Trabajo Fin de Grado
Grado en Ingeniería de Tecnologías Industriales, Mención Eléctrica

Metodología para el modelado de transformadores trifásicos y ensayos sistemáticos de intensidades de energización

Autor: Miguel Castro Molina

Tutor: José Antonio Rosendo Macías

Tutor: Miguel Ángel González Cagigal

Dpto. Ingeniería Eléctrica
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2023

Trabajo Fin de Grado
Grado en Ingeniería de Tecnologías Industriales, Mención Eléctrica

Metodología para el modelado de transformadores trifásicos y ensayos sistemáticos de intensidades de energización

Autor:
Miguel Castro Molina

Tutor:
José Antonio Rosendo Macías
Catedrático de Universidad

Tutor:
Miguel Ángel González Cagigal
Profesor Sustituto Interino

Dpto. Ingeniería Eléctrica
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2023

Trabajo Fin de Grado: Metodología para el modelado de transformadores trifásicos y ensayos sistemáticos de intensidades de energización

Autor: Miguel Castro Molina
Tutor: José Antonio Rosendo Macías
Tutor: Miguel Ángel González Cagigal

El tribunal nombrado para juzgar el trabajo arriba indicado, compuesto por los siguientes profesores:

Presidente:

Vocal/es:

Secretario:

acuerdan otorgarle la calificación de:

El Secretario del Tribunal

Fecha:

Resumen

En el desarrollo de proyectos con empresas del sector eléctrico por parte de la Universidad de Sevilla, ha quedado clara la necesidad de estudiar el comportamiento de transformadores trifásicos durante su reconexión. En este trabajo se ha abordado la tarea de crear una metodología para su modelado y estudio en ensayos sistemáticos, así como el análisis y presentación de los resultados. Para ello, se comenzó con una labor de investigación para determinar qué modelos se usaban principalmente en la literatura y por qué. A continuación, se evaluó la viabilidad de cada uno de ellos. Dicha evaluación requirió de múltiples ensayos en simulaciones, así como de la creación de nuevas herramientas tanto para el análisis de los resultados como para la propia implementación de los modelos. Una vez se seleccionó el modelo, se trató el problema de la estimación de los flujos. Dada la naturaleza matemática del flujo, las condiciones de su cálculo afectan a los resultados. El estudio que se llevó a cabo no solo esclareció la naturaleza íntima del problema, sino que dio una solución para lidiar con el mismo cualesquiera sean las circunstancias. Con el método de estimación establecido, se procedió al diseño del sistema de ejecución de las simulaciones sistemáticas. Esto comprendió la codificación de las condiciones deseadas en el programa de las simulaciones, supliendo sus carencias, y el desarrollo de nuevas herramientas. Tal desarrollo dio como resultado inesperado la creación de un módulo que satisface una necesidad real de los usuarios del programa de simulación empleado. Por último, se perfeccionó el formato de presentación de los resultados para transmitir de manera óptima la información relevante del fenómeno.

El resultado obtenido en la elaboración de este proyecto no solo ha cumplido el objetivo inicial, sino que lo ha sobrepasado al esclarecer y solucionar deficiencias de otros trabajos similares y al crear nuevas herramientas para el análisis de sistemas eléctricos mediante simulaciones.

Abstract

In the course of project development with companies in the electricity sector by the University of Seville, the need to study the behavior of three-phase transformers during their reconnection has become clear. In this study, the task of creating a methodology for their modeling and analysis in systematic trials has been addressed, as well as the interpretation and presentation of the results. To this end, a research effort was initiated to determine which models were mainly used in the literature and why. Then, the feasibility of each of them was evaluated. This assessment required multiple trials in simulations, as well as the creation of new tools for both the analysis of the results and the implementation of the models themselves. Once the model was selected, the issue of flux estimation was tackled. Given the mathematical nature of flux, the conditions of its calculation affect the results. The conducted study not only clarified the intimate nature of the problem but also provided a solution to deal with it, whatever the circumstances may be. With the estimation method established, the execution system's design for systematic trials proceeded. This included encoding the desired conditions into the simulation program, addressing its deficiencies, and developing new tools. Unexpectedly, this development resulted in the creation of a module satisfying a genuine need of the simulation program's users. Lastly, the result presentation format was refined to optimally communicate the phenomenon's relevant information.

The outcome achieved in the development of this project has not only met the initial objective but has exceeded it by clarifying and resolving deficiencies in comparable works and by generating novel tools for electrical system analysis through computational simulations.

Índice

1 Introducción y objetivos

Los transformadores son las máquinas eléctricas responsables de la estructura del sistema eléctrico actual. Su capacidad de transportar potencia alterando la relación entre tensión e intensidad es lo que permite tener diferentes niveles de tensión. Si la producción y el consumo de electricidad estuviese restringido a las tensiones de transporte, los equipos tendrían un coste inasumible. El nivel de aislamiento necesario sería altísimo, motivo del coste elevado, y aun así habría serios riesgos para la salud. Si, por el contrario, fuese el transporte el elemento restringido por las tensiones de generación y consumo, los niveles de pérdidas durante el proceso harían que el sistema no fuese viable a nivel técnico ni económico. Así pues, los transformadores son equipos sumamente importantes para el sistema. Por lo tanto, su entendimiento es una obligación para el correcto diseño y operación de la red eléctrica.

Cuando un transformador se conecta a una red, tiene lugar un transitorio. Las razones de este transitorio se explican con detalle en el Capítulo 5. Por ahora, lo importante es saber que dicho transitorio puede hacer que las intensidades de la conexión, denominadas intensidades de energización, alcancen unos valores muy elevados. El principal problema que presentan estas intensidades son los denominados huecos de tensión. Se denomina como hueco de tensión a la desviación negativa de la tensión de un nudo en un margen tal que afecte al correcto funcionamiento del sistema. En lo relativo a los huecos de tensión causados por las intensidades de energización, los equipos que pueden no funcionar correctamente son los generadores.

Los generadores son máquinas muy complejas en su construcción, por lo que no es de extrañar que su coste vaya en consonancia. A efecto de preservar la integridad de los mismos, sus instalaciones están dotadas de equipos de protección. Una de las funciones de este equipamiento es garantizar que la tensión a la que trabaja el generador está dentro de su rango operativo. Así pues, el problema que plantean los huecos de tensión es la desconexión de los generadores si son lo suficientemente severos.

Una red con el suficiente nivel de mallado y número de generadores puede tener el grado de resiliencia necesario para afrontar estos eventos. Sin embargo, existen redes, denominadas como redes débiles, cuya rigidez de tensiones puede no ser lo suficientemente elevada como para superar estos transitorios. El efecto potencial que tiene la desconexión de generadores es la caída de la red, alcanzando lo que se denomina en inglés como un *blackout*, o apagón en castellano. Una colapso de la red es un problema muy serio porque el proceso de reposición del servicio no es una tarea trivial. Es un proceso que requiere de un alto grado de coordinación y manejo muy preciso de las unidades generadores, especialmente con las primeras, ya que no todas pueden realizar un arranque en negro (en inglés, *black start*).

Este trabajo toma forma para resolver un problema surgido en el departamento de ingeniería eléctrica, concretamente en proyectos de investigación con empresas del sector eléctrico. En un punto, nace la necesidad de evaluar los peores escenarios de intensidades de energización. Si bien era posible encontrar los peores escenarios de cierre de interruptores antes unas condiciones de flujos residuales, no existía la posibilidad de ensayar todos los patrones de flujos remanentes.

Así pues, el objetivo de este trabajo ha sido desarrollar una metodología en ATP para encontrar todos los patrones de flujo y, para cada uno de ellos, estudiar el efecto de todos los instantes de cierre, pudiendo obtener así la intensidad de energización más desfavorable.

2 Simulaciones de transitorios electromagnéticos en ATP

EMTP/ATP (Electromagnetic Transients Program/Alternative Transients Program) es una herramienta computacional empleada en el campo de la ingeniería eléctrica para simular la respuesta de sistemas eléctricos de potencia durante transitorios electromagnéticos. Estos pueden incluir eventos como el accionamiento de interruptores, faltas por pérdida de aislamiento o descargas atmosféricas sobre los conductores de una línea aérea.

En 1964, Hermann Dommel hizo uso del análisis nodal junto con el modelo de parámetros concentrados de una línea de transmisión para simular transitorios electromagnéticos. El modelo implementado hacía uso de la regla de integración trapezoidal. Simultáneamente, Bonneville Power Administration (BPA) había estado desarrollando software para el estudio de las sobretensiones producidas por el accionamiento de interruptores en estudios de coordinación de protecciones. En 1966, BPA le propuso a Hermann Dommel trabajar en el desarrollo del software llamado Electromagnetic Transistens Program (EMTP®). Dicho trabajo tuvo lugar como parte de la creación de herramientas computacionales destinadas al reparto de cargas y estudios de estabilidad. Este proyecto fue dirigido por W. F. Tinney, cuyas contribuciones al problema de las matrices dispersas permitieron que EMTP fuese capaz de simular grandes sistemas de potencia. [2]

En 1973, W. S. Meyer relevó a H. Dommel en el proyecto, acelerando significativamente el progreso. W. S. Meyer colaboró con multitud de investigadores y expertos, culminando en la creación del Development Coordination Group (DGC) de EMTP®. Durante los siguientes años, un gran número de empresas se sumaron a la organización. Algunas de estas empresas son: ABB, AEP, CEA, etc. [2]

ATP nace en los inicios de 1984, cuando Drs. Meyer y Tsu-huei Liu rechazan la propuesta de comercialización de EMTP por parte de BPA [6]. Esencialmente, ATP es una adaptación del código orignal EMTP para ejecutarse en ordenadores personales, los cuales estaban aumentando en potencia y notoriedad. Durante los años, ATP ha tenido un gran número de actualizaciones y mejoras, mayoritariamente promovidas por las necesidades de la comunidad de usuarios. Estas actualizaciones han incluido desde nuevos modelos de componentes hasta algoritmos numéricos mejorados e incluso mejores interfaces gráficas.

A día de hoy, ATP es una de las herramientas de análisis de transitorios electromagnéticos más utilizadas en el estudio de sistemas eléctricos de potencia. La base de usuarios recoge a universidades de todo el mundo, empresas y entidades gubernamentales. Esta base de usuarios ha dado como resultado una gran disponibilidad de información y herramientas complementarias, así como la creación de un foro muy activo en el que buscar ayuda y consejo.

ATPDraw es un preprocesador gráfico. ATP, en esencia, es un programa que se ejecuta en la consola de comandos tomando como entradas diversos archivos. En este respecto, ATPDraw actúa como una capa intermediaria entre el usuario y el *solver*. Este programa permite generar los archivos necesarios para ATP mediante una interfaz gráfica (Figura 2.1), lo cual presenta una seria de ventajas. Por un lado, hace que la interacción con el entorno de simulación sea intuitivo no solo en el montaje del sistema, sino en la interacción con los diferentes componentes. Por otro lado, genera una capa de abstracción entre el usuario y las rutinas que necesita ATP, facilitando su manejo.

Además de las herramientas nativas de ATP para el modelado, tiene la capacidad de interactuar con los módulos TACS (Transient Analysis of Control Systems) y MODELS (un lenguaje de simulaciones), lo cual permite modelar sistemas de control y componentes con características no lineales. TACS es un módulo

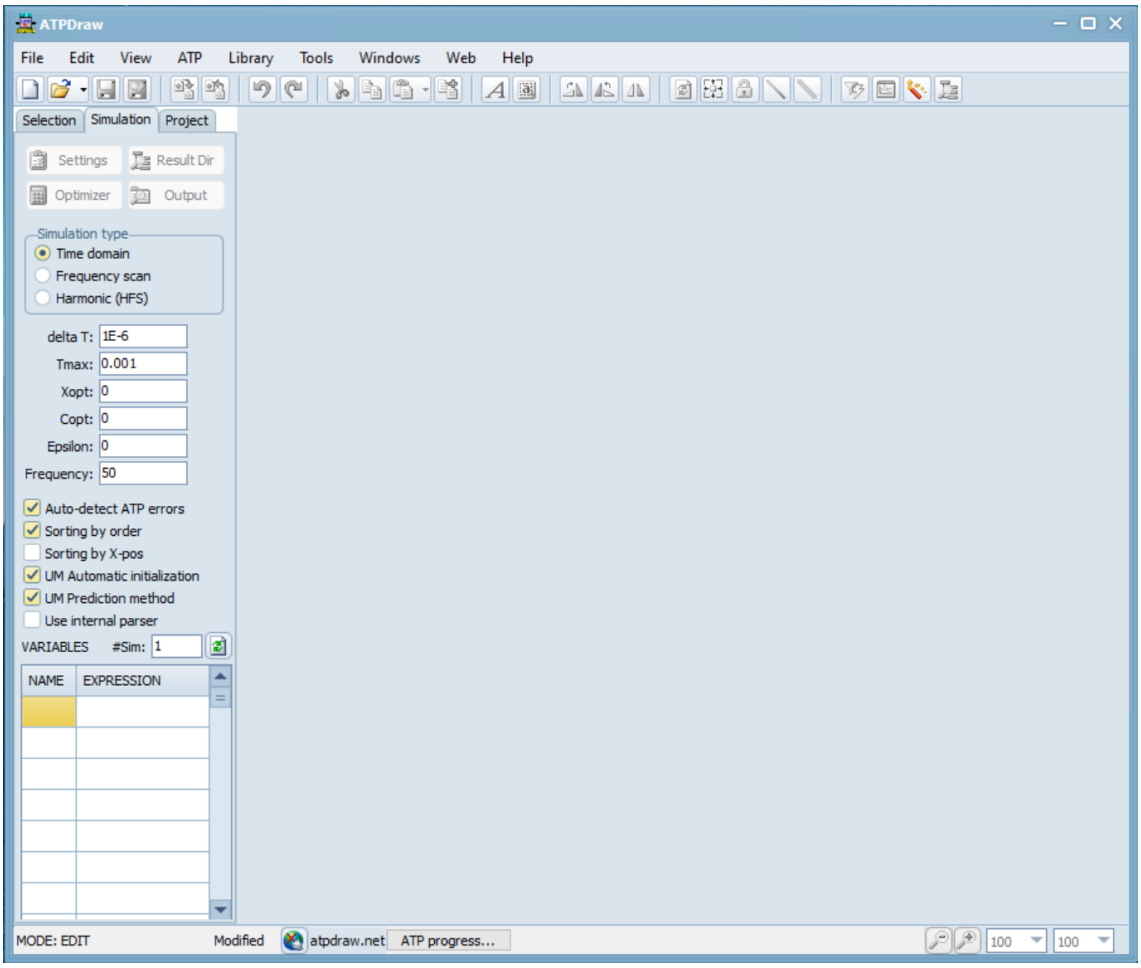


Figura 2.1 Interfaz ATPDraw.

de simulación para el análisis en el tiempo de sistemas de control, inicialmente pensado para convertidores HVDC (High Voltage Direct Current). Su implementación actual en ATP permite simular el control de convertidores HVDC, sistemas de excitación de máquinas síncronas, electrónica de potencia y arcos eléctricos (en interruptores o faltas). MODELS, al contrario que TACS, es un lenguaje descriptivo de uso genérico utilizado para representar sistemas variantes en el tiempo. La descripción de cada modelo atiende a un entorno local, independiente del formato de texto y basado en una sintaxis de palabras clave. Su implementación en ATP permite al usuario definir cualquier tipo de sistema de control o circuito y generar una interfaz sencilla para conectarlo a otros programas/modelos en ATP. Gracias a su naturaleza de uso genérico, MODELS puede usarse tanto para análisis en el dominio del tiempo como en el dominio de la frecuencia. [6]

Las funcionalidades nativas de ATP son extensas, pero, en el análisis de los resultados, se apoya en programas externos (Figura 2.2).

PlotXY es un programa originalmente diseñado para ATP-EMTP. Su diseño tiene como objetivo agilizar todo lo posible la representación de gráficas en el entorno Microsoft Windows. También permite llevar a cabo algunas tareas de post-procesamiento como operaciones algebraicas o el cálculo de los coeficientes de series de Fourier. Permite tener hasta tres archivos de resultados (PL4 o ADF) cargados en memoria simultáneamente para facilitar la comparación de diferentes casos de estudio (Figura 2.3). Tiene la capacidad de representar variables frente al tiempo o entre dos variables internas. Además, posee un sistema de escalado de ejes automático, utilizar dos ejes verticales independientes en la misma gráfica y múltiples herramientas para el escalado y visualización de valores específicos. [6]

ATP Analyzer se utiliza para observar y analizar señales analógicas y canales de información discretos asociados a sistemas de generación, transporte y distribución. El programa tiene la capacidad de leer y mostrar señales analógicas provenientes de ATP como archivos PL4 y COMTRADE, así como información producida por relés de protección y equipos de detección de faltas. [6]

PL42mat es una herramienta que permite la conversión de archivos PL4 a mat, el formato empleado

por Matlab®. Para el procesamiento de los resultados con otro tipo de software, la herramienta GTPPl32 (componente de GTPPLOT) realiza la conversión de los archivos PL4 a DAT. Dado que este formato es de texto plano, su tratamiento es mucho más sencillo.

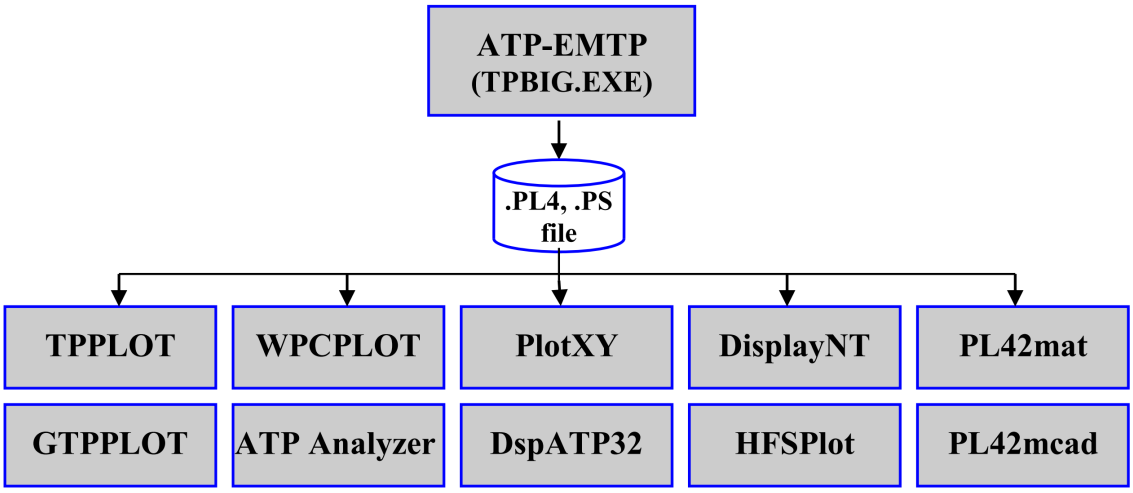


Figura 2.2 Programas de análisis de ATP[6].

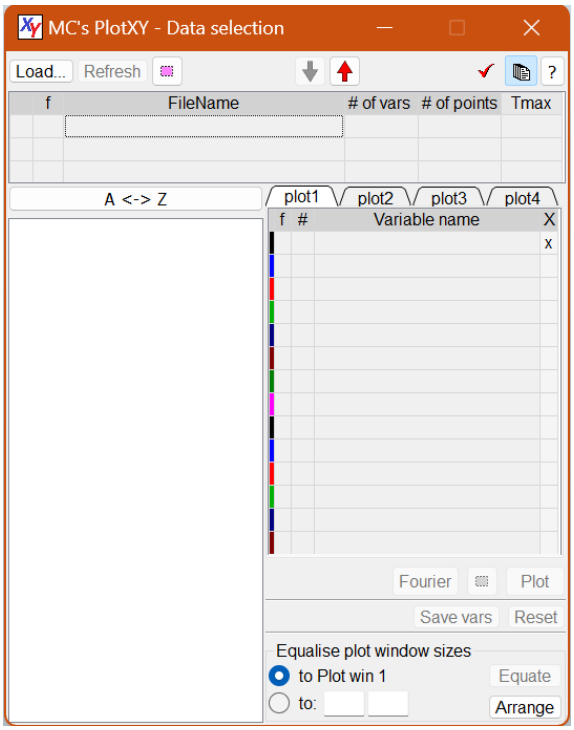


Figura 2.3 Interfaz PlotXY.

3 Modelo del transformador trifásico en ATP

La determinación del modelo a usar para el transformador atiende directamente a la necesidad de caracterizar su ciclo de histéresis y a la capacidad de modificar su flujo remanente. A tal efecto, se investigaron diferentes opciones.

3.1 Primeros acercamientos

En este apartado se exploran las diversas opciones que se estudiaron para modelar el transformador.

3.1.1 Hybrid Transformer - XFMR

En primera instancia, se pensó en utilizar el modelo "Hybrid Transformer - XFMR" (Hybrid Transfomer, del inglés: transformador híbrido. Utiliza diversos enfoques para modelar cada parte de la máquina. Comúnmente denominado como XFMR en el contexto de ATP. XFMR, abreviación en inglés de *transformer*. Similar a la abreviación en castellano de transformador como "trafo"). Este presenta una implementación topológicamente correcta del núcleo, así como la posibilidad de ser ajustado en función de datos de ensayos. Se distinguen tres elementos significativos en la caracterización del núcleo: la relación del enlace de flujo con la intensidad, la pendiente final de dicha curva y el modelo del ciclo de histéresis.

La ecuación de Frolich (3.1) es la base de la optimización del modelo [6]. Haciendo un ajuste con datos de ensayos, proporciona la curva enlaces de flujo - intensidades.

$$\lambda(i) = \frac{i}{a + b \cdot i + c \cdot \sqrt{i}} + L_a \cdot i \tag{3.1}$$

donde:

a, b, c son los parámetros del ajuste,

i es la intensidad,

L_a es la pendiente en saturación.

La pendiente final, L_a, es la inductancia cuando el núcleo alcanza la saturación y pasa a comportarse como un medio de permeabilidad μ_0. Su valor, que es fundamental en el estudio de intensidades de energización, se obtiene con la Ecuación (3.2) [6], la cual requiere de parámetros constructivos del transformador.

$$L_a = \mu_0 \cdot N^2 \cdot \frac{A_{col}}{L_{col}} \tag{3.2}$$

donde:

μ_0 es la permeabilidad del vacío,

N el número de vueltas del arrollamiento interior,

A_{col} es el área de una columna,

L_{col} es la longitud de una columna.

La histéresis se puede definir usando uno de los tres tipos disponibles. Dado que el modelo de reactancia tipo 96 es el único que presenta flujos remanentes tras una desenergización, este es el único viable para la tarea en cuestión.

Al principio solo se buscaba poder introducir cualquier patrón de flujo sin plantearnos si eran coherentes. El motivo por el que inicialmente se descartó su uso es la incapacidad de establecer dichos patrones. Para intentar subsanar esta deficiencia, se exploró la posibilidad de utilizar el modelo XFMR con el núcleo desactivado, modelándolo con un conjunto de inductores no lineales externos.

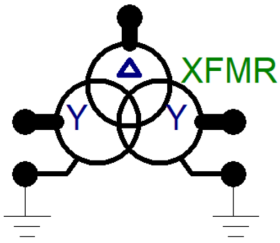


Figura 3.1 Diagrama del modelo XFMR particularizado para el caso de estudio.

3.1.2 XFMR e inductores no lineales externos

Los inductores no lineales requieren del segmento inferior de la curva de histéresis para construir la característica magnética (Código A.1), para lo cual se usaron los parámetros del ajuste realizado por el XFMR. Sin embargo, se vio que esta forma no era viable porque conseguir ternas de flujos remanentes concordantes era extremadamente complicado, no habría acoplamiento entre las fases y la propia implementación de la característica magnética no daba resultados satisfactorios al cambiar de devanado.

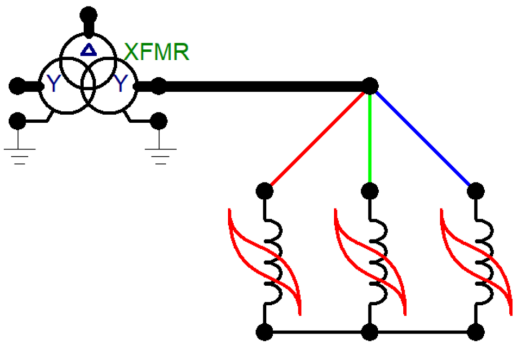


Figura 3.2 Diagrama del modelo XFMR con inductores no lineales externos.

3.2 Modelo adoptado

Finalmente, se eligió el XFMR al encontrar la manera de obtener flujos residuales coherentes entre las fases. El método para ello consiste en la energización del transformador con la consecuente apertura de sus fases en un instante concreto del periodo de la onda de tensión. Dicho instante de apertura determina las condiciones iniciales del transitorio que da lugar a los flujos remanentes.

3.3 Validación del modelo

Una vez seleccionado, se tuvo que verificar la bondad del modelo para poder justificar la veracidad de los resultados. Como ya se ha comentado en el Apartado 3.1.1, el XFMR en sí mismo es muy bueno. Sin embargo,

la reactancia tipo 96 es conocida por tener un comportamiento aceptable, pero no excepcional. Por lo tanto, se juzgó como necesario obtener las curvas de histéresis para ver si eran realistas.

Los ciclos obtenidos (Figuras 3.3, 3.4 y 3.5) difieren considerablemente de los empleados en el ámbito académico. Sin embargo, se confirmó que eran realistas al encontrar curvas de la misma naturaleza obtenidas de ensayos en publicaciones de investigación [10] (Figura 3.6). Las curvas académicas solo consideran una intensidad. En transformadores trifásicos, el flujo de una columna no solo está producido por la intensidad propia, sino por las intensidades adyacentes, las cuales están desfasadas.

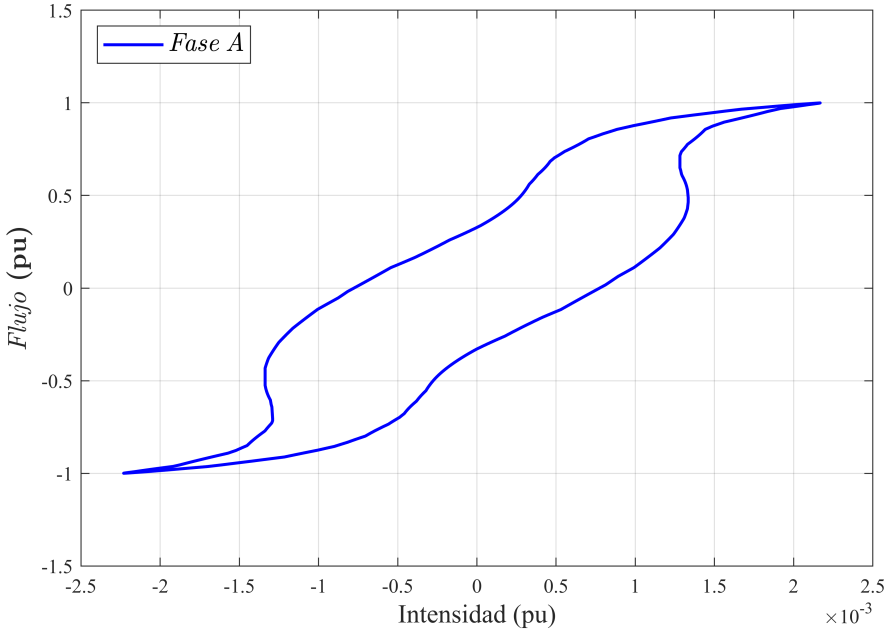


Figura 3.3 Ciclo de histéresis de la fase A.

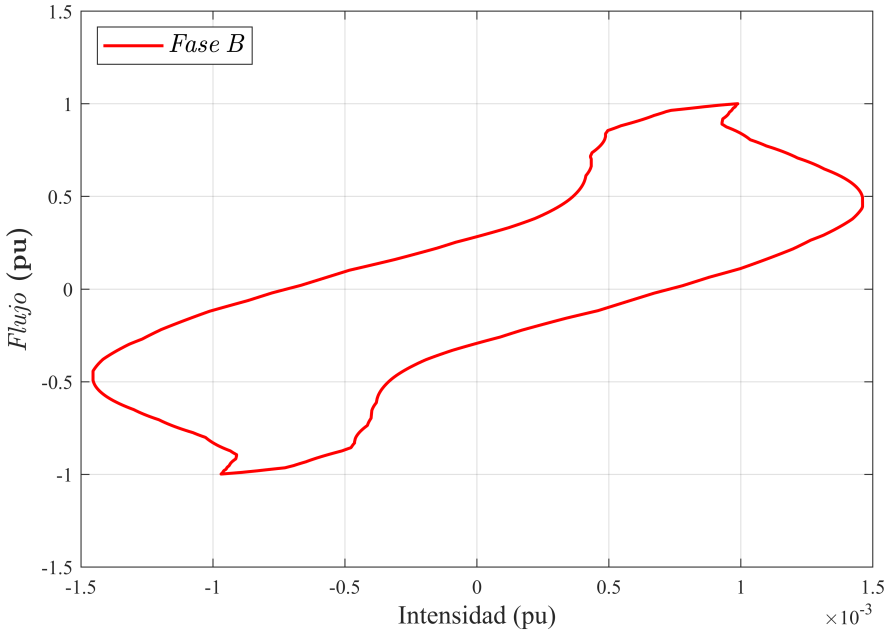


Figura 3.4 Ciclo de histéresis de la fase B.

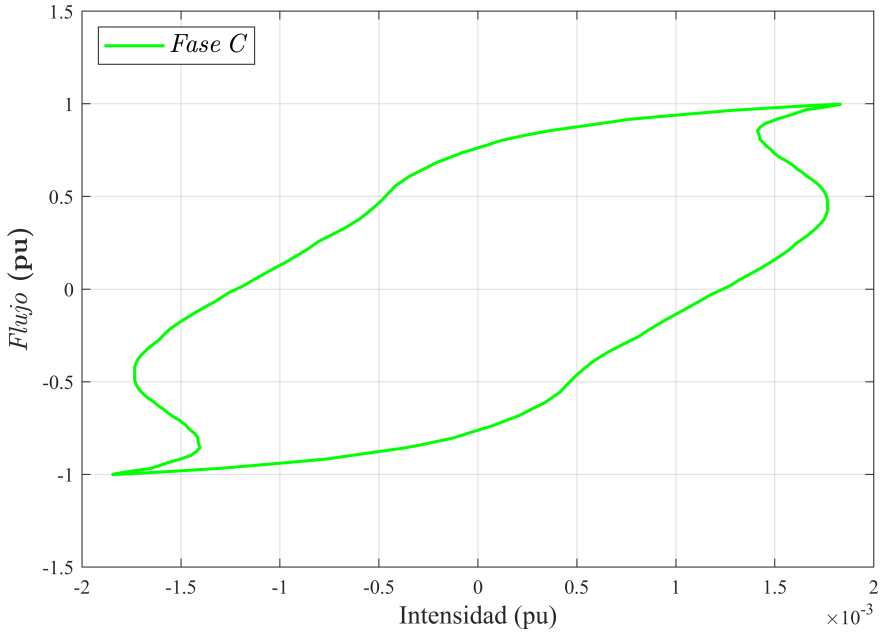


Figura 3.5 Ciclo de histéresis de la fase C.

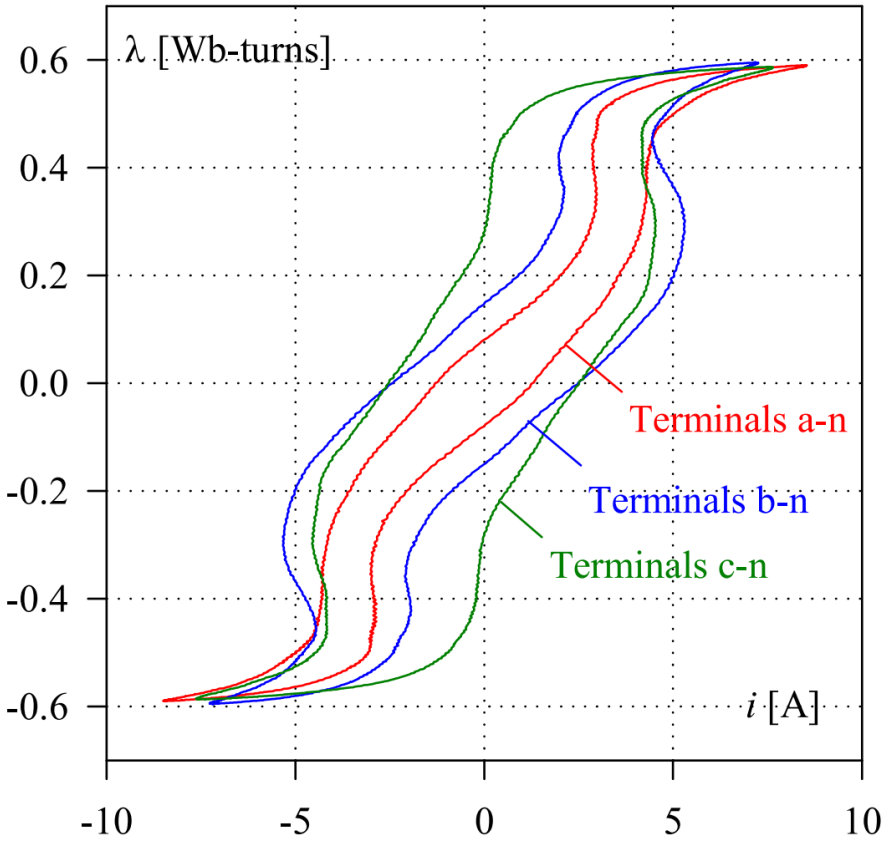


Figura 3.6 Ciclos de histéresis medidos en ensayos [10].

4 Herramientas para el análisis

En este análisis no se dispone de medidas de los flujos. Por lo tanto, es necesario estimarlos. Para ello, se bajaron diversas opciones.

4.1 Problema del flujo

La definición del flujo es la integral de una tensión. En esta sección se aborda qué método y bajo qué condiciones se tiene que realizar dicha integral para obtener una buena estimación de los flujos del núcleo del transformador.

4.1.1 Integración de la tensión en bornes del transformador

El primer método que se usó fue integrar la tensión de cada fase haciendo uso del integrador TACS que implementa ATP (Figura 4.1). De esta forma, se obtienen las estimaciones directamente del archivo de la simulación. Sin embargo, esta manera presenta un serio problema: al comienzo de la integración los flujos contaban con unas condiciones iniciales que no obedecían a ninguna condición física. Esto daba como resultado un *offset* que falseaba la estimación.

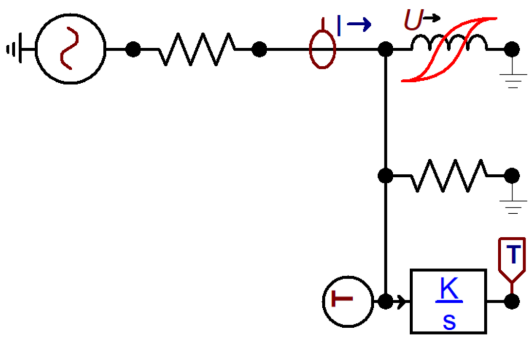


Figura 4.1 Prueba de concepto del integrador TACS.

4.1.2 Integración de la fuerza electromotriz

Se consideró la posibilidad de usar un modelo más preciso para integrar la tensión interna. Se terminó descartando tal idea por la conversión de impedancias entre fases a impedancias de fase (Figura 4.2). En dicha conversión, se obtenían impedancias de valor negativo, las cuales no podían justificarse físicamente. Esto, que de por sí solo puso en entredicho la hipótesis de que este modelo fuese más preciso, se vio reforzado con la escasez de datos para completar la caracterización.

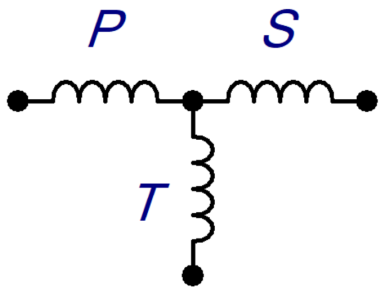


Figura 4.2 Diagrama de impedancias de fase.

4.1.3 Filtrado de la componente de DC

Una componente de continua en un régimen transitorio sinusoidal, per se, no es ni bueno ni malo, simplemente es la respuesta natural del sistema. Sin embargo, la componente de continua que se obtiene al principio de la simulación viene por las condiciones iniciales de la integral, pero dichas condiciones iniciales no están respaldadas por ninguna condición física. Esto hace que sea un artefacto de los cálculos. Por lo tanto, es necesario su filtrado para poder estimar debidamente los flujos mediante la integración de la tensión de cada fase en bornes del transformador.

4.2 Implementación de la estimación del flujo

Lo primero es caracterizar la función del flujo (4.1):

$$\Phi(t) = \int_{t}^{t_c} v(t)\,dt - \Phi_{offset}(t) \cdot \left(1 - e^{\frac{t}{\tau}}\right) \tag{4.1}$$

donde:

$\Phi(t)$ es el flujo,

$v(t)$ es la tensión en bornes del transformador,

$\Phi_{offset}(t)$ es la componente de continua,

τ es la constante de tiempo del decaimiento de flujo.

El modelado de la componente de continua se hace con un decaimiento exponencial, basándonos en evidencias experimentales.

Expresada para el cálculo numérico (Código A.2) mediante el método de Euler (4.2):

$$\Phi_k = \Phi_{k-1} + \frac{v_k + v_{k-1}}{2} - \Phi_{offset,k} \cdot \left(1 - e^{-\frac{\Delta t}{\tau}}\right) \tag{4.2}$$

donde:

Δt es el paso de integración.

La constante de tiempo determina la velocidad de la evolución de la componente de continua. El decaimiento real exhibe una τ en el entorno de 1 s, pero como la componente inicial no tiene base física, se puede modificar el valor de tau al inicio para extinguirla lo antes posibles. Deshaciendo este cambio antes de que el sistema sea sometido a un transitorio, se obtiene la respuesta natural del mismo. Así pues, se logra filtrar el artefacto inicial y modelar el comportamiento natural del sistema con la misma función teniendo que modificar un único parámetro.

El cálculo del *offset* se realiza mediante un algoritmo de ventana móvil (Código A.3), el cual determina en cada paso de integración el valor medio del último periodo de la onda.

5 Ensayos sistemáticos

Cuando un transformador se desconecta de la red, el núcleo permanece imantado en una proporción respecto al estado previo a la desconexión. Debido al mecanismo de este fenómeno, lo más a lo que se puede aspirar en su reconexión es a un óptimo que minimiza la discordancia entre las condiciones de la red y las remanentes del transformador. Dado que ese óptimo del instante de conexión no puede obtener un régimen permanente directo, un transitorio tiene lugar. Para estudiar cómo es el efecto de esta discordancia en la magnitud de dicho régimen transitorio no se puede estudiar solo el caso del óptimo, sino que es necesario barrer todo el espectro de condiciones remanentes e instantes de reconexión. Para poder realizar dicho análisis, es necesario recurrir a ensayos sistemáticos.

Un ensayo sistemático es aquel en el cual se estudia un mismo sistema variando un parámetro en cada simulación con el fin de determinar el impacto de dicha variación en el resultado final.

5.1 Método para el anidamiento de bucles en ATP

Se consideran dos periodos de una onda de tensión de 50 Hz, cada periodo dividido en veinte muestras. Como se explicó en la Sección 3.2, el flujo remanente se obtiene desconectando el transformador; de manera análoga, la condición de la red se obtiene conectando el transformador. Así pues, la combinación de los instantes de desconexión y conexión del transformador respecto al periodo de la onda proporciona un espectro de cuatrocientos escenarios.

ATPDraw permite llevar a cabo esta tarea gracias a un modelo de interruptor que permite ser actuado múltiples veces en una simulación, pero hay un problema. ATP utiliza un traductor que interpreta funciones matemáticas del usuario para que sean usadas en el cálculo de valores que pueda usar el circuito. El funcionamiento de dicho traductor, en este campo de utilización, es equivalente a un bucle que evalúa las funciones especificadas en cada iteración. Las combinaciones se forman por pares de manera que cada instante de apertura tiene que ser simulado con cada instante de cierre. Así pues, el problema es que las condiciones de simulación necesitan de un bucle anidado, pero ATPDraw no soporta tal funcionalidad. No al menos de manera explícita.

Como se ha comentado, el traductor evalúa en cada iteración las funciones especificadas. Además, pone a disposición del usuario una serie de funciones matemáticas, así como una variable especial que simboliza el número de la simulación que se va a ejecutar. Así pues, haciendo uso de dicha variable y de la función que redondea un número real al menor entero más cercano, en este trabajo se ha suplido la carencia codificando todos los escenarios con un único bucle y dos funciones (5.1).

$$\begin{aligned} t_{open} &= t_{ini,open} + 0.001 \cdot \left(\mathrm{KNT} - 20 \cdot \mathrm{FLOOR}\left(\frac{\mathrm{KNT}}{20} \right) \right) \\ t_{close} &= t_{ini,close} + 0.001 \cdot \mathrm{FLOOR}\left(\frac{\mathrm{KNT}}{20} \right) \end{aligned} \tag{5.1}$$

donde:

KNT es la variable del número de la simulación,

FLOOR es la función que redondea un número real al menor entero más cercano.

5.2 Metodología para el análisis

Para poder obtener resultados relevantes, es necesario procesar la información obtenida de las simulaciones, filtrarla y organizarla de manera adecuada. La metodología empleada, que quedará ilustrada en el caso de estudio, tiene como objetivo llevar a cabo estas tareas.

5.2.1 Acondicionamiento de los archivos de resultados

ATPDraw utiliza un tipo de archivo propio para los resultados de las simulaciones en formato binario. Para poder trabajar con los resultados, es necesario procesar estos archivos. Además, se necesita un método sistemático tanto para lidiar con el volumen de resultados por simulación como por el número de simulaciones que pueden tener que llevarse a cabo. A tal efecto, se ha diseñado un programa encargado de extraer dichos archivos, realizar la conversión a archivos de texto plano (.dat), traducir el contenido al formato “csv” preservando la procedencia de cada uno, realizar los cálculos y guardar los resultados. Debido a la magnitud del contenido, el sistema se ha implementado haciendo uso de computación en paralelo con el objetivo de acelerar el proceso (Código A.4).

5.2.2 Evolución temporal de las intensidades y sus valores máximos

Por la naturaleza del estudio, se hace interesante estudiar las intensidades en dos marcos diferentes. El primero atiende a los mecanismos implicados, para lo cual nos centramos en la evolución temporal de uno o dos casos en el periodo de interés. El segundo atiende a la magnitud de las intensidades como función de las condiciones que la propician. Para este caso, estamos interesados en todo el conjunto de simulaciones, pero solo en un valor concreto de cada una, su valor máximo, entendiendo por ello el máximo de los valores absolutos.

La obtención de la evolución temporal es trivial dado que es un resultado director de la simulación. En contra posición, determinar los máximos de las intensidades en cada caso sí que requiere de un cierto procesamiento (Código A.5). Para ello, se emplea un algoritmo de búsqueda del máximo valor. Sabiendo que el máximo tiene que estar contenido en el rango temporal del fenómeno de energización y que este es un máximo global, se puede optimizar el proceso para que sea muy rápido y eficiente.

6 Caso de estudio

Para ilustrar la metodología, se propone un caso de estudio basado en un transformador trifásico alimentado por el lado de baja tensión. Se toma este devanado por ser el de conexión con el generador, el cual es el principal sujeto de estudio en análisis de redes especialmente sensibles a las intensidades de energización.

6.1 Modelo de ATP

El sistema simulado se compone del modelo de transformador XFMR, una fuente ideal de tensión e interruptores de medida y operación ideales.

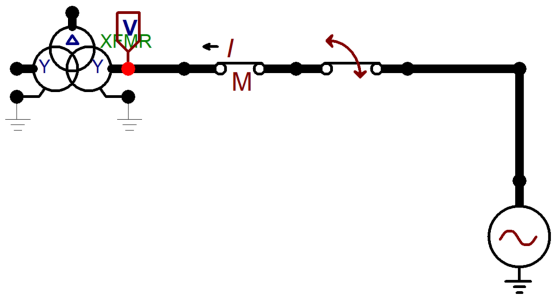


Figura 6.1 Modelo de ATPDraw.

Tabla 6.1 Parámetros del transformador.

	Primario	Secundario	Terciario
Tensión de línea [kV]	230	66	24
Potencia [MVA]	125	125	40
Conexiones	Y	Y	D
Desfase	0	0	330

Tabla 6.2 Parámetros de inductancia y resistencia.

	Impedancia [%]	Potencia [MVA]	Pérdidas [kW]
P-S	13.36	125	299.165
P-T	7.76	40	115.183
S-T	13.68	40	127.336

Tabla 6.3 Parámetros del núcleo.

Tensión [%]	Pérdidas [kW]	Intensidad media [%]
110	42.804	0.0592
110	56.131	0.1007

6.2 Resultados y discusión

En esta sección se abordan los resultados obtenidos del estudio.

6.2.1 Evolución temporal del flujo

A raíz de la importancia del modelado de los flujos, considero apropiado comenzar defendiendo el buen comportamiento del mismo.

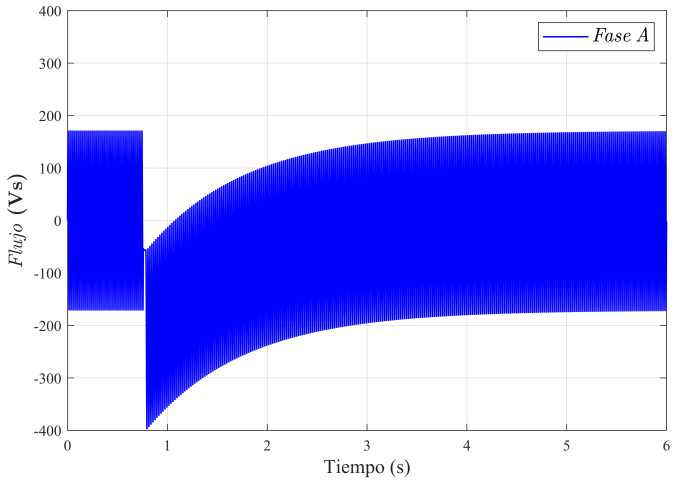
En la Figura 6.2 se diferencian los dos periodos relevantes descritos en la Sección 4.2. La τ inicial se ha elegido como compromiso entre eliminar rápidamente el artefacto y poder apreciar su evolución. Esto implica que, si hubiese restricciones en el tiempo de simulación a fin de incrementar la eficiencia computacional, podría seleccionarse un valor más pequeño.

6.2.2 Fenómenos de alta frecuencia en el flujo

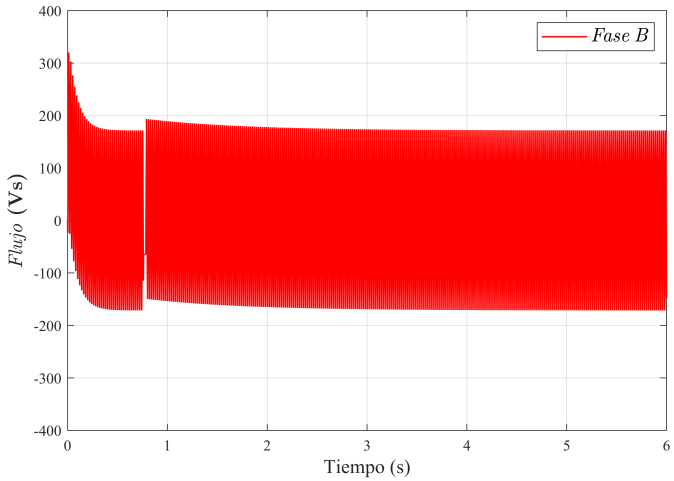
A continuación se exponen dos puntos importantes. El primero es el efecto que tienen los transitorios de alta frecuencia internos del transformador sobre el flujo cuando se realiza la desconexión. El instante exacto de la apertura de la Figura 6.3 es $t = 0.759\,\text{ms}$, mientras que para la Figura 6.4 es $t = 0.76\,\text{ms}$. Las diferencias entre ambas gráficas pone de manifiesto el segundo punto: la sensibilidad del problema a las condiciones iniciales.

6.2.3 Evolución temporal de las intensidades

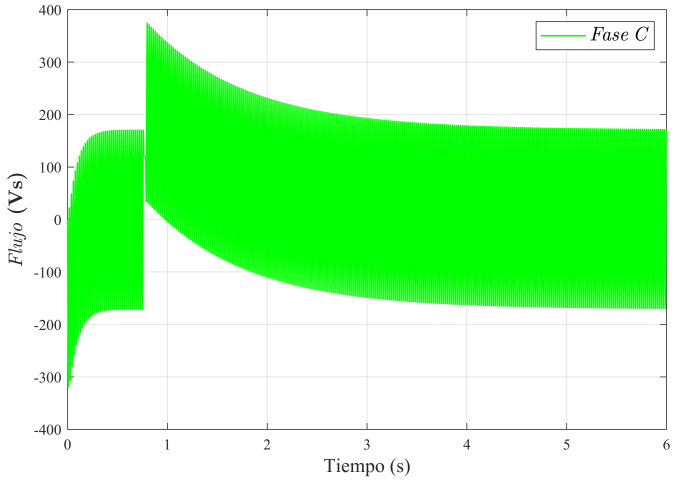
Las intensidades demuestran un comportamiento coherente con el observado en los flujos. Las Figuras 6.5 y 6.6 muestran en detalle la naturaleza de las intensidades durante el fenómeno de energización.



(a) Fase A.



(b) Fase B.



(c) Fase C.

Figura 6.2 Evolución temporal del flujo.

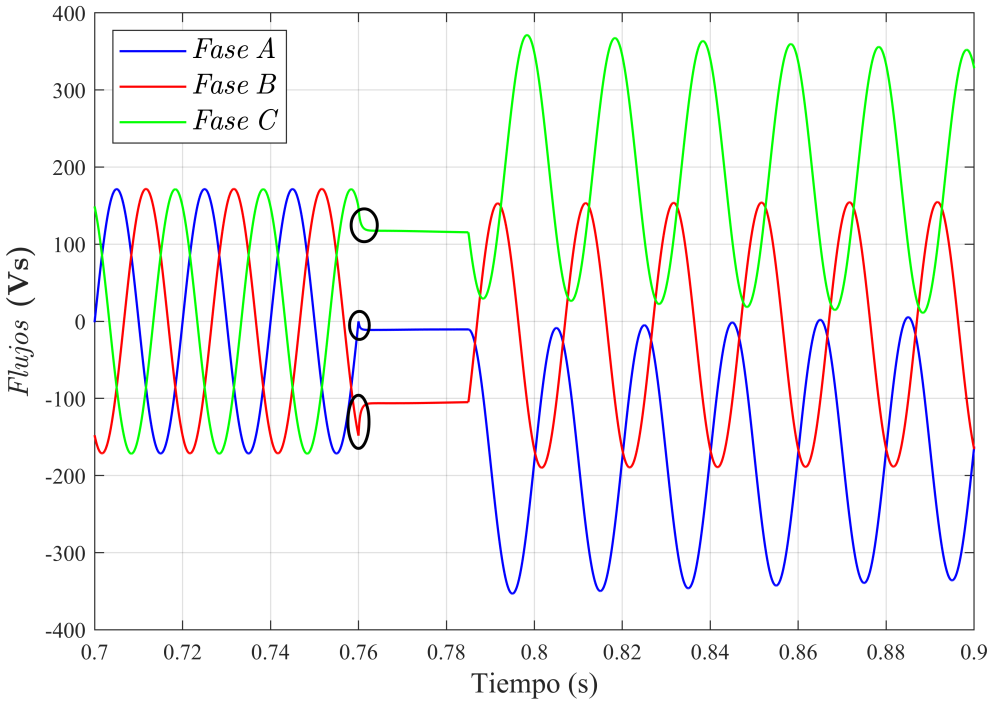


Figura 6.3 Evolución de flujos con desconexión a los 15 ms de un paso por cero. Fenómenos de alta frecuencia señalados.

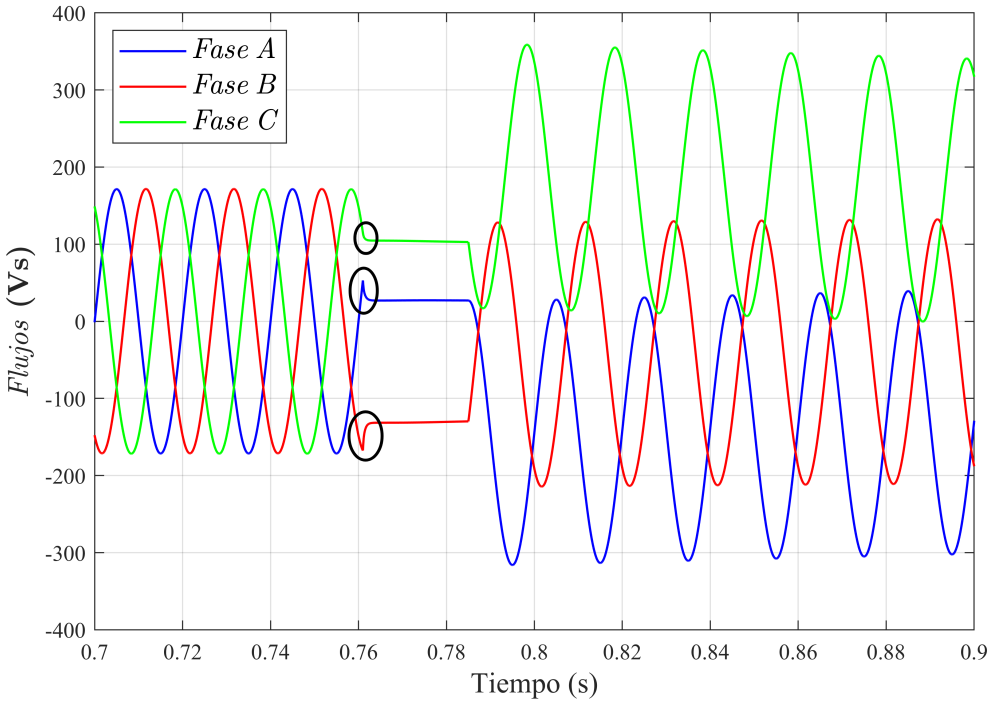


Figura 6.4 Evolución de flujos con desconexión a los 16 ms de un paso por cero. Fenómenos de alta frecuencia señalados.

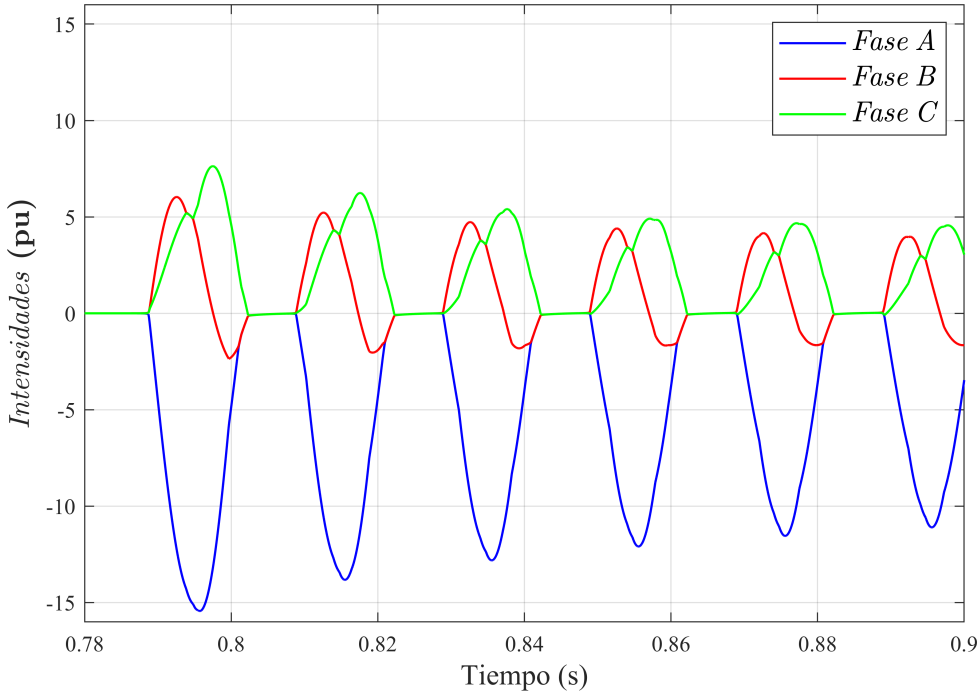


Figura 6.5 Detalle de las intensidades con desconexión a los 11 ms de un paso por cero.

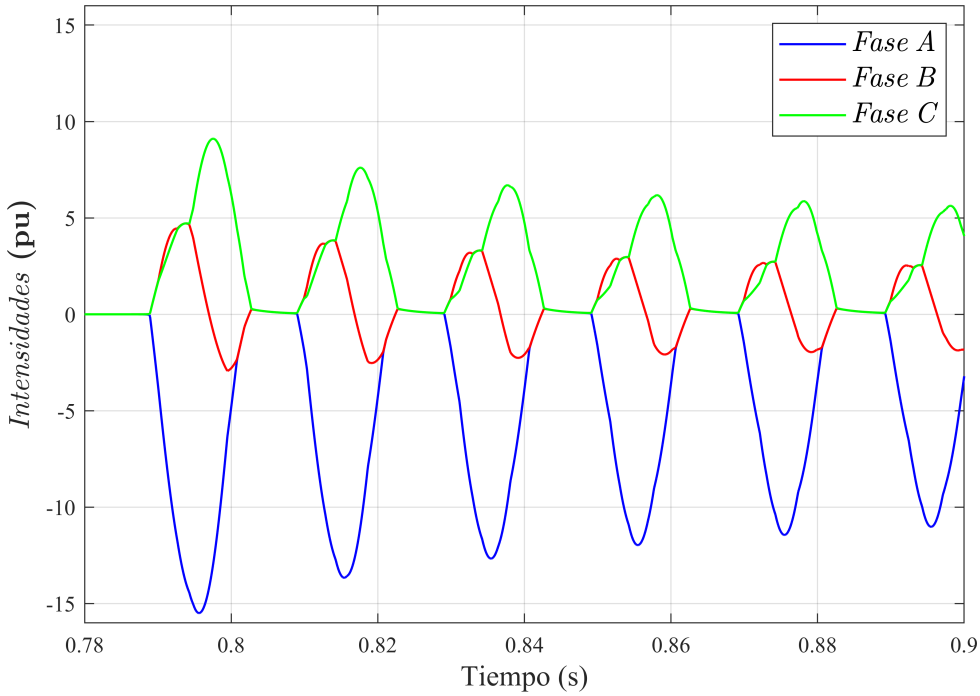


Figura 6.6 Detalle de las intensidades con desconexión a los 12 ms de un paso por cero.

6.2.4 Mapa completo de intensidades máximas

Tal y como se explicó en el Apartado 5.2.2, es necesario barrer todo el espacio de combinaciones de flujos remanentes e instantes de reconexión. El conjunto de Figuras desde 6.7 hasta 6.12 recogen la casuística de cada fase.

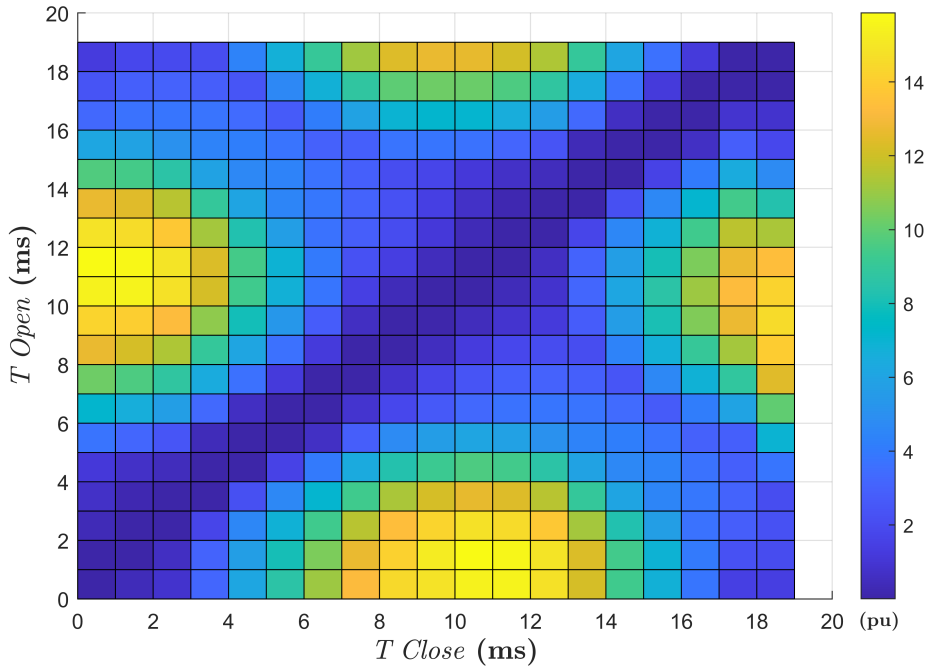


Figura 6.7 Intensidades máximas fase A: perspectiva cenital.

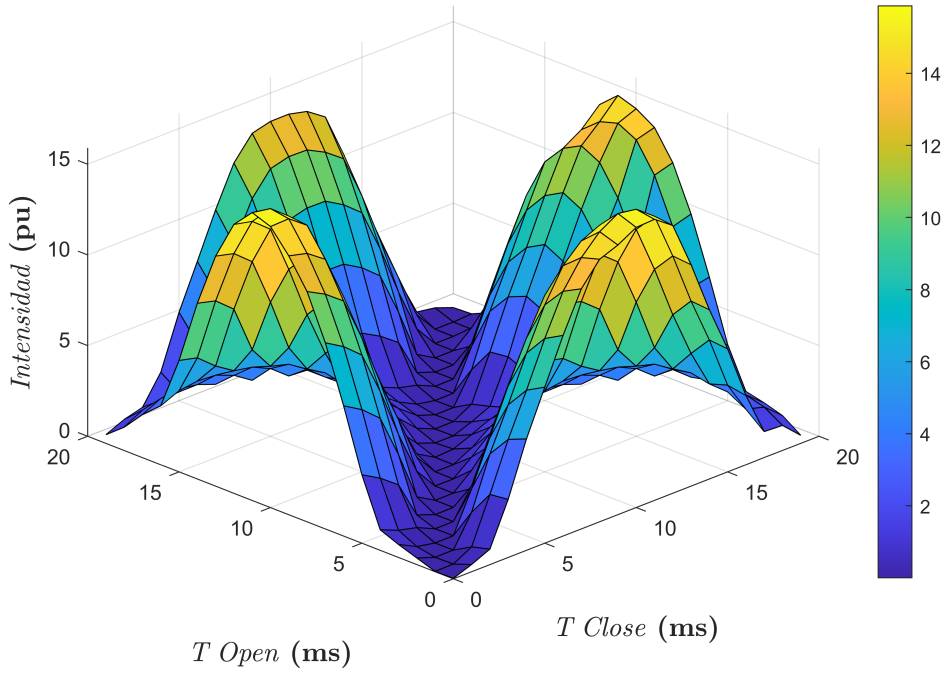


Figura 6.8 Intensidades máximas fase A: perspectiva en ángulo.

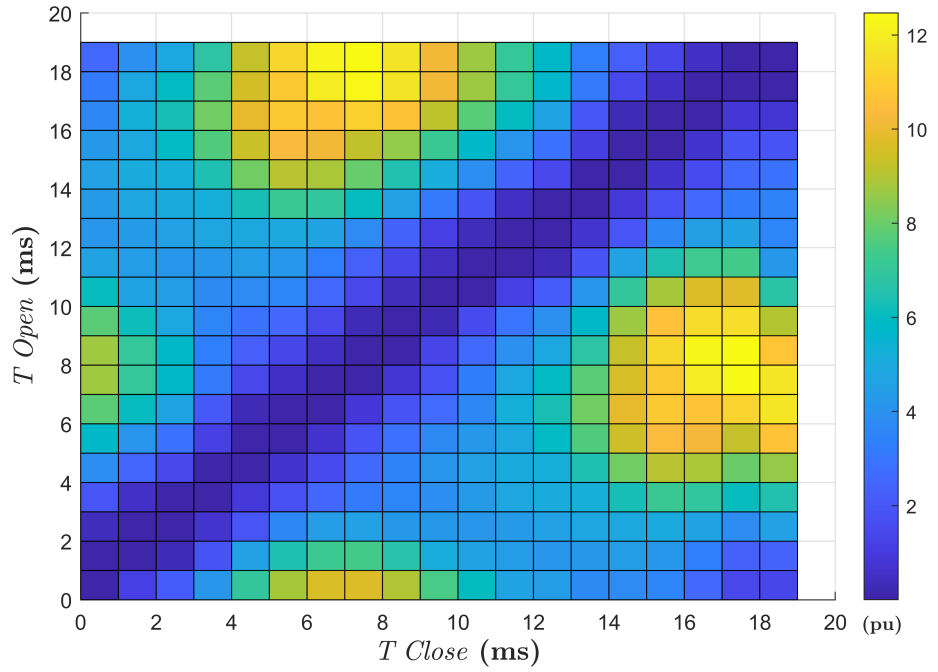


Figura 6.9 Intensidades máximas fase B: perspectiva cenital.

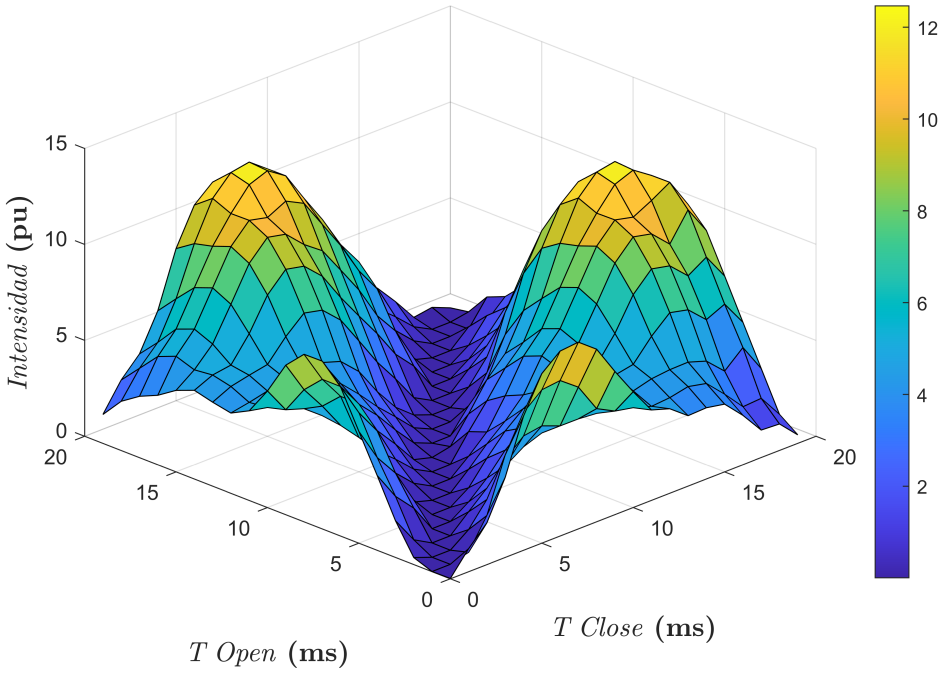


Figura 6.10 Intensidades máximas fase B: perspectiva en ángulo.

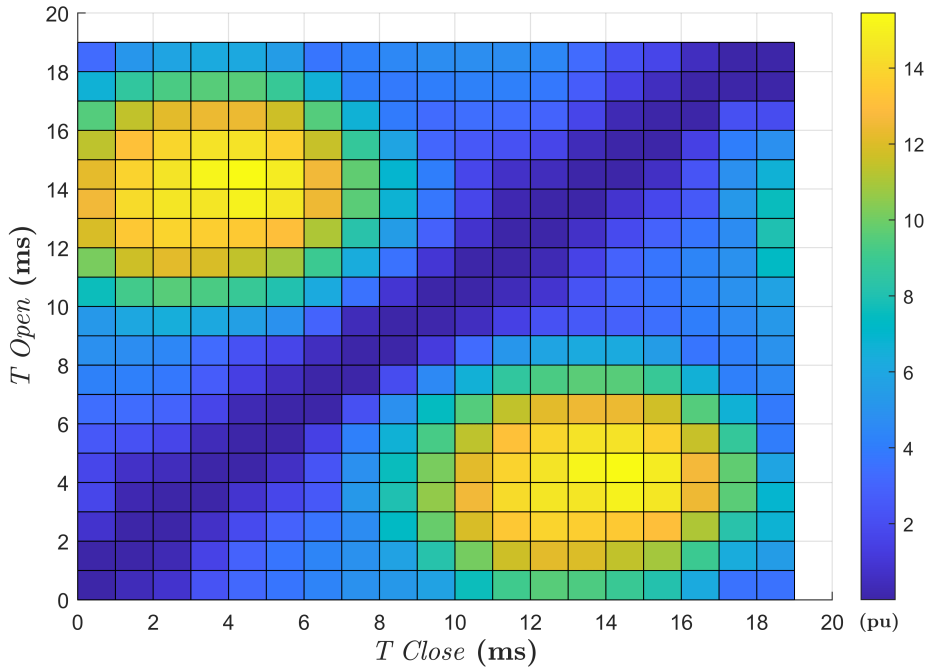


Figura 6.11 Intensidades máximas fase C: perspectiva cenital.

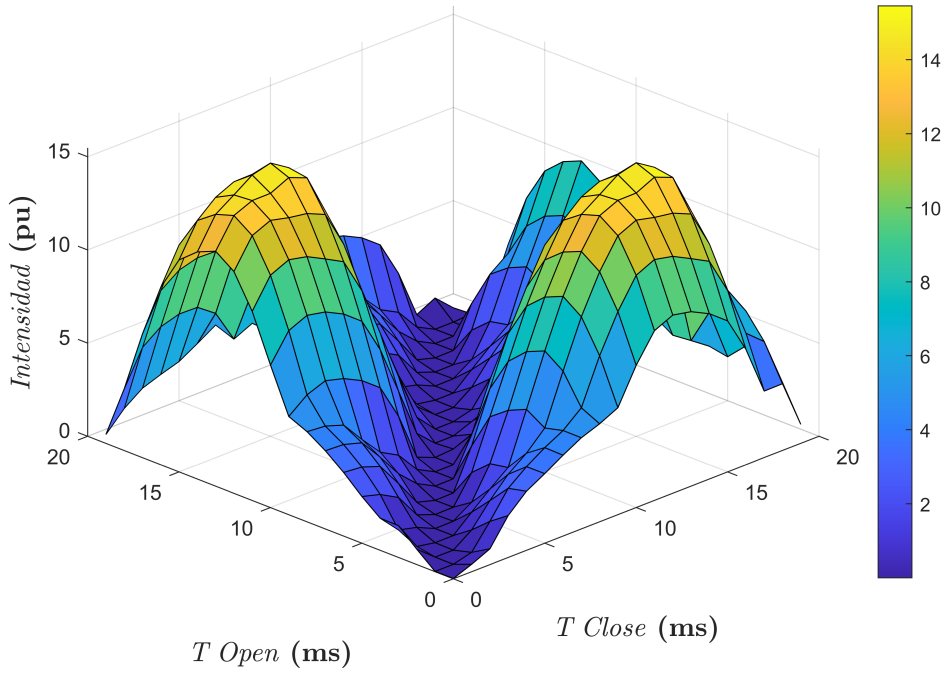


Figura 6.12 Intensidades máximas fase C: perspectiva en ángulo.

Tabla 6.4 Valores máximos de intensidades por fase e instantes de apertura y cierre.

	Fase A	Fase B	Fase C
Intensidad máxima (pu)	15.87	12.47	15.45
Instante de apertura (ms)	12	9	5
Instante de cierre (ms)	1	18	15

El fenómeno que manifiestan estas gráficas de manera más evidente es que la combinación de aperturas y reconexiones en momentos similares del periodo producen las intensidades de energización más pequeñas. Se ve claramente cómo todas las fases exhiben este comportamiento, lo cual podría llevar a la conclusión errónea de que existe una simetría entre ellas. El motivo por el cual no existe simetría es porque la fase B, que es la correspondiente a la columna central, es diferente cuando la relación entre las fases A y C se invierte.

7 Conclusiones y líneas de trabajo futuras

El objetivo del proyecto ha sido crear una metodología que permitiese estudiar toda la casuística de los fenómenos asociados a la energización de transformadores. Para ello, llevé a cabo una labor de investigación para familiarizarme con el fenómeno en cuestión. Le siguió una etapa de análisis en la que evalué los diferentes modelos para el transformador. El proceso de determinar qué modelo usar fue el más fructífero. Me permitió explorar el uso de ATP, herramienta que seguro me será útil en mi labor profesional. De igual manera, pude expandir mis conocimientos de programación al abordar un nuevo lenguaje desde cero y hacer uso de técnicas que no había utilizado hasta la fecha. Por último, la presentación de los resultados me hizo ver la importancia del cómo exponerlos. Los datos contienen mucha información, pero está en la mano de la persona recogerlos de manera que se pueda apreciar.

La finalización del proyecto ha dejado abierta la puerta a futuras líneas de trabajo. La inclusión de elementos de la red real en la que trabaja el transformador aumentaría el valor de los resultados obtenidos. Algunos, como interruptores, requerirían de un estudio más en profundidad para modelar adecuadamente los fenómenos asociados a sus transitorios. En lo referente al modelado del transformador, se podría explorar el uso de modelos del ciclo de histéresis más precisos, como el Jiles-Atherton. Por último, en la naturaleza sistemática de las simulaciones, se podrían tener en cuenta el estado de otros elementos de la red. Variables como el nivel de carga de otros equipos o el número de generadores en acoplamiento pueden tener un impacto notable.

El resultado de todo este trabajo ha sido la creación de una metodología que permite el estudio de las intensidades de energización de transformadores trifásicos, la apertura de vías para expandirla a escenarios más complejos y el desarrollo de herramientas de análisis y procesamiento para ATP.

Apéndice A
Código

Python fue el lenguaje de programación utilizado al principio del proyecto. Sin embargo, más adelante se pasó a utilizar Rust. El principal beneficio obtenido gracias al cambio fue un menor tiempo de computación. Algunos de los procesos pasaron de tardar cuarenta minutos en Python a dieciocho minutos en Rust. Posteriores mejoras en algunos de los algoritmos y el empleo de computación en paralelo incrementaron el rendimiento en escenarios de mayor carga.

A.1 Modelo del transformador

En esta sección se muestra el código empleado para la obtención del segmento inferior de la curva de histéresis, necesaria para los inductores no lineales externos.

Código A.1 Cálculo del segmento inferior de la curva de histéresis para inductores no lineales externos.

```
import numpy as np
from numpy import array as array
import matplotlib.pyplot as plt

def frolich(i,La,a,b,c):
    return i/(a + b*i + c*np.sqrt(i)) + La*i

def guarda_datos(nombre_archivo,data) -> None:
    with open(nombre_archivo, 'w') as file:
        file.write("C   Curr [A] -->  Flux [V.s] -->\n")
        for x, y in data:
            if y != None:
                file.write("{:>14.4f} {:>10.4f}\n".format(x, y))
            else:
                file.write("{:>14.4f}\n".format(x))
        file.write("{:>14.4f}\n".format(9999.))

intensidad_base_3 = 1.482139 # A

intensidades_pu = np.array([0,0.01,0.09,0.15,0.25,0.35,0.45,0.6,0.7,1,1.3,1.5,1.6,1.8],
 dtype=np.float64)
intensidades_A = intensidad_base_3 * intensidades_pu

La = 0.68714e-3
a = 0.0000092223702626
b = 0.029618961167
c = 0.0092498886616

flujos = np.array(frolich(intensidades_A,La,a,b,c),dtype=np.float64)
```

```
        inte_neg = intensidades_A * -1
        flujos_neg = flujos * -1

        inte_neg_rev = inte_neg[::-1]
        flujos_neg_rev = flujos_neg[::-1]

        inte_neg_rev_s0 = inte_neg_rev[:-1:1]
        inte_comp = np.append(inte_neg_rev_s0,intensidades_A)

        flujos_neg_rev_s0 = flujos_neg_rev[:-1:1]
        flujos_comp = np.append(flujos_neg_rev_s0,flujos)

        ancho_hist = 2 # A
        inte_comp_desp = array([(i+ancho_hist/2) for i in inte_comp])

        mag_x = inte_comp_desp
        mag_y = flujos_comp

        puntos = np.column_stack((mag_x,mag_y))
        guarda_datos('Saturacion_terc.txt',puntos)


        plt.style.use('_mpl-gallery')

        fig, ax = plt.subplots(ncols=1,nrows=1,sharex=True,sharey=True,figsize=(13, 9), dpi=120)
        fig.canvas.manager.set_window_title('Curva saturación terciario')

        mngr = plt.get_current_fig_manager()
        mngr.window.geometry("+700+100")

        puntos_frolich = ax.scatter(x=mag_x, y=mag_y, s=10, c='b', label="Puntos Frolich")

        ax.set_title('Curvas Frolich')
        ax.legend(handles=[puntos_frolich],loc='upper left')
        ax.set_xlabel('Current')
        ax.set_ylabel('Flux')

        plt.subplots_adjust(wspace=0, top=0.9, bottom=0.1, left=0.1, right=0.9)

        plt.show()
```

A.2 Herramientas de análisis

A partir de este punto, todo el código se desarrolló utilizando Rust. El código de esta sección recoge la implementación del filtrado y modelado de las componentes de continua exponenciales decrecientes.

Código A.2 Proceso de integración.

```
    pub fn decaimiento_de_flujo(
        df: &DataFrame,
        tension: &str,
        delta_t: &f64,
        tau1: &f64,
        tau2: &f64,
        t_cambio_tau: &f64,
    ) -> Result<Vec<f64>, Box<dyn std::error::Error>> {
        let _tension_iter = df_column_f64_into_iter(&df, tension).unwrap();
        let tension_vec = _tension_iter.map(|s| s.unwrap() as f64).collect::<Vec<f64>>();
        let mut tension_iter = tension_vec.iter().cloned();

        let _tiempo_iter = df_column_f64_into_iter(&df, "Tiempo").unwrap();
        let tiempo_vec = _tiempo_iter.map(|s| s.unwrap() as f64).collect::<Vec<f64>>();
        let mut tiempo_iter = tiempo_vec.iter().cloned();

        let num_muestras_20_ms = (20_f64 * 10_f64.powf(-3_f64) / delta_t).round();

        let mut vector_muestras: VecDeque<f64> = VecDeque::new();
        let mut vector_resultado_flujos: Vec<f64> = Vec::new();

        let mut tension_k_prev: Option<f64> = None;
        let mut flujo = 0_f64;
        let mut offset = 0_f64;
        let mut total_sum = 0_f64;
        let mut cc = 0_f64;

        let mut vector_offset: Vec<f64> = Vec::new();
        let mut vector_cc: Vec<f64> = Vec::new();

        let delta_t = *delta_t;
        let tau1 = *tau1;
        let tau2 = *tau2;
        let t_cambio_tau = *t_cambio_tau;

        while let (Some(tension_k), Some(tiempo_k)) = (tension_iter.next(), tiempo_iter.next
    ()) {
            if let Some(tension_k_prev) = tension_k_prev {
                if vector_muestras.len() < num_muestras_20_ms as usize {
                vector_muestras.push_back(flujo);
                total_sum += flujo;
                } else {
                    offset = calcula_offset(&mut vector_muestras, &flujo, &mut total_sum, &
    num_muestras_20_ms);
                    let exponente: f64;
                    if tiempo_k < t_cambio_tau {
                        exponente = - delta_t / tau1;
                    } else {
                        exponente = - delta_t / tau2;
                    }
                    cc = offset * (1_f64 - f64::exp(exponente));
                }

                flujo = flujo - cc + (tension_k + tension_k_prev)/2_f64 * delta_t;
```

```
            }

            vector_cc.push(cc);
            vector_offset.push(offset);
            vector_resultado_flujos.push(flujo);
            tension_k_prev = Some(tension_k);
        }

        Ok(vector_resultado_flujos)
    }
```

Código A.3 Cálculo del *offset* de la componente de continua.

```
    fn calcula_offset(
        muestras_vec: &mut VecDeque<f64>,
        new_value: &f64,
        sum: &mut f64,
        num_muestras_20_ms: &f64,
    ) -> f64 {
        let oldest_sample = muestras_vec.pop_front().unwrap();
        *sum -= oldest_sample;

        muestras_vec.push_back(*new_value);
        *sum += *new_value;

        let offset = *sum / *num_muestras_20_ms;

        offset
    }
```

A.3 Ensayos sistemáticos

Para los ensayos sistemáticos se tuvo que modificar la filosofía del código. Ahora, todo el proceso de procesamiento y cálculo se realiza utilizando computación en paralelo. Uno de los mayores beneficios que se obtiene de esto es la velocidad de la conversión de los archivos de resultados de las simulaciones. No solo en tanto que el proceso dura menos tiempo, sino que se encarga de estructurarlo en las carpetas correspondientes, garantizando un flujo de trabajo ordenado.

Código A.4 Procesamiento de los archivos de ATPDraw.

```
    use funcs_tfg::*;
    mod utilities;
    use funcs_tfg::func_extern::remove_prefix_for_glob;
    use funcs_tfg::func_polars::clear_dat_file;
    use polars::prelude::*;
    use utilities::*;

    use std::sync::Arc;
    use std::thread;
    use std::{time::Instant, path::PathBuf};
    use std::fs::{self, File};
    use std::io::{ErrorKind, Error};

    use regex::Regex;
    use glob::glob;
    use structopt::StructOpt;
```

```
#[derive(StructOpt, Debug)]
struct Opt {
    /// Procesa archivos pl4 para crear nuevos archivos csv. Necesario al simular de
nuevo
    #[structopt(short, long)]
    pl4: bool,

    /// Número deseado de archivos por hilo
    #[structopt(short = "c", long, default_value = "4")]
    chunk_size: usize,

}



fn main() -> std::result::Result<(), Box<dyn std::error::Error>> {
    let start_time = Instant::now();
    let opt = Opt::from_args();

    let dir_results = PathBuf::from(r"C:\EMTP\atpdraw\results");

    let dir_reactor = PathBuf::from(r"C:\EMTP\atpdraw\reactor");

    let bat_file = dir_reactor.join("script_archivos.bat");

    let exe_path = PathBuf::from(r"C:\EMTP\atpdraw\reactor\GTPPL32.EXE");


    let chunk_size = opt.chunk_size;


    //--------------------------------------------------------------------------
    // region: ACONDICIONAMIENTO DE LOS ARCHIVOS RESULTADO DE ATP
    //--------------------------------------------------------------------------
    if opt.pl4 {
        println!();
        println!();
        println!();
        println!("Procesando archivos pl4");
        println!("Número de archivos por hilo aprox: {}", opt.chunk_size);
        println!();

        // 1. LLEVAR LOS ARCHIVOS AL REACTOR
        let number_regex = Regex::new(r"^\d+$").unwrap();
        let entries_in_dir_resultados = fs::read_dir(dir_results).unwrap();

        let mut vec_pl4_files_reactor: Vec<PathBuf> = Vec::new();

        for entry in entries_in_dir_resultados.filter_map(|s|s.ok()).into_iter() {
            if entry.path().is_dir() {
                let entry_name = entry.file_name().to_string_lossy().to_string();
                if number_regex.is_match(&entry_name) {
                    let pl4_uncleaned_pattern = entry.path().join("*.pl4");
                    let pl4_cleared_pattern = remove_prefix_for_glob(&
pl4_uncleaned_pattern);

                    for pl4_file in glob(pl4_cleared_pattern.to_str().unwrap()).map_err
(|e| Error::new(ErrorKind::InvalidInput, e))? {
                        if let Ok(pl4_file_path) = pl4_file {
                            if let Some(pl4_file_name_os_str) = pl4_file_path.file_stem
() {
```

```
                                    if let Some(pl4_file_name) = pl4_file_name_os_str.to_str
        () {
                                        let new_pl4_name = [pl4_file_name.to_owned(), ent
        ry_name.clone()].join("_") + ".pl4";
                                        let pl4_file_reactor = dir_reactor.join(new_pl4_name
        );

                                        fs::copy(&pl4_file_path, &pl4_file_reactor)?;
                                        vec_pl4_files_reactor.push(pl4_file_reactor);
                                    }
                                }
                            }
                        }

                    }
                }
            }


            let mut threads_paths = Vec::new();

            while !vec_pl4_files_reactor.is_empty() {
                let chunk: Vec<PathBuf> = vec_pl4_files_reactor.drain(..std::cmp::min(
        chunk_size, vec_pl4_files_reactor.len())).collect();
                threads_paths.push(chunk);
            }

            // El reactor base es la plantilla para los reactores específicos de cada hilo
            let exe_path_to_copy = Arc::new(PathBuf::from(r"C:\EMTP\atpdraw\reactor_MT\
        reactor_base\GTPPL32.EXE"));
            // En este directorio tengo que crear los reactores específicos de cada hilo
            let dir_reactor_mt = Arc::new(PathBuf::from(r"C:\EMTP\atpdraw\reactor_MT"));
            // De este directorio tengo que sacar los archivos .pl4
            let dir_reactor_og = Arc::new(PathBuf::from(r"C:\EMTP\atpdraw\reactor"));

            let dir_dat_procesados = Arc::new(PathBuf::from(r"C:\EMTP\atpdraw\Resultados p
        rocesados\Simulaciones_Flujos_rev\Archivos dat"));

            let dir_csv_procesados = Arc::new(PathBuf::from(r"C:\EMTP\atpdraw\Resultados p
        rocesados\Simulaciones_Flujos_rev\Archivos csv"));

            let mut handles = Vec::new();
            for (thread_id, paths_for_thread) in threads_paths.into_iter().enumerate() {

                let shared_exe_path_to_copy = Arc::clone(&exe_path_to_copy);
                let shared_dir_reactor_mt = Arc::clone(&dir_reactor_mt);
                let shared_dir_reactor_og = Arc::clone(&dir_reactor_og);
                let shared_dir_dat_procesados = Arc::clone(&dir_dat_procesados);
                let shared_dir_csv_procesados = Arc::clone(&dir_csv_procesados);

                let reactor_thread_name = format!("{}{}", "reactor", thread_id.to_string());
                let dir_reactor_thread = shared_dir_reactor_mt.join(reactor_thread_name);
                let exe_path_thread = dir_reactor_thread.join("GTPPL32.EXE");
                let pl4_pattern = dir_reactor_thread.join("*.pl4");
                let bat_file_path = dir_reactor_thread.join("script_archivos.bat");


                let handle = thread::spawn(move || {
                    // Creamos el reactor del thread
                    fs::create_dir_all(&dir_reactor_thread).unwrap();
                    // Copiamos el ejecutable
                    fs::copy(&(*shared_exe_path_to_copy), &exe_path_thread).unwrap();
                    // Copiamos los archivos pertinentes al reactor
```

```
                for pl4_file_og_path in paths_for_thread {
                    // Obtenemos el nombre del archivo
                    let pl4_file_name = pl4_file_og_path.file_name().unwrap().to_st
ring_lossy().to_string();
                    // Creamos la ruta de destino
                    let new_pl4_file_path = dir_reactor_thread.join(pl4_file_name);
                    // Copiamos los archivos
                    fs::copy(&pl4_file_og_path, &new_pl4_file_path).unwrap();
                }

                // Creamos el archivo bat
                func_extern::write_bat_file(&bat_file_path, &pl4_pattern).unwrap();
                // Hacemos la conversion
                func_extern::call_gtppl32(&exe_path_thread).unwrap();
                // Quitamos los pl4
                let pl4_pattern_cleaned = remove_prefix_for_glob(&pl4_pattern);
                for pl4_file in glob(pl4_pattern_cleaned.to_str().unwrap()).map_err(|e|
Error::new(ErrorKind::InvalidInput, e)).unwrap() {
                    if let Ok(pl4_file_in_reactor) = pl4_file {
                        fs::remove_file(&pl4_file_in_reactor).unwrap();
                    }
                }

                let dat_pattern_uncleaned = dir_reactor_thread.join("*.dat");
                let dat_pattern_cleaned = remove_prefix_for_glob(&dat_pattern_uncleaned)
;


                let dat_pattern_str = dat_pattern_cleaned.to_str()
                    .expect("Failed to convert dat_pattern_cleaned to a string");

                let mut dat_file_path_in_resultados = Vec::new();

                match glob(dat_pattern_str) {
                    Err(e) => eprintln!("Error while globbing: {}", e),
                    Ok(files) => {
                        for dat_file in files {
                            match dat_file {
                                Err(e) => eprintln!("Error while processing file: {}", e
),
                                Ok(dat_file_path) => {
                                    match dat_file_path.file_name() {
                                        None => eprintln!("Error: Path has no file name"
),
                                        Some(dat_file_name_os_str) => {
                                            match dat_file_name_os_str.to_str() {
                                                None => eprintln!("Error: Failed to
convert file name to string"),
                                                Some(dat_file_name) => {
                                                    let dat_file_proyecto = shared_di
r_dat_procesados.join(dat_file_name);
                                                    if let Err(e) = fs::copy(&
dat_file_path, &dat_file_proyecto) {
                                                        eprintln!(
                                                            "Failed to copy file from:\n
'{}'\nto:\n'{}'\nError: {}\n",
                                                            dat_file_path.display(),
                                                            dat_file_proyecto.display(),
                                                            e
                                                        );
                                                    }
```

```
                                                    dat_file_path_in_resultados.push(
        dat_file_proyecto);

                                                    if let Err(e) = fs::remove_file(&
        dat_file_path) {
                                                        eprintln!(
                                                            "Failed to remove file '{}':
         {}",
                                                            dat_file_path.display(),
                                                            e
                                                        );
                                                    }

                                                }
                                            }
                                        }
                                    }
                                }
                            }
                        }
                    };

                    /*
                    Creamos los archivos csv => Estos son los que se usan para los cálculos.
                    */
                    for dat_file in dat_file_path_in_resultados {
                        let dat_file_stem = dat_file.file_stem().unwrap().to_string_lossy().
        to_string();
                        let csv_file_name = dat_file_stem + ".csv";
                        let csv_file = shared_dir_csv_procesados.join(csv_file_name);

                        // Limpiamos el dat file y lo convertimos en un csv
                        clear_dat_file(&dat_file, &csv_file).unwrap();

                        // Preparamos el csv
                        let schema = Schema::from(vec![
                            Field::new("Tiempo", DataType::Float64),
                            Field::new("Tension A", DataType::Float64),
                            Field::new("Tension B", DataType::Float64),
                            Field::new("Tension C", DataType::Float64),
                            Field::new("Intensidad A", DataType::Float64),
                            Field::new("Intensidad B", DataType::Float64),
                            Field::new("Intensidad C", DataType::Float64),
                        ].into_iter());

                        let mut df_simulacion = CsvReader::from_path(&csv_file).unwrap()
                        .with_schema(&schema)
                        .has_header(false)
                        .with_delimiter(b',')
                        .finish().unwrap();

                        let mut file = File::create(csv_file).unwrap();
                        CsvWriter::new(&mut file).finish(&mut df_simulacion).unwrap();
                    }

                });

                handles.push(handle);
            }

            for handle in handles {
```

```
            handle.join().unwrap();
        }

        let entries = fs::read_dir(r"C:\EMTP\atpdraw\Resultados procesados\
Simulaciones_Flujos_rev\Archivos dat").unwrap();
        for entry in entries.filter_map(|s| s.ok()).into_iter() {
            let dat_path = entry.path();
            fs::remove_file(dat_path).unwrap();
        }

    // endregion: ACONDICIONAMIENTO DE LOS ARCHIVOS RESULTADO DE ATP
    //--------------------------------------------------------------------

    }

    println!();

    let dir_csv_procesados = PathBuf::from(r"C:\EMTP\atpdraw\Resultados procesados\
Simulaciones_Flujos_rev\Archivos csv");
    let entries_dir_csv_procesados = fs::read_dir(&dir_csv_procesados).unwrap();
    let mut csv_files = vec![];

    for entry in entries_dir_csv_procesados.filter_map(|s| s.ok()).into_iter() {
        let entry_path = PathBuf::from(entry.path());
        csv_files.push(entry_path);
    }


    let mut csv_files_for_threads = vec![];

    while !csv_files.is_empty() {
        let chunk: Vec<PathBuf> = csv_files.drain(..std::cmp::min(chunk_size, csv_files.
len())).collect();
        csv_files_for_threads.push(chunk);
    }


    let shared_dir_csv_resultados = Arc::new(PathBuf::from(r"C:\EMTP\atpdraw\Resultados
procesados\Simulaciones_Flujos_rev\Archivos csv resultados"));

    let mut handles = Vec::new();
    for csv_files_assigned_to_thread in csv_files_for_threads.into_iter() {

        let dir_csv_resultados = Arc::clone(&shared_dir_csv_resultados);

        let handle = thread::spawn(move || {
            for csv_file in csv_files_assigned_to_thread {
                // Creamos el DataFrame leyendo el archivo csv
                let df_simulacion = CsvReader::from_path(&csv_file).unwrap()
                .infer_schema(Some(100))
                .has_header(true)
                .with_delimiter(b',')
                .finish()
                .unwrap();


                let delta_t = func_polars::extract_value_from_dataframe::<f64>(&
df_simulacion, 1, 0).unwrap();
                const TAU_INICIAL: f64 = 0.1;
                const TAU_REAL: f64 = 1.0;
                const INSTANTE_TRANSICION_A_TAU_REAL: f64 = 0.6;
```

```
                    let flujo_a = decaimiento_de_flujo(&df_simulacion, "Tension A", &delta_t
    , &TAU_INICIAL, &TAU_REAL, &INSTANTE_TRANSICION_A_TAU_REAL).unwrap();
                    let flujo_b = decaimiento_de_flujo(&df_simulacion, "Tension B", &delta_t
    , &TAU_INICIAL, &TAU_REAL, &INSTANTE_TRANSICION_A_TAU_REAL).unwrap();
                    let flujo_c = decaimiento_de_flujo(&df_simulacion, "Tension C", &delta_t
    , &TAU_INICIAL, &TAU_REAL, &INSTANTE_TRANSICION_A_TAU_REAL).unwrap();


                    let tiempo_iter = func_polars::df_column_f64_into_iter(&df_simulacion, "
    Tiempo").unwrap();
                    let tiempo = tiempo_iter.map(|s| s.unwrap() as f64).collect::<Vec<_>>();

                    let mut df_flujos = DataFrame::new(vec![
                        // Series::new("Tiempo", &tiempo[..tiempo.len()-1]),
                        Series::new("Tiempo", &tiempo),
                        Series::new("Flujo A", flujo_a.clone()),
                        Series::new("Flujo B", flujo_b.clone()),
                        Series::new("Flujo C", flujo_c.clone()),
                    ]).unwrap();


                    let csv_file_data_name: String = csv_file.file_name().unwrap().to_st
    ring_lossy().to_string();
                    let csv_file_result_name = format!("{}{}", "res_", csv_file_data_name);
                    let csv_file_result = dir_csv_resultados.join(csv_file_result_name);

                    let mut file = File::create(csv_file_result).unwrap();
                    CsvWriter::new(&mut file).finish(&mut df_flujos).unwrap();
                }
            });

            handles.push(handle);
        }
        for handle in handles {
            handle.join().unwrap();
        }



        let duration = start_time.elapsed();
        println!();
        println!("Time taken: {:.2} seconds", duration.as_secs_f64());
        println!();

        Ok(())
    }
```

Código A.5 Obtención de las intensidades máximas por fase.

```
use std::sync::Arc;
use std::thread;
use std::{time::Instant, path::PathBuf};
use std::fs::{self, File};
use std::io::{ErrorKind, Error};
use polars::export::num::complex::ComplexFloat;
use polars::prelude::{CsvReader, SerReader, DataFrame};
use polars::prelude::*;
use regex::Regex;
use glob::glob;
use structopt::StructOpt;
```

```
use funcs_tfg::func_polars::df_column_f64_into_iter;



#[derive(StructOpt, Debug)]
struct Opt {

    /// Número deseado de archivos por hilo
    #[structopt(short = "c", long, default_value = "4")]
    chunk_size: usize,

}

fn main() -> std::result::Result<(), Box<dyn std::error::Error>> {
    let start_time = Instant::now();
    let opt = Opt::from_args();
    let chunk_size = opt.chunk_size;


    // region: INTENSIDADES
    #[allow(non_snake_case)]
    let mut KNT = 1.0;
    let mut identificadores = Vec::new();
    for i in 1..= 400 {
        let coef: f64 = KNT / 20.0;
        let tclose = 0.785 + 0.001 * coef.floor();
        let topen = 0.744 + 0.001 * (KNT - 20.0 * coef.floor());

        let chunk = (KNT,tclose,topen);
        identificadores.push(chunk);
        KNT += 1.0;
    }

    let arc_identificadores: Arc<Vec<(f64, f64, f64)>> = Arc::from(identificadores);

    let mut handles = Vec::new();
    for csv_for_thread in csv_threads_paths.into_iter() {

        let shared_identificadores = Arc::clone(&arc_identificadores);


        let handle = thread::spawn(move || {

            let mut thread_result_vec = Vec::new();
            for path in csv_for_thread {
                let sim_num = extract_number_from_filename(&path).unwrap();
                let copy_identificadores = (*shared_identificadores).to_vec();
                let (sim_id, tclose, topen) = copy_identificadores[sim_num as usize -
1];

                // Leemos el csv creando un DataFrame
                let df_simulacion = CsvReader::from_path(&path).unwrap()
                .infer_schema(Some(100))
                .has_header(true)
                .with_delimiter(b',')
                .finish()
                .unwrap();

                let _intensidad_a_iter = df_column_f64_into_iter(&df_simulacion, "
Intensidad A").unwrap();
```

```
                let intensidad_a_vec = _intensidad_a_iter.map(|s| s.unwrap() as f64).
    collect::<Vec<f64>>();
                let intensidad_a_slice = &intensidad_a_vec[desired_t_index..
    intensidad_a_vec.len()];
                let mut intensidad_a_iter = intensidad_a_slice.iter().cloned();

                let _intensidad_b_iter = df_column_f64_into_iter(&df_simulacion, "
    Intensidad B").unwrap();
                let intensidad_b_vec = _intensidad_b_iter.map(|s| s.unwrap() as f64).
    collect::<Vec<f64>>();
                let intensidad_b_slice = &intensidad_b_vec[desired_t_index..
    intensidad_b_vec.len()];
                let mut intensidad_b_iter = intensidad_b_slice.iter().cloned();

                let _intensidad_c_iter = df_column_f64_into_iter(&df_simulacion, "
    Intensidad C").unwrap();
                let intensidad_c_vec = _intensidad_c_iter.map(|s| s.unwrap() as f64).
    collect::<Vec<f64>>();
                let intensidad_c_slice = &intensidad_c_vec[desired_t_index..
    intensidad_c_vec.len()];
                let mut intensidad_c_iter = intensidad_c_slice.iter().cloned();


                let mut max_int_a = 0.0;
                let mut max_int_b = 0.0;
                let mut max_int_c = 0.0;

                while let (Some(int_a), Some(int_b), Some(int_c)) = (intensidad_a_iter.
    next(), intensidad_b_iter.next(), intensidad_c_iter.next()) {
                    if int_a.abs() > max_int_a {
                        max_int_a = int_a.abs();
                    }

                    if int_b.abs() > max_int_b {
                        max_int_b = int_b.abs();
                    }

                    if int_c.abs() > max_int_c {
                        max_int_c = int_c.abs();
                    }
                }

                let result_vec = vec![sim_id, tclose, topen, max_int_a, max_int_b,
    max_int_c];
                thread_result_vec.push(result_vec);

            }
            thread_result_vec

        });
        handles.push(handle);
    }

    let mut threads_result_vec = Vec::new();
    for handle in handles {
        let thread_result = handle.join().unwrap();
        threads_result_vec.push(thread_result);
    }

    let mut vec_intensidades_maximas = Vec::new();
    for thread_vec_result in threads_result_vec {
        for vec_result in thread_vec_result {
```

```
                let result = (vec_result[0], vec_result[1], vec_result[2], vec_result[3],
    vec_result[4], vec_result[5]);
                vec_intensidades_maximas.push(result);
            }
        }


        let mut sim_id_vec = Vec::new();
        let mut tclose_vec = Vec::new();
        let mut topen_vec = Vec::new();
        let mut inta_vec = Vec::new();
        let mut intb_vec = Vec::new();
        let mut intc_vec = Vec::new();

        for (sim_id, tclose, topen, inta, intb, intc) in vec_intensidades_maximas {
            sim_id_vec.push(sim_id);
            tclose_vec.push(tclose);
            topen_vec.push(topen);
            inta_vec.push(inta);
            intb_vec.push(intb);
            intc_vec.push(intc);
        }

        let df_intensidades_maximas = DataFrame::new(vec![
            Series::new("Sim_ID", sim_id_vec),
            Series::new("T_close", tclose_vec),
            Series::new("T_open", topen_vec),
            Series::new("Int_A", inta_vec),
            Series::new("Int_B", intb_vec),
            Series::new("Int_C", intc_vec),
        ]).unwrap();

        let mut df_intensidades_maximas_sorted = df_intensidades_maximas.sort(["Sim_ID"],
    false).unwrap();

        println!();
        println!("{}", df_intensidades_maximas_sorted);


        let dir_csv_procesados = PathBuf::from(r"C:\EMTP\atpdraw\Resultados procesados\
    Simulaciones_Flujos_rev\Archivos csv procesados");
        let csv_file_intensidades_maximas = dir_csv_procesados.join("Intensidades maximas.
    csv");

        let mut file = File::create(csv_file_intensidades_maximas).unwrap();
        CsvWriter::new(&mut file).finish(&mut df_intensidades_maximas_sorted).unwrap();


        // endregion: INTENSIDADES

        let duration = start_time.elapsed();
        println!();
        println!("Time taken: {:.2} seconds", duration.as_secs_f64());
        println!();

        Ok(())
    }
```

Índice de Figuras

Índice de Códigos

Bibliografía

[1] Jose Ignacio Afonso, Pablo Toscano, and Isabel Briozzo, *Current transformer modeling for electromagnetic transient simulation in protection systems*, 2020.

[2] EMTP Alliance, *EMTP® History*, https://emtp.com/about-us/emtp-history, 2004.

[3] H K Høidalen, N Chiesa, A Avendaño, and B a Mork, *Developments in the hybrid transformer model – core modeling and optimization*, International Conference on Power Systems Transients (IPST 2011) (2011).

[4] H. K. Høidalen, A. Lotfi, S. Zirka, Y. Moroz, N. Chiesa, and B. A. Mork, *Benchmarking of hysteretic elements in topological transformer model*, Electric Power Systems Research **138** (2016).

[5] Hans K. Høidalen, Bruce A. Mork, Francisco Gonzalez, Dmitry Ishchenko, and Nicola Chiesa, *Implementation and verification of the hybrid transformer model in atpdraw*, Electric Power Systems Research **79** (2009).

[6] Hans Kristian Høidalen, László Prikler, and Francisco Peñaloza, *ATPDraw Users'Manual*, 2019.

[7] Shun Tsai Liu, Sy Ruen Huang, and Hung Wei Chen, *Using tacs functions within emtp to set up current-transformer model based on the jiles-atherton theory of ferromagnetic hysteresis*, IEEE Transactions on Power Delivery **22** (2007).

[8] Shun Tsai Liu, Sy Ruen Huang, Hung Wei Chen, and Ting Yen Hsien, *Current transformer module basing the jiles-atherton hysteresis model in emtp/atp simulation*, vol. 2005, 2005.

[9] Yoshitaka Tokunaga and Kunihiro Kubota, *Inrush current simulation of two-winding power transformer using machine parameters estimated by winding structure design procedure*, Electrical Engineering in Japan (English translation of Denki Gakkai Ronbunshi) **177** (2011).

[10] Sergey E. Zirka, Yuriy I. Moroz, Hans Kristian Høidalen, Abbas Lotfi, Nicola Chiesa, and Cesare M. Arturi, *Practical experience in using a topological model of a core-type three-phase transformer-no-load & inrush conditions*, IEEE Transactions on Power Delivery **32** (2017).