ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE CIENCIAS

ANÁLISIS Y RESOLUCIÓN NUMÉRICA DE PROBLEMAS DE PROGRAMACIÓN MATEMÁTICA CON RESTRICCIONES DE EQUILIBRIO Y COMPLEMENTARIEDAD

RESOLUCIÓN DE MPCCS CLÁSICOS CON MÉTODOS DE TIPO SQP (*SEQUENTIAL QUADRATIC PROGRAMMING*)

TRABAJO DE INTEGRACIÓN CURRICULAR PRESENTADO COMO REQUISITO PARA LA OBTENCIÓN DEL TÍTULO DE MATEMÁTICA

GABRIELA RENATA ARMENDÁRIZ GAVILANES

armendariz.gabriela@epn.edu.ec

DIRECTOR: SERGIO ALEJANDRO GONZÁLEZ ANDRADE

sergio.gonzalez@epn.edu.ec

DMQ, 02 2023

CERTIFICACIONES

Yo, GABRIELA RENATA ARMENDÁRIZ GAVILANES, declaro que el trabajo de integración curricular aquí descrito es de mi autoría; que no ha sido previamente presentado para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.

Gabriela Renata Armendáriz Gavilanes

Certifico que el presente trabajo de integración curricular fue desarrollado por Gabriela Renata Armendáriz Gavilanes, bajo mi supervisión.

Sergio Alejandro González Andrade
DIRECTOR

DECLARACIÓN DE AUTORÍA

A través de la presente declaración, afirmamos que el trabajo de integración curricular aquí descrito, así como el(los) producto(s) resultante(s) del mismo, es(son) público(s) y estará(n) a disposición de la comunidad a través del repositorio institucional de la Escuela Politécnica Nacional; sin embargo, la titularidad de los derechos patrimoniales nos corresponde a los autores que hemos contribuido en el desarrollo del presente trabajo; observando para el efecto las disposiciones establecidas por el órgano competente en propiedad intelectual, la normativa interna y demás normas.

Gabriela Renata Armendáriz Gavilanes

Sergio Alejandro González Andrade

RESUMEN

El presente trabajo contiene un estudio teórico de la técnica de *lifting* para la formulación de problemas de programación matemática con restricciones de complementariedad (MPCCs). También, se describe el sustento matemático requerido en la construcción de un algoritmo SQP no suave (nonsmooth SQP) para la resolución numérica de estos problemas. Este algoritmo utiliza la técnica antes mencionada con un especial enfoque en los B-subdiferenciales y las propiedades que permiten una correcta elección de la matriz definida positiva necesaria para garantizar su convergencia.

Se describe el funcionamiento del método de SQP semi-suave y cómo éste aprovecha las ventajas de aplicar el *lifting* al MPCC para construir subproblemas adecuados en cada iteración del algoritmo planteado.

Finalmente, se presenta una propuesta de implementación de este algoritmo en el lenguaje de programación Python y se presentan ejemplos numéricos.

Palabras clave: MPCC, lifting, SQP semi-suave, optimización no lineal, Python.

ABSTRACT

This document contains a theoretical study of the lifting technique for the formulation of mathematical problems with complementary constraints (MPCCs). Also, it describes the mathematical underpinning required in the construction of a nonsmooth SQP algorithm used for a numerical resolution of these problems. This algorithm uses the above mentioned technique with a special focus on B-subdifferentials and the properties that allow a correct choice of the positive definite matrix necessary to guarantee the convergence of the algorithm.

The operation of the semismooth SQP method is described and how it takes advantage of applying lifting to MPCC to construct suitable subproblems at each iteration of the proposed algorithm.

Finally, a proposed implementation of this algorithm in the Python programming language is presented and some numeric examples are given.

Keywords: MPCC, lifting, semismooth SQP, non-linear optimization, Python.

Índice general

1. Introducción **1**

1.1. Objetivo general 1

1.2. Objetivos específicos 1

1.3. Alcance 2

1.4. Marco Teórico 2

1.4.1. Definiciones Relevantes 2

1.4.2. MPCCs 4

1.4.3. SQP 4

2. Metodología **10**

2.1. Técnica de *'Lifting'* para MPCCs 10

2.2. Sistema de Optimalidad con Subdiferenciales 18

2.3. Semismooth SQP aplicado a un *lifted* MPCC 22

2.3.1. Convergencia local del método 24

2.3.2. Globalización de la convergencia del método 27

3. Resultados Computacionales **30**

3.1. Implementación 30

3.2. Resultados Numéricos 36

3.2.1. Ejemplo 1. 36

3.2.2. Ejemplo 2. 39

4. Conclusiones y recomendaciones **43**

4.1. Conclusiones 43

4.2. Recomendaciones 44

Bibliografía **45**

Índice de figuras

2.1. Conjunto L. 11
2.2. Conjunto S. 11

3.1. Decrecimiento de σ ejemplo 1. 39
3.2. Conjunto factible ejemplo 2. 40
3.3. Decrecimiento de σ ejemplo 2. 41

Capítulo 1

Introducción

1.1. Objetivo general

El objetivo central de este trabajo es estudiar la formulación de problemas de programación matemática con restricciones de complementariedad o MPCCs (*Mathematical Programs with Complementary Constraints*), y su solución numérica, a través de algoritmos de tipo SQP semi-suaves (*Semismooth Sequential Quadractic Programing*).

1.2. Objetivos específicos

1. OE1: Analizar la formulación por elevación (*lifting approach*) para los MPCCs, propuesta en [8].
2. OE2: Estudiar el algoritmo Semismooth SQP para los MPCCs, formulados con la técnica de lifting, propuesto en [3].
3. OE3: Implementar el algoritmo anterior en un ambiente Python y analizar sus propiedades.

1.3. Alcance

El alcance de este trabajo es generar un código en el lenguaje computacional Python para la implementación eficiente de un algoritmo SQP semi-suave para la resolución de MPCCs. Para esto, debemos estudiar y comprender en detalle la formulación de tipo *lifting* para los MPCCs, propuesta en [8]. Posteriormente, requerimos analizar en detalle el algoritmo de tipo SQP semi-suave para MPCCs propuesto en [3]. En particular, entender el uso de subdiferenciales para generar información de segundo orden, necesaria para la estructura SQP.

1.4. Marco Teórico

1.4.1. Definiciones Relevantes

Sea $A \subseteq \mathbb{R}^m$, denotamos la proyección ortogonal de A sobre $\mathbb{R}^\ell$ como $\pi_\ell(A)$, donde $\mathbb{R}^m = \mathbb{R}^\ell \times \mathbb{R}^{m-\ell}$

En [6] se define la *envolvente convexa* de $A \subset \mathbb{R}^n$ como la intersección de todos los conjuntos convexos que contienen a A.

Una función $\Phi : \mathbb{R}^n \to \mathbb{R}^m$ se dice Lipschitz continua alrededor de $u \in \mathbb{R}^n$ si existen una vecindad abierta V_u de u y $L > 0$ tales que

$$||\Phi(x) - \Phi(y)|| \leq L||x - y||, \quad \forall x, y \in V_u$$

Sea $\Phi : \mathbb{R}^n \to \mathbb{R}^m$ una función Lipschitz continua alrededor de $u \in \mathbb{R}^n$, el $B-$diferencial de Φ en el punto u está dado por

$$\partial_B \Phi(u) := \left\{ \Lambda \in \mathbb{R}^{m \times n} : \exists (u_k)_{k \in \mathbb{N}} \text{ tal que } u_k \in \mathcal{D}_\Phi \; \forall k \in \mathbb{N}, u_k \to u \text{ y } \Phi'(u_k) \to \Lambda \right\},$$

donde $\mathcal{D}_\Phi$ es el conjunto en el cual Φ es diferenciable.

El Jacobiano generalizado de Clarke de Φ en el punto u está dado por

$$\partial \Phi(u) = \operatorname{conv} \partial_B \Phi(u),$$

donde $\operatorname{conv} \partial_B \Phi(u)$ representa la envolvente convexa de $\partial_B \Phi(u)$.

Para una aplicación $\Phi : \mathbb{R}^p \times \mathbb{R}^q \to \mathbb{R}^m$, Lipschitz continuo alrededor de un punto $(u, v) \in \mathbb{R}^p \times \mathbb{R}^q$ se define el $B-$diferencial parcial de Φ en (u, v) respecto a u como el $B-$diferencial de la aplicación $\Phi(\cdot, v)$ y lo denotamos por $(\partial_B)_u \Phi(u, v)$. También, el Jacobiano generalizado de Clark parcial de Φ en (u, v) respecto a u es el Jacobiano generalizado de Clark del mapa $\Phi(\cdot, v)$, y se denota por $\partial_u \Phi(u, v)$.

Decimos que $\Phi : \mathbb{R}^n \to \mathbb{R}^m$ es *semismooth* en $u \in \mathbb{R}^n$ si es localmente Lipschitz continuo alrededor de u, posee derivada direccional en u en toda dirección, y satisface la condición

$$\sup_{\Lambda \in \partial\Phi(u+v)} ||\Phi(u + v) - \Phi(u) - \Lambda v|| = o(||v||).$$

En cambio, si se tiene que

$$\sup_{\Lambda \in \partial\Phi(u+v)} ||\Phi(u + v) - \Phi(u) - \Lambda v|| = O(||v||),$$

entonces decimos que Φ es *fuertemente semismooth* en u.

De acuerdo a [9], una formulación general de optimización no lineal con restricciones se define como

$$\min_{x \in \mathbb{R}^n} f(x) \tag{1.1}$$

$$s.a.$$

$$c_i(x) = 0, \;\; i = 1, \ldots, m_e \tag{1.2}$$

$$c_i(x) \geq 0, \;\; i = m_e + 1, \ldots, n, \tag{1.3}$$

donde la función objetivo y las funciones de restricción $f, c_i : \mathbb{R}^n \to \mathbb{R}$, con $i \in \{1, 2, \ldots, m\}$ son suaves, al menos una de ellas es no lineal; y m, m_e son enteros no negativos tales que $0 \leq m_e \leq m$.

Un punto $x \in \mathbb{R}^m$ se dice *factible* si y sólo si verifica (1.2) - (1.3) . Al conjunto de puntos factibles

$$\Omega := \{x \in \mathbb{R}^n : c_i(x) = 0 \; \forall i \in \{1, \ldots, m_e\}, \; c_j(x) \geq 0 \; \forall j \in \{m_e + 1, \ldots, m\}\}$$

se le denomina *conjunto factible*.

Para cualquier $x \in \mathbb{R}^n$, al conjunto

$$\mathcal{A}(x) = \{i \in \{1, 2, \ldots, m\} : c_i(x) = 0\}$$

le llamamos *conjunto de restricciones activas de x*.

Sea $x \in \mathbb{R}^n$ un punto factible de (1.1) - (1.3), decimos que la condición de calificación *LICQ* se cumple en x si los gradientes de las restricciones activas, $\nabla c_i(x)$, $i \in \mathcal{A}(x)$, son linealmente independientes ([5]).

1.4.2. MPCCs

Sean $f : \mathbb{R}^n \to \mathbb{R}$, $F^1, F^2 : \mathbb{R}^n \to \mathbb{R}^m$ funciones al menos dos veces diferenciables con segundas derivadas localmente acotadas, un *Problema Matemático con Restricciones de Complementariedad* (MPCC por sus siglas en inglés) es un problema de optimización del tipo

$$\text{mín } f(x) \text{ s.a. } x \in M, \tag{1.4}$$

donde

$$M := \left\{x \in \mathbb{R}^n : F^1(x) \geq 0, F^2(x) \geq 0, F^1(x)^\top F^2(x) = 0\right\}. \tag{1.5}$$

El estudio de este tipo de problemas es de interés en el estudio de sistemas de equilibrio económico o mecánico ([7]). Sin embargo, desde el punto de vista de la optimización no lineal, las restricciones de complementariedad pueden ser problemáticas pues no existe una solución factible que satisfaga todas las desigualdades estrictamente.

Por ello, para resolver numéricamente este tipo de problemas se han estudiado varias técnicas que sean computacionalmente estables y aproximen de la mejor manera una solución factible. En este contexto, una modelización adecuada de las condiciones de complementariedad es indispensable para el tratamiento computacional de los MPCCs.

1.4.3. SQP

La programación cuadrática secuencial (SQP) es un método iterativo usado para resolver problemas de optimización no lineal con restriccio-

nes, en los cuales la función objetivo y las restricciones son de clase C^2, es decir, dos veces diferenciables con derivadas continuas.

La forma de trabajo de los métodos SQP es que resuelven varios subproblemas de optimización, provenientes del problema original, donde cada uno de estos optimiza un modelo cuadrático de la función objetivo sujeta a las restricciones linealizadas.

Cuando el problema no tiene restricciones, este método se convierte en el de Newton para encontrar un punto donde el gradiente de la función objetivo se hace cero. En cambio, si el problema solo tiene restricciones de igualdad, entonces el método equivale a aplicar el método de Newton a las condiciones de Karush-Kuhn-Tucker[1] del problema.

Construcción de los Subproblemas de Programación Cuadráticos (QP)

Definimos el problema de optimización no lineal (P) de la siguiente manera

$$\min_{x \in \mathbb{R}^n} f(x) \tag{1.6}$$

s.a.

$$h(x) = 0 \tag{1.7}$$

$$g(x) \leq 0 \tag{1.8}$$

Los subproblemas QP que se deben resolver en cada iteración deben reflejar las propiedades locales de (P) respecto a la iteración actual. Así, en principio se reemplaza:

- la función objetivo f por su aproximación cuadrática

$$f(x) \approx f(x_k) + \nabla f(x_k)(x - x_k) + \frac{1}{2}(x - x_k)^\top Hf(x_k)(x - x_k),$$

- Las resticciones g y h por sus aproximacones afines locales

$$g(x) \approx g(x_k) + \nabla g(x_k)(x - x_k),$$
$$h(x) \approx h(x_k) + \nabla h(x_k)(x - x_k),$$

[1] revisar el Teorema de Karush-Kuhn-Tucker en [9]

Si definimos

$$d(x) := x - x_k \quad , \quad H_k := Hf(x_k),$$

tenemos el siguiente problema QP:

$$\min_{d(x)\in\mathbb{R}^n} \nabla f(x_k)^\top d(x) + \frac{1}{2} d(x)^\top H_k d(x) \tag{1.9}$$

s.a.

$$h(x_k) + \nabla h(x_k)^\top d(x) = 0 \tag{1.10}$$

$$g(x_k) + \nabla g(x_k)^\top d(x) \leq 0 \tag{1.11}$$

Este QP se relaciona con un modelo local cuadrático del Lagrangiano $\mathcal{L}$ como la función objetivo, lo que lleva al siguiente problema QP

$$\min_{d(x)\in\mathbb{R}^n} \frac{\partial \mathcal{L}}{\partial x}(x_k, \lambda_k, \mu_k)^\top d(x) + \frac{1}{2} d(x)^\top H\mathcal{L}(x_k, \lambda_k, \mu_k) d(x) \tag{1.12}$$

s.a.

$$h(x_k) + \nabla h(x_k)^\top d(x) = 0 \tag{1.13}$$

$$g(x_k) + \nabla g(x_k)^\top d(x) \leq 0, \tag{1.14}$$

donde λ_k y μ_k son los multiplicadores de Lagrange asociados con este QP.

Considerar este subproblema QP se justifica por el hecho de que las condiciones necesarias de primer orden y las condiciones suficientes de segundo orden para problemas con restricciones implican que x_* es un mínimo local del problema

$$\min_{x\in\mathbb{R}^n} \mathcal{L}(z, \lambda_*, \mu_*) \tag{1.15}$$

s.a.

$$h(x) = 0 \tag{1.16}$$

$$g(x) \leq 0 \tag{1.17}$$

Convergencia Local

El análisis de convergencia local del método SQ se va a hacer bajo el supuesto que las restricciones de desigualdad activas del problema

(P) en el mínimo local x_* son conocidas, lo que es justificable pues el subproblema QP para la iteración x_k tiene las mismas resticciones activas en x_k, con x_k suficientemente cerca de x_*.

En consecuencia, restringimos el análisis a los problemas QP de la forma:

$$\text{mín } \nabla f(x_k)^\top d_x + \frac{1}{2} d_x^\top H_k d_x, \tag{1.18}$$

s.a.

$$\nabla h(x_k)^\top d_x + h(x_k) = 0. \tag{1.19}$$

Un buen valor x_0 para la solución se puede usar para obtener un valor inicial apropiado λ_0 para el multiplicador de Lagrange inicial. En efecto, [2] resalta que las condiciones necesarias de primer orden implican que

$$\lambda_* = -[\nabla h(x_*)^\top A \nabla h(x_*)]^{-1} \nabla h(x_*)^\top A \nabla f(x_*) \tag{1.20}$$

para cualquier A no singular y definida positiva en el espacio nulo de $\nabla h(x_*)^\top$. En particular, si A es la matriz identidad, entonces (1.20) define la solución de mínimos cuadrados de las condiciones de optimalidad necesarias de primer orden. Con esto,

$$\lambda_0 = -[\nabla h(x_0)^\top \nabla h(x_0)]^{-1} \nabla h(x_0)^\top \nabla f(x_0) \tag{1.21}$$

la cual se acerca a λ_* a medida que x_0 se acerca a x_*.

Si denotamos λ_{k+1} al multiplicador de Lagrange óptimo de (1.18)-(1.19), las condiciones KKT son las siguientes:

$$\begin{aligned} H_k d_x + \nabla h(x_k) \lambda_{k+1} &= -\nabla f(x_k), \\ \nabla h(x_k)^\top d_x &= -h(x_k). \end{aligned}$$

Luego, tomando

$$d_\lambda := \lambda_{k+1} - \lambda_k \tag{1.22}$$

estas condiciones se pueden reescribir como

$$H_k d_x + \nabla h(x_k)^\top d_\lambda = -\nabla_x \mathcal{L}(x_k, \lambda_k) \tag{1.23}$$

$$\nabla h(x_k) d_x = -h(x_k) \tag{1.24}$$

El método de Newton SQP

La convergencia local del método SQP se sigue de la aplicación del método de Newton al sistema no lineal dado por las condiciones KKT

$$\Psi(x, \lambda) = \begin{bmatrix} \nabla \mathcal{L}(x, \lambda) \\ h(x) \end{bmatrix} = 0$$

Las condiciones necesarias de primer orden y las suficientes de segundo orden implican que el Jacobiano

$$J(x_*, \lambda_*) = \begin{bmatrix} H\mathcal{L}(x_*, \lambda_*) & \nabla h(x_*) \\ \nabla h(x_*)^\top & 0 \end{bmatrix} \tag{1.25}$$

es no singular en la solución local (x_*, λ_*). Por lo tanto, la iteración de Newton

$$x_{k+1} = x_k + s_x,$$
$$\lambda_{k+1} = \lambda_k + s_\lambda,$$

donde $s = (s_x, s_\lambda)$ es la solución de

$$J(x_k, \lambda_k) s = -\Psi(x_k, \lambda_k), \tag{1.26}$$

converge cuadrádicamente, dados (x_0, λ_0) suficientemente cercanos a (x_*, λ_*).

La ecuación (1.26) corresponde a (1.23), (1.24) para $H_k = H\mathcal{L}(x_k, \lambda_k)$, $d_x = s_x$, y $d_\lambda = s_\lambda$.

En consecuencia, las iteraciones (x_{k+1}, λ_{k+1}) son las generadas por el algoritmo SQP, con lo cual hemos probado el siguiente teorema:

Teorema 1.4.1. *Sea x_0 el valor inicial de la solución de (P). Supongamos que el valor inicial de λ_0 está dado por (1.21). Supongamos además que*

cada iteración está dada por

$$x_{k+1} = x_k + d_k,$$
$$\lambda_{k+1} = \lambda_k + d_\lambda,$$

donde d_x *y* $\lambda_k + d_\lambda$ *son las soluciones y el multiplicador asociado al subproblema QP* (1.18), (1.19) *con* $H_k = H\mathcal{L}(x_k, \lambda_k)$.

Si $||x_0 - x_*||$ *es suficientemente pequeño, la sucesión generada por las iteraciones del método SQP está bien definida y converge cuadráticamente a la solución óptima* (x_*, λ_*)

Capítulo 2

Metodología

2.1. Técnica de *'Lifting'* para MPCCs

Con el objetivo de encontrar un método de solución numérica para MPCCs, en el presente trabajo se hace uso de la técnica de *Lifting*.

La técnica consiste, principalmente, en reescribir el conjunto factible M descrito en (1.5) haciendo uso de la proyección ortogonal de algún conjunto *suave* adecuado en $\mathbb{R}^3$.

Definimos el conjunto

$$L := \left\{(a,b) \in \mathbb{R}^2 : a \geq 0, b \geq 0, ab = 0\right\}, \tag{2.1}$$

con el cual podemos reescribir M de la siguiente forma

$$M = \left\{x \in \mathbb{R}^n : \left(F_j^1(x), F_j^2(x)\right) \in L, \ \ j \in \{1,2,\ldots,m\}\right\}. \tag{2.2}$$

El problema de esta descripción de M es que L no es un conjunto suave, de hecho, tiene la siguiente forma:

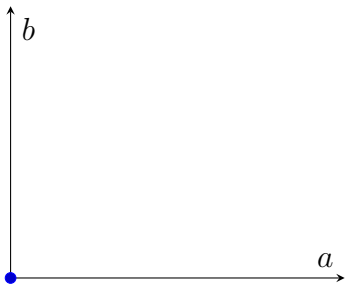

Figura 2.1: Conjunto L.

Gráficamente podemos ver que L presenta un problema en el origen de $\mathbb{R}^2$. Sin embargo, podemos ver a este conjunto como la proyección ortogonal de

$$S := \left\{(a,b,c) \in \mathbb{R}^3 : \psi(a,b,c) = 0\right\}, \tag{2.3}$$

donde $\psi : \mathbb{R}^3 \to \mathbb{R}^2$ se define componente a componente como

$$\psi_1(a,b,c) = a - (\text{máx}\{0,c\})^2 \quad \text{y} \quad \psi_2(a,b,c) = b - (\text{máx}\{0,-c\})^2 .$$

En [8] se presenta el siguiente gráfico de S:

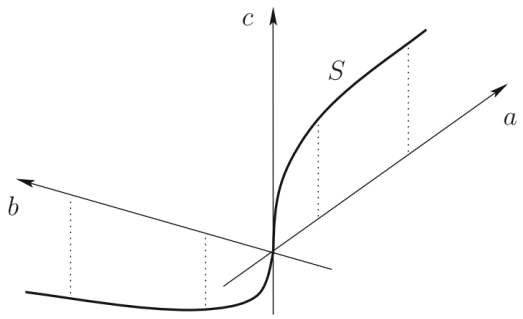

Figura 2.2: Conjunto S.

Así, tenemos el siguiente Lema

Lema 2.1.1. *Con las notaciones anteriores,* $\pi_2(S) = L$

Demostración. Para mostrar la igualdad entre ambos conjuntos, debemos verificar que se cumple

I. $\pi_2(S) \subseteq L$

II. $L \subseteq \pi_2(S)$

Probemos I.

Sea $(a, b) \in \pi_2(S)$, arbitrario pero fijo. Sabemos que existe $c \in \mathbb{R}$ tal que $(a, b, c) \in S$, con lo cual

$$a = (\text{máx}\{0, c\})^2 \quad \text{y} \quad b = (\text{máx}\{0, -c\})^2$$

Si $c \geq 0$, entonces $a = c^2$ y $b = 0$, por lo que $a \geq 0$, $b = 0$ y $ab = 0$ y de este modo, $(a, b) \in L$.

Por otro lado, si $c \leq 0$, entonces $a = 0$, $b = c^2 \geq 0$ y $ab = 0$, de donde $(a, b) \in L$.

Así, hemos mostrado que independientemente del c, $(a, b) \in L$, y dado que $(a, b) \in \pi_2(S)$ fue arbitrario, se ha probado I.

Ahora, mostremos II. Sea $(a, b) \in L$, cualquiera. Buscamos $c \in \mathbb{R}$ tal que $(a, b, c) \in S$ para probar que $(a, b) \in \pi_2(S)$.

Por la estructura de L, tenemos 3 posibilidades

1. $a = b = 0$,
2. $a > 0$, $b = 0$,
3. $a = 0$, $b > 0$

En el primer caso, basta tomar $c = 0$ para conseguir que $(a, b, c) \in S$.

En el segundo caso, ya que la función

$$\begin{aligned} s : \mathbb{R}^+ &\to \mathbb{R}^+ \\ x &\mapsto (\text{máx}\{0, x\})^2 \end{aligned}$$

es sobreyectiva $\left(p = s\left(\sqrt{p}\right), \forall p \in \mathbb{R}^+\right)$, entonces existe $c \in \mathbb{R}^+$ tal que $a = s(c) = (\text{máx}\{0, c\})^2$ y como $-c \leq 0$, entonces $b = 0 = (\text{máx}\{0, -c\})^2$, por lo que $(a, b, c) \in S$.

Análogamente, en el tercer caso existe $p \in \mathbb{R}^+$ tal que $b = s(p) = (\text{máx}\{0, p\})^2$ y $a = (\text{máx}\{0, -p\})^2$, por lo que basta tomar $c = -p \in \mathbb{R}$ para tener que $(a, b, c) \in S$.

En resumen, hemos mostrado que en los 3 casos posibles, existe $c \in \mathbb{R}$ de modo que $(a, b, c) \in S$, lo que nos permite concluir que $(a, b) \in \pi_2(S)$. Dado que $(a, b) \in L$ fue arbitrario, se tiene II., como se requería. □

Consideremos ahora el conjunto

$$M_P := \left\{(x, y) \in \mathbb{R}^n \times \mathbb{R}^m : \psi\left(F_j^1(x), F_j^2(x), y_j\right) = 0, \;\; j \in \{1, 2, \ldots, m\}\right\},$$

tenemos el siguiente Lema

Lema 2.1.2. $\pi_n\left(M_P\right) = M$

Demostración. Sea $x \in \pi_n\left(M_P\right)$, entonces por definición existe $y \in \mathbb{R}^m$ tal que $(x, y) \in M_P$.

De este modo, $\psi\left(F_j^1(x), F_j^2(x), y_j\right) = 0$ para cada $j \in \{1, 2, \ldots, m\}$, con lo cual por definición de S, $\left(F_j^1(x), F_j^2(x), y_j\right) \in S$.

Así, para cada $j \in \{1, 2, \ldots, m\}$ se tiene que $\left(F_j^1(x), F_j^2(x)\right) \in \pi_2(S)$ que junto con el Lema 2.1.1 nos da que $\left(F_j^1(x), F_j^2(x)\right) \in L$ para cada $j \in \{1, 2, \ldots, m\}$.

Finalmente, gracias a (2.2), concluimos que $x \in M$.

Recíprocamente, si $x \in M$ gracias a (2.2) y el Lema 2.1.1 se sigue que $\left(F_j^1(x), F_j^2(x)\right) \in \pi_2(S)$ para cada $j \in \{1, 2, \ldots, m\}$, con lo cual existe $y \in \mathbb{R}^m$ tal que $\left(F_j^1(x), F_j^2(x), y_j\right) \in S$.

Esto implica que $\psi\left(F_j^1(x), F_j^2(x), y_j\right) = 0$ para cada $j \in \{1, 2, \ldots, m\}$, de donde concluimos que $(x, y) \in M_p$ y por tanto $x \in \pi_n\left(M_p\right)$ □

Este Lema nos lleva a la siguiente proposición

Proposición 2.1.3. *Las soluciones de* (1.4) *son las componentes* $x \in \mathbb{R}^n$ *de las soluciones del problema*

$$P : \min_{(x,y) \in \mathbb{R}^n \times \mathbb{R}^m} f(x) \; \textbf{s.a} \; (x, y) \in M_P$$

Demostración. Se sigue directamente del Lema 2.1.2, pues si $(x, y) \in M_P$ minimiza f, entonces $x = \pi_n\left((x, y)\right) \in M$ minimiza f, pues f no depende de $y \in \mathbb{R}^m$. □

Del Lema 2.1.1, sabemos que para $(a,b) \in L$ podemos escoger

$$c > 0 \text{ si } a > 0, b = 0,$$
$$c = 0 \text{ si } a = 0, b = 0,$$
$$c < 0 \text{ si } a = 0, b > 0,$$

de manera que $(a,b,c) \in S$. Para determinar c, es necesario notar que la función s definida en la demostración del Lema 2.1.1 es inyectiva.

En efecto, veamos que en $\mathbb{R}^+$ se tiene que $(\text{máx}\{0,x\})^2 = x^2$, de donde seguimos que

$$s'(x) = 2x > 0 \quad \forall x \in \mathbb{R}^+ \setminus \{0\}$$

Así, s es estrictamente creciente, lo que indica que es inyectiva y, al ser también sobreyectiva, existe $s^{-1} : \mathbb{R}^+ \to \mathbb{R}^+$.

Con esto, revisando nuevamente la demostración del Lema 2.1.1, podemos definir

$$c(a,b) = \begin{cases} s^{-1}(a) & \text{si} \quad a > 0, b = 0 \\ -s^{-1}(b) & \text{si} \quad a = 0, b > 0 \\ 0 & \text{si} \quad a = 0, b = 0 \end{cases} \tag{2.4}$$

De este modo, $\pi_2 : S \to L$ es inyectiva pues para cada $(a,b) \in L$ existe un único $c = c(a,b) \in \mathbb{R}$ tal que $(a,b,c) \in S$ y es sobreyectiva por el Lema 2.1.1, y posee una inversa que además es continua (la función raíz cuadrada).

En resumen, π_2 es un homeomorfismo entre S y L. Ahora, utilizando la función c en (2.4), para cada $j \in K$ y cada x tal que $\left(F_j^1(x), F_j^2(x)\right) \in L$ podemos definir

$$y_j(x) = c\left(F_j^1(x), F_j^2(x)\right). \tag{2.5}$$

De esta manera $\left(F_j^1(x), F_j^2(x), y_j\right) \in S$. En particular, revisando a las equivalencias en la demostración del Lema 2.1.2, para cada $x \in M$ existe un único $y(x) \in \mathbb{R}^m$ tal que $(x, y(x)) \in M_P$.

Es importante notar que para cualquier $\overline{x} \in M$, al menos uno de los valores de $F_j^1(\overline{x})$, $F_j^2(\overline{x})$ se anula, esto es, la restricción $F_j^i(\overline{x}) \geq 0$ se vuelve activa para algún $i \in \{1,2\}$.

Sea

$$A(\overline{x}) := \big\{(i,j) \in \{1,2\} \times \{1,2\ldots,k\} : F_j^i(\overline{x}) = 0\big\}$$

el conjunto de índices activos en $\overline{x}$. Cuando $F_j^1(\overline{x}) = F_j^2(\overline{x}) = 0$, la restricción correspondiente $\big(F_j^1(\overline{x}), F_j^2(\overline{x})\big) \in L$ es llamada *biactiva*, y definimos

$$B(\overline{x}) := \{j \in \{1,2\ldots,k\} : (1,j),(2,j) \in A(\overline{x})\}$$

el conjunto de índices biactivos.

Para el siguiente resultado vamos a hacer uso de la condición MPEC-LICQ, que decimos que se cumple en el punto $\overline{x} \in M$ si los vectores $\nabla F_j^i(x)$, con $(i,j) \in A(\overline{x})$, son linealmente independientes.

Lema 2.1.4. *MPEC-LICQ se cumple en $\overline{x} \in M$ si y sólo si LICQ se cumple en $(\overline{x}, y(\overline{x}))$*

Demostración. Comencemos por derivar $\psi_1(F_j^1(x), F_j^2(x), y_j)$ respecto a x, aplicando la regla de la cadena para cada $j \in \{1,2,\ldots,m\}$, tenemos que

$$\partial_x \psi_1(F_j^1(x), F_j^2(x), y_j) = \nabla F_j^1(x).$$

Ahora, derivando $\psi_1(F_j^1(x), F_j^2(x), y_j)$ respecto a y_p, con $p \in \{1,2,\ldots,m\}$, para cada $j \in \{1,2,\ldots,m\}$ tenemos que

$$\partial_{y_p} \psi_1(F_j^1(x), F_j^2(x), y_j) = \begin{cases} 0 & \text{si } p \neq j \\ -s'(y_j) & \text{si } p = j \end{cases},$$

por lo cual para cada $j \in \{1,2,\ldots,m\}$,

$$\partial_y \psi_1(F_j^1(x), F_j^2(x), y_j) = \left(0 \ \cdots \ \underbrace{-s'(y_j)}_{\text{posición j-ésima}} \ \cdots \ 0\right).$$

Bajo un razonamiento análogo se tiene que

$$\partial_x \psi_2(F_j^1(x), F_j^2(x), y_j) = \nabla F_j^2(x);$$

$$\partial_y \psi_2(F_j^1(x), F_j^2(x), y_j) = \left(0 \ \cdots \ \underbrace{s'(-y_j)}_{\text{posición j-ésima}} \ \cdots \ 0\right).$$

De este modo, si tomamos $\overline{x} \in M$ y $\overline{y} = y(\overline{x})$, el Jacobiano de las condiciones que describen a M_P tiene la forma

$$J = \begin{pmatrix} \nabla F^1(\overline{x})^\top & \nabla F^2(\overline{x})^\top \\ -S'(\overline{y}) & S'(-\overline{y}) \end{pmatrix},$$

donde ∇F^i es de la forma

$$\nabla F^i = \begin{pmatrix} \nabla F_1^i \\ \nabla F_2^i \\ \vdots \\ \nabla F_k^i \end{pmatrix},$$

con $i \in \{1, 2\}$. $S'(\overline{y})$ representa la matriz diagonal que tiene en la posición (j, j) la cantidad $s'(\overline{y_j})$ y 0 en las posiciones restantes; y $S'(-\overline{y})$ representa la matriz diagonal que tiene en la posición (j, j) la cantidad $s'(-\overline{y_j})$ y 0 en las posiciones restantes, para cada $j \in \{1, 2, \ldots, m\}$.

Sin pérdida de generalidad podemos asumir que las primeras ℓ restricciones complementarias son biactivas, por lo que, de (2.4), $\overline{y_j} = 0$ para todo $j \in \{1, 2, \ldots, \ell\}$. Así, $s'(\overline{y_j}) = s'(-\overline{y_j}) = 0$ para todo $j \in \{1, 2, \ldots, \ell\}$.

Para las restricciones restantes, dado que son complementarias, para cada $j \in \{\ell, \ell + 1, \ldots, k\}$ se tiene que $F_j^1(\overline{x})$ o $F_j^2(\overline{x})$ es activa. Vamos a asumir, sin pérdida de generalidad, que $(1, j) \in A(\overline{x})$, entonces de (2.4) y (2.5), $\overline{y_j} < 0$, por lo que $s'(\overline{y_j}) = 0$ y $s'(-\overline{y_j}) > 0$ para cada $j \in \{\ell, \ell + 1, \ldots, k\}$. Con esto, el jacobiano J se ve de la siguiente manera

$$J = \begin{pmatrix} \nabla F^1(\overline{x}) & \nabla F_1^2(\overline{x}) & \cdots & \nabla F_\ell^2(\overline{x}) & \nabla F_{\ell+1}^2(\overline{x}) & \cdots & F_k^2(\overline{x}) \\ 0 & 0 & \cdots & 0 & 0 & \cdots & 0 \\ & & \ddots & & & & \\ \vdots & & & 0 & & & \vdots \\ & & & & s'(-\overline{y_{\ell+1}}) & & \\ & & & & & \ddots & 0 \\ 0 & & \cdots & & & 0 & s'(-\overline{y_k}) \end{pmatrix}.$$

Sabemos que *LICQ* se cumple en $(\overline{x}, \overline{y})$ si y sólo si J tiene rango completo, lo cual se da si y sólo si $\nabla F^1(\overline{x}), \nabla F_1^2(\overline{x}), \nabla F_2^2(\overline{x}), \ldots, \nabla F_\ell^2(\overline{x})$ son linealmente independientes, es decir, que se cumpla *MPEC-LICQ* en $\overline{x}$. □

Gracias al Lema 2.1.4 sabemos que al usar la técnica de *lifting* no se quebranta el cumplimiento de *LICQ* en el conjunto factible.

A partir de aquí vamos a asumir que *MPEC-LICQ* se cumple en M. Con multiplicadores de Lagrange $\lambda^1, \lambda^2 \in \mathbb{R}^m$, el Lagrangiano de P, definido en la Proposición 2.1.3, es de la forma

$$\mathcal{L}_P(x, y, \lambda^1, \lambda^2) = f(x) - \sum_{j=1}^{m} \lambda_j^1 \psi_1(F_j^1(x), F_j^2(x), y_j) - \sum_{j=1}^{m} \lambda_j^2 \psi_2(F_j^1(x), F_j^2(x), y_j),$$

El Lagrangiano del *MPCC* descrito en (1.4) - (1.5) tiene la forma

$$\mathcal{L}_{MPCC}(x, \lambda) = f(x) - \sum_{j=1}^{m} \lambda_j^1 F_j^1(x) - \sum_{j=1}^{m} \lambda_j^2 F_j^2(x), .$$

con $\lambda = (\lambda^1, \lambda^2)$.

De la demostración del Lema anterior sabemos que el gradiente de $\mathcal{L}_P$ respecto a (x, y) se ve de la siguiente manera

$$\begin{aligned}\nabla_{(x,y)}\mathcal{L}_P(x, y, \lambda) &= \begin{pmatrix} \nabla f(x) \\ 0 \end{pmatrix} - \begin{pmatrix} \sum_{j=1}^m \lambda_j^1 \nabla F_j^1(x) \\ \sum_{j=1}^m \lambda_j^1 (-S'(y)) \end{pmatrix} - \begin{pmatrix} \sum_{j=1}^m \lambda_j^2 \nabla F_j^2(x) \\ \sum_{j=1}^m \lambda_j^2 (S'(-y)) \end{pmatrix} \\ &= \begin{pmatrix} \nabla f(x) \\ 0 \end{pmatrix} - \begin{pmatrix} \nabla F^1(x)^\top \\ -S'(y) \end{pmatrix} \lambda^1 - \begin{pmatrix} \nabla F^2(x)^\top \\ S'(-y) \end{pmatrix} \lambda^2.\end{aligned}$$

Así, las condiciones de optimalidad de primer orden para el problema P son las siguientes:

$$\nabla f(x) - \nabla F^1(x)^\top \lambda^1 - \nabla F^2(x)^\top \lambda^2 = 0, \tag{2.6}$$

$$S'(y)\lambda^1 - S'(-y)\lambda^2 = 0, \tag{2.7}$$

$$\psi(F_j^1(x), F_j^2(x), y_j) = 0 \quad \forall j \in \{1, 2, \ldots, m\}. \tag{2.8}$$

De (2.8) tenemos que

$$\begin{aligned} F_j^1(x) &= s(y_j), \\ F_j^2(x) &= s(-y_j), \end{aligned}$$

de donde vemos que si $F_j^1(x) > 0$ entonces $s'(y_j) > 0$, y si $F_j^2(x) > 0$ entonces

$s'(-y_j) > 0$, para cada $j \in \{1, 2, \ldots, m\}$, de modo que (2.7) es una condición de complementariedad.

Con esto, un punto $(x, y) \in \mathbb{R}^n \times \mathbb{R}^m$ es un punto crítico de P si y sólo si existen $\lambda_j^i \in \mathbb{R}$, $(i, j) \in A(x)$, tales que

$$\nabla f(x) - \sum_{(i,j)\in A(x)} \lambda_j^i \nabla F_j^i(x) = 0 \tag{2.9}$$

$$\left(F_j^1(x), F_j^2(x)\right) \in L, \quad \forall j \in \{1, 2, \ldots, m\}. \tag{2.10}$$

Estas dos últimas condiciones definen la *estacionariedad débil* de x para una *MPCC*. Si, adicionalmente, se tiene que

$$\lambda_j^i \geq 0,$$

para cada $i \in B(x)$ y cada $j \in \{1, 2, \ldots, m\}$, entonces x se dice *fuertemente estacionario* para la *MPCC* y λ se denomina *MPCC-multiplicador* asociado al punto fuertemente estacionario x del problema (1.4) - (1.5).

Con esto, se tienen las siguientes proposiciones:

Proposición 2.1.5. *Un punto $(x, y(x)) \in \mathbb{R}^n \times \mathbb{R}^m$ es un punto crítico de P si y sólo si x es débilmente estacionario para (1.4).*

Proposición 2.1.6. *Si una solución local x del problema (1.4) - (1.5) satisface MPEC-LICQ, entonces x es un punto fuertemente estacionario y su MPCC-multiplicador asociado λ es único.*

2.2. Sistema de Optimalidad con Subdiferenciales

Para implementar el método SQP en el caso de funciones dos veces diferenciables, dado el sistema lineal (1.23) - (1.24), es natural escoger H_k como $\nabla_{xx}\mathcal{L}_{MPCC}(x_k, \lambda_k)$, i.e., la matriz Hessiana del Lagrangiano del MPCC. Sin embargo, computacionalmente calcular estas matrices es muy costoso (revisar [1]) y esto puede ser un obstáculo para computadoras de bajo rendimiento.

Así, una segunda opción es tomar H_k de la forma

$$H_k \in \partial_x \left(\nabla_x \mathcal{L}_{MPCC}(x_k, \lambda_k) \right), \tag{2.11}$$

con ∂_x el Jacobiano generalizado de Clark parcial respecto a x y ∇_x el gradiente respecto a x. El proceso iterativo definido por (1.23) - (1.24), escogiendo H_k como en (2.11) se denomina *semismooth SQP*.

El método semismooth SQP puede ser interpretado como el método semismooth de Newton aplicado al sistema de optimalidad de Lagrange asociado a (1.6) - (1.7). Definimos $\Phi : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n \times \mathbb{R}^m$ por la fórmula

$$\Phi(u) = \left(\nabla_x \mathcal{L}_{MPCC}(x, \lambda), h(x) \right), \quad u = (x, \lambda), \tag{2.12}$$

consideramos la ecuación

$$\Phi(u) = 0. \tag{2.13}$$

Una iteración del método generalizado de Newton (semismooth) para (2.13) consiste en resolver la ecuación

$$\Phi(u_k) + \Lambda_k(u - u_k) = 0,$$

para el actual $u_k \in \mathbb{R}^n \times \mathbb{R}^m$ y alguna matriz $\Lambda_k \in \partial\Phi(u_k)$. [3] establece que para cualquier $u = (x, \lambda) \in \mathbb{R}^n \times \mathbb{R}^m$ se tiene que

$$\partial\Phi(u) = \left\{ \begin{pmatrix} H & \nabla h(x)^\top \\ \nabla h(x) & 0 \end{pmatrix} : H \in \partial_x \left(\nabla_x \mathcal{L}_{MPCC}(x, \lambda) \right) \right\}, \tag{2.14}$$

de modo que si la matriz H_k se escoge como en (2.11), la matriz del sistema de iteración en (1.23) - (1.24) satisface que

$$\begin{pmatrix} H_k & \nabla h(x_k)^\top \\ \nabla h(x_k) & 0 \end{pmatrix} \in \partial\Phi(u_k).$$

Con esto, la convergencia local del semismooth SQP se deriva de la convergencia el método generalizado de Newton. Específicamente, la convergencia local superlineal primal-dual del método SQP semismooth se obtiene bajo los siguientes supuestos:

1. Las derivadas de f y h son *semismooth* en un punto estacionario $\overline{x}$

de (1.6) - (1.7) (se cumple *LICQ* en $\overline{x}$). De esto se puede concluir que Φ definido en (2.12) es *semismooth* en la solución $\overline{u} = (\overline{x}, \overline{\lambda})$ de la ecuación (2.13), con $\overline{\lambda}$ el multiplicador de Lagrange asociado a $\overline{x}$.

2. Φ es $CD-$regular en $\overline{u}$, es decir,

$$\text{Todo } \Lambda \in \partial\Phi(\overline{u}) \text{ es no singular.} \tag{2.15}$$

De (2.14) se sigue que este último supuesto se cumple si LICQ se cumple en $\overline{x}$ y

$$\langle H\xi, \xi\rangle > 0 \quad \forall \xi \in \ker\left(\nabla h(\overline{x})\right) \setminus \{0\}, \quad \forall H \in \partial_x \left(\nabla_x \mathcal{L}_{MPCC}(\overline{x}, \overline{\lambda})\right). \tag{2.16}$$

Por las propiedades de la envolvente convexa, la condición (2.16) se puede reemplazar por la condición equivalente

$$\langle H\xi, \xi\rangle > 0 \quad \forall \xi \in \ker\left(\nabla h(\overline{x})\right) \setminus \{0\}, \quad \forall H \in (\partial_B)_x \left(\nabla_x \mathcal{L}_{MPCC}(\overline{x}, \overline{\lambda})\right). \tag{2.17}$$

De acuerdo a [3], la condición de $CD-$regularidad se puede relajar eligiendo una multifunción $\Delta : \mathbb{R}^n \times \mathbb{R}^m \to \mathcal{P}(\mathbb{R}^n \times \mathbb{R}^n)$ tal que

$$\Delta(x, \lambda) \subset \partial_x \left(\nabla_x \mathcal{L}_{MPCC}(x, \lambda)\right) \quad \forall x \in \mathbb{R}^n, \ \forall \lambda \in \mathbb{R}^m, \tag{2.18}$$

con $\mathcal{P}(\mathbb{R}^n \times \mathbb{R}^n)$ el conjunto *partes de* $\mathbb{R}^n \times \mathbb{R}^n$. En el proceso iterativo descrito en (1.23) - (1.24) tomamos para cada $k \in \mathbb{N}$,

$$H_k \in \Delta(x_k, \lambda_k) \tag{2.19}$$

La multifunción Δ es de utilidad para elegir la matriz H_k cuando hay más de una opción, de modo que el proceso converja adecuadamente. Definimos

$$\overline{\Delta} = \overline{\Delta}(\overline{x}, \overline{\lambda}) = \left\{ H \in \mathbb{R}^{n\times n} : \begin{array}{l} \exists \left((x_k, \lambda_k)\right)_{k\in\mathbb{K}}, \ (x_k, \lambda_k) \in \mathbb{R}^n \times \mathbb{R}^l \ \forall k \in \mathbb{N}, \\ \exists (H_k)_{k\in\mathbb{N}}, H_k \in \Delta(x_k, \lambda_k), \ \forall k \in \mathbb{N}, \\ \text{tales que } (x_k, \lambda_k) \to (\overline{x}, \overline{\lambda}) \text{ y } H_k \to H \end{array} \right\} \tag{2.20}$$

De [3] se sabe que

$$\overline{\Delta} \subset \partial_x \left(\nabla_x \mathcal{L}_{MPCC}(\overline{x}, \overline{\lambda})\right). \tag{2.21}$$

Con esto, para el análisis de convergencia local podemos reemplazar (2.15) por la condición

$$\begin{pmatrix} H & \nabla h(\overline{x})^\top \\ \nabla h(\overline{x}) & 0 \end{pmatrix} \text{ es no singular } \forall H \in \overline{\Delta}. \tag{2.22}$$

Notemos que gracias a (2.21), (2.22) y la caracterización de $\partial\Phi$ en (2.14), esta condición es más débil que la $CD-$regularidad.

Una posibilidad natural es escoger

$$\Delta(x,\lambda) = (\partial_B)_x \left(\nabla_x \mathcal{L}_{MPCC}(x,\lambda)\right), \quad x \in \mathbb{R}^n, \ \lambda \in \mathbb{R}^m. \tag{2.23}$$

Tenemos el siguiente resultados concerniente a la convergencia bajo los supuestos mencionados anteriormente:

Teorema 2.2.1. *Si las derivadas de f y h son semismooth en un punto estacionario $\overline{x}$ de (1.6) - (1.7) y $\overline{\lambda}$ es el multiplicador de Lagrange asociado a $\overline{x}$, LICQ se cumple en $\overline{x}$, y*

$$\langle H\xi, \xi\rangle > 0 \quad \xi \in \textit{ker}\,(\nabla h(\overline{x})) \setminus \{0\}, \ \forall H \in \overline{\Delta}, \tag{2.24}$$

con $\overline{\Delta}$ definido en (2.20) para alguna multifunción $\Delta : \mathbb{R}^n \times \mathbb{R}^m \to \mathcal{P}(\mathbb{R}^n \times \mathbb{R}^n)$ que cumple (2.18).

Entonces para cualquier forma de elección de H_k de modo que (2.18) se satisface, cualquier punto inicial $(x_0, \lambda_0) \in \mathbb{R}^n \times \mathbb{R}^m$ lo suficientemente cerca de $(\overline{x}, \overline{\lambda})$ define una única sucesión de puntos $(x_k, \lambda_k) \in \mathbb{R}^n \times \mathbb{R}^m$ para cada $k \in \mathbb{N}$ tales que el punto (x_{k+1}, λ_{k+1}) satisface (1.23) - (1.24), y esta sucesión converge a $(\overline{x}, \overline{\lambda})$ superlinealmente. Más aún, la velocidad de convergencia es cuadrática si ;as derivadas de f y h son fuertemente semismooth en $\overline{x}$.

Recordemos que las matrices del conjunto $\partial_x \left(\nabla_x \mathcal{L}_{MPCC}(x_k, \lambda_k)\right)$, en general, no son definidas positivas; y para obtener direcciones de descenso para alguna función de penalización es necesario que H_k sea definida positiva, lo que conlleva a pensar en la posibilidad de que H_k no satisfaga (2.11).

En este contexto, se introduce la condición de tipo Dennis-Moré si-

guiente:

$$\max_{W \in \partial_x(\nabla_x \mathcal{L}_{MPCC}(x_k,\lambda_k))} ||\pi_{\ker \nabla h(\overline{x})}((H_k - W)(x_{k+1} - x_k))|| = o(||x_{k+1} - x_k||), \quad (2.25)$$

y se tiene el siguiente teorema.

Teorema 2.2.2. *Si las derivadas de f y h son semismooth en un punto estacionario $\overline{x}$ de (1.6) - (1.7) y $\overline{\lambda}$ es el multiplicador de Lagrange asociado a $\overline{x}$. Supongamos además que la sucesión $((x_k, \lambda_k))_{k\in\mathbb{N}}$, $(x_k, \lambda_k) \in \mathbb{R}^n \times \mathbb{R}^m$, para cada $k \in \mathbb{N}$ es tal que el punto (x_{k+1}, λ_{k+1}) satisface (1.23) - (1.24) para alguna matriz simétrica $H_k \in \mathbb{R}^{n\times n}$, y que (x_k, λ_k) converge a $(\overline{x}, \overline{\lambda})$.*

Si la velocidad de convergencia de $(x_k)_{k\in\mathbb{N}}$ es superlineal, entonces la condición de Dennis-Moré (2.25) se cumple.

Recíprocamente, si

$$\liminf_{k\to\infty} \max_{W \in \partial_x(\nabla_x \mathcal{L}_{MPCC}(x_k,\lambda_k))} \langle W\xi, \xi\rangle > 0 \quad \forall \xi \in \ker(\nabla h(\overline{x})) \setminus \{0\}, \quad (2.26)$$

y la condición (2.25) se cumple, entonces la velocidad de convergencia de $(x_k)_{k\in\mathbb{N}}$ es superlineal.

Este teorema indica que la condición de tipo Dennis-Moré es necesaria para la convergencia superlineal primal y es suficiente si además se cumple (2.26).

Ahora, para una globalización de la convergencia, dado que las estrategias de globalización para los métodos SQP no requieren que las funciones sean 2 veces diferenciables, se pueden usar técnicas similares para el *semismooth* SQP. Por ejemplo, mediante búsqueda lineal para la función de penalización l_1.

2.3. Semismooth SQP aplicado a un *lifted* MPCC

En primer lugar, es importante notar que el valor de $\overline{y}$ de la variable auxiliar y en el problema P definido en la proposición 2.1.3, gracias a

(2.4) y (2.5), está únicamente determinado de la siguiente manera:

$$\overline{y}_j = \begin{cases} -\sqrt{F_j^2(\overline{x})} & \text{si} \quad (1,j) \in A(\overline{x}), \\ \sqrt{F_j^1(\overline{x})} & \text{si} \quad (2,j) \in A(\overline{x}), \\ 0 & \text{si} \quad j \in B(\overline{x}). \end{cases} \tag{2.27}$$

Además, note que podemos reescribir las condiciones de optimalidad de primer orden para el problema P: (2.6), (2.7) y (2.8), de la siguiente manera:

$$\begin{aligned} \nabla_x \mathcal{L}_{MPCC}(x,\lambda) = 0, \quad \nabla_y \mathcal{L}_P(x,y,\lambda) = 0, \\ (\text{máx}\{0,y\})^2 - F^1(x) = 0, \quad (\text{mín}\{0,y\})^2 - F^2(x) = 0, \end{aligned} \tag{2.28}$$

esto último dado que $s(y_j) = (\text{máx}\{0,y_j\})^2$ y

$$s(-y_j) = (\text{máx}\{0,-y_j\})^2 = (-\,\text{mín}\{0,y_j\})^2 = (\text{mín}\{0,y_j\})^2.$$

Vemos que

$$\frac{\partial \mathcal{L}_P}{\partial y_j}(x,y,\lambda) = 2\lambda_j^1\,\text{máx}\{0,y_j\} + 2\lambda_j^2\,\text{mín}\{0,y_j\}, \tag{2.29}$$

para cada $j \in \{1,2,\ldots,m\}$.

Claramente, $\frac{\partial \mathcal{L}_P}{\partial y_j}(x,y,\lambda)$ no es diferenciable cuando $y_j = 0$, por lo que (2.28) no es *smooth*. Sin embargo, [3] corrobora que se tienen los siguientes hechos:

1. Si $\overline{x} \in \mathbb{R}^n$ es un punto fuertemente estacionario de (1.4) y $\overline{\lambda} = (\lambda^1, \lambda^2) \in \mathbb{R}^m \times \mathbb{R}^m$ es su *MPCC*-multiplicador asociado, entonces $(\overline{x}, \overline{y}) \in \mathbb{R}^n \times \mathbb{R}^m$ con $\overline{y}$ dado por (2.27), es un punto estacionario del problema P y $\overline{\lambda}$ es su multiplicador asociado.

2. Recíprocamente, de la proposición 2.1.5 sabemos que si $(\overline{x}, \overline{y}) \in \mathbb{R}^n \times \mathbb{R}^m$ es un punto estacionario de P, entonces $\overline{x}$ es un punto débilmente estacionario de (1.4). Si además existe un multiplicador de Lagrange $\overline{\lambda} = (\overline{\lambda}^1, \overline{\lambda}^2) \in \mathbb{R}^m \times \mathbb{R}^m$ asociado a $(\overline{x}, \overline{y})$ tal que $\overline{\lambda}_j^i \geq 0$ para cada $j \in B(\overline{x})$, entonces $\overline{x}$ es un punto fuertemente estacionario de (1.4) y $\overline{\lambda}$ es su *MPCC*-multiplicador asociado.

3. Si las derivadas de F^1 y F^2 son *semismooth* en $\overline{x}$, entonces las deri-

vadas de las restricciones en P son *semismooth* en $(\overline{x}, \overline{y})$.

4. Sea $\overline{x} \in \mathbb{R}^n$ un punto factible de (1.4), si $(\xi, \eta) \in \mathbb{R}^n \times \mathbb{R}^m$ pertenece al espacio nulo del Jacobiano de las restricciones de P en $(\overline{x}, \overline{y})$, entonces $\xi \in K(\overline{x})$, donde

$$K(\overline{x}) = \{\xi \in \mathbb{R}^n : \nabla F_j^i \xi = 0, \ \ i \in \{1, 2\}, \ \ (i, j) \in A(\overline{x})\}.$$

5. Para un punto estacionario $\overline{x} \in \mathbb{R}^n$ de (1.4) y su *MPCC*-multiplicador asociado $\overline{\lambda} = (\overline{\lambda}^1, \overline{\lambda}^2) \in \mathbb{R}^m \times \mathbb{R}^m$, y $\overline{y}$ como en (2.27), el conjunto $(\partial_B)_{(x,y)} \left(\nabla_{(x,y)} \mathcal{L}_P(\overline{x}, \overline{y}, \overline{\lambda})\right)$ es el conjunto de todas las matrices de la forma

$$\mathcal{H} = \begin{pmatrix} \frac{\partial^2 \mathcal{L}_{MPCC}}{\partial x^2}(\overline{x}, \overline{\lambda}) & 0 \\ 0 & 2\text{diag}(a) \end{pmatrix},$$

con $a \in \mathbb{R}^m$ dado por

$$a_j = \begin{cases} 0 & \text{si} \quad j \in \{1, 2, \ldots, m\} \cap B(\overline{x})^c \\ \lambda_j^1 \text{ o } \lambda_j^2 & \text{si} \quad j \in B(\overline{x}) \end{cases}$$

2.3.1. Convergencia local del método

Para la iteración $(x_k, y_k, \lambda_k) \in \mathbb{R}^n \times \mathbb{R}^m \times (\mathbb{R}^m \times \mathbb{R}^m)$, con $\lambda_k = (\lambda_k^1, \lambda_k^2)$, cada iteración del semismooth SQP (1.18) - (1.19) aplicado a P consiste en resolver el subproblema

$$\begin{aligned} &\text{mín } \nabla f(x_k)^\top (x - x_k) + \tfrac{1}{2} \begin{pmatrix} x - x_k \\ y - y_k \end{pmatrix}^\top \mathcal{H}_k \begin{pmatrix} x - x_k \\ y - y_k \end{pmatrix} \\ &\text{s.a.} \\ &(\text{máx}\{0, y_k\})^2 - F^1(x_k) - \nabla F^1(x_k)(x - x_k) + 2B_{\text{máx}}(y_k)(y - y_k) = 0, \\ &(\text{mín}\{0, y_k\})^2 - F^2(x_k) - \nabla F^2(x_k)(x - x_k) + 2B_{\text{mín}}(y_k)(y - y_k) = 0, \end{aligned} \tag{2.30}$$

con $\mathcal{H}_k$ es una matriz simétrica de dimensión $(n + m) \times (n + m)$, y

$$B_{\text{mín}}(y_k) = \text{diag}(\text{mín}\{0, y_k\}), \qquad B_{\text{máx}}(y_k) = \text{diag}(\text{máx}\{0, y_k\}). \tag{2.31}$$

También, la caracterización de los puntos estacionarios en (1.23) - (1.24)

del subproblema anterior tiene la forma

$$\begin{aligned}
&\mathcal{H}_k \begin{pmatrix} x - x_k \\ y - y_k \end{pmatrix} + \begin{pmatrix} -\nabla F^2(x_k)^\top \\ 2B_{\text{mín}}(y_k) \end{pmatrix} (\lambda^2 - \lambda_k^2) \\
&\qquad + \begin{pmatrix} -\nabla F^1(x_k)^\top \\ 2B_{\text{máx}}(y_k) \end{pmatrix} (\lambda^1 - \lambda_k^1) = - \begin{pmatrix} \nabla_x \mathcal{L}_{MPCC}(x_k, \lambda_k) \\ \nabla_y \mathcal{L}_P(x_k, y_k, \lambda_k) \end{pmatrix} \qquad (2.32) \\
&-\nabla F^2(x_k)(x - x_k) + 2B_{\text{mín}}(y_k)(y - y_k) = - \left((\text{mín}\{0, y_k\})^2 - F^2(x_k) \right), \\
&-\nabla F^1(x_k)(x - x_k) + 2B_{\text{máx}}(y_k)(y - y_k) = - \left((\text{máx}\{0, y_k\})^2 - F^1(x_k) \right).
\end{aligned}$$

Por la discusión anterior, la elección natural de $\mathcal{H}_k$ es la siguiente:

$$\mathcal{H}_k = \begin{pmatrix} \frac{\partial^2 \mathcal{L}_{MPCC}}{\partial x^2}(x_k, \lambda_k) & 0 \\ 0 & 2\text{diag}(a(y_k, \lambda_k)) \end{pmatrix}, \qquad (2.33)$$

donde el vector $a(y_k, \lambda_k) \in \mathbb{R}^m$ está dado por

$$a_j(y_k, \lambda_k) = \begin{cases} (\lambda_k^2)_j & \textbf{si} \quad (y_k)_j < 0, \\ (\lambda_k^2)_j \, o \, (\lambda_k^1)_j & \textbf{si} \quad (y_k)_j = 0, \\ (\lambda_k^1)_j & \textbf{si} \quad (y_k)_j > 0, \end{cases} \qquad (2.34)$$

para cada $j \in \{1, 2, \ldots, m\}$.

Tenemos el siguiente teorema:

Teorema 2.3.1. *Si f, F^1 y F^2 son dos veces diferenciables, con segundas derivadas continuas en un punto fuertemente estacionario $\overline{x} \in \mathbb{R}^n$ que satisface (1.4), MPEC-LICQ, y sea $\overline{\lambda}$ el único MPCC-multiplicador asociado a $\overline{x}$. Supongamos también que se cumplen*

$$\lambda_j^i > 0, \quad \forall (i, j) \in B(\overline{x}), \qquad (2.35)$$

y la condición de optimalidad suficiente de segundo orden

$$\left\langle \nabla_x \mathcal{L}_{MPCC}(\overline{x}, \overline{\lambda})\xi, \xi \right\rangle > 0 \quad \forall \xi \in K(\overline{x}) \setminus \{0\}. \qquad (2.36)$$

Entonces, para cualquier elección de $\mathcal{H}_k$ de modo que (2.33) y (2.34) se verifiquen, cualquier punto inicial $(x_0, y_0, \lambda_0) \in \mathbb{R}^n \times \mathbb{R}^m \times (\mathbb{R}^m \times \mathbb{R}^m)$ suficientemente cerca de $(\overline{x}, \overline{y}, \overline{\lambda})$ define de manera única una sucesión de

puntos $(x_k, y_k, \lambda_k) \in \mathbb{R}^n \times \mathbb{R}^m \times (\mathbb{R}^m \times \mathbb{R}^m)$ *tales que para cada* $k \in \mathbb{N}$ *el punto* $(x_{k+1}, y_{k+1}, \lambda_{k+1})$ *satisface* (2.32)*, y esta sucesión converge hacia* $(\overline{x}, \overline{y}, \overline{\lambda})$ *a una velocidad superlineal. Más aún, si las derivadas de* f*,* F^1 *y* F^2 *son localmente Lipschitz continuas alrededor de* $\overline{x}$*, entonces la velocidad de convergencia es cuadrática.*

Notemos que en (2.33), la parte no *smooth* del problema se concentra en la parte inferior derecha de $\mathcal{H}_k$, mientras que la parte superior izquierda es dos veces diferenciable. Recordemos que para asegurar la convergencia de la búsqueda lineal para la función de penalización l_1,$\mathcal{H}_k$ debe ser, necesariamente, definida positiva. Para lograr esto, [3] re-define $\mathcal{H}_k$ de la siguiente manera:

$$\mathcal{H}_k = \begin{pmatrix} H_k & 0 \\ 0 & 2\,\mathrm{diag}(a^k) \end{pmatrix}, \tag{2.37}$$

donde H_k es alguna aproximación simétrica definida positiva de la matriz $\frac{\partial^2 \mathcal{L}_{MPCC}}{\partial x^2}(x_k, \lambda_k)$ y el vector $a^m \in \mathbb{R}^m$ se define componente a componente de la siguiente forma:

$$a_j^k = \max\{a_j(y_k, \lambda_k), \rho\,(\sigma(x_k, y_k, \lambda_k))\}, \tag{2.38}$$

con a_j definido en (2.34), $\rho : \mathbb{R}^+ \to \mathbb{R}^+$ alguna función continua tal que $\rho(0) = 0$ y $\rho(t)$ es está separado de 0 cuando t está separado de 0, y $\sigma : \mathbb{R}^n \times \mathbb{R}^m \times (\mathbb{R}^m \times \mathbb{R}^m) \to \mathbb{R}^+$ es el residuo del sistema (2.28) del problema P, definido por:

$$\sigma(x, y, \lambda) = \left\| \begin{pmatrix} \nabla_x \mathcal{L}_{MPCC}(x, \lambda) \\ \nabla_y \mathcal{L}_P(x, y, \lambda) \\ (\min\{0, y\})^2 - F^2(x) \\ (\max\{0, y\})^2 - F^1(x) \end{pmatrix} \right\|. \tag{2.39}$$

Con estas modificaciones, $\mathcal{H}_k$ es una matriz definida positiva, por lo cual la dirección SQP obtenida es una dirección de descenso para la función de penalización l_1, y puede ser usada para la globalización de la convergencia mediante búsqueda lineal. También, se tiene el siguiente resultado:

Teorema 2.3.2. *Si las derivadas de f, F^1 y F^2 son semismooth en un punto fuertemente estacionario $\overline{x} \in \mathbb{R}^n$ del problema (1.4), con un MPCC-multiplicador asociado $\overline{\lambda}$. Sea $((x_k, y_k, \lambda_k))_{k\in\mathbb{N}}$ tal que para cada $k \in \mathbb{N}$, $(x_k, y_k, \lambda_k) \in \mathbb{R}^n \times \mathbb{R}^m \times (\mathbb{R}^m \times \mathbb{R}^m)$ y el punto $(x_{k+1}, y_{k+1}, \lambda_{k+1})$ satisface (2.32), con $\mathcal{H}_k$ como en (2.37) con alguna matriz simétrica $H_k \in \mathbb{R}^{n\times n}$, y con $a^k \in \mathbb{R}^n$ definido en (2.38), (2.34) y (2.39) para alguna función $\rho : \mathbb{R}^+ \to \mathbb{R}^+$ tal que $\rho(t)\to 0$ siempre que $t \to 0^+$. Supongamos también que $((x_k, y_k, \lambda_k))_{k\in\mathbb{N}}$ converge a $(\overline{x}, \overline{y}, \overline{\lambda})$.*

Si la velocidad de convergencia de $((x_k, y_k))_{k\in\mathbb{N}}$ es superlineal, entonces la condición de tipo Dennis-Moré

$$\left|\left|\pi_{K(\overline{x})}\left(\left(H_k - \frac{\partial^2 \mathcal{L}_{MPCC}}{\partial x^2}(x_k, \lambda_k)\right)(x_{k+1} - x_k)\right)\right|\right| = o\left(\left|\left|\begin{pmatrix} x_{k+1} - x_k \\ y_{k+1} - y_k \end{pmatrix}\right|\right|\right) \tag{2.40}$$

se cumple.

Recíprocamente, si (2.35), (2.36) y (2.40) se satisfacen, entonces la velocidad de convergencia de (x_k, y_k) es superlineal.

2.3.2. Globalización de la convergencia del método

Como se vió en el punto anterior, específicamente en el Teorema 2.3.1, la convergencia que se asegura a partir del método SQP semi-suave es únicamente local.

Para mejorar esto y alcanzar una convergencia global a partir de la sucesión generada por el método plantedo, [3] sugiere la introducción de una familia de funciones de penalización.

Sea $\psi : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^+$ la función de penalización l_1 para las restricciones en P, es decir,

$$\psi(x, y) = \left|\left|\begin{pmatrix} (\text{mín}\{0, y\})^2 - F^2(x) \\ (\text{máx}\{0, y\})^2 - F^1(x) \end{pmatrix}\right|\right|_1. \tag{2.41}$$

Definimos su correspondiente familia de funciones de penalización, $\varphi_c : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}$, con

$$\varphi_c(x, y) = f(x) + c\psi(x, y),$$

donde $c > 0$ es el parámetro de penalización. Gracias a [6] sabemos que φ_c posee derivada direccional en toda dirección $p \in \mathbb{R}^n \times \mathbb{R}^m$, y además su derivada direccional en la dirección (p_1, p_2) está dada por

$$\varphi_c'((x,y);(p_1,p_2)) = \nabla f(x)^\top p_1 - c\psi(x,y),$$

Para la elección de los parámetros de penalización c_k en cada iteración, se debe asegurar que éstos decrezcan a medida que las iteraciones aumenta.

En se [9] muestra que este objetivo se alcanza si

$$c_k > ||\lambda_{k+}||_\infty$$

Así, presentamos el siguiente algoritmo que toma en cuenta la estructura de P:

Algoritmo 1. *Inicializar los parámetros $c > 0$ y $\varepsilon, \theta \in (0,1)$ y una función continua $\rho : \mathbb{R}^+ \to \mathbb{R}^+$ tal que $\rho(0) = 0$ y $\rho(t)$ esté separado de 0 cuando t está separado de 0. Escogemos el punto inicial $(x_0, y_0, \lambda_0) \in \mathbb{R}^n \times \mathbb{R}^m \times (\mathbb{R}^m \times \mathbb{R}^m)$ e inicializamos k en 0.*

(1) Si $\sigma(x_k, y_k, \lambda_k) = 0$, parar. Si no, elegir una matriz simétrica definida positiva $H_k \in \mathbb{R}^{n\times n}$ y defina $\mathcal{H}_k$ de acuerdo a (2.34) - (2.39). Calcule $(\overline{x}_{k+1}, \overline{y}_{k+1}) \in \mathbb{R}^n \times \in \mathbb{R}^m$ como una solución del sistema de ecuaciones (2.32) y su multiplicador de Lagrange asociado $\lambda_{k+1} \in \mathbb{R}^m \times \in \mathbb{R}^m$. Definir $\xi_k = \overline{x}_{k+1} - x_k$, $\eta_k = \overline{y}_{k+1} - y_k$

(2) Escoger un $c_k > 0$ tal que

$$c_k \geq ||\lambda_{k+1}||_\infty + c,$$

y calcule

$$\varphi_{c_k}'((x_k,y_k);(\xi_k,\eta_k)) = \nabla f(x_k)^\top \xi_k - c_k\psi(x_k,y_k)$$

(3) Fije $\alpha = 1$. Si la desigualdad

$$\varphi_{c_k}((x_k,y_k) + \alpha(\xi_k,\eta_k)) \leq \varphi_{c_k}(x_k,y_k) + \varepsilon\alpha\varphi_{c_k}'((x_k,y_k);(\xi_k,\eta_k))$$

se satisface, entonces $\alpha_k = \alpha$. *Si no, reemplace* α *por* $\theta \cdot \alpha$, *verifique la desigualdad anterior y repita nuevamente hasta que ésta sea válida.*

(4) Actualice

$$x_{k+1} = x_k + \alpha_k \xi_k, \qquad y_{k+1} = y_k + \alpha_k \eta_k,$$

$k = k + 1$ *y regrese al paso (1).*

La convergencia de la sucesión generada por este algoritmo se puede encontrar en [3].

Capítulo 3

Resultados Computacionales

3.1. Implementación

Para la implementación del Algoritmo 1 se usó el Lenguaje de programación Python, haciendo uso de las librerías *numpy* y *sympy* para la construcción del algoritmo y la librería *time* para medir el tiempo de ejecución del mismo.

Para el paso (1) del algoritmo, se definió la función σ acorde a (2.39) de la siguiente manera:

```
def sigma(y,multiplicador,f,G,H):
 aux = np.array([])
 aux = np.append(aux,
                 diferenciacion(lagrangiano1(multiplicador,f,G,H),
                                3,'x'))
 aux = np.append(aux,dlagrangiano2(y,multiplicador,f,G,H))
 aux = np.append(aux, min_vect(y) ** 2 - G)
 aux = np.append(aux, max_vect(y) ** 2 - H)
 norma = sqrt((aux @ aux))
 return norma
```

donde *diferenciacion* calcula el gradiente de una función, *lagrangiano1* devuelve el Lagrangiano de (1.4) en función de la variable x, *dlagrangiano2* retorna (2.29) y las funciones *max_vect* y *min_vect* calculan los vectores $\max\{0, y\}$ y $\min\{0, y\}$ respectivamente.

También, para la construcción de $\mathcal{H}_k$ en (2.37) se definen las funciones

```
def ai(y,multiplicador):
  a = np.array([])
  for i in range(len(y)):
    if y[i] <= 0:
      a = np.append(a,multiplicador[0,i])
    else:
      a = np.append(a,multiplicador[1,i])
  return a
```

que corresponde al a_j definido en (2.34), y

```
def ak(x,y,multiplicador,f,G,H,n):
  a = ai(y,multiplicador)
  ak = np.array([])
  for i in range(len(y)):
    ak = np.append(ak ,
                   max_(a[i],
                   rho(sustituir(sigma(y,multiplicador,f,G,H),
                                 x,n,'x'))))
  return ak
```

que comprende el vector a^k construido a partir de (2.38). En este último código *sustituir* es una función que evalúa x en la función requerida.

En lo que concierne al paso (2), se construyó la función ψ correspondiente a (2.41) de la siguiente forma:

```
def psi(x,y,G,F,m,n):
  vector = np.empty(2*m)
  for i in range(m):
    vector[i] = (min_vect(y)**2)[i] - sustituir(G[i],x,n,'x')
    vector[i+m] = (max_vect(y)**2)[i] - sustituir(F[i],x,n,'x')
  suma   = 0
  for i in range(len(vector)):
    suma = suma + abs(vector[i])
  return suma
```

Así mismo, para la regla de Armijo en el paso (3) se compuso la función

```
def armijo_rule(f,x,y,G,F,grad,dir1,dir2,alph,eps,c,n,var,theta,m):
    x_act = x + alph * dir1
    y_act = y + alph * dir2
```

```
    phi_x     = sustituir(f,x,n,var) + c * psi(x,y,G,F,m,n)
    phi_x_new = sustituir(f,x_act,n,var) + c * psi(x_act,
                                                   y_act,
                                                   G,F,m,n)
    while phi_x_new > phi_x + eps * alph * grad:
      alph = alph * theta
      x_act = x + alph * dir1
      y_act = y + alph * dir2
      phi_x_new = sustituir(f,x_act,n,var) + c * psi(x_act,
                                                     y_act,
                                                     G,F,m,n)
    return alph
```

Con estas funciones, se implementó el Algoritmo 1 de la siguiente manera:

En primer lugar, para el criterio de parada se usó el comando

```
while sustituir(sigma(y,multiplicador,f,G,F),x,n,'x') > 10 ** (-6):
```

para indicar que si $\sigma(x_k, y_k, \lambda_k) \leq 10^{-6}$, entonces se ha llegado a la solución óptima de P.

A continuación, para el paso (1) se procedió de la siguiente manera:

Respecto a la construcción de $\mathcal{H}_k$, se tiene el siguiente código

```
#paso 1
#Construccion de H caligrafica
H_k = np.array([np.concatenate((H[0],np.zeros(m)))])
aux = np.array(([]))
for i in range(1,n):
  aux = np.array([np.append(H[i],np.zeros(m))])
  H_k = np.concatenate((H_k, aux), axis=0)
for i in range(m):
  aux2 = np.array([np.append(np.zeros(n),
                             np.diag(2*ak(x,y,multiplicador,
                                          f,G,F,n))[i])])
  H_k = np.concatenate((H_k, aux2), axis=0)
```

donde n es la dimensión de x, m la dimensión de y y H representa la matriz H_k del algoritmo.

Seguidamente, para la construcción del sistema de ecuaciones que permitirán la obtención de las direcciones de descenso ξ_k, η_k y el nuevo

multiplicador λ_{k+1}, se tiene lo siguiente

```
#Resolucion del sistema de ecuaciones
x_  = np.array([x1,x2,x3])
y_  = np.array([y1,y2])
l_  = np.array(([l_g1, l_g2],[l_f1,l_f2]))

aux_x = (x_ - x).reshape(n,1)
aux_y = (y_ - y).reshape(m,1)
aux_g = np.append(-diferenciacion(G1,n,'x'),
                  -diferenciacion(G2,n,'x')).reshape(m,n)
aux_2 = np.concatenate((aux_g.reshape(n,m) , 2 * Bmin(y)), axis = 0)
aux_f = np.append(-diferenciacion(F1,n,'x'),
                  -diferenciacion(F2,n,'x')).reshape(m,n)
aux_3 = np.concatenate((aux_f.reshape(n,m) , 2 * Bmax(y)), axis = 0)

lag1  = np.array([])
for i in range(n):
  lag1 = np.append(lag1,
                   sustituir(
                   diferenciacion(
                   lagrangiano1(
                   multiplicador,f,G,F),n,'x')[i],x,n,'x'))

lag2  = dlagrangiano2(y,multiplicador,f,G,F).reshape(m,1)

G_eval  = np.array([])
F_eval  = np.array([])
for i in range(m):
  G_eval = np.append(G_eval,sustituir(G[i],x,n,'x'))
  F_eval = np.append(F_eval,sustituir(F[i],x,n,'x'))

Ec1 = np.dot(H_k ,np.concatenate((aux_x,aux_y),axis = 0))
    +  np.dot(aux_2 ,(l_ - multiplicador)[0].reshape(m,1))
    + np.dot(aux_3 , (l_ - multiplicador)[1].reshape(m,1))
    + np.append(lag1.reshape(n,1),lag2, axis = 0)
Ec2 = np.dot(aux_g,aux_x)
    + np.dot(2 * Bmin(y), aux_y)
    + ((min_vect(y) ** 2 - G_eval).reshape(m,1))
Ec3 = np.dot(aux_f,aux_x)
    + np.dot(2 * Bmax(y), aux_y)
    + ((max_vect(y) ** 2 - F_eval).reshape(m,1))

sis = np.concatenate((Ec1,Ec2,Ec3), axis = 0)
```

donde las ecuaciones Ec1, Ec2 y Ec3 corresponden al sistema de ecuaciones en (2.32). Para la resolución del sistema lineal se hizo uso de la función *linalg.solve* de la librería *numpy*, de la siguiente manera

```
q = [x1,x2,x3,y1,y2,l_g1,l_g2,l_f1,l_f2]
A = matriz_alargada(q , sis)[:,:n + 3 * m]
b = -matriz_alargada(q , sis)[:,n + 3 * m]

resp = np.linalg.solve(A,b)
x_new = resp[:n]
y_new = resp[n:n + m]
l_new = np.append([resp[n + m:n + 2 * m]],
                  [resp[n + 2 * m:n + 3 * m]], axis = 0)

xi_k   = x_new - x
etha_k = y_new - y
```

Luego, la implementación del paso (2) se organizó como a continuación:

```
# paso 2
c_k = np.amax(abs(l_new)) + c

aux  = np.array([])
for i in range(n):
  aux = np.append(aux,sustituir(grad_f[i],x,n,'x'))

grad  = (aux @ xi_k) - c_k * psi(x,y,G,F,m,n)
```

en donde se puede ver que el c_k que se toma es, justamente, el que cumple la igualdad del paso (2) en el Algoritmo 1, como sugiere [3].

Ahora, para la ejecución de la búsqueda de Armijo en el paso (3) se obtuvo lo siguiente:

```
#paso 3

#armijo
alpha = armijo_rule(f,x,y,G,F,grad, xi_k,
                    etha_k, 1, epsi,c_k,n,'x',theta,m)
```

con la función *armijo_rule* especificada anteriormente.

Finalmente, para el paso (4) del algoritmo se usó el código que sigue:

```
  #paso 4
  #actualizamos H
  lag = diferenciacion(lagrangiano1(l_new,f,G,F),n,'x')
  H = damped_BFGS_updating(H,x,x + alpha * xi_k,lag,n)

  #actualizacion de los datos
  x = x + alpha * xi_k
  y = y + alpha * etha_k
  multiplicador = l_new
```

Para la actualización de cada H_k se utilizó la actualización BFGS propuesta en [6], que se encuentra implementada en la función auxiliar *damped_BFGS_updating*, que es la siguiente

```
def damped_BFGS_updating(B,x,xk,lag,n):
    sk = (xk - x).reshape(n,1)
    yk = np.empty(n)
    for i in range(n):
      yk[i] = sustituir(lag[i],xk,n,'x') - sustituir(lag[i],x,n,'x')
    yk = yk.reshape(n,1)

    if float(np.dot(sk.T,yk)) >= 0.2*float(np.dot(sk.T,np.dot(B,sk))):
      theta_k = 1
    else:
      theta_k = float((0.8*np.dot(sk.T,np.dot(B,sk))))
                /float((np.dot(sk.T,np.dot(B,sk))-np.dot(sk.T,yk)))

    rk = theta_k * yk + (1 - theta_k) * np.dot(B,sk)

    Bk = B - np.dot(B,np.dot(sk,np.dot(sk.T,B)))
             /float(np.dot(sk.T,np.dot(B,sk)))
         + np.dot(rk,rk.T) / float(np.dot(sk.T,rk))
    return Bk
```

3.2. Resultados Numéricos

3.2.1. Ejemplo 1.

Consideremos el problema (1.4), con $n = 3$, $m = 2$ y

$$f(x) = 0{,}2x_1^2 + 0{,}1x_2^2 + 0{,}8x_3^3,$$
$$F^1(x) = \begin{pmatrix} x_2 \\ x_1 + x_2 - x_3 - 1 \end{pmatrix},$$
$$F^2(x) = \begin{pmatrix} x_1 \\ x_1 + x_2 + x_3 - 1 \end{pmatrix}.$$

El experimento numérico se realizó midiendo el tiempo de ejecución del algoritmo implementado, haciendo variaciones en la elección de α y H_k para medir la eficiencia del mismo y la proximidad a la solución óptima $x_* = (0, 1, 0)^\top$.

En todas las experiencias, se usaron los siguientes datos iniciales:

$$x_0 = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}, \quad y_0 = \begin{pmatrix} 1 \\ -1 \end{pmatrix}, \quad \lambda_0 = \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix}, \quad H_0 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix},$$

las constantes $c = 1$, $\varepsilon = 10^{(-4)}$, $\theta = 0{,}5$ y la función $\rho : \mathbb{R}^+ \to \mathbb{R}^+$ tal que

$$\rho(t) = \begin{cases} 0{,}1 & \text{si} \quad t > 0{,}1 \\ t & \text{si} \quad t \leq 0{,}1 \end{cases},$$

que son las constantes y función sugeridas en [8].

Así, para el algoritmo original descrito en la sección anterior se obtuvieron los siguientes resultados:

Solución $\overline{x}$	Iteraciones	Tiempo de ejecución (s)	Error $\|\|x_* - \overline{x}\|\|$
$\begin{pmatrix} 0 \\ 1 \\ 5{,}8869 \times 10^{-11} \end{pmatrix}$	7	1.6348	$8{,}3254 \times 10^{-11}$

Cuadro 3.1: Resultado Algoritmo 1.

Ahora, si utilizamos la matriz Hessiana exacta de $\mathcal{L}_{MPCC}$

$$\frac{\partial^2 \mathcal{L}_{MPCC}}{\partial x^2} = \begin{pmatrix} 0{,}4 & 0 & 0 \\ 0 & 0{,}2 & 0 \\ 0 & 0 & 4{,}8x_3 \end{pmatrix}, \tag{3.1}$$

obtenemos el resultado siguiente:

Solución $\overline{x}$	Iteraciones	Tiempo de ejecución (s)	Error $\|\|x_* - \overline{x}\|\|$
$\begin{pmatrix} 0 \\ 1 \\ 3{,}4256 \times 10^{-12} \end{pmatrix}$	7	1.7520	$4{,}8445 \times 10^{-12}$

Cuadro 3.2: Resultado Algoritmo 1 utilizando $\frac{\partial^2 \mathcal{L}_{MPCC}}{\partial x^2}$.

Finalmente, si no se actualiza H_k, es decir, $H_k = H_0 \ \forall k \in \mathbb{N}$, se tiene lo siguiente:

Solución $\overline{x}$	Iteraciones	Tiempo de ejecución (s)	Error $\|\|x_* - \overline{x}\|\|$
$\begin{pmatrix} 0 \\ 1 \\ -7{,}4801 \times 10^{-15} \end{pmatrix}$	8	1.8842	$1{,}0628 \times 10^{-14}$

Cuadro 3.3: Resultado Algoritmo 1 sin actualizar H_k

Como siguiente experimento, reemplazamos el α obtenido de la regla de Armijo por la constante $\alpha = 0{,}5$. Con esto, utilizando H_k como aproximaciones de $\frac{\partial^2 \mathcal{L}_{MPCC}}{\partial x^2}$ se tiene que:

Solución $\overline{x}$	Iteraciones	Tiempo de ejecución (s)	Error $\|\|x_* - \overline{x}\|\|$
$\begin{pmatrix} 1{,}1921 \times 10^{-7} \\ 0{,}9999 \\ -3{,}3749 \times 10^{-7} \end{pmatrix}$	23	11.5249	$5{,}8024 \times 10^{-7}$

Cuadro 3.4: Resultado Algoritmo 1 utilizando $\alpha = 0{,}5$.

Luego, manteniendo α constante y utilizando (3.1), se obtienen las cantidades a continuación:

Solución $\overline{x}$	Iteraciones	Tiempo de ejecución (s)	Error $\|\|x_* - \overline{x}\|\|$
$\begin{pmatrix} 4{,}7684 \times 10^{-7} \\ 0{,}9999 \\ -2{,}3729 \times 10^{-7} \end{pmatrix}$	21	4.6486	$8{,}7334 \times 10^{-7}$

Cuadro 3.5: Resultado Algoritmo 1 utilizando $\alpha = 0{,}5$ y $\frac{\partial^2 \mathcal{L}_{MPCC}}{\partial x^2}$.

Por último, manteniendo $\alpha = 0{,}5$ y sin actualizar H_k, tenemos lo siguiente:

Solución $\overline{x}$	Iteraciones	Tiempo de ejecución (s)	Error $\|\|x_* - \overline{x}\|\|$
$\begin{pmatrix} 1{,}1921 \times 10^{-7} \\ 0{,}9999 \\ -1{,}2317 \times 10^{-7} \end{pmatrix}$	23	5.6099	$2{,}9687 \times 10^{-7}$

Cuadro 3.6: Resultado Algoritmo 1 utilizando $\alpha = 0{,}5$ y sin actualizar H_k.

Como último experimento, se utilizó únicamente el método SQP para la resolución del problema sin utilizar la búsqueda lineal con la función de penalización l_1, es decir, las actualizaciones de x_{k+1}, y_{k+1} y λ_{k+1} emergen de la resolución del sistema de ecuaciones (2.32). Se obtuvo el siguiente resultado:

Solución $\overline{x}$	Iteraciones	Tiempo de ejecución (s)	Error $\|\|x_* - \overline{x}\|\|$
$\begin{pmatrix} 0 \\ 0{,}9999 \\ 5{,}39448 \times 10^{-10} \end{pmatrix}$	6	3.3643	$7{,}6294 \times 10^{-7}$

Cuadro 3.7: Resultado Algoritmo 1 sin la penalización l_1.

Decrecimiento de σ a partir del Algoritmo 1

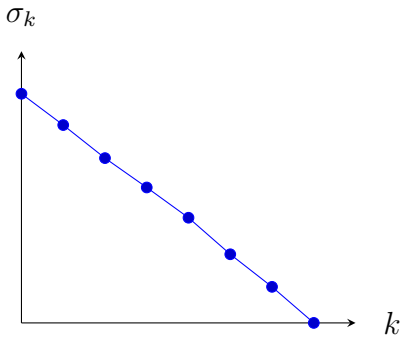

Figura 3.1: Decrecimiento de σ ejemplo 1.

donde $\sigma_k = \sigma(x_k, y_k, \lambda_k)$. Vemos que el valor de σ en cada iteración disminuye, lo que refleja que el algoritmo funciona correctamente para el ejemplo 1.

3.2.2. Ejemplo 2.

Estudiemos las soluciones del problema (1.4) en con $n = 2$, $m = 1$ y

$$\begin{aligned} f(x) &= (x_1 + 1)^2 + (x_2 - 2)^2, \\ F^1(x) &= x_2 - x_1, \\ F^2(x) &= x_2, \end{aligned}$$

que sabemos, gracias a [4], tiene las soluciones óptimas $x_*^1 = (-1, 0)^\top$ y $x_*^2 = (0{,}5, 0{,}5)^\top$.

A continuación, la gráfica del conjunto factible de este problema (en color azul) y sus soluciones óptimas

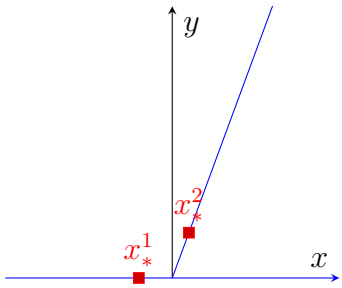

Figura 3.2: Conjunto factible ejemplo 2.

Si se toman los valores iniciales

$$x_0 = \begin{pmatrix} 10 \\ 10 \end{pmatrix}, \quad y_0 = 1, \quad \lambda_0 = \begin{pmatrix} 1 & 1 \end{pmatrix}, \quad H_0 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix},$$

las constantes $c = 1$, $\varepsilon = 10^{(-4)}$, $\theta = 0{,}5$ y la función $\rho : \mathbb{R}^+ \to \mathbb{R}^+$ tal que

$$\rho(t) = \begin{cases} 0{,}1 & \textbf{si} \quad t > 0{,}1 \\ t & \textbf{si} \quad t \leq 0{,}1 \end{cases}$$

y haciendo los mismos experimentos que en ejemplo anterior, se tienen los siguientes resultados:

Solución $\overline{x}$	Iteraciones	Tiempo de ejecución (s)	Error $\|\|x_*^1 - \overline{x}\|\|$
$\begin{pmatrix} -1 \\ 3{,}1275 \times 10^{-17} \end{pmatrix}$	6	0.7426	$1{,}3979 \times 10^{-8}$

Cuadro 3.8: Resultado Algoritmo 1.

Solución $\overline{x}$	Iteraciones	Tiempo de ejecución (s)	Error $\|\|x_*^2 - \overline{x}\|\|$
$\begin{pmatrix} 0{,}50000001 \\ 0{,}50000001 \end{pmatrix}$	4	0.4550	$1{,}0805 \times 10^{-8}$

Cuadro 3.9: Resultado Algoritmo 1 utilizando $\frac{\partial^2 \mathcal{L}_{MPCC}}{\partial x^2}$.

Solución $\overline{x}$	Iteraciones	Tiempo de ejecución (s)	Error $\|\|x_*^1 - \overline{x}\|\|$
$\begin{pmatrix} -0{,}9999 \\ 1{,}4708 \times 10^{-16} \end{pmatrix}$	66	10.7623	$1{,}9877 \times 10^{-8}$

Cuadro 3.10: Resultado Algoritmo 1 sin actualizar H_k.

Solución $\overline{x}$	Iteraciones	Tiempo de ejecución (s)	Error $\|\|x_*^1 - \overline{x}\|\|$
$\begin{pmatrix} -0{,}9999 \\ 1{,}7881 \times 10^{-7} \end{pmatrix}$	24	4.9250	$1{,}9877 \times 10^{-8}$

Cuadro 3.11: Resultado Algoritmo 1 utilizando $\alpha = 0{,}5$.

Solución $\overline{x}$	Iteraciones	Tiempo de ejecución (s)	Error $\|\|x_*^2 - \overline{x}\|\|$
$\begin{pmatrix} 0{,}5000003 \\ 0{,}5000003 \end{pmatrix}$	25	2.04611	$4{,}0915 \times 10^{-7}$

Cuadro 3.12: Resultado Algoritmo 1 utilizando $\alpha = 0{,}5$ y $\frac{\partial^2 \mathcal{L}_{MPCC}}{\partial x^2}$.

Solución $\overline{x}$	Iteraciones	Tiempo de ejecución (s)	Error $\|\|x_*^1 - \overline{x}\|\|$
$\begin{pmatrix} -1 \\ 3{,}5763 \times 10^{-7} \end{pmatrix}$	23	2.3654	$3{,}5763 \times 10^{-7}$

Cuadro 3.13: Resultado Algoritmo 1 utilizando $\alpha = 0{,}5$ y sin actualizar H_k.

Solución $\overline{x}$	Iteraciones	Tiempo de ejecución (s)	Error $\|\|x_*^1 - \overline{x}\|\|$
$\begin{pmatrix} -1{,}00000001 \\ -5{,}8541 \times 10^{-16} \end{pmatrix}$	6	2.3241	$1{,}3979 \times 10^{-8}$

Cuadro 3.14: Resultado Algoritmo 1 sin la penalización l_1.

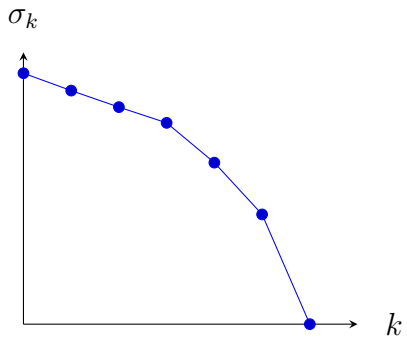

Figura 3.3: Decrecimiento de σ ejemplo 2.

Vemos que, nuevamente, el valor de σ decrece en cada iteración, lo que implica que el algoritmo también funciona correctamente para el ejemplo 2.

Estos resultados reflejan, en primer lugar, la importancia de implementar la regla de Armijo en conjunto con la búsqueda lineal con la función de penalización l_1, ya que juntas disminuyen considerablemente el tiempo de ejecución y el número de iteraciones que realiza el algoritmo. Vemos que cuando se fija $\alpha = 0{,}5$, el tiempo de ejecución del método SQP usual con la actualización BFGS de H_k (último experimento) realiza menos iteraciones que el Algoritmo 1 implementado, y los tiempos de ejecución de ambos experimentos son cercanos, por lo que la búsqueda lineal se vuelve importante para la globalización de la convergencia del algoritmo, mas no para la eficiencia del mismo.

Es importante notar que por la complejidad de ambos ejemplos, la matriz Hessiana de $\mathcal{L}_{MPCC}$ no requiere de demasiados cálculos, por lo que no existe mucha diferencia entre la ejecución del primer y segundo experimento.

Por último, notemos que los resultados del algoritmo implementado presentan una solución bastante aproximada a la solución real, por lo que los experimentos numéricos respaldan la teoría detallada en este trabajo.

Capítulo 4

Conclusiones y recomendaciones

4.1. Conclusiones

Existen varias técnicas para el estudio de problemas de programación matemática con restricciones de complementariedad dada la complejidad numérica para encontrar soluciones factibles si se trabaja directamente con las restricciones y la función objetivo de los mismos. Por este motivo, principalmente, éstas técnicas consisten en formular una representación adecuada del conjunto factible que no vulnere, sobretodo, las condiciones de calificación que se puedan cumplir en éste.

En este sentido, la técnica de *lifting* propuesta en [8] es de gran utilidad pues, a pesar de introducir una nueva variable al problema, conserva la condición de calificación necesaria para caracterizar puntos estacionarios del problema original, además de que permite obtener las soluciones óptimas a partir de un problema menos complejo en el cual se pueden utilizar técnicas de optimización no lineal conocidas, como el método SQP.

El algoritmo SQP semi-suave propuesto en [3] aprovecha la estructura del nuevo problema asociado al MPCC original, pues utiliza esta reformulación para construir los sub-problemas que se resolverán a través del método SQP.

Para la obtención de las matrices definidas positivas requeridas al implementar el algoritmo SQP semi-suave, el concepto de subdiferencial

juega un rol muy importante, pues la caracterización de éste bajo los supuestos descritos a lo largo del presente trabajo, permite alcanzar una matriz definida positiva de la cual se conoce su estructura y se puede implementar de manera más sencilla.

Se implementó el algoritmo SQP semi-suave en un ambiente Python y, a partir de los resultados numéricos obtenidos, se entiende el uso de la función de penalización y de la implementación de la regla de Armijo, pues le brindan eficiencia al algoritmo al disminuir el tiempo de ejecución y el número de iteraciones. Además, se comprendió la relevancia de elegir una correcta aproximación para la matriz Hessiana dentro del algoritmo.

4.2. Recomendaciones

Se sugiere probar distintas técnicas para la aproximación de la Hessiana del Lagrangiano del MPCC que disminuya el tiempo de ejecución del cálculo de la misma, para aumentar la eficiencia del algoritmo presentado.

Se recomienda estudiar el comportamiento del algoritmo para problemas de mayor escala y los cambios, de ser necesarios, que se requieren para una generalización del mismo.

Se plantea buscar opciones distintas para la resolución numérica del sistema de ecuaciones (2.32) que mejoren la eficiencia del algoritmo.

Referencias bibliográficas

[1] Philip Caplan. Numerical computation of second derivatives with applications to optimization problems. *Massachusetts Institute of Technology*, Dec 2012.

[2] Ronald H.W. Hoppe. Lecture notes in optimization theory, Fall 2006.

[3] A. Izmailov, A. Pogosyan, and M. Solodov. Semismooth sqp method for equality-constrained optimization problems with an application to the lifted reformulation of mathematical programs with complementarity constraints. *Optimization Methods & Software*, 26:847–872, 10 2011.

[4] Houyuan Jiang and Daniel Ralph. Qpecgen, a matlab generator for mathematical programs with quadratic objectives and affine variational inequality constraints. *Computational Optimization and Applications*, 13(1):25–59, Apr 1999.

[5] Zhi-Quan Luo, Jong-Shi Pang, and Daniel Ralph. *Mathematical Programs with Equilibrium Constraints*. Cambridge University Press, 1996.

[6] Jorge Nocedal and Stephen J. Wright. *Numerical Optimization*. Springer, New York, NY, USA, 2e edition, 2006.

[7] Holger Scheel and Stefan Scholtes. Mathematical programs with complementarity constraints: Stationarity, optimality, and sensitivity. *Mathematics of Operations Research*, 25(1):1–22, 2000.

[8] O. Stein. Lifting mathematical programs with complementary constraints. *Math. Program. Ser. A*, 131:71–94, 2012.

[9] W. Sun and Ya-Xiang Yuan. *Optimization Theory and Methods for Nonlinear Programming*. Springer Optimization and Its Applications, 2006.