Algoritmos de optimización para un problema de control óptimo discreto

MAT282 / 2023-2

Ernesto Antu Treumún Araya

Noviembre 2023

Abstract

In this work, we deal with a discrete quadratic optimal control problem derived from a Modular Multilevel Cascaded Converters model. We present two approaches to address this problem: the classic method and a transformed one. In the end, we analyze which classic quadratic programming algorithms are more suitable for this case, observing their behavior in terms of objective function values and execution time.

1. Descripción del problema

Una de las áreas de la matemática que ha tomado más relevancia en el contexto ingenieril en los últimos años es el Control Óptimo. Esta área surge con la motivación de, como su nombre lo indica, controlar óptimamente un sistema, donde "óptimamente" es entendido en cada contexto particular. En términos generales, un problema de control óptimo puede ser escrito como:

$$\begin{aligned} &\min J(x) \\ &\text{s.t.} \begin{cases} \dot{x}(t) = f(t, x, u(t)), & t \in [0, T] \\ x(0) \in K \\ u(t) \in U, \end{cases} \end{aligned}$$

donde u representa nuestro *control*, es decir, las distintas formas en las que podemos actuar en un sistema. La restricción $u(t) \in U$ refleja nuestras limitaciones en el modo de actuación, mientras que la condición $x(0) \in K$ representa los posibles estados iniciales del sistema.
Existen distintos tipos de sistema de control: algunas veces se intenta minimizar el tiempo, otras veces un costo asociado al movimiento, etc. Un ejemplo clásico de control óptimo es el siguiente:

Ejemplo: Al conducir un automóvil, el conductor tiene la capacidad de controlar la aceleración del vehículo junto con el ángulo en el que se mueve (mediante el acelerador y el manubrio, respectivamente). En algunos casos, el conductor podría estar interesado en minimizar el combustible consumido al estacionar un automóvil partiendo desde un punto inicial x_0, teniendo una tolerancia máxima de tiempo T.
Si el lector está interesado en más detalles, puede consultar en [1]. ■
Algunos otros ejemplos clásicos de control pueden ser encontrados en el libro [2].

En el presente trabajo, nos centraremos en un problema de control óptimo *a tiempo discreto* en sistemas de redes eléctricas, el cual detallaremos a continuación:
En el paradigma energético mundial actual, uno de los principales objetivos es crear matrices energéticas libres de carbono. Para lograrlo, es esencial que distintos sistemas eléctricos, tales como generación, transmisión y almacenamiento de energía, se conecten a una red eléctrica a través de Convertidores Electrónicos de Potencia. El propósito de estos sistemas es manejar eficientemente el uso y producción de la energía eléctrica, manteniéndose siempre a niveles de energía compatibles con los equipos y la misma red eléctrica.
En este contexto nacen los Modular Multilevel Cascaded Converters (MMCC) como una excelente opción a tener en consideración para trabajar con altas potencias. Países como España, Francia y Alemania han creado plantas eléctricas con estos sistemas. Para más detalles puede consultar aquí para el caso de España y Francia y aquí para el caso de Alemania.

Como se puede evidenciar en la descripción anterior, la herramienta clave a considerar es el Control Óptimo: para manejar eficientemente los MMCC, es fundamental controlar de manera óptima los voltajes del condensador de la máquina sincrónica. Existen diversas formas de abordar este problema, pero en nuestro caso nos centraremos en controlar las corrientes circulantes $\boldsymbol{i}_z \in \mathbb{R}^{N_z}$ inter-cluster. Específicamente, dicho sistema se puede modelar, después de emplear un esquema de Euler-explícito, como un problema de control óptimo a tiempo discreto dado por

$$\begin{aligned} \min_{\boldsymbol{i}_z(\ell)} \quad & \sum_{\ell \in \mathcal{H}} \left\| \mathbf{E}_C(\ell+1) - \mathbf{E}_C^* \right\|_{\boldsymbol{Q}}^2 + \lambda_z \left\| \boldsymbol{i}_z(\ell) \right\|_2^2 + \lambda_0 v_0^2(\ell) \\ \text{s.t.} \quad & \begin{cases} \mathbf{E}_C(\ell+1) = \mathbf{E}_C(\ell) + T_s \left[\operatorname{diag}\left\{ \mathbf{A}^{\mathrm{T}} \boldsymbol{v}_{\Psi\chi}(\ell) \right\} + v_0(\ell) \mathbf{I}_9 \right] \left(\boldsymbol{i}_{\Psi\chi}(\ell) + \boldsymbol{i}_z(\ell) \right) \\ \mathbf{A} \boldsymbol{i}_z(\ell) = 0 \\ \check{v}_0(\ell) \leqslant v_0(\ell) \leqslant \widehat{v_0}(\ell) \\ \check{i}_{\kappa z}(\ell) \leqslant i_{\kappa z}(\ell) \leqslant \widehat{i}_{\kappa z}(\ell), \quad \forall \ell \in \mathcal{H} = \{0, 1, \ldots, N_p - 1\}, \quad \forall \kappa \in \mathcal{K} = \{1, \ldots, 9\} \end{cases} , \end{aligned} \tag{1}$$

donde $\mathbf{E}_C(\ell)$ denota la energía del sistema en tiempo ℓ y $v_0(\ell) \in \mathbb{R}$ al voltaje común del sistema. Si pensamos el problema de control como uno que tiene como variables de control a $\boldsymbol{i}_z$ y v_0 **simultáneamente**, resulta ser un problema no lineal debido al producto cruzado $v_0^\top \boldsymbol{i}_z$. Dada esta gran dificultad, en el presente trabajo se asumirá v_0 como un dato y se trabajará solo como un problema de control en $\boldsymbol{i}_z$.
Para el caso particular de este problema, hay dos observaciones importantes a tener en consideración:

1. Se necesita obtener una solución en tiempos menores a 0,1 ms.
2. Por esta misma razón, se prioriza, en general, factibilidad por sobre minimización.

Estudios teniendo en cuenta las características mencionadas anteriormente pueden ser encontrados en [3].

Teniendo en cuenta el paradigma anterior, los objetivos generales del presente trabajo fueron:

1. Estudiar distintos algoritmos de control óptimo discreto. Principalmente, estamos interesados en aquellos con tiempos de ejecución más bajos.
2. Optimizar los algoritmos anteriores mediante transformaciones, para así lograr mejores tiempos de ejecución.

Para lograr aquello, los objetivos específicos fueron:

1. Investigar sobre distintos algoritmos de optimización cuadrática e implementar en MatLab aquellos que no se encuentren en librerías usuales, para hacer, finalmente, una comparativa en el desempeño obtenido.
2. Implementar una transformación del problema y hacer un nuevo análisis de desempeño.

Finalizamos la introducción comentando que las herramientas matemáticas necesarias para abordar nuestro problema fueron conocimientos sobre álgebra lineal y algoritmos de problemas cuadráticos convexos.

2. Metodología de trabajo

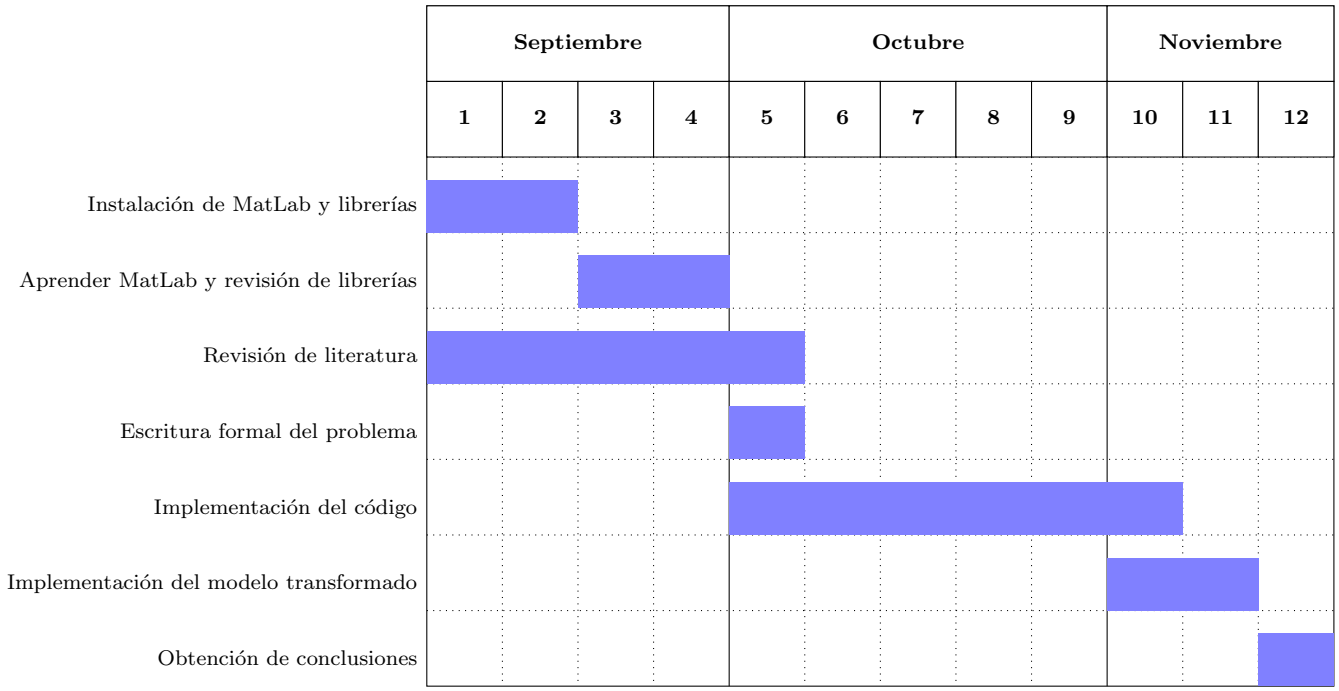
2.1. Fechas de reuniones con el especialista

Las fechas de reuniones con el especialista Andrés Mora fueron las siguientes:

- Lunes 4 de septiembre de 2023.
- Martes 3 de octubre de 2023.
- Miércoles 18 de octubre de 2023.
- Viernes 10 de noviembre de 2023.
- Viernes 17 de noviembre de 2023.

2.2. Metodología de trabajo

Decidimos tener reuniones solamente de carácter presencial, con una frecuencia aproximada de cada dos o tres semanas, según la disponibilidad del especialista y del estudiante. Durante cada reunión discutimos los resultados obtenidos desde la última reunión hasta la presente, obteniendo las conclusiones correspondientes. Adicionalmente, en cada reunión, el especialista guiaba al estudiante en cómo proceder en el trabajo futuro y, de este modo, el estudiante realizaba de manera independiente las tareas señaladas. El detalle de cada tarea se puede visualizar en la carta Gantt adjunta.



3. Desarrollo del problema

3.1. Background matemático

Como se verá más adelante, el problema planteado se puede reescribir como un problema lineal cuadrático de la forma

$$\min_{\boldsymbol{i}} \quad \boldsymbol{i}^\top H\boldsymbol{i} + b^\top \boldsymbol{i}, \qquad \text{s.t. } \begin{cases} \boldsymbol{i}_{\text{mín}} \leq \boldsymbol{i} \leq \boldsymbol{i}_{\text{máx}} \\ \mathbf{A}\boldsymbol{i} = 0, \end{cases}$$

el cual resulta ser convexo. Es por ello que daremos una pequeña pincelada de teoría respecto a los algoritmos a utilizar para resolver el problema de minimización.

3.1.1. Frank-Wolfe Algorithm

Como se mencionó en la introducción, es importante obtener una solución *factible* en lugar de minimizar la función objetivo. Es por ello que el algoritmo "Frank-Wolfe" nace como una buena opción. A grandes rasgos, este algoritmo simplifica la función objetivo a su aproximación lineal, lo que mejora considerablemente los tiempos de ejecución. De manera general, si tenemos un problema de minimización

$$\min_{x \in B} g(x),$$

donde B es un conjunto convexo de restricciones, en cada iteración x_t Frank-Wolfe considera la solución s^t del problema linealizado

$$\min_{s \in B} \langle \nabla g(x_i), s \rangle$$

y considera $x_{t+1} = (1 - \eta_t)x_t + \eta_t s^t$. En general, existen dos maneras clásicas de definir el paso óptimo, las cuales están dadas por

$$\eta_t = \frac{2}{t+2} \qquad \text{ó} \qquad \eta_t = \arg\min_{\eta \in [0,1]} g\left((1-\eta)w^t + \eta s^t\right).$$

En lo que sigue del documento, llamaremos "Frank-Wolfe sin paso óptimo" al algoritmo de Frank-Wolfe $\eta_t = \frac{2}{t+2}$, y "Frank-Wolfe con paso óptimo" en el segundo caso.
En general, el algoritmo de Frank-Wolfe tiene una tasa de convergencia de $O(1/t)$, lo que en general no lo hace una buena opción. Sin embargo, como se mencionó anteriormente, en nuestro caso a estudiar resulta ser un buen candidato.
Una buena referencia para investigar sobre este algoritmo es [4, Chapter 2.2], donde el algoritmo es llamado como "Conditional gradient method".

3.1.2. Interior-Point Algorithm

Un algoritmo clásico para resolver problemas no lineales con restricciones de desigualdad e igualdad es el algoritmo "Interior Point". La manera en la que está implementado en MatLab, y por tanto, la que usaremos se detalla como sigue:
Al momento de estudiar el problema de minimización

$$\min_{x \in \mathbb{R}^n} f(x), \qquad \text{s.t. } h(x) = 0, g(x) \leq 0,$$

donde $h : \mathbb{R}^n \to \mathbb{R}^m, g : \mathbb{R}^n \to \mathbb{R}^k$ son funciones vectoriales, la restricción de desigualdad $g(x) \leq 0$ genera problemas, por lo que la solución que propone este método es definir un nuevo problema agregando variables de holgura $s \geq 0$. Explícitamente, en cada iteración se resuelve el subproblema

$$\min_{x,s} f_\mu(x,s)) := f(x) - \mu \sum_{i=1}^{k} \ln(s_i), \qquad \text{s.t. } s \geq 0, h(x) = 0, g(x) + s = 0,$$

donde, a medida que $\mu \to 0$, las soluciones x_μ del problema aproximado, bajo condiciones razonables sobre las funciones, convergen hacia una solución del problema original. En general, estos sub-problemas se resuelven ocupando las condiciones de KKT y un paso del método de Newton o bien del gradiente conjugado.
Destacamos que el nombre de este método es debido a que cada solución x_μ obtenida del problema aproximado

pertenece en el interior del conjunto de restricciones $g(x) \leq 0$, y por ende el algoritmo busca óptimos a partir de puntos en el interior del conjunto.

Para el caso cuadrático se puede explotar un poco más su estructura y realizar algunos cálculos teóricos explícitos para mejorar el desempeño del algoritmo. En [5, Chapter 16.6] se pueden encontrar más detalles. Destacamos también la página de soporte de MatLab, a la cual se puede acceder desde aquí.

3.1.3. Active-Set Algorithm

Un algoritmo clásico para resolver problemas cuadráticos de minimizaciónes el algoritmo de "Active Set". Existen tres versiones de este algoritmo, pero en este documento solo consideraremos el algoritmo "Active-Set" primal.
El funcionamiento de este algoritmo, como su nombre indica, consiste en detectar las restriciónes *activas* de un punto dado, y resolver un subproblema ocupando dichas restricciones. Explícitamente, si tenemos el problema cuadrático

$$\text{Minimizar} \qquad x^\top G x + b^\top x, \qquad \text{s.t.} \qquad a_i^\top x \geq b_i, \quad c_j^\top x = d_j, \qquad \forall i \in I, \forall j \in J,$$

y tenemos un punto x_0 dado, el conjunto de *restricciones activas* I' son los índices $i \in I$ tales que $g_i(x_0) = 0$. Active-set parte de un punto inicial x_0, detecta el conjunto I' de restricciones activas y considera una dirección d_0 de movimiento de tal suerte que $x_0 + d_0$ resuelve el problema

$$\text{Minimizar} \qquad x^\top G x + b^\top x, \qquad \text{s.t.} \qquad g_i(x) = 0, \quad h_j(x) = 0, \qquad \forall i \in I', \forall j \in J.$$

Posteriormente, elige un paso $\alpha_0 \in [0, 1]$ de tal suerte que $x_0 + \alpha_0 d_0$ satisfaga todas las restricciones del problema original. Generalmente, se escoje α_0 como el mayor número en $[0, 1]$ tal que $x_0 + \alpha_0 d_0$ satisface las restricciones, donde cabe destacar que siempre existe gracias a que si hay una restricción no activa, por continuidad podemos seguir manteniendo dicha restricción bajo pequeñas perturbaciones. El resto de las iteraciones se realizan de la misma manera, reemplazando los índices "0" por índices "k".
Como comentario final, nos gustaría recalcar que es sabido que este algoritmo funciona bien a pequeña escala (tanto en iteraciones como en dimensión), pero a gran escala empieza a generar problemas. Esto lo veremos evidenciado en los resultados obtenidos.
Finalmente, si el lector lo desea puede consultar [5, Chapter 16.5] para más detalles respecto a este algoritmo.

3.2. Análisis del modelo

3.2.1. Problema original

Las librerías de Matlab necesitan inputs de la forma

$$\text{mín} \quad x^\top Hx + b^\top \boldsymbol{i}, \qquad \text{s.t.} \begin{cases} x_{\text{mín}} \leq x \leq x_{\text{máx}} \\ \mathbf{A}\boldsymbol{i} = 0, \end{cases}$$

Por ende, nuestro primer trabajo es transcribir el problema propuesto a la forma deseada, y para ello incorporaremos la variable temporal como parte de nuestra variable espacial. En nuestro caso, nuestra variable de decisión es $\mathbf{i}_z(\ell) \in \mathbb{R}^9$. De este modo, si definimos

$$B(k) := T_s[\text{diag}\left\{\mathbf{A}^{\text{T}}\boldsymbol{v}_{\Psi\chi}(k)\right\} + v_0(k)\mathbf{I}_9] \in \mathcal{M}_{9\times 9}, \qquad \widehat{K}(k) := T_s\left[\text{diag}\left\{\mathbf{A}^{\text{T}}\boldsymbol{v}_{\Psi\chi}(k)\right\} + v_0(k)\mathbf{I}_9\right]\boldsymbol{i}_{\Psi\chi}(k) \in \mathbb{R}^9$$

entonces $E_C(\ell+1) = E(\ell) + B(\ell)\boldsymbol{i}(\ell) + \widehat{K}(\ell)$, de donde,

$$E(\ell) = E(0) + \sum_{k=1}^{\ell} B(k)\boldsymbol{i}(k) + \underbrace{\left(\sum_{k=1}^{\ell}\widehat{K}(k)\right)}_{:=C(\ell)}.$$

Si escribimos $\boldsymbol{i} := (\boldsymbol{i}(1), ..., \boldsymbol{i}(N_p-1))^\top \in \mathbb{R}^{9\cdot(N_p-1)}$ y $\widehat{B}(\ell) := [B(1)|B(2)|\cdots|B(\ell)|0\cdots|0] \in \mathcal{M}_{9\times 9(N_p-1)}$, entonces $E(\ell) = E_0 + \widehat{B}(\ell)\boldsymbol{i} + C(\ell)$. Por tanto,

$$J(\boldsymbol{i}) = \sum_{\ell=1}^{N_p-1}\left(\|\mathbf{E}_0 + C(\ell) - \mathbf{E}_C^* + \widehat{B}(\ell)\boldsymbol{i}\|_{\mathbf{Q}}^2\right) + \lambda\|\boldsymbol{i}\|_2^2 + \lambda_0\sum_{\ell=1}^{N_p-1} v_0^2(\ell).$$

Dado que el último término es constante, no afecta en nuestro problema de minimización, y por ende obviaremos dicho térmno en J. Por otro lado, llamaremos $W(\ell)$ al término en rojo. De este modo,

$$J(\boldsymbol{i}) = \sum_{\ell=1}^{N_p-1}\left(\|W(\ell)\|_{\mathbf{Q}}^2 + 2W(\ell)^\top\mathbf{Q}\widehat{B}(\ell)\boldsymbol{i} + \|\widehat{B}(\ell)\boldsymbol{i}\|_{\mathbf{Q}}^2\right) + \lambda\|\boldsymbol{i}\|_2^2.$$

Nuevamente, el término $\sum_{\ell=1}^{N_p-1}\|W(\ell)\|_{\mathbf{Q}}^2$ es constante en $\boldsymbol{i}$, por lo que lo despreciaremos en el funcional J. Por otro lado, la linealidad del producto interno nos permite escribir

$$\sum_{\ell=1}^{N_p-1} 2W(\ell)^\top\mathbf{Q}\widehat{B}(\ell)\boldsymbol{i} = \left(2\sum_{\ell=1}^{N_p-1} W(\ell)^\top\mathbf{Q}\widehat{B}(\ell)\right)\mathbf{i}.$$

Del mismo modo,

$$\lambda\|\boldsymbol{i}\|_2^2 + \sum_{\ell=1}^{N_p-1}\|\widehat{B}(\ell)\boldsymbol{i}\|_{\mathbf{Q}}^2 = \lambda\boldsymbol{i}^\top\mathbf{I}_{n\cdot(N_p-1)}\boldsymbol{i} + \sum_{\ell=1}^{N_p-1}\boldsymbol{i}^\top\widehat{B}(\ell)^\top\mathbf{Q}\widehat{B}(\ell)\boldsymbol{i} = \boldsymbol{i}^\top\left(\lambda\mathbf{I}_{n\cdot(N_p-1)} + \left[\sum_{\ell=1}^{N_p-1}\widehat{B}(\ell)^\top\mathbf{Q}\widehat{B}(\ell)\right]\right)\boldsymbol{i}$$

Si definimos, por tanto,

$$\mathbf{b}_1 := \left(2\sum_{\ell=1}^{N_p-1}\widehat{B}(\ell)^\top\mathbf{Q}\mathbf{W}(\ell)\right), \quad \mathbf{H_1} := \left(\sum_{\ell=\mathbf{1}}^{\mathbf{N_p-1}}\widehat{\mathbf{B}}(\ell)^\top\mathbf{Q}\widehat{\mathbf{B}}(\ell)\right) + \lambda\mathbf{I}_{\mathbf{n}\cdot(\mathbf{N_p-1})},$$
$$\mathbf{A}_1 := [A|A|\cdots|A], \qquad (A \text{ repetida } N_p-1 \text{ veces.})$$

podemos reescribir finalmente nuestro problema como

$$\min_{\boldsymbol{i}\in\mathbb{R}^{9\cdot(N_p-1)}} \boldsymbol{i}^\top\mathbf{H}_1\boldsymbol{i} + \mathbf{b}_1^\top\boldsymbol{i}, \qquad \text{s.t.} \begin{cases} i_{\text{mín}} \leq \boldsymbol{i} \leq i_{\text{máx}} \\ \mathbf{A}_1\boldsymbol{i} = 0 \end{cases}$$

Observación 1. Este problema resulta ser convexo. En efecto, dado que $\mathbf{Q}$ es definida positiva, y por ende cada término de la suma en H lo es.

En el caso particular de nuestro problema, tenemos que

$$A = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ -1 & 0 & 0 & -1 & 0 & 0 & -1 & 0 & 0 \\ 0 & -1 & 0 & 0 & -1 & 0 & 0 & -1 & 0 \\ 0 & 0 & -1 & 0 & 0 & -1 & 0 & 0 & -1 \end{pmatrix}.$$

Notamos que A tiene kernel de dimensión 4 y por ende **no es** de rango completo, lo que dificulta enormemente la implementación de algunos algoritmos de optimización, incluso para el caso de la linealización ocupando Frank-Wolfe, pues, por ejemplo, el método simplex usual solo funciona para matrices de rango completo.
Por otro lado, debido a la alta *no invertibilidad* de $\mathbf{A}$, abordar el problema con la restricción $\mathbf{A}\boldsymbol{i} = \mathbf{b_{eq}}$ parece no ser buena idea. De hecho, veremos más adelante que los condicionamientos de las matrices involucradas en los procesos internos de los algoritmos tienen números de condicionamiento muy cercanos a cero. Es por ello que, en la siguiente sección, abordaremos un problema *transformado*, esperando, de antemano, que mejoren los desempeños de los algoritmos.

3.3. Escritura de problema transformado

Como se mencionó anteriorente, la restrición de igualdad $\mathbf{A}\boldsymbol{i}_z = 0$ que provoca que la mayoría de algoritmos tengan problemas de convergencia o no tengan un buen desempeño. Es por ello que se intentó otro *approach* para el problema, en el cual se evita trabajar explícitamente con la matriz $\mathbf{A}$ haciendo una transformación de variable. Explícitamente: debido a la restricción $\mathbf{A}\boldsymbol{i}_z = 0$ (con el problema no aumentado), tenemos que los $\boldsymbol{i}$ son elementos del kernel de A, ie, están el subespacio generada por los vectores columna de la matriz

$$\mathbf{N}' = \begin{bmatrix} 1 & 1 & 1 & 1 \\ -1 & 0 & -1 & 0 \\ 0 & -1 & 0 & -1 \\ -1 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & -1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

En particular, como $\mathbf{A}\boldsymbol{i} = 0 \in \mathbb{R}^{6\cdot(N_p-1)}$, podemos notar que $\boldsymbol{i} \in \bigoplus_{k=1}^{N_p-1} \langle \mathbf{N}' \rangle$, donde $\langle \mathbf{N}' \rangle$ es el espacio generado por las columnas de $\mathbf{N}$ y donde entendemos la suma directa en el sentido de agregar "ceros" en las primeras $(k-1)\cdot 9$ coordenadas y en las últimas $(N_p - k - 1)\cdot 9$ coordenadas. Bajo esta premisa, podemos definir la matriz $\mathbf{N}$ dada por

$$\mathbf{N} := \begin{pmatrix} \mathbf{N}' & 0 & 0 & \cdots & 0 \\ 0 & \mathbf{N}' & 0 & \ddots & 0 \\ 0 & 0 & \mathbf{N}' & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 & \mathbf{N}' \end{pmatrix}$$

y obtener que $\boldsymbol{i} \in \langle \mathbf{N} \rangle$. Es por ende que podemos definir la variable transformada $\widehat{\mathbf{i}} \in \mathbb{R}^{4\cdot(N_p-1)}$ que guarda los coeficientes respecto a la base descrita anteriormente del vector $\mathbf{i} \in \ker(\mathbf{A})$, es decir, $\boldsymbol{i} = \mathbf{N} \cdot \widehat{\boldsymbol{i}}$.
Bajo esta relación, la función objetivo $J(\boldsymbol{i}) = \boldsymbol{i}^\top \mathbf{H}_1 \boldsymbol{i} + \mathbf{b}_1^\top \boldsymbol{i}$ del problema se puede escribir ahora como

$$\mathbf{J}(\widehat{\boldsymbol{i}}) = (\mathbf{N}\widehat{\boldsymbol{i}})^\top \mathbf{H}_1 (\mathbf{N}\widehat{\boldsymbol{i}}) + \mathbf{b}_1^\top \mathbf{N}\widehat{\boldsymbol{i}} = \widehat{\boldsymbol{i}}^\top (\mathbf{N}^\top \mathbf{H}_1 \mathbf{N})\widehat{\boldsymbol{i}} + (\mathbf{N}^\top \mathbf{b}_1)^\top \widehat{\boldsymbol{i}}.$$

Definiendo $\mathbf{H} := \mathbf{N}^\top \mathbf{H}_1 \mathbf{N}$ y $\mathbf{b} := \mathbf{N}^\top \mathbf{b}_1$, obtenemos que nuestra función de costos es

$$\mathbf{J}(\widehat{\boldsymbol{i}}) = \widehat{\boldsymbol{i}}^\top \mathbf{H}\widehat{\mathbf{i}} + \mathbf{b}^\top \widehat{\mathbf{i}},$$

ie, la naturaleza del problema **no** cambia. Por otro lado, por construcción la restricción $\mathbf{A}\boldsymbol{i} = 0$ **no** nos entrega condiciones sobre $\widehat{\boldsymbol{i}}$. Respecto a la restricciones de caja, lamentablemente son transformadas a una restricción tipo *polítopo*, quedando de la forma

$$i_{\text{mín}} \leq \boldsymbol{i} \leq i_{\text{máx}} \iff \begin{pmatrix} -\mathbf{N} \\ \mathbf{N} \end{pmatrix} \cdot \widehat{\mathbf{i}} \leq \begin{pmatrix} -i_{\text{mín}} \\ i_{\text{máx}} \end{pmatrix}.$$

Definiendo

$$\mathbf{B} := \begin{pmatrix} -\mathbf{N} \\ \mathbf{N} \end{pmatrix} \qquad \mathbf{c} := \begin{pmatrix} -i_{\text{mín}} \\ i_{\text{máx}} \end{pmatrix},$$

obtenemos, finalmente, que nuestro problema transformado está dado por

$$\text{mín} \quad \widehat{\boldsymbol{i}}^{\top} \mathbf{H} \widehat{\mathbf{i}} + \mathbf{b}^{\top} \widehat{\mathbf{i}}, \qquad \text{s.t. } \mathbf{B}\widehat{\boldsymbol{i}} \leq \mathbf{c}$$

Para más detalles el lector puede consultar [3].

4. Implementación

Uno de los primeros problemas que enfrentamos al trabajar fue que, hasta donde sabemos, el algoritmo de Frank-Wolfe no se encuentra implementado en ninguna librería de MatLab. Por lo tanto, se procedió a implementarlo manualmente. Particularmente, se implementó el algoritmo de Frank-Wolfe con y sin paso óptimo.
Por otro lado, para facilitar la accesibilidad y manipulación del código, se decidió crear distintos archivo con las distintas funciones a trabajar. Específicamente:

1. *"quadratic.m"* es una función $f(A, x)$ la cual calcula $x^{\top} Ax$.
2. *"RealModel.m"* es una función que toma todos los parámetros del problema, tales como $\mathbf{Q}, A, v_{\psi_\xi}, E_C^*, \lambda$, etc, y arroja los valores $\mathbf{H}_1$ y $\mathbf{b}_1$ de la sección 3.1.
3. *"FWOptStep.m"* es el algoritmo implementado de Frank-Wolfe, el cual acepta, dentro de sus argumentos, un 0 si se quiere ocupar el algoritmo sin paso óptimo y un 1 con paso óptimo (c.f. 3.1.1).
4. *"TotalComparison.m"* es el código más grande y es donde se hace la comparativa entre los desempeños de los algoritmos en cada uno de los casos.

En el siguiente link de GitHub pueden ser encontrados todos los códigos implementados. Cabe destacar que los códigos están comentados para facilitar la comprensión del lector.

5. Resultados e interpretaciones

En esta sección se presenta una evaluación de desempeño entre los distintos algoritmos programados para el problema original y el problema transformado. Como se mencionó anteriormente, los algoritmos que utilizamos para comparar el desempeño fueron "Frank Wolfe" con y sin paso óptimo, "Interior Point" y "Active set". Para los últimos dos, se utilizó la librería *fmincon*. Cabe destacar que existen otras librerías de MatLab específicas para problemas cuadráticos, como *QuadProg*, pero estas librerías **no** permiten un máximo número de iteraciones, lo que nos llevó a descartarlas debido al objetivo del informe.
Durante todo lo que sigue, tomamos los siguientes valores de referencia:

$$N_p = 10, \qquad \lambda_z = 1, \qquad \mathbf{E}_C^* = 875 \cdot \mathbb{1} \in \mathbb{R}^9, \qquad T_s = 1, \qquad \check{i}_{\kappa z}(\ell) = -5, \widehat{i}_{\kappa z}(\ell) = 5, \quad \forall \ell \in \{0, ..., N_p - 1\}$$

$$v_0 = 0 \in \mathbb{R}^9, \qquad \boldsymbol{v}_{\Psi\chi}(\ell) = \begin{pmatrix} 300\cos\left(100\pi t_\ell\right) \\ 300\cos\left(100\pi t_\ell - \frac{2\pi}{3}\right) \\ 300\cos\left(100\pi t_\ell + \frac{2\pi}{3}\right) \\ 110\cos\left(\frac{100}{3}\pi t_\ell\right) \\ 110\cos\left(\frac{100\pi}{3} t_\ell - \frac{2\pi}{3}\right) \\ 110\cos\left(\frac{100\pi}{3} t_\ell + \frac{2\pi}{3}\right) \end{pmatrix}, \quad t_\ell = \ell T_s, \qquad \boldsymbol{i}_{\psi_\chi} = \mathbb{1} \in \mathbb{R}^{9\cdot(N_p-1)}$$

Finalmente, destacamos que se decidió **no** incluir los gráficos del algoritmo de Frank-Wolfe sin paso óptimo debido a que, en comparación, los valores obtenidos por él para la función objetivo eran demasiado altos, lo que dificultaba la visualización de las imágenes. De todas maneras, si el lector tiene curiosidad y desea ver los gráficos con dicho algoritmo, puede consultar el siguiente link de GitHub, donde se encuentran todos los algoritmos confeccionados e imágenes obtenidas.

5.1. Problema original

Respecto al problema original, a continuación se presentan los gráficos con los desempeños obtenidos.

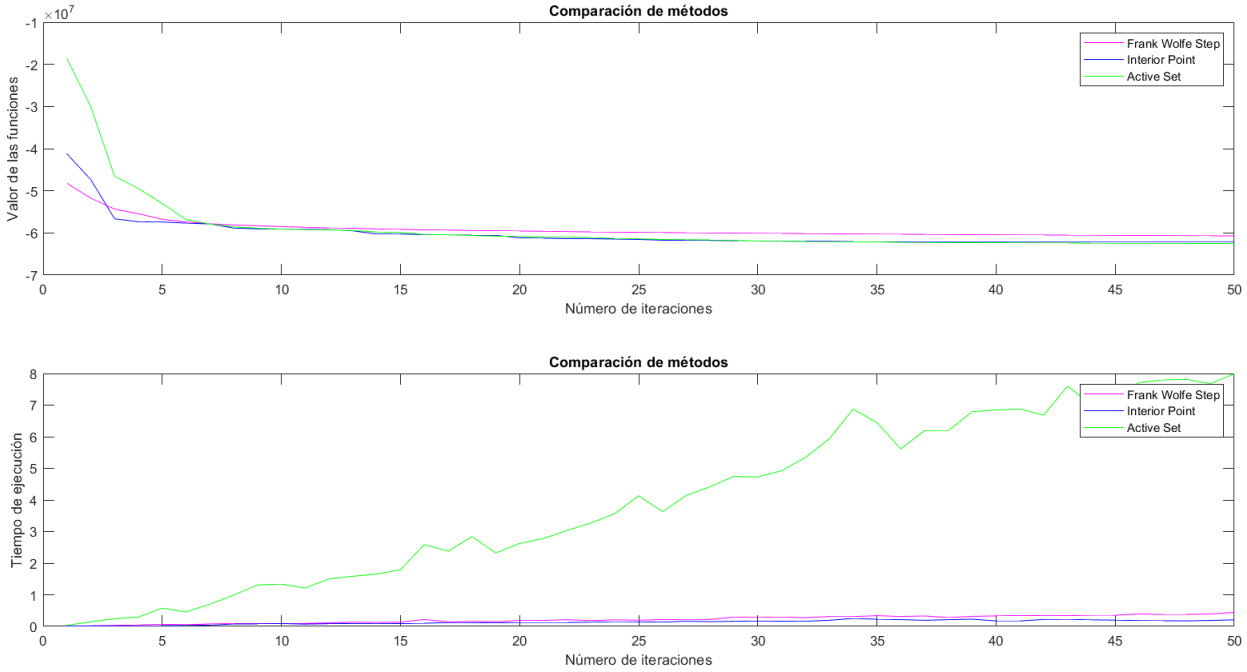


Figura 1: Comparación en el desempeño de los algoritmos.

Destacamos lo siguiente:

- El método de Active-Set fue el que tuvo un peor desempeño en cuanto a tiempos de ejecución, lo cual se condice con la teoría, pues Active-Set es un algoritmo con buen comportamiento solamente a pequeña escala.

- El algoritmo de Interior-Point fue el que tuvo el mejor desempeño en ambos casos. Sin embargo creemos que la razón por la que los tiempos de ejecuciones son menores a los de Frank-Wolfe es debido a la mejor implementación de la librería *fmincon*, pues teóricamente el algoritmo de Frank-Wolfe debiese tener mejores tiempos de ejecución.

- A pesar de la simpleza del modelo de Frank-Wolfe, los valores obtenidos en la función objetivo son bastante cercanos a los obtenidos por el resto de algoritmos, lo que lo hace un tentador candidato a utilizar debido a los bajos tiempos de ejecución.

5.2. Problema transformado

Respecto al problema transformado, a continuación se presentan los gráficos con los desempeños obtenidos.

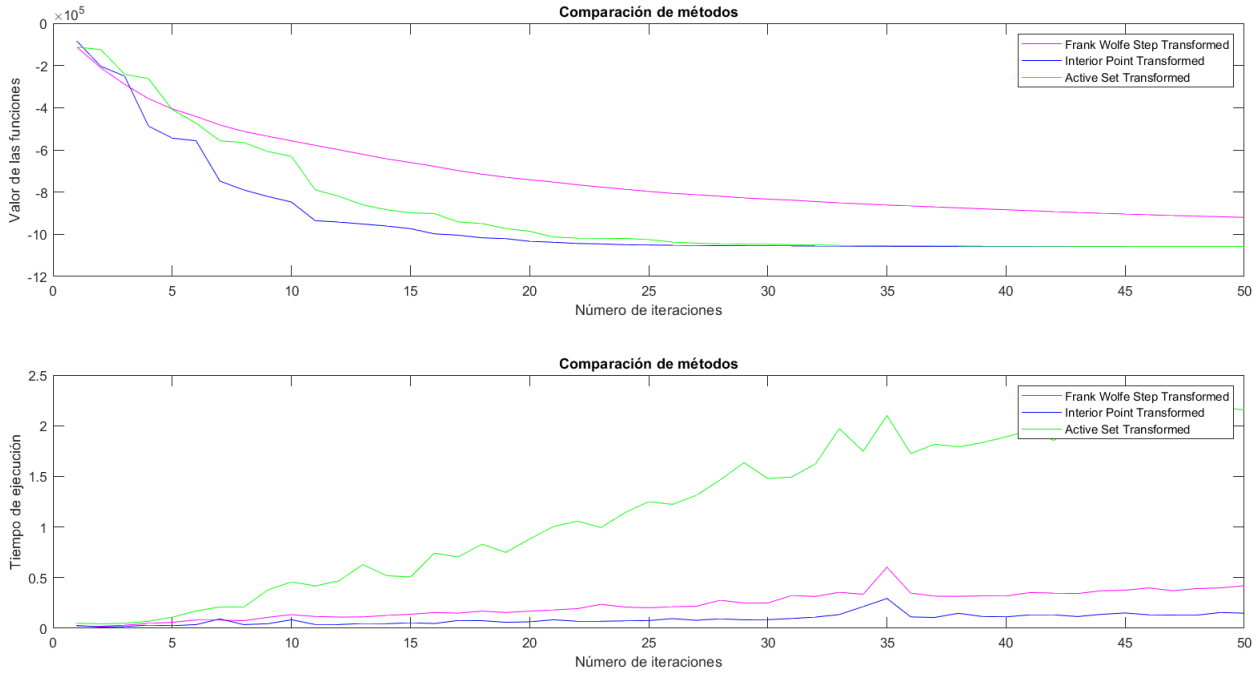


Figura 2: Comparación en el desempeño de los algoritmos.

Destacamos lo siguiente:

- A diferencia del caso anterior, los valores de la función objetivo obtenidas por Frank-Wolfe son notoriamente mayores a los de los otros dos algoritmos. Esto de un punto de vista teórico es esperable, debido a que Frank-Wolfe solo minimiza el primer término de la serie de Taylor del problema.

- Al igual que en el caso anterior, en cuanto a tiempos de ejecución Interior-Point y Frank-Wolfe tuvieron comportamientos muy similares y significativamente menores a los de Active-Set, lo cual se condice con la teoría, pues Active-Set es un algoritmo con buen comportamiento solamente a pequeña escala. Sin embargo, al igual que en el caso anterior, creemos que el mejor desempeño del algoritmo Interior-Point se debe a la mejor implementación en la librería *fmincon*.

5.3. Comparativa entre ambos problemas

A continuación se muestran tablas conjuntas con los desempeños de cada uno de los algoritmos. Para efectos visuales, los gráficos con líneas segmentadas corresponden a los obtenidos con el problema transformado y los con líneas continuas para el problema original.

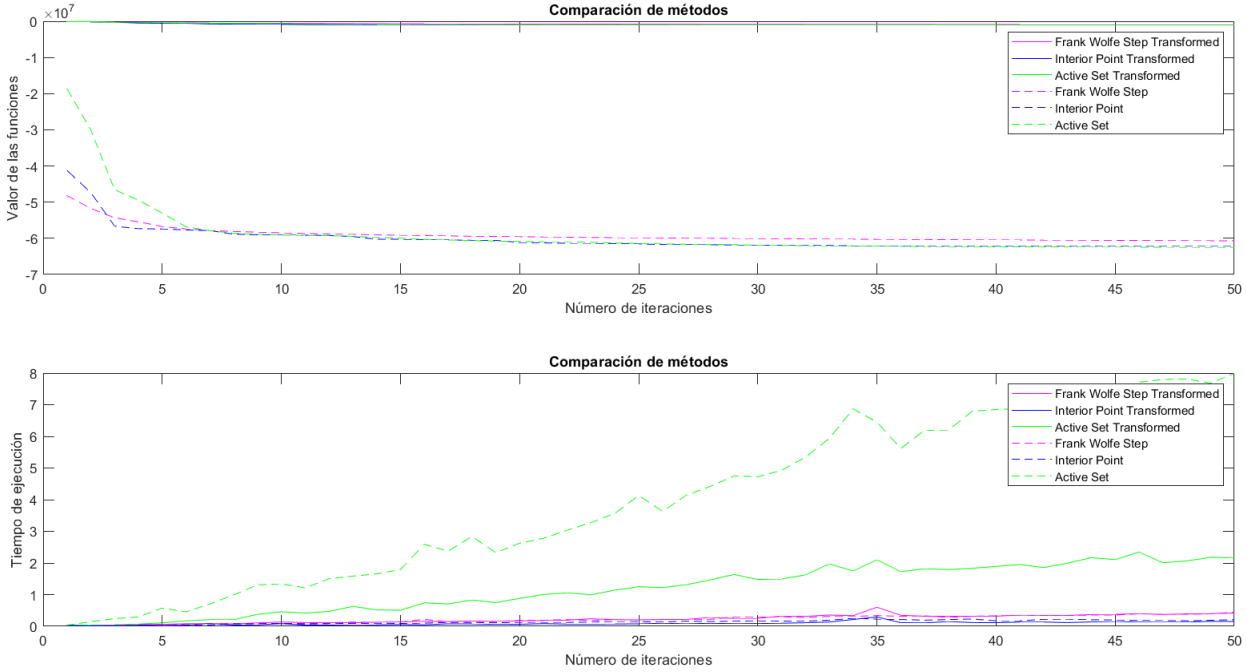


Figura 3: Comparativa de los algoritmos. En el primer caso valores mínimos obtenidos y para el segundo tiempos de ejecución.

Observamos dos comportamientos claros:

- Los algoritmos con la función **no** transformada obtienen valores de la función objetivo significativamente menor a los del problema transformado, pues el primero obtiene soluciones en el orden de 10^7 y el segundo de 10^5. La razón de esta gran diferencia es que los condicionamientos de las matrices intermedias en los algoritmos para el caso del problema transformado tienen un número de condicionamiento de cond $\approx 2,15 \cdot 10^{-16}$, lo que provoca que los errores de máquina sean significativos. Este no es el caso del problema transformado, pues allí los condicionamientos de las matrices no generan problemas.

- Los algoritmos con la función **no** transformada tienen un peor desempeño en cuanto a tiempo de ejecución, es decir, los algoritmos que operan con la función transformada son más rápidos.

A partir de lo anterior podemos afirmar que no es fiable trabajar con el problema no transformado debido al extremadamente bajo número de condicionamiento de las matrices intermedias.

6. Conclusiones

A partir de los resultados y comentarios obtenidos en la secciones anteriores, concluimos que trabajar con el modelo sin transformar no es recomendable debido a la poca fiabilidad en los cálculos de la máquina, consecuencia directa de la nula invertibilidad (ie, condicionamientos cercanos a 0) de las matrices involucradas.

Por otro lado, creemos que con una implementación más meticulosa, el algoritmo de Frank-Wolfe podría incluso sobrepasar a Interior-Point. Estas mejoras pueden ir en dirección de explotar la estructura particular del problema lineal cuadrático trabajado. Nuestro sustento principal es que, a pesar de que el desempeño del primero en cuanto a tiempos de ejecución haya sido peor, se pudo apreciar que la diferencia fue ínfima con el segundo.

En este contexto, consideramos que el principal aporte realizado en este trabajo fue proporcionar respuestas satisfactorias a los desempeños de los distintos algortimos situados en el problema de MMCC, respaldadas por las implementaciones de código necesarias. Estas contribuciones no solo fortalecen el ámbito teórico, sino que también tienen el potencial de guiar futuros trabajos en el escenarios del mundo real.

Por otro lado, nos gustaría destacar que el hecho de haber asumido v_0 como dato y no como variable nos permitió ocupar técnicas de problemas lineales cuadráticos, o siendo incluso un poco más generales, de problemas convexos. Sin embargo, si asumimos v_0 como variable, el término no lineal $v_0^\top \boldsymbol{i}_z$ imposibilita por completo el uso de estos algoritmos, por lo que todavía no es claro cómo abodarlo.

Para finalizar, algunas ideas para posibles trabajos futuros es aprovechar la estructura cíclica de las matrices $\mathbf{A} = [A|A\cdots|A]$ y $\mathbf{N} = \text{diag}(N, N, \cdots, N)$ para simplificar y disminuir los cálculos intermedios en los procesos de minimización. Para lograr esto, se debiese implementar manualmente los algoritmos para este caso particular haciendo cálculos teóricos de multiplicaciones y, de este modo, disminuir la cantidad de *flops* que deba hacer la máquina.

Referencias

[1] Frédéric Jean. Géométrie différentielle et application au contrôle géométrique. 2022.

[2] Alberto Bressan and Benedetto Piccoli. *Introduction to the mathematical theory of control*, volume 1. American institute of mathematical sciences Springfield, 2007.

[3] Rodrigo Cuzmar, Andrés Montenegro, Andrés Mora, Javier Pereda, and Ricardo P Aguilera. Constrained mpc for intercluster energy control of modular multilevel matrix converters. *IEEE Transactions on Industrial Electronics*, 2023.

[4] Dimitri P Bertsekas. Nonlinear programming. *Journal of the Operational Research Society*, 48(3):334–334, 1997.

[5] Jorge Nocedal and Stephen J Wright. *Numerical optimization*. Springer, 1999.