{R}_n^{-1}\boldsymbol{H}$$

De acuerdo con esto, la matriz de MSE queda así

$$\boldsymbol{E} = \left(\boldsymbol{I} + \boldsymbol{B}^H\boldsymbol{R}_H\boldsymbol{B}\right)^{-1}$$

El problema del transmisor se reduce a obtener la matriz $\boldsymbol{B}$ óptima que minimice la función f_0 dependiente del error cuadrático medio

$$\begin{aligned} \underset{\boldsymbol{B}}{\text{minimizar}} \quad & f_0\left(\left\{\left[\left(\boldsymbol{I} + \boldsymbol{B}^H\boldsymbol{R}_H\boldsymbol{B}\right)^{-1}\right]_{ii}\right\}_{i=1}^{L}\right) \\ \text{sujeto a} \quad & \operatorname{Tr}\left(\boldsymbol{B}\boldsymbol{B}^H\right) \leq P_T \end{aligned}$$

Este problema no es convexo, pero se puede simplificar para que lo sea. En [Pal03a] se demuestra que si f_0 es una función cóncava de Schur, el sistema se puede diagonalizar, de forma que la función matricial $\left[\left(\boldsymbol{I} + \boldsymbol{B}^H\boldsymbol{R}_H\boldsymbol{B}\right)^{-1}\right]_{ii}$ se convierte en un conjunto de expresiones escalares.

La solución que se da en [Pal03a] para este caso es una matriz óptima $\boldsymbol{B}$ que como máximo tendrá rango $\check{L} \triangleq \min\left(L, \operatorname{rango}\left(\boldsymbol{R}_H\right)\right)$ con la siguiente estructura

$$\boldsymbol{B} = \boldsymbol{U}_H\boldsymbol{\Sigma}_B$$

$\boldsymbol{U}_H \in \mathbb{C}^{n_t \times \check{L}}$ tiene como columnas los autovectores de $\boldsymbol{R}_H$ correspondientes a los $\check{L}$ autovalores más grandes en orden creciente, y $\boldsymbol{\Sigma}_B \in \mathbb{C}^{\check{L} \times L}$ se rellena con columnas de ceros excepto en la diagonal de la derecha, que contiene las variables $\boldsymbol{\sigma}_{B,i}$ a optimizar

$$\boldsymbol{\Sigma}_B = \left[\boldsymbol{0} \quad \operatorname{diag}\left(\{\sigma_{B,i}\}\right)\right]$$

Esta solución es igual que la que vimos en la asignatura de comunicaciones móviles, añadiendo en este caso la posibilidad de establecer un número de flujos de datos L distinto del rango de la matriz del canal.

El proceso total de comunicación queda completamente diagonalizado, y también la matriz $\boldsymbol{E}$. Entre los L flujos establecidos, únicamente se asocian $\check{L}$ a los autovalores significativos del canal, mientras que para el resto $L - \check{L}$ no se destinará potencia en el transmisor.

En resumen, el vector recibido queda

$$\hat{\boldsymbol{x}} = \left(\boldsymbol{I} + \boldsymbol{\Sigma}_B^H\boldsymbol{D}_H\boldsymbol{\Sigma}_B\right)^{-1}\boldsymbol{\Sigma}_B^H\boldsymbol{D}_H^{1/2}\left(\boldsymbol{D}_H^{1/2}\boldsymbol{\Sigma}_B\boldsymbol{x} + \boldsymbol{w}\right)$$

donde $\boldsymbol{D}_H = \left(\{\lambda_{H,i}\}_{k=1}^{\check{L}}\right)$ contiene los $\check{L}$ autovalores más grandes de $\boldsymbol{R}_H$ en orden creciente, y $\boldsymbol{w}$ es un vector equivalente al ruido normalizado. Esto equivale a

$$\hat{x}_i = \begin{cases} 0 & 1 \leq i \leq L - \check{L} \\ \frac{\sigma^2_{B,(i-(L-\check{L}))}\lambda_{H,(i-(L-\check{L}))}}{1+\sigma^2_{B,(i-(L-\check{L}))}\lambda_{H,(i-(L-\check{L}))}} x_i + \frac{\sigma_{B,(i-(L-\check{L}))}\lambda^{1/2}_{H,(i-(L-\check{L}))}}{1+\sigma^2_{B,(i-(L-\check{L}))}\lambda_{H,(i-(L-\check{L}))}} w_i & L - \check{L} < i \leq L \end{cases}$$

La matriz de covarianza del error cuadrático medio queda

$$\boldsymbol{E} = \left(\boldsymbol{I} + \boldsymbol{\Sigma}_B^H \boldsymbol{D}_H \boldsymbol{\Sigma}_B\right)^{-1}$$

y será una matriz diagonal con los siguientes elementos

$$\mathrm{MSE}_i = \begin{cases} 1 & 1 \leq i \leq L - \check{L} \\ \frac{1}{1+\sigma^2_{B,(i-(L-\check{L}))}\lambda_{H,(i-(L-\check{L}))}} & L - \check{L} < i \leq L \end{cases} \tag{7.8}$$

Las relaciones SINR tienen el siguiente valor

$$\mathrm{SINR}_i = \begin{cases} 0 & 1 \leq i \leq L - \check{L} \\ \sigma^2_{B,(i-(L-\check{L}))}\lambda_{H,(i-(L-\check{L}))} & L - \check{L} < i \leq L \end{cases}$$

Queda claro entonces que si intentamos establecer más flujos de datos que el rango del canal, en los flujos que estén por encima tendremos un MSE=1, es decir, BER=0.5.

7.3.3.2 Múltiples canales MIMO

En el caso de MIMO-OFDM con canales independientes el filtro de Wiener para cada portadora k en el receptor será

$$\boldsymbol{A}_k = \left(\boldsymbol{H}_k \boldsymbol{B}_k \boldsymbol{B}_k^H \boldsymbol{H}_k^H + \boldsymbol{R}_{n_k}\right)^{-1} \boldsymbol{H}_k \boldsymbol{B}_k \tag{7.9}$$

Este caso corresponde al esquema no cooperativo, con procesamiento independiente para cada portadora.La matriz cuadrada del canal en el dominio de la frecuencia se define como

$$\boldsymbol{R}_{H_k} \triangleq \boldsymbol{H}_k^H \boldsymbol{R}_{n_k}^{-1} \boldsymbol{H}_k$$

Las matrices de error cuadrático tienen este valor

$$\boldsymbol{E}_k = \left(\boldsymbol{I} + \boldsymbol{B}_k^H \boldsymbol{R}_{H_k} \boldsymbol{B}_k\right)^{-1}$$

El sistema se optimiza al calcular las matrices $\boldsymbol{B}_k$ que minimizan la función f_0 del error cuadrático con restricción de potencia en el transmisor

$$\begin{aligned} \underset{\boldsymbol{B}}{\text{minimizar}} \quad & f_0\left(\left\{\left\{\left[\left(\boldsymbol{I} + \boldsymbol{B}_k^H \boldsymbol{R}_{H_k} \boldsymbol{B}_k\right)^{-1}\right]_{ii}\right\}_{i=1}^{L}\right\}_{k=1}^{N}\right) \\ \text{sujeto a} \quad & \mathrm{Tr}\left(\boldsymbol{B}_k \boldsymbol{B}_k^H\right) \leq P_T \end{aligned}$$

Igual que en el caso de un solo canal MIMO, se define $\check{L}_k$ como el número de subcanales disponibles para los L_k flujos de datos en la portadora k.

$$\check{L}_k \triangleq \text{mín}\left(L_k, \text{rango}\left(\boldsymbol{R}_{H_k}\right)\right)$$

La solución tiene la siguiente forma

$$\boldsymbol{B}_k = \boldsymbol{U}_{H_k} \boldsymbol{\Sigma}_{B_k}$$

donde $\boldsymbol{U}_{H_k} \in \mathbb{C}^{n_t \times \check{L}}$ tiene como columnas los autovectores de $\boldsymbol{R}_{H_k}$ correspondientes a los $\check{L}_k$ autovalores más grandes en orden creciente, y $\boldsymbol{\Sigma}_{B_k} \in \mathbb{C}^{\check{L}_k \times L_k}$ se rellena con columnas de ceros excepto en la diagonal de la derecha, que contiene las variables $\boldsymbol{\Sigma}_{B_k,i}$ a optimizar

$$\boldsymbol{\Sigma}_{B_k} = \left[\boldsymbol{0} \quad \text{diag}\left(\{\boldsymbol{\Sigma}_{B_k,i}\}\right)\right]$$

7.3.4 Criterios de optimización utilizados

En este apartado se dan expresiones concretas a la función objetivo f_0 de la que hablábamos en §7.3.3. Utilizaré la notación del esquema no cooperativo para no perder de vista la correcta utilización del subíndice k correspondiente a cada portadora. A partir de aquí se pueden obtener fácilmente las expresiones para el caso cooperativo eliminando el subíndice k.

Una vez más, para simplificar la notación definimos $z_{k,i} \triangleq \sigma^2_{B_k,i}$ y $\lambda_{k,i} \triangleq \lambda_{H_k,i}$, donde k es el índice correspondiente a cada portadora, e i es el índice de cada subcanal.

7.3.4.1 Suma de MSE

El primer caso de simulación que se presenta en este trabajo corresponde a la siguiente función objetivo

$$f_0\left(\{\text{MSE}_{k,i}\}\right) = \sum_{k,i} \text{MSE}_{k,i}$$

Al tratarse de una función cóncava de Schur (según se demuestra en [Pal03a]), el procedimiento óptimo es diagonalizar el canal, y teniendo en cuenta la expresión de los valores de MSE en (7.8), se asignarán los valores de $\boldsymbol{B}_k$ a partir del siguiente problema

$$\begin{aligned} \underset{z_{k,i}}{\text{minimizar}} \quad & \sum_{k,i} \frac{1}{1+\lambda_{k,i} z_{k,i}} \\ \text{sujeto a} \quad & \sum_{k,i} z_{k,i} \leq P_T \\ & z_{k,i} \geq 0 \qquad\qquad 1 \leq k \leq N,\, 1 \leq i \leq \check{L}_k \end{aligned} \tag{7.10}$$

A continuación voy a presentar la forma analítica de resolver este problema revisando la teoría de §2.4. En primer lugar, voy a asignar nombres a las variables duales del problema (7.10) asociadas a sus restricciones

$$f_1(\boldsymbol{z}) = \mathbf{1}^T\boldsymbol{z} - P_T \leq 0 \rightarrow \mu$$
$$g_{k,i}(\boldsymbol{z}) = -z_{k,i} \leq 0 \rightarrow \nu_{k,i}$$

donde $\boldsymbol{z}$ es el vector que contiene todos los $z_{k,i}$ correspondientes a cada portadora k y a cada automodo i. Siguiendo esta notación, la función lagrangiana queda como

$$L(\boldsymbol{z}, \mu, \nu_{k,i}) = \sum_{k,i} \frac{1}{1 + \lambda_{k,i} z_{k,i}} + \mu \left(\mathbf{1}^T\boldsymbol{z} - P_T\right) - \sum_{k,i} \nu_{k,i} z_{k,i}$$

Con estos datos podemos plantear las ecuaciones KKT para cumplir las condiciones de punto óptimo tal como vimos en §2.4.3.2

$$-\frac{\lambda_{k,i}}{\left(1 + \lambda_{k,i} z_{k,i}\right)^2} + \mu - \nu_{k,i} = 0 \qquad 1 \leq k \leq N,\ 1 \leq i \leq \check{L}_k \tag{7.11}$$
$$\mathbf{1}^T\boldsymbol{z} \leq P_T \tag{7.12}$$
$$z_{k,i} \geq 0 \tag{7.13}$$
$$\mu \geq 0 \tag{7.14}$$
$$\nu_{k,i} \geq 0 \qquad 1 \leq k \leq N,\ 1 \leq i \leq \check{L}_k \tag{7.15}$$
$$\mu \left(\mathbf{1}^T\boldsymbol{z} - P_T\right) = 0 \tag{7.16}$$
$$\nu_{k,i} z_{k,i} = 0 \qquad 1 \leq k \leq N,\ 1 \leq i \leq \check{L}_k \tag{7.17}$$

De la ecuación (7.11) podemos despejar $\nu_{k,i}$

$$\nu_{k,i} = \mu - \frac{\lambda_{k,i}}{\left(1 + \lambda_{k,i} z_{k,i}\right)^2} \tag{7.18}$$

Sustituyendo (7.18) en (7.17), que conocemos como condición de holgura complementaria para $\nu_{k,i}$, tenemos

$$\left(\mu - \frac{\lambda_{k,i}}{\left(1 + \lambda_{k,i} z_{k,i}\right)^2}\right) z_{k,i} = 0$$

Si vemos esta condición como en (2.11) llegamos a la siguiente descripción

$$\mu - \frac{\lambda_{k,i}}{\left(1 + \lambda_{k,i} z_{k,i}\right)^2} > 0 \Rightarrow z_{k,i} = 0$$
$$\mu - \frac{\lambda_{k,i}}{\left(1 + \lambda_{k,i} z_{k,i}\right)^2} = 0 \Rightarrow z_{k,i} > 0$$

por tanto, $z_{k,i}$ únicamente puede valer cero o el valor que haga cero a $\nu_{k,i}$, es decir

$$z_{k,i} = \begin{cases} 0 & \text{si } \mu \geq \lambda_{k,i} \\ \lambda_{k,i}^{-1/2}\mu^{-1/2} - \lambda_{k,i}^{-1} & \text{si } \mu \leq \lambda_{k,i} \end{cases}$$

Esto se resume en que el valor de $z_{k,i}$ es

$$z_{k,i} = \text{máx}\left\{0, \lambda_{k,i}^{-1/2}\mu^{-1/2} - \lambda_{k,i}^{-1}\right\} = \left(\lambda_{k,i}^{-1/2}\mu^{-1/2} - \lambda_{k,i}^{-1}\right)^{+} \tag{7.19}$$

Para calcular el valor de $z_{k,i}$ asignamos el valor de μ a través de la restricción de potencia (7.12)

$$\sum_{k,i} \text{máx}\left\{0, \lambda_{k,i}^{-1/2}\mu^{-1/2} - \lambda_{k,i}^{-1}\right\} \leq P_T$$

La expresión anterior es una función creciente y lineal por tramos respecto a $\mu^{-1/2}$, con discontinuidades en cada $\mu = \lambda_{k,i}$, por lo que es fácil buscar un algoritmo que vaya incrementando el nivel $\mu^{-1/2}$ hasta llegar a la potencia P_T, y que encontrará el valor óptimo dual μ^* en menos de $N \times L$ iteraciones. Este tipo de algoritmos se conoce como llenado de agua o *water-filling*. A partir de μ^* es inmediato obtener los $z_{k,i}^*$ óptimos con (7.19).

La condición de holgura complementaria (7.16) de la variable dual μ indica que cuando $\mu = 0$ estaremos transmitiendo por debajo de la potencia máxima, y cuando transmitimos P_T es porque $\mu > 0$.

7.3.4.2 Información mutua

El caso que vimos en clase de comunicaciones móviles consiste en maximizar la información mutua del canal, con la siguiente expresión

$$\begin{array}{ll} \underset{\boldsymbol{B}}{\text{maximizar}} & \log|\boldsymbol{I} + \boldsymbol{R}_n^{-1}\boldsymbol{H}\boldsymbol{B}\boldsymbol{B}^H\boldsymbol{H}^H| \\ \text{sujeto a} & \text{Tr}\left(\boldsymbol{B}\boldsymbol{B}^H\right) \leq P_T \end{array}$$

De aquí podemos poner la información mutua como

$$I = \log|\boldsymbol{I} + \boldsymbol{B}^H\boldsymbol{H}^H\boldsymbol{R}_n^{-1}\boldsymbol{H}\boldsymbol{B}|$$

Podemos comparar esta ecuación con (7.6) y llegar a lo siguiente

$$I = -\log|\boldsymbol{E}| = -\log\prod_{k,i}\text{MSE}_{k,i}$$

por tanto, la matriz óptima de transmisión se obtiene del siguiente problema convexo

$$\begin{array}{lll} \underset{z_{k,i}}{\text{minimizar}} & \displaystyle\sum_{k,i}\log\left(\frac{1}{1+\lambda_{k,i}z_{k,i}}\right) & \\ \text{sujeto a} & \displaystyle\sum_{k,i} z_{k,i} \leq P_T & \\ & z_{k,i} \geq 0 & 1 \leq k \leq N,\ 1 \leq i \leq \check{L}_k \end{array} \tag{7.20}$$

Como vimos en clase, este problema también se resuelve con algoritmos de *water-filling*. Para llegar a su descripción analítica, empezamos replanteando (7.20) del siguiente modo

$$\begin{array}{ll} \underset{z_{k,i}}{\text{minimizar}} & -\sum_{k,i} \log\left(1+\lambda_{k,i} z_{k,i}\right) \\ \text{sujeto a} & \sum_{k,i} z_{k,i} \leq P_T \\ & z_{k,i} \geq 0 \qquad\qquad 1 \leq k \leq N,\, 1 \leq i \leq \check{L}_k \end{array}$$

Igual que en §7.3.4.1, definimos las variables duales asociadas a cada restricción

$$\begin{aligned} f_1(\boldsymbol{z}) = \mathbf{1}^T \boldsymbol{z} - P_T \leq 0 \rightarrow\ & \mu \\ g_{k,i}(\boldsymbol{z}) = -z_{k,i} \leq 0 \rightarrow\ & \nu_{k,i} \end{aligned}$$

Las ecuaciones KKT son las siguientes

$$-\frac{\lambda_{k,i}}{1+\lambda_{k,i} z_{k,i}} + \mu - \nu_{k,i} = 0 \qquad 1 \leq k \leq N,\, 1 \leq i \leq \check{L}_k \tag{7.21}$$

$$\mathbf{1}^T \boldsymbol{z} \leq P_T \tag{7.22}$$

$$z_{k,i} \geq 0 \tag{7.23}$$

$$\mu \geq 0 \tag{7.24}$$

$$\nu_{k,i} \geq 0 \qquad 1 \leq k \leq N,\, 1 \leq i \leq \check{L}_k \tag{7.25}$$

$$\mu\left(\mathbf{1}^T \boldsymbol{z} - P_T\right) = 0 \tag{7.26}$$

$$\nu_{k,i} z_{k,i} = 0 \qquad 1 \leq k \leq N,\, 1 \leq i \leq \check{L}_k \tag{7.27}$$

Con el mismo razonamiento que en §7.3.4.1 llegamos al valor de $z_{k,i}$

$$z_{k,i} = \begin{cases} 0 & \text{si } \mu > \lambda_{k,i} \\ \mu^{-1} - \lambda_{k,i}^{-1} & \text{si } \mu < \lambda_{k,i} \end{cases}$$

O lo que es equivalente,

$$z_{k,i} = \text{máx}\left\{0, \mu^{-1} - \lambda_{k,i}^{-1}\right\} = \left(\mu^{-1} - \lambda_{k,i}^{-1}\right)^+ \tag{7.28}$$

con lo que la restricción de potencia (7.22) nos permite describir el algoritmo de *water-filling* de acuerdo con la siguiente ecuación creciente y lineal por tramos respecto a μ^{-1}

$$\sum_{k,i} \text{máx}\left\{0, \mu^{-1} - \lambda_{k,i}^{-1}\right\} \leq P_T$$

7.3.5 Restricciones adicionales

Como hemos visto, los problemas convexos a los que llegamos en §7.3.4 se pueden resolver con algoritmos sencillos sin necesidad de utilizar ninguna herramienta de las vistas en el capítulo 5. La utilización de estas herramientas en estos casos tiene sentido cuando necesitamos añadir restricciones adicionales que impidan la resolución analítica de las ecuaciones KKT.

La modulación OFDM presenta como inconveniente que la señal transmitida suele presentar picos de potencia demasiado elevados con respecto al valor medio. Esto se conoce como PAR o PAPR (*Peak to Average Power Ratio*), y supone un problema a la hora de convertir la señal analógica en digital [Han05]. La solución más sencilla consiste en recortar los picos de amplitud superiores a un nivel determinado. En este capítulo he utilizado una restricción de potencia que limita la probabilidad de recorte de la señal transmitida para paliar los efectos de la distorsión producida por dicho recorte. La relación PAR tiene la siguiente expresión

$$\mathrm{PAR} \triangleq \frac{\underset{0\leq t\leq T_s}{\max} S^2_{\mathrm{ofdm}}(t)}{\int_0^{T_s} |S_{\mathrm{ofdm}}(t)|^2 \mathrm{d}t}$$

Cuando el número de portadoras es alto, la amplitud de la señal $S_{\mathrm{ofdm}}(t)$ se puede modelar como un proceso aleatorio gaussiano por el teorema central del límite. Con esta suposición, la probabilidad de que la señal exceda el límite A_{clip} es

$$\Pr\left\{|S_{\mathrm{ofdm}}(t)| > A_{\mathrm{clip}}\right\} = 2\mathcal{Q}\left(\frac{A_{\mathrm{clip}}}{\sigma_{S_{\mathrm{ofdm}}}}\right) \tag{7.29}$$

En esta ecuación $\mathcal{Q}$ es la función de distribución normal acumulada de una variable aleatoria gaussiana, y $\sigma_{S_{\mathrm{ofdm}}}$ es la desviación típica de la amplitud de S_{ofdm}, y se definen como

$$\mathcal{Q}(x) = \frac{1}{\sqrt{2\pi}} \int_x^{\infty} e^{-t^2/2} \mathrm{d}t$$

$$\sigma_{S_{\mathrm{ofdm}}} = \sqrt{\int_0^{T_s} |S_{\mathrm{ofdm}}(t)|^2 \mathrm{d}t}$$

Con estos datos, la probabilidad de recorte en un símbolo OFDM es

$$P_{\mathrm{clip}}(\sigma_{S_{\mathrm{ofdm}}}) = 1 - \left(1 - 2\mathcal{Q}\left(\frac{A_{\mathrm{clip}}}{\sigma_{S_{\mathrm{ofdm}}}}\right)\right)^{2N}$$

El exponente doble viene por la probabilidad de corte en las componentes en fase y cuadratura de la señal transmitida. Este cálculo resulta demasiado conservativo para las simulaciones, y por eso en los casos presentados he

asignado la expresión (7.29) directamente a P_{clip}, por lo que la restricción de potencia de la señal queda así

$$\sigma_{S_{\text{ofdm}}} \leq \frac{A_{\text{clip}}}{\mathcal{Q}^{-1}\left(P_{\text{clip}}/2\right)}$$

Por otro lado, la potencia transmitida en cada antena se puede calcular como

$$P_i = \frac{1}{N}\sum_{k=1}^{N}\sum_{l=1}^{L}\left|[\boldsymbol{B}_k]_{i,l}\right|^2 = \frac{1}{N}\sum_{k=1}^{N}\sum_{l=1}^{L}\sigma^2_{B_k,l}\left|[\boldsymbol{U}_{H_k}]_{i,l}\right|^2$$

Esta expresión es lineal en las variables de optimización $\sigma^2_{B_k,i}$, por lo que podemos añadirla a nuestro problema convexo como

$$\frac{1}{N}\sum_{k=1}^{N}\sum_{l=1}^{L}\sigma^2_{B_k,l}\left|[\boldsymbol{U}_{H_k}]_{i,l}\right|^2 \leq \left(\frac{A_{\text{clip}}}{\mathcal{Q}^{-1}\left(P_{\text{clip}}/2\right)}\right)^2 \qquad 1 \leq i \leq n_t \tag{7.30}$$

El efecto más evidente que tiene esta restricción es que se reduce la potencia transmitida, pero también se modifica la distribución entre portadoras con respecto a cuando no se aplica esta restricción.

7.4 Implementación del optimizador

La carga de complejidad de las simulaciones en este capítulo está en la estructura del sistema MIMO-OFDM, y no tanto en la resolución convexa, porque el código de la optimización consiste en implementar el problema (7.10) o el (7.20) dependiendo del criterio utilizado.

7.4.1 CVX

7.4.1.1 Criterio de suma de MSE

A continuación vemos el código con el que se describe el problema (7.10) incluyendo la restricción de PAR (7.30), es decir

$$\begin{aligned}
\underset{z_{k,i}}{\text{minimizar}} \quad & \sum_{k=1}^{N}\sum_{i=1}^{\check{L}_k}\frac{1}{1+\lambda_{k,i}z_{k,i}} & \\
\text{sujeto a} \quad & \sum_{k=1}^{N}\sum_{i=1}^{\check{L}_k} z_{k,i} \leq P_T & \\
& z_{k,i} \geq 0 & 1 \leq k \leq N,\ 1 \leq i \leq \check{L}_k \\
& \frac{1}{N}\sum_{k=1}^{N}\sum_{i=1}^{\check{L}_k} z_{k,i}\left|[\boldsymbol{U}_{H_k}]_{l,i}\right|^2 \leq \sigma^2_{\text{clip}} & 1 \leq l \leq n_t
\end{aligned}$$

donde $\sigma^2_{\text{clip}} = \left(\frac{A_{\text{clip}}}{\mathcal{Q}^{-1}\left(P_{\text{clip}}/2\right)}\right)^2$ es el umbral de recorte. Este problema se codifica del siguiente modo

```
function z = mseCvx(Lambda, Psym, mu, var, Upar)
cvx_quiet(true);
cvx_begin
    variable z(length(Lambda));
    minimize(sum(inv_pos(1+Lambda.*z)))
    subject to
        sum(z)  <= Psym;
        z       >= 0;
        if mu >= 0;
            abs(Upar).^2*z <= var;
        end
cvx_end
```

Independientemente de las restricciones, las normas de programación convexa disciplinada [Gra06, Gra11] no permiten la utilización de funciones no convexas como objetivo a minimizar. Aunque la función

$$\sum_{k,i} \frac{1}{1+\lambda_{k,i} z_{k,i}}$$

es convexa en el conjunto viable de $\boldsymbol{z}$, no lo es para $1+\lambda_{k,i} z_{k,i} < 0$. Por este motivo, *CVX* incluye la función `inv_pos` que devuelve $+\infty$ si le pasamos un argumento negativo. Esto es lo que llamamos en §2.3.1 la función de valor extendido de $1/x$. De este modo, la función utilizada es convexa en todo el dominio de $\boldsymbol{z}$. Recordemos del tema 4 que los algoritmos de optimización pueden comenzar por puntos no viables, por lo que sin esta modificación, la función inversa podría no converger hacia el conjunto viable.

Cuando utilizamos esta función en el esquema no cooperativo, el problema tiene que abarcar a todas las portadoras para optimizar la potencia total transmitida. Por tanto, el argumento `Lambda` tiene que contener los $\check{L}_k$ autovalores mayores de cada $\boldsymbol{R}_{H_k}$ formando un vector de $\check{L}_T$ filas, de acuerdo con la siguiente definición

$$\check{L}_T = \sum_{k=1}^{N} \check{L}_k$$

El esquema cooperativo es más sencillo, porque sólo contiene una matriz $\boldsymbol{R}_H$.

En caso de no querer utilizar la restricción de PAR, pasamos un valor negativo en el argumento `mu`. La matriz $\boldsymbol{U}_{\text{par}} \in \mathbb{C}^{n_t \times \check{L}_T}$ contiene concatenadas todas las matrices de los autovectores de $\boldsymbol{R}_{H_k}$ correspondientes a sus $\check{L}_k$ autovalores más altos, de modo que el producto $\boldsymbol{U}_{\text{par}}\boldsymbol{z} \in \mathbb{C}^{n_t}$ se puede utilizar para construir la restricción (7.30), tomando en cada uno de los elementos de $\boldsymbol{U}_{\text{par}}$ el cuadrado del valor absoluto.

7.4.1.2 Criterio de información mutua

Este criterio lo he planteado únicamente con *CVX* porque no he llegado a una descripción cónica de (7.20). La expresión del problema a optimizar es la siguiente:

$$
\begin{aligned}
\underset{z_{k,i}}{\text{minimizar}} \quad & \sum_{k=1}^{N}\sum_{i=1}^{\check{L}_k} \log\left(\frac{1}{1+\lambda_{k,i}z_{k,i}}\right) \\
\text{sujeto a} \quad & \sum_{k=1}^{N}\sum_{i=1}^{\check{L}_k} z_{k,i} \leq P_T \\
& z_{k,i} \geq 0 \qquad\qquad 1 \leq k \leq N,\ 1 \leq i \leq \check{L}_k \\
& \frac{1}{N}\sum_{k=1}^{N}\sum_{i=1}^{\check{L}_k} z_{k,i}\left|[\boldsymbol{U}_{H_k}]_{l,i}\right|^2 \leq \sigma^2_{\text{clip}} \qquad 1 \leq l \leq n_t
\end{aligned}
$$

El código de optimización es similar al utilizado en §7.4.1.1

```
function z = imCvx(Lambda, Psym, mu, var, Upar)
cvx_quiet(true);
cvx_begin
    variable z(length(Lambda));
    maximize(sum(log(1+Lambda.*z)))
    subject to
        sum(z)  <= Psym;
        z       >= 0;
        if mu >= 0;
            abs(Upar).^2*z <= var;
        end
cvx_end
```

Lo único que cambia es la función objetivo. La función `log` utilizada en este problema también está modificada por *CVX* para dar su función de valor extendido que devuelva $-\infty$ al pasarle argumentos negativos, y así se trata de una función cóncava de acuerdo con las normas de programación convexa disciplinada, tal como veíamos en §2.3.1.

7.4.2 Sedumi

Para resolver el problema (7.10) con *SeDuMi* hay que transformarlo en un problema cónico. El primer paso es pasar la función objetivo a una restric-

ción, con la técnica del epígrafo vista en §3.1.7, de modo que nos queda

$$\begin{array}{ll} \underset{z_{k,i},t_{k,i}}{\text{minimizar}} & \displaystyle\sum_{k,i} t_{k,i} \\ \text{sujeto a} & \displaystyle\sum_{k,i} z_{k,i} \leq P_T \\ & z_{k,i} \geq 0 \\ & \dfrac{1}{1+\lambda_{k,i} z_{k,i}} \leq t_{k,i} \end{array} \tag{7.31}$$

A continuación, el problema cónico equivalente tiene la siguiente expresión

$$\begin{array}{ll} \underset{z_{k,i},t_{k,i}}{\text{minimizar}} & \displaystyle\sum_{k,i} t_{k,i} \\ \text{sujeto a} & \displaystyle\sum_{k,i} z_{k,i} \leq P_T \\ & z_{k,i} \geq 0 \\ & \left\| \begin{bmatrix} 1+z_{k,i}\lambda_{k,i} - t_{k,i} \\ 2 \end{bmatrix} \right\| \leq 1 + z_{k,i}\lambda_{k,i} + t_{k,i} \end{array} \tag{7.32}$$

Para demostrar la equivalencia entre ambos problemas, es sencillo desarrollar la siguiente desigualdad para llegar a que las restricciones de (7.31) y (7.32) son las mismas

$$\left\| \begin{bmatrix} 1+z_{k,i}\lambda_{k,i} - t_{k,i} \\ 2 \end{bmatrix} \right\|^2 \leq (1+z_{k,i}\lambda_{k,i} + t_{k,i})^2$$

$$\Updownarrow$$

$$(1+z_{k,i}\lambda_{k,i})^2 + t_{k,i}^2 - 2\,(1+z_{k,i}\lambda_{k,i})\,t_{k,i} + 4 \leq (1+z_{k,i}\lambda_{k,i})^2 + t_{k,i}^2 + 2\,(1+z_{k,i}\lambda_{k,i})\,t_{k,i}$$

$$\Updownarrow$$

$$4 \leq 4\,(1+z_{k,i}\lambda_{k,i})\,t_{k,i}$$

$$\Updownarrow$$

$$\frac{1}{1+z_{k,i}\lambda_{k,i}} \leq t_{k,i}$$

Para comprender la resolución del problema (7.32) con *SeDuMi* voy a plantear un caso sencillo con dos variables:

$$\begin{array}{ll} \underset{t_1,t_2,z_1,z_2}{\text{minimizar}} & t_1 + t_2 \\ \text{sujeto a} & z_1 + z_2 \leq P_T \\ & z_1 \geq 0 \\ & z_2 \geq 0 \\ & \left\| \begin{bmatrix} 1+z_1\lambda_1 - t_1 \\ 2 \end{bmatrix} \right\| \leq 1 + z_1\lambda_1 + t_1 \\ & \left\| \begin{bmatrix} 1+z_2\lambda_2 - t_2 \\ 2 \end{bmatrix} \right\| \leq 1 + z_2\lambda_2 + t_2 \end{array} \tag{7.33}$$

La variable del problema en *SeDuMi* será $\boldsymbol{x}$ definida del siguiente modo

$$\begin{aligned}
x_1 &= t_1 \\
x_2 &= t_2 \\
x_3 &= z_1 \\
x_4 &= z_2 \\
x_5 &= P_T - z_1 - z_2 = P_T - x_3 - x_4 \\
x_6 &= 1 + \lambda_1 z_1 + t_1 = 1 + \lambda_1 x_3 + x_1 \\
x_7 &= 1 + \lambda_1 z_1 - t_1 = 1 + \lambda_1 x_3 - x_1 \\
x_8 &= 2 \\
x_9 &= 1 + \lambda_2 z_1 + t_1 = 1 + \lambda_2 x_3 + x_2 \\
x_{10} &= 1 + \lambda_2 z_2 - t_2 = 1 + \lambda_2 x_3 - x_2 \\
x_{11} &= 2
\end{aligned} \tag{7.34}$$

Es importante que el orden de las componentes sea el siguiente: primero variables libres, que pueden tomar cualquier valor; después variables no negativas, y por último las variables que definen conos de segundo orden.

Comprobamos que para la restricción de potencia $z_1 + z_2 \leq P_T$ he añadido la componente $x_5 = P_T - z_1 - z_2 \geq 0$.

El problema (7.33) en formato estándar de *SeDuMi* es el siguiente

$$\begin{aligned}
\underset{\boldsymbol{x}}{\text{minimizar}} \quad & x_1 + x_2 \\
\text{sujeto a} \quad & x_3 \geq 0 \\
& x_4 \geq 0 \\
& x_5 \geq 0 \\
& x_6 \geq \left\| \begin{bmatrix} x_7 \\ x_8 \end{bmatrix} \right\| \\
& x_9 \geq \left\| \begin{bmatrix} x_{10} \\ x_{11} \end{bmatrix} \right\| \\
& \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}
\end{aligned} \tag{7.35}$$

En la restricción de igualdad $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ se incluyen las definiciones de (7.34)

del siguiente modo

$$\begin{bmatrix} 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & \lambda_1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & -\lambda_1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & \lambda_2 & 0 & 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 1 & 0 & -\lambda_2 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ x_8 \\ x_9 \\ x_{10} \\ x_{11} \end{bmatrix} = \begin{bmatrix} P_T \\ -1 \\ 1 \\ 2 \\ -1 \\ 1 \\ 2 \end{bmatrix} \tag{7.36}$$

Cuando el tamaño del argumento `Lambda` y de la variable inicial de optimización $\boldsymbol{z}$ es `n` en lugar de 2, tendremos `n` variables libres, `n+1` variables no negativas, y `n` conos de segundo orden en $\mathbb{R}^3$. Esto se describe en el siguiente código

```
1  function z = mseSedumi(Lambda, Psym)
2  n = length(Lambda);
3  % restricción lineal de igualdad
4  A = sparse(3*n+1,5*n+1);
5  b = Psym;
6  A(1,n+1:2*n+1) = 1;
7  for i = 1:n
8      fila = 3*i-1;
9      A(fila,i) = 1;
10     A(fila,n+i) = Lambda(i);
11     A(fila,2*n+3*i-1) = -1;
12     fila = 3*i;
13     A(fila,i) = 1;
14     A(fila,n+i) = -Lambda(i);
15     A(fila,2*n+3*i) = 1;
16     fila = 3*i+1;
17     A(fila,2*n+3*i+1) = 1;
18     b = [b; -1; 1; 2];
19 end
20 % constante de la función objetivo
21 c = [ones(n,1); zeros(4*n+1,1)];
22 % estructura de la variable de optimización
23 clear K; clear pars;
24 K.f = n;              % variables libres t
25 K.l = n+1;            % variables >= 0
26 K.q = 3*ones(1,n);    % conos cuadráticos
27 pars.fid = 0;         % Para ejecutar sin mostrar mensajes
28 [x y info] = sedumi(A,b,c,K,pars);
29 z = x(n+1:2*n);       % extraer z de x
```

El bucle entre las líneas 4 y 19 describe la restricción de igualdad lineal

de (7.36). En las líneas 24, 25 y 26 se especifica el orden de las componentes de $\boldsymbol{x}$ tal como se describe en (7.35). Esto se hace por medio de la estructura `K`. Una vez definido todo esto, se puede llamar directamente al optimizador y extraer el resultado.

Como principal diferencia con respecto a los ejemplos de los capítulos anteriores, hemos añadido las variables no negativas por medio del campo `K.l`.

En este caso no he añadido la restricción de PAR. Para incluirlo hay que volver a plantear el problema tal como se ha descrito en este apartado.

7.4.3 Mosek

Los requerimientos de *Mosek* son muy parecidos a los de *SeDuMi*, por lo que he utilizado el mismo código para asignar valores a la restricción lineal $\boldsymbol{Ax} = \boldsymbol{b}$. Fijándonos en el ejemplo anterior con dos variables, en *Mosek* no haría falta añadir la componente $x_5 = P_T - z_1 - z_2 \geq 0$ para la limitación de potencia, porque nos permite incluir restricciones lineales de desigualdad. De hecho, las restricciones lineales únicamente pueden ser de desigualdad, siendo la igualdad un caso particular en el que el límite inferior y el superior son lo mismo. No obstante, al ser compatible la descripción de *SeDuMi* para *Mosek*, no merece la pena volver a plantear el mismo problema con ligeras variaciones.

A continuación vemos el código necesario para optimizar el problema (7.10) con *Mosek*

```
1  function z = mseMosek(Lambda, Psym)
2  (...)
3  clear prob;
4  prob.cones = cell(n,1);
5  for i = 1:n
6  (...)
7      % Definición de conos cuadráticos
8      prob.cones{i}.type = 'MSK_CT_QUAD';
9      prob.cones{i}.sub = [2*n+3*i-1:2*n+3*i+1];
10 end
11 % constante de la función objetivo
12 c = [ones(n,1); zeros(4*n+1,1)];
13 % Asignación de campos para Mosek
14 prob.a = A;      % Desigualdad lineal
15 prob.blc = b;    % Límite inferior de Ax
16 prob.buc = b;    % Límite superior de Ax
17 prob.c = c;      % Constante del objetivo
18 % Límites de la variable x
19 prob.blx = [-inf*ones(n,1); zeros(n+1,1); -inf*ones(3*n,1)];
20 prob.bux = inf*ones(5*n+1,1);    % Límite superior de x
21 param = [];
22 param.MSK_IPAR_LOG=0;    % Desactivar mensajes
23 % Llamada al optimizador
```

```
24 [r,res] = mosekopt('minimize',prob,param);
25 z = full(res.sol.itr.xx(n+1:2*n));   % Extraer resultado
```

Las líneas donde aparece `(...)` son las que están copiadas directamente de la función de *SeDuMi* para asignar valores a `A` y `b`. Los campos `prob.blx` y `prob.bux` de esta estructura nos sirven para asignar los límites superior e inferior de cada componente del vector de optimización, y a diferencia de *SeDuMi*, nos sirven tanto para las variables libres como para las variables no negativas.

Como vemos, Aunque la forma de describir el problema es muy parecida a la de *SeDuMi*, en este caso el orden en que se definen las componentes de la variable de optimización puede ser cualquiera. Por eso, la descripción de los conos de segundo orden se hace dentro del bucle, ya que al no necesitar un orden concreto en las componentes de `x`, hay que indicar los índices de las componentes que forman parte de dichos conos. Esto se hace en la agrupación de celdas `prob.cones{i}`.

7.5 Arquitectura del sistema de simulación

Las gráficas de obtenidas en este capítulo se basan en la consola de pruebas de tasa de error (*Error Rate Test Console*) incluida en el complemento *Communications Toolbox* de *Matlab* en la versión R2010a. Esta herramienta sirve para que el usuario tenga que implementar una sola transmisión de una trama de datos, y la consola se encarga de repetir las transmisiones para diferentes valores de los parámetros que se necesiten hasta conseguir las curvas de tasa de error.

Para que funcione es necesario utilizar algunos conceptos de programación orientada a objetos. Siguiendo la ayuda del programa es sencillo encontrar una plantilla desde la que podemos crear nuestra simulación. En nuestro caso los objetos utilizados están definidos en la clase `cBER`, que encontramos en el archivo del mismo nombre (con la extensión .m).

Todas las simulaciones se obtienen ejecutando uno de los siguientes guiones de *Matlab*: `BERvsSNR.m` y `PclipyBERvsMuySNR.m` en el directorio `MIMO` del CD adjunto a este trabajo.

Para resumir los procesos de los que se encarga cada fichero y función de este directorio, el esquema de la figura 7.4 contiene a grandes rasgos las funciones más representativas y la interacción entre ellas.

Los procesos necesarios para simular una transmisión MIMO-OFDM son los siguientes

1. Descripción del canal. Los datos del canal se crean en el fichero principal de la simulación, bien `BERvsSNR.m` o `PclipyBERvsMuySNR.m`, dependiendo de si la simulación consiste en las curvas de BER en función de SNR, o en las curvas de probabilidad de recorte y BER en función de la SNR y del nivel de recorte, respectivamente. Las primeras son

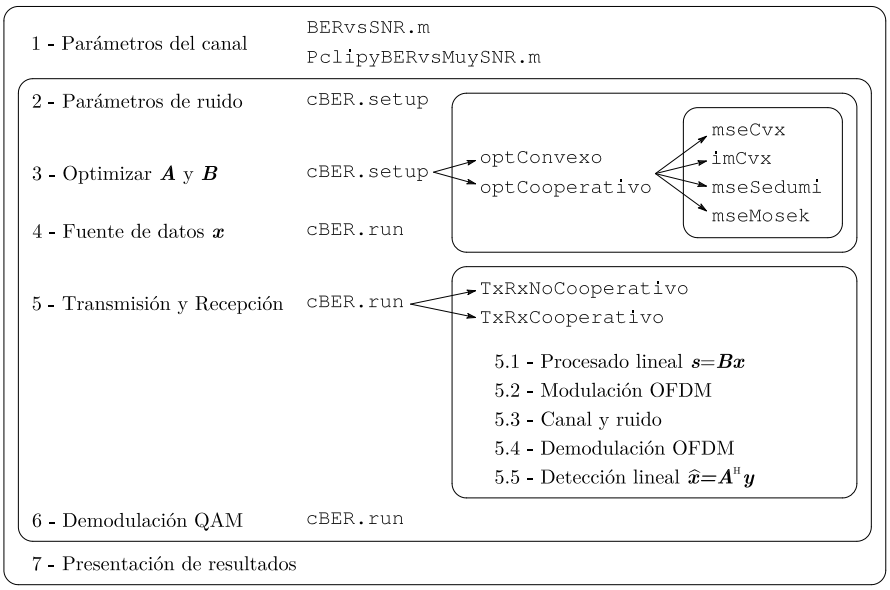

Figura 7.4: Procesos realizados por cada función del código.

las que encontramos en §7.6.1 y §7.6.2, y el segundo fichero sirve para generar las gráficas de §7.6.3.

2. Asignación de parámetros de ruido. La optimización depende de la matriz de covarianzas del ruido, y en las simulaciones necesitamos hacer un barrido de valores de la relación SNR. Esto se define en la función `setup`, miembro de la clase `cBER`, que definimos para ser utilizada por la consola de pruebas de error de *Matlab*.

3. Optimización. Con los datos de ruido y el canal definido, la función `cBER.setup` llama a la función `optConvexo` u `optCooperativo`, dependiendo del esquema utilizado, conforme al apartado §7.3. Estas funciones se encargan de obtener y ordenar los autovalores y autovectores del canal, y de llamar a la función de optimización que se necesite en cada caso, y que puede ser `mseCvx`, `imCvx`, `mseSedumi`, o `mseMosek`. Por tanto, el contenido de `optConvexo` y `optCooperativo` son los procedimientos de diseño del transmisor descritos en §7.3.3, mientras que la optimización está implementada en las funciones de §7.4.

4. Generar datos de información. En cada ejecución de la simulación, la función `cBER.run` se encarga de generar tramas de datos binarios, y los modula en símbolos QAM[18] de forma que se crea un conjunto de

[18] *Quadrature Amplitude Modulation*

datos con 3 dimensiones: $\boldsymbol{x} \in \mathbb{C}^{N \times N_{\text{sym}} \times L}$, donde N es el número de portadoras, N_{sym} es el número de símbolos OFDM que se transmiten por cada portadora y flujo, y L es el número de flujos. La organización en tres dimensiones facilita su adaptabilidad a los esquemas cooperativo y no cooperativo.

5. Transmisión y recepción de los símbolos. Para transmitir los símbolos generados en `cBER.run`, esta función llama a `TxRxNoCooperativo` o `TxRxCooperativo`, donde se realizan los procesos descritos en §7.3, que se resumen en la figura 7.4, es decir, procesamiento lineal de los datos $\boldsymbol{s} = \boldsymbol{B}\boldsymbol{x}$ (7.2), modulación OFDM, transmisión por el canal y ruido, demodulación OFDM, y detección lineal, $\hat{\boldsymbol{x}} = \boldsymbol{A}^H \boldsymbol{y}$ (7.3).

6. Paso de símbolos a bits. Una vez recibidas las tramas de símbolos, `cBER.run` se encarga de la demodulación QAM para que la consola pueda medir la tasa de error.

Para las gráficas de BER en función de SNR, indicamos a la consola de pruebas de tasa de error el rango de valores de SNR que queremos probar, y también el conjunto de opciones que queremos comparar. La lista de opciones disponibles está en la tabla 7.1.

Opción	Criterio	Optimizador	Esquema	Restricción de PAR	Aplicar recorte
1	suma de MSE	*CVX*	no cooperativo	no	no
2	suma de MSE	*CVX*	no cooperativo	sí	no
3	suma de MSE	*CVX*	no cooperativo	no	sí
4	suma de MSE	*CVX*	no cooperativo	sí	sí
5	suma de MSE	*CVX*	cooperativo	no	no
6	suma de MSE	*SeDuMi*	no cooperativo	no	no
7	suma de MSE	*SeDuMi*	cooperativo	no	no
8	suma de MSE	*Mosek*	no cooperativo	no	no
9	suma de MSE	*Mosek*	cooperativo	no	no
10	información mutua	*CVX*	no cooperativo	no	no
11	información mutua	*CVX*	no cooperativo	sí	no
12	información mutua	*CVX*	no cooperativo	no	sí
13	información mutua	*CVX*	no cooperativo	sí	sí
14	información mutua	*CVX*	cooperativo	no	no

Tabla 7.1: Opciones de simulación

Por cada valor de SNR y en cada opción, la consola llama a la función `cBER.setup`, en la que se ejecuta la optimización de acuerdo con los parámetros de ruido. A continuación, se encarga de ejecutar la función `cBER.run` tantas veces como sea necesario para acumular un número suficiente de errores que permita conocer la tasa de error.

En el caso de medida de BER se comparan la secuencia de bits transmitida y recibida. Cuando queremos medir probabilidad de recorte en lugar de BER, hacemos que se comparen las muestras de la señal OFDM recortada y sin recortar. La tasa de muestras que diferencian ambas señales es la probabilidad de recorte.

7.6 Simulaciones

Los resultados se presentan en tres apartados en función de la comparativa que revelan. En primer lugar tenemos las curvas de BER para las diferentes herramientas utilizadas. Después vemos el efecto de los dos criterios de optimización utilizados: Minimización de suma del error cuadrático medio o maximización de la información mutua en los flujos de datos transmitidos. La última serie de gráficas muestra el efecto de una restricción que añadimos para combatir el efecto de los picos elevados de nivel que se producen en las señales OFDM.

Los resultados de este capítulo vienen de la simulación de un sistema MIMO-OFDM optimizado con los dos criterios descritos en §7.3.4, suma de MSE e información mutua.

El canal utilizado viene del modelo del estándar HIPERLAN[19]/2 para WLAN[20] definido en [Med98]. El objetivo es reproducir los resultados de [Pal03a]. Por este motivo, he utilizado 64 portadoras, y el perfil de potencia y retardo de la tabla 7.2, con las matrices de covarianza que aparecen en la misma tabla.

De acuerdo con esto, se trata de un canal selectivo en frecuencia correspondiente a un entorno típico de interior sin visión directa, con un valor medio de dispersión de retardo de 150ns. El período de muestreo es 50ns según [ETS01]. Para simular el canal he utilizado la función `mimochan` disponible en el complemento *Communications Toolbox* de *Matlab* a partir de la versión R2010a.

Teniendo en cuenta los resultados del tema 6, he utilizado únicamente herramientas de optimización convexa, y he omitido el uso de *Matlab Optimization Toolbox* y *Mathematica*, no sólo por la cantidad de resultados no convergentes que se producen, sino por el tiempo que consumen.

7.6.1 Comparativa de herramientas de optimización

En este apartado se muestran las curvas de BER respecto a la SNR que nos permiten comparar las herramientas utilizadas en base al criterio de optimización de la suma de MSE descrito en §7.3.4.1. En la figura 7.5 vemos el resultado para un sistema con 2 antenas en recepción, 2 antenas en

[19] *High Performance Local Area Network*
[20] *Wireless Local Area Network*

Retardo (ns)	**Potencia relativa (dB)**
0	-3.3
10	-3.6
20	-3.9
30	-4.2
50	0.0
80	-0.9
110	-1.7
140	-2.6
180	-1.5
230	-3.0
280	-4.4
330	-5.9
400	-5.3
490	-7.9
600	-9.4
730	-13.2
880	-16.3
1050	-21.2

$$\boldsymbol{R_{tx}} = \begin{bmatrix} 1{,}0000 & 0{,}4154 & 0{,}2057 & 0{,}1997 \\ 0{,}4154 & 1{,}0000 & 0{,}3336 & 0{,}3453 \\ 0{,}2057 & 0{,}3336 & 1{,}0000 & 0{,}5226 \\ 0{,}1997 & 0{,}3453 & 0{,}5226 & 1{,}0000 \end{bmatrix}$$

$$\boldsymbol{R_{rx}} = \begin{bmatrix} 1{,}0000 & 0{,}3644 & 0{,}0685 & 0{,}3566 \\ 0{,}3644 & 1{,}0000 & 0{,}3245 & 0{,}1848 \\ 0{,}0685 & 0{,}3245 & 1{,}0000 & 0{,}3093 \\ 0{,}3566 & 0{,}1848 & 0{,}3093 & 1{,}0000 \end{bmatrix}$$

Tabla 7.2: Perfil de potencia y retardo de tipo C en HIPERLAN/2, y matrices de correlación en la estación base (tx) y en el equipo móvil (rx).

transmisión y un flujo de datos.

Los tres optimizadores utilizados dan resultados similares. La diferencia entre ellos viene de mostrar una sola realización del canal. Los valores más bajos de BER requieren un tiempo de ejecución muy alto. Por este motivo no he simulado los estadísticos suficientes para mostrar cifras en términos de una probabilidad determinada de desconexión, como se suele hacer con los canales aleatorios. Se aprecia claramente que el esquema cooperativo da mejores resultados al permitir la distribución de datos entre varias portadoras.

En la figura 7.6 se ha aumentado el número de antenas en recepción, y el índice de modulación a 16QAM.

También se aprecia claramente la ventaja que supone el esquema cooperativo. En cuanto a las herramientas, como era de esperar, dan aproximadamente el mismo resultado. El aumento en número de antenas y en la constelación permite un aumento en la velocidad de transmisión a costa de empeorar el índice de errores.

Volvemos a la constelación QPSK, y utilizamos 4 antenas en transmisión y en recepción, con dos flujos de datos. La curva resultante es la figura 7.7. También se aprecia mejor resultado con el esquema cooperativo.

Utilizando 4 antenas en transmisión y en recepción, y con el sistema completamente cargado con cuatro flujos de datos, obtenemos las curvas de

la figura 7.8.

En este caso ya no se aprecia la diferencia entre el esquema cooperativo y el de portadoras independientes. Al utilizar todos los automodos del canal, y debido a la estructura diagonal de esta solución, ambos esquemas son equivalentes. Esto no sucede al optimizar funciones convexas de Schur, no incluidas en este trabajo.

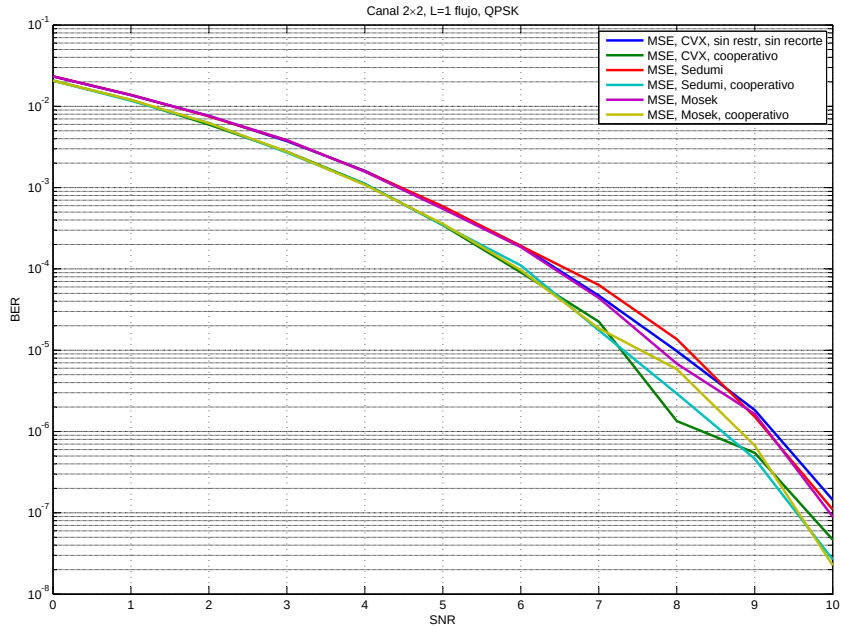

Figura 7.5: Curva de BER para un sistema con $n_r = 2$, $n_t = 2$ y $L = 1$, comparativa de herramientas de optimización.

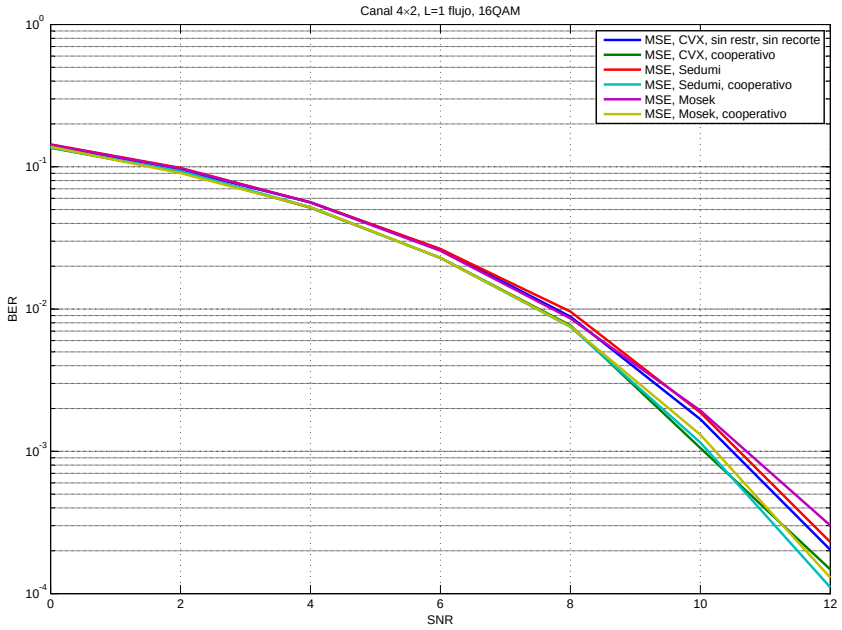

Figura 7.6: Curva de BER para un sistema con $n_r = 4$, $n_t = 2$ y $L = 1$, comparativa de herramientas de optimización.

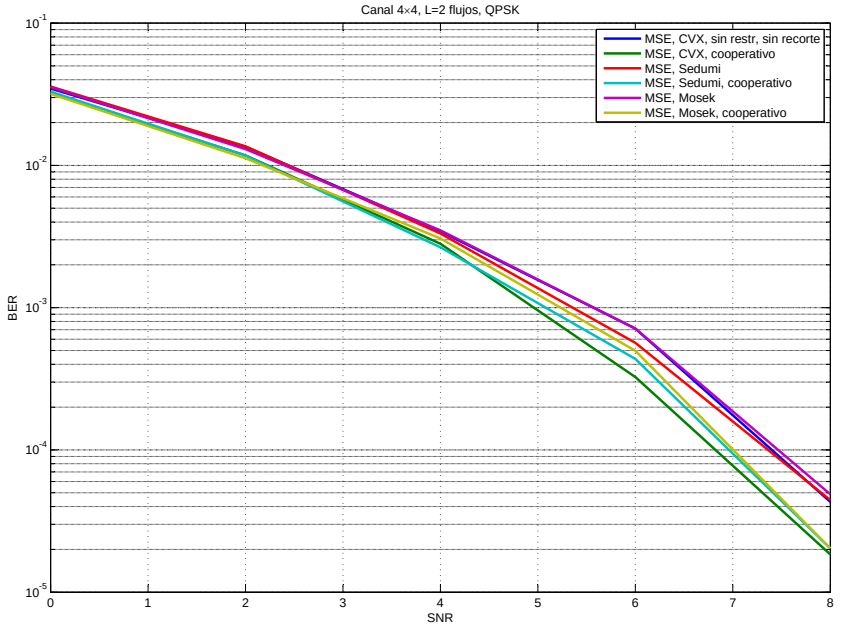

Figura 7.7: Curva de BER para un sistema con $n_r = 4$, $n_t = 4$ y $L = 2$, comparativa de herramientas de optimización.

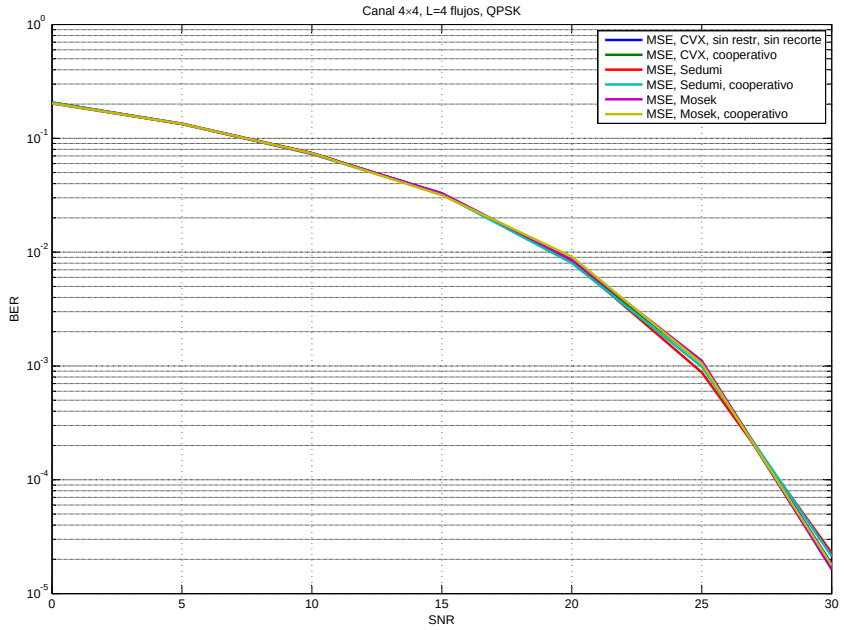

Figura 7.8: Curva de BER para un sistema con $n_r = 4$, $n_t = 4$ y $L = 4$, comparativa de herramientas de optimización.

7.6.2 Comparativa de criterios

En este apartado se superponen las curvas de BER utilizando también el criterio de maximización de la información mutua descrito en §7.3.4.2. Esta segunda opción únicamente la he implementado con CVX, tal como indicaba en §7.4.1.2, por no haber llegado a una descripción cónica del problema (7.20).

Podemos apreciar en la figura 7.9 que el resultado con el criterio de información mutua es peor que al optimizar en función de la suma de MSE. Por otro lado, CVX utiliza un algoritmo de aproximaciones sucesivas cuando encuentra la función `log`. Esto supone que la ejecución es más lenta, y en algunos casos no converge a un resultado viable. Tal como se plantea en esta simulación no tiene ningún sentido utilizar este planteamiento, ya que con la descripción del problema tal como se describe en §7.4.1.2 se puede fácilmente implementar un algoritmo de *water-filling* dando una solución en menos de $\check{L}_T$ iteraciones. También sería sencillo implementar el siguiente problema con cualquier herramienta de optimización convexa

$$\begin{aligned} \underset{\mu^{-1}}{\text{maximizar}} \quad & \mu^{-1} \\ \text{sujeto a} \quad & \sum_{k,i} \text{máx}\left\{0, \mu^{-1} - \lambda_{k,i}^{-1}\right\} \leq P_T \end{aligned}$$

Con la variable $x \triangleq \mu^{-1}$ el problema anterior es lineal por tramos, y a partir de su solución se obtienen los valores de $z_{k,i}$ según (7.28).

En la figura 7.10 tenemos la misma comparación pero con 4 antenas en transmisión y en recepción, y con 2 flujos de datos.

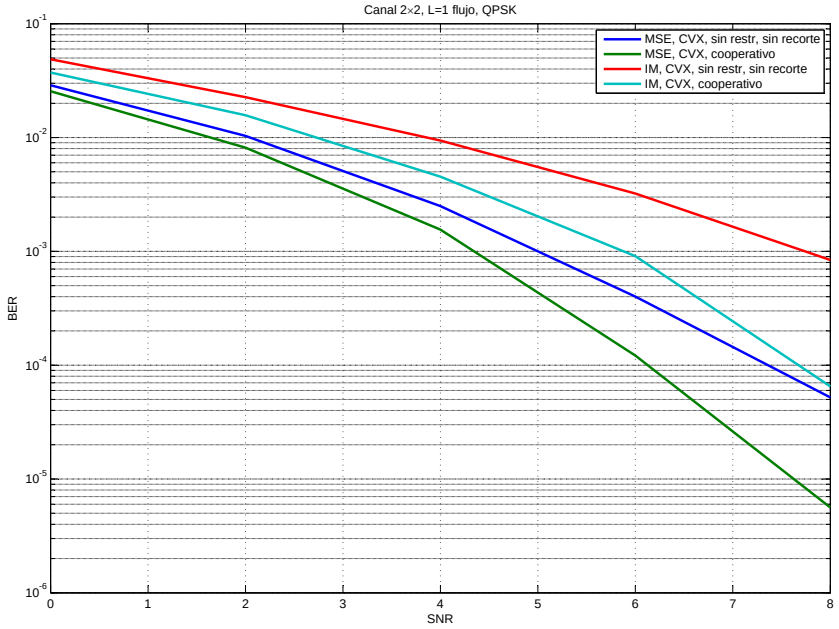

Figura 7.9: Curva de BER para un sistema con $n_r = 2$, $n_t = 2$ y $L = 1$, con los dos criterios de optimización: minimizar la suma de MSE y maximizar la información mutua.

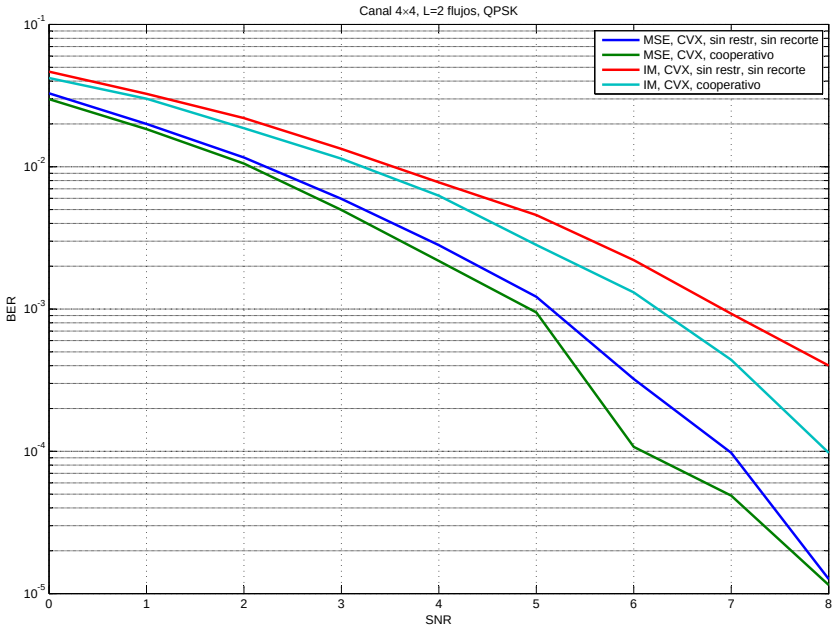

Figura 7.10: Curva de BER para un sistema con $n_r = 4$, $n_t = 4$ y $L = 2$, con los dos criterios de optimización: minimizar la suma de MSE y maximizar la información mutua.

7.6.3 Efecto de la restricción de PAR

En esta sección se aplica la restricción (7.30), y vemos la repercusión en el funcionamiento del sistema. El nivel de recorte está parametrizado con respecto a μ para que los resultados no dependan de la potencia transmitida P_T

$$A_{\text{clip}} = \mu\sqrt{\frac{P_T}{n_t}}$$

En primer lugar, para decidir el valor adecuado del parámetro μ, se muestra en la figura 7.11 la probabilidad de recorte y la tasa de error en función de μ. Cuando $\mu < 1{,}75$ la restricción hace efecto manteniendo la probabilidad de recorte en el 1 %. Sin embargo, la limitación en potencia hace que aumente la tasa de error con respecto al sistema sin la restricción. Claro que en esta simulación la transmisión se hace sin aplicar el recorte, del mismo modo que en los resultados de [Pal03b, Pal05, Pal03a]. En estas referencias también se presentan gráficas de probabilidad de recorte y BER con respecto a la SNR, para un valor fijo de μ, pero he preferido presentar superficies en función de μ y SNR, porque la consola de pruebas de tasa de error permite el barrido de parámetros sin mayor complicación.

La figura 7.12 muestra la probabilidad de recorte respecto a diferentes

valores de μ y de SNR. Vemos que la variación de SNR no influye en esta probabilidad. Sin embargo, en la figura 7.13 comprobamos que como en cualquier sistema de comunicaciones, la SNR afecta a la tasa de error, y las curvas se alejan entre sí a medida que μ disminuye, por efecto de la reducción de potencia impuesta por la restricción que hemos añadido.

Cuando hacemos que se aplique la restricción, obtenemos las curvas de la figura 7.14.

En este caso vemos que el recorte en la señal OFDM influye, como cabía esperar, en la tasa de error, pero aun así no llega a superar a las curvas en las que se aplica la restricción de potencia. Esto no es un buen resultado para la técnica propuesta. El objetivo de fijar la probabilidad se cumple, pero la limitación que se impone en la potencia transmitida no favorece más que el recorte simple. También mostramos en este caso las superficies de variación de probabilidad de recorte respecto a μ y SNR en la figura 7.15, y la de BER respecto a μ y SNR en la figura 7.16.

Aumentamos ahora el número de antenas a 4 en transmisión y en recepción. La figura 7.17 nos indica que un valor razonable para el nivel de recorte corresponde a $\mu = 1{,}5$. Teniendo esto en cuenta, en la figura 7.18 se comparan las curvas de BER vs. SNR con y sin restricción de PAR, y también ambos casos cuando se aplica el recorte. En todos estos casos el criterio utilizado es la suma de MSE, y el nivel de recorte se fija con $\mu = 1{,}5$

Cambiando el criterio a maximizar la información mutua, la figura 7.19 contiene los resultados aplicando y sin aplicar la restricción, y también podemos comprobar el efecto de aplicar o no el recorte en la transmisión.

Esta gráfica nos lleva a la misma conclusión que los casos anteriores. Hemos utilizado un criterio con peores resultados, pero dentro del mismo se puede comprobar el aumento de tasa de error producido por la reducción de potencia impuesta por la restricción, y también el empeoramiento producido al recortar la señal transmitida para reducir el PAR.

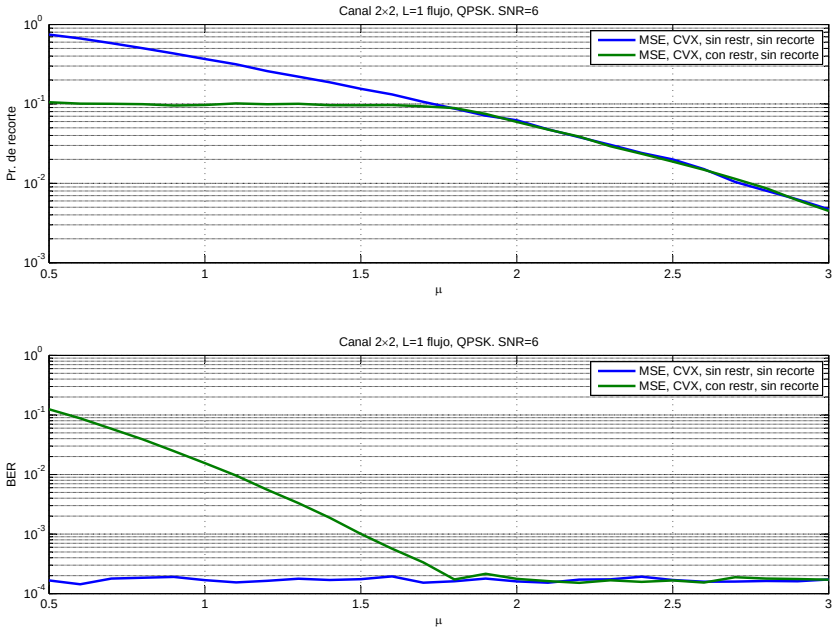

Figura 7.11: Efecto del parámetro μ en un sistema con $n_r = 2$, $n_t = 2$ y $L = 1$, sin recortar la señal.

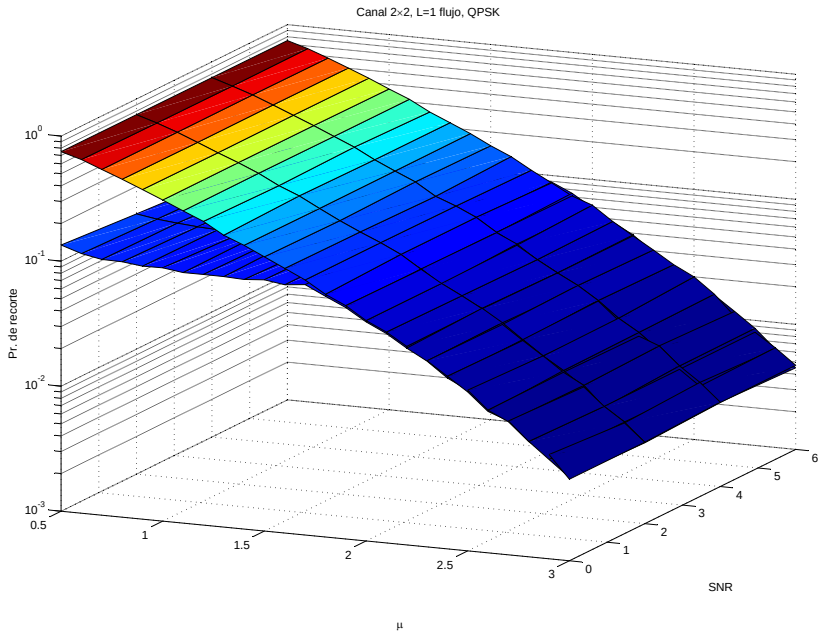

Figura 7.12: Variación de la probabilidad de recorte en un sistema con $n_r = 2$, $n_t = 2$ y $L = 1$, sin recortar la señal.

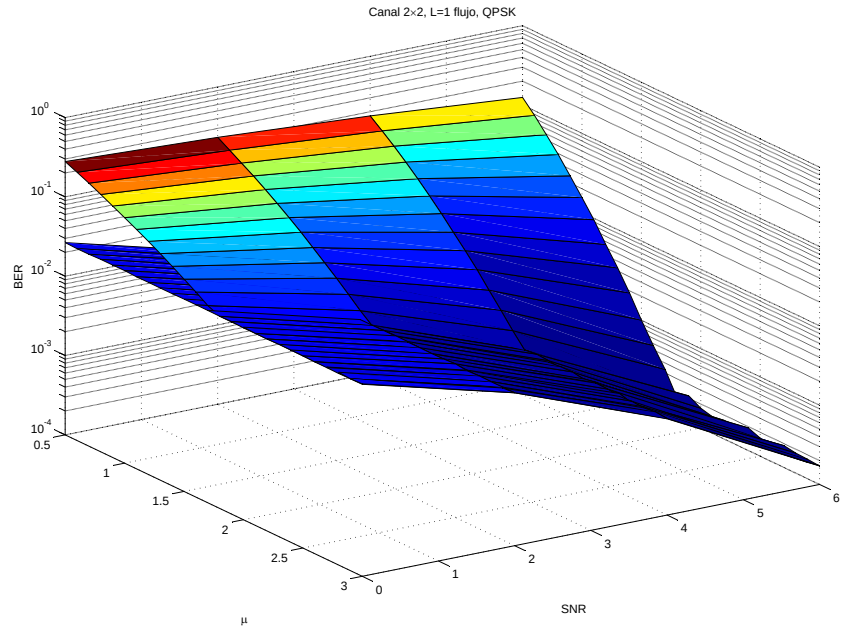

Figura 7.13: Variación de la tasa de error en un sistema con $n_r = 2$, $n_t = 2$ y $L = 1$, sin recortar la señal.

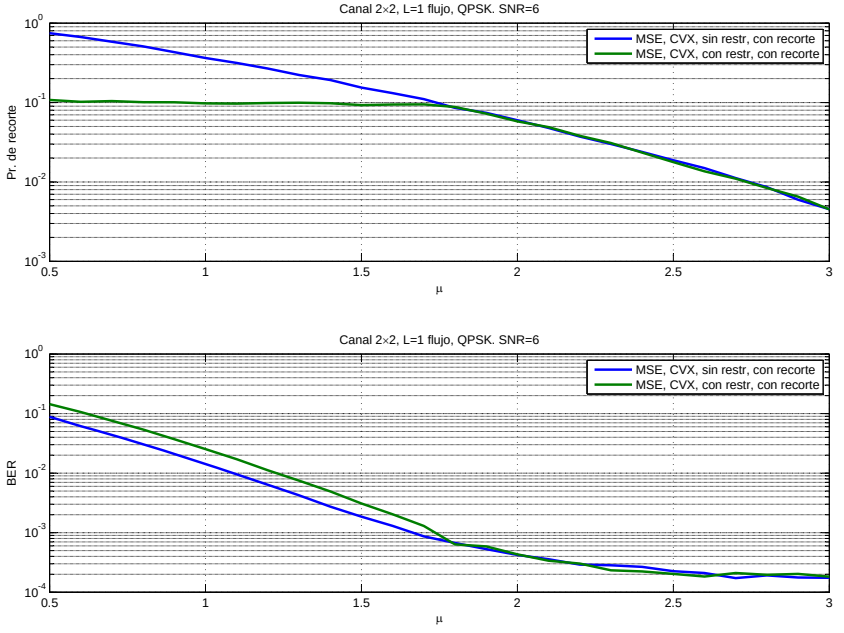

Figura 7.14: Efecto del parámetro μ en un sistema con $n_r = 2$, $n_t = 2$ y $L = 1$ cuando se aplica el recorte en la señal transmitida.

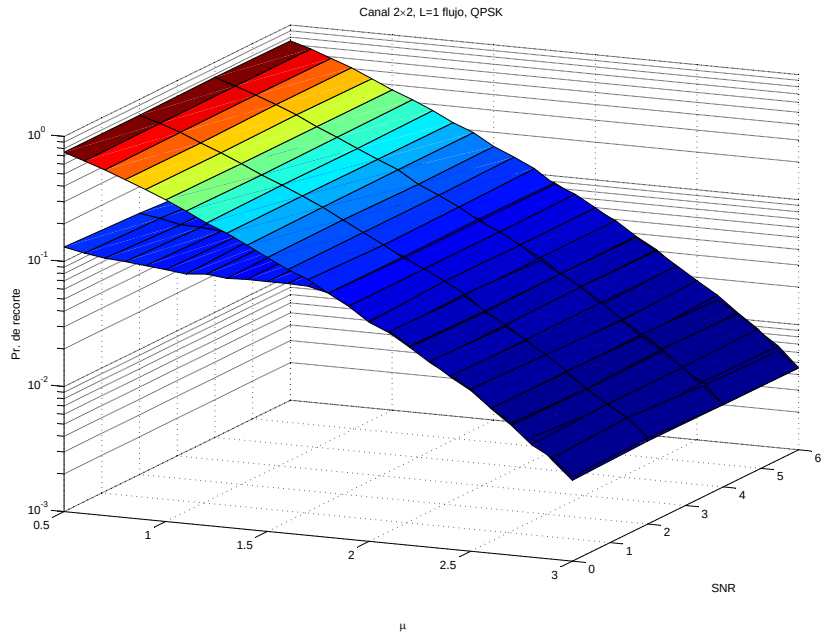

Figura 7.15: Variación de la probabilidad de recorte en un sistema con $n_r = 2$, $n_t = 2$ y $L = 1$ cuando se aplica el recorte en la señal transmitida.

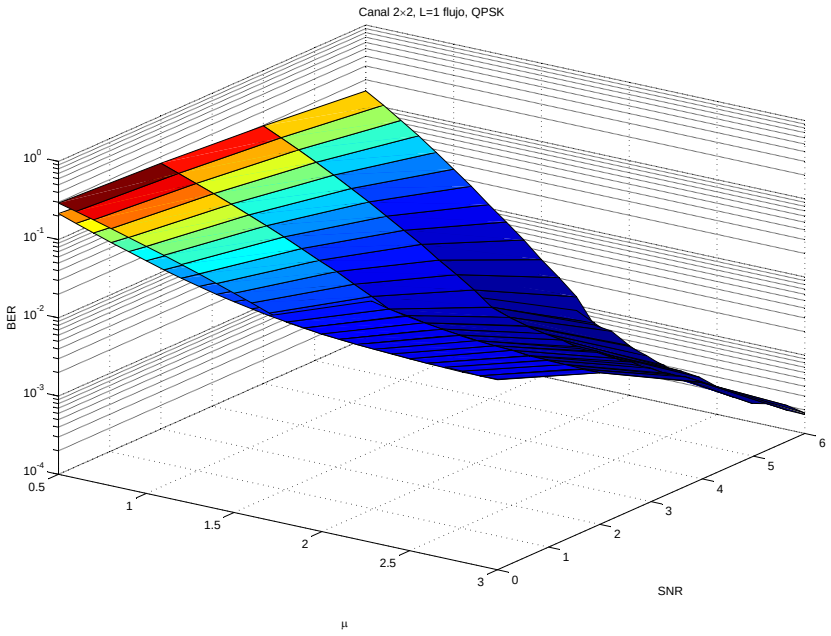

Figura 7.16: Variación de la tasa de error en un sistema con $n_r = 2$, $n_t = 2$ y $L = 1$ cuando se aplica el recorte en la señal transmitida.

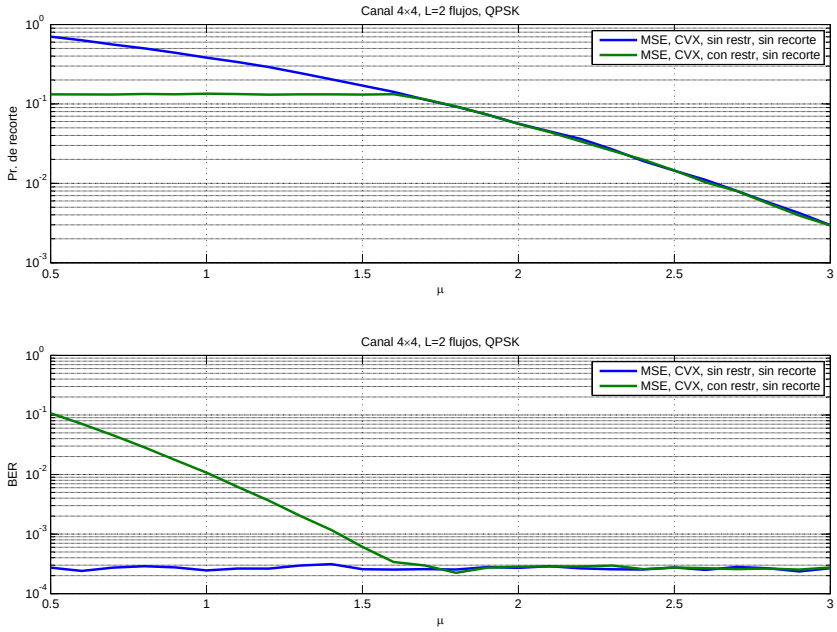

Figura 7.17: Efecto del parámetro μ en un sistema con $n_r = 4$, $n_t = 4$ y $L = 2$.

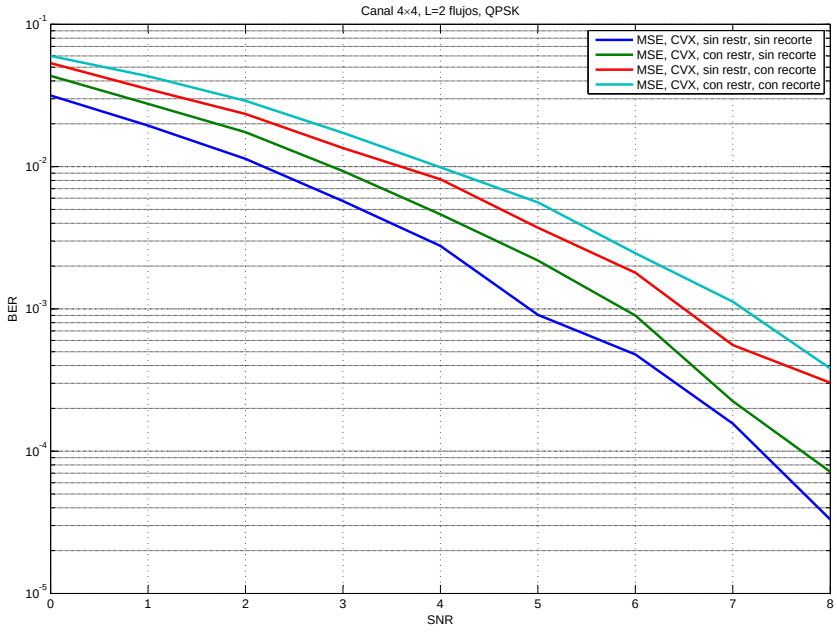

Figura 7.18: BER vs. SNR para el criterio de suma de MSE en un sistema con $n_r = 4$, $n_t = 4$ y $L = 2$, aplicando recorte y sin recortar, con $\mu = 1{,}5$.

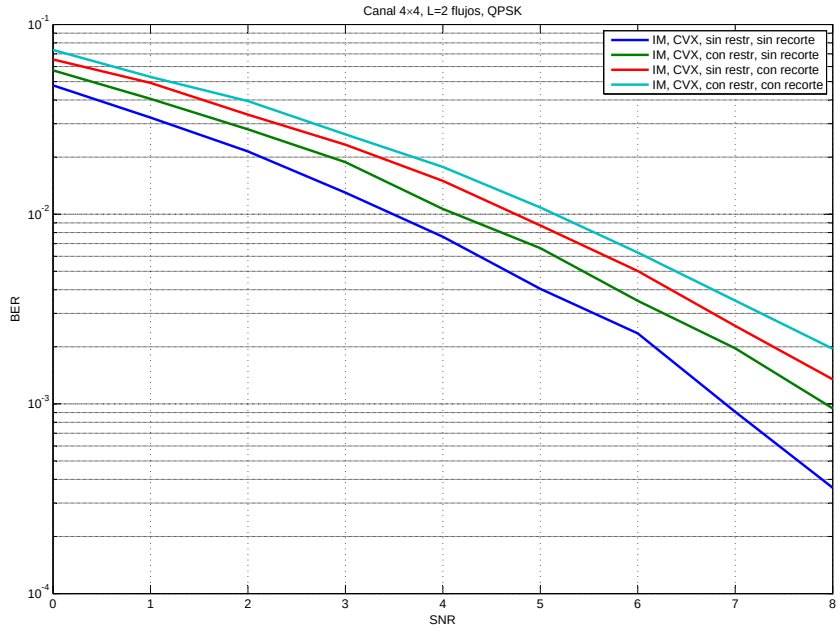

Figura 7.19: BER vs. SNR para el criterio de información mutua en un sistema con $n_r = 4$, $n_t = 4$ y $L = 2$, aplicando recorte y sin recortar, con $\mu = 1{,}5$.

7.7 Conclusiones

En este capítulo he reproducido una parte de los resultados de [Pal03a] aportando la utilización de las herramientas de optimización convexa del tema 5. La principal ventaja de estas herramientas es la facilidad para añadir nuevas restricciones, y se ha demostrado con la limitación de la relación PAR en señales OFDM.

Hemos visto que esta restricción produce una limitación en la potencia transmitida, y los resultados indican que la reducción de potencia tiene un coste en tasa de error, pero se consigue acotar la probabilidad de recorte de la señal. Los resultados también muestran que la utilización de un esquema cooperativo supone una ventaja cuando el sistema no está totalmente cargado, utilizando todos los automodos del canal. Esto viene de haber implementado únicamente criterios cóncavos de Schur, en los que la estructura del problema queda diagonalizada.

Se han utilizado únicamente las herramientas *CVX*, *SeDuMi* y *Mosek* por haber mostrado en el capítulo 6 más fiabilidad y también mayor rapidez. En los tres casos los resultados han sido similares. Lo que ha quedado más claro son los procedimientos que hay que seguir para utilizar estos programas en aplicaciones de comunicaciones. Hemos visto lo sencillo que resulta modelar el problema con *CVX*, por lo que puede ser adecuado para una primera apro-

ximación y ver si nuestro planteamiento es correcto, y una vez comprobado se puede elaborar un planteamiento más complicado para utilizar los *solver* *SeDuMi* y *Mosek* cuyo funcionamiento es más rápido y más adecuado para prototipos o aplicaciones mas allá de la simulación.

El criterio de maximización de la información mutua ha dado los peores resultados, y hemos visto que para resolverlo por *CVX* se utiliza un método de aproximaciones sucesivas que en algunos casos puede no converger a una solución viable, mientras que por otro lado se puede llegar a la solución óptima con mucha facilidad con algoritmos de *water-filling*, tal como vimos en clase de comunicaciones móviles.

Capítulo 8

Conclusiones y líneas futuras

Este proyecto se plantea con el objetivo de realizar un estudio comparativo de las herramientas de optimización convexa más importantes que podemos encontrar en el mercado, y definir casos de aplicación relevantes en el ámbito de las telecomunicaciones.

Para evaluar de forma objetiva el funcionamiento y las características que describen a estas herramientas, se ha presentado en la primera parte de esta memoria un marco teórico que comprende los conceptos imprescindibles para asimilar de forma eficiente la información obtenida sobre cada producto y sus aplicaciones. A continuación, en la segunda parte encontramos una revisión de un conjunto de estos, elegidos con los siguientes criterios de selección: en primer lugar he escogido cuatro sistemas de modelado, *CVX*, *AMPL*, *GAMS* y *YALIMP*, en base a la cantidad de referencias encontradas en la bibliografía consultada sobre optimización convexa y sobre aplicaciones en el ámbito de las telecomunicaciones, y seguidamente he elegido los *solvers* más utilizados en esos paquetes de modelado. Después de examinar todas estas herramientas, en la tercera parte de este documento he presentado dos casos de aplicación directamente relacionados con los estudios de ingeniería de telecomunicación: un conformador robusto de haz en receptores con agrupaciones de antenas, y un sistema de transmisión sobre canales MIMO. Finalmente, este apartado en que nos encontramos sirve para recopilar las conclusiones extraídas en cada uno de los bloques precedentes de la memoria, aunque también podemos encontrar secciones al final de los capítulos 5, 6 y 7 dedicadas a este mismo argumento y con un desarrollo más extenso y específico sobre cada tema.

La idea más importante que he sacado de los capítulos de teoría que componen la primera parte es que la optimización convexa se centra en el estudio de un tipo específico de problemas de minimización o maximización de funciones porque sus propiedades nos permiten resolver casos con grandes dimensiones de forma muy eficiente en términos de coste computacional y precisión. En una gran cantidad de ocasiones nos encontramos con la

evidencia de que la convexidad de un problema constituye la línea divisoria entre su condición de ser fácil o difícil de resolver.

En la segunda parte de este documento (el capítulo 5) hemos resuelto el objetivo de aclarar la diferencia entre sistemas de modelado y programas de resolución, por medio del análisis de características de un conjunto representativo de herramientas. De este modo, hemos visto que los *solvers* son programas especializados en un número reducido de tipos de problemas convexos, y se encargan de implementar algoritmos de resolución. En la mayor parte de los productos que hemos repasado se utiliza alguna variante de los métodos de punto interior estudiados en el capítulo 4. Por otro lado, los sistemas de modelado se encargan de proporcionar al usuario un lenguaje de programación de alto nivel que le permita describir un modelo matemático convexo consistente en un conjunto de variables, funciones, conjuntos, y otros elementos, que definen la función objetivo y las restricciones del problema. A partir de ahí, las herramientas que hemos visto se encargan de transformar el problema a un formato compatible con los *solvers* que éstas incluyen, para obtener un resultado y demostrar su viabilidad o la carencia de ésta. También hemos podido analizar tres paquetes de optimización global orientados a problemas que no necesariamente han de ser convexos, y la conclusión a la que me ha llevado es que existe un compromiso entre generalidad y especialización, que en nuestro caso se traduce entre ser capaces de resolver una gran variedad de problemas frente a poder tratar un número elevado de variables y funciones de restricción.

El primer caso práctico, de conformación de haz, nos sirve para consolidar todos los conceptos estudiados en los capítulos precedentes. Por un lado, nos muestra en detalle la forma de trabajar con diferentes herramientas: un sistema de modelado, *CVX*; dos *solvers*, *SeDuMi* y *Mosek*; y dos paquetes de optimización global, *Matlab Optimization Toolbox* y *Mathematica*. Al tener que trabajar en una aplicación real, hemos podido comprobar las diferencias en cuanto al modo de utilización de cada herramienta y también las ventajas y desventajas que nos ofrece cada enfoque. En el caso del sistema de modelado *CVX* hemos visto que desde el punto de vista del usuario que parte de un problema convexo, este programa es la aproximación más sencilla de utilizar porque permite una sintaxis muy similar a la descripción matemática del problema, mientras que los programas de resolución *SeDuMi* o *Mosek* requieren un procesamiento y unas transformaciones más exigentes en cuanto a las aptitudes matemáticas del usuario, que necesita convertir el problema original en un formato estándar, como por ejemplo un problema cónico. En algunos casos, podemos encontrarnos con que el problema a resolver es directamente cuadrático o lineal, como ocurre en el ejercicio de §5.5. En este caso podremos utilizar cualquier herramienta de las que vimos en §5.2. En concreto, su implementación en *Mosek* no presentaba ninguna dificultad. La principal ventaja que aporta invertir más esfuerzo por parte del usuario es que en la implementación final con un *solver* nos evitamos

incluir el resto de recursos del sistema de modelado que no se necesitan en la aplicación final. Además, en el caso de *Mosek*, podemos extraer el código en varios lenguajes de programación como C, Java, etc. para conseguir un programa ejecutable, liberando a la aplicación final de la necesidad de instalar *Matlab*. Por otro lado, los resultados del capítulo 6 nos muestran que las herramientas de resolución se ejecutan con mayor velocidad que el resto. También nos hacen ver que los paquetes de optimización global pueden llevarnos a resultados erróneos con cierta facilidad, por lo que entendemos que su utilización es más adecuada en casos en los que no tengamos claro el camino para convertir el problema en convexo y las funciones utilizadas sean fácilmente derivables.

En el segundo caso práctico, de transmisión en canales MIMO, nos limitamos a utilizar herramientas de optimización convexa. Hemos aplicado el método de diseño más razonable que se puede extraer de las conclusiones del capítulo anterior: en la primera aproximación al problema utilizamos el sistema de modelado *CVX*, que nos permite obtener los primeros resultados, y comprobar que el planteamiento es correcto y da soluciones viables. Una vez resuelto de este modo, pasamos a desarrollar el proceso de transformación del problema en un formato estándar, en nuestro caso en un problema cónico de segundo orden. Esto nos permite utilizar directamente las herramientas de resolución *SeDuMi* y *Mosek* para poder comparar los resultados. Sin embargo, el enfoque que he buscado en este ejemplo es buscar la facilidad para añadir restricciones adicionales al diseño, por lo que la herramienta más adecuada que refleja esta propiedad sería *CVX* o en general, los sistemas de modelado. En concreto, se ha añadido una restricción que reduce el efecto de los picos de potencia que se producen en las señales OFDM, tal como se describe en §7.3.5. La técnica de recorte utilizada para reducir la relación PAR es la solución más simple, pero la complementamos con una reducción de potencia que según los resultados de §7.6.3 consigue controlar la probabilidad de recorte. Sin embargo, a pesar de que las gráficas 7.11, 7.12 y 7.13 reproducen los resultados publicados en [Pal03a], en este trabajo ampliamos las simulaciones añadiendo el efecto que tiene la reducción de potencia de nuestra restricción en la tasa de error, y comprobamos en las figuras 7.14, 7.15 y 7.16 que en este caso, la solución propuesta no supone un beneficio en tasa de error respecto al problema que inicialmente suponía el efecto del recorte. Nuestro desarrollo abre la puerta a la ampliación con técnicas más avanzadas que se pueden añadir como restricciones adicionales. Además de las conclusiones respecto a las herramientas de optimización convexa, también he profundizado en el conocimiento sobre transmisión MIMO-OFDM.

En definitiva, si agrupamos las conclusiones de este trabajo, el rango de aplicaciones implementadas hace que se haya presentado una muestra importante de técnicas estudiadas en la primera parte, y también de diferentes variedades de herramientas existentes. Así, por ejemplo, hemos visto cómo se utiliza *Mosek* como optimizador cuadrático en §5.5.2, de forma muy distinta

a su uso como optimizador cónico en §6.3.3 o en §7.4.3. De las herramientas utilizadas, *CVX* es más apropiada para la primera aproximación a los problemas, ya que permite hacer una descripción de forma muy intuitiva. En algunos casos puede resultar más sencilla la sintaxis de *Matlab Optimization Toolbox* o especialmente la de *Mathematica*, que permite representaciones simbólicas y no sólo numéricas. Sin embargo, los resultados del capítulo 6 muestran que nos podemos encontrar con fallos de convergencia que no aparecen en el mismo problema desde otras herramientas de optimización convexa.

8.1 Líneas futuras

Al terminar de implementar cada aplicación me he planteado un gran abanico de ampliaciones que han quedado como posibles líneas futuras por la necesidad de acotar el tiempo de este proyecto. La bibliografía revisada para profundizar en cada tema presenta una gran variedad de alternativas en las que las herramientas de optimización convexa pueden ser un recurso importante.

En el capítulo de conformación de haz me he centrado en el diseño del receptor, aunque se puede extender también al transmisor, o incluso a esquemas cooperativos en redes de retransmisión [Cov79, Gas05], así como a sistemas de multidifusión. La conformación de haz para difusión múltiple es un servicio que forma parte del sistema UMTS-LTE [Mot07, Kar08]. Un estudio detallado de esta aplicación se encuentra en [Ntr09].

Después de las simulaciones del capítulo 7 la primera ampliación que me he planteado consiste en proporcionar las curvas con valores medidos para una probabilidad de corte determinada (lo que estudiábamos como valores de *outage*), pero el objetivo principal de este proyecto no es tanto aportar una medida objetiva de los resultados como llegar a conocer la metodología necesaria y los recursos disponibles para añadir restricciones a los problemas existentes desde el enfoque de optimización convexa.

Por otro lado, he presentado únicamente dos criterios de optimización con funciones cóncavas de Schur principalmente por enlazar con el caso que hemos visto en la asignatura de comunicaciones móviles. En [Pal03a, Pal05, Pal03b] encontramos un conjunto generalizado de alternativas, y de hecho, los resultados en las funciones convexas de Schur permiten una redistribución de la información entre los flujos de datos utilizados, de forma que se puede sobredimensionar la utilización por encima del rango de la matriz del canal.

En cualquier caso, con la selección de simulaciones y aplicaciones presentadas en este proyecto se ha utilizado una gran variedad de recursos que me han permitido profundizar en el conocimiento no sólo de las técnicas de optimización convexa, sino también en el amplio abanico de líneas de investigación abiertas en las que se puede aplicar.

Apéndice A

Técnicas de resolución de ecuaciones lineales

En este apéndice he querido agrupar algunas de las técnicas de álgebra que he mencionado en algunos capítulos de este proyecto, y que son en gran medida un motivo de que los problemas de optimización convexa se puedan resolver de forma muy eficiente. Recordemos que en definitiva, los algoritmos de resolución terminan descomponiendo el problema en una serie de sistemas de ecuaciones lineales, por lo que es necesario que estos se resuelvan de forma inteligente, aprovechando la estructura de las matrices involucradas. Para ampliar la teoría de este capítulo podemos recurrir a [Tre97] y [Zha05].

A.1 Factorización LU

Toda matriz no singular $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ se puede factorizar como

$$\boldsymbol{A} = \boldsymbol{PLU}$$

donde $\boldsymbol{P} \in \mathbb{R}^{n\times n}$ es una matriz de permutaciones, $\boldsymbol{L} \in \mathbb{R}^{n\times n}$ es una matriz triangular inferior compuesta por unos, y $\boldsymbol{U} \in \mathbb{R}^{n\times n}$ es una matriz triangular superior y no singular. Esto se conoce como la factorización LU de $\boldsymbol{A}$. También podemos escribir la factorización como $\boldsymbol{P}^T\boldsymbol{A} = \boldsymbol{LU}$, donde la matriz $\boldsymbol{P}^T\boldsymbol{A}$ se obtiene de una reordenación de las filas de $\boldsymbol{A}$. El algoritmo estándar para calcular la factorización LU se llama *eliminación gaussiana con pivotado parcial* o *eliminación gaussiana con pivotado de filas*.

Este método se utiliza habitualmente para resolver ecuaciones lineales $\boldsymbol{Ax} = \boldsymbol{b}$ según vemos en el algoritmo A.1. El razonamiento es muy sencillo, y se basa en añadir variables auxiliares y componer las siguientes ecuaciones

$$\begin{aligned}
\boldsymbol{Ax} = \boldsymbol{b} \quad &\Rightarrow \quad \boldsymbol{PLUx} = \boldsymbol{b} \\
\boldsymbol{Pz}_1 = \boldsymbol{b} \quad &\Rightarrow \quad \boldsymbol{PLUx} = \boldsymbol{Pz}_1 \quad \Rightarrow \quad \boldsymbol{LUx} = \boldsymbol{z}_1 \\
\boldsymbol{Lz}_2 = \boldsymbol{z}_1 \quad &\Rightarrow \quad \boldsymbol{LUx} = \boldsymbol{Lz}_2 \quad \Rightarrow \quad \boldsymbol{Ux} = \boldsymbol{z}_2
\end{aligned}$$

Algoritmo A.1: Resolución de ecuaciones lineales con factorización LU

Datos: un conjunto de ecuaciones lineales $\boldsymbol{Ax} = \boldsymbol{b}$, con $\boldsymbol{A}$ no singular

Factorización LU: $\boldsymbol{A} = \boldsymbol{PLU}$
Permutación: Resolver $\boldsymbol{Pz}_1 = \boldsymbol{b}$
Sustitución directa: Resolver $\boldsymbol{Lz}_2 = \boldsymbol{z}_1$
Sustitución inversa: Resolver $\boldsymbol{Ux} = \boldsymbol{z}_2$

En algunos casos, la estructura de la matriz $\boldsymbol{A} \in \mathbb{R}^{n \times n}$ permite que la factorización LU tenga un coste computacional mucho menor, como por ejemplo cuando es una matriz es dispersa o cuando también cuando está limitada en banda, es decir, $a_{ij} = 0$ si $|i - j| > k$, con $k < n - 1$. En este caso decimos que el ancho de banda de $\boldsymbol{A}$ es k.

A.2 Factorización de Cholesky

Si $\boldsymbol{A} \in \mathbb{R}^{n \times n}$ es simétrica y definida positiva, entonces se puede factorizar como

$$\boldsymbol{A} = \boldsymbol{LL}^T$$

donde $\boldsymbol{L}$ es triangular inferior y no singular, con elementos positivos en la diagonal. Esto se llama factorización de Cholesky de $\boldsymbol{A}$ y se puede interpretar como una factorización LU simétrica, con $\boldsymbol{L} = \boldsymbol{U}^T$. La matriz $\boldsymbol{L}$ es el factor de Cholesky de $\boldsymbol{A}$.

El coste computacional de esta factorización es la mitad que la técnica LU, y se puede utilizar para resolver ecuaciones lineales con el algoritmo A.2.

Algoritmo A.2: Resolución de ecuaciones lineales con factorización de Cholesky

Datos: un conjunto de ecuaciones lineales $\boldsymbol{Ax} = \boldsymbol{b}$, con $\boldsymbol{A} \in \mathbb{S}^n_{++}$

Factorización de Cholesky: $\boldsymbol{A} = \boldsymbol{LL}^T$
Sustitución directa: Resolver $\boldsymbol{Lz}_1 = \boldsymbol{b}$
Sustitución inversa: Resolver $\boldsymbol{L}^T\boldsymbol{x} = \boldsymbol{z}_1$

A.3 Complemento de Schur

El complemento de Schur se puede utilizar para resolver el sistema $\boldsymbol{Ax} = \boldsymbol{b}$ eliminando un subconjunto de variables y resolviendo después un sistema

más pequeño con las variables restantes. Este método permite una gran efectividad de cálculo cuando la submatriz asociada a las variables eliminadas tiene una estructura que se puede factorizar fácilmente, por ejemplo, si es diagonal por bloques, o en la forma matricial de las ecuaciones KKT del capítulo 4.

En este método partimos la variable $\boldsymbol{x} \in \mathbb{R}^n$ en dos bloques o subvectores,

$$\boldsymbol{x} = \begin{bmatrix} \boldsymbol{x}_1 \\ \boldsymbol{x}_2 \end{bmatrix}$$

donde $\boldsymbol{x}_1 \in \mathbb{R}^{n_1}$ y $\boldsymbol{x}_2 \in \mathbb{R}^{n_2}$. Siguiendo esta línea, las ecuaciones lineales $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ quedan del siguiente modo

$$\begin{bmatrix} \boldsymbol{A}_{11} & \boldsymbol{A}_{12} \\ \boldsymbol{A}_{21} & \boldsymbol{A}_{22} \end{bmatrix} \begin{bmatrix} \boldsymbol{x}_1 \\ \boldsymbol{x}_2 \end{bmatrix} = \begin{bmatrix} \boldsymbol{b}_1 \\ \boldsymbol{b}_2 \end{bmatrix} \tag{A.1}$$

donde $\boldsymbol{A}_{11} \in \mathbb{R}^{n_1 \times n_1}$ y $\boldsymbol{A}_{22} \in \mathbb{R}^{n_2 \times n_2}$. Si se puede invertir $\boldsymbol{A}_{11}$ podemos eliminar $\boldsymbol{x}_1$ expresándolo en función de $\boldsymbol{x}_2$ de acuerdo con la primera ecuación de (A.1)

$$\boldsymbol{x}_1 = \boldsymbol{A}_{11}^{-1} \left(\boldsymbol{b}_1 - \boldsymbol{A}_{12}\boldsymbol{x}_2 \right) \tag{A.2}$$

Sustituyendo esta expresión en la segunda ecuación de (A.1) obtenemos la siguiente ecuación reducida

$$\left(\boldsymbol{A}_{22} - \boldsymbol{A}_{21}\boldsymbol{A}_{11}^{-1}\boldsymbol{A}_{12} \right) \boldsymbol{x}_2 = \boldsymbol{b}_2 - \boldsymbol{A}_{21}\boldsymbol{A}_{11}^{-1}\boldsymbol{b}_1 \tag{A.3}$$

La matriz que aparece entre paréntesis en (A.3) se llama complemento de Schur del primer bloque $\boldsymbol{A}_{11}$:

$$\boldsymbol{S} = \boldsymbol{A}_{22} - \boldsymbol{A}_{21}\boldsymbol{A}_{11}^{-1}\boldsymbol{A}_{12}$$

Cuando es sencillo factorizar $\boldsymbol{A}_{11}$ calculamos el complemento de Schur $\boldsymbol{S}$ y resolvemos el sistema con el algoritmo A.3.

Algoritmo A.3: Resolución de ecuaciones lineales por eliminación en bloque

---

Datos: un conjunto de ecuaciones lineales (A.1), en el que $\boldsymbol{A}_{11}$ no es singular

Calcular $\boldsymbol{A}_{11}^{-1}\boldsymbol{A}_{12}$ y $\boldsymbol{A}_{11}^{-1}\boldsymbol{b}_1$
Formar $\boldsymbol{S} = \boldsymbol{A}_{22} - \boldsymbol{A}_{21}\boldsymbol{A}_{11}^{-1}\boldsymbol{A}_{12}$ y $\tilde{\boldsymbol{b}} = \boldsymbol{b}_2 - \boldsymbol{A}_{21}\boldsymbol{A}_{11}^{-1}\boldsymbol{b}_1$
Determinar $\boldsymbol{x}_2$ resolviendo $\boldsymbol{S}\boldsymbol{x}_2 = \tilde{\boldsymbol{b}}$
Determinar $\boldsymbol{x}_1$ resolviendo $\boldsymbol{A}_{11}\boldsymbol{x}_1 = \boldsymbol{b}_1 - \boldsymbol{A}_{12}\boldsymbol{x}_2$

En el caso de las ecuaciones KKT encontramos la siguiente estructura:

$$\boldsymbol{A} = \begin{bmatrix} \boldsymbol{A}_{11} & \boldsymbol{A}_{12} \\ \boldsymbol{A}_{12}^T & \boldsymbol{0} \end{bmatrix}$$

donde $\boldsymbol{A}_{11} \in \mathbb{S}_{++}^p$ y $\boldsymbol{A}_{12} \in \mathbb{R}^{p\times m}$, y el rango de $\boldsymbol{A}_{12}$ es m. Como $\boldsymbol{A}_{11} \succ \boldsymbol{0}$ podemos factorizarlo por Cholesky. El complemento de Schur $\boldsymbol{S} = -\boldsymbol{A}_{12}^T\boldsymbol{A}_{11}^{-1}\boldsymbol{A}_{12}$ es definido negativo, por lo que podemos factorizar $-\boldsymbol{S}$ por Cholesky, y con estos datos podemos aplicar el algoritmo A.3 de forma muy eficiente.

A.4 El lema de inversión matricial

La idea de la eliminación en bloque en el algoritmo A.3 es quitar variables y resolver un conjunto de ecuaciones más pequeño utilizando el complemento de Schur de la matriz original con respecto a las variables eliminadas. La misma idea se puede hacer a la inversa: cuando reconocemos que una matriz como un complemento de Schur, podemos introducir nuevas variables y crear un conjunto de ecuaciones más grande para resolverlo. En la mayoría de los casos esto no supone ninguna ventaja, ya que acabamos con un sistema de ecuaciones más grande, pero cuando este sistema aumentado tiene una estructura especial, entonces esta opción puede llevar a un método eficiente. El caso más común se da cuando se elimina otro bloque de variables de la matriz más grande.

Comenzamos con el siguiente sistema de ecuaciones:

$$(\boldsymbol{A} + \boldsymbol{BC})\,\boldsymbol{x} = \boldsymbol{b} \tag{A.4}$$

donde $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ se puede invertir, y $\boldsymbol{B} \in \mathbb{R}^{n\times p}$, $\boldsymbol{C} \in \mathbb{R}^{p\times n}$. Añadimos una nueva variable $\boldsymbol{y} = \boldsymbol{Cx}$ y volvemos a escribir las ecuaciones como

$$\boldsymbol{Ax} + \boldsymbol{By} = \boldsymbol{b}, \qquad \boldsymbol{y} = \boldsymbol{Cx}$$

que se puede expresar en forma matricial

$$\begin{bmatrix} \boldsymbol{A} & \boldsymbol{B} \\ \boldsymbol{C} & -\boldsymbol{I} \end{bmatrix} \begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{y} \end{bmatrix} = \begin{bmatrix} \boldsymbol{b} \\ \boldsymbol{0} \end{bmatrix} \tag{A.5}$$

Observamos que la matriz de coeficientes original, $\boldsymbol{A} + \boldsymbol{BC}$, es el complemento de Schur de $-\boldsymbol{I}$ en la matriz aumentada que aparece en (A.5). Si tuviésemos que eliminar la variable $\boldsymbol{y}$ de (A.5), volveríamos a la ecuación anterior (A.4).

En algunos casos, puede ser más eficiente resolver el sistema aumentado (A.5) en lugar del sistema de ecuaciones original (A.4) a pesar de ser más grande. Este sería el caso, por ejemplo, si $\boldsymbol{A}$, $\boldsymbol{B}$ y $\boldsymbol{C}$ fuesen mucho menos densas que la matriz $\boldsymbol{A} + \boldsymbol{BC}$.

Después de introducir la nueva variable $\boldsymbol{y}$, podemos eliminar la variable original $\boldsymbol{x}$ del conjunto aumentado de ecuaciones (A.5), usando $\boldsymbol{x} = \boldsymbol{A}^{-1}(\boldsymbol{b} - \boldsymbol{B}\boldsymbol{y})$. Si lo sustituimos en la segunda ecuación $\boldsymbol{y} = \boldsymbol{C}\boldsymbol{x}$ obtenemos

$$\left(\boldsymbol{I} + \boldsymbol{C}\boldsymbol{A}^{-1}\boldsymbol{B}\right)\boldsymbol{y} = \boldsymbol{C}\boldsymbol{A}^{-1}\boldsymbol{b}$$

de donde podemos despejar $\boldsymbol{y}$

$$\boldsymbol{y} = \left(\boldsymbol{I} + \boldsymbol{C}\boldsymbol{A}^{-1}\boldsymbol{B}\right)^{-1}\boldsymbol{C}\boldsymbol{A}^{-1}\boldsymbol{b}$$

Si usamos $\boldsymbol{x} = \boldsymbol{A}^{-1}(\boldsymbol{b} - \boldsymbol{B}\boldsymbol{y})$ obtenemos

$$\boldsymbol{x} = \left(\boldsymbol{A}^{-1} - \boldsymbol{A}^{-1}\boldsymbol{B}\left(\boldsymbol{I} + \boldsymbol{C}\boldsymbol{A}^{-1}\boldsymbol{B}\right)^{-1}\boldsymbol{C}\boldsymbol{A}^{-1}\right)\boldsymbol{b}$$

Como $\boldsymbol{b}$ es arbitrario, llegamos a la conclusión de que

$$(\boldsymbol{A} + \boldsymbol{B}\boldsymbol{C})^{-1} = \boldsymbol{A}^{-1} - \boldsymbol{A}^{-1}\boldsymbol{B}\left(\boldsymbol{I} + \boldsymbol{C}\boldsymbol{A}^{-1}\boldsymbol{B}\right)^{-1}\boldsymbol{C}\boldsymbol{A}^{-1}$$

Esto es lo que se conoce como el lema de inversión matricial o fórmula de Sherman-Woodbury-Morrison.

Apéndice B

Presupuesto del proyecto

B.1 Fases del proyecto

Para justificar los costes de este proyecto, en este apartado se describen las fases de elaboración y desarrollo que se han seguido, con comentarios sobre aspectos y observaciones que han surgido en la elaboración de cada tarea. La duración de cada una se resume en la tabla B.1.

- Fundamentos teóricos
  - Formación básica sobre optimización convexa.
  - Resumen de los conceptos más importantes. Extraer los fundamentos imprescindibles a incluir en esta memoria.
- Revisión de herramientas
  - Búsqueda y selección de herramientas más relevantes.
  - Extracción de características de cada producto.
  - Solución del ejemplo §5.5.
- Selección de casos prácticos. Buscar bibliografía sobre aplicaciones de optimización convexa en casos relacionados con las telecomunicaciones.
- Conformación de haz
  - Conseguir el diagrama de radiación de una simulación sencilla (no robusta) con *CVX*. Este primer caso no queda reflejado en esta memoria por resultar trivial.
  - Simulación robusta optimizada con *CVX*.
  - Selección de la región de incertidumbre.
  - Implementación con otras herramientas. Adaptar el problema a varios programas.

  - Elaboración de las gráficas de resultados.
  - Extraer conclusiones y preparar el capítulo correspondiente en la memoria.

- Canales MIMO-OFDM
  - Simulación de una transmisión simple, con *CVX* y utilizando el criterio de minimizar la suma de MSE.
  - Adaptación a la consola de pruebas de error de *Matlab* para poder obtener curvas de BER en función de SNR.
  - Añadir restricciones de PAPR en una transmisión simple, y adaptarlo a la consola de pruebas de error para conseguir las curvas de probabilidad de recorte.
  - Implementar el problema con otras herramientas y añadir el criterio de maximización de información mutua.
  - Ejecutar los ensayos y recopilar gráficas de resultados.
  - Extraer conclusiones y añadirlas al capítulo correspondiente en la memoria.

- Revisión de la memoria del proyecto. Tarea ejecutada con el director del proyecto.

Fase 1	*Fundamentos teóricos*	460 *horas*
Fase 2	*Herramientas de optimización*	330 *horas*
Fase 3	*Selección de casos prácticos*	195 *horas*
Fase 4	*Conformación de haz*	525 *horas*
Fase 5	*Canales MIMO-OFDM*	720 *horas*
Fase 6	*Revisión de la memoria*	525 *horas*
	Total	2.755 *horas*

Tabla B.1: Fases del Proyecto

En definitiva, una vez repasado el proceso de elaboración del proyecto, cuantificamos el total de horas destinadas a este proyecto según la tabla B.1 y obtenemos de 2,755, que equivalen a 20,99 hombres-mes.

B.2 Desglose presupuestario

Hasta el año 2008 el COIT[1] ha venido publicando unas listas de honorarios orientativos que en los dos últimos años se denominaron *costes estimados*

[1] Colegio oficial de ingenieros de telecomunicación

de trabajos profesionales [COI11]. Por modificación de la Ley de Colegios Profesionales 25/2009 de 22 de diciembre, no es posible seguir publicando estas listas. El Colegio no puede elaborar baremos de honorarios, ni siquiera orientativos, por lo que he utilizado los valores de la plantilla publicada en nuestra universidad [UC311], según la cual, 1 hombre-mes equivale a 131, 25 horas, y los costes estimados por hombre-mes para un ingeniero de telecomunicación vienen reflejados en la tabla B.2, correspondiente a los costes de personal.

Cargo	Categoría	Dedicación	Coste por hombre-mes	Coste
Director de proyecto	*Doctor Ingeniero*	2,20 *h-m*	4.289,54	9.436,99 €
Ingeniero de proyecto	*Ingeniero*	20,99 *h-m*	2.694,39	56.555,25 €
	Hombres-mes	23,19	Total	65.992,23 €

Tabla B.2: Costes de personal

En la tabla B.3 encontramos el coste de amortización de los equipos y el *software* comercial utilizados en el proyecto. La fórmula de cálculo corresponde a $\frac{A}{B} \times C \times D$, donde A es el número de meses desde la fecha de facturación en que se utiliza el equipo; B es el período de depreciación, cuyo valor es de 60 meses conforme a [UC311]; C es el coste del equipo sin IVA[2]; y D es el porcentaje de uso que se dedica al proyecto, que en nuestro caso es el 100 %.

Descripción	Coste	Uso	Dedicación	Período de depreciación	Coste imputable
Ordenador	850,00	100 %	21 *meses*	60 *meses*	297,50 €
Matlab r2010a Student Version	500,00	100 %	12 *meses*	60 *meses*	100,00 €
Matlab Optimization Toolbox	200,00	100 %	12 *meses*	60 *meses*	40,00 €
Matlab Communications Toolbox	200,00	100 %	12 *meses*	60 *meses*	40,00 €
Mathematica 8 for students	153,60	100 %	12 *meses*	60 *meses*	30,72 €
				Total	508,22 €

Tabla B.3: Costes de equipos

La tabla B.4 recoge otros costes directos del proyecto, como el local, material fungible, etc., que ascienden a 1.640 €.

A partir de todos estos datos, el presupuesto total se desglosa en la tabla B.5.

[2]Impuesto sobre el Valor Añadido

Descripción	Coste imputable
Local	1.440,00 €
Gastos administrativos	200,00 €
Total	1.640,00 €

Tabla B.4: Otros costes directos del proyecto

Concepto	Costes totales
Personal	65.992,23 €
Amortización de equipos	508,22 €
Costes de funcionamiento	1640,00 €
Costes indirectos	13.621,95 €
Total	81.768,54 €

Tabla B.5: Coste total del proyecto

Por todo lo anterior, el presupuesto asciende a la cantidad de OCHENTA Y UN MIL SETECIENTOS SESENTA Y OCHO EUROS CON CINCUENTA Y CUATRO CÉNTIMOS.

Leganés a 6 de Septiembre de 2011

El ingeniero proyectista

Fdo. Emilio Mejía Fernández de Velasco

Bibliografía

[Aba69] J. Abadie y J. Carpentier, *Generalization of the Wolfe reduced gradient method to the case of nonlinear constraints. (With discussion)*, pp. 37–47, en "Optimization (Sympos., Univ. Keele, Keele, 1968)" (R. Fletcher, ed.), Academic Press, London, 1969. MR 0284206 (44 #1435)

[AMP11] AMPL Optimization LLC, *Key AMPL Features*, 2011, Web, `http://www.ampl.com/key.html`.

[Art07] A. Artés, F. Pérez, J. Cid, R. López, C. Mosquera, y F. Pérez, *Comunicaciones Digitales*, Pearson Educación, 2007.

[Bar04] B. Barrowes, *Mathematica Symbolic Toolbox for MATLAB - Version 2.0*, 2004, disponible online, `http://library.wolfram.com/infocenter/MathSource/5344/#downloads`.

[Ber11] M. Berkelaar, J. Dirks, K. Eikland, P. Notebaert, y J. Ebert, *lp_solve Reference Guide*, Eindhoven University of Technology, 2011, Web, `http://lpsolve.sourceforge.net/5.0/`.

[Boy04] S. Boyd y L. Vandenberghe, *Convex Optimization*, Cambridge University Press, mar 2004.

[Boy08] S. Boyd, *Convex Optimization I, EE364A Lecture Videos*, 2008, Web de la Universidad de Stanford, `http://www.stanford.edu/class/ee364a/videos.html`.

[Car02] A. Cardama, L. Jofre, J. M. Rius, J. Romeu, S. Blanch, y M. Ferrando, *Antenas*, Ediciones UPC, S.L., sep 2002.

[Chi09] W.-Y. Chiu y B.-S. Chen, *Mobile location estimation in urban areas using mixed Manhattan/Euclidean norm and convex optimization*, Wireless Communications, IEEE Transactions on **8** (2009), no. 1, 414 –423.

[Chi10] J. W. Chinneck, *Practical Optimization: A Gentle Introduction*, 2010, Web, `http://www.sce.carleton.ca/faculty/chinneck/po.html`.

[COI11] Colegio Oficial de Ingenieros de Telecomunicación, *La práctica del ejercicio profesional por los ingenieros de telecomunicación*, jul 2011, Web, `http://www.coit.es/descargar.php?idfichero=4696`.

[Cor09] T. H. Cormen, C. E. Leiserson, R. L. Rivest, y C. Stein, *Introduction to Algorithms*, 3rd ed., The MIT Press, Cambridge Massachussets, 2009.

[Cov79] T. Cover y A. Gamal, *Capacity theorems for the relay channel*, Information Theory, IEEE Transactions on **25** (1979), no. 5, 572 – 584.

[CPL10] IBM Corporation, Software Group, *IBM ILOG CPLEX Optimizer, High Performance Mathematical Optimization Engines*, 2010, Web, `http://public.dhe.ibm.com/common/ssi/ecm/en/wsd14044usen/WSD14044USEN.PDF`.

[Dat06] J. Dattorro, *Convex Optimization & Euclidean Distance Geometry*, Meboo Publishing, jul 2006.

[Eij92] V. Eijkhout, *Distributed sparse data structures for linear algebra operations*, Tech. Report CS 92-169, Computer Science Department, University of Tennessee, Knoxville, TN, 1992, `http://www.netlib.org/lapack/lawns/lawn50.ps`.

[Esc10] J. J. Escudero, *Optimización de Energía y Eficiencia de Transmisión con Análisis de Imparcialidad en Comunicaciones Inalámbricas Adaptativas*, Tesis doctoral, Universidad Carlos III de Madrid, feb 2010.

[ETS01] *Broadband Radio Access Networks (BRAN); HIPERLAN type 2; physical(PHY) layer*, Tech. Report ETSI TS 101 475 V1.3.1, European Telecommunications Standards Institute, mar 2001.

[Fel94] D. Feldman y L. Griffiths, *A projection approach for robust adaptive beamforming*, Signal Processing, IEEE Transactions on **42** (1994), no. 4, 867 –876.

[Fou02] R. Fourer, D. M. Gay, y B. W. Kernighan, *AMPL: A Modeling Language for Mathematical Programming*, Duxbury Press, nov 2002.

[GAM11] GAMS Development Corporation, *An Introduction to GAMS*, 2011, Web, `http://www.gams.com/docs/intro.htm`.

[Gas05] M. Gastpar y M. Vetterli, *On the capacity of large gaussian relay networks*, Information Theory, IEEE Transactions on **51** (2005), no. 3, 765 – 779.

[Ger03] A. Gershman, Z.-Q. Luo, S. Shahbazpanahi, y S. Vorobyov, *Robust adaptive beamforming using worst-case performance optimization*, Signals, Systems and Computers, 2003. Conference Record of the Thirty-Seventh Asilomar Conference on, vol. 2, nov 2003, pp. 1353 – 1357 Vol.2.

[Gol98] J. Goldberg y H. Messer, *Inherent limitations in the localization of a coherently scattered source*, Signal Processing, IEEE Transactions on **46** (1998), no. 12, 3441 –3444.

[Gra06] M. Grant, S. Boyd, y Y. Ye, *Disciplined Convex Programming*, Nonconvex Optimization and its Applications, cap. 7, pp. 155–210, Springer, 2006.

[Gra11] M. Grant y S. Boyd, *CVX: Matlab Software for Disciplined Convex Programming, version 1.21*, feb 2011, disponible online, `http://cvxr.com/cvx`.

[Gu08] H. Gu y C. Yang, *Power and Admission Control for UWB Cognitive Radio Networks*, Communications, 2008. ICC '08. IEEE International Conference on, may 2008, pp. 4933 –4937.

[Han05] S. H. Han y J. H. Lee, *An overview of peak-to-average power ratio reduction techniques for multicarrier transmission*, Wireless Communications, IEEE **12** (2005), no. 2, 56 – 65.

[Her04] J. M. Hernando, *Comunicaciones Móviles*, 2 ed., Centro de Estudios Ramón Areces, 2004.

[Hes52] M. R. Hestenes y E. Stiefel, *Methods of Conjugate Gradients for Solving Linear Systems*, Journal of Research of the National Bureau of Standards **49** (1952), 409–436.

[Kar08] E. Karipidis, N. Sidiropoulos, y Z.-Q. Luo, *Quality of Service and Max-Min Fair Transmit Beamforming to Multiple Cochannel Multicast Groups*, Signal Processing, IEEE Transactions on **56** (2008), no. 3, 1268 –1279.

[Kim08] S.-J. Kim, A. Magnani, A. Mutapcic, S. Boyd, y Z.-Q. Luo, *Robust Beamforming via Worst-Case SINR Maximization*, Signal Processing, IEEE Transactions on **56** (2008), no. 4, 1539 –1547.

[Li06] J. Li y P. Stoica, *Robust adaptive beamforming*, Wiley series in telecommunications and signal processing, John Wiley, 2006.

[Lob98] M. S. Lobo, L. Vandenberghe, S. Boyd, y H. Lebret, *Applications of second-order cone programming*, Linear Algebra and Its Applications **284** (1998), 193–228.

[Lor05] R. Lorenz y S. Boyd, *Robust minimum variance beamforming*, Signal Processing, IEEE Transactions on **53** (2005), no. 5, 1684 – 1696.

[Mak11] A. Makhorin, *GLPK (GNU Linear Programming Kit)*, Department for Applied Informatics, Moscow Aviation Institute, 2011, Web,
`http://www.gnu.org/software/glpk/`.

[Mat10] The MathWorks, Inc., *Matlab Optimization Toolbox, User Guide*, 2010, Web,
`http://www.mathworks.es/access/helpdesk/help/toolbox/optim/index.html`.

[Med98] J. Medbo y P. Schramm, *Channel Models for HIPERLAN in Different Indoor Scenarios*, Tech. Report ETSI EP BRAN 3ERI085B, European Telecommunications Standards Institute, mar 1998.

[Mo09] Y. Mo, L. Shi, R. Ambrosino, y B. Sinopoli, *Network lifetime maximization via sensor selection*, Asian Control Conference, 2009. ASCC 2009. 7th, ago 2009, pp. 441 –446.

[Mos11] MOSEK ApS, *The MOSEK optimization toolbox for MATLAB manual. Version 6.0 (Revision 114).*, 2011, Web,
`http://www.mosek.com/fileadmin/products/6_0/tools/doc/html/toolbox/index.html`.

[Mot07] *Long Term Evolution (LTE): A Technical Overview*, Tech. report, Motorola, Inc., 2007, Disponible online,
`http://www.motorola.com/web/Business/Solutions/IndustrySolutions/ServiceProviders/WirelessOperators/LTE/_Document/StaticFiles/6834_MotDoc_New.pdf`.

[Més96] C. Mészáros, *The Efficient Implementation of Interior Point Methods for Linear Programming and their Applications*, Tesis doctoral, Eötvös Loránd University of Sciences. PhD School of Operations Research, Applied Mathematics and Statistics, 1996.

[Més98] C. Mészáros, *The BPMPD interior point solver for convex quadratic problems*, Tech. Report WP-98-8, Laboratory of Operations Research and Decision Systems, Budapest, Hungría, 1998, `http://www.sztaki.hu/~meszaros/bpmpd/`.

[Mur83a] B. A. Murtagh y M. A. Saunders, *MINOS 5.5 User Guide*, Tech. Report SOL 83-20R, Systems Optimization Laboratory, Dep. of Operations Research, Stanford University, dic 1983, `http://www.sbsi-sol-optimize.com/manuals/Minos%20Manual.pdf`.

[Mur83b] K. G. Murty, *Introduction to Algorithms*, John Wiley & Sons Inc., New York, 1983.

[Nes95] Y. Nesterov y M. J. Todd, *Primal-Dual Interior-Point Methods for Self-Scaled Cones*, SIAM Journal on Optimization **8** (1995), 324–364.

[Ngo10] H. Q. Ngo, T. Quek, y H. Shin, *Amplify-and-Forward Two-Way Relay Networks: Error Exponents and Resource Allocation*, Communications, IEEE Transactions on **58** (2010), no. 9, 2653 –2666.

[Ntr09] V. Ntranos, N. Sidiropoulos, y L. Tassiulas, *On multicast beamforming for minimum outage*, Wireless Communications, IEEE Transactions on **8** (2009), no. 6, 3172 –3181.

[Oh05] J. Oh, S.-J. Kim, y K.-L. Hsiung, *A computationally efficient method for robust minimum variance beamforming*, Vehicular Technology Conference, 2005. VTC 2005-Spring. 2005 IEEE 61st, vol. 2, may 2005, pp. 1162 – 1165 Vol. 2.

[Pal03a] D. P. Palomar, *A Unified Framework for Communications through MIMO Channels*, Tesis doctoral, Universidad Politécnica de Cataluña, may 2003.

[Pal03b] D. P. Palomar, J. M. Cioffi, y M. A. Lagunas, *Joint Tx-Rx beamforming design for multicarrier MIMO channels: a unified framework for convex optimization*, Signal Processing, IEEE Transactions on **51** (2003), no. 9, 2381 – 2401.

[Pal05] D. P. Palomar, A. Pascual-Iserte, J. M. Cioffi, y M. A. Lagunas, *Convex Optimization Theory Applied to Joint Transmitter-Receiver Design in MIMO Channels*, cap. 8, pp. 269 – 318, John Wiley & Sons, abr 2005.

[Pha09] K. Phan, T. Le-Ngoc, S. Vorobyov, y C. Tellambura, *Power allocation in wireless multi-user relay networks*, Wireless Communications, IEEE Transactions on **8** (2009), no. 5, 2535 –2545.

[Pin10] J. D. Pintér, *MathOptimizer Professional User Guide.*, 2010, Web, `http://www.pinterconsulting.com/`.

[Pol72] E. Polak, *A survey of methods of feasible directions for the solution of optimal control problems*, Automatic Control, IEEE Transactions on **17** (1972), no. 5, 591 – 596.

[Sha03] S. Shahbazpanahi, A. Gershman, Z.-Q. Luo, y K. M. Wong, *Robust adaptive beamforming for general-rank signal models*, Signal Processing, IEEE Transactions on **51** (2003), no. 9, 2257 – 2269.

[Shi01] W. Shi, T. Sun, y R. Wesel, *Quasi-convexity and optimal binary fusion for distributed detection with identical sensors in generalized gaussian noise*, Information Theory, IEEE Transactions on **47** (2001), no. 1, 446 –450.

[Stu98] J. F. Sturm, *Using SeDuMi 1.02, a MATLAB toolbox for optimization over symmetric cones*, Advances Optimization Lab, McMaster University, 1998, incluido en la descarga del *software*.

[Toh06] K. C. Toh, C. R. H. Tütüncü, y M. J. Todd, *On the Implementation and Usage of SDPT3 - a Matlab Software Package for Semidefinite-Quadratic-Linear Programming, version 4.0*, jul 2006.

[Tre97] L. N. Trefethen y D. Bau, *Numerical Linear Algebra*, SIAM: Society for Industrial and Applied Mathematics, jun 1997.

[UC311] Universidad Carlos III de Madrid, *Plantilla presupuesto del proyecto fin de carrera*, 2011, Web, `http://www.uc3m.es/portal/page/portal/administracion_campus_leganes_est_cg/proyecto_fin_carrera/Formulario_PresupuestoPFC-TFG(3)_1.xlsx`.

[Vor03] S. A. Vorobyov, A. B. Gershman, y Z.-Q. Luo, *Robust adaptive beamforming using worst-case performance optimization: a solution to the signal mismatch problem*, Signal Processing, IEEE Transactions on **51** (2003), no. 2, 313 – 324.

[Vor04] S. Vorobyov, A. Gershman, Z.-Q. Luo, y N. Ma, *Adaptive beamforming with joint robustness against mismatched signal steering vector and interference nonstationarity*, Signal Processing Letters, IEEE **11** (2004), no. 2, 108 – 111.

[Vor08] S. Vorobyov, H. Chen, y A. Gershman, *On the Relationship Between Robust Minimum Variance Beamformers With Probabilistic and Worst-Case Distortionless Response Constraints*, Signal Processing, IEEE Transactions on **56** (2008), no. 11, 5719 –5724.

[Wol10] Wolfram Research, Inc., *Mathematics and Algorithms: Optimization*, 2010, Web, `http://reference.wolfram.com/mathematica/guide/Optimization.html`.

[Wri97] S. J. Wright, *Primal-Dual Interior-Point Methods*, SIAM: Society for Industrial and Applied Mathematics, jun 1997.

[Wu96] S.-P. Wu, S. Boyd, y L. Vandenberghe, *FIR filter design via semidefinite programming and spectral factorization*, Decision and Control, 1996., Proceedings of the 35th IEEE, vol. 1, dec 1996, pp. 271 –276 vol.1.

[Ye94] Y. Ye, M. J. Todd, y S. Mizuno, *An $O(\sqrt{nL})$-iteration homogeneous and self-dual linear programming algorithm*, Mathematics of Operations Research **19** (1994), 53–67.

[Zha05] F. Zhang, *The Schur Complement and Its Applications (Numerical Methods and Algorithms)*, Springer, mar 2005.

Acrónimos

AMPL *A Mathematical Programming Language* (un lenguaje de programación matemática) 83

ANSI *American National Standards Institute* (instituto nacional americano de estándares) 83

API *Application Programming Interface* (interfaz de programación de aplicaciones) 83

ATLAS *Automatically Tuned Linear Algebra Software* (programa de álgebra lineal ajustado automáticamente) 92

BER *Bit Error Rate* (tasa de bits erróneos) 5

BLAS *Basic Linear Algebra Subprograms* (subprogramas de álgebra lineal básica) 81

BLAST *Bell Laboratories Layered Space-Time* (espacio-tiempo en capas de los laboratorios Bell) 172

CD *Compact Disc* (disco compacto) 106

CNS *Constrained Non-Linear System* (sistema no lineal con restricciones) 99

COIT Colegio oficial de ingenieros de telecomunicación 220

CP *Conic Program* (programa cónico) 43

CPU *Central Processing Unit* (unidad central de procesamiento) 81

CSIR *Channel State Information at the Receiver* (información de estado del canal en el receptor) 171

CSIT *Channel State Information at the Transmitter* (información de estado del canal en el transmisor) 171

DIMACS *Center for Discrete Mathematics and Theoretical Computer Science* (centro para matemáticas discretas y ciencia informática teórica) 94

DL *Diagonal Loading* (carga diagonal) 125

DLL *Dynamic Link Library* (Biblioteca de enlace dinámico) . . 84

DSL *Digital Subscriber Line* (línea digital de abonado) 168

FEC *Forward Error Correction* (corrección de errores hacia delante) 172

FIR *Finite Impulse Response* (respuesta al impulso infinita) . 37

GAMS *General Algebraic Modeling System* (sistema de modelado algebraico general) 87

GLPK *GNU Linear Programming Kit* (Equipo de programación lineal de GNU) 82

GNU *GNU's Not Unix* (GNU no es Unix) 5

GP *Geometric Program* (programa geométrico)

GPL *GNU Public License / General Public License* (licencia pública de GNU / general) 5

HIPERLAN *High Performance Local Area Network* (red de área local de alto rendimiento) 195

HKM *Helmberg-Rendl-Vanderbei-Wolkowick (1996), Kojima-Shindoh-Hara (1997), Monteiro (1995)* (referencia a 3 publicaciones) 93

IDE *Integrated Development Environment* (entorno integrado de desarrollo) 83

INR *Interference to Noise Ratio* (relación interferencia-ruido) 144

IVA Impuesto sobre el Valor Añadido 221

KKT *Karush-Kuhn-Tucker* 11

LAPACK *Linear Algebra Package* (paquete de álgebra lineal) 81

LGO *Lipschitz Global Optimizer* (optimizador global de Lipschitz)

LGPL *Lesser General Public License* (licencia pública general en menor grado)

LMI *Linear Matrix Inequality* (desigualdad lineal matricial) . . 37

LMMSE *Linear Minimum Mean Square Error* (filtro lineal de mínimo error cuadrático medio) 176

LSMI *Loaded Sample Matrix Inversion* (inversión de matriz muestral cargada) 125

LP *Linear Program* (programa lineal) 40

LST *Layered Space-Time* (espacio-tiempo por capas) 172

MCP *Mixed Complementarity Problem* (problema complementario mezclado) 99

MILP *Mixed Integer Linear Program* (programa lineal con mezcla de enteros) 74

MIMO *Multiple Input Multiple Output* (entrada múltiple, salida múltiple) 1

MIP *Mixed Integer Program* (programa con mezcla de enteros) 74

MIQCP *Mixed Integer Quadratically Constrained Program* (programa con mezcla de enteros con restricciones cuadráticas) 99

MIQP *Mixed Integer Quadratic Program* (programa cuadrático con mezcla de enteros)

NLP *Non-Linear Program* (programa no lineal) 99

MINLP *Mixed Integer Non-Linear Program* (programa no lineal con mezcla de enteros)

MISO *Multiple Input, Single Output* (entrada múltiple, salida única) 170

ML *Maximum Likelihood* (máxima verosimilitud) 173

MPEC *Mathematical Program with Equilibrium Constraints* (programa matemático con restricciones de equilibrio) . . 99

MSE *Mean Square Error* (error cuadrático medio) 175

MTA SZTAKI *Magyar Tudományos Akadémia Számítástechnikai és Automatizálási Kutatóintézete* (Instituto de Investigación de Informática y Automática, Academia Húngara de Ciencias) 84

NT *Nesterov-Todd* 93

OFDM *Orthogonal Frequency Division Multiplexing* (multiplexado por división ortogonal en frecuencia) 6

OOQP *Object Oriented Quadratic Program* (programa cuadrático orientado a objetos) 85

PAPR *Peak to Average Power Ratio* (relación de potencia de pico a media) 167

PAR *Peak to Average Ratio* (relación de pico a media) 167

QAM *Quadrature Amplitude Modulation* (modulación de amplitud en cuadratura) 193

QCQP *Quadratically Constrained Quadratic Program* (programa cuadrático con restricciones cuadráticas) 43

QP *Quadratic Program* (programa cuadrático) 42

SDP *SemiDefinite Program* (programa semidefinido) 44

SDPA *SemiDefinite Programming Algorithm* (algoritmo de programación semidefinida) 94

SeDuMi *Self Dual Minimization* (Minimización autodual) 5

SIMO *Single Input Multiple Output* (entrada única, salida múltiple) 170

SINR *Signal to Interference and Noise Ratio* (relación señal a ruido e interferencia) 124

SMI *Sample Matrix Inversion* (inversión de matriz muestral)125

SMI-MV *Sample Matrix Inversion based Minimum Variance* (varianza mínima basada en inversión de matriz muestral) 125

SNR *Signal to Noise Ratio* (relación señal a ruido) 146

SOCP *Second Order Cone Program* (programa cónico de segundo orden) 43

STBC *Space-Time Block Coding* (codificación espacio-tiempo en bloque) 172

STC *Space-Time Coding* (codificación espacio-tiempo) 172

STTC *Space-Time Trellis Coding* (codificación espacio-tiempo en rejilla) 172

TDD *Time Division Duplex* (duplexado por división en el tiempo) 171

UWB *Ultra Wide Band* (banda ultra ancha) 37

WLAN *Wireless Local Area Network* (red de área local inalámbrica) 195