UNIVERSIDAD TECNICA
FEDERICO SANTA MARIA

Memoria de Titulación

" IMPLEMENTACIÓN Y EVALUACIÓN DE ALGORITMOS DE OPTIMIZACIÓN QP DE PEQUEÑA ESCALA EN PLATAFORMAS DE CONTROL DIGITAL PARA APLICACIONES DE ALTO DESEMPEÑO DINÁMICO "

Braulio Alejandro Vera Vergara

Profesor Patrocinante
Andrés Mora

Correferente
Alejandro Angulo

6 de octubre de 2025

CONSTANCIA DE VALIDACIÓN Y CONFIDENCIALIDAD DE MONOGRAFÍA A REPOSITORIO ACADÉMICO

1.- IDENTIFICACIÓN DEL TRABAJO ACADÉMICO

Tipo de monografía (marcar una opción): ☒ Memoria o trabajo de título ☐ Tesis de Postgrado

Título del trabajo: Implementación y evaluación de algoritmos de optimización QP de pequeña escala en plataformas de control digital para aplicaciones de alto desempeño dinámico

Nombre del candidato(a): Braulio Alejandro Vera Vergara

Carrera / Grado: Ingeniería Civil Eléctrica

Campus: Casa Central **Departamento:** Ingeniería Eléctrica

2.- VALIDACIÓN DEL PROFESOR GUÍA/DIRECTOR DE TESIS

Yo, Andrés Mora Castro, en mi calidad de profesor(a) guía/director(a) del trabajo académico mencionado anteriormente **DEJO CONSTANCIA** que:

- He revisado esta versión del documento y corresponde a la versión final aprobada del trabajo.
- El trabajo cumple con los requisitos académicos y de formato establecidos por la institución.

3.- EVALUACIÓN DE CONFIDENCIALIDAD POR PROPIEDAD INDUSTRIAL (marcar una opción)

☒ El trabajo **NO contiene** información que amerite confidencialidad y puede ser publicado de inmediato en repositorio con acceso abierto.

☐ El trabajo **CONTIENE** información con potenciales implicancias de propiedad industrial o intelectual y requiere un periodo de confidencialidad (**embargo**) por **(marcar una opción)**:

☐ 6 meses ☐ 12 meses ☐ 2 años ☐ 3 años ☐ 5 años ☐ 10 años

Fundamentación de la necesidad de confidencialidad (obligatorio si se solicita embargo):

4.- FIRMAS

Profesor(a) guía o director(a) de memoria o tesis:

Fecha: 11-12-2025 **Firma:** 

Estudiante o Candidato(a):

Fecha: 05-11-2025 **Firma:** 

Este formulario debe ser insertado como página 2 de la memoria o tesis, completado y firmado por estudiante y profesor(a) antes de la entrega en portal PRISMA de Biblioteca USM.

Tabla de Contenidos

Figuras

Tablas

Resumen

Este trabajo presenta la formulación y evaluación de un problema de control predictivo basado en modelo (MPC) para un convertidor tipo Grid-Following (GFL), considerando la dinámica del sistema de potencia y las restricciones de operación del convertidor. Se describe la modelación del sistema y la formulación del problema de optimización, implementándose y comparando dos algoritmos: Active-Set y Frank-Wolfe. La evaluación incluye escenarios con cambios escalonados de potencia, lo que permite analizar tanto el seguimiento de referencias en régimen estacionario como la carga computacional durante los transitorios. Las pruebas se realizan mediante simulaciones en Simulink y experimentos Hardware-in-the-Loop (HIL), donde la planta se emula en RTDS y el controlador se ejecuta en una tarjeta Imperix. Los resultados muestran que el algoritmo Active-Set alcanza un desempeño superior en eficiencia computacional, incluso frente a la función integrada `mpcActiveSetSolver` de MATLAB, confirmando su viabilidad para aplicaciones en tiempo real en sistemas de potencia con limitaciones de cómputo.

1. Introducción

La creciente preocupación por la sostenibilidad y la reducción de emisiones de gases de efecto invernadero ha impulsado la integración progresiva de fuentes de energía renovable (Renewable Energy Sources, RES) en las redes eléctricas [2]. La generación de energía mediante RES requiere el uso de convertidores electrónicos conectados a la red, que actúan como interfaz entre las fuentes renovables y la red eléctrica. En la mayoría de los casos, estos convertidores operan bajo un esquema de control *Grid-Following* (GFL), en el cual se sincronizan con la tensión de la red en el punto de conexión común (PCC) y regulan la potencia activa y reactiva inyectada mediante lazos de control de potencia y corriente [2].

El correcto funcionamiento de los convertidores GFL es fundamental para garantizar la estabilidad del sistema, mantener la calidad de la tensión y la corriente, y permitir una mayor penetración de energías renovables. Sin embargo, a medida que aumenta la proporción de dispositivos electrónicos de potencia en la red, la inercia del sistema disminuye y el factor de cortocircuito (Short-Circuit Ratio, SCR) en el PCC se reduce, lo que da lugar a características de red débil y riesgos de inestabilidad. Para abordar estos desafíos, se han propuesto estrategias de control híbridas que combinan las ventajas de los modos Grid-Following y Grid-Forming (GFM), mejorando la estabilidad y el soporte de tensión bajo condiciones de red débil [2].

En este contexto, las estrategias de control predictivo, particularmente el *Model Predictive Control* (MPC), permiten predecir el comportamiento futuro del sistema y optimizar la acción de control en función de referencias deseadas. Esto se logra resolviendo, en cada instante de muestreo, un problema de optimización cuadrática (QP) que determina la secuencia de control óptima [3].

Dado que los métodos genéricos de optimización utilizados en software comerciales pueden no ser lo suficientemente eficientes para ejecutarse en plataformas de control digital con recursos limitados, resulta de interés implementar algoritmos de optimización especializados para este tipo de problema. En este trabajo se consideran, en particular, los métodos *Active-Set* y *Frank-Wolfe*, los cuales ofrecen enfoques distintos de resolución. La eficiencia de ambos será analizada en términos de velocidad de solución, convergencia y viabilidad de implementación en hardware orientado a control en tiempo real.

2. Objetivos

2.1. Objetivo general

Implementar dos algoritmos de optimización QP en una plataforma digital de control para evaluar su desempeño en convertidores Grid-Following.

2.2. Objetivos específicos

- Formular problema de control óptimo de horizonte rodante MPC para convertidores GFL.
- Programar dos algoritmos de resolución de problemas QP: Active-Set y Frank-Wolfe.
- Evaluar y comparar computacionalmente el desempeño de los dos algoritmos, usando como criterio de desempeño la convergencia y el número de operaciones en punto flotante (FLOPs).
- Evaluar experimentalmente el desempeño del control predictivo utilizando metodología hardware-in-the-loop con los algoritmos programados en plataforma de control Imperix, usando como criterio el tiempo de solución.

3. Descripción y modelado del sistema

El sistema bajo estudio corresponde a un convertidor de potencia conectado a un Sistema Eléctrico de Potencia (SEP), cuya función es inyectar energía de forma controlada manteniendo la tensión de salida en los niveles deseados. Este convertidor opera bajo un esquema Grid-Following (GFL), sincronizando su salida con la tensión del SEP en el punto de conexión común (PCC). A diferencia de los esquemas convencionales basados en lazos PI en cascada, en este trabajo se emplea una estrategia de control predictivo (MPC) para regular la inyección de potencia activa y reactiva.

Se utiliza un convertidor trifásico de tres niveles tipo *Neutral Point Clamped* (3L-NPC) con un filtro LCL, lo que permite una mejor calidad de tensión en el PCC y una menor distorsión armónica en comparación con un convertidor de dos niveles. La modulación del convertidor se realiza mediante *Space Vector Pulse Width Modulation* (SVPWM), que además de un control preciso de los niveles de salida, permite aumentar la utilización máxima de la tensión de entrada, alcanzando un factor de modulación teórico de hasta 1.15 [4], en lugar del límite de 1 de la modulación sinusoidal convencional.

Las Figuras 1 y 2 muestran, respectivamente, la topología general del sistema trifásico y el detalle del convertidor 3L-NPC utilizado.

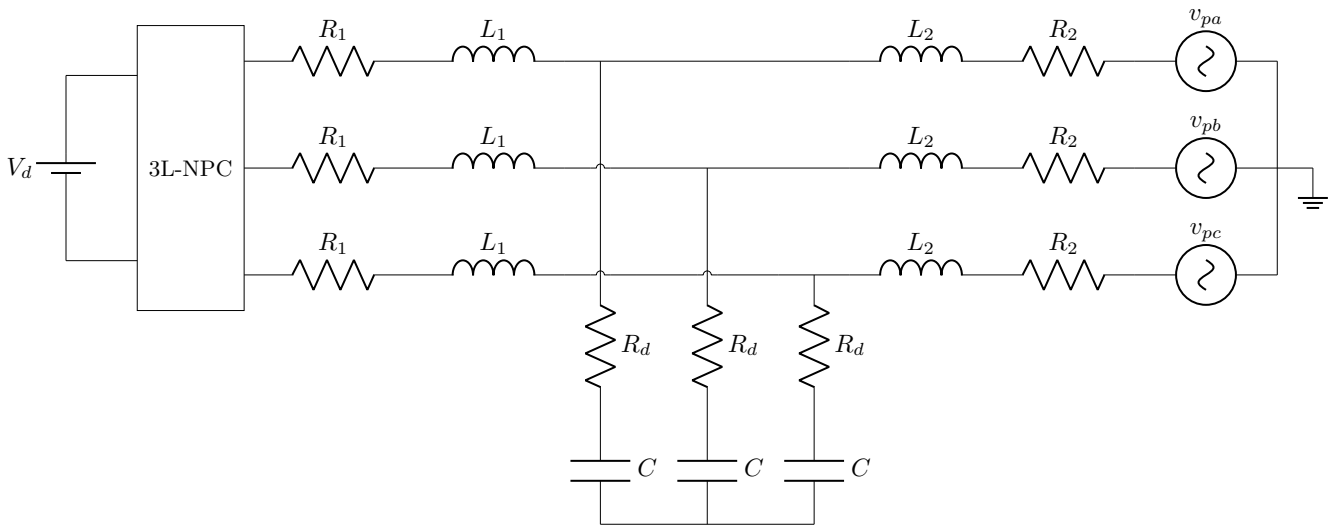

Fig. 1: Sistema trifásico modelado (Elaboración propia).

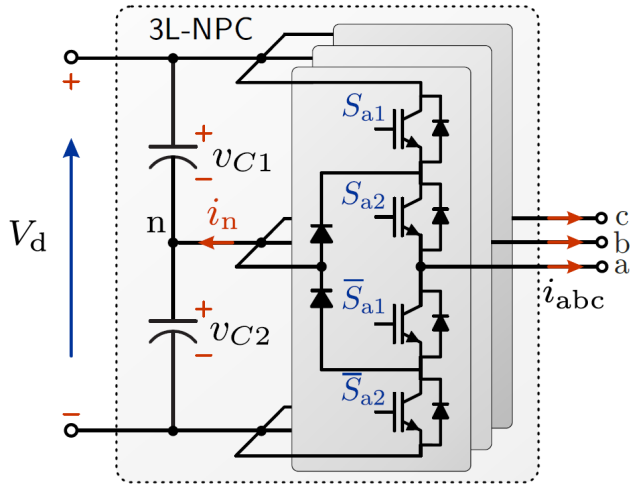

Fig. 2: Convertidor 3L NPC [1].

3.1. Obtención del modelo en variables de estado

Para facilitar el análisis y el diseño del control, se modela el equivalente monofásico en coordenadas $\alpha\beta$ del sistema, presentado en la Fig. 3, en el cual la tensión alterna v_s corresponde a la generada por el convertidor.

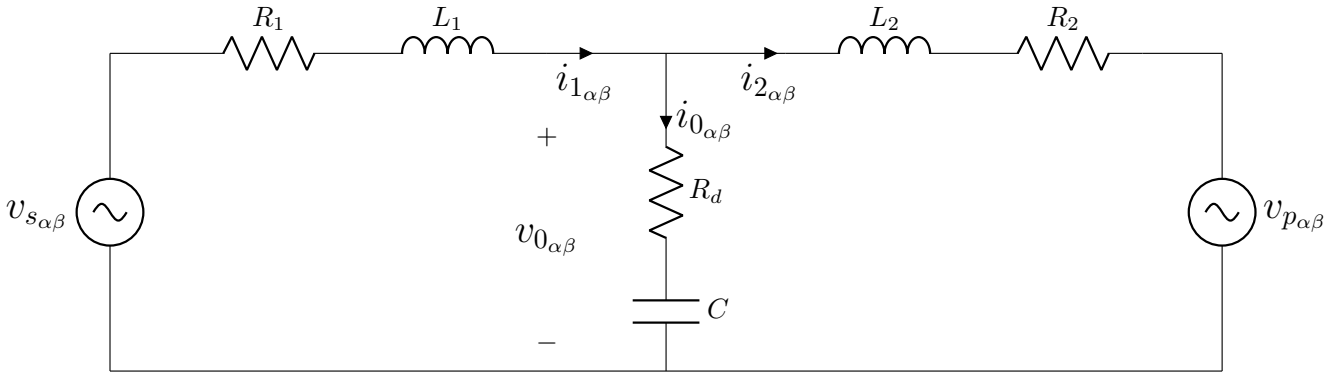

Fig. 3: Equivalente monofásico del sistema modelado (Elaboración propia).

Esta representación permite aplicar leyes de Kirchhoff de voltaje y corriente (LVK y LCK) de manera directa y obtener un modelo dinámico del sistema en variables de estado, el cual constituye la base para la implementación del controlador predictivo basado en modelo (MPC).

Las ecuaciones diferencias resultantes de aplicar los LVK y LCK son:

$$v_{s_{\alpha\beta}} = v_{0_{\alpha\beta}} + R_1 i_{1_{\alpha\beta}} + L_1 \frac{di_{1_{\alpha\beta}}}{dt}, \tag{1}$$

$$v_{0_{\alpha\beta}} = v_{p_{\alpha\beta}} + R_2 i_{2_{\alpha\beta}} + L_2 \frac{di_{2_{\alpha\beta}}}{dt}, \tag{2}$$

$$i_{0_{\alpha\beta}} = i_{1_{\alpha\beta}} - i_{2_{\alpha\beta}}. \tag{3}$$

Por otra parte, para modelar el filtro LCL con resistencia de amortiguamiento R_d, se descompone la tensión en el nodo intermedio como la suma de la caída en el capacitor y en R_d:

$$v_{0_{\alpha\beta}} = v_{c_{\alpha\beta}} + v_{rd_{\alpha\beta}}, \tag{4}$$

donde

$$v_{rd_{\alpha\beta}} = R_d i_{0_{\alpha\beta}}, \qquad i_{0_{\alpha\beta}} = C \frac{dv_{c_{\alpha\beta}}}{dt} \tag{5}$$

De esta forma, la tensión $v_{0_{\alpha\beta}}$ puede escribirse como:

$$v_{0_{\alpha\beta}} = v_{c_{\alpha\beta}} + R_d C \frac{dv_{c_{\alpha\beta}}}{dt}. \tag{6}$$

Reordenando, el sistema dinámico queda expresado como:

$$L_1 \frac{di_{1_{\alpha\beta}}}{dt} + R_d C \frac{dv_{c_{\alpha\beta}}}{dt} = -R_1 i_{1_{\alpha\beta}} - v_{c_{\alpha\beta}} + v_{s_{\alpha\beta}} \tag{7}$$

$$L_2 \frac{di_{2_{\alpha\beta}}}{dt} - R_d C \frac{dv_{c_{\alpha\beta}}}{dt} = -R_2 i_{2_{\alpha\beta}} + v_{c_{\alpha\beta}} - v_{p_{\alpha\beta}} \tag{8}$$

$$C \frac{dv_{c_{\alpha\beta}}}{dt} = i_{1_{\alpha\beta}} - i_{2_{\alpha\beta}} \tag{9}$$

Por otro lado, la tensión de salida del convertidor $v_{s_{\alpha\beta}}$ puede expresarse en función de la tensión continua V_d y de las señales de control u_a, u_b y u_c:

$$v_{s_{\alpha\beta}} = \frac{V_d}{2} u_{\alpha\beta} \tag{10}$$

$$= \frac{V_d}{2} \begin{bmatrix} 1 & -\frac{1}{2} & -\frac{1}{2} \\ 0 & \frac{\sqrt{3}}{2} & -\frac{\sqrt{3}}{2} \end{bmatrix} \begin{bmatrix} u_a \\ u_b \\ u_c \end{bmatrix} \tag{11}$$

En forma matricial, el conjunto de ecuaciones del sistema puede escribirse como:

$$M\dot{x} = Nx + Ou + P_e V_p \tag{12}$$

donde:

$$x = \begin{bmatrix} i_{1\alpha} \\ i_{1\beta} \\ i_{2\alpha} \\ i_{2\beta} \\ v_{c\alpha} \\ v_{c\beta} \end{bmatrix}, \quad M = \begin{bmatrix} L_1 & 0 & 0 & 0 & R_dC & 0 \\ 0 & L_1 & 0 & 0 & 0 & R_dC \\ 0 & 0 & L_2 & 0 & -R_dC & 0 \\ 0 & 0 & 0 & L_2 & 0 & -R_dC \\ 0 & 0 & 0 & 0 & C & 0 \\ 0 & 0 & 0 & 0 & 0 & C \end{bmatrix}, \quad N = \begin{bmatrix} -R_1 & 0 & 0 & 0 & -1 & 0 \\ 0 & -R_1 & 0 & 0 & 0 & -1 \\ 0 & 0 & -R_2 & 0 & 1 & 0 \\ 0 & 0 & 0 & -R_2 & 0 & 1 \\ 1 & 0 & -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & -1 & 0 & 0 \end{bmatrix}$$

$$u = \begin{bmatrix} u_a \\ u_b \\ u_c \end{bmatrix}, \quad O = \begin{bmatrix} \frac{V_d}{2} & 0 \\ 0 & \frac{V_d}{2} \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} 1 & -\frac{1}{2} & -\frac{1}{2} \\ 0 & \frac{\sqrt{3}}{2} & -\frac{\sqrt{3}}{2} \end{bmatrix}, \quad V_p = \begin{bmatrix} v_{p\alpha} \\ v_{p\beta} \end{bmatrix}, \quad P_e = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ -1 & 0 \\ 0 & -1 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}$$

Finalmente, la ecuación (12) se reescribe en forma de variables de estado como:

$$\dot{x} = Ax + Bu + DV_p \tag{13}$$

donde las matrices del sistema se obtienen como:

$$A = M^{-1}N, \quad B = M^{-1}O, \quad D = M^{-1}P_e$$

La representación en tiempo continuo obtenida en (13) constituye la base del modelo dinámico del sistema. Sin embargo, dado que la implementación del control se realiza de manera digital, es necesario llevar este modelo al dominio discreto.

3.2. Discretización del sistema

La discretización de las matrices A y B ha sido ampliamente documentada en la literatura. En particular, una discretización exacta del modelo continuo se obtiene bajo el supuesto de retención de orden cero (ZOH) de la entrada durante cada período de muestreo. Siguiendo la explicación del curso de la Universidad de Newcastle [5], se cumple que:

$$A_d = e^{AT_s}, \qquad B_d = A^{-1}(A_d - I)B, \tag{14}$$

siempre que A sea invertible, utilizando la identidad

$$\int_0^{T_s} e^{A\tau}\, d\tau = A^{-1}(A_d - I). \tag{15}$$

Donde I es la matriz identidad, y T_s corresponde al período de muestreo.

Por otro lado, la discretización de la perturbación externa es un aspecto menos abordado en la literatura. Una primera opción consiste en tratarla análogamente a la entrada U, obteniendo entonces la discretización exacta

$$D_d = A^{-1}\left(A_d - I\right) D. \tag{16}$$

Sin embargo, al implementar esta formulación no se obtuvieron resultados satisfactorios en las simulaciones, por lo que se recurrió a una aproximación alternativa de menor complejidad. Esta aproximación se basa en la discretización de Euler hacia adelante de la ecuación diferencial de estado [6]:

$$\dot{x}(t) \approx \frac{x[l+1] - x[l]}{T_s}. \tag{17}$$

Reemplazando en la ecuación de estado continua (13) y reorganizando, se obtiene:

$$\begin{aligned} x[l+1] &= x[l] + T_s\big(Ax[l] + Bu[l] + DV_p[l+1]\big) && (18)\\ &= (I + T_sA)x[l] + T_sBu[l] + T_sDV_p[l+1]. && (19) \end{aligned}$$

De esta forma, el modelo discreto queda expresado como

$$x[l+1] = A_d x[l] + B_d u[l] + D_d V_p[l+1], \tag{20}$$

donde, bajo esta aproximación de primer orden,

$$\begin{aligned} A_d &= I + T_sA,\\ B_d &= T_sB,\\ D_d &= T_sD. \end{aligned}$$

En conclusión, se utilizaron expresiones exactas para la discretización de A y B (ecuaciones (21)–(22)), mientras que para la matriz de perturbación se adoptó la aproximación de primer orden (23).

$$\begin{aligned} A_d &= e^{AT_s}, && (21)\\ B_d &= A^{-1}\left(A_d - I\right)B, && (22)\\ D_d &= T_sD. && (23) \end{aligned}$$

Cabe destacar que, para este problema modelado, el período de muestreo T_s corresponde al período de conmutación del modulador SVPWM, ya que la actualización de las variables de control se realiza en cada ciclo de conmutación.

Una vez obtenido el modelo en tiempo discreto, es posible plantear formalmente la estrategia de control predictivo, la cual se formula como un problema de optimización cuadrática.

4. Formulación del control predictivo basado en modelo

4.1. Planteamiento general

De acuerdo a [3], el *Model Predictive Control* (MPC) puede establecerse como un problema de optimización planteado de la siguiente manera:

$$\begin{aligned} \mathbb{P}_N(x(k)) : V_N^{\text{opt}}(x(k)) &= \min_{U(k)} \{V_N(x(k), U(k))\} \\ \text{Sujeto a:} \quad x(l+1) &= Ax(l) + Bu(l), \\ u(l) &\in \mathbb{R}^{3N}. \end{aligned} \tag{24}$$

Donde la función de costo $V_N(x(k), U(k))$ está definida como:

$$V_N(x(k), U(k)) = \sum_{l=0}^{N-1} x^\top(l) Q x(l) + u^\top(l) R u(l) + x^\top(N) Q x(N), \tag{25}$$

la cual puede reorganizarse en forma matricial:

$$V_N(x(k), U(k)) = x^\top(0) Q x(0) + X_{[1:N]}^\top Q_N X_{[1:N]} + U(k)^\top R_N U(k), \tag{26}$$

Donde $X_{[1:N]}$ representa los estados de predicción en el horizonte de control, Q y R son matrices de ponderación asociadas a los costos de los estados y entradas. Por otro lado, Q_N y R_N son matrices diagonales compuestas por Q y R respectivamente, es decir:

$$Q_N = \begin{bmatrix} Q & 0_{n\times n} & \cdots & 0_{n\times n} \\ 0_{n\times n} & Q & \cdots & 0_{n\times n} \\ \vdots & \vdots & \ddots & \vdots \\ 0_{n\times n} & 0_{n\times n} & \cdots & Q \end{bmatrix}, \quad R_N = \begin{bmatrix} R & 0_{m\times m} & \cdots & 0_{m\times m} \\ 0_{m\times m} & R & \cdots & 0_{m\times m} \\ \vdots & \vdots & \ddots & \vdots \\ 0_{m\times m} & 0_{m\times m} & \cdots & R \end{bmatrix}$$

4.2. Predicción de estados sin perturbaciones

La evolución de los estados en el horizonte de predicción, según [3], puede expresarse como

$$X_{[1:N]} = \Lambda x(0) + \Phi U(k), \tag{27}$$

Donde:

$$X_{[1:N]} = \begin{bmatrix} x(1) \\ x(2) \\ \vdots \\ x(N) \end{bmatrix}, \quad U(k) = \begin{bmatrix} u(0) \\ u(1) \\ \vdots \\ u(N-1) \end{bmatrix}, \quad \Lambda = \begin{bmatrix} A \\ A^2 \\ \vdots \\ A^N \end{bmatrix}, \quad \Phi = \begin{bmatrix} B & 0_{n\times m} & \cdots & 0_{n\times m} \\ AB & B & \cdots & 0_{n\times m} \\ \vdots & \vdots & \ddots & \vdots \\ A^{N-1}B & A^{N-2}B & \cdots & B \end{bmatrix}$$

Con:

$$X_{[1:N]} \in \mathbb{R}^{Nn}, \quad \Lambda \in \mathbb{R}^{Nn\times n}, \quad \Phi \in \mathbb{R}^{Nn\times Nm}, \quad U(k) \in \mathbb{R}^{Nm}$$

Siendo n y m las dimensiones de $x(0)$ y $u(0)$ respectivamente, mientras que los valores de A y B se desprenden del cálculo:

$$x(l+1) = Ax(l) + Bu(l) \tag{28}$$

4.3. Extensión con perturbaciones

En el caso bajo estudio, el sistema depende además de una perturbación V_p, modelada en la dinámica como

$$x(l+1) = Ax(l) + Bu(l) + Dv_p(l+1). \tag{29}$$

Al extender la relación (27), la predicción de estados en presencia de perturbaciones se expresa como

$$X_{[1:N]} = \Lambda x(0) + \Phi U(k) + \Gamma V_{p[1:N]}, \tag{30}$$

donde $V_{p[1:N]}$ agrupa la secuencia de perturbaciones y Γ es una matriz bloque-diagonal definida como

$$\Gamma = \begin{bmatrix} D & 0 & \cdots & 0 \\ 0 & D & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & D \end{bmatrix}. \tag{31}$$

4.3.1. Modelado temporal de la perturbación

Para completar la formulación, es necesario describir explícitamente cómo se construye el vector $V_{p[1:N]}$ a lo largo del horizonte de predicción.

Asumiendo que la tensión $v_p(t)$ es una señal cosenoidal de la forma

$$v_p(t) = V_p \cos(\omega t + \theta_p), \tag{32}$$

se obtiene que, para $t = 0$,

$$v_p(l) = V_p \cos(\theta_p), \tag{33}$$
$$v_p(l+1) = V_p \cos(\theta_p + \omega\Delta t). \tag{34}$$

Al expresar esta perturbación en el marco $\alpha\beta$, se obtiene:

$$v_{p\alpha}(l) + jv_{p\beta}(l) = V_p \cos(\theta_p) + jV_p \sin(\theta_p), \tag{35}$$
$$v_{p\alpha}(l+1) + jv_{p\beta}(l+1) = V_p \cos(\theta_p + \omega\Delta t) + jV_p \sin(\theta_p + \omega\Delta t). \tag{36}$$

Aplicando identidades trigonométricas se deduce la relación de recurrencia

$$\begin{bmatrix} v_{p\alpha}(l+1) \\ v_{p\beta}(l+1) \end{bmatrix} = \begin{bmatrix} \cos(\omega\Delta t) & -\sin(\omega\Delta t) \\ \sin(\omega\Delta t) & \cos(\omega\Delta t) \end{bmatrix} \begin{bmatrix} v_{p\alpha}(l) \\ v_{p\beta}(l) \end{bmatrix}, \tag{37}$$

lo cual muestra que la perturbación evoluciona como una rotación en el plano $\alpha\beta$, con ángulo de avance $\omega\Delta t$ en cada paso discreto.

De manera más general, si se consideran N instantes futuros, la secuencia de perturbaciones se expresa como

$$V_{p[1:N]} = C_{nt} \begin{bmatrix} v_{p\alpha}(0) \\ v_{p\beta}(0) \end{bmatrix}, \tag{38}$$

donde

$$V_{p[1:N]} = \begin{bmatrix} v_{p\alpha}(1) \\ v_{p\beta}(1) \\ v_{p\alpha}(2) \\ v_{p\beta}(2) \\ \vdots \\ v_{p\alpha}(N) \\ v_{p\beta}(N) \end{bmatrix}, \quad C_{nt} = \begin{bmatrix} \cos(\omega\Delta t) & -\sin(\omega\Delta t) \\ \sin(\omega\Delta t) & \cos(\omega\Delta t) \\ \cos(2\omega\Delta t) & -\sin(2\omega\Delta t) \\ \sin(2\omega\Delta t) & \cos(2\omega\Delta t) \\ \vdots & \vdots \\ \cos(N\omega\Delta t) & -\sin(N\omega\Delta t) \\ \sin(N\omega\Delta t) & \cos(N\omega\Delta t) \end{bmatrix}.$$

4.4. Definición de la función de costo

Una vez modelada la evolución de los estados y de las perturbaciones a lo largo del horizonte de predicción, el siguiente paso consiste en establecer un criterio que permita evaluar la calidad de las acciones de control. Para ello, en el marco del MPC, se define una función de costo cuadrática que penaliza simultáneamente el error de seguimiento respecto a una referencia de estados X_{ref} y a una referencia de entradas U_{ref}.

$$V_N(x(k), U(k)) = (X_{[1:N]} - X_{ref})^\top Q_N (X_{[1:N]} - X_{ref}) + (U(k) - U_{ref})^\top R_N (U(k) - U_{ref}), \tag{39}$$

donde X_{ref} y U_{ref} corresponden a las trayectorias de referencia de estados y entradas, respectivamente.

Al reemplazar $X_{[1:N]}$ en función de la ecuación modificada (30), la función de costo puede reescribirse únicamente en función de la secuencia de entradas $U(k)$. De esta manera, el problema de MPC se transforma en la resolución de un problema de optimización cuadrático estándar.

$$\begin{aligned} V_N(x(k), U(k)) = {} & (\Lambda x(0) + \Phi U(k) + \Gamma V_{p[1:N]} - X_{ref})^T Q_N (\Lambda x(0) + \Phi U(k) + \Gamma V_{p[1:N]} - X_{ref}) \\ & + (U(k) - U_{ref})^T R_N (U(k) - U_{ref}) \end{aligned} \tag{40}$$

Donde las matrices Λ, Φ y Γ se construyen con las matrices discretas A_d, B_d y D_d.
Reordenando, la función de costo queda escrita como:

$$\begin{aligned} V_N(x(k),U(k)) = {} & 2U(k)^T\Phi^T Q_N(\Lambda x(0) + \Gamma V_{p[1:N]} - X_{ref}) + U(k)^T\Phi^T Q_N \Phi U(k) \\ & + U(k)^T R_N U(k) - 2U(k)^T R_N U_{ref} + U_{ref}^T R_N U_{ref} \\ & + (\Lambda x(0) + \Gamma V_{p[1:N]} - X_{ref})^T Q_N (\Lambda x(0) + \Gamma V_{p[1:N]} - X_{ref}) \end{aligned} \tag{41}$$

Simplificando, se obtiene la forma final de la función de costo:

$$V_N(x(k),U(k)) = U(k)^T W U(k) + U(k)^T F + \nu(x(0)) \tag{42}$$

Donde:

$$\begin{aligned} W &= \Phi^T Q_N \Phi + R_N \\ F &= 2\Phi^T Q_N(\Lambda x(0) + \Gamma V_{p[1:N]} - X_{ref}) - 2R_N U_{ref} \\ \nu(x(0)) &= (\Lambda x(0) + \Gamma V_{p[1:N]} - X_{ref})^T Q_N (\Lambda x(0) + \Gamma V_{p[1:N]} - X_{ref}) + U_{ref}^T R_N U_{ref} \end{aligned}$$

Finalmente, el problema de optimización cuadrática a resolver para determinar la secuencia de control óptima se formula como:

$$\begin{aligned} \min_{U(k)} \quad & \tfrac{1}{2} U(k)^\top H U(k) + f^\top U(k), \\ \text{s.a.} \quad & -1{,}15 \le u_i(l) \le 1{,}15, \\ & i = 1,2,3, \quad l = 1,\ldots,N, \end{aligned} \tag{43}$$

donde los límites de $u_i(l)$ corresponden al factor de modulación máximo permitido por SVPWM en el convertidor 3L-NPC.

4.5. Cálculo de referencias

En el enfoque *Grid Following*, el punto de partida son las potencias activa y reactiva que se desea entregar al SEP. En coordenadas $\alpha\beta$, estas se expresan a partir de la transformación de Clarke aplicada a las definiciones clásicas de potencia instantánea [7]:

$$p = \frac{3}{2}\left(v_\alpha i_\alpha + v_\beta i_\beta\right), \qquad q = \frac{3}{2}\left(v_\beta i_\alpha - v_\alpha i_\beta\right). \tag{44}$$

Considerando entonces potencias de referencia p^* y q^*, y la medición de la tensión en el SEP, se puede establecer la relación entre estas y la referencia de la corriente en el lado de la red como:

$$\begin{bmatrix} p^* \\ q^* \end{bmatrix} = \frac{3}{2} \begin{bmatrix} v_{p\alpha} & v_{p\beta} \\ v_{p\beta} & -v_{p\alpha} \end{bmatrix} \begin{bmatrix} i_{2\alpha}^* \\ i_{2\beta}^* \end{bmatrix}. \tag{45}$$

De aquí se despejan las componentes de la corriente de referencia en $\alpha\beta$:

$$\begin{bmatrix} i^*_{2\alpha} \\ i^*_{2\beta} \end{bmatrix} = \frac{2}{3} \begin{bmatrix} v_{p\alpha} & v_{p\beta} \\ v_{p\beta} & -v_{p\alpha} \end{bmatrix}^{-1} \begin{bmatrix} p^* \\ q^* \end{bmatrix}. \tag{46}$$

Sea $\mathbf{i}^*_2(l) = \begin{bmatrix} i^*_{2\alpha}(l) \\ i^*_{2\beta}(l) \end{bmatrix}$ el vector de referencia en el instante discreto l. A partir de la referencia calculada en el instante de medida, las referencias futuras se obtienen aplicando la misma rotación en el plano $\alpha\beta$ que se utilizó para la perturbación del SEP en (37):

$$\mathbf{i}^*_2(l+1) = \underbrace{\begin{bmatrix} \cos(\omega\Delta t) & -\sin(\omega\Delta t) \\ \sin(\omega\Delta t) & \cos(\omega\Delta t) \end{bmatrix}}_{:=\mathbf{R}(\omega\Delta t)} \mathbf{i}^*_2(l), \tag{47}$$

donde Δt es el período de muestreo y ω la frecuencia angular de la red. De este modo se generan las referencias $\mathbf{i}^*_2(l)$ para $l = 1, \ldots, N$.

A continuación se obtiene la referencia de la tensión en v_0. En notación fasorial se tiene

$$V^*_0 = j\omega L_2 I^*_2 + V_p, \tag{48}$$

es decir, en forma compleja

$$V^*_{0\alpha} + jV^*_{0\beta} = j\omega L_2 \left(I^*_{2\alpha} + jI^*_{2\beta}\right) + (V_{p\alpha} + jV_{p\beta}). \tag{49}$$

Descomponiendo en componentes reales $\alpha\beta$ se obtienen las referencias instantáneas de la tensión del nodo medio (para el instante discreto l):

$$v^*_{0\alpha}(l) = -\omega L_2\, i^*_{2\beta}(l) + v_{p\alpha}(l), \qquad v^*_{0\beta}(l) = \omega L_2\, i^*_{2\alpha}(l) + v_{p\beta}(l). \tag{50}$$

Por otro lado, la corriente que circula por la rama del capacitor en el nodo común incluye el efecto de la resistencia de amortiguamiento R_d. En el dominio fasorial, la impedancia equivalente de la rama es:

$$Z_c = R_d + \frac{1}{j\omega C}, \tag{51}$$

por lo que la corriente se obtiene como

$$I_0 = \frac{V_0}{Z_c} = \frac{j\omega C}{1 + j\omega R_d C}\, V_0 \tag{52}$$

En coordenadas $\alpha\beta$, esta corriente define la referencia

$$\mathbf{i}^*_0(l) = \begin{bmatrix} i^*_{0\alpha}(l) \\ i^*_{0\beta}(l) \end{bmatrix}, \tag{53}$$

que se suma directamente a la corriente hacia la red:

$$\mathbf{i}^*_1(l) = \mathbf{i}^*_2(l) + \mathbf{i}^*_0(l). \tag{54}$$

Finalmente, la tensión sobre el capacitor puede calcularse a partir de la descomposición

$$V_0 = V_c + V_{Rd}, \qquad V_{Rd} = R_d I_0, \tag{55}$$

de modo que la referencia de la tensión del capacitor resulta

$$V_c^* = V_0^* - R_d I_0^*. \tag{56}$$

A partir de $\mathbf{i}_1^*(l)$ y de $V_0^*(l)$ se obtiene la tensión que debe generar el convertidor. En dominio fasorial:

$$V_s^* = j\omega L_1 I_1^* + R_1 I_1^* + V_0^*, \tag{57}$$

y en componentes $\alpha\beta$ para el instante l:

$$v_{s\alpha}^*(l) = -\omega L_1\, i_{1\beta}^*(l) + R_1\, i_{1\alpha}^*(l) + v_{0\alpha}^*(l), \tag{58}$$

$$v_{s\beta}^*(l) = \omega L_1\, i_{1\alpha}^*(l) + R_1\, i_{1\beta}^*(l) + v_{0\beta}^*(l). \tag{59}$$

Dado que el convertidor genera la tensión de salida mediante modulación trifásica, la relación entre la señal de control $u = [u_a\ u_b\ u_c]^\mathsf{T}$ y las componentes $\alpha\beta$ de la tensión es

$$\begin{bmatrix} v_{s\alpha} \\ v_{s\beta} \end{bmatrix} = \frac{V_d}{2} \begin{bmatrix} 1 & -\frac{1}{2} & -\frac{1}{2} \\ 0 & \frac{\sqrt{3}}{2} & -\frac{\sqrt{3}}{2} \end{bmatrix} \begin{bmatrix} u_a \\ u_b \\ u_c \end{bmatrix}. \tag{60}$$

Por tanto, la referencia de actuación se obtiene como:

$$\begin{bmatrix} u_a^* \\ u_b^* \\ u_c^* \end{bmatrix} = \frac{2}{V_d} \begin{bmatrix} 1 & 0 \\ -\frac{1}{2} & \frac{\sqrt{3}}{2} \\ -\frac{1}{2} & -\frac{\sqrt{3}}{2} \end{bmatrix} \begin{bmatrix} v_{s\alpha}^* \\ v_{s\beta}^* \end{bmatrix}. \tag{61}$$

Finalmente, al agrupar las referencias de las variables y de la actuación en los vectores:

$$x^*(l) = \begin{bmatrix} i_{1\alpha}^*(l) \\ i_{1\beta}^*(l) \\ i_{2\alpha}^*(l) \\ i_{2\beta}^*(l) \\ v_{c\alpha}^*(l) \\ v_{c\beta}^*(l) \end{bmatrix}, \quad u^*(l) = \begin{bmatrix} u_a^*(l) \\ u_b^*(l) \\ u_c^*(l) \end{bmatrix} \tag{62}$$

Se puede establecer las referencias de estado X_{ref} y de entradas U_{ref} como:

$$X_{ref} = \begin{bmatrix} x^*(1) \\ x^*(2) \\ \vdots \\ x^*(N) \end{bmatrix}, \quad U_{ref} = \begin{bmatrix} u^*(0) \\ u^*(1) \\ \vdots \\ u^*(N-1) \end{bmatrix} \tag{63}$$

A partir de este desarrollo se obtiene la información necesaria para formular y resolver el problema de optimización predictiva. En la siguiente sección se describen los algoritmos considerados para su resolución y se analiza su desempeño en términos de eficiencia computacional y calidad de control.

5. Algoritmos de optimización considerados

Para resolver el problema cuadrático planteado en la sección anterior, se requiere implementar un solucionador eficiente que pueda ejecutarse en tiempo real sobre una plataforma de control digital. A continuación, se describen dos algoritmos de optimización QP de pequeña escala: Active-Set y Frank-Wolfe, los cuales ofrecen enfoques distintos para abordar este tipo de problema. Ambos algoritmos son evaluados en función de su eficiencia computacional, exactitud de la solución y compatibilidad con sistemas embebidos.

5.1. Active–Set

El método de conjunto activo (*active–set method*) es un procedimiento iterativo utilizado para resolver problemas de programación cuadrática con restricciones de desigualdad. La idea central consiste en mantener y actualizar un *conjunto activo* de restricciones, es decir, aquellas que se satisfacen como igualdades en la solución candidata [8].

En cada iteración, el algoritmo resuelve un subproblema cuadrático restringido únicamente por las restricciones activas, actualizando luego dicho conjunto de acuerdo con las condiciones de optimalidad de Karush–Kuhn–Tucker (KKT). La estructura del algoritmo es la siguiente [8]:

1. Se inicia desde una solución factible inicial y se define un conjunto activo de restricciones que se tratan como igualdades.

2. En cada iteración, se resuelve el subproblema cuadrático solo con aquellas restricciones activas, y se calcula la dirección de búsqueda.

3. Se actualizan los multiplicadores de Lagrange. Si alguno de ellos resulta menor a cero, la restricción asociada se elimina del conjunto activo.

4. Se verifica si alguna restricción inactiva se vuelve violada por la dirección de búsqueda; en ese caso, se añade al conjunto activo.

5. Se repite hasta que no hay cambios en el conjunto activo y se cumplen las condiciones KKT, lo cual implica convergencia al óptimo local.

Este método aprovecha la estructura del problema reduciendo el espacio de búsqueda en cada iteración, ya que solo considera las restricciones activas. En la práctica, el algoritmo converge en un número finito de pasos hacia una solución que satisface las condiciones KKT del problema cuadrático.

Condiciones KKT

Las condiciones de Karush–Kuhn–Tucker (KKT) son un criterio de optimalidad que generaliza las condiciones de primer orden en presencia de restricciones. Para un problema cuadrático de la forma

$$\begin{aligned} \min_{x} \quad & \tfrac{1}{2} x^\top H x + c^\top x, \\ \text{s.a.} \quad & a_i^\top x \geq b_i, \quad i = 1, \ldots, m, \end{aligned}$$

las condiciones KKT establecen que en el óptimo $x^\star$ deben cumplirse:

1. **Factibilidad primal:** $a_i^\top x^\star \geq b_i \;\; \forall i$.
2. **Factibilidad dual:** existen multiplicadores $\lambda_i \geq 0$ asociados a cada restricción.
3. **Condición de estacionariedad:**

$$H x^\star + c - \sum_{i=1}^{m} \lambda_i a_i = 0. \tag{64}$$

4. **Condición de complementariedad:**

$$\lambda_i \left(a_i^\top x^\star - b_i \right) = 0, \quad \forall i. \tag{65}$$

Estas condiciones garantizan que la solución encontrada es un punto estacionario que respeta las restricciones activas e inactivas.

Resolución del subproblema

En cada iteración, dado un conjunto activo $\mathcal{W}_k$, se resuelve un *subproblema cuadrático restringido*:

$$\begin{aligned} \min_{p} \quad & \tfrac{1}{2} p^\top H p + \nabla f(x_k)^\top p, \\ \text{s.a.} \quad & a_i^\top p = 0, \quad i \in \mathcal{W}_k. \end{aligned}$$

Esto equivale a buscar la dirección p en el subespacio tangente a las restricciones activas. El sistema lineal resultante se resuelve típicamente mediante la factorización de KKT [8]:

$$\begin{bmatrix} H & A_{\mathcal{W}}^\top \\ A_{\mathcal{W}} & 0 \end{bmatrix} \begin{bmatrix} p \\ \lambda \end{bmatrix} = \begin{bmatrix} -\nabla f(x_k) \\ 0 \end{bmatrix} \tag{66}$$

donde $A_{\mathcal{W}}$ contiene las restricciones activas en la iteración k.

5.1.1. Implementación propuesta del método Active–Set

En el contexto del problema de control predictivo (43), las restricciones de control ya son conocidas y acotadas de forma simétrica en el intervalo [−1,15, 1,15]. Este tipo de restricciones corresponde a un conjunto factible de tipo *box*, es decir, una caja simétrica en $\mathbb{R}^n$ de la forma:

$$\mathcal{P} = [-M, M]^n,$$

donde en este caso $M = 1{,}15$.

Gracias a esta estructura, se puede simplificar el esquema clásico del método de conjunto activo: no es necesario calcular un tamaño de paso α_k, sino que la solución U_k obtenida en cada iteración se proyecta directamente dentro de la región factible, saturando los valores que exceden los límites permitidos.

De esta manera, las restricciones se incorporan gradualmente al conjunto activo en el momento en que alguna componente de U_k viola el intervalo de factibilidad.

Descripción del algoritmo propuesto.

Se implementó en Matlab la función `ActiveSet_Sim(H, f, Mm)`, cuya entrada son la matriz H y el vector f de la función de costo, además de la magnitud del *box* M_m que define el dominio de las variables u_i. En este trabajo se emplea $M_m = 1{,}15$, aunque el algoritmo permite variar dicho valor para otros esquemas de modulación.

La salida es U, la solución óptima del problema restringido. La función entrega además el número de iteraciones que tardó en encontrar la solución óptima.

1. Inicialización:

- Se calcula una solución inicial sin restricciones $U_0 = -H^{-1}f$ y se proyecta al *box* definido por M_m.
- Se inicializa la matriz A_{sub} y el vector b_{sub} para la resolución de subproblemas cuando existan restricciones activas. A_{sub} incluye inicialmente la matriz H en su bloque superior izquierdo, mientras que b_{sub} se inicializa como un vector de ceros.
- Variables auxiliares:
  - $lu0$: número de variables (length(U)).
  - N_{ac}: número de restricciones activas.
  - $L_{\text{ras}} = lu0 + N_{\text{ac}}$: tamaño actual del subproblema.

2. Cálculo de la dirección de búsqueda p_k: El algoritmo itera hasta que se cumple la condición de optimalidad. En cada iteración, se determina la dirección de movimiento p_k:

- **Primera iteración:**

- Si no hay restricciones activas, p_k se obtiene como la diferencia entre la solución sin restricciones y su proyección dentro del *box*.
- Si todos los valores de U_k están dentro del *box* y p_k es suficientemente pequeño ($\|p_k\| < tol$): el algoritmo termina.
- En caso contrario, se actualiza $U_k = U_{k-1} + p_k$ y se revisan las variables que violan el *box*. Si se detecta alguna violación, se agrega una única restricción activa por iteración. Esta restricción se representa en A_{sub} mediante una fila y columna con un 1 en la posición de la variable correspondiente, indicando que su movimiento p_k se debe restringir, mientras que el valor efectivo que fija p_k a cero se establece en el vector b_{sub} del subproblema.

- **Iteraciones siguientes:**
  - Si existen restricciones activas $A_{\mathcal{W}}$, se calcula R_{as} resolviendo el subproblema cuadrático restringido:
$$\begin{bmatrix} H & A_{\mathcal{W}}^{\top} \\ A_{\mathcal{W}} & 0 \end{bmatrix} \begin{bmatrix} p_k \\ \lambda \end{bmatrix} = \begin{bmatrix} -g_k \\ 0 \end{bmatrix}, \quad g_k = HU_{k-1} + f$$
  - El vector p_k, así como los multiplicadores de Lagrange, se extraen de la solución R_{as} obtenida del sistema lineal.

3. Actualización de variables y gestión de restricciones:

- Si $\|p_k\| > tol$, se actualiza $U = U + p_k$ y se verifica si alguna variable supera los límites del *box*. Si es así, se agrega la restricción correspondiente a A_{sub} y se incrementa N_{ac}.
- Si p_k es suficientemente pequeño ($\|p_k\| < tol$):
  - Se verifica la factibilidad de los multiplicadores de Lagrange λ asociados a las restricciones activas.
  - Si todos los $\lambda \geq 0$, el algoritmo termina y se entrega U como solución óptima.
  - Si existe alguna $\lambda < 0$, se elimina la restricción asociada al λ más negativo, actualizando A_{sub} y decrementando N_{ac}.

4. Gestión de la matriz de restricciones A_{sub}:

- Cada restricción activa se incorpora en A_{sub} agregando una fila y columna que fija a cero la componente de p_k correspondiente a la variable violada.
- Para remover restricciones, se copia la última restricción activa sobre la fila y columna de la restricción que se desea eliminar, asegurando que no queden huecos intermedios en A_{sub}. A continuación, se ponen en cero los valores de la fila y columna correspondientes a la última restricción, que ahora ya ha sido "movida" a la posición de la restricción eliminada. Este procedimiento preserva la consistencia y la estructura del subproblema mientras se reduce el contador de restricciones activas.

5. Consideraciones finales:

- Se define una tolerancia *tol* para considerar que p_k es efectivamente cero. Esto evita detener el algoritmo debido a errores numéricos.
- La función está diseñada para manejar problemas con cualquier número de variables N, donde $lu0 = 3N$ para el caso particular de este problema.
- El algoritmo asegura que la solución siempre cumple con las restricciones de caja y optimiza la función de costo cuadrática.

A continuación se presenta el código que aplica este algoritmo en Matlab.

```
function [U, iter] = ActiveSet_Sim(H, f, Mm)
    max_iter = 100;
    tol = 1e-4;
    iter = 0;

    U = max(min(-H\f, Mm), -Mm);
    lu0 = size(U,1); % 3N


    %El uso de 2*lu0 se debe a que se suma el largo de U mas el numero de
       multiplicadores de lagrange

    R_as = ones(2*lu0, 1); % Resultado subproblema
    A_sub = [H, zeros(lu0,lu0); zeros(lu0,2*lu0)];
    b_sub = zeros(2*lu0, 1);
    Nac = 0;

    for k = 1:max_iter
        iter = iter + 1;
        Lras = lu0 + Nac;

        % Resolver sub sistema

        if Nac == 0
            pk = -H\f - U;
        else
             % Calcular gradiente
            gk = H*U + f;

            % Se crea la matriz del subproblema:
            % [H, A' ; A, 0] * [x;Lambda] = [-gk, 0]

            b_sub(1:lu0) = -gk;

            R_as(1:Lras) = A_sub(1:Lras,1:Lras)\b_sub(1:Lras);


            % Separar solucion
```

```
            pk = R_as(1:lu0);
        end


        % Evaluar condicion de optimalidad
        if all(abs(pk) < tol)
            if all(R_as(lu0 + 1:end) >= 0)
                break;
            else
                [~, idx] = min(R_as(lu0 + 1:end));

                A_sub(lu0+idx,1:lu0) = A_sub(lu0+Nac,1:lu0);
                A_sub(1:lu0, lu0+idx) = A_sub(1:lu0, lu0+Nac);
                A_sub(lu0+Nac,1:lu0) = 0;
                A_sub(1:lu0, lu0+Nac) = 0;
                Nac = Nac -1;
            end
        else
            U = U + pk;

            rac = 0;
            for i = 1:lu0
                if (U(i) - Mm > 1e-6)
                    rac = 1;
                    Nac = Nac + 1;
                    A_sub(lu0+Nac,i)   = 1;
                    A_sub(i,lu0+Nac)   = 1;
                    break;
                elseif (-U(i) - Mm > 1e-6)
                    rac = 1;
                    Nac = Nac + 1;
                    A_sub(lu0+Nac,i)   = 1;
                    A_sub(i,lu0+Nac)   = 1;
                    break;
                end
            end


            if rac
                U = max(min(U, Mm), -Mm);
                if Nac == lu0
                    break
                end
            else
                break
            end
        end
    end
end
```

Listing 1: Función ActiveSet_Sim en Matlab

5.2. Frank-Wolfe

El algoritmo de Frank-Wolfe, también conocido como método del gradiente condicional, es un procedimiento iterativo para resolver problemas de optimización convexa diferenciable en conjuntos factibles convexos y compactos. Su principal atractivo radica en que evita el uso de proyecciones sobre el conjunto factible, sustituyéndolas por la resolución de un subproblema lineal más sencillo en cada iteración [9].

El problema general considerado es:

$$\min_{x \in \mathcal{P}} f(x),$$

donde $f : \mathbb{R}^n \to \mathbb{R}$ es una función convexa y diferenciable, y $\mathcal{P} \subset \mathbb{R}^n$ es un conjunto convexo compacto.

A diferencia de métodos como el descenso por gradiente proyectado, el algoritmo Frank-Wolfe resuelve en cada iteración un problema de optimización lineal sobre $\mathcal{P}$, lo que permite aprovechar la estructura geométrica del conjunto factible. Su esquema general puede describirse de la siguiente manera [9]:

1. Se elige un punto inicial factible $x_0 \in \mathcal{P}$.
2. En la iteración k, se calcula el gradiente $\nabla f(x_k)$.
3. Se resuelve el subproblema lineal:

   $$s_k = \arg\min_{s \in \mathcal{P}} \langle \nabla f(x_k), s \rangle,$$

   obteniendo un vértice extremo $s_k \in \mathcal{P}$ que define la dirección de descenso.
4. Se define $d_k = s_k - x_k$ y se determina el tamaño de paso $\alpha_k \in [0, 1]$ que minimiza $f(x_k + \alpha d_k)$.
5. Se actualiza la solución:

   $$x_{k+1} = x_k + \alpha_k d_k.$$

6. El proceso se repite hasta que se cumpla un criterio de parada, como que el *Frank-Wolfe gap* sea menor que una tolerancia prefijada.

5.2.1. Caso particular: conjunto factible tipo box

En este trabajo, el conjunto factible $\mathcal{P}$ corresponde a un *box* simétrico ya introducido en la Sección anterior, definido como:

$$\mathcal{P} = [-1{,}15,\ 1{,}15]^n.$$

Esto simplifica considerablemente el paso 3 del algoritmo de Frank–Wolfe. En efecto, el subproblema lineal:

$$\min_{s \in \mathcal{P}} \langle \nabla f(x_k), s \rangle$$

puede resolverse explícitamente mediante una regla de signos:

$$s_k = -1{,}15 \cdot \text{sign}(\nabla f(x_k)),$$

donde la función `sign` se aplica componente a componente. De esta manera, cada componente de s_k toma el valor $-1{,}15$ si la correspondiente derivada parcial es positiva, y $+1{,}15$ en caso contrario, seleccionando así el vértice del box que produce el mayor descenso en la dirección del gradiente.

Criterio de parada mediante el dual gap

En este trabajo se utiliza como criterio de parada el denominado *dual gap*, definido como:

$$g(x_k) = \langle \nabla f(x_k),\, x_k - s_k \rangle,$$

donde s_k es la solución del subproblema lineal en la iteración k. El operador $\langle \cdot, \cdot \rangle$ corresponde al producto interno euclidiano en $\mathbb{R}^n$, es decir:

$$\langle u, v \rangle = u^\top v.$$

En la implementación, esto equivale a la operación en MATLAB:

```
gap = grad' * (x - s);
```

El dual gap $g(x_k)$ proporciona una cota superior de la suboptimalidad de x_k; por lo tanto, el algoritmo se detiene cuando $g(x_k) \leq \varepsilon$ para una tolerancia predefinida $\varepsilon > 0$. De esta manera, se garantiza que la solución obtenida es ε-óptima con respecto al problema original.

En particular, el objetivo al definir la tolerancia del criterio de parada fue garantizar que las formas de onda de tensión y corriente obtenidas mediante el algoritmo Frank-Wolfe difirieran en menos de un 1 % respecto de las generadas por el método *Active-Set*. Tras realizar múltiples pruebas experimentales, se determinó que un valor adecuado para la tolerancia es:

$$\varepsilon = \frac{\|f\|_2}{200N^2}, \tag{67}$$

donde $\|f\|_2$ denota la norma euclidiana del vector f, definida como:

$$\|f\|_2 = \sqrt{f_1^2 + f_2^2 + \cdots + f_n^2}.$$

Esta norma entrega una medida de la magnitud global de los términos lineales del costo, permitiendo escalar la tolerancia de acuerdo con la intensidad numérica del problema de optimización. De esta manera, el parámetro ε se adapta tanto al tamaño del vector f como al horizonte de predicción N, evitando el uso de una tolerancia fija que podría resultar inadecuada para distintas configuraciones del sistema.

El factor divisor 200 fue determinado empíricamente de modo que la solución obtenida cumpliera con el objetivo planteado.

Cálculo del paso α

Una vez determinada la dirección $d = s - x$, la actualización de Frank-Wolfe se realiza mediante una combinación convexa:

$$x_{\text{next}} = (1 - \alpha)x + \alpha s = x + \alpha d, \tag{68}$$

con $\alpha \in [0, 1]$.

En el caso de una función objetivo cuadrática de la forma:

$$g(x) = \tfrac{1}{2}x^\top Hx + f^\top x, \tag{69}$$

la evaluación en el punto $x + \alpha d$ conduce a una expresión cuadrática en α:

$$g(x + \alpha d) = \tfrac{1}{2}(x + \alpha d)^\top H(x + \alpha d) + f^\top(x + \alpha d), \tag{70}$$

que puede reescribirse como:

$$g(\alpha) = a\alpha^2 + b\alpha + c, \tag{71}$$

donde:

$$a = \tfrac{1}{2}d^\top Hd, \qquad b = d^\top(Hx + f) = d^\top \nabla g(x), \qquad c = g(x). \tag{72}$$

El valor óptimo de α se obtiene como:

$$\alpha^* = -\frac{b}{2a} = -\frac{d^\top \nabla g(x)}{d^\top Hd}. \tag{73}$$

Este valor asegura la minimización de la función objetivo a lo largo de la dirección d, manteniendo la factibilidad de la solución.

Implementación en Matlab

La siguiente función en Matlab implementa el algoritmo Frank-Wolfe para el caso particular de un problema cuadrático convexo con restricciones tipo caja simétrica:

```
function [U, iter] = frankWolfeQP_GAP(H, f, max_iter, Mm)

    iter = 0;
    U = max(min(-H\f, Mm), -Mm);
    lu0 = size(U,1); % 3N, size(H,1)
    n = lu0/3;
    s = zeros(lu0,1);

    tol = norm(f,2)/((n^2) * 2e2); %Tolerancia depende del problema


    for k = 1:max_iter
        iter = iter + 1;

        % Gradiente
        grad = H * U + f;
```

```
17
18          % Vertice y direccion
19          for i = 1:lu0
20              if grad(i) > 1e-4
21                  s(i) = -Mm;    % limite inferior
22              elseif grad(i) < -1e-4
23                  s(i) =  Mm;    % limite superior
24              else
25                  s(i) = U(i);   % si grad(i) esta dentro del rango [-tol2, tol2]
26              end
27          end
28          d = s - U;
29
30          % Frank-Wolfe gap
31          gap = -grad' * d;   % grad*(x-s)
32
33          % Verificacion de convergencia
34          if gap < tol
35              break;
36          end
37
38
39          % Calculo del paso
40          alpha = gap / (d' * H * d);
41          alpha = max(0, min(1, alpha));
42
43          % Actualizacion
44          U = U + alpha * d;
45      end
46  end
```

Listing 2: Función `frankWolfeQP_GAP` en Matlab

6. Simulación del Sistema

Para evaluar el desempeño del esquema de control propuesto, se requiere definir valores adecuados para los elementos del circuito mostrado en la Figura 1. En esta sección se especifican los parámetros adoptados para la simulación, los cuales se han determinado siguiendo criterios de diseño de filtros LCL [10] y considerando restricciones típicas de los convertidores de potencia.

6.1. Tensión de enlace de continua

La tensión V_d del enlace de continua se establece en función de la capacidad del convertidor. Según valores típicos reportados en la literatura, los dispositivos semiconductores de potencia (IGBT o MOSFET) presentan tensiones de bloqueo en el orden de 2.5 [kV], lo que permite operar con enlaces de continua de hasta 5 [kV]. Por lo tanto, se adopta $V_d = 5$ [kV] como tensión de continua para la simulación.

6.2. Parámetros del filtro LCL

El diseño del filtro LCL se basa en las recomendaciones de [10]. Para las inductancias, se considera que la suma de las reactancias debe ser aproximadamente 0,1 [pu]. En lugar de fijar primero la potencia nominal, se define directamente $L_1 = L_2 = 0{,}2$ [mH]. A partir de este valor se obtiene la potencia base del sistema.

Sea

$$Z_{\text{máx}} = 3 \cdot 2 \cdot 100\pi L_1, \qquad Z_b = 10\, Z_{\text{máx}}. \tag{74}$$

Con una tensión base de $V_b = 1{,}5$ [kV] (en valor *peak*), se obtiene

$$I_b = \frac{V_b}{Z_b} = 397{,}89 \text{ [A]}, \qquad S_b = \frac{3V_bI_b}{2} = 895{,}25 \text{ [kVA]}.$$

Cabe señalar que todos los valores base se expresan en magnitud *peak*, y no en RMS.

Capacitor del filtro

El valor máximo del capacitor se calcula como

$$C_{\text{máx}} = \frac{0{,}05P_n}{2\pi f_{\text{grid}} U_{gn}^2}, \tag{75}$$

donde U_{gn} corresponde a la tensión línea–línea RMS. Considerando un factor de potencia de 0.8, se obtiene

$$C_{\text{máx}} = \frac{0{,}05 \cdot 0{,}8\, S_b}{2\pi \cdot 50 \cdot \left(V_b\sqrt{\frac{3}{2}}\right)^2}.$$

De esta forma se obtiene $C_{\text{máx}} = 33{,}77$ [μF], y se selecciona $C = 30$ [μF].

Condición de diseño

El filtro debe cumplir la restricción

$$10 f_{\text{grid}} \leq f_{\text{res}} \leq \tfrac{f_{\text{sw}}}{2}, \tag{76}$$

donde la frecuencia de resonancia se calcula como

$$\omega_{\text{res}} = \sqrt{\frac{2(L_1 + L_2)}{L_1 L_2 C}}, \qquad f_{\text{res}} = \frac{\omega_{\text{res}}}{2\pi}. \tag{77}$$

Sustituyendo los valores adoptados, se obtiene $f_{\text{res}} = 2{,}74$ [kHz].

Considerando $f_{\text{grid}} = 50$ [Hz] y $f_{\text{sw}} = 20$ [kHz], se verifica:

$$0{,}5 \text{ [kHz]} \leq 2{,}74 \text{ [kHz]} \leq 10 \text{ [kHz]},$$

lo que confirma el cumplimiento de la condición (76).

Resistencia de amortiguamiento

Para mitigar la aparición de oscilaciones, se introduce una resistencia de amortiguamiento $R_d = 10$ [Ω], la cual, de acuerdo a [10], debe cumplir que disipe una potencia inferior al 10 % de la potencia nominal monofásica.

Por otra parte, la resistencia R_d debe disipar una potencia P_{loss,R_d} inferior al 10 % de la potencia nominal monofásica.

De acuerdo a (52), la corriente en estado estacionario que circuila a través de la resistencia se calcula como:

$$I_0 = \frac{j\omega C}{1 + j\omega R_d C} V_0,$$

obteniéndose $I_0 = -0{,}83 + j17{,}07$ [A]. La potencia disipada es entonces:

$$P_{\text{loss},R_d} = |I_0|^2 R_d = 2{,}92 \text{ [W]},$$

valor muy inferior al límite del 10 % de la potencia nominal (23,87 [kW]). Por lo tanto, R_d cumple con la restricción.

Resistencias parásitas en las inductancias

Finalmente, se añade una resistencia serie de 1 [Ω] a las inductancias L_1 y L_2. Esta consideración busca representar pérdidas resistivas parásitas propias de los devanados y, además, contribuye al amortiguamiento de oscilaciones en el sistema.

6.3. Matrices de peso

Para la formulación del problema de control predictivo, se requiere definir matrices de ponderación Q y R que permiten ajustar la importancia relativa de los estados y de las acciones de control en la función de costo.

En este caso, se ha considerado una matriz $Q \in \mathbb{R}^{6\times 6}$ de la forma:

$$Q = \begin{bmatrix} Q_i & 0 & 0 & 0 & 0 & 0 \\ 0 & Q_i & 0 & 0 & 0 & 0 \\ 0 & 0 & Q_i & 0 & 0 & 0 \\ 0 & 0 & 0 & Q_i & 0 & 0 \\ 0 & 0 & 0 & 0 & Q_v & 0 \\ 0 & 0 & 0 & 0 & 0 & Q_v \end{bmatrix},$$

donde Q_i corresponde al peso asociado a las corrientes, mientras que Q_v es el peso asignado a las tensiones.

Por otra parte, la matriz $R \in \mathbb{R}^{3\times 3}$ utilizada para penalizar las acciones de control se define como:

$$R = \begin{bmatrix} Q_r & 0 & 0 \\ 0 & Q_r & 0 \\ 0 & 0 & Q_r \end{bmatrix}.$$

Tras realizar diversas simulaciones del sistema, se seleccionaron los valores $Q_i = 1$, $Q_v = 20$ y $Q_r = 10^4$. Con ello, las matrices finales empleadas en las simulaciones quedan expresadas como:

$$Q = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 20 & 0 \\ 0 & 0 & 0 & 0 & 0 & 20 \end{bmatrix}, \quad R = \begin{bmatrix} 10^4 & 0 & 0 \\ 0 & 10^4 & 0 \\ 0 & 0 & 10^4 \end{bmatrix}.$$

Resumen de parámetros del sistema

Parámetro	**Descripción**	**Valor**
V_d	Tensión de continua	5 [kV]
L_1	Inductancia lado convertidor	0.2 [mH]
L_2	Inductancia lado red	0.2 [mH]
R_1, R_2	Resistencia serie de L_1, L_2	1 [Ω]
C	Capacitor del filtro	30 [μF]
R_d	Resistencia de amortiguamiento	10 [Ω]
f_{res}	Frecuencia de resonancia	2.74 [kHz]
f_{sw}	Frecuencia de conmutación	20 [kHz]
f_{grid}	Frecuencia de red	50 [Hz]
S_b	Potencia base	895.25 [kVA]
Q_i	Peso asociado a corrientes	1 [-]
Q_v	Peso asociado a tensiones	20 [-]
Q_r	Peso asociado a las acciones de control	10^4 [-]

Tabla 1: Parámetros adoptados para la simulación.

6.4. Simulación del sistema en Simulink

La simulación se implementó en Simulink dividiendo el modelo en dos bloques principales: la **planta** y el **controlador**.

Planta. La planta corresponde al sistema eléctrico compuesto por la fuente de tensión continua, el convertidor 3L-NPC, el filtro LCL y el SEP, como se muestra en la Figura 1.

Adicionalmente, se incluyó en el modelo el modulador *Space Vector PWM* (SVPWM), encargado de generar las señales de disparo para los IGBT del convertidor a partir de las acciones de control $U = [u_1, u_2, u_3]$. Este conjunto fue implementado en *PLECS* dentro del entorno de Simulink.

Las variables de salida de la planta corresponden a las magnitudes eléctricas medidas, es decir, las corrientes de salida del convertidor $i_{1_{\alpha\beta}}$, las corriente de entrada al SEP $i_{2_{\alpha\beta}}$, las tensiones de los condensadores $v_{c_{\alpha\beta}}$ y las tensiones del SEP $v_{p_{\alpha\beta}}$.

Controlador. El bloque de control fue implementado mediante una función de MATLAB (`MATLAB Function`) en Simulink. Este recibe como entradas las variables medidas desde la planta, junto con los parámetros eléctricos y de control del sistema, mostrados en la Tabla 1, y entrega como salida las señales de actuación u_1, u_2 y u_3 que alimentan al modulador SVPWM.

El controlador realiza, en cada instante de muestreo, las siguientes operaciones:

1. **Cálculo de referencias de predicción.** A partir de las potencias activas y reactivas de referencia $(P_{\text{ref}}, Q_{\text{ref}})$ y de las condiciones instantáneas de tensión del SEP, se calculan las corrientes de referencia y las tensiones en el punto de acoplamiento, generando así la trayectoria de referencia X_{ref} y U_{ref} a lo largo del horizonte de predicción N.

2. **Construcción del problema de optimización.** Se forman las matrices H y f asociadas al problema cuadrático de control predictivo, de acuerdo con la formulación desarrollada en secciones anteriores.

3. **Resolución del problema de control predictivo.** El problema de optimización se resuelve utilizando alguno de los dos algoritmos implementados en este trabajo:

- Método *Active Set*,
- Método de Frank-Wolfe.

El algoritmo seleccionado entrega la secuencia de acciones de control U, de la cual solo la primera entrada $[u_1, u_2, u_3]$ se aplica al sistema.

Además de las señales de control, el bloque retorna las referencias utilizadas y el número de iteraciones y FLOPs asociados a la resolución del problema, lo que permite evaluar el desempeño y la complejidad computacional de cada algoritmo en la simulación.

6.4.1. Seguimiento de referencias

Con el sistema y parámetros definidos previamente, se procede a validar la correcta operación del esquema de control mediante el seguimiento de referencias de las variables i_1, i_2 y v_c. Para este propósito, se considera una condición nominal con $P_{\text{ref}} = 0{,}8\,[\text{pu}]$ y $Q_{\text{ref}} = 0{,}6\,[\text{pu}]$, simulando un intervalo de 60 [ms] con un horizonte de predicción $N = 10$, utilizando el algoritmo Active-Set. Las formas de onda obtenidas se muestran en la Fig. 4.

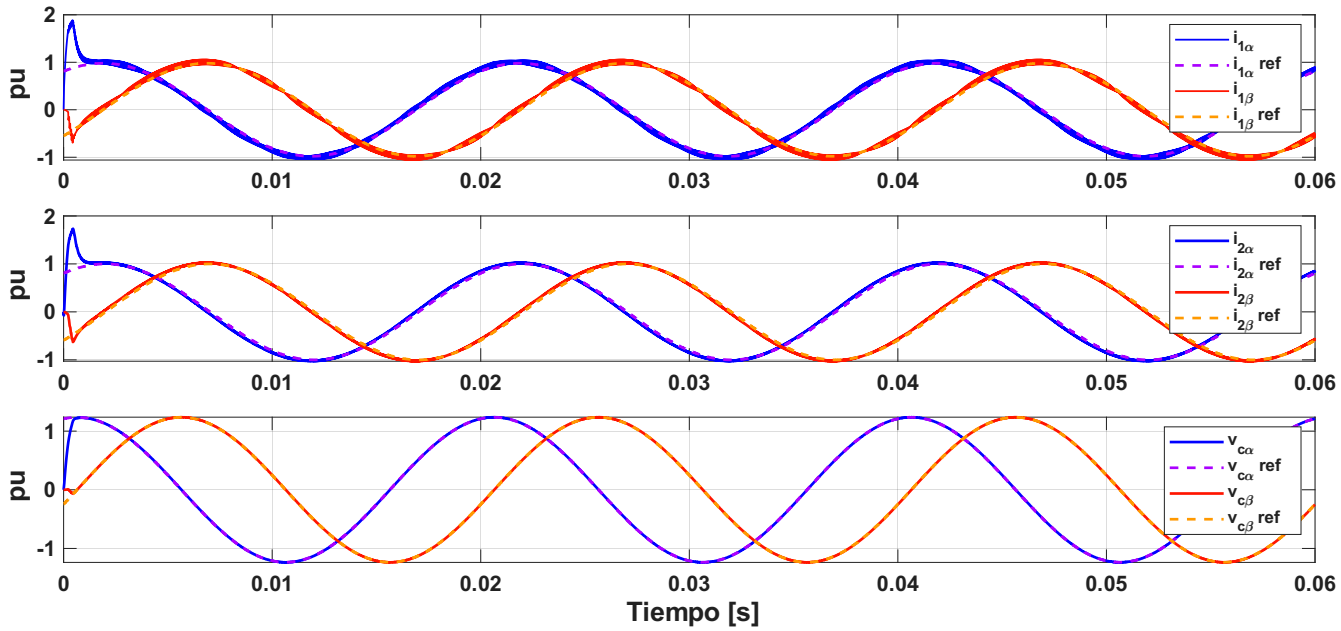

Fig. 4: Seguimiento de referencia de i_1, i_2 y v_c en condición nominal.

Posteriormente, se evalúa la respuesta del sistema frente a un cambio escalonado en la referencia de potencia activa, manteniendo constante la potencia reactiva. La Fig. 5 ilustra las formas de onda de las corrientes y tensiones utilizando los algoritmos Active-Set y Frank-Wolfe para el siguiente escenario:

- $t = 0\,[\text{ms}]$: $P_{\text{ref}} = 0{,}4\,[\text{pu}]$,
- $t = 20\,[\text{ms}]$: $P_{\text{ref}} = 1{,}5\,[\text{pu}]$,
- $t = 40\,[\text{ms}]$: $P_{\text{ref}} = 1{,}0\,[\text{pu}]$.

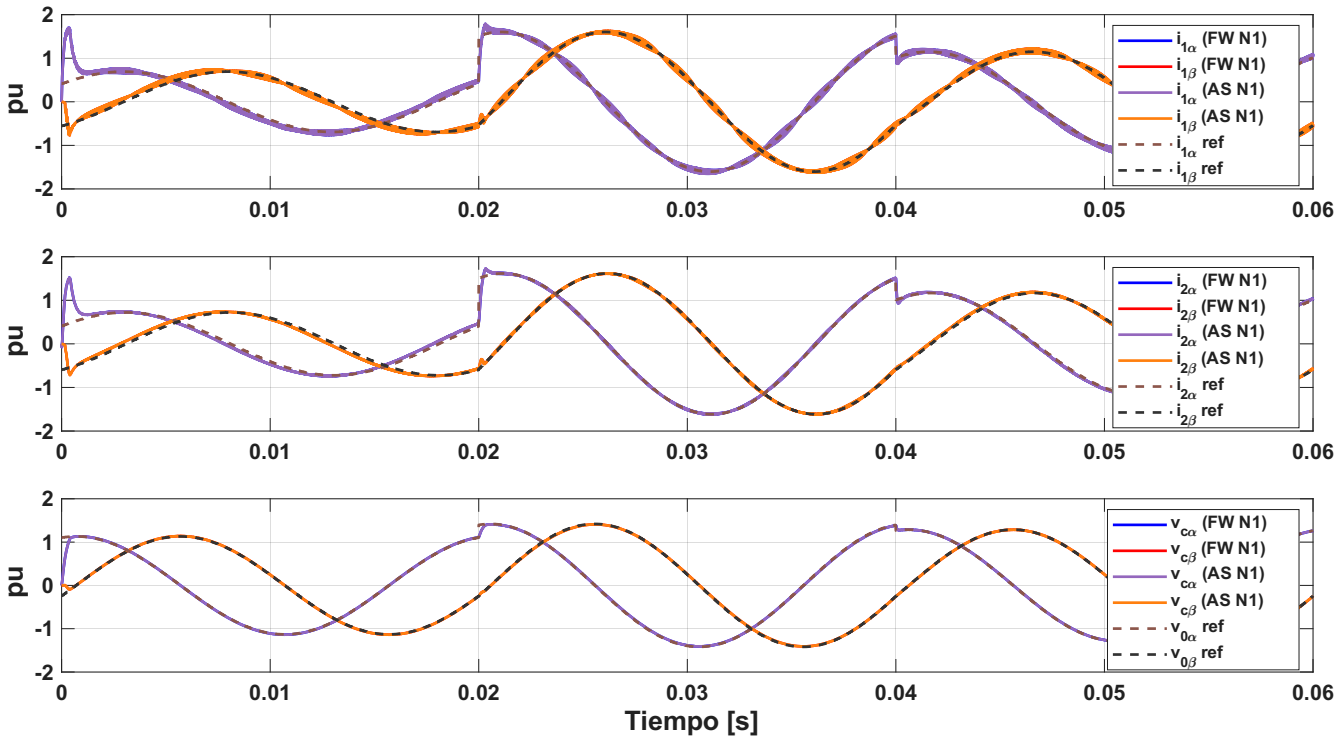

Fig. 5: Seguimiento de referencia de i_1, i_2 y v_0 frente a un escalón en la potencia activa, utilizando los algoritmos Active-Set y Frank-Wolfe.

Se puede observar que ambos algoritmos logran un correcto seguimiento de referencias por parte de las tensiones y corrientes del sistema en estado estacionario para los tres niveles de potencia planteados.

Por otro lado, para observar el comportamiento de las formas de onda durante el transitorio, se realiza un acercamiento al intervalo de tiempo donde ocurre el primer escalón de potencia, lo cual se presenta en la Fig. 6.

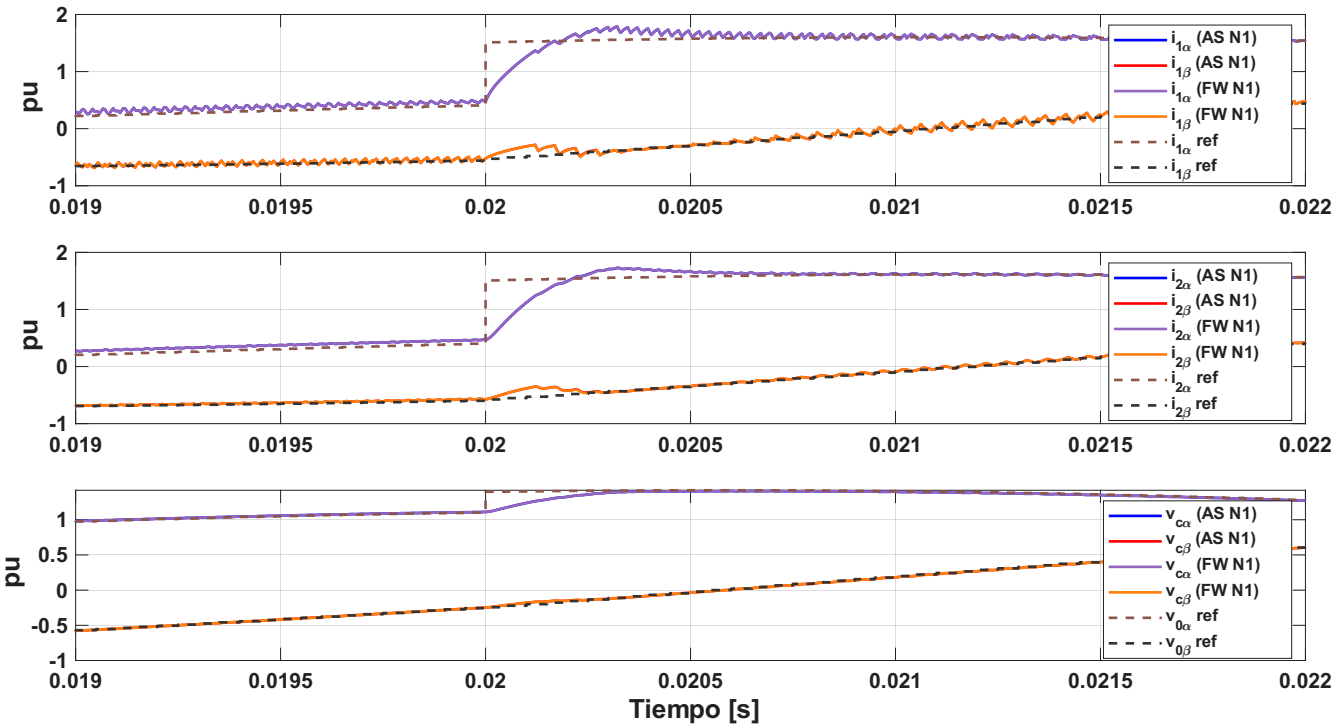

Fig. 6: Seguimiento de referencia de i_1, i_2 y v_0 durante el transitorio utilizando los algoritmos Active-Set y Frank-Wolfe.

Los resultados confirman que el sistema de control es capaz de seguir adecuadamente las trayectorias de referencia, tanto en régimen estacionario como en condiciones transitorias. Una vez verificada la validez del modelo y del esquema de control, se procede a la comparación del desempeño de los algoritmos de optimización implementados, teniendo en cuenta que ambos logran llegar al mismo resultado.

6.5. Evaluación comparativa de algoritmos de optimización

Para comparar el desempeño de los algoritmos desarrollados —Active-Set y Frank-Wolfe — se analiza el transitorio ocurrido en $t = 0{,}04$ [s] en la Fig. 5. Para ello, se registran el número máximo de iteraciones que requiere cada algoritmo para resolver el problema de optimización y el número de operaciones en punto flotante (FLOPs) asociadas. Este procedimiento se repite variando el valor de N entre 1 y 10, resumiéndose los resultados en la Tabla 2 y su representación gráfica en la Fig. 7.

El conteo de FLOPs se estimó con el fin de caracterizar la carga computacional de los algoritmos, independientemente del hardware empleado. Siguiendo la convención descrita en [11], se contabilizan únicamente las operaciones aritméticas elementales (sumas y multiplicaciones) involucradas en las operaciones de álgebra lineal, omitiéndose asignaciones, comparaciones lógicas y operaciones escalares por considerarse despreciables en relación con el costo dominante. De este modo, el costo de cada operación se aproxima mediante expresiones estándar, tales como:

- Producto matriz–vector $(m \times n)$: $2mn$ FLOPs.

- Producto interno entre vectores de tamaño n: $2n$ FLOPs.
- Suma o resta de vectores de tamaño n: n FLOPs.
- Resolución de un sistema lineal $n \times n$: $\frac{2}{3}n^3$ FLOPs, correspondiente a la factorización LU y sustituciones.

Durante la simulación de cada algoritmo se registró, en cada instante de cálculo de la acción de control, el número de iteraciones y la cantidad de FLOPs requeridas. A partir de estos registros, se seleccionó el valor máximo alcanzado a lo largo de toda la simulación, correspondiente al escenario de mayor carga computacional. En el Anexo A se muestra el detalle de los códigos de Matlab utilizados para el conteo de FLOPs.

	Iteraciones		**FLOPs**	
N	**Active-Set**	**Frank-Wolfe**	**Active-Set**	**Frank-Wolfe**
1	3	5	216	291
2	5	177	2358	34 038
3	6	824	7890	326 601
4	6	2486	15 438	1 671 420
5	6	5790	26 730	5 907 560
6	6	10 734	42 522	15 460 100
7	6	17 316	63 570	33 459 700
8	6	26 880	90 630	67 100 500
9	6	38 907	124 458	121 868 000
10	6	53 002	165 810	203 544 000

Tabla 2: Comparación de iteraciones y FLOPs para distintos valores de N.

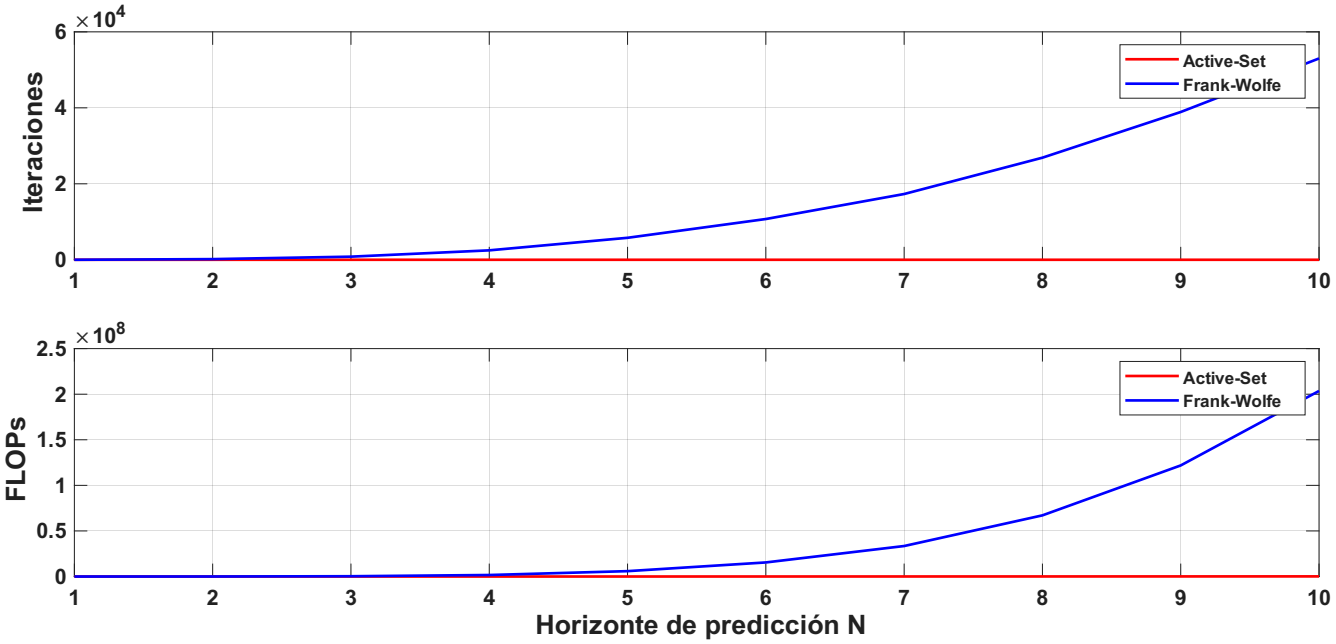

Fig. 7: Comparación de iteraciones y FLOPs entre Active-Set y Frank-Wolfe.

Como se aprecia en la Tabla 2, el algoritmo Active-Set mantiene un número prácticamente constante de iteraciones y un crecimiento moderado en los FLOPs al aumentar N. En contraste, el algoritmo Frank-Wolfe exhibe un crecimiento acelerado tanto en el número de iteraciones como en los FLOPs, alcanzando varios órdenes de magnitud por encima de Active-Set. Como consecuencia, en la representación gráfica de la Fig. 7, las curvas de Active-Set quedan prácticamente superpuestas al eje horizontal, dando la impresión de una recta en cero debido a la gran disparidad de escalas.

En conclusión, aunque ambos métodos logran resolver el problema planteado, el enfoque Active-Set resulta significativamente más eficiente desde el punto de vista computacional, lo que lo hace más adecuado para aplicaciones donde el tiempo de cómputo y los recursos disponibles son factores críticos. Además, se observa que, a medida que aumenta el horizonte de predicción N, la diferencia de desempeño entre Active-Set y Frank-Wolfe se amplifica considerablemente, reforzando la superioridad del primero frente al crecimiento del problema.

7. Hardware-in-the-Loop (HIL)

Para validar el desempeño del controlador en condiciones de ejecución en tiempo real, se empleó la técnica de Hardware-in-the-Loop (HIL), implementada mediante el simulador en tiempo real RTDS y una tarjeta de control Imperix. El modelo de la planta, que incluye la red eléctrica y el inversor de potencia, fue desarrollado en el software RSCAD y ejecutado en el RTDS, lo que permitió emular la dinámica del sistema eléctrico con alta fidelidad.

El controlador, previamente diseñado y probado en Simulink, fue posteriormente compilado y

descargado en la tarjeta Imperix, donde se ejecuta en tiempo real. El entorno *Cockpit* de Imperix sirvió como interfaz de usuario para supervisar variables internas del controlador y modificar referencias de potencia durante las pruebas. De esta manera, el sistema HIL queda conformado por: (i) el RTDS, que ejecuta la planta simulada en RSCAD; (ii) la tarjeta Imperix, que implementa el controlador en tiempo real; y (iii) Cockpit, que facilita la interacción con la simulación.

7.1. Modificaciones con respecto a Simulink

Durante la implementación en tiempo real, fue necesario introducir una serie de ajustes respecto al modelo empleado en Simulink, con el objetivo de asegurar la correcta ejecución del controlador dentro de las restricciones temporales impuestas por el sistema HIL.

Frecuencia de conmutación. Debido a las limitaciones de tiempo de cómputo, la frecuencia de conmutación debió reducirse a fin de garantizar que el cálculo de la acción de control se completara dentro del período de muestreo. Tras un análisis exploratorio, se determinó que una frecuencia de 8 [kHz] ofrecía un compromiso adecuado entre precisión temporal y capacidad de procesamiento, permitiendo además evaluar hasta $N = 3$ sin sobrepasar el límite de 125 [μs] disponible por ciclo.

Corrección del algoritmo Active-Set. Durante la compilación del controlador en la tarjeta Imperix, el algoritmo Active-Set original presentó incompatibilidades en la asignación dinámica del índice de restricciones activas. En particular, la instrucción:

```
R_as(1:Lras) = A_sub(1:Lras,1:Lras)\b_sub(1:Lras);
```

generaba errores de tamaño en tiempo de ejecución. Para resolverlo, se reemplazó la asignación dinámica por una estructura condicional que define explícitamente el rango de operación en función del número de restricciones activas (`Nac`), evitando el uso de índices variables. El fragmento de código corregido se detalla en el Anexo B. Esta modificación permite la ejecución estable del algoritmo con hasta 12 restricciones activas, un número significativamente superior al observado en las simulaciones (donde el máximo fue seis), asegurando así un margen operativo suficiente.

Ajuste del parámetro Q_v. Finalmente, durante las pruebas experimentales se observó que el desempeño dinámico del controlador mejoraba al incrementar el peso asociado a la tensión en la matriz Q. En consecuencia, se adoptó un valor de $Q_v = 30$ en lugar de $Q_v = 20$, lo que permitió una respuesta más estable y una mejor compensación frente a perturbaciones rápidas sin comprometer la estabilidad general del sistema.

7.2. Simulación en tiempo real

La Fig. 8 muestra la operación nominal del sistema con referencias $P_{\text{ref}} = 0{,}8\,[\text{pu}]$ y $Q_{\text{ref}} = 0{,}6\,[\text{pu}]$ para un horizonte de predicción $N = 2$.

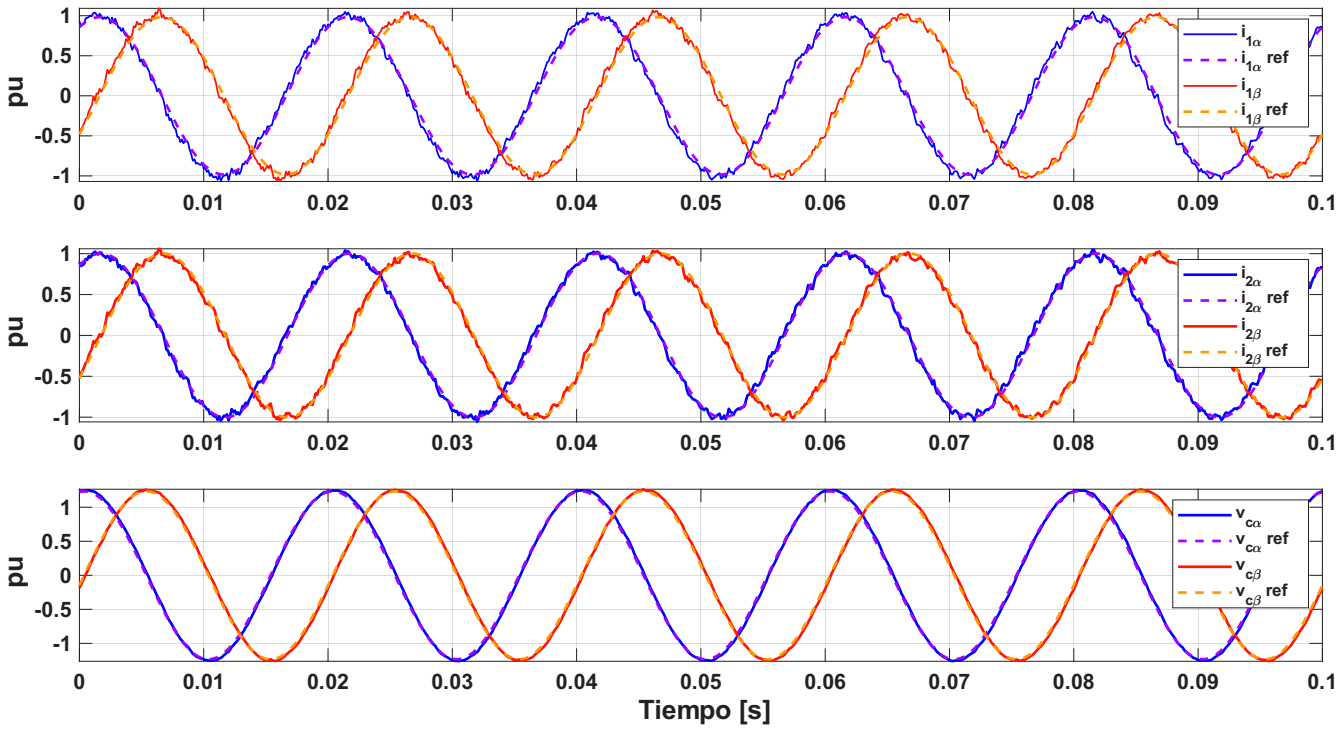

Fig. 8: Seguimiento de referencia de i_1, i_2 y v_0 en operación nominal utilizando HIL.

Como se aprecia en la figura, el controlador implementado en la tarjeta Imperix logra mantener un seguimiento preciso de las referencias de corriente y tensión, reproduciendo de manera fiel el comportamiento obtenido en las simulaciones de Simulink. Esto evidencia que la interacción entre la planta emulada en RTDS y el controlador en tiempo real es correcta, y que el algoritmo Active-Set puede ejecutarse de manera estable dentro de las restricciones de tiempo del sistema HIL.

Por otra parte, la Fig. 9 ilustra la respuesta del sistema ante escalones de potencia activa, simulando la transición de $0{,}4 \rightarrow 1{,}5 \rightarrow 1$ [pu].

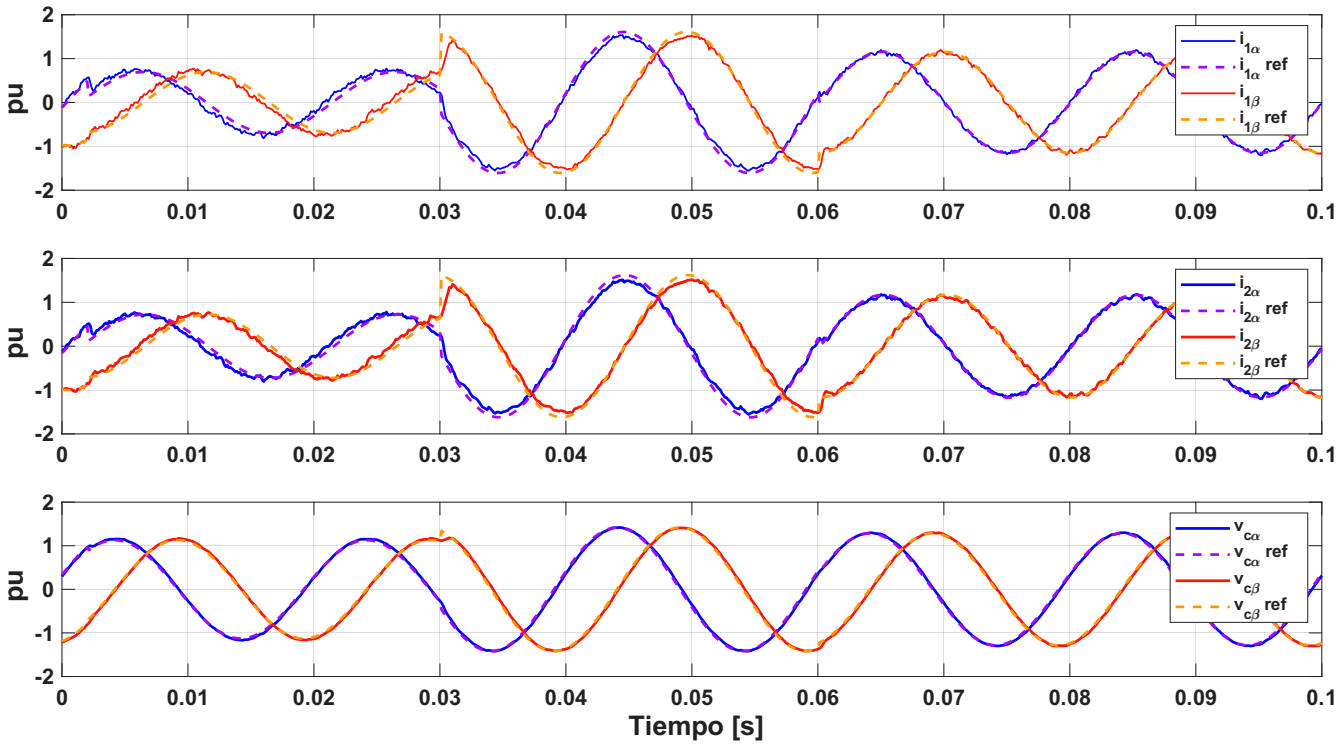

Fig. 9: Seguimiento de referencia de i_1, i_2 y v_0 frente a un escalón en la potencia activa utilizando HIL.

Durante estos transitorios, se observa un pequeño desfase en el seguimiento de las corrientes cuando las referencias se alejan del valor nominal, atribuible principalmente a la reducción de la frecuencia de conmutación para cumplir con las restricciones de tiempo real. Sin embargo, el error es mínimo y no compromete la estabilidad del sistema ni la capacidad del controlador para regular las variables eléctricas de interés. Además, se puede apreciar que la tensión del condensador v_0 mantiene un comportamiento muy cercano a su referencia, lo que indica un correcto amortiguamiento de las oscilaciones y una adecuada compensación de perturbaciones rápidas.

En conjunto, estos resultados confirman que el esquema de control predictivo diseñado es capaz de operar de manera confiable en tiempo real, cumpliendo tanto con los objetivos de seguimiento de referencia como con las limitaciones computacionales del sistema HIL.

7.3. Comparación de tiempos de cómputo

Una vez comprobado que el sistema funciona como corresponde, se procede a analizar el tiempo de cómputo requerido por cada algoritmo durante el cálculo de la acción de control. Para ello, se registró en cada simulación el *tiempo máximo de cómputo* observado, definido como el mayor tiempo empleado por el solver en alguna iteración durante la simulación. Este parámetro resulta más representativo que el simple conteo de iteraciones, ya que refleja directamente el tiempo real que insume la ejecución del controlador.

La metodología consistió en aplicar el mismo escenario de escalones de potencia para dis-

tintos valores de N (1, 2 y 3) y repetir el experimento 100 veces por cada configuración. Se compararon tres algoritmos: el método Active-Set propuesto, el algoritmo Frank-Wolfe y la función `mpcActiveSetSolver` incluida en MATLAB, utilizada como referencia. Con la información recolectada se construyeron curvas de probabilidad acumulada (CDF), que permiten visualizar la probabilidad de que el cálculo de la acción de control se complete en un tiempo inferior a un umbral dado. Estas curvas se presentan en la Fig. 10.

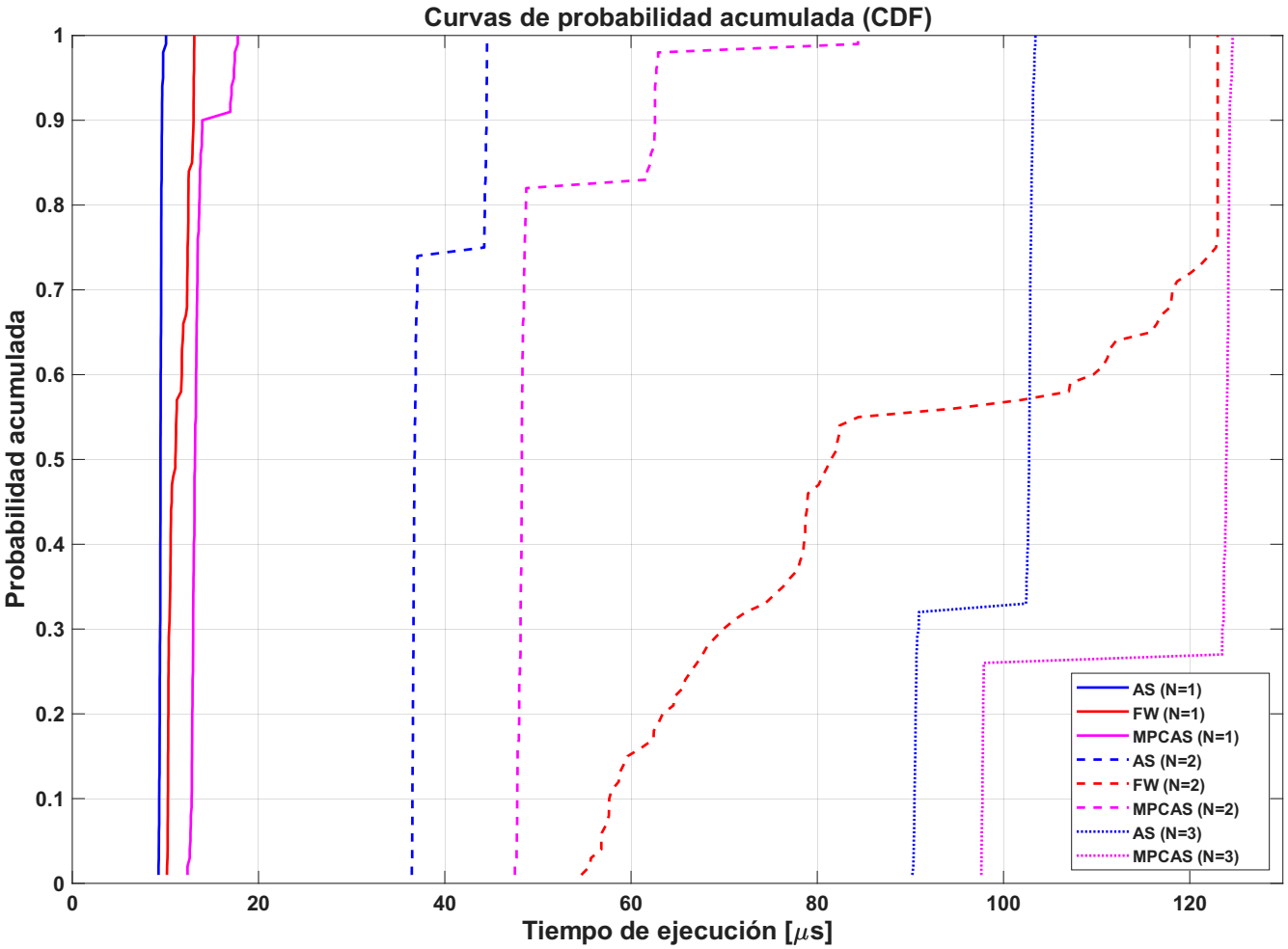

Fig. 10: Curvas de probabilidad acumulada (CDF) del tiempo de cómputo para los algoritmos Active-Set, Frank-Wolfe y `mpcActiveSetSolver`.

Un aspecto relevante es la ausencia de resultados para Frank-Wolfe en el caso $N = 3$. Esto se debe a que, con una frecuencia de muestreo de 8 [kHz], el sistema dispone solo de 125 [μs] para calcular la acción de control antes de entregarla al modulador del convertidor. Dentro de este margen temporal, Active-Set requiere entre 90 y 103 [μs], mientras que `mpcActiveSetSolver` alcanza valores de entre 97 y 124 [μs], situándose justo en el límite de lo permitido. En cambio, Frank-Wolfe supera siempre los 125 [μs], por lo que no logra entregar la solución a tiempo y, en

consecuencia, no resulta factible de implementar. Para $N = 4$ y superiores, todos los algoritmos exceden incluso los 200 [μs], motivo por el cual se descartó también la posibilidad de operar a una frecuencia reducida.

En cuanto a la comparación entre algoritmos, se observa que para $N = 1$ los tres métodos presentan tiempos similares, siendo Active-Set el más rápido, seguido por Frank-Wolfe y finalmente `mpcActiveSetSolver`. Al aumentar a $N = 2$, Active-Set mantiene el mejor desempeño, con una diferencia aproximada de 10 [μs] respecto de `mpcActiveSetSolver`, mientras que Frank-Wolfe experimenta un incremento considerable en su tiempo de cómputo, en concordancia con el aumento de iteraciones observado en las simulaciones de Simulink. Para $N = 3$, Active-Set continúa mostrando la menor latencia, `mpcActiveSetSolver` se aproxima al límite temporal, y Frank-Wolfe queda completamente fuera del rango operativo. Se aprecia así que, a medida que crece el horizonte de predicción N, la ventaja relativa de Active-Set frente a los demás métodos se hace más pronunciada. Estos resultados confirman que, bajo las condiciones de operación evaluadas, Active-Set constituye la alternativa más viable para una implementación en tiempo real.

8. Conclusiones generales

Los resultados obtenidos permiten concluir que, pese a las modificaciones implementadas para reducir su complejidad, el algoritmo Frank–Wolfe continúa siendo ineficiente para la resolución del problema predictivo planteado. Si bien para horizontes de predicción cortos ($N = 1$) su desempeño resulta comparable al del método Active–Set e incluso superior al del `mpcActiveSetSolver`, el incremento de N provoca un crecimiento exponencial en el número de iteraciones, los FLOPs y el tiempo de cómputo requerido, volviéndolo inviable para su implementación en tiempo real.

En contraste, el método Active–Set demostró un comportamiento notablemente más estable y eficiente, manteniendo un número de iteraciones prácticamente constante y un crecimiento moderado en su carga computacional al aumentar N. Este comportamiento se reflejó tanto en las simulaciones en Simulink como en las pruebas *hardware-in-the-loop*, donde Active–Set fue el único algoritmo capaz de cumplir consistentemente con las restricciones temporales impuestas por la frecuencia de muestreo.

De esta manera, se confirma que el enfoque Active–Set constituye la alternativa más adecuada para la implementación de control predictivo en sistemas de electrónica de potencia con requerimientos de operación en tiempo real. Su desempeño robusto frente al aumento del horizonte de predicción evidencia una mayor escalabilidad y viabilidad práctica, aspectos esenciales para aplicaciones de control rápido como las abordadas en este trabajo.

Referencias

[1] A. M. Castro, "Apuntes para la asignatura Accionamientos Eléctricos: Control Vectorial de Máquinas de Inducción," 2021.

[2] C. Yu, Q. Wang, W. Fang, Y. Wang, H. Diao, H. Xu, and L. Guo, "Research on dynamic and steady-state characteristics of grid-following/grid-forming hybrid control based on model predictive control," *IEEE Open Journal of Power Electronics*, vol. 6, pp. 909–918, 2025.

[3] R. P. Aguilera, "Control Studio B (CSB) Model Predictive Control (MPC)." School of Electrical and Data Engineering, Faculty of Engineering and IT, University of Technology Sydney, Australia, 2020.

[4] V. Selarka, P. Shah, D. J. Vaghela, and M. T. Shah, "Close loop control of three phase active front end converter using svpwm technique," in *2016 International Conference on Electrical Power and Energy Systems (ICEPES)*, pp. 339–344, 2016.

[5] U. of Newcastle, "Exact discretisation of state-space equations (lecture notes)," 2025. Lecture 11: State Space Equations.

[6] M. E. Salgado, J. I. Yuz, and R. A. Rojas, *Análisis de sistemas lineales*. Pearson Educación, 2005.

[7] C. J. O'Rourke, M. M. Qasim, M. R. Overlin, and J. L. Kirtley, "A geometric interpretation of reference frames and transformations: dq0, clarke, and park," *IEEE Transactions on Energy Conversion*, vol. 34, no. 4, pp. 2070–2083, 2019.

[8] J. Nocedal and S. J. Wright, *Numerical optimization*. Springer, 2006.

[9] S. Pokutta, "The frank-wolfe algorithm: a short introduction," 2023.

[10] M. Saïd-Romdhane, M. Naouar, I. Slama-Belkhodja, and E. Monmasson, "Lcl filter design for three-phase grid-connected converters," 05 2014.

[11] G. H. Golub and C. F. Van Loan, *Matrix computations*. JHU press, 2013.

Anexos

A. Códigos en Matlab considerando conteo de FLOPs

A.1. Active-Set

```
function [U, iter, flops] = ActiveSet_Sim(H, f, Mm)
    max_iter = 100;
    tol = 1e-4;
    iter = 0;
    flops = 0;

    U = max(min(-H\f, Mm), -Mm);
    lu0 = size(U,1); % 3N

    % Flops asociados al calculo de U
    flops = flops + (2/3)*lu0^3;

    %El uso de 2*lu0 se debe a que se suma el largo de U mas el numero de
       multiplicadores de lagrange

    R_as = ones(2*lu0, 1); % Resultado subproblema
    A_sub = [H, zeros(lu0,lu0); zeros(lu0,2*lu0)];
    b_sub = zeros(2*lu0, 1);
    Nac = 0;

    for k = 1:max_iter
        iter = iter + 1;
        Lras = lu0 + Nac;

        % Resolver sub sistema
        if Nac == 0
            pk = -H\f - U;
            flops = flops + (2/3)*lu0^3 + lu0; % por solve y suma
        else
             % Calcular gradiente
            gk = H*U + f;
            flops = flops + 2*numel(H) + lu0; %Multiplicacion de matriz y suma
            % Se crea la matriz del subproblema:
            % [H, A' ; A, 0] * [x;Lambda] = [-gk, 0]

            b_sub(1:lu0) = -gk;

            R_as(1:Lras) = A_sub(1:Lras,1:Lras)\b_sub(1:Lras);
            flops = flops + (2/3)*(Lras)^3;


            % Separar solucion
            pk = R_as(1:lu0);
```

```
        end


        % Evaluar condicion de optimalidad
        if all(abs(pk) < tol)
            if all(R_as(lu0 + 1:end) >= 0)
                break;
            else
                [~, idx] = min(R_as(lu0 + 1:end));

                A_sub(lu0+idx,1:lu0) = A_sub(lu0+Nac,1:lu0);
                A_sub(1:lu0, lu0+idx) = A_sub(1:lu0, lu0+Nac);
                A_sub(lu0+Nac,1:lu0) = 0;
                A_sub(1:lu0, lu0+Nac) = 0;
                Nac = Nac -1;
            end
        else
            U = U + pk;
            flops = flops + lu0; %Suma de vectores

            rac = 0;
            for i = 1:lu0
                if (U(i) - Mm > 1e-6)
                    rac = 1;
                    Nac = Nac + 1;
                    A_sub(lu0+Nac,i)    = 1;
                    A_sub(i,lu0+Nac)    = 1;
                    break;
                elseif (-U(i) - Mm > 1e-6)
                    rac = 1;
                    Nac = Nac + 1;
                    A_sub(lu0+Nac,i)    = 1;
                    A_sub(i,lu0+Nac)    = 1;
                    break;
                end
            end


            if rac
                U = max(min(U, Mm), -Mm);
                if Nac == lu0
                    break
                end
            else
                break
            end
        end
    end
end
```

Listing 3: Función ActiveSet_Sim en Matlab incluyendo el conteo de FLOPs

A.2. Frank-Wolfe

```
function [U, iter, flops] = frankWolfeQP_GAP(H, f, max_iter, Mm)
    % Algoritmo de Frank-Wolfe con conteo de FLOPs
    iter = 0;
    flops = 0;
    U = max(min(-H\f, Mm), -Mm);
    lu0 = size(U,1); % 3N, size(H,1)
    n = lu0/3;
    % Flops asociados al calculo de U
    flops = flops + (2/3)*lu0^3;
    s = zeros(lu0,1);

    tol = norm(f,2)/((n^2) * 2e2); %Tolerancia depende del problema


    for k = 1:max_iter
        iter = iter + 1;

        % Gradiente
        grad = H * U + f;
        flops = flops + 2*numel(H) + lu0; %Multiplicacion de matriz y suma

        % Vertice y direccion
        for i = 1:lu0
            if grad(i) > 1e-4
                s(i) = -Mm;    % limite inferior
            elseif grad(i) < -1e-4
                s(i) =  Mm;    % limite superior
            else
                s(i) = U(i);  % si grad(i) esta dentro del rango [-tol2, tol2]
            end
        end
        d = s - U;
        flops = flops + 2*lu0;   % s + Mm*sign(grad)



        % Frank-Wolfe gap
        gap = -grad' * d;  % grad*(x-s)
        flops = flops + 2*lu0;       % grad' * (x - s)

        % Verificacion de convergencia
        if gap < tol
            break;
        end


        % Calculo del paso
        alpha = gap / (d' * H * d);
        alpha = max(0, min(1, alpha));
```

```
50
51          flops = flops + 2*numel(H) + 2*lu0;   % d'*H*d
52
53
54          % Actualizacion
55          U = U + alpha * d;
56          flops = flops + lu0;
57
58      end
59  end
```

Listing 4: Función FrankWolfeQP_GAP en Matlab incluyendo el conteo de FLOPs

B. Código Active-Set adaptado para ejecución en Imperix

El siguiente código corresponde a la versión del algoritmo *Active-Set* utilizada en la implementación con Imperix. A diferencia de la versión empleada en Matlab para el análisis de carga computacional (Sección A), este código fue optimizado para su ejecución en tiempo real, eliminando el conteo de FLOPs y reemplazando la estructura de resolución de subproblemas por una secuencia de condiciones `if` explícitas.

Esta modificación permite evitar la asignación dinámica de matrices dentro del bucle, lo que mejora la predictibilidad del tiempo de cómputo y asegura el cumplimiento de los límites de ejecución impuestos por la frecuencia de muestreo.

```
1  function [U, iter] = ActiveSet_Imperix(H, f, Mm)
2      max_iter = 12;
3      tol = 1e-4;
4      iter = 0;
5
6      U = max(min(-H\f, Mm), -Mm);
7      lu0 = size(U,1); % 3N
8
9      R_as = ones(2*lu0, 1); % Resultado del linesolve que contiene pk y Lka
10     A_sub = [H, zeros(lu0,lu0); zeros(lu0,2*lu0)];
11     b_sub = zeros(2*lu0, 1);
12     Nac = 0;
13
14     for k = 1:max_iter
15         iter = iter + 1;
16
17         % Resolver sub sistema
18         if Nac == 0
19             pk = -H\f - U;
20         else
21             % Calcular gradiente
22             gk = H*U + f;
23             b_sub(1:lu0) = -gk;
24
```

```
25            % Resolver subproblema de acuerdo al numero de restricciones
                  activas
26            if Nac == 1
27                R_as(1:lu0+1) = A_sub(1:lu0+1,1:lu0+1)\b_sub(1:lu0+1);
28            elseif Nac == 2
29                R_as(1:lu0+2) = A_sub(1:lu0+2,1:lu0+2)\b_sub(1:lu0+2);
30            elseif Nac == 3
31                R_as(1:lu0+3) = A_sub(1:lu0+3,1:lu0+3)\b_sub(1:lu0+3);
32            elseif Nac == 4
33                R_as(1:lu0+4) = A_sub(1:lu0+4,1:lu0+4)\b_sub(1:lu0+4);
34            elseif Nac == 5
35                R_as(1:lu0+5) = A_sub(1:lu0+5,1:lu0+5)\b_sub(1:lu0+5);
36            elseif Nac == 6
37                R_as(1:lu0+6) = A_sub(1:lu0+6,1:lu0+6)\b_sub(1:lu0+6);
38            elseif Nac == 7
39                R_as(1:lu0+7) = A_sub(1:lu0+7,1:lu0+7)\b_sub(1:lu0+7);
40            elseif Nac == 8
41                R_as(1:lu0+8) = A_sub(1:lu0+8,1:lu0+8)\b_sub(1:lu0+8);
42            elseif Nac == 9
43                R_as(1:lu0+9) = A_sub(1:lu0+9,1:lu0+9)\b_sub(1:lu0+9);
44            elseif Nac == 10
45                R_as(1:lu0+10) = A_sub(1:lu0+10,1:lu0+10)\b_sub(1:lu0+10);
46            elseif Nac == 11
47                R_as(1:lu0+11) = A_sub(1:lu0+11,1:lu0+11)\b_sub(1:lu0+11);
48            elseif Nac == 12
49                R_as(1:lu0+12) = A_sub(1:lu0+12,1:lu0+12)\b_sub(1:lu0+12);
50            end
51
52            % Separar solucion
53            pk = R_as(1:lu0);
54        end
55
56        % Evaluar condicion de optimalidad
57        if all(abs(pk) < tol)
58            if all(R_as(lu0 + 1:end) >= 0)
59                break;
60            else
61                [~, idx] = min(R_as(lu0 + 1:end));
62                A_sub(lu0+idx,1:lu0) = A_sub(lu0+Nac,1:lu0);
63                A_sub(1:lu0, lu0+idx) = A_sub(1:lu0, lu0+Nac);
64                A_sub(lu0+Nac,1:lu0) = 0;
65                A_sub(1:lu0, lu0+Nac) = 0;
66                Nac = Nac -1;
67            end
68        else
69            U = U + pk;
70            rac = 0;
71            for i = 1:lu0
72                if (U(i) - Mm > 1e-6)
73                    rac = 1;
74                    Nac = Nac + 1;
```

```
                    A_sub(lu0+Nac,i)    = 1;
                    A_sub(i,lu0+Nac)    = 1;
                    break;
                elseif (-U(i) - Mm > 1e-6)
                    rac = 1;
                    Nac = Nac + 1;
                    A_sub(lu0+Nac,i)    = 1;
                    A_sub(i,lu0+Nac)    = 1;
                    break;
                end
            end

            if rac
                U = max(min(U, Mm), -Mm);
                if Nac == lu0
                    break
                end
            else
                break
            end
        end
    end
end
```

Listing 5: Función ActiveSet_Imperix implementada para ejecución en tiempo real.