UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE INGENIERÍA ELÉCTRICA

DISEÑO EFICIENTE DE ALGORITMOS DE PRONÓSTICO EN TIEMPO REAL EN BASE A OPTIMIZACIÓN DE TIEMPOS DE ACTUALIZACIÓN

TESIS PARA OPTAR AL GRADO DE MAGÍSTER EN CIENCIAS DE LA INGENIERÍA, MENCIÓN ELÉCTRICA

MEMORIA PARA OPTAR AL TÍTULO DE INGENIERO CIVIL ELÉCTRICO

KEVIN RACSO ESPINOZA OYANEDEL

PROFESOR GUÍA:
MARCOS ORCHARD CONCHA

PROFESOR CO-GUÍA:
DAVID ACUÑA URETA

MIEMBROS DE LA COMISIÓN:
FRANCISCO JARAMILLO MONTOYA
FELIPE TOBAR HENRÍQUEZ

SANTIAGO DE CHILE
2024

RESUMEN DE LA TESIS PARA OPTAR
AL GRADO DE MAGÍSTER EN CIENCIAS DE LA INGENIERÍA, MENCIÓN ELÉCTRICA
RESUMEN DE LA MEMORIA PARA OPTAR
AL TÍTULO DE INGENIERO CIVIL ELÉCTRICO
POR: KEVIN RACSO ESPINOZA OYANEDEL
FECHA: 2024
PROF. GUÍA: MARCOS ORCHARD CONCHA
PROF. CO-GUÍA: DAVID ACUÑA URETA

DISEÑO EFICIENTE DE ALGORITMOS DE PRONÓSTICO EN TIEMPO REAL EN BASE A OPTIMIZACIÓN DE TIEMPOS DE ACTUALIZACIÓN

En los últimos tiempos, el paradigma de PHM (Prognostic Health Management) se ha popularizado como solución para el mantenimiento predictivo de maquinaria industrial. Esto muchas veces requiere de algoritmos de bajo costo computacional, que puedan ejecutarse en tiempo real varias veces durante la operación del equipo. Para ello, se propone un enfoque basado en filtro de partículas que, a partir de reducir la frecuencia de muestreo del sistema, pretende cubrir extensos horizontes de predicción realizando menos cálculos. Modelos XGBoost son entrenados para decidir frecuencias, a partir de datos obtenidos mediante algoritmos genéticos, buscando minimizar el error de estimación del *Just in Time Point* (JITP) con respecto a la tasa de muestreo original, y respetando a la vez un límite máximo de tiempo de ejecución. Se toma como caso de estudio el problema de determinar el instante de descarga (EoD, End of Discharge) de una batería de Ión-Litio, para la cual se efectúan simulaciones usando condiciones iniciales del estado de carga (SOC, State of Charge) que abarcan todo el ciclo de operación, consiguiendo errores menores en el JITP, de menos de un minuto en promedio, en comparación con un horizonte de predicción de casi dos horas. Por otro lado, se reduce significativamente el tiempo de cómputo, a menos de 5 segundos en cualquier caso, equivalente a menos del 20 % de lo que demora el algoritmo de pronóstico estándar.

A mi madre, que hasta el día de hoy ha estado al mi lado, y a mi padre, que no alcanzó a estar

Agradecimientos

Quiero agradecer en primer lugar a todos mis amigos y colaboradores del día a día, a todos aquellos que, directa o indirectamente, aportaron un granito de arena, tanto a mi trabajo como en acompañarme.

Quiero agradecer también a mi madre y a mi padre. A mi madre, Edith, que siempre ha estado preocupada por mí y mi carrera, y me acompaña hasta hoy, y a mi padre, Oscar, que no alcanzó a estar en este momento, pero sé que le hubiese gustado verme donde estoy.

A Rubén, por su apoyo semanal y disposición para lograr organizar mi trabajo, mantener el avance, y también tranquilizarme y darme ánimos cuando el estrés parecía demasiado.

A Richard, Nataly, Cony y Cristóbal, amigos cuyo apoyo ha sido invaluable en esta difícil etapa, por su compañía, por su preocupación, y por saber que podía contar con ellos cuando la tarea se hacía compleja y me sentía abrumado por las dificultades.

A David Acuña, Francisco Jaramillo, y el profesor Felipe Tobar por haber aceptado ser parte de la comisión revisora de este trabajo, y por sus comentarios y sugerencias para perfeccionarlo.

A Loreto, por su tremendo trabajo administrativo en la secretaría de posgrado.

Y por último, pero no menos importante, al profesor Marcos Orchard, mi profesor guía, por su paciencia, su apoyo, su disposición y su expertiz en mi tema de investigación, y por su preocupación para llevar adelante esta difícil labor.

A todos ustedes, muchas gracias.

Kevin

Tabla de Contenido

Índice de Tablas

Índice de Ilustraciones

Capítulo 1

Introducción

1.1. Motivación y Contexto

Durante los últimos tiempos, el paradigma de PHM (Prognostic Health Management) ha cobrado fuerza como solución para el mantenimiento de maquinaria industrial, gracias a los numerosos avances en Inteligencia Artificial (IA) de los que esta disciplina se vale [1]. PHM puede definirse como un campo de la ingeniería cuyo objetivo es entregar al usuario un análisis exhaustivo de la condición de salud de una máquina y sus componentes, lo que incluye estado actual y pronóstico, para estimar el tiempo de vida remanente, o RUL (Remaining Useful Life), todo esto con el fin de apoyar los procesos de toma de decisiones [2]. La toma de decisiones (Decision Making, DM), en este contexto, implica la utilización de información proporcionada por PHM para realizar acciones correctivas o preventivas antes de la ocurrencia de fallos, permitiendo así evitar reparaciones costosas y tiempos de inactividad extendidos [3, 4, 5]. Además, PHM facilita la organización eficaz de actividades de mantenimiento simultáneo, reduciendo significativamente los períodos en los cuales la maquinaria está fuera de operación y optimizando la producción, especialmente en escenarios críticos como en la industria minera, donde la paralización de un solo elemento puede detener todo el proceso productivo, según lo conversado con expertos en el área. Incluso, más importante aún que la optimización operativa y la reducción del impacto económico negativo, el correcto empleo de PHM y DM puede ser muy útil pare prevenir accidentes que afecten la salud, en incluso la vida de las personas involucradas en el manejo de la maquinaria, proporcionando así un entorno de trabajo más seguro.

Aunque en PHM es habitual enfocarse en la precisión de la estimación del tiempo de falla, no es tan común que se discuta sobre el costo computacional de los algoritmos de pronóstico y su tiempo de ejecución [6]. Esta consideración es crucial, ya que numerosas aplicaciones requieren de la ejecución de algoritmos de pronóstico en tiempo real, necesitando actualizaciones constantes durante la operación sin disponer de equipos de alto poder computacional, como es el caso en bicicletas eléctricas [7] o en vehículos aéreos no tripulados (UAM, Unmanned Aerial Mobility) [8]. En contextos como estos, puede resultar conveniente en determinados momentos del ciclo operativo el sacrificar un grado de precisión, a cambio de reducir el tiempo de cálculo. Por lo tanto, se necesita encontrar un equilibrio entre eficacia

y eficiencia en los algoritmos de pronóstico.

Un caso de estudio interesante en PHM al estudio de baterías de Ión Litio (Ion-Li a partir de ahora) para aplicaciones de electromovilidad. Durante las últimas décadas, el cambio climático ha pasado a ser una preocupación a escala global, alcanzando hoy en día un consenso prácticamente universal en la comunidad científica [9]. Una de las principales causas de este fenómeno es justamente el alto nivel de emisiones de carbono a partir de la época industrial hasta la actualidad [10, 11], como consecuencia de un uso generalizado de combustibles fósiles para propósito de transporte, entre otros factores [12]. Por estas razones, y como respuesta a una creciente preocupación ambiental, la industria del transporte ha reaccionado impulsando la búsqueda de fuentesde energía alternativas, siendo una de las exploradas más exhaustivamente el área de electromovilidad [13].

Al igual que cualquier dispositivo electrónico, desde artefactos de uso cotidiano como laptops o *smartphones*, hasta equipamiento especializado como robots o satélites, la autonomía de un vehículo eléctrico (EV, por sus siglas en inglés) requiere de una fuente de alimentación, que es típicamente un dispositivo de almacenamiento de energía (ESD, Energy Storage Device). Hoy en día, el Ión-Litio (Ion-Li a partir de ahora), presenta varias ventajas por sobre otros compuestos químicos para el propósito de fabricación de EDS's, como por ejemplo Ni-MH (níquel-metalhidruro) de algunas pilas recargables, Ni-Cd o plomo [14]. Algunas de las cualidades de las ESDs basadas en Ion-Li que las hacen preferibles por sobre otros materiales para la industria automotriz son una gran densidad de carga por unidad de masa (o volumen), permitiendo diseñar celdas compactas, un ciclo de vida más extenso, ausencia de efecto memoria, y una tasa limitada de auto-descarga [15, 7]. Esto explica la alta demanda de litio registrada en los últimos años a nivel mundial, y que se espera que se siga incrementando durante esta década [16, 17], justamente para la manufactura de baterías, en un esfuerzo de masificar la electromovilidad y tratar de convertirlo en el estándar en la industria del transporte [13].

El aumento en popularidad de los ESDs en base a Ion-Li en la industria automotriz ha llevado al desarrollo de sistemas de monitoreo y gestión de baterías (BMS, Battery Management Systems) [15] dentro de la disciplina de PHM. Estos sistemas a su vez integran algoritmos que entregan información sobre el estado de carga de la batería (SOC, State of Charge), estado de máxima potencia disponible (SoMPA, State of Maximum Power Available), y estado de salud (SOH, State of Health). El objetivo de un BMS es monitorear un cierto número de variables de interés de la batería y proveer información en tiempo real sobre ella, como por ejemplo la corriente de entrada o salida, voltaje y temperatura de operación [7, 18], o maximizar su tiempo de uso, asegurando la autonomía del EV, entre otras. Un BMS debe emplear información sobre estas variables para estimar el SOC, SoMPA, SOH en incluso pronosticar el fin de descarga (EoD, End of Discharge).

En la literatura pueden encontrarse varias definiciones para el SOC, por tanto es necesario remarcar que en este trabajo se considera como la energía remanente en la batería, medido como el porcentaje con respecto a la máxima capacidad de la celda [19], y no como la energía disponible que ésta es capaz de entregar, lo cual es impreciso de acuerdo a estudios recientes [20]. Por otro lado, el SoMPA se define como la máxima potencia que puede ser extraída o inyectada a la batería, sin violar la zona de operación segura (SOA, Safe Operating Area)

[21]. Finalmente, el EoD corresponde al instante en el cual la batería falla debido a que al menos uno de sus parámetros se encuentra fuera de la SOA. En un caso general de falla de sistemas, no relacionados a descarga de baterías, este instante se conoce como el tiempo de falla (ToF, Time of Failure). En este trabajo se analiza un caso de estudio en donde la condición de falla corresponde a la insuficiencia de potencia, es decir, el EV exige una potencia superior a la máxima que la batería puede entregar en un instante determinado, que corresponde justamente al SoMPA. El SoMPA, depende del estado de carga, el SOC, por tanto, para identificar o predecir una falla por insuficiencia de potencia, es necesario primero conocer esta variable.

Diversos métodos se han reportado para estimación del SOC: (i) contador de Ampere-hora, (ii) medición de voltaje en circuito abierto (OCV, Open Circuit Voltage), (iii) espectroscopía electroquímica de impedancia (EIS, Electrochemical Impedance Sprectroscopy), y (iv) métodos basados en modelamiento de la batería [7, 19]. Sin embargo, algunos de estos métodos presentan desventajas importantes. El contador de Ampere-hora requiere de sensores de alto costo, debido a su alta sensibilidad a errores de medición; OCV sólo puede emplearse en aplicaciones en las cuales la batería pasa largos períodos de tiempo en reposo, haciendo que no sea conveniente en electromovilidad; y finalmente, EIS es más apropiado para situaciones en que la temperatura se mantiene constante, pues la impedancia interna de la batería depende de esta variable. Por otro lado, las estrategias basadas en modelos empíricos son flexibles, permitiendo trabajar con data limitada y ruidosa [14], caracterización apropiada de las fuentes de incertidumbre, y lo más importante, pueden implementarse en algoritmos en tiempo real. En esta categoría de métodos es posible mencionar enfoques con Lógica Difusa, Redes Neuronales, y Procesadores Bayesianos. El presente trabajo está enfocado en esta última metodología, específicamente en algoritmos de pronóstico basados en filtro de partículas (PF, Particle Filter), considerados actualmente el estado del arte dentro de la comunidad de PHM para pronóstico basado en modelos [22].

1.2. Presentación del problema

Ya que el proceso de descarga de una batería posee una naturaleza estocástica, influenciado por diversas fuentes de incertidumbre al adoptar un enfoque Bayesiano del problema surge la necesidad de actualizar constantemente el pronóstico del EoD, integrando nueva información obtenida a través de mediciones realizadas durante el ciclo de descarga. Tal como se mencionó al comienzo, esto podría resultar enormemente problemático si es que el costo computacional del algoritmo de pronóstico es alto, ya sea en términos de tiempo de ejecución, o como ocurre aveces, en consumo de energía. Actualmente, un algoritmo de pronóstico en tiempo real basado en filtro de partículas puede tomar algunos minutos en generar una predicción, o incluso horas si se requiere un alto grado de convergencia estocástica [6], lo que haría imposible mantener una tasa de actualización apropiada en algunas situaciones [23]. En términos de energía, el mismo algoritmo podría gastar una cantidad no despreciable de ella, lo que en el caso del problema de pronóstico de EoD podría llegar a perturbar el ciclo de descarga de la batería bajo supervisión. Además, un BMS cuya tasa de actualización de pronóstico restringida por estas condiciones podría estar impedido de predecir con la suficiente antelación un inminente evento de falla catastrófica, en especial si durante la ejecución del algoritmo el sistema sufre súbitamente una perturbación en sus condiciones de operación. Esto, junto

con el esfuerzo computacional de los algoritmos de toma de decisiones [24], significa que el usuario podría estar incapacitado para llevar a cabo una acción preventiva, provocando una inesperada interrupción de la misión en curso.

Para abordar este problema, este trabajo propone un enfoque para el diseño eficiente de algoritmos de pronóstico basados en filtro de partículas, que si bien aquí se prueba en un caso de estudio de descarga de baterías de Ion-Li, en principio es aplicable a cualquier sistema con un modelo de espacio de estado continuo, no lineal, de tiempo discreto, y que satisface la propiedad de Markov. Con respecto a la incertidumbre del modelo, la formulación aquí presentada supone que ésta puede representarse mediante un ruido de proceso Gaussiano y puramente aditivo, aunque eventualmente podría extenderse para una mayor variedad de casos. Mientras que un algoritmo convencional propaga el estado del sistema en el tiempo un instante a la vez, es decir, si se comienza en el instante k_0, el algoritmo calcula el estado en el instante $k_0 + 1$, luego $k_0 + 2$, y así sucesivamente. El problema con esta metodología es que, si el horizonte de predicción es demasiado extenso, podría implicar también un largo tiempo de ejecución. Por tanto, la estrategia que se plantea aquí consiste en lo siguiente: en vez de propagar el estado un instante de tiempo a la vez, se propone hacerlo mediante "saltos de tiempo" de varios instantes, es decir, si se inicia en k_0, se procede a calcular el estado en el instante $k_0 + p$, luego $k_0 + 2p$, y así sucesivamente, con p un entero positivo, sin considerar de esta forma los valores intermedios del estado. Mediante este enfoque, un extenso horizonte de predicción podría ser eventualmente cubierto en una ventana de tiempo de ejecución mucho más breve, reduciendo drásticamente el esfuerzo computacional del algoritmo. Sin embargo, la implementación de dicha estrategia requiere en primer lugar una expresión matemática que relacione el estado a un instante $k_0 + p$, con el estado en un momento previo k_0. Para ello, como se explica más adelante, es necesario comenzar linealizando el modelo del sistema.

1.3. Hipótesis

Usando una frecuencia de muestreo menor a la original, es posible diseñar algoritmos de pronóstico computacionalmente eficientes basados en filtro de partículas, cuyo tiempo de ejecución sea drásticamente inferior al de un algoritmo tradicional, al punto de ser viables para aplicaciones en tiempo real en que el ciclo de vida sea breve. Esto requiere de la existencia de una expresión matemática que describa el sistema a una frecuencia de muestreo distinta al modelo original; dicha expresión debe ser manejable, posible de derivar analíticamente, y además debe aproximar correctamente el comportamiento del sistema, de forma de que pueda usarse para el pronóstico. Por otro lado, se requiere también de un mecanismo que pueda seleccionar, en tiempo real, una frecuencia de muestreo que otorgue un balance entre exactitud del pronóstico, tomando como referencia un algoritmo tradicional, y minimizar el tiempo de cómputo. De esta forma, se probaría que es provechosa la exploración de frecuencias de muestreo alternativas a la estándar.

1.4. Objetivos

1.4.1. Objetivo general

Desarrollar una metodología para obtener un esquema de pronóstico para sistemas no lineales, basado en filtro de partículas, que sea eficiente computacionalmente gracias a una reducción de la frecuencia de muestreo, y sin mayores pérdidas de precisión en la estimación del tiempo de falla.

1.4.2. Objetivos específicos

1. Derivar analíticamente una expresión matemática compacta y manejable que relacione directamente el estado del sistema en el instante actual y su valor p muestras más en el futuro, representando así una reducción de la frecuencia de muestreo. Esta expresión debe ser capaz de replicar la dinámica original del sistema.

2. Obtener, mediante algoritmos genéticos, un conjunto de datos de tasas de muestreo óptimas, según algún criterio previamente establecido, para diferentes condiciones iniciales del sistema estudiado, y restricciones de tiempo de ejecución.

3. Diseñar un sistema basado en modelos XGBoost capaz de determinar, en tiempo real, y en función del estado actual del sistema, una frecuencia de muestreo apropiada para estimar el instante de descarga en un tiempo de cómputo menor a un máximo establecido por el usuario.

4. Comparar resultados de estimación del tiempo de falla del enfoque propuesto con un algoritmo tradicional, para un rango amplio de condiciones iniciales y límites máximos de tiempo de cómputo.

1.5. Estructura de Tesis

Este documento sigue la siguiente estructura: en el Capítulo 2 se presenta un resumen del estado del arte, se exponen los fundamentos teóricos de las principales herramientas matemáticas utilizadas, así como los algoritmos más importantes empleados, además de algunos antecedentes del problema a abordar; el Capítulo 3 presenta el caso de estudio analizado, junto con su modelo de espacio de estado y condición de falla; el Capítulo 4 describe la metodología seguida en esta investigación, tanto en la parte teórica como práctica; en el Capítulo 5 se muestran los principales resultados de la investigación. Finalmente, se presentan las Conclusiones de este trabajo.

Capítulo 2

Marco Teórico

2.1. Estado del Arte

Durante los últimos años, se han reportado varios esfuerzos enfocados en reducir la cantidad de recursos computacionales empleados en pronóstico de fallas. Por ejemplo, en [23] se propone una estrategia enfocada en disminuir la cantidad de veces que se actualiza el pronóstico durante la operación del sistema bajo monitoreo, cada una requiriendo una nueva ejecución el algoritmo correspondiente. Por una parte, el paradigma convencional recomienda efectuar una actualización cada vez que se obtiene una nueva medición, lo que sería altamente costoso en términos de recursos computacionales. En contraste, en [23] se sugiere que una actualización sólo es necesaria cuando la discrepancia entre la distribución de probabilidad del indicador de falla estimada por el pronóstico más reciente, y aquella obtenida en la etapa de filtrado, supera un cierto umbral de tolerancia definido por el usuario. Otro estudio formaliza teóricamente el problema de EoD, empleando una función de verosimilitud de evento para el indicador de falla [6], método propuesto en [25] en contraposición al umbral fijo que se suele emplear [26]. Además, los autores proponen un método numérico basado en dicha estrategia, cuyos resultados muestran una disminución significativa de la carga computacional, en comparación con el método usual de cruce de umbral. Por otro lado, también se ha propuesto el uso de unidades gráficas de procesamiento, o GPU, con el fin de acelerar los algoritmos de pronóstico [27].

2.2. Filtro de Partículas

Filtrado Bayesiano (BF, Bayesian Filtering) es un marco metodológico matemático empleado para estimar recursivamente la función de densidad de probabilidad (PDF, Probability Density Function) *a posteriori* del estado x_k de un sistema dinámico, a cualquier instante k, dadas las mediciones $z_{1:k}$ hasta el mismo instante k [28]. En la práctica, los sistemas que se estudian suelen ser no lineales y no Gaussianos, lo que implica que la solución óptima al problema de BF no puede ser calculada analíticamente. [29].

Una clase de algoritmos eficaces y ampliamente usados para obtener una solución subóptima al problema de BF son los algoritmos de Filtro de Partículas (PF, Particle Filter).

Estos algoritmos basados en simulaciones de Monte Carlo buscan representar la PDF *a posteriori* del estado objetivo de la estimación como un conjunto de $N_p \gg 1$ muestras aleatorias, denominadas *partículas*, cada una con un *peso* asociado, como se define en la Ec. (2.1):

$$\left\{ \boldsymbol{x}_k^{(\mathrm{i})}, w_k^{(\mathrm{i})} \right\}_{\mathrm{i}=1}^{N_p}, \sum_{\mathrm{i}=1}^{N_p} w_k^{(\mathrm{i})} = 1. \tag{2.1}$$

Esta colección de partículas se obtiene de manera secuencial a partir de una PDF alternativa, denominada *distribución de importancia*, que se denota por $q(\cdot)$. Esto significa que la PDF *a posteriori* de $\boldsymbol{x}_k$ es representada por la Ec. (2.2) [30]:

$$p(x_k|\boldsymbol{z}_{1:k}) = \sum_{\mathrm{i}=1}^{N_p} w_k^{(\mathrm{i})} \delta(x_k - x_k^{(\mathrm{i})}), \tag{2.2}$$

mientras que los pesos $w_k^{(\mathrm{i})}$ se actualizan de acuerdo a la Ec (2.3):

$$w_k^{(\mathrm{i})} = w_{k-1}^{(\mathrm{i})} \frac{p\left(\boldsymbol{z}_k|x_k^{(\mathrm{i})}\right) p\left(x_k^{(\mathrm{i})}|x_{k-1}^{(\mathrm{i})}\right)}{q\left(x_k^{(\mathrm{i})}|x_{k-1}^{(\mathrm{i})}, z_k\right)}. \tag{2.3}$$

En la implementación más básica de PF, *Sequential Importance Sampling* (SIS) [30], la distribución de importancia se escoge para que sea igual a la PDF *a priori* del estado, esto es lo que se muestra en Ec. (2.4):

$$q\left(x_k|x_{k-1}^{(\mathrm{i})}, \boldsymbol{z}_k\right) = p\left(x_k|x_{k-1}^{(\mathrm{i})}\right), \tag{2.4}$$

lo que significa que, dada la expresión en la Ec. (2.3), el cálculo de actualización de los pesos requiere únicamente de la verosimilitud de las nuevas observaciones, $p\left(\boldsymbol{z}_k|x_k^{(\mathrm{i})}\right)$.

2.3. Pronóstico de fallas basado en filtro de partículas

En PHM, los algoritmos de pronóstico de fallas son métodos dirigidos a generar predicciones a largo plazo de algún indicador de falla, y caracterizar futuras fuentes de incertidumbre que pudieran afectar su evolución. Todo esto, con el propósito de estimar el tiempo de vida útil remanente, o RUL (Remaining Useful Life) del componente que puede fallar o de algún subsistema bajo monitorización [31]. En general, los módulos de BF se usan para estimar una condición inicial apropiada para estas predicciones a largo plazo. Sin embargo, la necesidad de, además de esto, determinar la secuencia de PDFs más probable del indicador de falla durante la evolución del sistema conlleva desafíos adicionales para el enfoque convencional de BF. Una solución en tiempo real ampliamente aceptada para esta clase de problemas dentro de la comunidad de PHM está precisamente basada en los algoritmos de PF previamente

descritos. Esto permite trabajar con modelos no lineales para la degradación del sistema, y fuentes de incertidumbre no Gaussianas [14], haciendo que sea posible extender la aplicación de esta estrategia a una variedad de problemas, como propagación de fractura en materiales, degradación de rodamientos, o sistemas de supervisión de baterías, entre otros ejemplos [23].

Si bien en estudios previos relacionados [7, 23] se ha considerado una formulación matemática de los algoritmos de pronósticos desarrollada en [32, 33], aquí se enuncia, de forma resumida, una versión incluso más general, desarrollada detalladamente en [25] y empleada en [6], que no asume algunos supuestos como independencia estadística, y un umbral fijo para la condición de falla. En primer lugar, se considera un evento de falla ε que puede ocurrir en el sistema bajo estudio:

$$\varepsilon = \textit{“Falla catastrófica del sistema”}. \tag{2.5}$$

Además, se define un proceso estocástico binario $\{E_k\}_{k\in\mathbb{N}}$, dependiente del estado $x(k)$, que a cada instante k indica la ocurrencia o no del evento ε con una cierta probabilidad. De esta forma, se tiene:

$$\mathbb{P}(E_k = \varepsilon) = 1 - \mathbb{P}(E_k = \varepsilon^c), \tag{2.6}$$

donde ε^c denota la no ocurrencia del evento ε. Sea también $k_p \in \mathbb{N}$ un instante de tiempo en el que se desea efectuar una predicción de la primera ocurrencia de ε, momento que se representa por $\tau_\varepsilon = \tau_\varepsilon(k_p)$. Este instante se puede definir formalmente como se muestra en la Ec. (2.7):

$$\tau_\varepsilon(k_p) \coloneqq \text{ínf}\{k \in \mathbb{N} : \{k > k_p\} \wedge \{E_k = \varepsilon\}\}. \tag{2.7}$$

A su vez, la función de probabilidad de masa de $\tau_\varepsilon(k_p)$, $\mathbb{P}(\tau_\varepsilon = k)$, se puede expresar como la probabilidad de que el evento de falla ocurra en el instante k, pero sin que haya ocurrido desde k_p hasta $k-1$, es decir [25]:

$$\begin{aligned}
\mathbb{P}(\tau_\varepsilon = k) &= \mathbb{P}\Big(\{E_k = \varepsilon\}, \{E_j = \varepsilon^c\}_{j=k_p}^{k-1}\Big) \\
&= \mathbb{P}\Big(\{E_k = \varepsilon\} | \{E_j = \varepsilon^c\}_{j=k_p}^{k-1}\Big)\mathbb{P}\Big(\{E_j = \varepsilon^c\}_{j=k_p}^{k-1}\Big) \\
&= \mathbb{P}\Big(\{E_k = \varepsilon\} | \tau_\varepsilon \geq k\Big) \prod_{j=k_p+1}^{k-1} \mathbb{P}\Big(\{E_k = \varepsilon^c\} | \tau_\varepsilon \geq j\Big),
\end{aligned} \tag{2.8}$$

donde se han usado recursivamente las propiedades de la probabilidad de intersección de sucesos y probabilidad condicional. Además, la estadística de $\tau_\varepsilon(k_p)$ depende además de la trayectoria de algunas variables del sistema bajo estudio, como pueden ser el estado $x(k)$, la entrada exógena $u(k)$, y el ruido de medición $\eta(k)$, desde el instante k_p+1 hasta k, por lo que,

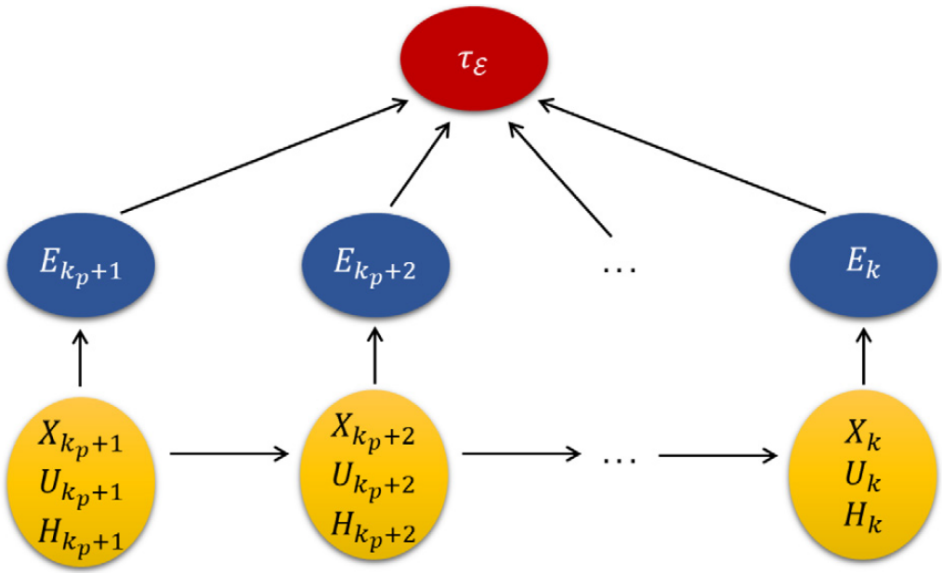


Figura 2.1: Diagrama de la dependencia estadística de las variables relevantes para calcular la distribución de probabilidad de falla en el tiempo. El tiempo de falla τ_ε depende del proceso estocástico E_k, que indica la ocurrencia o no del evento de falla ε a cada instante k, mientras que a su vez E_k depende únicamente de las variables del sistema, el estado x_k, la entrada exógena u_k, y el ruido de proceso η_k, en el instante k. En esta Figura, H_k corresponde al espacio del ruido, $\eta(k)$. (Figura tomada de [6]).

en estricto rigor, la expresión para la probabilidad $\mathbb{P}(\tau_\varepsilon = k)$ se debe descomponer también en una integral sobre la densidad de probabilidad conjunta de $x_{k_p+1:k}$, $u_{k_p+1:k}$ y $\eta_{k_p+1:k}$. Sin embargo, se debe considerar además que la probabilidad de E_k depende únicamente de las variables del sistema en el mismo instante k, x_k, u_k y η_k. En resumen, la estadística de $\tau_\varepsilon(k_p)$ depende de E_k, que a su vez depende de x_k, u_k y η_k, como se ilustra en el diagrama de la Fig. 2.1. Esto simplifica considerablemente la expresión analítica para $\mathbb{P}(\tau_\varepsilon = k)$, quedando finalmente como se muestra en la Ec. (2.9) [6].

$$\begin{aligned}\mathbb{P}(\tau_\varepsilon = k) = \int_{\mathbb{X}_{k_p+1:k}} \int_{\mathbb{U}_{k_p+1:k}} \int_{\mathbb{H}_{k_p+1:k}} \mathbb{P}(E_k = \varepsilon | x_k, u_k, \eta_k) \dots \\ \dots \prod_{j=k_p+1}^{k-1} \Big(1 - \mathbb{P}(E_j = \varepsilon | x_j, u_j, \eta_j)\Big) \dots \\ \dots p(x_{k_p+1:k}, u_{k_p+1:k}, \eta_{k_p+1:k}) \mathrm{d}\eta_{k_p+1:k} u_{k_p+1:k} x_{k_p+1:k}.\end{aligned} \tag{2.9}$$

En [34] se puede encontrar una verificación práctica de este enfoque, incluyendo fragmentos de código para mostrar su implementación.

Una forma de aproximar la integral de la Ec. (2.9) es justamente mediante métodos basados en Monte Carlo, en este caso, filtro de partículas. Sin embargo, un problema relevante que surge al implementar algoritmos de pronóstico basados en PF es cómo determinar los pesos $w_k^{(\mathrm{i})}, \mathrm{i} = 1, \dots, N_p$ de cada partícula. Una descripción exhaustiva de tres métodos existentes

para ello puede encontrarse en [31]. El enfoque más común y sencillo consiste en simplemente usar los mismos pesos de la etapa de filtrado, y mantenerlos constantes al generar la predicción a largo plazo. De ser así, la distribución de probabilidad de las trayectorias de las variables queda aproximada por una suma de N_p funciones delta de Dirac, cada una correspondiente a una de las N_p partículas, como se expresa en la Ec. (2.10):

$$p(x_{k_p+1:k}, u_{k_p+1:k}, \eta_{k_p+1:k}) \approx \frac{1}{N_p} \sum_{\mathrm{i}=1}^{N} \delta_{x^{(i)}_{k_p+1:k}, u^{(i)}_{k_p+1:k}, \eta^{(i)}_{k_p+1:k}}(x_{k_p+1:k}, u_{k_p+1:k}, \eta_{k_p+1:k}). \tag{2.10}$$

Por otro lado, si se considera como condición de falla el cruce de un umbral, la función de probabilidad $\mathbb{P}(E_k = \varepsilon | x_k, u_k, \eta_k)$ se convierte en una indicatriz:

$$\mathbb{P}(E_k = \varepsilon | x_k, u_k, \eta_k)) = \mathbb{1}_{\{(x_k, u_k, \eta_k): g(x_k, u_k, \eta_k) \leq g^*\}}(x_k, u_k, \eta_k), \tag{2.11}$$

es decir, la probabilidad de que ocurra el evento de falla es 1 si $g(x_k, u_k, \eta_k) \leq g^*$, y 0 en caso contrario, con $g(x_k, u_k, \eta_k)$ alguna función de las variables del sistema que define la zona segura de operación, el SOA en el caso de una batería, y g^* el umbral de falla. Al insertar la Ec. (2.10) y (2.11) en la Ec. (2.9), se tiene la siguiente estimación de $\mathbb{P}(\tau_\varepsilon = k)$:

$$\hat{\mathbb{P}}_{N_p}(\tau_\varepsilon = k) = \frac{1}{N_p} \sum_{\mathrm{i}=1}^{N_p} \mathbb{1}(x_k^{(i)}, u_k^{(i)}, y_k^{(i)}) \prod_{j=k_p+1}^{k-1} \Big(1 - \mathbb{1}(x_k^{(i)}, u_k^{(i)}, y_k^{(i)})\Big). \tag{2.12}$$

Finalmente, una observación esencial a la Ec. (2.12) es que de ella se desprende que la función de probabilidad $\mathbb{P}(\tau_\varepsilon = k)$ se puede aproximar de la forma:

$$\mathbb{P}(\tau_\varepsilon = k) \approx \frac{\text{N° trayectorias en cruzar por primera vez el umbral en instante } k}{\text{N° total de trayectorias simuladas}}. \tag{2.13}$$

De esta manera, se vuelve sencillo el construir un histograma para representar la función de probabilidad de masa de τ_ε, que corresponde a la metodología adoptada en esta investigación.

Ya que los resultados de la etapa de filtrado se emplean como condición inicial para el algoritmo de pronóstico, en principio es posible actualizar la estimación del estado actual cada vez que se obtiene una nueva medición, y por tanto, también el pronóstico mismo. Esto permitiría mejorar la caracterización para la probabilidad del tiempo de falla, gracias a la nueva información disponible y la disminución del horizonte de predicción. Sin embargo, estos cálculos son costosos en términos de procesamiento, lo que es precisamente el problema que esta investigación pretende abordar.

2.4. Cadenas de Markov

Las cadenas de Markov son modelos estocásticos que describen una secuencia de posibles eventos en la cual la probabilidad de cada suceso depende exclusivamente del estado alcanzado en el evento previo [7]. Una cadena de Markov se define por un conjunto de estados y probabilidades de transición entre ellos. El número de estados m de un modelo de cadena de Markov se puede determinar por métodos heurísticos u otras herramientas de probabilidad y estadística, mientras que las probabilidades de transición se pueden calcular mediante estimadores de máxima verosimilitud. Generalmente, las probabilidades de transición de una cadena de Markov son representadas en una matriz cuadrada de tamaño $m \times m$, denominada *matriz de transición*, como se muestra en la Ec (3.2), donde m es el núméto de estados de la cadena.

$$\mathbf{P} = \begin{pmatrix} p_{1,1} & \cdots & p_{1,m} \\ \vdots & \ddots & \vdots \\ p_{m,1} & \cdots & p_{m,m} \end{pmatrix}. \tag{2.14}$$

Aquí, la componente $p_{\mathrm{i},j} = P(X_{k+1} = j | X_k = \mathrm{i})$ de la matriz representa la probabilidad condicional de que la cadena se encontrará en el j-ésimo estado al instante $k+1$, dado que en el instante previo k se haya encontrado en el i-ésimo estado [35].

Al contrario de lo que se podría pensar, el uso de un mayor número de estados para un modelo de cadena de Markov no implica necesariamente una mejora en la caracterización del fenómeno que se busca describir, incluyendo el perfil de corriente de descarga de una batería. Esto se debe a que más estados significa que se cuenta con menos datos disponibles para estimar las probabilidades de transición, y por tanto, la calidad de los estimadores de éstas empeora. La precisión de dichos estimadores se puede evaluar mediante las cotas mostradas en las Ec. (2.15) y (2.16).

$$P\Big\{ |\hat{p}_{\mathrm{i},j} - p_{\mathrm{i},j}| > t \Big\} \leq \frac{1}{4nt^2} \tag{2.15}$$

$$P\Big\{ |\hat{p}_{\mathrm{i},j} - p_{\mathrm{i},j}| > t \Big\} \leq 2\exp\big(-2nt^2\big), \tag{2.16}$$

donde $\hat{p}_{\mathrm{i},j}$ es el estimador de máxima verosimilitud para $p_{\mathrm{i},j}$, definido como:

$$\hat{p}_{\mathrm{i},j} = \frac{n_{\mathrm{i},j}}{\sum_{j=1}^{N} n_{\mathrm{i},j}}. \tag{2.17}$$

Las ecuaciones (2.15) y (2.16) representan las cotas para la probabilidad de obtener una estimación de máxima verosimilitud $P_{\mathrm{i},j}$ que difiera de la probabilidad de transición real en una cantidad mayor a t. El máximo número de estados de un modelo de cadena de Markov está sujeto a los parámetros de diseño $p^* \in (0,1)$ y $t^* \in (0,1)$, los cuales son escogidos por el usuario. Estos parámetros se relacionan mediante la Ec. (2.18), donde p^* denota la

máxima probabilidad que el usuario está dispuesto a aceptar tal que la diferencia entre $p_{\mathrm{i},j}$ y su estimador es mayor o igual a t.

$$P\Big\{|\hat{p}_{\mathrm{i},j} - p_{\mathrm{i},j}| > t\Big\} \leq p^*. \tag{2.18}$$

Esta investigación considera los mismos dos estados, niveles de corriente de descarga en este caso, que fueron obtenidos en [7] por medio de k-means *clustering*, mientras que las probabilidades de transición fueron calculadas por estimadores de máxima similitud, metodología presentada en [36].

Finalmente, ya que la propuesta de esta investigación consiste en el uso de saltos de tiempo, un último aspecto sobre las cadenas de Markov que debe ser mencionado es cómo calcular probabilidad de transición entre estados no consecutivos, es decir, estados en los instantes k y $k+n$, con $n > 1$ un entero. Usando las propiedades de probabilidad condicional, se puede demostrar que si $\mathbf{P}$ es la matriz de transición de la cadena de Markov, entonces las probabilidades de transición entre estados separados por n pasos corresponden a los componentes de la matriz $\mathbf{P}^n$ [35]. Esta propiedad permite que algoritmo de pronóstico aquí propuesto no sólo ignore estados intermedios del sistema bajo monitorización, sino también los valores intermedios de la entrada exógena, la corriente de descarga de la batería en este caso, entre el instante actual k_p y un instante futuro $k_p + n$.

2.5. Algoritmos Genéticos

Los algoritmos genéticos (GA, Genetic Algorithms) son métodos de búsqueda para resolver problemas de optimización, inspirados en la genética y selección natural que guía el proceso biológico de evolución. Pertenece a la familia de algoritmos metaheurísticos basados en población, en particular al grupo de algoritmos evolutivos, donde una colección de soluciones candidatas son modificadas de manera iterativa, con el fin de alcanzar una solución óptima, ya sea global o localmente, del problema abordado [37].

En los GA, cada solución candidata, denominada individuo, corresponde a un punto en el espacio de búsqueda, codificado en un *string* denominado cromosoma, tradicionalmente conformado por dígitos binarios 0 y 1, aunque otras opciones son también posibles, por ejemplo, codificación real o entera. En general, a cada elemento del cromosoma se le conoce como un gen. El desempeño de un cromosoma se evalúa por una función de *fitness*, que usualmente corresponde a la misma función que se busca optimizar mediante el GA. Esto permite seleccionar los individuos más aptos para crear nuevos de manera iterativa, y de esa forma mejorar la calidad de las soluciones candidatas.

En cada generación, la población de cromosomas atraviesa un proceso para mejorar su desempeño, comenzando por una inicialización aleatoria de cada individuo. El proceso consiste principalmente en la aplicación de tres operadores genéticos: selección, *crossover*, y mutación. El operador de selección escoge de manera aleatoria los individuos con las mejores puntuaciones, de acuerdo a la función de *fitness*, donde los cromosomas con un valor más alto (si es un problema de maximización) tienen una mayor probabilidad de ser escogidos para la etapa de reproducción. Algunos métodos existentes para el operador de selección son el método de

la ruleta, donde la probabilidad de cada cromosoma de ser seleccionado es proporcional a su valor de *fitness*, o por torneo, en el cual se escogen al azar varios pares de cromosomas, en cada par ambos cromosomas compiten, y aquel que cuente con la mejor puntuación es seleccionado para reproducirse. Esta reproducción se lleva a cabo mediante el operador de *crossover*, que consiste en mezclar los genes de individuos de la generación previa, produciendo así una descendencia. También existen diversas variantes en la operación de *crossover*, por ejemplo, de punto único, de dos puntos o k puntos, dependiendo de en cuántos segmentos se divida cada cromosoma antes de ser combinado con otros. Otra alternativa es el *crossover* uniforme, que decide aleatoriamente si cada gen debe ser intercambiado con uno de la misma ubicación de otro cromosoma. Finalmente, el operador de mutación produce cambios al azar de los genes del cromosoma, lo que permite explorar más exhaustivamente el espacio de búsqueda, evitando así un estancamiento en óptimos locales.

Para finalizar la ejecución de un GA existen varias alternativas de criterio de término. Entre otras, es posible mencionar: alcanzar una solución que satisface ciertos requerimientos mínimos, una cota máxima de generaciones, o un estancamiento en la puntuación de las mejores soluciones luego de una cantidad determinada de generaciones.

Una revisión exhaustiva de GA, con más detalles acerca de cada aspecto, está disponible en [37].

2.6. XGBoost

Extreme Gradient Boost, abreviado como XGBoost, es un algoritmo clásico de aprendizaje de máquinas, o *machine learning*, para resolver problemas de *Gradient Tree Boosting*. *Boosting* es una técnica de *ensemble learning*, es decir una combinación de *weak learners*, predictores o clasificadores usualmente sencillos y de bajo rendimiento, como árboles de decisión en el caso de *tree boosting*, pero que en combinación con otros pueden construir un modelo robusto. *Boosting* es un caso específico de *ensemble learning*, en el cual los miembros del modelo se agregan de manera secuencial, corrigiendo los errores de los integrantes previos, y la salida final es un promedio ponderado de todos ellos. Se ha demostrado que *Tree Boosting* entrega resultados de estado del arte para una variedad de problemas de regresión y clasificación en diferentes áreas [38], de acuerdo a varios estándares de clasificación. XGBoost es una extensión del algoritmo de *tree boosting* tradicional, ampliamente reconocido en una variedad de desafíos de minería de datos y aprendizaje de máquinas, resolviendo problemas en comportamiento de clientes, detección de movimiento, clasificación de eventos en física de altas energías, predicción de riesgos, entre otros [39].

La virtud más importante de XGBoost es la escalabilidad, siendo más de 10 veces más rápido que otras soluciones populares que funcionan en una única máquina que no ejecute computación paralela. Esto se debe a varias optimizaciones tanto algorítmicas como de sistema, como por ejemplo una estructura de bloques para aprendizaje fuera de núcleo que tenga en cuenta la data almacenada en memoria caché, computación paralela y distribuida, un algoritmo que considera ausencia de algunas entradas en los datos, y una estrategia para encontrar el punto de separación en cada árbol de decisión basada en cuantiles ponderados y justificada teóricamente. Además, XGBoost incluye un término de regularización a la típica función de costo de *gradient boosting*, y lleva a cabo la optimización mediante una

aproximación de Taylor de segundo orden [39].

La predicción $\hat{y}_\mathrm{i}$ entregada por un modelo XGBoost para un único dato $\mathbf{x}_\mathrm{i}$ viene dada por la Ec. (2.19):

$$\hat{y}_\mathrm{i} = \phi(\mathbf{x}_\mathrm{i}) = \sum_{k=1}^{K} f_k(\mathbf{x}_\mathrm{i}), \quad f_k \in \mathcal{F}, \tag{2.19}$$

donde K es el número de árboles, y $\mathcal{F}$ es el espacio de los árboles de regresión. Por otro lado, la función objetivo regularizada se muestra en la Ec. (2.20) y Ec. (2.21).

$$\mathcal{L}(\phi) = \sum_{\mathrm{i}} l(y_\mathrm{i}, \hat{y}_\mathrm{i}) + \sum_{k} \Omega(f_k) \tag{2.20}$$

$$\Omega(f_k) = \gamma T + \frac{1}{2}\lambda \|w\|^2. \tag{2.21}$$

Aquí, $\Omega(f_k)$ es el término de regularización, con T el número de hojas en cada árbol, y w el vector que contiene los resultados de cada hoja, mientras que γ y λ son parámetros definidos por el usuario. El rol de este término, junto con la opción de efectuar sub-muestreo de columnas o características (*features*), es evitar el sobreajuste del modelo, aunque reduce la sensibilidad de éste con respecto a la data de entrenamiento.

XGBoost contruye árboles de manera secuencial, de tal forma que cada nuevo de árbol busca corregir los errores de los anteriores. Ya que en general es imposible enumerar todas las posibles estructuras de árboles, la optimización de la función de costo para nuevos árboles es resuelta mediante un algoritmo avaro (*greedy*), iniciando la construcción de cada árbol a partir de una única hoja. En cada rama, el algoritmo debe analizar la data ya ordenada de acuerdo a los valores de sus características, decidir si acaso es óptimo o no generar una nueva división en otras dos hojas, y en ese caso, cuál característica y valor de umbral para ésta se debe usar para dicha división. El parámetro γ aumenta el costo de generar más hojas, favoreciendo la poda del árbol y así mantenerlo más simple. Ya que probar cada posible valor de umbral para cada característica es extremadamente ineficiente si el conjunto de datos es grande, XGBoost también incluye una estrategia de cuantiles para reducir el número de candidatos a umbral de separación. La base teórica de ese procedimiento, conocido como algoritmo *greedy* aproximado, puede encontrarse en [39].

Finalmente, es necesario mencionar que XGBoost está disponible como un paquete de código abierto en [40], para Python, R, Julia, C, C++, entre otros lenguajes de programación, y cuya documentación se puede encontrar en [41]. Más detalles sobre el algoritmo pueden encontrarse en [39].

2.7. Modelo de la batería

Ya que la propuesta presentada en esta investigación está enfocada en algoritmos de pronóstico de fallas basados en filtro de partículas, en primer lugar se requiere una modelo de

espacio de estado para el *pack* de baterías que se estudia. El modelo que se ha considerado corresponde a una versión simplificada de aquel usado en [7] y propuesto en [14], que se muestra en Ec. (2.22) y Ec. (2.23). La primera representa la dinámica de la única variable de estado del sistema, el SOC (State of Charge), estado de carga en español, en términos de la potencia entregada desde la batería al motor, y la segunda corresponde a la ecuación de observación, que caracteriza el voltaje medido en los terminales de la batería, como función del SOC:

$$x_{k+1} = x_k - v(x_k) i_k \Delta t E_c^{-1} + \omega_k \tag{2.22}$$

$$\begin{aligned} v(x_k) = {} & v_l + (v_o - v_l)\mathrm{e}^{\gamma(x_k - 1)} + \alpha v_l (x_k - 1) \\ & + (1-\alpha) v_l (\mathrm{e}^{\beta} - \mathrm{e}^{\beta\sqrt{x_k}}) - i_k \cdot R + \eta_k \,. \end{aligned} \tag{2.23}$$

Aquí, x_k representa el SOC, variable de estado, $v(x_k)$ el voltaje en los terminales de la batería en función del SOC, i_k la corriente de descarga y entrada exógena del modelo, y R la impedancia interna de la batería, que se considera como una resistencia de valor constante, mientras que ω_k y η_k corresponden a los ruidos de proceso y observación, respectivamente. Por otro lado, los parámetros α, β, γ, v_0 y v_l de la ecuación de observación son dependientes del arreglo de celdas de Ion-Li usadas en el paquete de baterías [42], cuyos valores pueden estimarse mediante el procedimiento presentado en [14]. Finalmente, el tiempo de muestreo es denotado por Δt, y E_c corresponde a la máxima energía nominal que la batería es capaz de almacenar. Es importante recalcar aquí que el SOC es una variable normalizada por E_c, es decir, mide la razón entre la energía remanente en la batería con respecto a la máxima capacidad de ésta, por tanto sólo puede tomar valores entre 0 y 1, además de ser adimensional, motivo por el cual no es acompañada por ninguna unidad en los gráficos presentados más adelante.

En analogía a la investigación presentada en [6], es necesario enfatizar que la contribución de este trabajo apunta en la técnica del algoritmo de pronóstico en sí, y no en el modelo de descarga de la batería. Esto se debe a que, al contrario de [14, 7] donde la resistencia interna se modela como un *random walk*, o caminata aleatoria, esta investigación la considera simplemente como una constante, aunque con escepticismo con respecto a su exactitud. Sin embargo, de esta forma es posible mantener el foco en los efectos de la pérdida de información debido al uso de los saltos de tiempo. Además, esta simplificación implica también que el modelo aquí presentado no considera los efectos de variables como SOC, SOH y corrientes de descarga en la impedancia interna [7].

2.7.1. Condición de falla: insuficiencia de potencia y SoMPA

En esta investigación el tiempo de falla, o fin de descarga en este caso de estudio particular, ha sido definido como un evento desencadenado por cruce de umbral, como es usual en la literatura [43, 26]. La condición empleada para caracterizar el evento de falla es la insuficiencia de potencia, lo que ocurre la demanda de ésta a la batería, $P_k = v(x_k) \cdot i_k$, excede el máximo que puede entregar a un determinado instante, cuantificada por un parámetro denominado *estado de máxima potencia disponible*, o SoMPA (State of Maximum Available Power) [7]. El SoMPA puede ser estimado como una función del SOC, siguiendo el procedimiento explicado a continuación.

En primer, debe ser notado que la Ec. (2.23) puede reescribirse como se muestra en la Ec. (2.24), lo que significa que la potencia entregada por la batería al instante k puede ser descrita también por la Ec. (2.25):

$$v(x_k) = v_{oc}(x_k) - i_k \cdot R \tag{2.24}$$

$$P_k = (v_{oc}(x_k) - i_k \cdot R) \cdot i_k, \tag{2.25}$$

donde $v_{oc}(x_k)$ se define como el voltaje de circuito abierto, parametrizado por α, β, γ, v_l y v_0 en la Ec. (2.23). De esta forma, el SoMPA puede calcularse resolviendo el siguiente problema de optimización, mostrado en la Ec. (2.26):

$$\begin{aligned} \underset{i(k)}{\text{máx}} \quad & P_k = v_{oc}(x_k) i_k - R \cdot ik^2 \\ \text{s.a.} \quad & r_1 : 0 \leq i_k \\ & r_2 : i_k \leq I_{max} \\ & r_3 : V_c \leq v_{oc}(x_k) - i_k \cdot R. \end{aligned} \tag{2.26}$$

Aquí, r_1 significa simplemente que la batería se debe estar descargando y no cargando, r_2 exige que la corriente de esta descarga debe ser siempre menor o igual a I_{max}. Finalmente, r_3 corresponde al requerimiento de que el voltaje en los terminales de la batería, $v(x_k)$, debe ser mayor o igual al voltaje de corte V_c. Resolver este problema permite determinar la máxima corriente como función del SOC, y por tanto, el SoMPA.

Además, es necesario notar que la Ec. (2.25) describe una parábola cuyo vértice, y punto máximo en este caso, está localizado en $i_{NR}(x_k) = \frac{v_{oc}(x_k)}{2R}$. Teniendo esto en cuenta, además de los cálculos presentados en [7], se desprende que encontrar la corriente de descarga i_k^* que resuelve el problema de optimización se reduce a lo que se muestra en la Ec. (2.27):

$$i_k^* = \min\left\{ \frac{v_{oc}(x_k)}{2R}, \frac{v_{oc}(x_k) - V_c}{R}, I_{max} \right\}. \tag{2.27}$$

Remplazando este valor en Ec. (2.25) se tiene el valor del SoMPA para un instante determinado k. Es fundamental también recalcar que, al ser $v_{oc}(x_k)$ función del SOC, se cuenta con una relación ananlítica entre SOC y SoMPA.

Capítulo 3

Caso de Estudio

Este capítulo tiene como objetivo presentar el caso de estudio utilizado para validar la estrategia desarrollada, que corresponde a una bicicleta eléctrica y la descarga de su batería, al conducirla por un cierto trayecto [7]. En primer lugar, se detallan las características relevantes de la batería de dicha bicicleta, seguido por una explicación sobre el modelamiento de su corriente de descarga para las posteriores simulaciones. Por último, se examina el comportamiento de algunas variables de interés de la batería a lo largo del ciclo de descarga.

3.1. Características de la batería

En la Tabla 3.1 se presentan los valores usados para I_{max} y V_c, los parámetros que limitan la potencia máxima que puede entregar la batería, y que se presentan en las restricciones de la Ec. (2.26). Las cifras empleadas corresponden a las que han sido reportadas también en [7], y entregados por el fabricante de dicha batería.

Por otra parte, en la Tabla 3.2 se muestran los valores específicos empleados para los parámetros del modelo de descarga de la batería, presentado anteriormente en las Ec. (2.22) y Ec. (2.23). Dichos valores corresponden en general a aquellos que se han reportado previamente en [7], salvo dos excepciones. En primer lugar, como ya se explicó previamente en el Capítulo 2, a la resistencia interna se le asignó una cifra constante, de $R = 0{,}26\,\Omega$ en este caso, correspondiente a la media de la distribución de probabilidad inicial del estado durante la etapa de filtrado también en [7]. En segundo lugar, para la desviación estándar del ruido de proceso se usó una magnitud menor, de $\sigma_\omega = 10^{-6}$, misma cifra empleada en [6].

Tabla 3.1: Valores usados para I_{max} y V_c, dados por el fabricante [7].

Parámetros	**Descripción**	**Valores**
I_{max}	Máxima corriente de descarga recomendada	11.5 A
V_c	Voltaje de corte de la batería	33 V

Tabla 3.2: Valores de los parámetros del modelo de batería, mismos que se usan en [7], a excepción de la resistencia interna R que aquí se considera como constante, y el ruido de proceso σ_ω, para el cual se tomó el valor usado en [6].

Parámetros	**Descripción**	**Valores**
R	Resistencia interna de la batería	0.26 Ω
Δt	Tiempo de muestreo	1 s
α	Parámetro de ecuación de observación	$5{,}319 \times 10^{-3}$
β	Parámetro de ecuación de observación	11.505
γ	Parámetro de ecuación de observación	1.5538
v_0	Parámetro de ecuación de observación	41.405 V
v_l	Parámetro de ecuación de observación	33.481 V
E_c	Máxima energía almacenada por la batería	1389900 J
σ_ω	Desviación estándar ruido de proceso	10^{-6}
σ_η	Desviación estándar ruido de medición	0.9 V

3.2. Entrada exógena: perfil de corriente de descarga

Ante la incertidumbre inherente en los problemas de pronóstico sobre las entradas futuras del sistema estudiado, es necesario caracterizarlas de alguna manera para elaborar una predicción. En el caso de estudio utilizado en esta investigación, la entrada exógena del sistema corresponde a la corriente de descarga de la batería, la cual se modeló adoptando un enfoque basado una cadena de Markov de dos estados, como se propuso en [31], y luego se aplicó en [7]. Allí, se presentan tres conjuntos de datos experimentales, obtenidos al conducir la bicicleta eléctrica por diferentes terrenos. En este trabajo se considera sólo uno de ellos, correspondiente a una superficie plana, el cual comprende 29 recorridos alrededor de la elipse en el Parque O'Higgins en Santiago de Chile. Los valores de corriente empleados para los estados de la cadena estados se muestran en Ec. (3.1), también usados en [7]. Por otro lado, las probabilidades de transición entre estos estados se determinó mediante estimadores de máxima verosimilitud, Ec. (2.17). La matriz de transición obtenida se encuentra en Ec. (3.2).

$$C = \begin{bmatrix} 3{,}4979 & 5{,}0526 \end{bmatrix} \tag{3.1}$$

$$P = \begin{bmatrix} 0{,}9388 & 0{,}0612 \\ 0{,}0554 & 0{,}9446 \end{bmatrix}. \tag{3.2}$$

En la Fig. 3.1 se muestra el perfil de corriente de descarga según los datos experimentales presentados en [7], así como la equivalencia de éste en los niveles del modelo de cadena de Markov.

Es necesario aquí recalcar que la secuencia de valores de corriente del modelo de cadena de Markov que se muestra en Fig. 3.1, equivalente al perfil experimental, no es la misma que se usa en las simulaciones, sino que éstas se efectúan mediante realizaciones aleatorias del modelo.

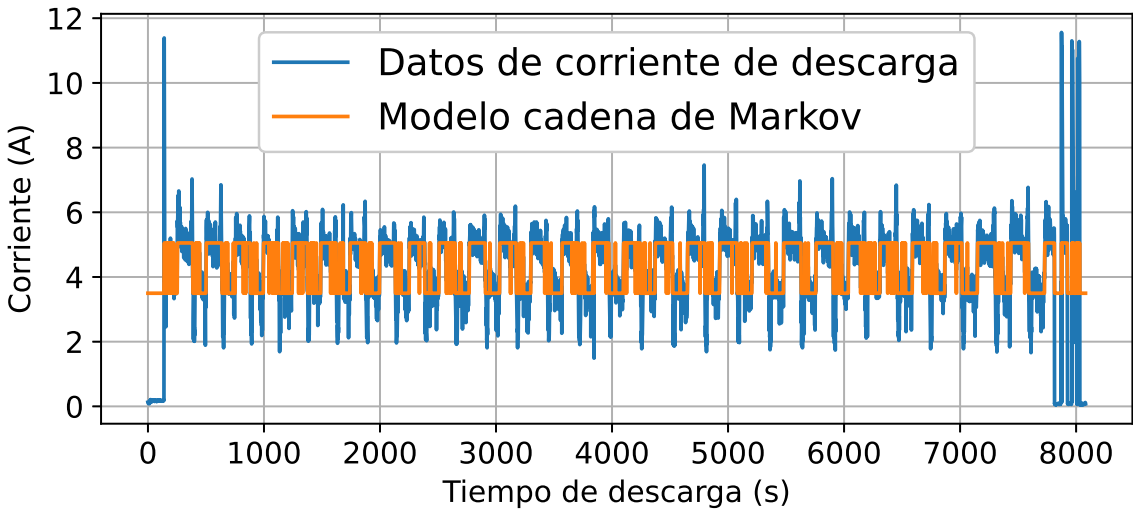


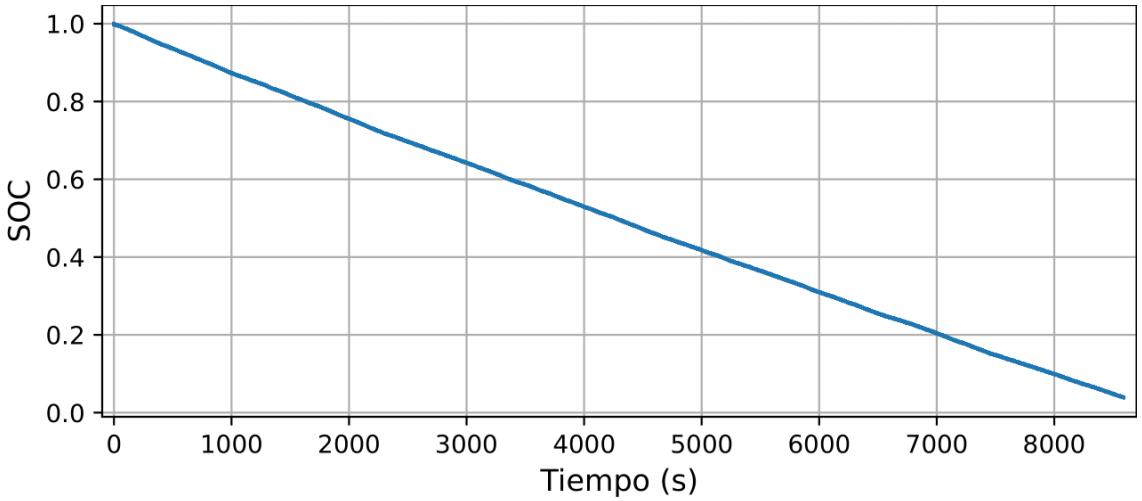
Figura 3.1: Comparación perfil de corriente de descarga experimental con modelo de cadena de Markov.

3.3. Comportamiento de la batería

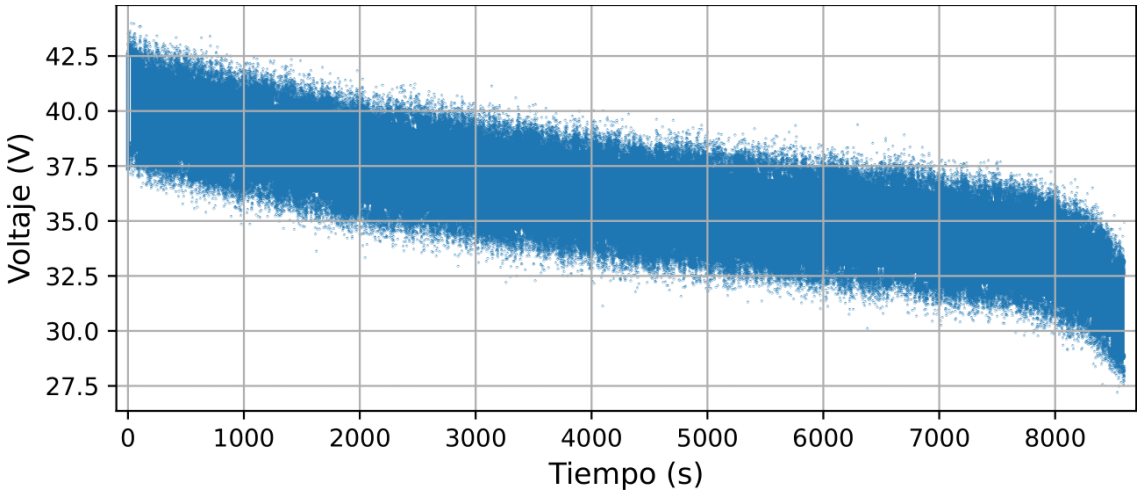
Para ilustrar el comportamiento de la batería bajo un perfil de descarga modelado por la cadena de Markov antes presentada, en la Fig. 3.2 se muestran los resultados de una simulación del sistema por filtro de partículas. La imagen superior muestra la evolución del SOC (State of Charge), mientras que en la parte inferior se puede apreciar el voltaje en los terminales de la batería. La simulación se llevó a cabo suponiendo como condición inicial una distribución Gaussiana de media $x_0 = 1$, es decir, la batería cargada al 100 %, y una desviación estándar de $\sigma_x = 10^{-3}$. El horizonte de pronóstico es de más de 8000 s, equivalente a más de 2 horas de operación de la bicicleta eléctrica, cubriendo así casi completamente el ciclo de descarga. En cada instante del horizonte de predicción, se muestra la "nube" de 500 partículas de la simulación, todas alimentadas por una única realización de la cadena de Markov para representar la entrada exógena.

Como se puede apreciar, el decaimiento del SOC es prácticamente lineal, mientras que el caso del voltaje pueden distinguirse las zonas exponenciales y lineales establecidas aproximadamente según la Ec. (2.23). La variabilidad entre partículas es más apreciable en el caso del voltaje, tal como sugieren las magnitudes de los ruidos de proceso y medición, según la Tabla 3.2.

Por otro lado, en la Fig. 3.3 se muestra la evolución de la nube de partículas que representa tanto el SoMPA como la potencia demandada según el método estándar, durante todo el horizonte de descarga, en función de la evolución del SOC y voltaje mostrado en la Fig. 3.2, de acuerdo a las relaciones expuestas en el Capítulo 2. En el caso de la potencia demandada, se pueden apreciar dos bandas debido a los dos distintos niveles de corriente del modelo de cadena de Markov, y por tanto dos niveles de potencia. Ya que esta nube de partículas corresponde a una única realización de la cadena, la potencia puede tomar el valor de sólo un nivel de corriente a la vez, provocando que ambas bandas de potencia sean intermitentes.



(a) Nube de partículas de SOC (State of Charge) durante todo el horizonte de predicción.



(b) Nube de partículas de voltaje durante todo el horizonte de predicción.

Figura 3.2: Nubes de partículas de SOC y voltaje de la batería, relacionadas por la Ec. (2.23), bajo una única realización del modelo de cadena de Markov para la corriente de descarga.

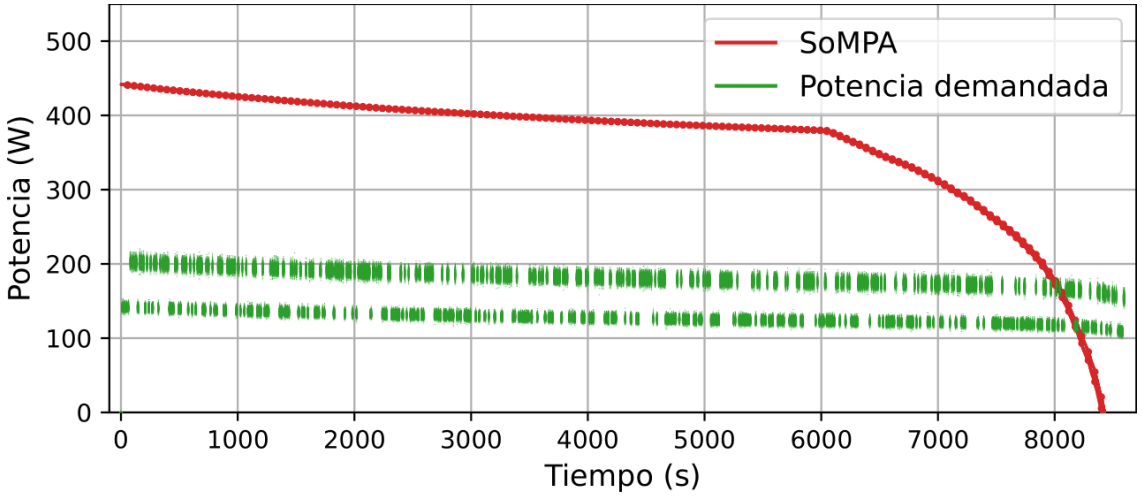


Figura 3.3: Comparativa de nubes de partículas de SoMPA y potencia demandada durante el horizonte de predicción, para una realización de la cadena de Markov como entrada, ambas generadas por el método estándar.

Capítulo 4

Metodología

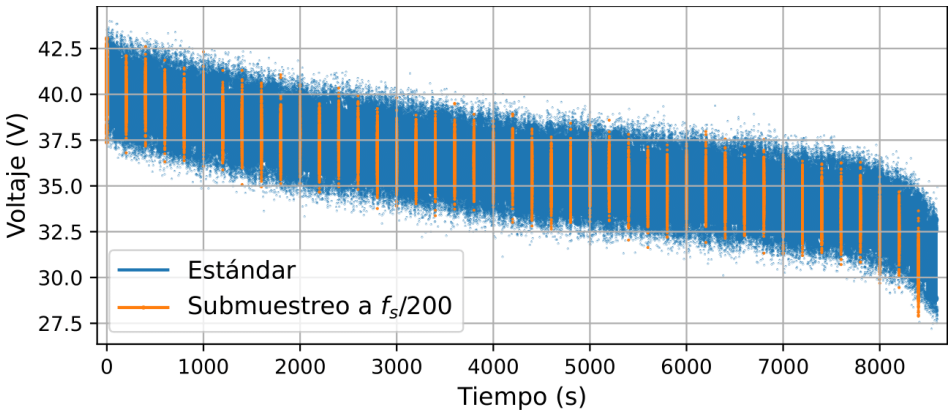
4.1. Adaptación de la ecuación de estado para frecuencias de submuestreo

Como se mencionó anteriormente, el objetivo de esta investigación es probar que se puede reducir drásticamente el costo computacional de los algoritmos de pronóstico basados en PF, permitiendo así cubrir extensos horizontes de predicción durante una reducida ventana de tiempo de ejecución. Si bien el caso de estudio corresponde a la descarga de una batería de Ion-Li, la formulación matemática de la propuesta es aplicable a problemas más generales. En concreto, el desarrollo teórico aquí presentado es válido para cualquier sistema con un modelo de espacio de estado continuo, a tiempo discreto, no lineal, que satisface la propiedad de Markov, y cuya incertidumbre está representada por un ruido de proceso aditivo Gaussiano. En la Ec. (4.1) se presenta la forma general que para describir a un sistema que posee las características mencionadas.

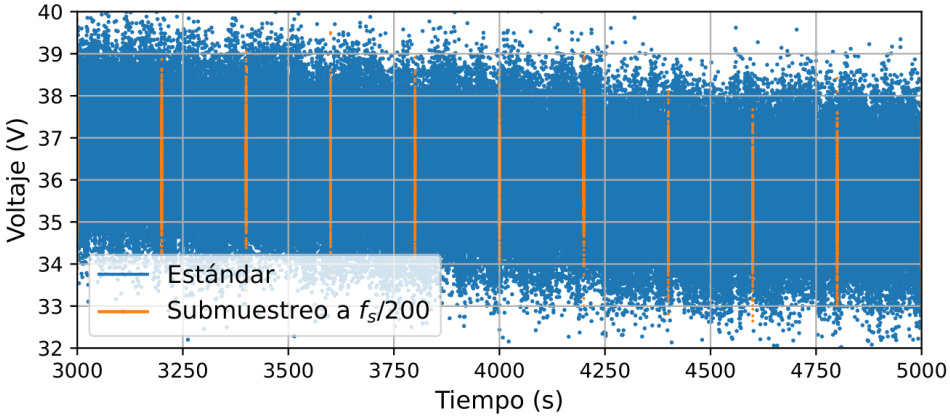
$$x_{k+1} = f(x_k, u_k) + \omega_k, \tag{4.1}$$

donde x_k es el vector de estado, u_k la entrada exógena, ω_k el ruido de proceso, y $f(\cdot, \cdot)$ una función de dos variables arbitraria que puede ser no lineal.

En la literatura existen numerosos resultados satisfactorios en términos de precisión para algoritmos de pronóstico de fallas basados en PF [7, 6] cuando la simulación se lleva a cabo usando el tiempo de muestreo estándar del sistema. Sin embargo, esto significa que para determinar el valor del estado durante todo el horizonte de predicción, y por tanto, estimar el ToF, el algoritmo debe usar la Ec. (4.1) para calcular x_{k_0+1} a partir de x_{k_0}, siendo k_0 el instante actual, luego x_{k_0+2}, x_{k_0+3}, y así sucesivamente. Si bien esto es posible de hacer contando con una adecuada caracterización de los valores futuros de la entrada exógena, una gran desventaja de este enfoque es el alto costo computacional que implica, en especial cuando el horizonte de predicción es extenso. Por este motivo, la presente investigación propone una estrategia novedosa, hasta donde se tiene conocimiento, que podría reducir drásticamente el tiempo de cómputo, mientras que la exactitud del pronóstico no sufre pérdidas significativas.



(a) Nube de partículas durante el horizonte de predicción completo.



(b) Acercamiento a nube de partículas durante la ventana de tiempo entre 3000 a 5000 s.

Figura 4.1: Comparativa de nubes de partículas de voltaje en terminales de la batería, para una realización de la cadena de Markov como entrada, generadas por el método estándar que emplea frecuencia de muestreo original (azul), y por el método propuesto, efectuando un submuestreo cada 200 instantes (naranja).

Esta propuesta sugiere que para esto no es necesario determinar el estado durante todo el horizonte de predicción, sino sólo en algunos instantes de éste, haciendo "saltos" entre instantes no consecutivos en el tiempo. Esto significa que se requiere calcular directamente la transición entre desde un estado x_k a x_{k+p}, $p > 1$ con p un entero positivo arbitrario, sin pasar por los valores intermedios como x_{k+1}, x_{k+2}, hasta x_{k+p-1}. De esta manera, un largo horizonte de predicción h puede ser cubierto haciendo varios "saltos de tiempo", que pueden ser de un tamaño fijo p, cuyo valor depende de la precisión con la que se requiera conocer el pronóstico, las condiciones de operación del sistema, la batería en este caso, y la ventana de tiempo con la que se cuenta para la ejecución del algoritmo. Es necesario hacer notar que propagar el estado con un salto de tiempo, o tamaño de paso temporal de tamaño p, es equivalente a usar una frecuencia de submuestreo de f_S/p, siendo f_S la frecuencia de muestreo original del sistema. Por tanto, desde ahora en adelante, se usarán ambos conceptos indistintamente. En la Fig. 4.1 se muestra una comparativa entre las nubes de partículas de la estimación del voltaje en terminales de la batería, determinadas usando la frecuencia de muestreo estándar, y la estrategia propuesta con un tamaño de paso temporal de $p = 200$.

Para calcular directamente x_{k+p} a partir de x_k se requiere una expresión matemática

general que relacione dos estados del sistema en instantes no consecutivos. Es decir, una expresión como la que se muestra en la Ec. (4.2):

$$x_{k+p} = \mathcal{F}(x_k, u, \omega). \tag{4.2}$$

Para un modelo lineal no es difícil obtener una expresión como ésta, mediante un cálculo iterativo. No obstante, este no es el caso para un modelo no lineal como el que se plantea en la Ec. (4.1), que son justamente más comunes en aplicaciones reales. Mediante una linealización de esta ecuación, junto con un cálculo recursivo, se ha derivado la expresión en la Ec. (4.3) para x_{k+p} a partir x_k, donde k puede ser un instante arbitrario:

$$x_{k+p} \approx f(x_k, u_k) + \tilde{A}_k(p)\delta x_k + \sum_{i=1}^{p-1} A_k^{i-1} B_k\big(u_{k+p-i} - u_k\big) + \sum_{i=0}^{p-1} A_k^i \omega_{k+p-1+i}, \tag{4.3}$$

con

$$\delta x_k = f(x_k, u_k) - x_k \tag{4.4}$$

$$\tilde{A}_k(p) = \sum_{i=1}^{p-1} A_k^i \tag{4.5}$$

$$A_k = \left.\frac{\partial f}{\partial x}\right|_{(x_k, u_k)} \tag{4.6}$$

$$B_k = \left.\frac{\partial f}{\partial u}\right|_{(x_k, u_k).} \tag{4.7}$$

La expresión que se muestra en Ec. (4.3) condensa la dinámica intrínseca del sistema, la entrada exógena y el ruido de proceso. No obstante, ya que se ha obtenido mediante linealización, sólo es válida mientras que los valores tanto del estado como de la entrada exógena al instante $k + p$, x_{k+p} y u_{k+p}, sean lo suficientemente cercanos a los que se tenían en k, lo que en la práctica implica una pérdida de exactitud para valores demasiado grandes de p, el tamaño del salto.

A partir de la Ec. (4.4) hasta la (4.7), no es difícil notar que, en efecto, todo excepto los valores de entrada exógena y el término de ruido pueden determinarse únicamente a partir de los valores del estado y la entrada exógena, x_k y u_k respectivamente, en el instante k. Esto significa que, en principio, es posible predecir x_{k+p} usando esta información, y caracterizando adecuadamente la entrada y el ruido durante el intervalo del salto de tiempo.

Como se puede apreciar, la obtención de la Ec. (4.3) para cada modelo de espacio de estado se requiere de cálculo matricial, que involucra las derivadas de la función no lineal que describe el sistema bajo estudio. Con el objetivo de reducir al mínimo posible el tiempo

de ejecución de un algoritmo de pronóstico en tiempo real, las expresiones analíticas de las derivadas deben de calcularse fuera de línea y estar programadas en el dispositivo que lleva a cabo el pronóstico. De esta forma, la única tarea que debe ser efectuada en tiempo real es la evaluación de estas expresiones en el punto de operación, x_k y u_k, y la respectiva álgebra matricial, permitiendo así el uso de dispositivos poco costosos y de baja potencia de cómputo.

Otro aspecto de esta formulación que puede ralentizar el algoritmo de pronóstico, agregando un costo extra en términos de tiempo de cómputo, es el cálculo de las sumas sobre los términos de la entrada exógena y ruido de proceso, que puede resultar engorroso. Para evitar este paso, se proponen otras dos simplificaciones. La primera consiste en asumir que la entrada exógena se mantiene constante durante el intervalo $(k, k+p]$, es decir, $u_{k+p} = u_{k+p-1} = u_{k+p-2} = u_{k+p-3} = ... = u_{k+2} = u_{k+1} \neq u_k$. de esta manera, se tiene que $u_{k+p-\mathrm{i}} - u_k = u_{k+p-1} - u_k$, para cualquier i tal que $1 \leq i \leq p-1$. De esta forma, es posible factorizar la sumatoria de los términos exógenos, como se muestra en la Ec. (4.8):

$$\sum_{i=1}^{p-1} A_k^{i-1} B_k \big(u_{k+p-i} - u_k\big) = \left(\sum_{i=1}^{p-1} A_k^{i-1} B_k\right)\Big(u_{k+p} - u_k\Big). \tag{4.8}$$

Es importante destacar que esta expresión requieren sólo de dos valores de la entrada exógena, u_k y u_{k+p}. Esto resulta particularmente conveniente para el caso de estudio en cuestión, pues ya que la corriente de descarga está modelada por cadenas de Markov, calcular la transición entre dos valores separados por p pasos en sencillo, haciendo uso de la propiedad mencionada previamente en la Sección 2.4. Esta característica contribuye además a la reducción del tiempo de cómputo, pues los valores de la entrada se generan aleatoriamente sólo para algunos instantes del horizonte de predicción, utilizando una matriz de transición calculada una única vez para un paso temporal de tamaño p. Sin embargo, se debe recalcar que esta condición no no limita el uso de otras herramientas para caracterizar las entradas futuras; en tales casos, sería necesario adaptar la metodología escogida para extraer dos muestras separadas por p instantes.

En el caso del ruido de proceso, la simplificación propuesta evita la necesidad de extraer un alto número de muestras de la variable aleatoria, como en principio exige la Ec. (4.3). Para esto, el efecto acumulado del ruido durante el intervalo de p instantes se modela como una única variable aleatoria del mismo tipo que el ruido original, Gaussiana, pero cuya matriz de covarianza W_p viene dada por la Ec. (4.9), calculada usando las propiedades de la varianza de una combinación lineal de variables aleatorias.

$$W(p) = \sum_{i=0}^{p-1} A_k^i W (A_k^i)^T, \tag{4.9}$$

donde W es la matriz de covarianza del ruido de proceso original.

Luego de estas simplificaciones, se obtiene la expresión final que aproxima la Ec. (4.3), que se muestra en la Ec. (4.10):

$$
\begin{aligned}
x_{k+p} \approx & f(x_k, u_k) + \tilde{A}_k(p)\delta x_k \\
& + \left(\sum_{i=1}^{p-1} A_k^{i-1} B_k \right) \left(u_{k+p-1} - u_k \right) + \omega_k(p),
\end{aligned} \tag{4.10}
$$

donde $\omega_k(p)$ es una muestra una variable aleatoria del mismo tipo que el ruido de proceso, pero cuya matriz de covarianza está dada por $W(p)$, Ec. (4.9).

Todas las aproximaciones descritas en esta sección fueron implementadas en un código en Python, aplicadas en particular al modelo de descarga de la batería de Ion-Li descrito en la Sección 2, con el fin de ser incorporadas al algoritmo de pronóstico de filtro de partículas.

4.2. Esquema de pronóstico con submuestreo

Luego de exponer el tratamiento matemático de la ecuación de estado, quedan aún varias interrogantes que abordar sobre el algoritmo de pronóstico propuesto. Por ejemplo, es necesario aún responder cuál es el tamaño óptimo de los saltos de tiempo p y el procedimiento para determinarlo, la cantidad de tamaños de salto que deben emplearse durante una única ejecución del algoritmo y, en caso de utilizar más de uno, el instante adecuado para alternar entre ellos. Estos parámetros varían en función de variables como la carga inicial SOC de la batería al momento de ejecutar el pronóstico, x_0, o el máximo tiempo de cómputo fijado por el usuario, por lo tanto se debe estudiar dicha dependencia.

En este punto, ya que el enfoque de pronóstico propuesto se basa en efectuar un submuestreo del sistema estudiado, es válida la interrogante acerca del rol que podría cumplir aquí el Teorema del muestreo de Shannon-Nyquist. El objetivo de esta herramienta es determinar la frecuencia mínima que se debe emplear de tal forma que sea posible reconstruir la señal original. Sin embargo, ya que en este caso se tiene un problema de pronóstico, donde lo que interesa es únicamente anticipar del instante de falla, no es necesario recuperar el historial completo del sistema bajo monitoreo, siempre y cuando la predicción cuente con la suficiente precisión según alguna métrica apropiada. De todas maneras, si bien un análisis empleando el teorema de Shannon-Nyquist no podría entregar una frecuencia óptima para pronóstico, sí podría ser de utilidad como directriz para determinar una frecuencia mínima recomendada, y así acotar el espacio de búsqueda.

Para abordar las interrogantes acerca de cuáles frecuencias de submestreo se deben utilizar, se propone una metodología que consta principalmente de dos grandes etapas. En primer lugar, de manera *offline* y mediante algoritmos genéticos, se genera un conjunto de datos de frecuencias óptimas para pronóstico. Este proceso implica la exploración de diversas condiciones iniciales de la batería y distintos límites de tiempo ejecución. Posteriormente, también *offline*, estos datos se utilizan para entrenar varios modelos XGBoost. Dichos modelos están diseñados para puedan seleccionar un tamaño de paso temporal recomendado en función del estado de carga actual de la batería, y debe ser tal que respete la restricción de tiempo máximo de ejecución definida por el usuario, procurando al mismo tiempo minimizar la pérdida de precisión. De esta forma, se puede determinar en tiempo real la frecuencia de muestreo

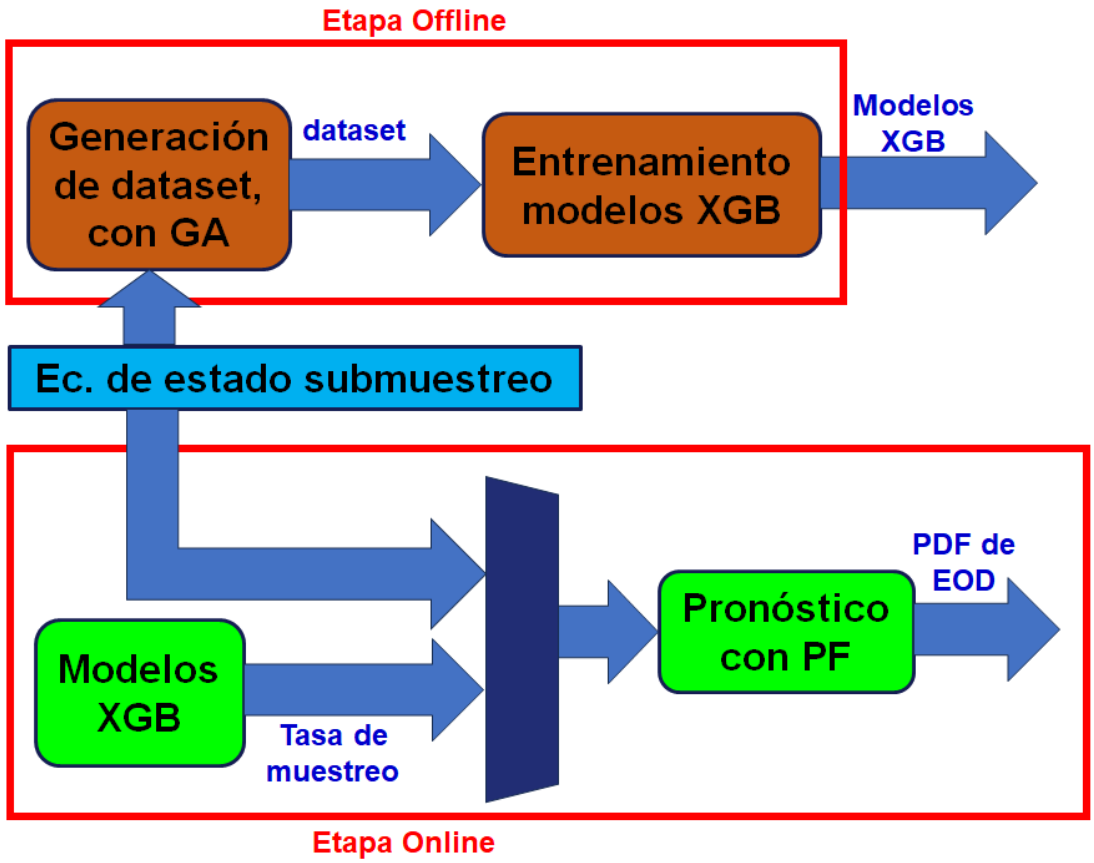


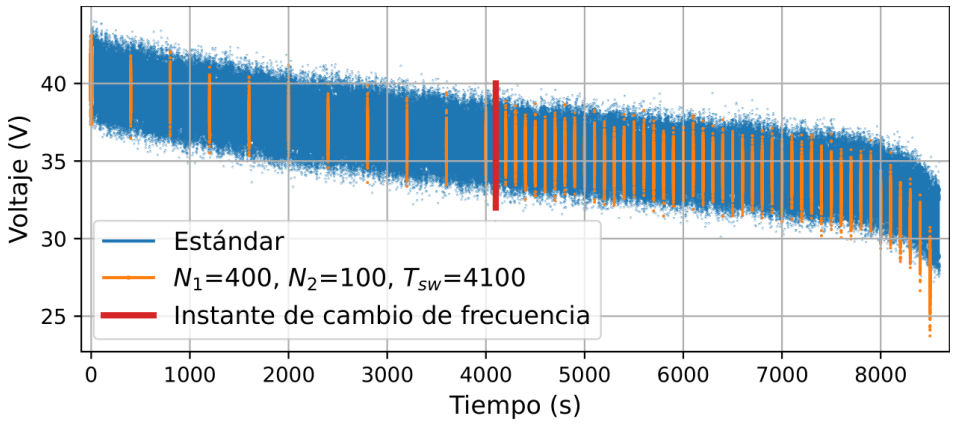
Figura 4.2: Diagrama de esquema de pronóstico propuesto. De la etapa *offline* se obtienen los modelos XGBoost ya entrenados para seleccionar en tiempo real la tasa de muestreo. En la fase en *online*, se efectúa el pronóstico usando la tasa de muestreo recomendada, para obtener así la distribución de probabilidad de EoD.

recomendada para elaborar el pronóstico, usando la Ec. (4.10). En la Fig. 4.2 se muestra un diagrama de las principales etapas de la metodología, indicando cuáles se llevan a cabo *offline*, y cuáles en tiempo real.

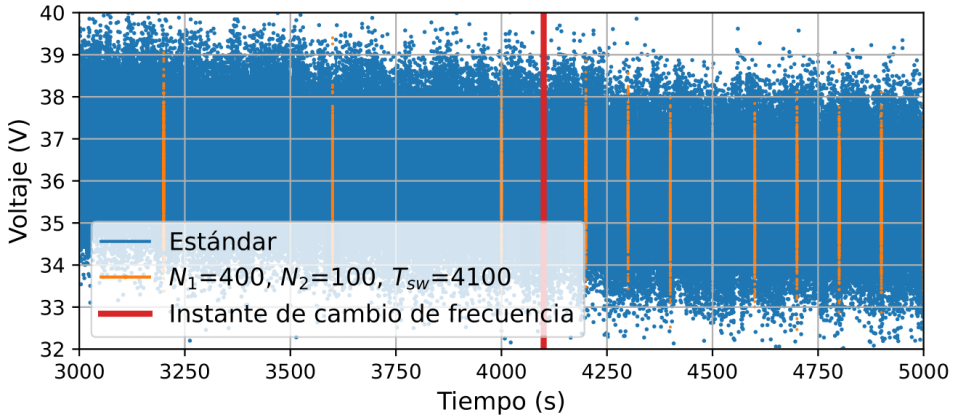
A continuación, se describe en detalle cada una de las etapas del algoritmo.

4.2.1. Generación de base de datos con GA

Por simplicidad, en esta investigación se decidió explorar únicamente el caso en que el algoritmo de pronóstico emplea sólo dos frecuencias de muestreo distintas, usando la más baja de éstas al comienzo, y la más alta hacia el final de la descarga, con el objetivo de contar con una mayor resolución temporal al acercarse el instante de EoD. Recordando que una frecuencia de submuestreo f_S/N equivale a un salto de N instantes del estado durante el pronóstico, se tiene entonces que los parámetros a determinar son tres: N_1, N_2 y T_{sw}, que corresponden a los divisores de frecuencia para la primera etapa, segunda etapa, y el instante de cambio entre ambas, respectivamente. En la Fig. 4.3 se muestra el rol de estos tres parámetros en las nubes de partículas generadas por el algoritmo, para ambos métodos.



(a) Nube de partículas durante el horizonte de predicción completo.



(b) Acercamiento a nube de partículas durante la ventana de tiempo entre 3000 a 5000 s.

Figura 4.3: Ilustración de los roles de los parámetros N_1, N_2 y T_{sw} en la estrategia de pronóstico propuesta, para una realización de la cadena de Markov como entrada, en comparación con el enfoque tradicional.

En resumen, el objetivo específico es, para cada condición inicial de la batería x_0, y tiempo de ejecución $T_{max} = 5, 4, 3, 2$ s, encontrar las combinaciones óptimas de parámetros $[N_1, N_2, T_{sw}]$ para la estrategia de pronóstico aquí propuesta, de tal forma que exista un balance entre la eficiencia y eficacia del algoritmo.

Para evaluar la calidad del pronóstico obtenido a partir de las diferentes combinaciones de parámetros, se ha decidido usar como criterio la similitud del JITP (*Just in Time Point*) del enfoque propuesto versus el tradicional. El JITP es una métrica que busca caracterizar el riesgo asociado a decisiones basadas en los algoritmos de pronóstico, asumiendo que dicho riesgo depende de las colas de la distribución de probabilidad de falla [32]. El JITP se puede definir como se muestra en la Ec. (4.11):

$$JITP_{\alpha\,\%} = \underset{k}{\operatorname{argmin}}(P(ToF \leq k) \geq \alpha\,\%), \tag{4.11}$$

es decir, el primer instante en el que la probabilidad de que el sistema haya fallado cruza un cierto umbral $\alpha\,\%$. De esta manera, el parámetro α permite definir un nivel de tolerancia al riesgo.

La búsqueda de los divisores de frecuencia óptimos se ha llevado a cabo mediante un algoritmo genético. Para ello, se ha construido una función de *fitness* que considera tanto la diferencia entre el JITP de ambos enfoques, como el hecho de si se satisface o no la restricción de tiempo de cómputo, esto debido a que, por la naturaleza del problema, es prácticamente imposible incluir dicha restricción como parte del algoritmo de búsqueda. La función de *fitness* definida se muestra en la Ec. (4.12):

$$f(\cdot) = \text{máx}\, error\,\%_{\{JITP5\,\%,10\,\%,15\,\%\}} + \tau H(T_{\text{ex}} - T_{max}), \tag{4.12}$$

donde τ es una constante, $H(\cdot)$ la función escalón de Heaviside, y $error\,\%_{JITP\alpha\,\%}$, α un porcentaje arbitrario, se define en la Ec. (4.13):

$$error\,\%_{JITP\alpha\,\%} = \frac{\left|JITP_{\alpha B} - JITP_{\alpha A}\right|}{JITP_{\alpha B}} \times 100, \tag{4.13}$$

con $JITP_{\alpha B}$ y $JITP_{\alpha A}$ los valores del JITP para un valor $\alpha\,\%$ para la CDF del instante de descarga, de acuerdo al *baseline* (B, estrategia estándar), y la propuesta aproximada por esta investigación (A), respectivamente.

La función de *fitness* se ha construido principalmente por métodos heurísticos. En primer lugar, los porcentajes escogidos para el JITP son bajos con el objetivo de mantener un criterio conservador para dar una alerta de un inminente EoD, de tal manera que permita una cierta holgura para tomar acciones preventivas cuando esto ocurra. Por otro lado, debido a que el término del error de JITP en la función es extremadamente complejo y prácticamente imposible de graficar como función de la condición inicial del SOC y el tiempo de cómputo, se decidió incluir esta última restricción en la misma función de *fitness*, como una restricción

dura [44]. En consecuencia, el valor exacto de la constante τ no es relevante, y se requiere únicamente que sea lo suficientemente alto como para penalizar fuertemente aquellas soluciones cuyo tiempo de cómputo sobrepasen el límite establecido, pero no activarse en lo absoluto cuando dicho límite se respeta.

Ya que el estado de carga SOC es una variable continua, es imposible determinar el conjunto de parámetros óptimos para cada valor factible. Sin embargo, la búsqueda se ha llevado a cabo para un amplio rango de estos, desde $x = 0{,}2$ hasta $x = 1$, espaciados por una diferencia de $\Delta x = 0{,}05$, sumando 17 valores en total. Por otro lado, los límites de tiempo de cómputo explorados han sido de $T_{max} = 5, 4, 3, 2$ s. La búsqueda se efectuó usando una versión levemente modificada de una implementación de GA disponible en línea [45], donde el método de ruleta para la selección de cromosomas ha sido reemplazado por el de torneo. La estructura básica del cromosoma ha sido definida como (T_{sw}, N_1, N_2), lo que implica que, de acuerdo a la definición previa de estas variables, el GA considera sólo soluciones enteras positivas. Además, para restringir aún más el espacio de búsqueda, se decidió limitar tanto N_1 como N_2, los divisores de frecuencia, entre el rango $[1, 200]$, mientras que los valores posibles de T_{sw} se han limitado al intervalo $[1, k_{20\,\%}]$, donde $k_{20\,\%}$ se define como el instante en que se alcanza una probabilidad de ya haber fallado del 20 %, según la CDF calculada mediante el enfoque clásico. En otras palabras, $k_{20\,\%} = JITP_{20\,\%B}$. Como condiciones de término del GA se consideró un máximo de 40 generaciones, cada una con una población de 100 individuos, o un límite de 10 iteraciones sin mejora en la solución encontrada.

El GA se ha ejecutado múltiples veces para cada par de variables de entrada (x_0, T_{max}), con el fin de obtener varias alternativas de parámetros óptimos, y así un conjunto de datos numeroso para posteriormente entrenar los modelos XGBoost. En cada ejecución del GA, el algoritmo recibe en primer lugar los resultados de la simulación según el método estándar, y los compara con cientos de simulaciones llevadas a cabo mediante el enfoque aproximado propuesto, usando diferentes conjuntos de parámetros (T_{sw}, N_1, N_2), generados aleatoriamente. En total, el GA se ejecutó de tal forma que se obtuvo un conjunto de 680 datos, considerando 40 de ellos para cada uno de los 17 valores para la condición inicial, divididos en 10 por cada tiempo máximo de ejecución. La estructura general del conjunto de datos se muestra en la Tabla 4.1.

4.2.2. Entrenamiento modelos XGBoost

Los modelos XGBoost fueron entrenados con el conjunto de datos representado en la Tabla 4.1, de manera tal que puedan determinar, para cualquier condición inicial factible del SOC, una combinación de parámetros que respete el tiempo máximo de ejecución establecido por el usuario, entre 2 y 5 segundos, y a la vez procurando sacrificar un mínimo de precisión en la estimación de EoD. Para ello, el *dataset* se dividió en un 80 % para entrenamiento, mientras que el 20 % restante se usó para validación. Se entrenaron modelos especializados para cada uno de los parámetros (T_{sw}, N_1, N_2), buscando los hiperparámetros óptimos empleando GridSearch. En el siguiente capítulo se explicitan dichos hiperparámetros, y la secuencia en la que se emplearon los modelos. De esta manera, pueden entregar una aproximación para decidir los mejores parámetros de un pronóstico de falla basado en PF, y usando frecuencias de submuestreo. Por supuesto, esto es válido únicamente para el caso específico en que sólo se emplean dos frecuencias, y bajo las condiciones de operación previamente estipuladas para

Tabla 4.1: Representación del conjunto de datos obtenido por el GA. Las columnas representan, de izquierda a derecha: condición inicial del SOC x_0, tiempo máximo de ejecución T_{max}, instante de *switch* de frecuencia T_{sw}, primer divisor de frecuencia N_1, segundo divisor de frecuencia N_2, y valor de función de *fitness* $J(\theta)$.

x_0	T_{max}	T_{sw}	N_1	N_2	$J(\theta)$
1,00	5	T_{sw_1}	$N_{1,1}$	$N_{1,2}$	J_1
1,00	5	T_{sw_2}	$N_{1,2}$	$N_{2,2}$	J_2
⋮	⋮				
1,00	5	$T_{sw_{10}}$	$N_{1,10}$	$N_{2,10}$	J_{10}
1,00	4	$T_{sw_{11}}$	$N_{1,11}$	$N_{2,11}$	J_{11}
⋮	⋮				
1,00	4	$T_{sw_{20}}$	$N_{1,20}$	$N_{2,20}$	J_{20}
1,00	3	$T_{sw_{21}}$	$N_{1,21}$	$N_{2,21}$	J_{21}
⋮	⋮				
1,00	3	$T_{sw_{30}}$	$N_{1,30}$	$N_{2,30}$	J_{30}
1,00	3	$T_{sw_{31}}$	$N_{1,31}$	$N_{2,31}$	J_{31}
⋮	⋮				
1,00	2	$T_{sw_{40}}$	$N_{1,40}$	$N_{2,40}$	J_{40}
0,95	5	$T_{sw_{41}}$	$N_{1,41}$	$N_{2,41}$	J_{41}
⋮	⋮				
0,95	5	$T_{sw_{50}}$	$N_{1,50}$	$N_{2,50}$	J_{50}
0,95	4	$T_{sw_{51}}$	$N_{1,51}$	$N_{2,51}$	J_{51}
⋮	⋮				
0,95	4	$T_{sw_{60}}$	$N_{1,60}$	$N_{2,60}$	J_{60}
0,95	4	$T_{sw_{61}}$	$N_{1,61}$	$N_{2,61}$	J_{61}
⋮	⋮				
⋮	⋮				
⋮	⋮				
0,25	2	$T_{sw_{640}}$	$N_{1,640}$	$N_{2,640}$	J_{640}
0,20	5	$T_{sw_{641}}$	$N_{1,641}$	$N_{2,641}$	J_{641}
⋮	⋮				
⋮	⋮				
0,20	2	$T_{sw_{680}}$	$N_{1,680}$	$N_{2,680}$	J_{680}

el caso de estudio.

4.3. Validación

Para probar la metodología propuesta en esta investigación, se llevaron a cabo muchas simulaciones de pronóstico, usando como condiciones iniciales los datos de *ground truth* SOC durante el ciclo de descarga completo mostrados en [7]. Con el fin de evitar extender excesivamente las simulaciones de pronóstico, se tomó sólo una de cada cinco muestras de la lista de condiciones iniciales, lo que es equivalente a ejecutar el algoritmo de pronóstico cada 5 s de operación de la batería, en tiempo real. De esta forma, se elabora un pronóstico para 1561 valores distintos x_0, que van desde la batería cargada al 100 % hasta un 15 %, que es incluso más bajo que el punto mínimo para el cual los modelos XGBoost fueron entrenados, e incluso debajo del mínimo umbral de carga recomendado en la literatura [24]. Así, es posible evaluar el desempeño para cualquier etapa del ciclo de descarga. Para cada valor del SOC, se ejecutó tanto el algoritmo de pronóstico estándar como el enfoque eficiente propuesto, con el fin de comparar ambos resultados.

El desempeño de la propuesta se ha evaluado en términos de la similitud de estimación del JITP con respecto al pronóstico estándar. Esta métrica se ha escogido ya que permite incorporar el concepto de riesgo [14, 33, 32], útil en *Decision Making* para estimar cuándo el componente monitorizado debería ser sujeto a mantención, o derechamente reemplazarse. Se consideran tres umbrales diferentes para el JITP, de 5 %, 10 % y 15 %.

Capítulo 5

Resultados y discusión

En este capítulo se presentan los principales resultados de esta investigación. En primer lugar, se entregan algunos detalles específicos de la implementación del esquema de pronóstico, ya sea como el filtro de partículas como tal, incluyendo las características del computador empleado, como el algoritmo genético y los modelos XGBoost. Luego, se presentan gráficamente los resultados de divisores de frecuencia óptimos para construir el conjunto de datos de entrenamiento, obtenidos a partir del algoritmo genético, además del diseño de uso de los modelos XGBoost durante la etapa en tiempo real del algoritmo. Posteriormente, se presentan los resultados producidos mediante el pronóstico tradicional, para usarse después como comparación al evaluar el desempeño del enfoque con submuestreo. Finalmente, se muestran los resultados correspondientes al tiempo de ejecución del algoritmo en tiempo real.

5.1. Detalles de implementación de esquema de pronóstico

A continuación se dan algunos detalles sobre la implementación del algoritmo. En primer lugar, todo el código empleado, al igual que las aproximaciones de la dinámica del caso de estudio, se escribió en Python, usando la versión disponible en Colab para todo, excepto los resultados finales de JITP, que se obtuvieron ejecutando las respectivas rutinas en una máquina personal. Se hizo de esta forma pues en este caso se requiere conocer con certeza las características de ésta, lo que no es posible en Colab. El equipo empleado cuenta con un procesador AMD Ryzen 5 4500U de 2.375 GHz y 8 GB de RAM. En el caso del filtro de partículas, se emplearon 500 de estas, y 25 realizaciones de la cadena de Markov que simula la corriente de descarga. Además, se definieron semillas para varios procesos aleatorios involucrados, como la generación de los ruidos de proceso y medición, la generación de la entrada exógena, y la incertidumbre del estado de carga inicial de la batería en cada experimento. Para esta última, en cada caso se consideró una distribución Gaussiana con una desviación estándar de $\sigma_\omega = 10^{-6}$, tal como se hizo en [7]. En la Tabla. 5.1 las principales de estas características.

En el caso de los algoritmos genéticos, ya que se requiere variabilidad en los cromosomas generados, se define de manera aleatoria una semilla distinta para cada ejecución, indepen-

Tabla 5.1: Resumen de características relevantes para la simulación.

Parámetros	**Descripción**
Procesador	AMD Ryzen 5 4500U de 2.375 GHz
Memoria RAM	8 Gb
N° de partículas	500
N° realizaciones cadena de Markov	25
Desviación estándar condición inicial	10^{-6}

diente de la semilla empleada para las simulaciones de descarga de la batería. Se trabajó con 100 individuos por generación, estableciendo un máximo de 40 generaciones, y un máximo de 10 iteraciones sin mejora como condición para detenerse. Además, se definió una probabilidad tanto de mutación como de *crossover* uniforme de 0.9, manteniendo en cada generación un 30 % de padres para la generación siguiente, y una proporción de élite del 1 %.

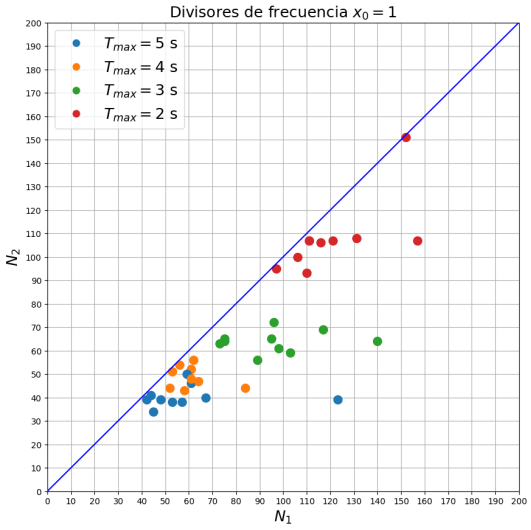
Por otra parte, la mayoría de los hiperparámetros de los modelos XGBoost se determinaron mediante una GridSearchCV, o exploración en cuadrícula. La excepción a esto fue el número de árboles de modelo, que se definió en base a una condición de *early stopping*, o detención temprana, de 10 iteraciones como máximo sin mejora. En la Tabla 5.2 se muestran los valores de los hiperparámetros para los modelos que determinan N_2, N_1 y T_{sw} en cada pronóstico, donde Máx *depth* corresponde a la profundidad máxima de cada árbol, lr el *learning rate*, γ y λ los parámetros de regularización mencionados previamente en la Sección 2.6, mientras que α se refiere a una constante de regularización para la norma L_1 del vector w del árbol respectivo [41].

Tabla 5.2: Valores de los hiperparámetros de los modelos XGBoost para determinar N_2, N_1 y T_{sw} en cada pronóstico. Máx *depth* corresponde a la profundidad máxima de árbol, lr el *learning rate*, mientras que γ, λ y α son los parámetros de regularización.

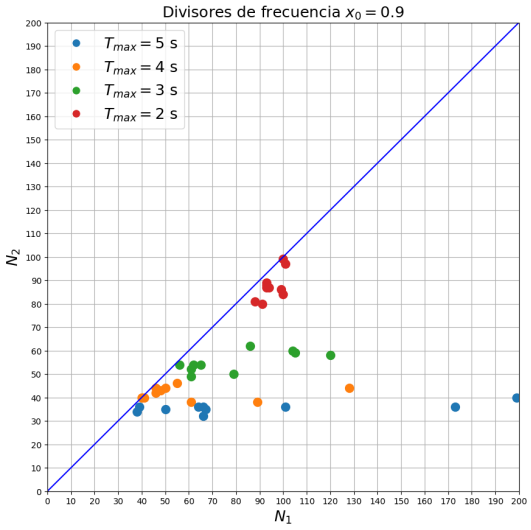
Parámetros	**N° árboles**	**Máx *depth***	**lr**	γ	λ	α
N_2	45	4	0.1	0.25	0	1
N_1	89	4	0.5	0.25	15	10
T_{sw}	29	3	0.2	0	10	0

5.2. Dataset divisores de frecuencia óptimos obtenidos con GA

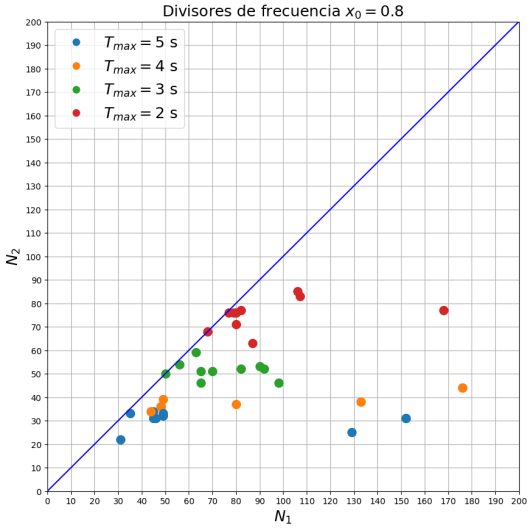
En las Fig. 5.1 y 5.2 se muestran gráficamente los datos obtenidos de los divisores de frecuencia N_1 y N_2, para condiciones iniciales del SOC desde $x_0 = 0{,}3$ hasta $x_0 = 1$, espaciados por una diferencia de $\Delta x_0 = 0{,}1$. Los colores de cada punto indican eñ tiempo máximo de ejecución exigido. Aquí se puede apreciar que en cada gráfico los datos se encuentran por debajo de la diagonal, marcada por una línea azul. Esto se debe a que en la implementación de la función de *fitness* del algoritmo genético se fuerza a siempre escoger el entero más pequeño en cada cromosoma como N_1 y el mayor como N_2, de manera que se cuente con una mayor resolución hacia el final del pronóstico, cuando se acerca el instante de falla.



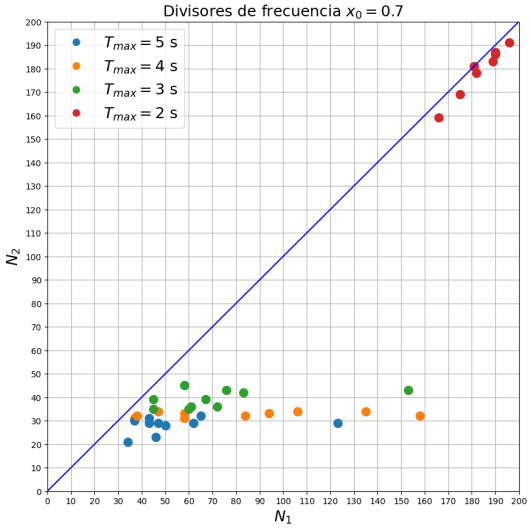
(a) $x_0 = 1$



(b) $x_0 = 0{,}9$

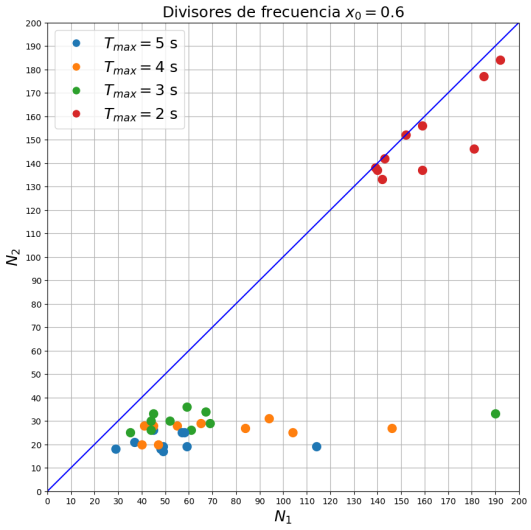


(c) $x_0 = 0{,}8$

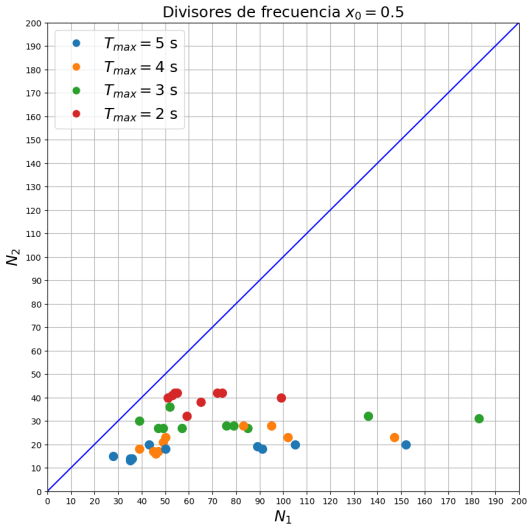


(d) $x_0 = 0{,}7$

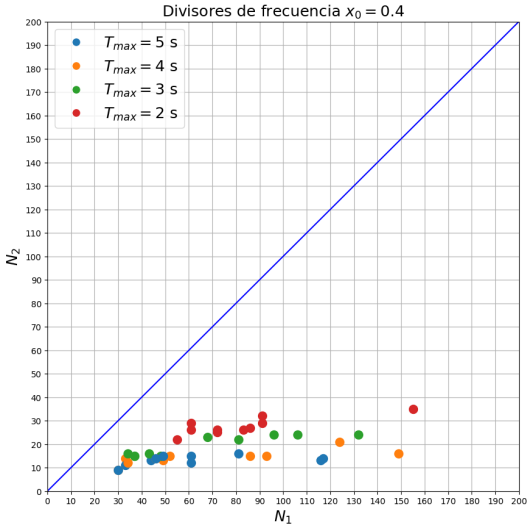
Figura 5.1: Conjunto de datos para entrenamiento y validación de los parámetros N_1 y N_2, y para condiciones iniciales de SOC $x_0 = 1, x_0 = 0{,}9, x_0 = 0{,}8$, y $x_0 = 0{,}7$.



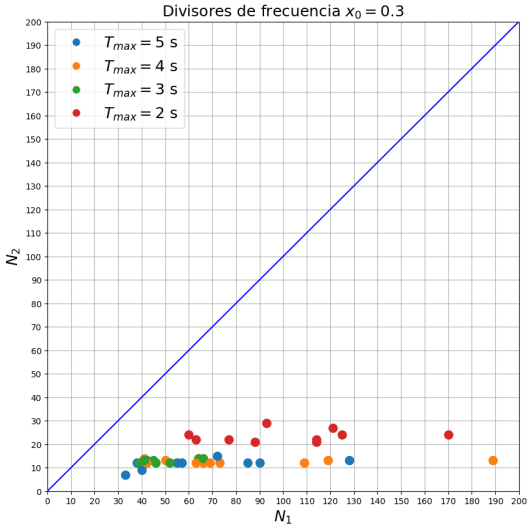
(a) $x_0 = 0{,}6$



(b) $x_0 = 0{,}5$



(c) $x_0 = 0{,}4$



(d) $x_0 = 0{,}3$

Figura 5.2: Conjunto de datos para entrenamiento y validación de los parámetros N_1 y N_2, y para condiciones iniciales de SOC $x_0 = 0{,}6, x_0 = 0{,}5, x_0 = 0{,}4$, y $0{,}3$.

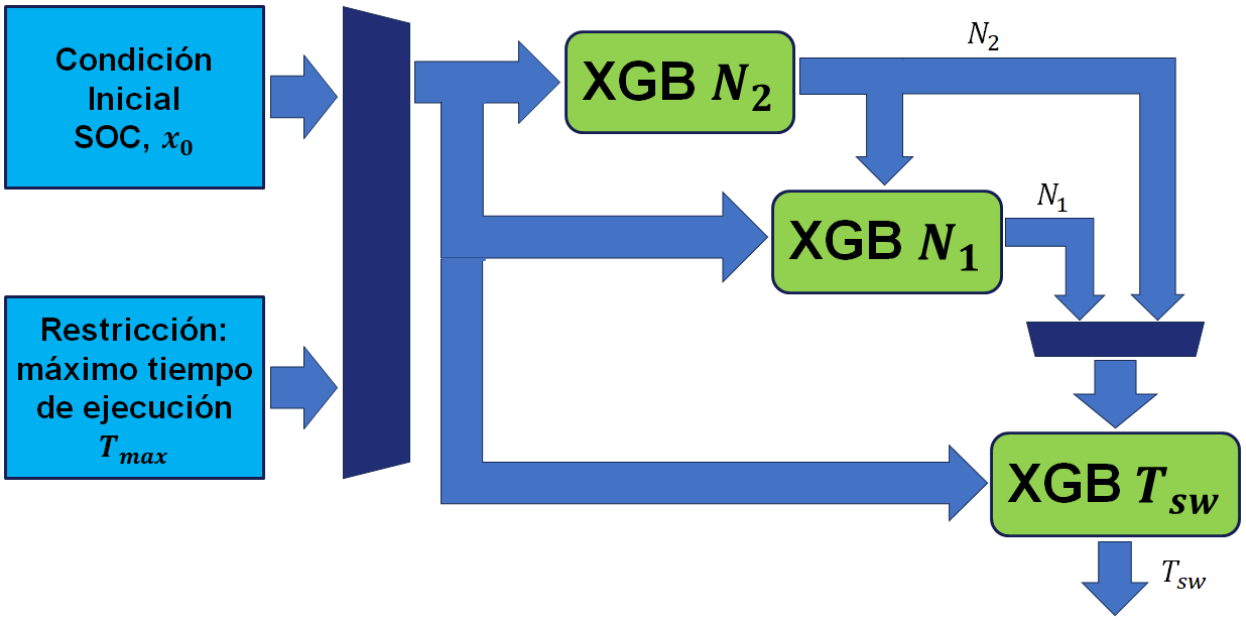


Figura 5.3: Diagrama de flujo de modelos XGBoost para determinar parámetros recomendados para el pronóstico por submuestreo. Las salidas de este módulo son los divisores de frecuencia N_2 y N_1, además del instante de *switch*, T_{sw}.

De las imágenes aquí presentadas, se observa que el parámetro N_2 es el que presenta la menor variabilidad, ya que en cada figura los puntos de igual color, es decir misma restricción para el tiempo de ejecución, tienden a agruparse en posiciones verticalmente cercanas. Esto sugiere que las dos variables originales de entrada, (x_0, T_{max}) proporcionan información suficiente para seleccionar un valor adecuado para la frecuencia de submuestreo hacia el final del pronóstico. Para el parámetro N_1, en contraste, no se identifica un patrón claro basado para valores con las mismas entradas, pues se aprecia una alta variabilidad en la ubicación horizontal de los puntos de igual color.

Otro aspecto a mencionar sobre las Fig. 5.1 y 5.2 es que en los 5 primeros gráficos, los divisores de frecuencia para una restricción de $T_{max} = 2$ s, la más restrictiva, se encuentran más separados del resto de los datos, particularmente para las condiciones iniciales de $x_0 = 0{,}7$ y $x_0 = 0{,}6$. Esto sugiere que especialmente en estos dos últimos casos se le puede estar exigiendo mucho al algoritmo, y por tanto estar cerca de su límite. Sin embargo, estos comportamientos parecen desaparecer al disminuir el SOC al momento en que se inicia el pronóstico, donde todos los datos se encuentran muy cercanos entre sí, independiente de la restricción en tiempo de ejecución, con valores muy pequeños aunque bien determinados para el parámetro N_2, y N_1 con una distribución que aparenta ser aleatoria.

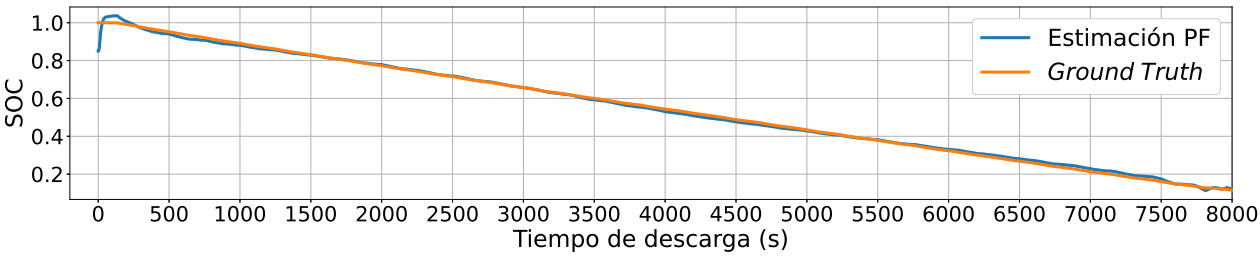
En base a este análisis se decidió la siguiente secuencia de uso de los modelos XGBoost para la etapa en tiempo real del algoritmo. En primer, se usó el modelo para N_2, considerando únicamente (x_0, T_{max}) como variables de entrada. Luego, sigue el modelo para N_1, usando como entradas (x_0, T_{max}) y además N_2, la salida del primero. Finalmente, se emplea modelo para T_{sw}, que recibe tanto (x_0, T_{max}) como (N_1, N_2). La Fig. 5.3 ilustra esta secuencia.

5.3. Resultados pronóstico tradicional

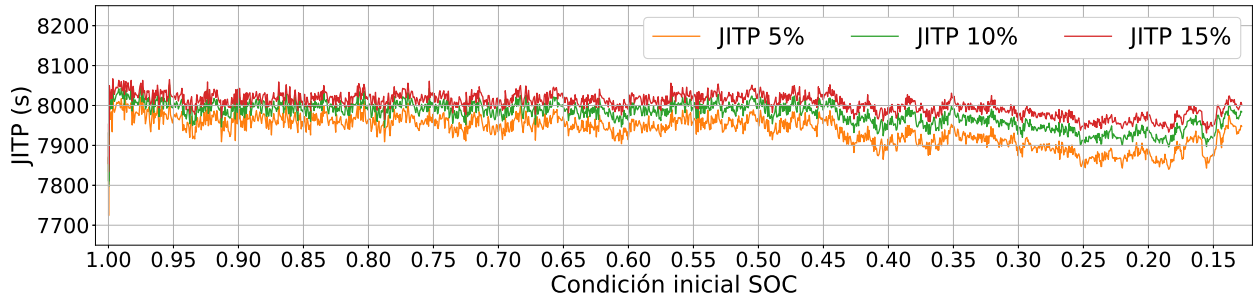
Como ya se mencionó en el capítulo anterior, la principal métrica que se empleó para evaluar el desempeño del esquema de pronóstico propuesto es el JITP para el EoD, en particular su similitud con los resultados obtenidos siguiendo el enfoque tradicional, como se muestra en la Ec. (4.13). Un detalle que se debe mencionar es que en los resultados que se presentan más adelante, el JITP se mide para el tiempo de falla o ToF, y no para el tiempo *hasta* la falla, análogo al RUL, es decir, no se toma como instante $k = 0$ aquel en que se inicia el pronóstico, sino aquel en el cual se comienza a operar la batería. Esto permite poder comparar la evolución de la estimación del JITP durante el ciclo de descarga completo, y en consecuencia, para distintos horizontes de predicción, tomando siempre como referencia el tiempo que ha pasado desde que, en teoría, se comenzó a usar la batería, con un estado inicial de carga cercano al 100 %.

En primer lugar, se presentan los resultados según el método estándar, en la Fig. 5.4. La imagen de más arriba muestra los datos de estimación del SOC durante la etapa de filtrado y el *ground truth*, obtenidos en el procedimiento experimental mencionado en [7]. En la segunda imagen, se puede apreciar la estimación del JITP para los valores de $\alpha = 5\,\%, 10\,\%, 15\,\%$ a medida que se va descargando la batería, donde el eje horizontal representa la condición inicial con la cual se elaboró el pronóstico, mientras que tercera imagen presenta los tiempos de ejecución de dichos pronósticos. Con respecto a las curvas del JITP, se puede apreciar que las tres de ellas se mantienen siempre entre 7800 y 8100 segundos luego del inicio de operación de la batería, lo que significa que, siguiendo el esquema de pronóstico convencional, aproximadamente a las dos horas de uso se espera que se alcance una probabilidad significativa del EoD, bajo el perfil de descarga ya especificado. Las estimaciones muestra cierta inestabilidad menor en un comienzo, aunque posteriormente convergen rápidamente y se mantienen dentro de un intervalo pequeño de menos de 200 instantes. Esto sucede aproximadamente un valor del SOC de $x_0 = 0{,}45$ donde la estimación del JITP desciende, es decir, se espera que el EoD ocurra antes de lo previsto por pronósticos anteriores. Más adelante, dichas estimaciones vuelven a subir, hasta alcanzar sus valores previos.

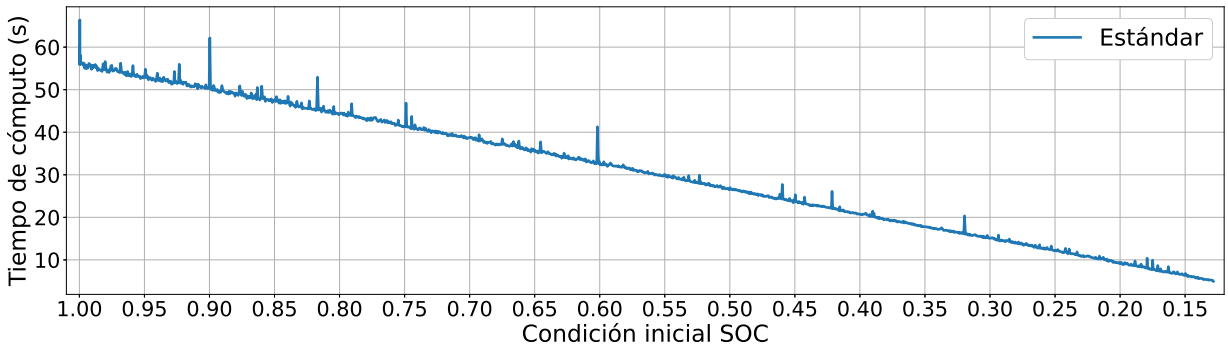
Sobre el tiempo de cómputo, Fig. 5.4c, se aprecia que sigue una tendencia aproximadamente lineal, con sólo algunos pocos *outliers*. Esta característica se explica simplemente porque, ya que la frecuencia de muestreo es siempre la misma, y el horizonte de predicción decrece a medida que se usan condiciones iniciales más cercanas al EoD, por lo tanto se requiere menos tiempo de cómputo para alcanzar ese instante. Al comienzo del ciclo de descarga, cuando la batería está casi completamente cargada, el pronóstico toma aproximadamente un minuto en ejecutarse, mientras que hacia el final éste toma alrededor de 5 segundos, muy cercano al límite máximo de tiempo de cómputo explorado para la metodología aproximada de pronóstico propuesta.



(a) Datos de SOC de estimación durante la etapa de filtrado y *Ground truth*, obtenidos experimentalmente en [7].



(b) Estimación de $JITP_{5\,\%}$, $JITP_{10\,\%}$ y $JITP_{15\,\%}$ para diferentes condiciones iniciales de pronóstico, empleando el enfoque tradicional.



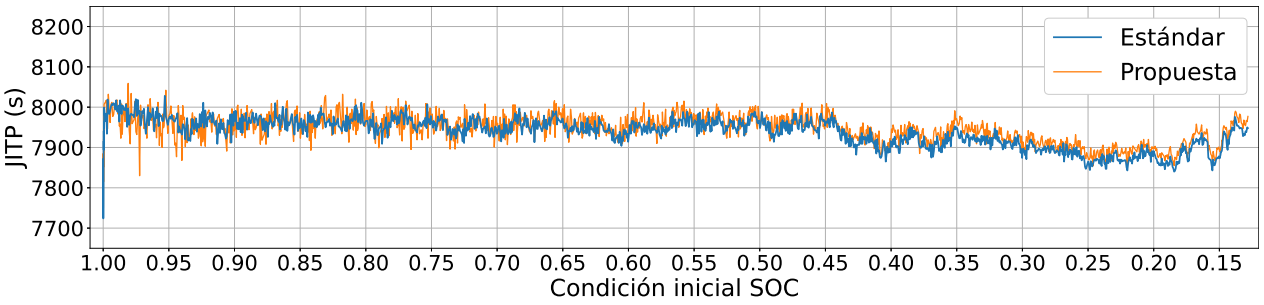
(c) Tiempo de cómputo a lo largo del ciclo de descarga, para distintas condiciones iniciales de pronóstico, usando el enfoque tradicional.

Figura 5.4: Condiciones iniciales de SOC empleadas para pronóstico, resultados de estimación de JITP y tiempo de cómputo según método estándar.

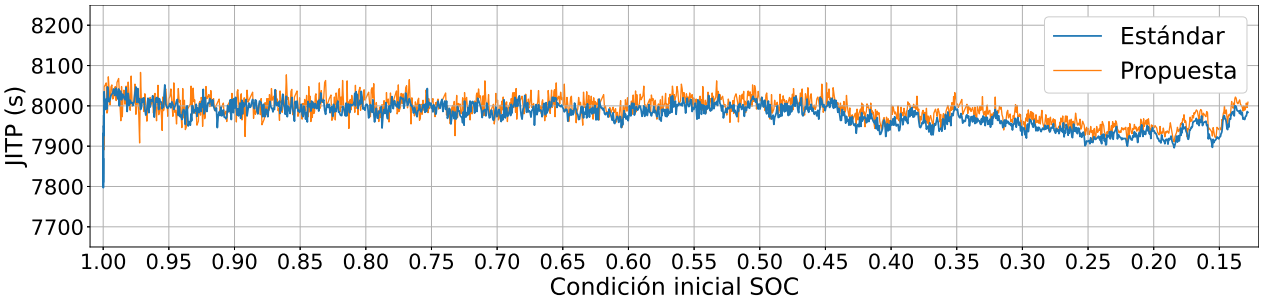
5.4. Precisión de estimación de EoD

Todos los resultados de JITP obtenidos a partir del método propuesto, presentados a continuación, se muestran siempre en comparación con los del enfoque estándar, con el fin de facilitar la comparación entre ambos.

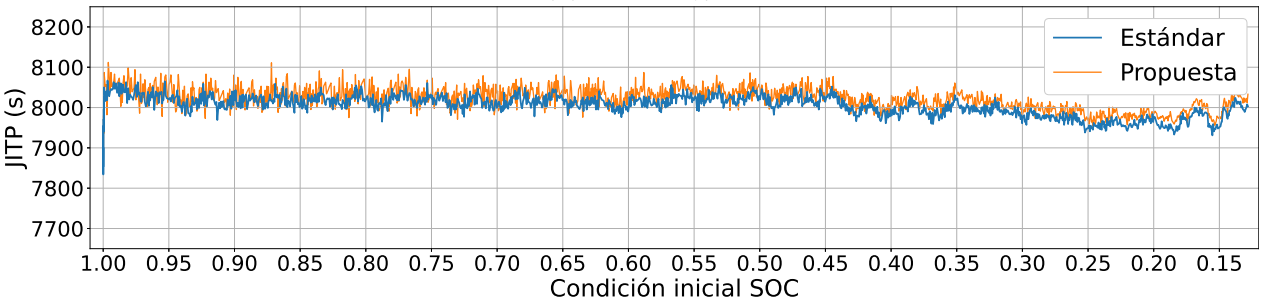
En la Fig. 5.5 se muestran las estimaciones del JITP generadas por el método propuesto, usando durante todo el ciclo de descarga las frecuencias de submuestreo determinadas por los modelos XGBoost, restringiendo el tiempo de cómputo a un máximo de $T_{max} = 5$ s. Como se puede apreciar, los resultados son ligeramente más irregulares y menos conservadoras que los del enfoque estándar empleando la frecuencia de muestreo original, ya que en los tres casos la curva correspondiente (azul) se encuentra levemente por debajo de las curvas naranjas. A pesar de esto, en los comportamientos de ambos métodos se aprecian similitudes visuales



(a) $JITP_{5\,\%}$.



(b) $JITP_{10\,\%}$.



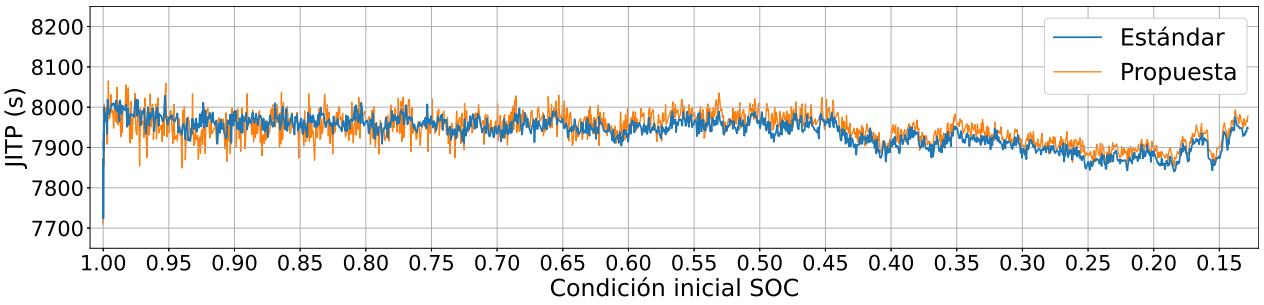
(c) $JITP_{15\,\%}$.

Figura 5.5: Resultados método estándar vs enfoque propuesto, usando frecuencia de submuestreo determinadas por modelos XGBoost, con restricción de tiempo de ejecución $T_{max} = 5$ s.

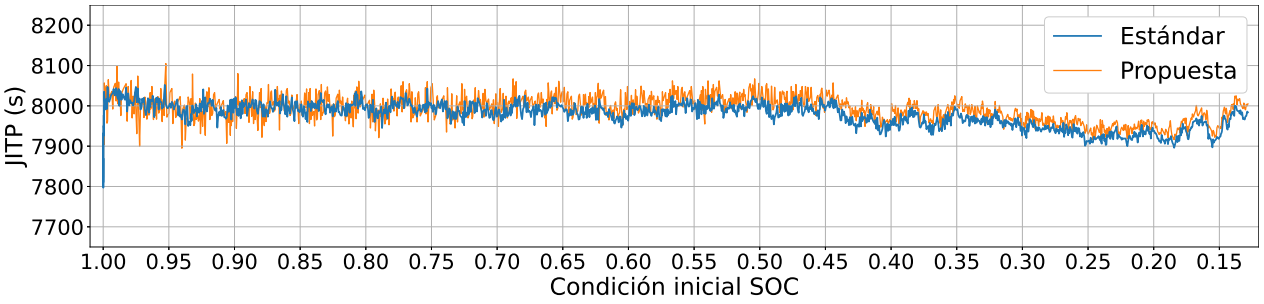
importantes, y las diferencias entre ambas son pequeñas en comparación con el horizonte de predicción, dejando muy claro que se sigue la misma tendencia para cada uno de los tres umbrales de probabilidad del JITP estudiados.

En las 5.6 y 5.7 se muestran, respectivamente, la comparación del enfoque tradicional con el método de submuestreo, pero usando como restricción para los modelos XGBoost un tiempo límite de cómputo de $T_{max} = 4$ s y $T_{max} = 3$ s. Aquí tampoco se aprecian diferencias significativas con la restricción anterior, se mantiene la similitud entre las estimaciones, salvo algunas irregularidades ligeramente más relevantes, pero esperables debido a una ventana de tiempo de ejecución más reducida, y por tanto, el requerimiento de una frecuencia de submuestreo más baja que implica menos precisión.

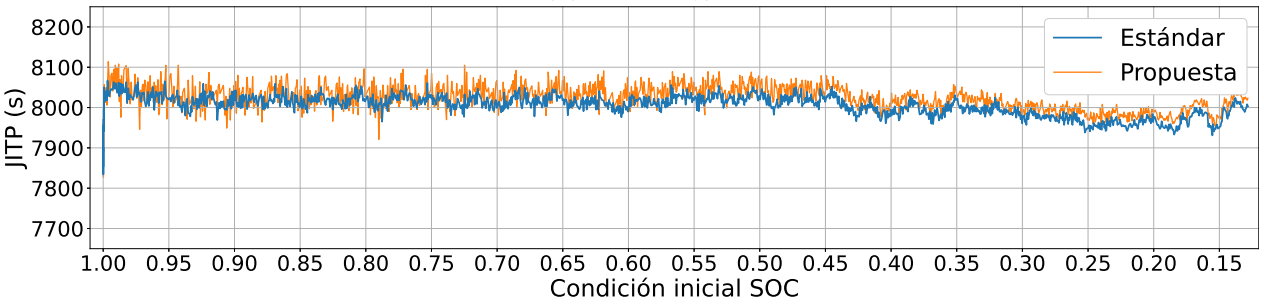
Otro aspecto que se puede observar es cómo hacia el final del ciclo de descarga, independiente del tiempo límite de cómputo, las pequeñas diferencias de estimación del JITP que se aprecian en un comienzo van disminuyendo. Esto podría explicar las características del conjunto de datos que se muestra en la Fig. 5.2, en dónde los valores del segundo divisor de



(a) $JITP_{5\,\%}$.



(b) $JITP_{10\,\%}$.

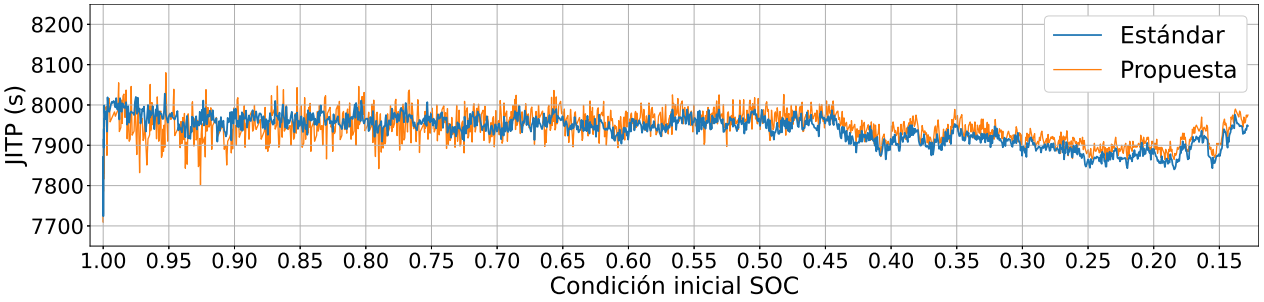


(c) $JITP_{15\,\%}$.

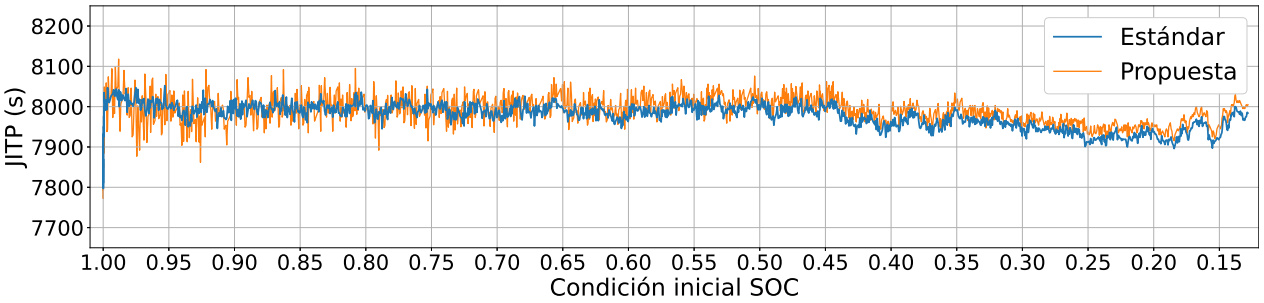
Figura 5.6: Resultados método estándar vs enfoque propuesto, usando frecuencia de submuestreo determinadas por modelos XGBoost, con restricción de tiempo de ejecución $T_{max} = 4$ s.

frecuencia, N_2, aparenta no variar independiente de la restricción de tiempo de ejecución. Ambos comportamientos sugieren que, hacia el final del ciclo de descarga, la frecuencia de submuestro escogida por lo modelos XGBoost converge a la frecuencia original, independiente de la restricción de tiempo de cómputo.

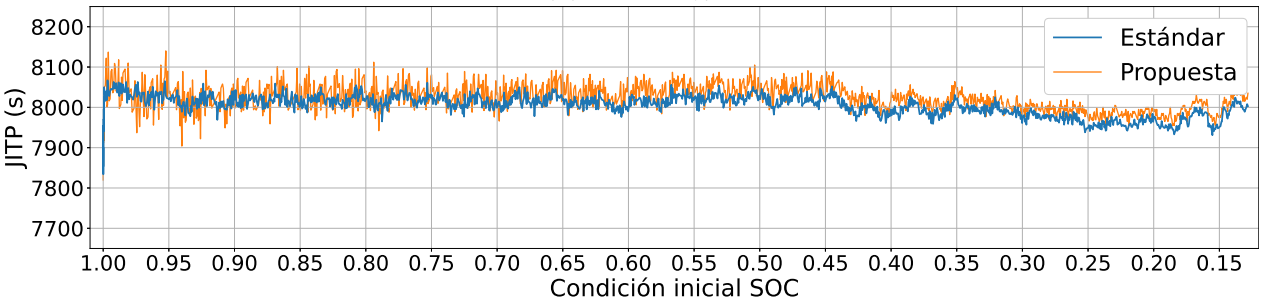
En la Fig. 5.8 se presentan los resultados donde el pronóstico está restringido a un máximo de $T_{max} = 2$ s. En contraste a los resultados previos, aquí es evidente la presencia de diferencias más significativas con el método convencional, especialmente desde el inicio del ciclo de descarga hasta un valor de SOC del $x_0 = 0{,}85$, e incluso irregularidades mayores entre los instantes para condiciones iniciales entre $x_0 = 0{,}85$ y $x_0 = 0{,}55$. Los cambios abruptos del JITP en este intervalo, y para condiciones de operación similares entre sí, significa que el desempeño del algoritmo propuesto bajo la restricción de tiempo de cómputo ya especificada se aleja del óptimo durante esa parte específica del ciclo de descarga. Este problema se podría atribuir al conjunto de datos con el que se entrenaron los modelos XGBoost, ya que como se puede apreciar en Fig. 5.1, para condiciones iniciales aproximadamente entre $x_0 = 0{,}75$ y $x_0 = 0{,}6$, y restringido a $T_{max} = 2$ s, dichos datos de N_2 generalmente se encuentran muy



(a) $JITP_{5\,\%}$.

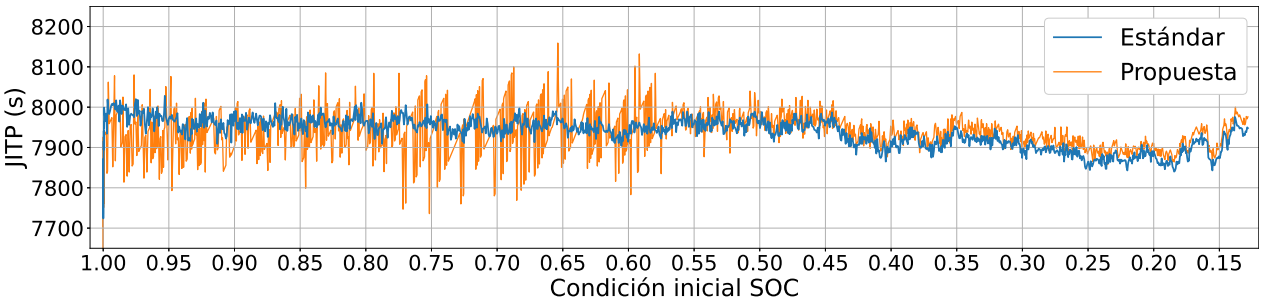


(b) $JITP_{10\,\%}$.

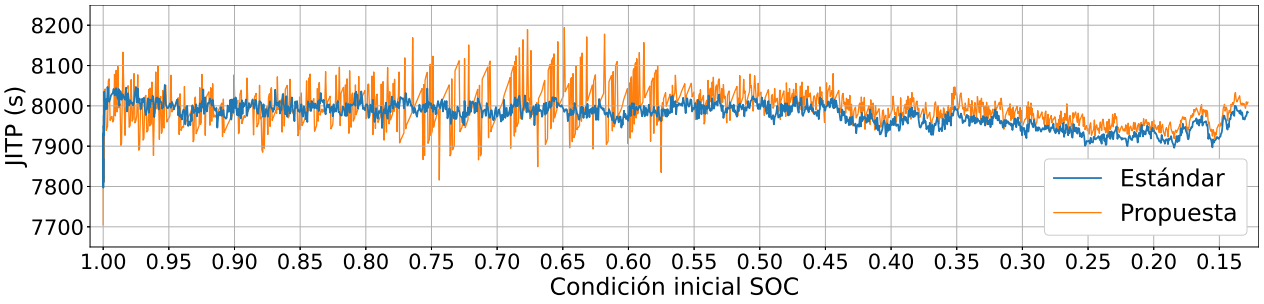


(c) $JITP_{15\,\%}$.

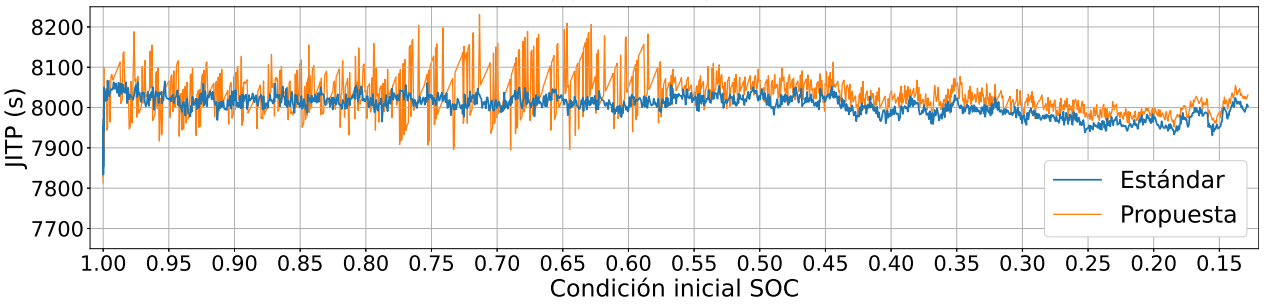
Figura 5.7: Resultados método estándar vs enfoque propuesto, usando frecuencia de submuestreo determinadas por modelos XGBoost, con restricción de tiempo de ejecución $T_{max} = 3$ s.



(a) $JITP_{5\,\%}$.



(b) $JITP_{10\,\%}$.



(c) $JITP_{15\,\%}$.

Figura 5.8: Resultados método estándar vs enfoque propuesto, usando frecuencia de submuestreo determinadas por modelos XGBoost, con restricción de tiempo de ejecución $T_{max} = 2$ s.

cerca del borde del espacio de búsqueda, es decir, toma valores muy altos, y por tanto la resolución temporal es mucho más gruesa que bajo las otras restricciones de tiempo de ejecución. Esto podría sugerir que el óptimo se podría encontrar fuera de la región explorada, o en un caso más extremo, que es muy difícil mejorar la precisión de pronóstico cuando el tiempo límite es demasiado estricto.

Además de la comparación visual de las curvas del JITP, se decidió cuantificar el desempeño del pronóstico empleando frecuencias de submuestreo mediante algunas métricas de error clásicas, evaluando el error a lo largo de todo el ciclo de descarga. Para ello, en la Tabla 5.4 se muestran los valores de las métricas RMSE, MAE and MAPE, todas en comparación con los resultados del enfoque estándar.

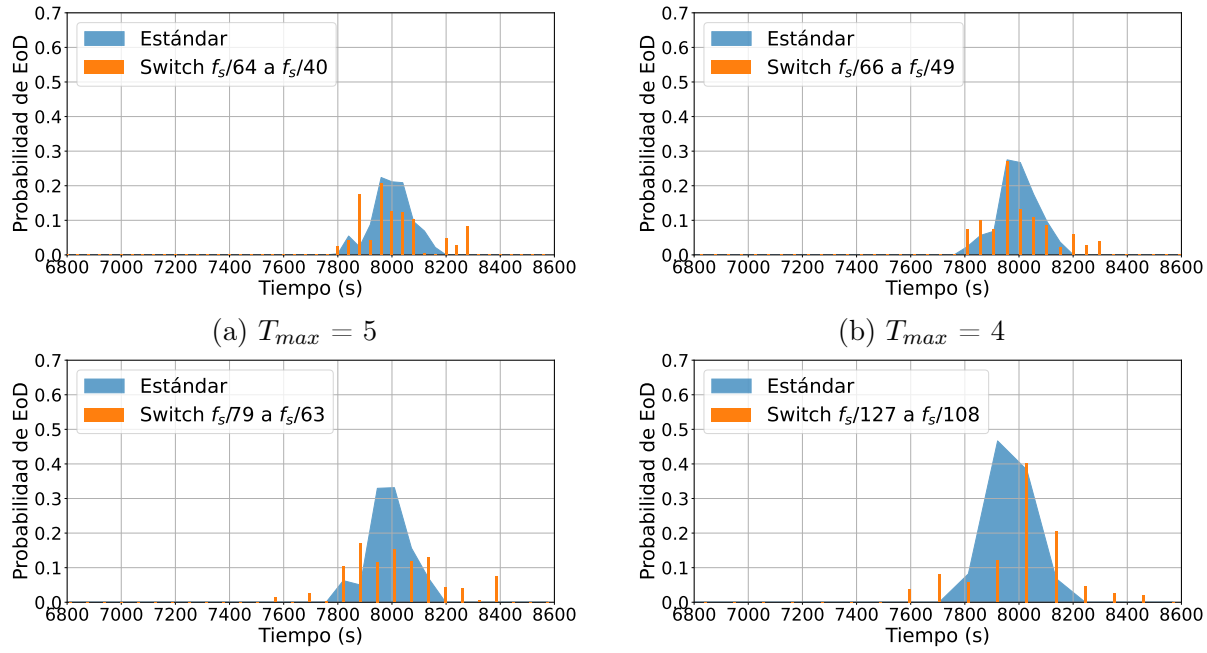
Tal como cabría esperar, las métricas de error son más bajas cuando se cuenta con más holgura en la restricción del tiempo de ejecución. Además, se confirma la similitud entre la estimación del JITP usando el enfoque eficiente en comparación con los resultados del método convencional, ya que los errores que se obtuvieron son siempre menores a un minuto,

Tabla 5.3: Métricas de error del enfoque de pronóstico eficiente, usando frecuencias de submuestreo determinadas por modelos XGBoost.

T_{max} (s)	Umbral de JITP	RMSE (s)	MAE (s)	MAPE
5	5 %	25.94	20.71	0.26 %
	10 %	26.08	21.41	0.27 %
	15 %	26.91	22.26	0.28 %
4	5 %	29.51	23.85	0.30 %
	10 %	28.77	23.69	0.30 %
	15 %	29.80	24.74	0.31 %
3	5 %	32.45	25.92	0.33 %
	10 %	31.92	25.85	0.32 %
	15 %	32.25	26.82	0.34 %
2	5 %	51.99	38.71	0.49 %
	10 %	47.99	37.34	0.47 %
	15 %	52.70	40.92	0.51 %

en un horizonte de predicción de 7800 segundos, es decir, más de dos horas de operación de la batería. Esto puede apreciarse también en la columna de la Tabla que indica el valor del MAPE, donde todos los errores se encuentran por debajo del 1 %, lo que implica que efectivamente, si caso bajo estudio cuenta con un modelo de espacio de estado que presenta una alta exactitud para la estimación del JITP, el enfoque eficiente propuesto puede replicarlo con sólo algunas desviaciones menores, y siendo capaz de evaluar los riesgos de operación durante el resto del ciclo de descarga a pesar de la pérdida de información. Incluso cuando la restricción de tiempo es demasiado estricta, y donde el error es fácilmente apreciable visualmente, como en la Fig. 5.8, las métricas de error son bajas y dan cuenta de un desempeño satisfactorio de la estrategia propuesta.

Para complementar los resultados recién expuestos, a continuación se presentan algunos ejemplos de la función de probabilidad de masa de EoD durante el horizonte de predicción, con el fin de ilustrar el comportamiento de pronósticos individuales. En las Fig 5.9, 5.10 y 5.11 respectivamente, se muestran las distribuciones de probabilidad de falla para condiciones iniciales de la batería de $x_0 = 1$, $x_0 = 0{,}68$ y $x_0 = 0{,}4$. En cada caso, se compara el desempeño del algoritmo estándar con el propuesto, para tiempos de ejecución máximos de $T_{max} = 5$, 4, 3, y 2 s. La leyenda en las imágenes se refiere a las dos frecuencias de muestreo empleadas en el algoritmo propuesto, donde un intervalo de N pasos sin muestrear corresponde a usar una frecuencia de muestreo f_S/N, siendo f_S la frecuencia original del sistema. Cabe mencionar, previo a mostrar los resultados, que el eje x en cada gráfico representa el tiempo contabilizado desde el inicio de operación de la batería con carga completa, de forma tal que las figuras permitan además comparar la evolución del pronóstico dependiendo de en qué punto del ciclo de descarga se ejecuta el algoritmo. Por otro lado, dado que en el caso estándar la distribución de probabilidad puede ser distinta de cero en cualquier instante, no así para el pronóstico eficiente donde esto sólo es posible en los instantes de muestreo, los valores de la probabilidad en cada instante según el método tradicional serán en general menores, pues dicha probabilidad se distribuye entre más instantes. Esto dificulta efectuar comparaciones visualmente, por tanto, para sortear este inconveniente, y únicamente para efectos de las figuras, se ha decidido condensar las probabilidades del caso estándar en los instantes de



(a) $T_{max} = 5$ (b) $T_{max} = 4$

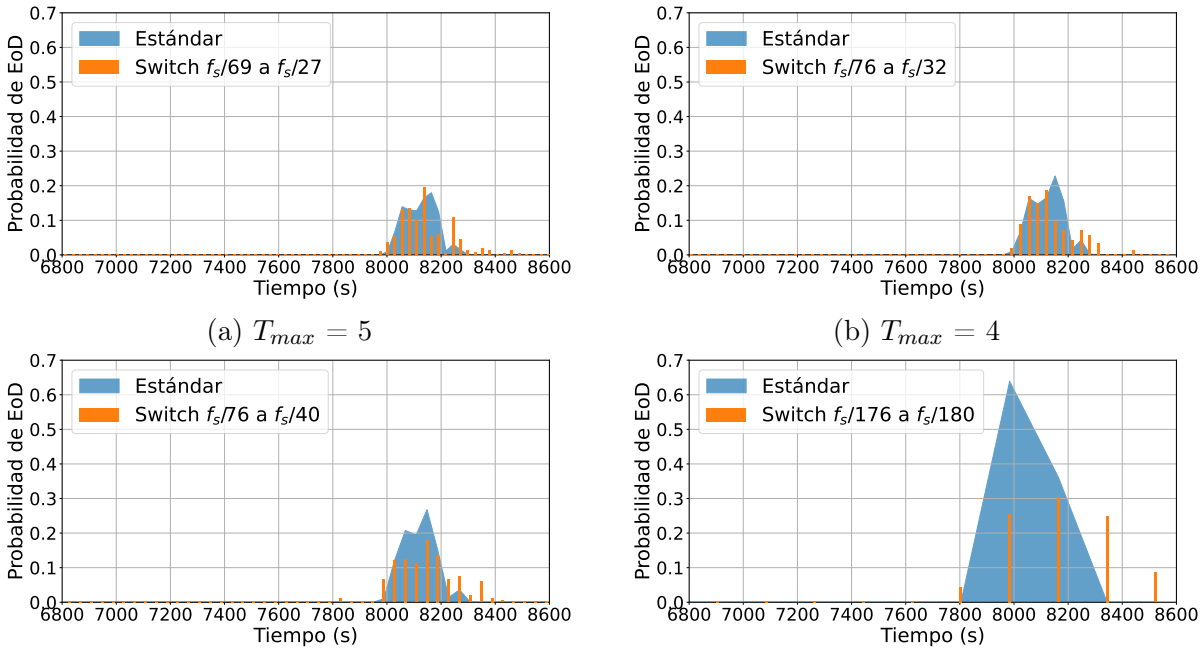
(c) $T_{max} = 3$ (d) $T_{max} = 2$

Figura 5.9: Distribución de probabilidad de masa de EoD durante el horizonte de pronóstico, para $x_0 = 1$.

muestreo del método propuesto.

En Fig. 5.9 se puede apreciar que, efectivamente, la distribución obtenida mediante el método eficiente muestra una similitud visual con el caso estándar cuando se permite un tiempo máximo de ejecución del algoritmo de 4 o 5 s, con $x_0 = 1$. Esto se da más claramente hacia el extremo izquierdo de ambas distribuciones, que coincide justamente con la zona de mayor interés para el análisis de riesgo de falla. Es más, en el caso de $T_{max} = 5$ s, la magnitud de las tres primeras barras de según el pronóstico sugiere que éste es levemente más conservador que la referencia, lo que en un sistema en tiempo real implicaría una alerta más temprana. En los otros dos casos, cuando el tiempo máximo de ejecución es de 2 o 3 s, si bien la forma de la distribución estimada por el algoritmo propuesto es algo más pareja y ancha que la estándar, la probabilidad sigue concentrándose principalmente en la misma zona.

En el caso de $x_0 = 0{,}68$, Fig. 5.10, se repite lo visto anteriormente para $T_{max} = 4$ o 5 s: la distribución entregada por el algoritmo eficiente se ajusta visualmente a la forma del estándar, principalmente por el lado izquierdo. Incluso cuando $T_{max} = 3$ s se puede apreciar una similitud considerable, aunque las discrepancias comienzan a ser más notorias debido a la mayor exigencia en cuanto al tiempo de cómputo. Muy distinto es el caso de $T_{max} = 2$ s, donde el intervalo de muestreo es lo suficientemente extenso como para distorsionar las formas de las distribuciones, perdiendo tanto el caso estándar como el eficiente las similitudes con los 3 anteriores casos, algo que no se dio para $x_0 = 1$. Si bien este es un resultado negativo para estas condiciones de operación particulares, va en línea con lo mostrado previamente en los resultados de JITP, donde el comportamiento más irregular del algoritmo de pronóstico se da aproximadamente para valores entre $x_0 = 0{,}85$ y $x_0 = 0{,}55$, intervalo que contiene la



(a) $T_{max} = 5$ (b) $T_{max} = 4$

(c) $T_{max} = 3$ (d) $T_{max} = 2$

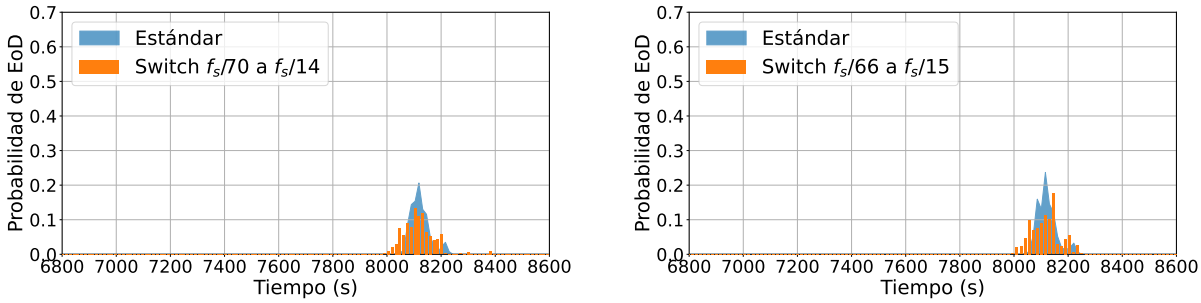
Figura 5.10: Distribución de probabilidad de masa de EoD durante el horizonte de pronóstico, para $x_0 = 0{,}68$.

condición inicial analizada en Fig. 5.10.

Por último, en Fig. 5.11 se presentan las distribuciones de probabilidad para $x_0 = 0{,}4$. Se repiten varias de las características vistas previamente en el caso $x_0 = 0{,}68$, como la similitud visual entre los resultados de ambos métodos para $T_{max} = 3$, 4 o 5 s, especialmente hacia el extremo izquierdo de la distribución. Incluso en el escenario límite $T_{max} = 2$ s, se conserva cierta similitud, a pesar de que la distribución estimada por el método eficiente es levemente más ancha. Todo esto podría explicarse porque, al estar más cerca del instante de falla, el horizonte de pronóstico es más reducido, y por tanto, los tiempos máximos de ejecución permiten usar frecuencias más cercanas a la original, y por tanto, evitando así demasiada pérdida de precisión.

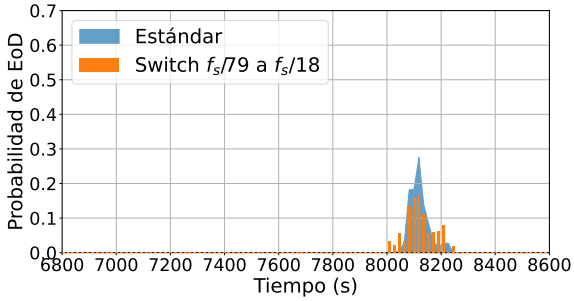
Además de la comparación visual entre las distribuciones ya presentadas, se decidió cuantificar la similitud de éstas mediante la divergencia de Jensen-Shannon (JS) [46]. Para esta métrica, un valor nulo significa que ambas distribuciones son idénticas, mientras el máximo valor posible es 1; mientras más cerca de este valor, más distintas son las distribuciones. En la Tabla 5.4 se muestran los valores calculados para cada uno de los 12 casos ya mostrados. Para estos cálculos se consideró la distribución según el método tradicional tal como se calculó mediante las simulaciones, es decir, la probabilidad de falla puede ser no nula en cualquier instante, no sólo en aquellos que el método eficiente muestrea.

En primer lugar, se puede apreciar que, en general, las distribuciones pierden similitud a medida que se reduce el tiempo máximo de cómputo, tal como cabe esperarse. Las dos excepciones de este comportamiento ocurren para $x_0 = 1$ y $x_0 = 0{,}4$, al pasar de $T_{max} = 5$ s a 4 s, donde dicha similitud se incrementa ligeramente. Fuera de eso, también se observa que

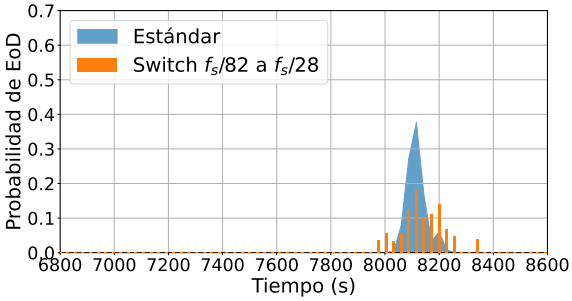


(a) $T_{max} = 5$

(b) $T_{max} = 4$



(c) $T_{max} = 3$



(d) $T_{max} = 2$

Figura 5.11: Distribución de probabilidad de masa de EoD durante el horizonte de pronóstico, para $x_0 = 0{,}4$.

Tabla 5.4: Divergencia de Jensen-Shannon (JS) entre las distribuciones de probabilidad de EoD según el método estándar y eficiente.

x_0	T_{max} (s)	Div de JS
1.00	5	0.63507
	4	0.63480
	3	0.66255
	2	0.68509
0.68	5	0.62506
	4	0.62941
	3	0.64023
	2	0.65623
0.40	5	0.57911
	4	0.57138
	3	0.59781
	2	0.61181

la divergencia disminuye cuando la batería está más descargada al inicio del pronóstico, y por tanto se encuentra más cerca del instante de falla. Este fenómeno es mucho más notorio cuando $x_0 = 0{,}4$, donde incluso la peor puntuación es menor a todas los que se obtuvieron para $x_0 = 1$ y $x_0 = 0{,}68$, lo que coincide con los análisis visuales hechos previamente.

Para finalizar el análisis en cuanto a la precisión de la estimación de EoD, en las Tablas 5.5, 5.6 y 5.7 se muestran los valores de JITP calculados para las condiciones iniciales $x_0 = 1$, $x_0 = 0{,}68$ y $x_0 = 0{,}4$, para ambos métodos, junto con los errores respectivos. Al igual que en los gráficos previamente presentados, el para la métrica del JITP se considera como instante cero aquel en que se comienza a operar la batería con carga completa. Por otro lado, también es necesario destacar que, dentro de una misma tabla, el valor del JITP según el método estándar, para un mismo umbral pero distinto tiempo de cómputo, es idéntico ya que dicha restricción no aplica para esa estrategia. Se decidió mantener este formato ya que es el mismo que se emplea en la Tabla 5.4. Los errores porcentuales, mostrados en la última columna, se calculan como la razón entre el Error en segundos dividido por $JITP_B$, el JITP según el enfoque tradicional, multiplicado por 100 para llevar la cifra a porcentaje.

Tabla 5.5: Valores de JITP para $x_0 = 1$, y error de enfoque por submuestreo en comparación con el método estándar. $JITP_B$ (*baseline*) corresponde a la métrica según el método estándar, y $JITP_A$ de acuerdo al método aproximado.

T_{max} (s)	Umbral de JITP	$JITP_B$ (s)	$JITP_A$ (s)	**Error (s)**	Error %
5	5 %	7875	7840	**35**	0.44
	10 %	7933	7880	**53**	0.67
	15 %	7956	7880	**76**	0.97
4	5 %	7875	7809	**66**	0.84
	10 %	7933	7858	**75**	0.95
	15 %	7956	7858	**98**	1.24
3	5 %	7875	7820	**55**	0.70
	10 %	7933	7820	**113**	1.43
	15 %	7956	7883	**73**	0.93
2	5 %	7875	7705	**170**	2.16
	10 %	7933	7705	**228**	2.87
	15 %	7956	7813	**143**	1.70

En la Tabla 5.5 se logra apreciar que los errores de estimación del JITP del método por submuestreo, tomando como referencia el valor obtenido mediante el enfoque tradicional, presenta variaciones importantes, entre 35 segundos como mínimo, hasta 228 segundos, un poco menos de 4 minutos, como máximo. Sin embargo, considerando que la batería está cargada completamente, y por tanto restan aún más de 2 horas de operación bajo el perfil de uso ya explicitado previamente, este error de estimación es muy pequeño. Esto se puede notar también gracias a los valores de los errores porcentuales, estando más de la mitad por debajo del 1 %. Tal como cabe esperar, el error en general aumenta cuando el tiempo máximo de cómputo disminuye, y por tanto se le exige al algoritmo sacrificar precisión por eficiencia.

Algo importante de notar en la Tabla 5.6, $x_0 = 0{,}68$, es el hecho de que los errores en general disminuyen al compararlos con el caso anterior. Esto concuerda con el análisis hecho previamente sobre las distribuciones de probabilidad: a medida que se acerca el instante

Tabla 5.6: Valores de JITP para $x_0 = 0{,}68$, y error de enfoque por submuestreo en comparación con el método estándar. $JITP_B$ (*baseline*) corresponde a la métrica según el método estándar, y $JITP_A$ de acuerdo al método aproximado.

T_{max} (s)	Umbral de JITP	$JITP_B$ (s)	$JITP_A$ (s)	**Error (s)**	Error %
5	5 %	8049	8029	**20**	0.25
	10 %	8061	8056	**5**	0.06
	15 %	8072	8056	**16**	0.20
4	5 %	8049	8024	**25**	0.31
	10 %	8061	8024	**37**	0.46
	15 %	8072	8056	**16**	0.20
3	5 %	8049	7988	**61**	0.76
	10 %	8061	8028	**33**	0.41
	15 %	8072	8028	**44**	0.55
2	5 %	8049	7984	**65**	0.81
	10 %	8061	7984	**77**	0.96
	15 %	8072	7984	**88**	1.09

de descarga, las distribuciones de probabilidad presentan más similitudes, lo que se podría explicar porque la tasa de muestreo se acerca más a la original. Por otro lado, se puede observar también que todos los errores porcentuales de JITP, con la excepción de uno de ellos, están por debajo del 1 %, y en cuanto al error en segundos, solamente en cuatro casos se supera un minuto, mostrando así que el error sigue siendo reducido en comparación con el horizonte de predicción.

Tabla 5.7: Valores de JITP para $x_0 = 0{,}40$, y error de enfoque por submuestreo en comparación con el método estándar. $JITP_B$ (*baseline*) corresponde a la métrica según el método estándar, y $JITP_A$ de acuerdo al método aproximado.

T_{max} (s)	Umbral de JITP	$JITP_B$ (s)	$JITP_A$ (s)	**Error (s)**	Error %
5	5 %	8084	8034	**50**	0.62
	10 %	8089	8048	**41**	0.51
	15 %	8094	8062	**32**	0.39
4	5 %	8084	8041	**43**	0.53
	10 %	8089	8056	**33**	0.41
	15 %	8094	8056	**38**	0.47
3	5 %	8084	8028	**56**	0.69
	10 %	8089	8046	**43**	0.53
	15 %	8094	8082	**12**	0.15
2	5 %	8084	8004	**80**	0.99
	10 %	8089	8032	**57**	0.71
	15 %	8094	8060	**34**	0.42

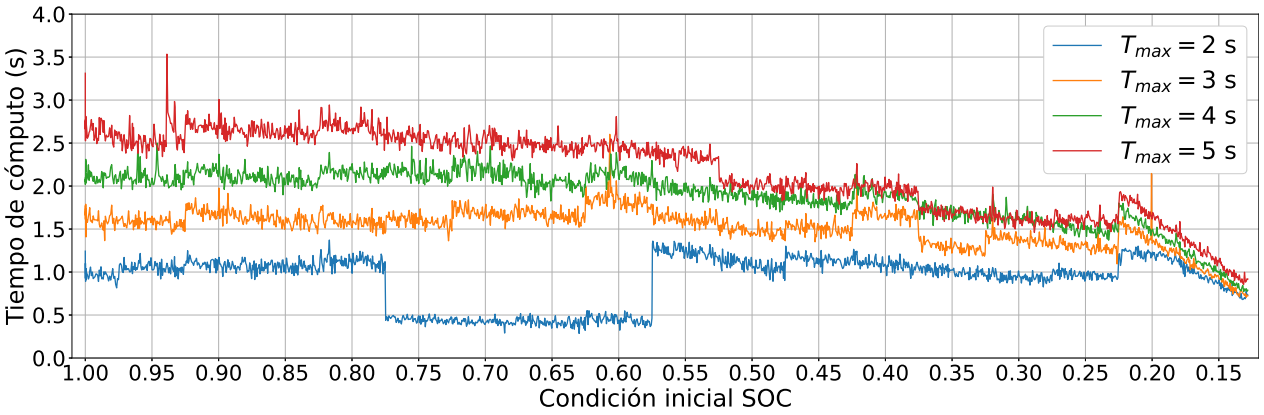
Por otro lado, en la Tabla5.7 no se distingue a primera vista una clara disminución del error con respecto a la condición inicial analizada anteriormente. Sin embargo, al observar detenidamente, se puede notar que aquí todos los errores porcentuales son menores al 1 %, mientras que el error en segundos sólo en un caso es superior a un minuto, mostrando nueva-

mente que hacia el final del ciclo de descarga aumenta la concordancia entre ambos métodos.

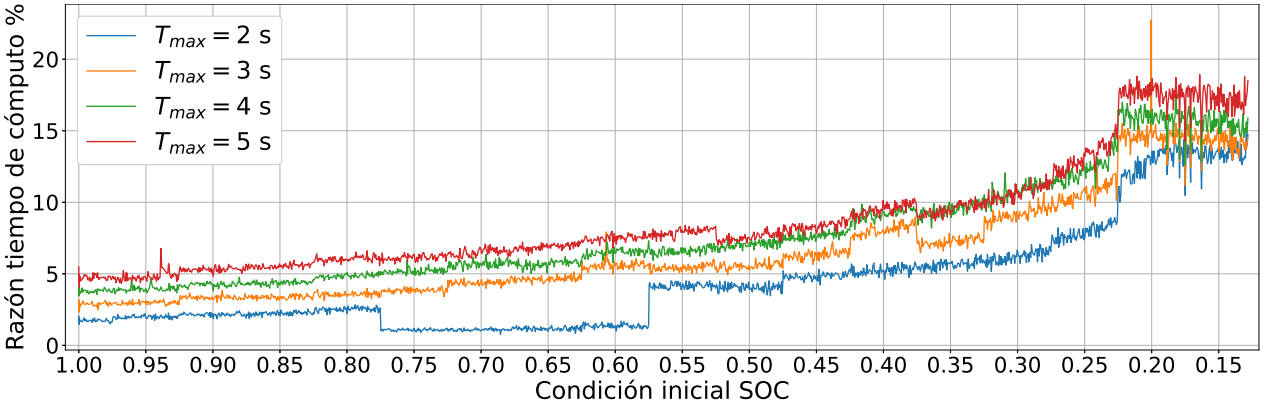
Luego de este análisis, es posible ver que en general las pérdidas de precisión del enfoque por submuestreo son menores en comparación al método estándar. A continuación, se estudiará la diferencia en cuanto a los tiempos de ejecución de ambos algoritmos.

5.5. Tiempo de ejecución

Finalmente, es necesario comparar el rendimiento del pronóstico en términos del esfuerzo computacional. En la Fig. 5.12 se muestra el tiempo de cómputo del esquema propuesto, primero en segundos, y luego como porcentaje del tiempo que le tomaría al algoritmo convencional, en 5.4c.



(a) Tiempo de ejecución en segundos.



(b) Razón porcentual entre los tiempos de ejecución del pronóstico eficiente y el *baseline*, 5.4c.

Figura 5.12: Tiempos de ejecución del esquema de pronóstico propuesto para distintas condiciones iniciales del SOC.

Como primera observación, es claro que la propuesta de pronóstico eficiente conlleva un ahorro significativo de recursos computacionales, reduciendo el tiempo de ejecución del algoritmo desde casi un minuto al inicio del ciclo de descarga, como indica la Fig. 5.4c, a menos de 3 segundos, salvo unos pocos casos aislados. Ya que el tiempo de ejecución del método convencional decrece linealmente al avanzar el ciclo de descarga, y las variaciones del enfoque eficiente son mucho más pequeñas, el ahorro de recursos disminuye cuando la batería se

acerca al instante de EoD, como se aprecia al final del ciclo en la Fig. 5.12b. Sin embargo, se puede apreciar que incluso hasta el punto en que el pronóstico se ejecuta con un valor del SOC de $x_0 = 0{,}3$, equivalente a casi un 77 % del horizonte de predicción según la Fig. 5.4a, el pronóstico eficiente demora en el peor caso sólo un 10 % del tiempo de cómputo del método que usa la frecuencia de muestreo original, es decir, es 10 veces más rápido o más, y al mismo tiempo, mantiene un nivel satisfactorio de precisión en la estimación del JITP, como ya se demostró previamente.

Otro aspecto que debe mencionarse es el hecho de que el pronóstico eficiente no usa al máximo el tiempo límite de cómputo, sino que sólo alrededor de la mitad de éste. Esto se debe a que los datos de entrenamiento para los divisores de frecuencia fueron generados en Colab, pero los resultados finales de pronóstico se obtuvieron en un computador personal con mayor poder de cómputo que Colab, y cuyas características son conocidas, como se menciona en la Sección 5.1. Por una parte, esto asegura que la restricción de tiempo de ejecución siempre se satisface con holgura, aunque por otro lado, implica que el desempeño en precisión de la estimación de JITP puede mejorar incluso más, al costo de aumentar levemente el tiempo de cómputo, pero estando aún muy por debajo de lo que toma el enfoque tradicional. De todos modos, esto no invalida la comparación con éste último, pues ambos se obtuvieron en la una máquina local, cuyas propiedades son conocidas.

Hay dos últimos comentarios que hacer sobre los gráficos de esfuerzo computacional. En primer lugar, no es difícil apreciar que la propuesta de pronóstico eficiente presenta un tiempo de ejecución inferior al promedio durante una sección del ciclo de descarga cuando está restringido a $T_{max} = 2$ s, en específico en el intervalo de valores de SOC entre $x_0 = 0{,}80$ y $x_0 = 0{,}55$. Esto coincide justamente con el intervalo en la Fig. 5.8 ya analizada, donde la propuesta muestra su peor desempeño, posiblemente debido a una elección sub-óptima de divisores de frecuencia por parte del modelo XGBoost, generando así un pronóstico más eficiente, pero al costo de un error más significativo y estimaciones de JITP más irregulares y menos fiables. En segundo lugar, también es posible apreciar que durante la segunda mitad del horizonte de predicción existe una superposición de las curvas correspondientes a las restricciones $T_{max} = 5$ s y $T_{max} = 4$ s para el tiempo de ejecución, es decir que el esfuerzo computacional en ambos casos no se diferencia significativamente como se exige. Estos dos problemas aquí presentados podrían ser atribuidos al hecho de que los datos en entrenamiento se obtuvieron resolviendo un problema de optimización de manera numérica pero no analítica, en particular mediante un algoritmo genético, por tanto el grado de cumplimiento de las restricciones no está garantizado. Sin embargo, a pesar de esto, dichos problemas aparecen sólo durante una sección del horizonte de predicción, y aún sin teniendo esto en cuenta, el ahorro significativo de recursos computacionales sigue siendo considerable, y sin pérdida excesiva de exactitud en las predicciones como muestra la Tabla 5.4, por lo que no se contradice con la principal hipótesis de esta investigación.

Todos estos resultados ya presentados sugieren que el enfoque de pronóstico eficiente propuesto en esta investigación, basado en usar frecuencias de muestreo menores a la original, es en efecto, apropiado para elaborar pronósticos de falla basados en filtro de partículas, al menos para este caso de estudio en particular, bajo las condiciones de operación estipuladas, durante una mayor parte del ciclo de descarga. Esto pues, los experimentos aquí reportados muestran que a un costo menor en cuanto a exactitud en la estimación del JITP, con respecto

al método estándar, es posible reducir drásticamente el tiempo de ejecución de un algoritmo de pronóstico, es decir, una reducción importante en el uso de recursos computacionales. Con esto, se puede concluir que efectivamente, puede ser provechoso explorar frecuencias de submuestreo alternativas para ejecutar algoritmos de pronóstico, no sólo en el mismo problema bajo otras condiciones, por ejemplo un perfil de descarga diferente, sino también otros sistemas no relacionados, ya sea empleando la metodología aquí expuesta o alguna distinta.

Capítulo 6

Conclusión

Se ha presentado un esquema de pronóstico de fallas de bajo costo computacional, basado en emplear frecuencias de submuestreo distintas a la original, cuyo valor depende del estado actual de la batería y una restricción de tiempo de ejecución del algoritmo. Para ello, se exploraron distintas tasas de muestro mediante algoritmos genéticos, usando como función de *fitness* el error porcentual entre la estimación del JITP de $5\,\%, 10\,\%, 15\,\%$ según el método estándar que usa la frecuencia original, y el enfoque propuesto. Con estos datos, se entrenaron modelos XGBoost capaces de determinar las mejores frecuencias de muestreo en cada etapa del ciclo de descarga de la batería. Finalmente, se comprobó la eficacia y eficiencia de la estrategia propuesta, elaborando pronósticos tomando como condiciones iniciales un amplio rango de valores de SOC que abarcan todo el ciclo de descarga, y con restricciones de tiempo de cómputo de $5, 4, 3$ y 2 segundos. Al contrastar las estimaciones del JITP obtenidas de esta forma con los resultados según el enfoque estándar, se aprecia visualmente una gran similitud entre las curvas durante todo el ciclo de descarga, y siempre que la restricción de tiempo de ejecución sea mayor a 2 segundos, situación en que las discrepancias se vuelven más significativas. Esta similitud se comprueba al observar las métricas de error promedio del JITP, que son de menos de un minuto incluso en el peor caso, cuando se exige un pronóstico que se ejecute en sólo 2 segundos, y que apenas pasan medio minuto en el resto de los casos, errores que son muy pequeños comparados con un horizonte de predicción de casi 2 horas. Esto se reafirma al observar la comparación de las distribuciones de probabilidad de EoD, y finalmente el error de estimación del JITP para ciertos casos particulares. Por otro lado, en cuanto a eficiencia se demuestra una drástica disminución del esfuerzo computacional, donde la estrategia propuesta puede demorar desde un $20\,\%$ del tiempo de ejecución de la metodología estándar hacia el final del ciclo de descarga, hasta incluso sólo un $5\,\%$ cuando la batería se encuentra cerca de un estado de carga completa. En base a estos análisis, se comprueba que, a pesar de un posible perfeccionamiento que pueda requerir la metodología aquí descrita, **es posible efectuar pronósticos de falla basados en filtro de partículas que sean eficientes y aplicables a una operación en tiempo real, mediante una disminución de la frecuencia de muestreo**.

Si bien ya se ha reportado una serie de esfuerzos exitosos por encontrar formas efectivas y eficiente de elaborar pronósticos de falla en tiempo real, estos no son excluyentes con la metodología, sino que complementarios. En aplicaciones reales, será decisión del operador a

cargo el decidir cuáles herramientas se emplearán para reducir el tiempo de cómputo de los algoritmos de pronóstico, lo cuál dependerá de las características del sistema estudiado.

Como ya se especificó previamente, los resultados aquí presentados apuntan principalmente a las ventajas del enfoque de pronóstico propuesto en comparación con el método tradicional, pero sin dar demasiada credibilidad al modelo empleado para representar la descarga de la batería, ya que corresponde a una primera prueba. Por tanto, aún es necesario efectuar pruebas con modelos más realistas. Además, esta investigación ha considerado un perfil corriente específico, y una caracterización particular de éste mediante cadenas de Markov. Por tanto, de momento, no se ha comprobado su desempeño para casos en que la corriente de descarga presenta un comportamiento distinto, o ésta se modele empleando una metodología distinta. Otra desventaja considerable tiene relación con el tiempo requerido para obtener, en el estudio fuera de línea, un conjunto de datos numeroso como para entrenar correctamente los modelos XGBoost.

Esta investigación deja abierta varias posibilidades interesantes de trabajo futuro, como por ejemplo, estudiar en detalle si acaso existe un momento durante el ciclo de descarga de la batería en el cual sea más conveniente elaborar el pronóstico usando la frecuencia estándar, esto en base a que al acercarse el instante de EoD, la frecuencia de submuestreo aparenta converger a la original, independientemente de la restricción de tiempo máximo de ejecución. Por otra parte, si bien las conclusiones de esta investigación cuentan con una validez limitada por las razones ya expuestas, esto de ninguna forma descarta que, en un futuro cercano, estas sean efectivamente aplicables a una variedad más amplia de casos de estudio, ya sea condiciones de uso de la misma batería, o incluso para resolver problemas de otra índole. Esto justamente abre la oportunidad para trabajos futuros, ya sea validar la estrategia propuesta para modelos más realistas de la batería, distintos perfiles de descarga, e incluso otros sistemas no relacionados pero que también se estudian típicamente dentro de la disciplina de PHM. Finalmente, es esencial poder comprobar eventualmente la efectividad y eficiencia de esta estrategia de pronóstico en tiempo real, pues de ser exitoso dicho experimento, podría resultar en un gran aporte para elaborar pronósticos de falla precisos y poco costosos computacionalmente, que permitan actualizar permanentemente el tiempo de falla estimado, facilitando así la planificación del usuario, evitando altos costos económicos y, lo más importante, resguardando la seguridad de vidas humanas cuando sea el caso.

Bibliografía

[1] L. Biggio and I. Kastanis. Prognostics and health management of industrial assets: Current progress and road ahead. *Frontiers in Artificial Intelligence*, 3, 2020.

[2] F. Calabrese, A. Regattieri, L. Botti, and F. Galizia. Prognostic health management of production systems. new proposed approach and experimental evidences. *Procedia Manufacturing*, 39:260–269, 2019.

[3] M. Kordestani, M. Saif, M. E. Orchard, R. Razavi-Far, and K. Khorasani. Failure prognosis and applications—a survey of recent literature. *IEEE Transactions on Reliability*, 70:728–748, 2021.

[4] I. Shin, J. Lee, J. Y. Lee, K. Jung, D. Kwon, B. Youn, H. Jang, and J. Choi. A framework for prognostics and health management applications toward smart manufacturing systems. *International Journal of Precision Engineering and Manufacturing-Green Technology*, 5:535–554, 2018.

[5] E. Balaban, S. Johnson, and M. Kochenderfer. Unifying system health management and automated decision making. *Journal of Artificial Intelligence Research*, 65:487–518, 2019.

[6] D. E. Acuña and M. E. Orchard. Near-instantaneous battery end-of-discharge prognosis via uncertain event likelihood functions. *ISA Transactions*, 135:199–212, 2023.

[7] C. Díaz, V. Quintero, A. Pérez, F. Jaramillo, C. Burgos-Mellado, H. Rozas, M. E. Orchard, D. Sáez, and R. Cárdenas. Particle-filtering-based prognostics for the state of maximum power available in lithium-ion batteries at electromobility applications. *IEEE Transactions on Vehicular Technology*, 69:7187–7200, 2020.

[8] M. González. *A fast-running failure prognostic algorithm based on a non-homogeneous Markov Chain.* Tesis de magíster en ciencias de la ingeniería, mención eléctrica, Universidad de Chile, 2021.

[9] K. Myers, P. Doran, J. Cook, J. Kotcher, and T. Myers. Consensus revisited: quantifying scientific agreement on climate change and climate expertise among earth scientists 10 years later. *Environmental Research Letters*, 16, 2021.

[10] Our World in Data. Annual co emissions. `https://ourworldindata.org/grapher/`

annual-co2-emissions-per-country?country=~OWID_WRL, 2022.

[11] J. Wang, Q. Wu, J. Liu, H. Yang, M. Yin, S. Chen, P. Guo, J. Ren, X. Luo, W. Linghu, and Q. Huang. Vehicle emission and atmospheric pollution in china: problems, progress, and prospects. *PeerJ*, 7, 2019.

[12] International Energy Agency. Global energy review 2021. https://www.iea.org/reports/global-energy-review-2021, 2021. Consultado en Octubre, 2023.

[13] REN 21. Renewables 2021 global status report. https://www.ren21.net/reports/global-status-report/, 2021. Consultado en Octubre, 2023.

[14] D. Pola, H. Navarrete, M. E. Orchard, R. Rabié, M. Cerda, B. Olivares, J. Silva, P. Espinoza, and A. Pérez. Particle-filtering-based discharge time prognosis for lithium-ion batteries with a statistical characterization of use profiles. *IEEE Transactions on Reliability*, 64:710–720, 2015.

[15] M. Hannan, M. Hoque, A. Hussain, Y. Yusof, and P. Ker. State-of-the-art and energy management system of lithium-ion batteries in electric vehicle applications: Issues and recommendations. *IEEE, Access*, 6:19362–19378, 2018.

[16] The Advanced Rechargeable Lithium Batteries Association. The batteries report 2018, 2018.

[17] F. Maisel, C. Neef, F. Marscheider-Weidemann, and N. Nissen. A forecast on future raw material demand and recycling potential of lithium-ion batteries in electric vehicles. *Resources, Conservation and Recycling*, 192:106920, 2023.

[18] H. Dai, G. Pingjing, W. Xuezhe, S. Zechang, and W. Jiayuan. Anfis (adaptive neuro-fuzzy inference system) based online soc (state of charge) correction considering cell divergence for the ev (electric vehicle) traction batteries. *Energy*, 80:350–360, 2015.

[19] M. E. Orchard, M. Cerda, B. Olivares, and J. Silva. Sequential monte carlo methods for discharge time prognosis in lithium-ion batteries. *International Journal of Prognostics and Health Management*, 3:1–12, 2012.

[20] H. Rozas, D. Troncoso, C. Ley, and M. E. Orchard. Lithium-ion battery state-of-latent-energy (sole): A fresh new look to the problem of energy autonomy prognostics in storage systems. *Journal of Energy Storage*, 40, 2021.

[21] C. Burgos, M. E. Orchard, M. Kazerani, R. Cárdenas, and D. Sáez. Particle-filtering-based estimation of maximum available power state in lithium-ion batteries. *Applied Energy*, 161:349–363, 2016.

[22] M. Jouin, R. Gouriveau, D. Hissel, M. Péra, and N. Zerhouni. Particle filter-based prognostics: Review, discussion and perspectives. *Mechanical Systems and Signal Processing*, 72-73:2–31, 2016.

[23] H. Rozas, F. Jaramillo, A. Pérez, D. Jiménez, M. E. Orchard, and K. Medjaher. A

method for the reduction of the computational cost associated with the implementation of particle-filter-based failure prognostic algorithms. *Mechanical Systems and Signal Processing*, 135, 2020.

[24] D. Troncoso. *A prognostic decision-making approach under uncertainty for an electric vehicle fleet routing problem.* Tesis de magíster en ciencias de la ingeniería, mención eléctrica, Universidad de Chile, 2023.

[25] D. E. Acuña, M. E. Orchard, and P. Wheeler. Computation of time probability distributions for the occurrence of uncertain future events. *Mechanical Systems and Signal Processing*, 150, 2021.

[26] S. Redner. *A Guide to First-Passage Processes.* Cambridge University Press, 2001.

[27] G. E. Gorospe Jr., M. J. Daigle, S. Sankararaman, and C. S. Kulkarni. Gpu accelerated prognostics. In *Annual Conference of the PHM Society*, volume 9, 2017.

[28] S. Särkkä. *Bayesian Filtering and Smoothing.* Cambridge University Press, 2013.

[29] B. Ristic, S. Arulampalam, and N. Gordon. *Beyond the Kalman Filter: Particle Filters for Tracking Applications.* Artech House, 2003.

[30] M. S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp. A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking. *IEEE Transactions on Signal Processing*, 50:174–188, 2002.

[31] M. E. Orchard and G. Vachtsevanos. A particle-filtering approach for on-line fault diagnosis and failure prognosis. *Transactions of the Institute of Measurement and Control*, 31:221–246, 2009.

[32] D. E. Acuña and M. E. Orchard. Particle-filtering-based failure prognosis via sigma-points: Application to lithium-ion battery state-of-charge monitoring. *Mechanical Systems and Signal Processing*, 85:827–848, 2017.

[33] D. E. Acuña and M. E. Orchard. A theoretically rigorous approach to failure prognosis. volume 10, 9 2018.

[34] D. E. Acuña and M. E. Orchard. A gentle correctness verification of the theory of uncertain event prognosis to compute failure time probability. In *Annual Conference of the PHM Society*, volume 14, 2022.

[35] C. Grinstead and J. Snell. *Introduction to Probability.* American Mathematical Society, 2006.

[36] H. F. Navarrete. *Caracterización estadística del perfil de uso de baterías para el pronóstico del estado de carga.* Memoria ingeniería civil eléctrica, Universidad de Chile, 2014.

[37] S. Katoch, S. Singh, and V. Kumar. A review on genetic algorithm: past, present, and

future. *Multimedia Tools and Applications*, 80:8091–8126, 2021.

[38] A. Natekin and A. Knoll. Gradient boosting machines, a tutorial. *Frontiers in Neurorobotics*, 7, 2013.

[39] T. Chen and C. Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, page 785–794. Association for Computing Machinery, 2016.

[40] xgboost. `https://github.com/dmlc/xgboost`, 2021.

[41] Xgboost documentation - xgboost parameters. `https://xgboost.readthedocs.io/en/stable/index.html`, 2022.

[42] C. Burgos, D. Sáez, M. E. Orchard, and R. Cárdenas. Fuzzy modelling for the state-of-charge estimation of lead acid batteries. *Journal of Power Sources*, 274:355–366, 2015.

[43] F. Jaramillo, M. Valderrama, V. Quintero, A. Perez, and M. Orchard. Time-of-failure probability mass function computation using the first-passage-time method applied to particle filter-based prognostics. In *Annual Conference of the PHM Society*, volume 12, 11 2020.

[44] A. Khan and R. Mir. Optimization of constrained function using genetic algorithm. *Computer Engineering and Intelligent Systems*, 8:11–15, 2017.

[45] R. Solgi. geneticalgorithm. `https://github.com/rmsolgi/geneticalgorithm/blob/master/setup.py`, 2020.

[46] T. Osán, D. Bussandri, and P. Lamberti. Monoparametric family of metrics derived from classical jensen–shannon divergence. *Physica A: Statistical Mechanics and its Applications*, 495:336–344, 2018.