INSTITUTO POLITÉCNICO NACIONAL

ESCUELA SUPERIOR DE INGENIERÍA MECÁNICA Y ELÉCTRICA

SECCIÓN DE ESTUDIOS DE POSGRADO E INVESTIGACIÓN

DEPARTAMENTO DE INGENIERÍA ELÉCTRICA

IMPLEMENTACIÓN DE MODELOS DE TRANSFORMADORES PARA ANÁLISIS DE TRANSITORIOS ELECTROMAGNÉTICOS RÁPIDOS

T E S I S

QUE PARA OBTENER EL GRADO DE
MAESTRO EN CIENCIAS EN INGENIERÍA ELÉCTRICA

PRESENTA

JUAN MANUEL VILLANUEVA RAMÍREZ

MÉXICO D.F., DICIEMBRE 2013

SIP-14-BIS

INSTITUTO POLITÉCNICO NACIONAL
SECRETARÍA DE INVESTIGACIÓN Y POSGRADO

ACTA DE REVISIÓN DE TESIS

En la Ciudad de México, D. F. siendo las 12:00 horas del día 19 del mes de Noviembre del 2013 se reunieron los miembros de la Comisión Revisora de la Tesis, designada por el Colegio de Profesores de Estudios de Posgrado e Investigación de E.S.I.M.E.-ZAC. para examinar la tesis titulada:

IMPLEMENTACIÓN DE MODELOS DE TRANSFORMADORES PARA ANÁLISIS DE TRANSITORIOS ELECTROMAGNÉTICOS RÁPIDOS

Presentada por el alumno:

VILLANUEVA	RAMÍREZ	JUAN MANUEL
Apellido paterno	Apellido materno	Nombre(s)

Con registro:

B	1	1	0	7	2	4

aspirante de:

MAESTRO EN CIENCIAS EN INGENIERÍA ELÉCTRICA

Después de intercambiar opiniones los miembros de la Comisión manifestaron ***APROBAR LA TESIS***, en virtud de que satisface los requisitos señalados por las disposiciones reglamentarias vigentes.

LA COMISIÓN REVISORA

Directores de tesis

DR. PABLO GÓMEZ ZAMORANO

DR. FERMÍN PASCUAL ESPINO CORTÉS

Presidente

DR. DANIEL OLGUÍN SALINAS

Segundo Vocal

DR. FERMÍN PASCUAL ESPINO CORTÉS

Tercer Vocal

M. EN C. DAVID JUÁREZ AGUILAR

Secretario

DR. GERMÁN ROSAS ORTÍZ

PRESIDENTE DEL COLEGIO DE PROFESORES

DR. MAURO ALBERTO ENCISO AGUILAR

ESCUELA SUPERIOR DE INGENIERÍA MECÁNICA Y ELÉCTRICA · ESTADOS UNIDOS MEXICANOS · I P N

INSTITUTO POLITÉCNICO NACIONAL

SECRETARÍA DE INVESTIGACIÓN Y POSGRADO

CARTA CESIÓN DE DERECHOS

En la Ciudad de México, D.F. el día 19 del mes de Noviembre del año 2013, el que suscribe **Juan Manuel Villanueva Ramírez** alumno del Programa de Maestría en Ciencias en Ingeniería Eléctrica, con número de registro **B110724**, adscrito a la Sección de Estudios de Posgrado e Investigación de la ESIME Zacatenco del IPN, manifiesta que es el autor intelectual del presente trabajo de Tesis bajo la dirección del **Dr. Pablo Gómez Zamorano** y del **Dr. Fermín Pascual Espino Cortés** y cede los derechos del trabajo titulado **Implementación de modelos de transformadores para análisis de transitorios electromagnéticos rápidos**, al Instituto Politécnico Nacional para su difusión, con fines académicos y de investigación.

Los usuarios de la información no deben reproducir el contenido textual, gráficas o datos del trabajo sin el permiso expreso del autor y/o directores del trabajo. Este puede ser obtenido escribiendo a las siguientes direcciones **juanmvillanuevar@gmail.com**, **pablo.gomez.78@gmail.com** y/o **fpespino@gmail.com**. Si el permiso se otorga, el usuario deberá dar el agradecimiento correspondiente y citar la fuente del mismo.

Juan Manuel Villanueva Ramírez

AGRADECIMIENTOS

A mis padres, por el apoyo incondicional que siempre me han otorgado, por todos los valores que me han inculcado y por ser mi ejemplo a seguir.

A mi hermano, por todos los momentos de alegría compartidos y por las palabras de aliento brindadas.

A mis asesores: Dr. Pablo Gómez Zamorano y Dr. Fermín Pascual Espino Cortés por transmitirme sus conocimientos, por la confianza y la paciencia brindada, y por sus consejos y apoyo que fueron fundamentales para la culminación de este trabajo.

Al la comisión revisora por sus valiosos comentarios que permitieron mejorar este trabajo.

A todos los profesores de la Sección de Estudios de Posgrado e Investigación de la Escuela Superior de Ingeniería Mecánica y Eléctrica Unidad Zacatenco por compartir conmigo sus conocimientos y experiencias.

Al personal de apoyo por la amabilidad y disponibilidad demostrada.

A mis amigos y compañeros por todas las aventuras compartidas y por hacer de mis estudios de posgrado una experiencia inolvidable.

Al Instituto Politécnico Nacional por permitirme cursar mis estudios de posgrado en uno de sus programas y por la formación integral que me ha brindado.

Al Consejo Nacional de Ciencia y Tecnología y al Programa Institucional de Formación de Investigadores por el apoyo económico recibido para la realización de esta investigación.

RESUMEN

Los transformadores son componentes fundamentales en los Sistemas Eléctricos de Potencia, ya que permiten la transmisión de energía eléctrica que se produce en las centrales generadoras, así como su distribución a los diferentes tipos de consumidores. Sin embargo, después de las líneas de transmisión, son los elementos del sistema con mayor propensión a transitorios electromagnéticos.

Existen diferentes modelos matemáticos que son capaces de calcular la distribución de tensión a lo largo de los devanados del transformador ante un impulso de frente de onda rápido (un impulso atmosférico, por ejemplo); no obstante, ninguno de ellos ha sido implementado en programas comerciales de simulación hasta la fecha.

En el presente trabajo se realiza la implementación en MATLAB-Simulink® de dos modelos de devanado de transformador para el análisis de transitorios electromagnéticos rápidos: uno de parámetros concentrados basado en ecuaciones de estado y otro de parámetros distribuidos basado en la teoría de la línea de transmisión multiconductora. Una vez definidas sus características, ambos modelos se utilizan para el cálculo de la tensión transitoria en diferentes vueltas a lo largo del devanado de un transformador considerando el caso ideal y el caso con inclusión de pérdidas en serie.

Después de realizar las simulaciones correspondientes se constató que los dos modelos implementados cumplen satisfactoriamente con las funciones para las que fueron desarrollados. Además, comparando las formas de tensión obtenidas por cada uno de los modelos, se observó que existe una gran similitud entre ellas, aunque para el caso ideal, el modelo de parámetros concentrados no es capaz de reproducir algunos picos de tensión relacionados con las mayores frecuencias presentes en el fenómeno.

La explicación detallada del proceso llevado a cabo para la implementación de los modelos proporciona una metodología básica para la inclusión a futuro de nuevos modelos desarrollados por los usuarios de MATLAB-Simulink®. Las implementaciones fueron realizadas en este programa comercial debido a la sencillez de su lenguaje de programación y a las facilidades que otorga para realizar modelos propios.

ABSTRACT

Transformers are fundamental components in Electric Power Systems, because they allow the transmission of electric energy produced in power plants, as well as its distribution to different types of consumers. However, after transmission lines, they are the elements most affected by electromagnetic transients.

There are several mathematical models capable of computing voltage distribution along transformer windings when excited by a fast front impulse (such as a lightning impulse); nevertheless, to date none of them has been implemented in commercial simulation programs.

In this work, implementation of two transformer winding models for fast transient analysis in MATLAB-Simulink® is performed: a lumped-parameter model based on state-space equations and a distributed-parameter model based on multiconductor transmission line theory. Once their features have been established, both models are used to compute the transient voltage at different turns along a transformer winding considering the lossless case and the case including series losses.

After performing the corresponding simulations, it was noticed that both implemented models satisfactorily meet all functions for which they were developed. Also, comparing voltage waveforms from each model, their similarity is readily seen, although for the lossless case the lumped-parameter model is unable to reproduce some voltage peaks related to the highest frequencies involved in the phenomenon.

Detailed explanation of the process for implementing the models provides a basic methodology for future inclusion of new models delevoped by MATLAB-Simulink® users. Implementations were performed using this commercial software because of its simple programming language and easiness in developing custom models.

CONTENIDO

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

NOMENCLATURA

L	Inductancia serie del devanado
R	Resistencia serie del devanado
C_s	Capacitancia entre vueltas del devanado
R_s	Componente de pérdidas de C_s
C_g	Capacitancia a tierra del devanado
R_g	Componente de pérdidas de C_g
$\boldsymbol{C'}$	Matriz de capacitancia en forma nodal incluyendo el nodo de entrada
$\boldsymbol{G'}$	Matriz de conductancia en forma nodal incluyendo el nodo de entrada
$\boldsymbol{\Gamma'}$	Matriz de inductancia inversa en forma nodal incluyendo el nodo de entrada
$\boldsymbol{y'}(t)$	Vector de tensiones nodales incluyendo la tensión de entrada
$\dot{\boldsymbol{y}}'(t)$	Derivada respecto al tiempo del vector de tensiones nodales
$\boldsymbol{Q_C}, \boldsymbol{Q_G}, \boldsymbol{Q_L}$	Matrices de incidencia para los elementos capacitivos, conductivos e inductivos
$\boldsymbol{C_b}, \boldsymbol{G_b}, \boldsymbol{L_b}$	Matrices de rama de los elementos capacitivos, conductivos e inductivos de la red
$\boldsymbol{C}$	Matriz de capacitancia en forma nodal sin incluir el nodo de entrada
$\boldsymbol{G}$	Matriz de conductancia en forma nodal sin incluir el nodo de entrada
$\boldsymbol{\Gamma}$	Matriz de inductancia inversa en forma nodal sin incluir el nodo de entrada
$\boldsymbol{y}(t)$	Vector de salida de tensiones nodales sin incluir el nodo de entrada
$v(t)$	Tensión conocida del nodo de entrada
$\boldsymbol{C_k}, \boldsymbol{G_k}, \boldsymbol{\Gamma_k}$	k^{th} columna de $\boldsymbol{C'}, \boldsymbol{G'}$ y $\boldsymbol{\Gamma'}$ con la fila k^{th} eliminada
$\boldsymbol{X}(t)$	Vector de variables de estado
$\boldsymbol{A}, \boldsymbol{F}$	Matrices de coeficientes constantes
$\boldsymbol{B}, \boldsymbol{D}$	Matrices columna de coeficientes constantes
$\boldsymbol{U}$	Matriz identidad
$\boldsymbol{C_2}$	Matriz de capacitancia nodales eliminando la primera columna de la matriz $\boldsymbol{C}$
$\boldsymbol{Q_a}$	Matriz de incidencias de corriente eliminando la primera columna de la matriz $\boldsymbol{Q_i}$
$\boldsymbol{Y}(s)$	Matriz de admitancias nodales del circuito en el dominio s
$\boldsymbol{I}(s)$	Vector de corrientes nodales en el dominio s
$v(x,t)$	Tensión en un punto x del devanado
$i(x,t)$	Corriente en un punto x del devanado
Z_0	Impedancia característica de la línea de transmisión
$i_{k,m}, i_{m,k}$	Corrientes de nodales
I_k, I_m	Corrientes de historia
$\boldsymbol{Y}$	Matriz de admitancias nodales
$\boldsymbol{e}(t)$	Vector de tensiones nodales en el tiempo t
$\boldsymbol{i}(t)$	Vector de corrientes nodales inyectadas en el tiempo t
$\boldsymbol{I}$	Vector de términos de historia conocidos

Y_0	Admitancia característica de la línea $(1/Z_0)$
$v_F(t)$	Fuente de alimentación de tensión
R_F	Resistencia interna de la fuente de alimentación
Z_{eq}	Impedancia equivalente que representa el resto del devanado o la conexión al final del mismo
$\boldsymbol{T_v}$, $\boldsymbol{T_i}$	Matrices de transformación de similaridad
Z_{Cmi}	Impedancia característica de cada modo
v_{mi}	Velocidad de propagación de cada modo
Y_{con}	Admitancia de conexión entre el extremo final de una vuelta y el inicio de la siguiente
Y_{fin}	Admitancia que representa la conexión al final del devanado $(1/Z_{eq})$
$v_F(t)$	Fuente de alimentación de tensión
R_F	Resistencia interna de la fuente de alimentación
C_s	Capacitancia en serie
C_g	Capacitancia a tierra
ε_0	Permitividad del espacio libre
ε_r	Permitividad relativa del material dieléctrico entre vueltas
h	Altura del conductor rectangular
d_s	Distancia entre vueltas
w	Ancho de un conductor rectangular
d_g	Distancia entre una vuelta y el plano de tierra
C_{ev}	Capacitancia entre vueltas
C_{ed}	Capacitancia entre discos
ε_{ev}	Permitividad relativa del aislamiento entre vueltas
ε_{ed}	Permitividad relativa del aislamiento entre discos
ε_{ac}	Permitividad relativa del aceite aislante
d_{ev}	Distancia entre vueltas
d_{ed}	Distancia entre discos
k	Fracción del espacio circunferencial ocupado por el aceite
R_d	Profundidad radial del devanado
C_{ij}	Capacitancia entre las vueltas i y j (no es un elemento de la matriz $\boldsymbol{C}$)
ΔU_{ij}	Diferencia de potencial entre dichas vueltas
W_j	Energía electrostática obtenida por medio del MEF
$\boldsymbol{E}$	Campo eléctrico
d_{xW1}	Distancia en x del conductor al punto W1
d_{yW1}	Distancia en y del conductor al punto W1
d_{xW2}	Distancia en x del conductor al punto W2
d_{yW2}	Distancia en y del conductor al punto W2
μ_0	Permeabilidad del espacio libre
$K(k)$, $E(k)$	Integrales elípticas completas de primer y segundo orden
DMG	Distancia Media Geométrica
$\boldsymbol{L_g}$	Matriz de inductancia geométrica
$\boldsymbol{L_c}$	Matriz de inductancia de los conductores
ε_r	Permitividad relativa del material dieléctrico
c	Velocidad de la luz en el espacio libre

Z_c	Impedancia del conductor debida al efecto piel
ω	Frecuencia angular
W_{mag}	Energía magnética
R_{dc}	Resistencia de corriente directa
ρ	Resistividad del conductor
A	Área de la sección transversal del conductor
σ_c	Conductividad del conductor del devanado
μ_c	Permeabilidad del conductor del devanado
ρ_c	Resistividad del material del conductor
Z_{hf}	Impedancia a alta frecuencia
p	Profundidad de penetración compleja debida al efecto piel
Z_{nucleo}	Pérdidas en el núcleo debido a corrientes eddy
d_l	Anchura de la laminación
$\tan(\delta)$	Factor de pérdidas
C_{IT}	Capacitancia entre vueltas
C_a	Capacitancia entre dos vueltas separadas por una vuelta entre ellas
C_b	Capacitancia entre dos vueltas separadas por dos vueltas entre ellas
C_{ID}	Capacitancia entre dos discos adyacentes
C_{FS}	Capacitancia entre una vuelta y tierra
C_{LV}	Capacitancia entre los devanados de alta y baja tensión
t_f	Tiempo de frente de onda
t_h	Tiempo de decaimiento al valor medio
A_0	Valor pico

CAPÍTULO 1. INTRODUCCIÓN

1.1 DEFINICIÓN DEL PROBLEMA

Un Sistema Eléctrico de Potencia (SEP) tiene la finalidad de proveer energía eléctrica a todos los usuarios de manera confiable e ininterrumpida. Sin embargo, debido a que el consumo de energía eléctrica se ha incrementado de manera considerable en los últimos años a causa del crecimiento de la población, la complejidad y el tamaño de los SEP's han ido en aumento y por ende también se ha incrementado la dificultad de cumplir los criterios de calidad en el suministro de energía eléctrica.

El incremento en la velocidad de cómputo, el desarrollo de la simulación en tiempo real y el mejoramiento de los modelos y métodos numéricos existentes, han propiciado que las simulaciones realizadas y las herramientas de análisis permitan estudiar de manera precisa el comportamiento de los SEP's y, con ello, lograr que su operación y diseño sean eficientes [1].

Aunque la mayor parte del tiempo los SEP's se encuentran operando de forma estable, es de vital importancia realizar un análisis detallado de su comportamiento cuando existe un cambio súbito en las condiciones del circuito y se propicia la aparición de un transitorio electromagnético, debido a que es en este periodo cuando sus componentes se someten al máximo nivel de esfuerzo dieléctrico. El análisis adecuado de estos fenómenos provee herramientas para poder calcularlos e implementar medidas para su control [2].

Una representación precisa de los componentes eléctricos del sistema es esencial para realizar un análisis confiable de fenómenos transitorios. Aun cuando el desarrollo de modelos válidos para todo el rango de frecuencias es deseable, esto es muy difícil de conseguir o, en algunos casos, prácticamente imposible [1], [3]. Por lo anterior, el modelado de componentes eléctricos puede lograrse por medio de modelos matemáticos que son precisos para un rango de frecuencias correspondiente a un fenómeno en particular.

Una de las clasificaciones más aceptadas es la propuesta por la Comisión Electrotécnica Internacional (IEC, por sus siglas en inglés) y el Consejo Internacional de Grandes Redes Eléctricas (CIGRE, por sus siglas en francés), en la que los rangos de frecuencias se clasifican en cuatro grupos: oscilaciones de

baja frecuencia, desde 0.1 Hz hasta 3 kHz; sobretensiones de frente de onda lento, desde 50/60 Hz hasta 20 kHz; sobretensiones de frente de onda rápido, desde 10 kHz hasta 3MHz; y sobretensiones de frente de onda muy rápido, desde 100 kHz hasta 50 MHz [4].

Al existir representaciones de los componentes eléctricos para los diferentes rangos de frecuencia, se podría pensar que la selección del modelo debería realizarse de forma automática tomando como base el rango de frecuencia del transitorio que se desea simular; sin embargo, el rango de frecuencia del caso de prueba normalmente es desconocido antes de realizar la simulación. Esto puede resolverse al consultar tablas de clasificaciones de fenómenos transitorios como la que se muestra en la Tabla 1.1.

Tabla 1.1 Origen y rangos de frecuencia de transitorios electromagnéticos en Sistemas de Potencia [1].

Origen	Rango de frecuencia
Ferroresonancia	0.1 Hz a 1 kHz
Rechazo de carga	0.1 Hz a 3 kHz
Liberación de una falla	50 Hz a 3 kHz
Cambio de línea	50 Hz a 20 kHz
Tensiones de recuperación transitorias	50 Hz a 100 kHz
Descarga atmosférica	10 kHz a 3 MHz
Desconexión de subestaciones aisladas en gas	100 kHz a 50 MHz

Los transformadores son un componente fundamental en los SEP’s, ya que permiten la transmisión de energía eléctrica que se produce en las centrales generadoras, así como su distribución a los diferentes tipos de consumidores. Debido a que los transformadores se encuentran conectados a la red eléctrica, se encuentran sujetos a diferentes tipos de tensiones transitorias cuyas magnitudes y formas deben identificarse para definir los esfuerzos dieléctricos soportados por el aislamiento.

Cuando se trata de frecuencias bajas, la distribución de tensión a lo largo de los devanados es lineal. En el caso de los transitorios rápidos, una gran porción de la tensión aplicada se distribuye en las primeras vueltas del devanado. A su vez, las oscilaciones de alta frecuencia pueden generar resonancias internas que pueden dañar el aislamiento del transformador [1].

Dado que no todas las características del transformador toman parte en todos los fenómenos transitorios, en la Tabla 1.2 se muestra una guía comúnmente

utilizada en el modelado de transformadores cuando se toman en consideración los rangos de frecuencia.

Tabla 1.2 Guía para el modelado de transformadores [1].

Parámetro/ Efecto	Transitorios de baja frecuencia	Transitorios de frente de onda lento	Transitorios de frente de onda rápido	Transitorios de frente de onda muy rápido
Impedancia de corto circuito	Muy importante	Muy importante	Importante	Despreciable
Saturación	Muy importante	Importante	Despreciable	Despreciable
Pérdidas en el núcleo	Importante	Despreciable	Despreciable	Despreciable
Corrientes eddy	Muy importante	Importante	Despreciable	Despreciable
Acoplamiento capacitivo	Despreciable	Importante	Muy importante	Muy importante

El modelado de transformadores para el análisis de transitorios electromagnéticos es complicado debido a diferentes factores: su comportamiento es no lineal y dependiente de la frecuencia; existen muchas posibilidades en la construcción del núcleo y del devanado; muchos de sus atributos necesitan ser representados de forma correcta (inductancias propias y mutuas entre bobinas, flujos magnéticos de dispersión, efecto piel y efecto de proximidad en bobinas, saturación magnética del núcleo, histéresis, pérdidas por corrientes eddy en el núcleo y efectos capacitivos) [1].

1.2 OBJETIVO

Implementar modelos del devanado del transformador para transitorios electromagnéticos rápidos en un programa comercial de simulación.

1.3 JUSTIFICACIÓN

Actualmente, los modelos del transformador para análisis de transitorios electromagnéticos disponibles en programas comerciales de simulación sólo contemplan transitorios de baja y mediana frecuencia (en el orden de los kHz). Sin embargo, uno de los estudios más importantes en el diseño de un transformador está relacionado precisamente con un evento de alta frecuencia: la prueba al impulso atmosférico, ya que en la mayoría de los casos esta prueba define el máximo esfuerzo dieléctrico al que el transformador estará sujeto al ser instalado

en el campo. La simulación de transitorios rápidos se realiza, en la actualidad, por medio de modelos de propósito específico, los cuales requieren para su implementación de un alto grado de experiencia y conocimiento por parte del usuario.

En este trabajo se realiza la implementación directa de modelos de transformador para transitorios de alta frecuencia en un programa comercial de simulación (MATLAB-Simulink), de tal forma que sean de uso general sin la necesidad de conocer a profundidad los modelos matemáticos que los rigen y que puedan realizarse con ellos estudios de forma sencilla.

1.4 ANTECEDENTES

Antes del desarrollo de los primeros programas computacionales, los problemas relacionados con ondas viajeras fueron estudiados en las décadas de 1920 y 1930 con métodos gráficos; principalmente se desarrollaron dos técnicas: la técnica del diagrama de Bewley y el método de Bergeron [5], [6].

En la técnica del diagrama de Bewley se calculan la posición y dirección del movimiento de las ondas incidentes, reflejadas y transmitidas en el sistema en cada instante de tiempo por medio de coeficientes de reflexión y refracción [7], [2]. Por otro lado, el método de Bergeron (llamado así porque fue Bergeron quien lo aplicó inicialmente para problemas de propagación de sistemas hidráulicos y planeó su analogía con la propagación de ondas electromagnéticas), no necesita coeficientes de reflexión y refracción; en su lugar, usa relaciones lineales entre voltajes y corrientes, las cuales son invariantes cuando son vistas por un observador ficticio que viaja con la onda [5], [3].

Con el desarrollo de la computadora digital, ambas técnicas fueron adoptadas para su implementación; sin embargo, el método de Bergeron logró ajustarse de mejor manera para la realización de simulaciones computacionales y la mayoría de los programas de uso general existentes a la fecha lo utilizan.

En los primeros programas que utilizaban este método, los elementos concentrados de inductancia y capacitancia eran representados por ramas, o las ecuaciones diferenciales eran transformadas en ecuaciones de diferencias algebraicas por medio del método numérico de integración basado en la regla trapezoidal, el cual posee una estabilidad numérica absoluta. Este método pertenece a los esquemas de integración implícitos, los cuales han ganado importancia en la solución de sistemas rígidos [5].

Originalmente, los programas fueron escritos en su mayoría para resolver redes monofásicas, pero desde entonces han sido extendidos para la solución de redes polifásicas.

Los métodos mencionados anteriormente son eficientes únicamente para líneas sin pérdidas y sin distorsiones. Concentrar el valor de las pérdidas en uno o varios puntos de la línea produce un grado de exactitud razonable en los resultados. Sin embargo, esta técnica sólo es aceptable si la resistencia es constante y pequeña comparada con la impedancia característica de la línea [8], [3], [5].

Actualmente, las herramientas de simulación de transitorios electromagnéticos se clasifican en dos categorías principales: fuera de línea (off-line) y en tiempo real (real-time).

El propósito de una herramienta de simulación fuera de línea es realizar simulaciones mediante una computadora personal. Aun cuando una herramienta fuera de línea se debe diseñar para ser altamente eficiente usando métodos numéricos poderosos y técnicas de programación, no tiene ninguna restricción de tiempo y puede hacerse lo más precisa posible con los datos disponibles, modelos y operaciones matemáticas relacionadas. Entre las herramientas fuera de línea se encuentran las del tipo de análisis nodal para sistemas de potencia y circuitos electrónicos, las de ambientes de modelado de propósito general, métodos híbridos y métodos en el dominio de la frecuencia [1].

Las simulaciones en tiempo real son capaces de generar resultados en sincronismo con un reloj de tiempo real. Estas herramientas tienen la ventaja de ser capaces de interconectarse con dispositivos físicos y mantener un intercambio de datos dentro del reloj de tiempo real. La capacidad de realizar cálculos e interconexiones en tiempo real impone restricciones importantes en el diseño de dichas herramientas [4]. Entre los principales tipos de simuladores se encuentran: Analizadores de Transitorios en Redes (TNA), Simuladores Digitales de Tiempo Real y Sistemas de Reproducción en Tiempo Real [4].

A pesar de que las técnicas en el dominio del tiempo han tenido preferencia en el cálculo de transitorios electromagnéticos, las técnicas en el dominio de la frecuencia tienen algunas ventajas que pueden ser consideradas en algunas aplicaciones: el modelado y cálculos puede ser más riguroso con elementos de parámetros distribuidos dependientes de la frecuencia y los errores numéricos pueden ser determinados y controlados de mejor manera [1], [4].

Existen muchos casos en los cuales el análisis en el dominio de la frecuencia ha sido fundamental para el avance de las técnicas en el dominio del tiempo; algunos de estos desarrollos son: modelos completos de líneas y cables dependientes de la frecuencia; técnicas de convolución, ajuste racional y síntesis de redes equivalentes [4], [1].

Diferentes técnicas en el dominio de la frecuencia han sido desarrolladas a través del tiempo y se encuentran basadas en la transformada de Fourier, la transformada numérica de Laplace o la transformada Z [1], [6], [9], [10], [11].

1.5 ESTADO DEL ARTE

Se han realizado diversas investigaciones para determinar la distribución de tensión en los devanados de los transformadores al ser excitados por impulsos de frente de onda rápido. Los modelos propuestos por los autores de dichas investigaciones han considerado representaciones de parámetros concentrados y parámetros distribuidos, principalmente en el dominio de la frecuencia. Esto se debe esencialmente a que la descripción de los parámetros dependientes de la frecuencia se realiza de manera directa en ese dominio; caso contrario al modelado en el dominio del tiempo, donde se tienen que realizar convoluciones y aproximaciones racionales para su representación. Algunos de estos trabajos se mencionan a continuación:

En 1960, Rabins [12] realizó el cálculo de la distribución de tensión en una bobina de una sola capa considerándola como una línea de transmisión mutliconductora. Este concepto sería retomado por Guardado y Cornick [13] para su aplicación en el cálculo de la distribución de tensión en devanados de máquinas. Este modelo se basa en la conexión en zig-zag de los diferentes conductores de la línea, de tal forma que cada uno de ellos represente una vuelta de un devanado completo.

En 1978, Miki et al. [14] propusieron un método para el cálculo de la distribución de tensión y la transferencia de tensión entre devanados de un transformador. Además, proponen un método para calcular la inductancia mutua entre dos bobinas con sección transversal de forma arbitraria.

En 1997, Shibuya et al. [15] utilizaron un modelo en el domino de la frecuencia que combina la teoría de la línea de transmisión monofásica con la teoría de la línea de transmisión multiconductora y se aplica a bobinas tipo disco de un transformador. La primera se utiliza para simular el devanado completo del transformador y la segunda para analizar el fenómeno que ocurre en la primera

bobina del devanado. Posteriormente, los resultados en el dominio del tiempo se obtienen empleando la transformada discreta de Fourier.

En 1998, Gustavsen y Semylen [16] presentaron una metodología para el modelado de transitorios en transformadores de potencia, basada en la medición o cálculo de respuestas en el dominio de la frecuencia. Las respuestas se aproximan por medio de funciones racionales utilizando el método de vector fitting.

En 2001, Shibuya et al. [17] utilizaron nuevamente un modelo en el dominio de la frecuencia, pero esta vez utilizan la teoría de la línea multiconductora para representar el devanado en su totalidad. En el mismo año, Al Fuhaid [18] presentó un modelo de parámetros distribuidos en el dominio de Laplace para realizar el análisis en la frecuencia de un transformador monofásico de dos devanados. Las expresiones desarrolladas toman en cuenta los acoplamientos capacitivos e inductivos entre los dos devanados, además del acoplamiento capacitivo entre vueltas dentro de cada devanado.

En 2004, Gustavsen [19] describió la configuración para la medición por medio de un analizador de redes en las terminales de un transformador y con ello desarrollar un modelo de caja negra de amplio espectro dependiente de la frecuencia.

En 2006, Liang et al. [20] utilizaron un modelo de parámetros concentrados en el dominio de la frecuencia para calcular las sobretensiones en los devanados de un transformador expuesto a transitorios muy rápidos. Posteriormente la respuesta en el dominio del tiempo se obtiene utilizando la técnica de vector fitting y convoluciones recursivas.

En 2007, Popov et al. [21] propusieron un modelo basado en la teoría de la línea de transmisión multiconductora en el dominio de la frecuencia que incluye las pérdidas dependientes de la frecuencia en el núcleo y en los conductores. Para realizar el cálculo de parámetros, el modelo sólo requiere información de la geometría del núcleo y del devanado, así como de los parámetros eléctricos y magnéticos de los materiales usados. Los resultados obtenidos verifican que el modelo puede utilizarse para simular la distribución de tensión en los devanados del transformador en un amplio rango de frecuencias

En 2008, Zhang et al. [22] utilizaron un modelo híbrido en el dominio de la frecuencia basado en las dos teorías de la línea de transmisión. Primero utilizan la teoría de la línea monofásica para calcular las tensiones en los extremos finales de cada bobina y posteriormente utilizan la teoría de la línea multiconductora para

obtener las sobretensiones a lo largo de cada una de las vueltas. En el mismo año, Zhu et al. [23] utilizaron un modelo híbrido en el dominio de la frecuencia para analizar la distribución de tensión en las primeras vueltas del devanado de un transformador. El devanado se divide en tres partes, las cuales se simulan por medio de la teoría de la línea multiconductora, la teoría de la línea monofásica y por una impedancia equivalente, respectivamente.

En la Sección de Estudios de Posgrado en Investigación de la Escuela Superior de Ingeniería Mecánica y Eléctrica Unidad Zacatenco se han realizado algunos trabajos con relación al modelado y cálculo de la distribución de tensión en devanados de transformadores.

En 2002, David Juárez Aguilar [24] realizó un estudio de la distribución de tensión en transformadores tipo columna. Obtuvo la distribución de tensión inicial y transitoria para dos casos: en el primero de ellos sólo considera el devanado que se encuentra energizado; para el segundo caso se considera además la presencia del devanado de baja tensión para observar los efectos que éste produce en los valores de tensión del devanado energizado. Los resultados obtenidos se compararon con mediciones hechas por Industrias IEM.

En 2009, José Antonio de León Brito [25] realizó un estudio del efecto de las tensiones tipo PWM en los sistemas de aislamiento de transformadores de media tensión alimentados por variadores de velocidad. Implementó un modelo de parámetros concentrados cuyos parámetros fueron calculados previamente por medio de un programa basado en el método de elemento finito. Obtuvo la distribución de tensión en los devanados primario y secundario de un transformador alimentado por fuentes de tensión PWM de uno y dos niveles

En 2012, Diana Soto Meza [26] desarrolló una metodología para el diseño dieléctrico de un transformador. Utilizando la interconexión de MATLAB® y COMSOL Multiphysics®, realizó el cálculo de parámetros del transformador para utilizarlos en un modelo de transformador de parámetros distribuidos en el dominio de la frecuencia al cual se le aplica una onda de impulso atmosférico y, posteriormente, calculó los esfuerzos dieléctricos presentes en el devanado.

En 2013, Williams Giovanni Nájera Gutiérrez [27] realizó un estudio del esfuerzo dieléctrico que producen excitaciones tipo PWM en devanados de transformadores utilizando un modelo de parámetros concentrados en el dominio del tiempo implementado en MATLAB®. Adicionalmente, realizó mediciones de descargas parciales en laboratorio para conocer el efecto del incremento de la

relación *dv/dt* en excitaciones aplicadas al material dieléctrico utilizado entre espiras del devanado del transformador.

1.6 APORTACIONES

- Se obtienen dos nuevos bloques en MATLAB-Simulink® que representan modelos de transformador para transitorios electromagnéticos rápidos en el dominio del tiempo.

- Mediante estos bloques, se obtienen gráficas de contorno de la distribución de tensión, las cuales constituyen una herramienta importante en la localización de esfuerzos dieléctricos a lo largo del devanado del transformador.

- Se obtiene una metodología básica para la inclusión a futuro de nuevos modelos desarrollados por los usuarios de MATLAB-Simulink®.

1.7 LIMITACIONES Y ALCANCES

1.7.1 LIMITACIONES

- Los modelos del transformador implementados no incluyen parámetros dependientes de la frecuencia como son las pérdidas por efecto piel y pérdidas por efecto de proximidad.

- Los modelos implementados no incluyen los parámetros relacionados con las pérdidas en derivación.

- Los resultados obtenidos no se validan experimentalmente. Sin embargo, los modelos implementados ya han sido validados previamente por otros autores.

1.7.2 ALCANCES

- Se implementan dos modelos de transformador en MATLAB-Simulink®: uno de parámetros concentrados y otro de parámetros distribuidos.

- Se realiza la simulación de un transitorio electromagnético rápido en el devanado de un transformador.

- Se comparan los resultados obtenidos con ambos modelos para los casos ideal y con la inclusión de pérdidas en serie.

- Se obtienen las gráficas de contorno de la distribución de tensión de todas las vueltas del devanado durante el tiempo de simulación.

1.8 ESTRUCTURA DE LA TESIS

Capítulo 1, *Introducción*: Se describen los objetivos, justificación, antecedentes, limitaciones y alcances de este trabajo. Además, se presenta el estado del arte sobre el tema.

Capítulo 2, *Modelos del Transformador para transitorios electromagnéticos rápidos*: Se describen los modelos utilizados para el análisis de transitorios electromagnéticos rápidos en los devanados del transformador, mencionando las ventajas y desventajas de cada uno de ellos.

Capítulo 3, *Determinación de parámetros para transitorios electromagnéticos rápidos*: Se presentan algunos de los métodos de cálculo de parámetros para modelos de transformador de alta frecuencia. Además, se describe un algoritmo alternativo de cálculo de capacitancia basado en el método de imágenes.

Capítulo 4, *Implementación de modelos de transformador en MATLAB-Simulink*: Se explica de forma detallada el proceso realizado para la implementación de modelos de transformador de parámetros concentrados y parámetros distribuidos como parte de una biblioteca dentro del programa de simulación MATLAB-Simulink®.

Capítulo 5, *Pruebas y análisis de resultados*: Se realizan algunas pruebas de funcionamiento de las características de los modelos implementados. Ambos se utilizan en la simulación de un fenómeno transitorio rápido en el devanado de un transformador.

Capítulo 6, *Conclusiones*: Resume los principales resultados y logros del trabajo, además de las recomendaciones para trabajos a futuro relacionados con la implementación de modelos de transformador para transitorios electromagnéticos rápidos.

Anexo: Muestra de forma completa los códigos de los modelos implementados en MATLAB-Simulink®

CAPÍTULO 2. MODELOS DEL TRANSFORMADOR PARA TRANSITORIOS ELECTROMAGNÉTICOS RÁPIDOS

En este capítulo se hace una descripción de los modelos utilizados para el análisis de transitorios electromagnéticos rápidos en los devanados del transformador, mencionando las ventajas y desventajas que cada uno de ellos.

2.1 INTRODUCCIÓN

Los transformadores, después de las líneas de transmisión, son los elementos del sistema eléctrico con mayor propensión a transitorios electromagnéticos [2]. El análisis del comportamiento de los transformadores sometidos a este tipo de fenómenos es de gran importancia porque proporciona los valores de tensión a los que se encontrarán expuestos todos sus componentes y, por medio de ellos, se puede llevar a cabo el diseño de su sistema dieléctrico. La forma común de llevar a cabo este tipo de estudios es haciendo un análisis de la respuesta de los devanados aplicando formas de onda establecidas en estándares [28].

Existen diferentes métodos para determinar la distribución de tensión dentro de los devanados del transformador, entre los que se encuentran mediciones a tensión reducida en modelos físicos reales o a escala y el análisis computacional de modelos matemáticos. El método más conveniente de utilizar es aquel que representa el devanado del transformador por medio de un modelo matemático y/o circuital al que se le aplica una señal de entrada dependiente del tiempo. Con este método, y el apoyo de programas computacionales de simulación, es económicamente factible examinar una gran variedad de alternativas de diseño del devanado en un lapso de tiempo relativamente corto.

Los modelos de devanado del transformador más utilizados para el análisis de transitorios electromagnéticos rápidos son aquellos denominados de caja gris [1], los cuales consideran la propagación y la distribución de un pulso incidente a lo largo de sus espiras.

La representación de un segmento diferencial de un devanado del transformador se muestra en la Figura 2.1

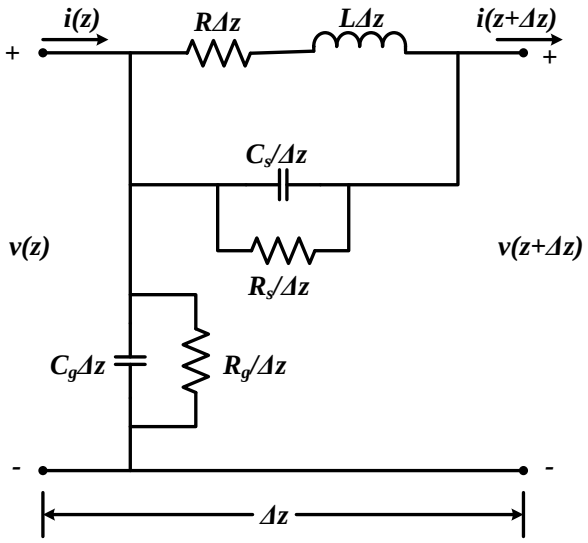


Figura 2.1 Circuito equivalente por unidad de longitud del devanado de un transformador [28], [18].

donde:

L = Inductancia serie del devanado
R = Resistencia serie del devanado
C_s = Capacitancia entre vueltas del devanado
R_s = Componente de pérdidas de C_s
C_g = Capacitancia a tierra del devanado
R_g = Componente de pérdidas de C_g

La representación mostrada en la Figura 2.1 puede ser descrita usando un modelo de parámetros concentrados conectados en escalera o un modelo de parámetros distribuidos utilizando la teoría de la línea de transmisión.

2.2 MODELOS DE PARÁMETROS CONCENTRADOS

La validez de un modelo de parámetros concentrados se encuentra limitada por la longitud correcta de cada segmento. Para un estudio transitorio en el orden de cientos de kilohertz un segmento por bobina puede ser suficiente, mientras que para un estudio en el orden de megahertz, podrá ser necesario considerar un segmento por vuelta del devanado [29]. Esto conlleva la solución de sistemas de gran tamaño y, en consecuencia, la simulación puede ser muy costosa computacionalmente. La solución de este tipo de modelos puede llevarse a cabo mediante herramientas de simulación de transitorios electromagnéticos (EMTP) o por medio de sus correspondientes ecuaciones de estado.

2.2.1 Modelo basado en Ecuaciones de Estado sin inclusión de pérdidas en serie

El circuito equivalente del devanado del transformador sin inclusión de pérdidas en serie se muestra en la Figura 2.2. El devanado se encuentra dividido en secciones, cada una de las cuales contiene una inductancia, una capacitancia en serie, una capacitancia a tierra y una conductancia que representa las pérdidas en el dieléctrico. Los elementos inductivos se encuentran acoplados mutuamente.

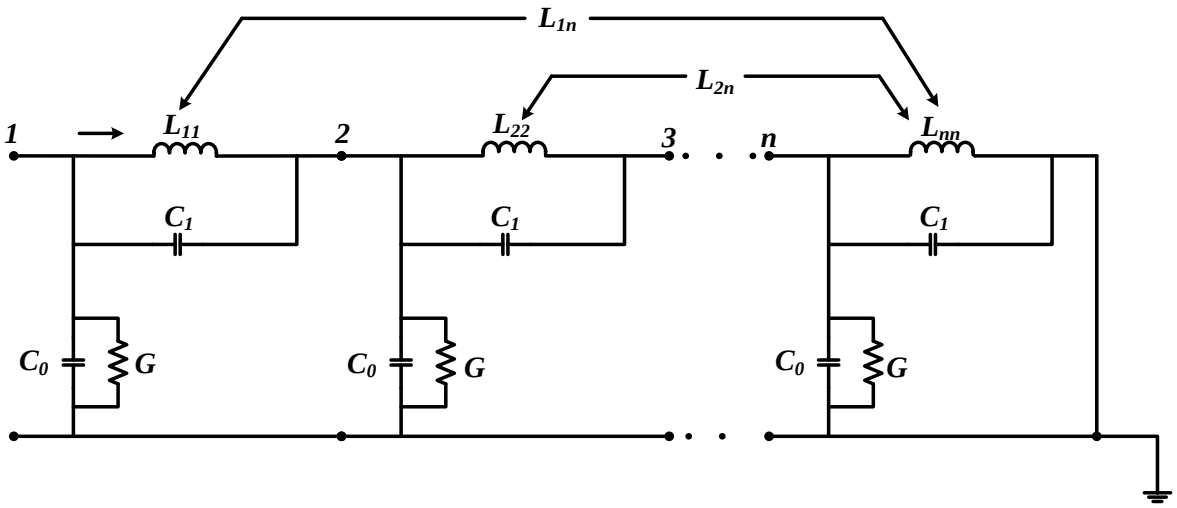


Figura 2.2. Circuito equivalente del devanado del transformador sin inclusión de pérdidas en serie [30].

Las ecuaciones de red para el circuito mostrado se obtienen en forma nodal [31] como:

$$\boldsymbol{C'}\ddot{\boldsymbol{y}}'(t) + \boldsymbol{G'}\dot{\boldsymbol{y}}'(t) + \boldsymbol{\Gamma'}\boldsymbol{y}'(t) = \boldsymbol{0} \qquad (2.1)$$

donde:

$\boldsymbol{C'}$	=	Matriz de capacitancia en forma nodal incluyendo el nodo de entrada
$\boldsymbol{G'}$	=	Matriz de conductancia en forma nodal incluyendo el nodo de entrada
$\boldsymbol{\Gamma'}$	=	Matriz de inductancia inversa en forma nodal incluyendo el nodo de entrada
$\boldsymbol{y}'(t)$	=	Vector de tensiones nodales incluyendo la tensión de entrada
$\dot{\boldsymbol{y}}'(t)$	=	Derivada respecto al tiempo del vector de tensiones nodales

La relación entre las matrices en forma nodal y las matrices de rama está definida por:

$$\left.\begin{aligned} \boldsymbol{C'} &= \boldsymbol{Q_C C_b Q_C^T} \\ \boldsymbol{G'} &= \boldsymbol{Q_G G_b Q_G^T} \\ \boldsymbol{\Gamma'} &= \boldsymbol{Q_L L_b^{-1} Q_L^T} \end{aligned}\right\} \tag{2.2}$$

donde $\boldsymbol{Q_C}$, $\boldsymbol{Q_G}$ y $\boldsymbol{Q_L}$ son las matrices de incidencia para los elementos capacitivos, conductivos e inductivos, y $\boldsymbol{C_b}$, $\boldsymbol{G_b}$ y $\boldsymbol{L_b}$ son las matrices de rama de los elementos capacitivos, conductivos e inductivos de la red, respectivamente.

El número de ecuaciones puede ser reducido al extraer el nodo de entrada k, debido a que la tensión de este nodo es conocida. La ecuación (2.1) puede ser escrita de la siguiente manera:

$$\boldsymbol{C}\ddot{\boldsymbol{y}}(t) + \boldsymbol{G}\dot{\boldsymbol{y}}(t) + \boldsymbol{\Gamma}\boldsymbol{y}(t) = -\boldsymbol{C_k}\ddot{v}(t) - \boldsymbol{G_k}\dot{v}(t) - \boldsymbol{\Gamma_k}v(t) \tag{2.3}$$

donde:

$\boldsymbol{C}$	=	Matriz de capacitancia en forma nodal sin incluir el nodo de entrada
$\boldsymbol{G}$	=	Matriz de conductancia en forma nodal sin incluir el nodo de entrada
$\boldsymbol{\Gamma}$	=	Matriz de inductancia inversa en forma nodal sin incluir el nodo de entrada
$\boldsymbol{y}(t)$	=	Vector de salida de tensiones nodales sin incluir el nodo de entrada
$v(t)$	=	Tensión conocida del nodo de entrada
$\boldsymbol{C_k}, \boldsymbol{G_k}, \boldsymbol{\Gamma_k}$	=	k^{th} columna de $\boldsymbol{C'}, \boldsymbol{G'}$ y $\boldsymbol{\Gamma'}$ con la fila k^{th} eliminada

Se puede observar en la Figura 2.2 que existen $2n$ ramas capacitivas en la red. Por lo tanto, el tamaño de la matriz de capacitancia de rama $\boldsymbol{C_b}$ es de $2n \times 2n$. Si las ramas se enumeran como en la Figura 2.3, ésta puede escribirse de la siguiente manera:

$$\boldsymbol{C_b} = \begin{bmatrix} C_0 & 0 & 0 & \cdots & 0 \\ 0 & C_1 & 0 & \cdots & 0 \\ 0 & 0 & C_0 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & 0 & \cdots & C_1 \end{bmatrix} \tag{2.4}$$

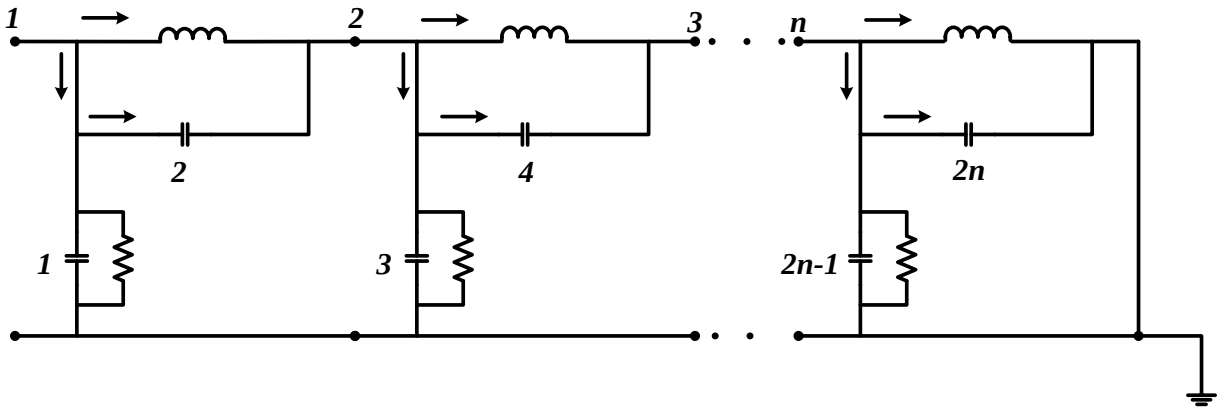


Figura 2.3. Numeración de las ramas capacitivas [30].

La matriz de incidencia correspondiente $\boldsymbol{Q_C}$ de orden $n \times 2n$ es:

$$\boldsymbol{Q_C} = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 \\ 0 & -1 & 1 & 1 & 0 & 0 & \cdots & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 1 & 1 & \cdots & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & \cdots & 0 & 0 & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & 0 & 0 & 0 & 0 & \cdots & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \cdots & -1 & 1 & 1 \end{bmatrix} \tag{2.5}$$

De forma similar, la matriz de conductancia de rama $\boldsymbol{G_b}$ y la matriz de incidencia $\boldsymbol{Q_G}$ de orden $n \times n$ para la Figura 2.4 están dadas por:

$$\boldsymbol{G_b} = \begin{bmatrix} G & 0 & 0 & \cdots & 0 \\ 0 & G & 0 & \cdots & 0 \\ 0 & 0 & G & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & 0 & \cdots & G \end{bmatrix} \tag{2.6}$$

$$\boldsymbol{Q_G} = \begin{bmatrix} 1 & 0 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 1 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 0 & 1 & \cdots & 0 & 0 \\ 0 & 0 & 0 & 0 & \cdots & 0 & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & 0 & 0 & \cdots & 1 & 0 \\ 0 & 0 & 0 & 0 & \cdots & 0 & 1 \end{bmatrix} \tag{2.7}$$

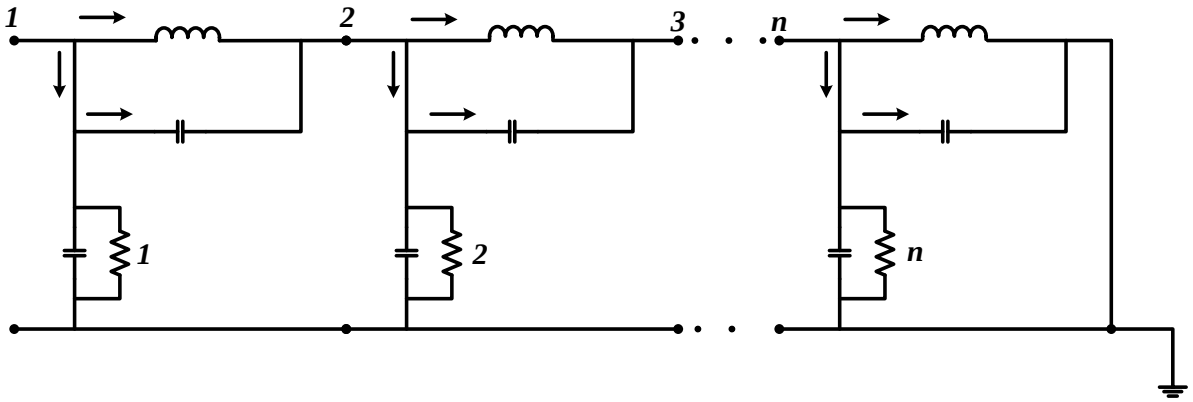


Figura 2.4. Numeración de las ramas conductivas [30].

La matriz de inductancia de rama $\boldsymbol{L_b}$ y su matriz de correspondiente matriz de incidencia $\boldsymbol{Q_L}$ de orden $n \times n$ para la Figura 2.5 están definidas por:

$$\boldsymbol{L_b} = \begin{bmatrix} L_{11} & L_{12} & L_{13} & \cdots & L_{1n} \\ L_{21} & L_{22} & L_{23} & \cdots & L_{2n} \\ L_{31} & L_{32} & L_{33} & \cdots & L_{3n} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ L_{n1} & L_{n2} & L_{n3} & \cdots & L_{nn} \end{bmatrix} \tag{2.8}$$

$$\boldsymbol{\Gamma} = \boldsymbol{L_b^{-1}} = \begin{bmatrix} \Gamma_{11} & \Gamma_{12} & \Gamma_{13} & \cdots & \Gamma_{1n} \\ \Gamma_{21} & \Gamma_{22} & \Gamma_{23} & \cdots & \Gamma_{2n} \\ \Gamma_{31} & \Gamma_{32} & \Gamma_{33} & \cdots & \Gamma_{3n} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ \Gamma_{n1} & \Gamma_{n2} & \Gamma_{n3} & \cdots & \Gamma_{nn} \end{bmatrix} \tag{2.9}$$

$$\boldsymbol{Q_L} = \begin{bmatrix} 1 & 0 & 0 & 0 & \cdots & 0 & 0 \\ -1 & 1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & -1 & 1 & 0 & \cdots & 0 & 0 \\ 0 & 0 & -1 & 1 & \cdots & 0 & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & 0 & 0 & \cdots & 1 & 0 \\ 0 & 0 & 0 & 0 & \cdots & -1 & 1 \end{bmatrix} \tag{2.10}$$

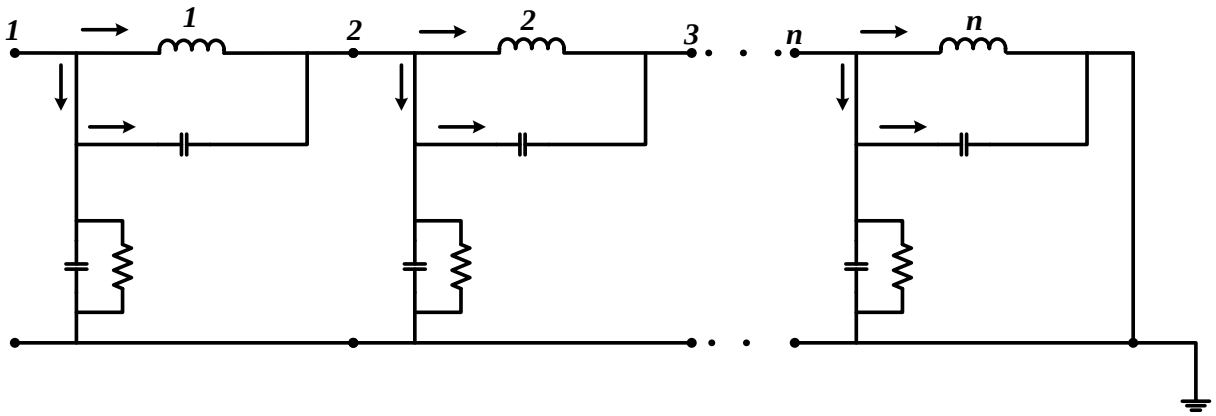


Figura 2.5. Numeración de las ramas inductivas [30].

El modelo de espacio de estado de la ecuación diferencial (2.3) es:

$$\dot{\boldsymbol{X}}(t) = \boldsymbol{A}\boldsymbol{X}(t) + \boldsymbol{B}v(t) \tag{2.11}$$

$$\boldsymbol{y}(t) = \boldsymbol{F}\boldsymbol{X}(t) + \boldsymbol{D}v(t) \tag{2.12}$$

donde:

$\boldsymbol{X}(t)$ = Vector de variables de estado
$\boldsymbol{A}, \boldsymbol{F}$ = Matrices de coeficientes constantes
$\boldsymbol{B}, \boldsymbol{D}$ = Matrices columna de coeficientes constantes

La ecuación (2.3) puede ser reescrita de la siguiente manera:

$$\ddot{\boldsymbol{y}}(t) + \boldsymbol{C}^{-1}\boldsymbol{G}\dot{\boldsymbol{y}}(t) + \boldsymbol{C}^{-1}\boldsymbol{\Gamma}\boldsymbol{y}(t) = -\boldsymbol{C}^{-1}\boldsymbol{C}_{\boldsymbol{k}}\ddot{v}(t) - \boldsymbol{C}^{-1}\boldsymbol{G}_{\boldsymbol{k}}\dot{v}(t) - \boldsymbol{C}^{-1}\boldsymbol{\Gamma}_{\boldsymbol{k}}v(t) \tag{2.13}$$

Haciendo el desarrollo matemático correspondiente, se obtienen las siguientes ecuaciones de estado que representan a la ecuación (2.13):

$$\dot{\boldsymbol{X}}_{\boldsymbol{1}}(t) = -\boldsymbol{C}^{-1}\boldsymbol{G}\boldsymbol{X}_{\boldsymbol{1}}(t) + \boldsymbol{X}_{\boldsymbol{2}}(t) + \boldsymbol{C}^{-1}(\boldsymbol{G}\boldsymbol{C}^{-1}\boldsymbol{C}_{\boldsymbol{k}} - \boldsymbol{G}_{\boldsymbol{k}})v(t) \tag{2.14}$$

$$\dot{\boldsymbol{X}}_{\boldsymbol{2}}(t) = -\boldsymbol{C}^{-1}\boldsymbol{\Gamma}\boldsymbol{X}_{\boldsymbol{1}}(t) + \boldsymbol{C}^{-1}(\boldsymbol{\Gamma}\boldsymbol{C}^{-1}\boldsymbol{C}_{\boldsymbol{k}} - \boldsymbol{\Gamma}_{\boldsymbol{k}})v(t) \tag{2.15}$$

Expresando las ecuaciones (2.14) y (2.15) en forma matricial:

$$\begin{bmatrix}\dot{\boldsymbol{X}}_{\boldsymbol{1}}(t)\\ \dot{\boldsymbol{X}}_{\boldsymbol{2}}(t)\end{bmatrix} = \begin{bmatrix}-\boldsymbol{C}^{-1}\boldsymbol{G} & \boldsymbol{U}\\ -\boldsymbol{C}^{-1}\boldsymbol{\Gamma} & \boldsymbol{0}\end{bmatrix}\begin{bmatrix}\boldsymbol{X}_{\boldsymbol{1}}(t)\\ \boldsymbol{X}_{\boldsymbol{2}}(t)\end{bmatrix} + \begin{bmatrix}\boldsymbol{C}^{-1}(\boldsymbol{G}\boldsymbol{C}^{-1}\boldsymbol{C}_{\boldsymbol{k}} - \boldsymbol{G}_{\boldsymbol{k}})\\ \boldsymbol{C}^{-1}(\boldsymbol{\Gamma}\boldsymbol{C}^{-1}\boldsymbol{C}_{\boldsymbol{k}} - \boldsymbol{\Gamma}_{\boldsymbol{k}})\end{bmatrix}v(t) \tag{2.16}$$

Además la ecuación de salida está expresada como:

$$\boldsymbol{y}(t) = [\boldsymbol{U} \quad \boldsymbol{0}] \begin{bmatrix} \boldsymbol{X_1}(t) \\ \boldsymbol{X_2}(t) \end{bmatrix} - \boldsymbol{C^{-1}C_k}v(t) \tag{2.17}$$

donde $\boldsymbol{U}$ es la matriz identidad

Comparando las ecuaciones (2.16) y (2.17) con las ecuaciones (2.11) y (2.12) se pueden obtener las matrices $\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{F}$ y $\boldsymbol{D}$:

$$\boldsymbol{A} = \begin{bmatrix} -\boldsymbol{C^{-1}G} & \boldsymbol{U} \\ -\boldsymbol{C^{-1}\Gamma} & \boldsymbol{0} \end{bmatrix} \tag{2.18}$$

$$\boldsymbol{B} = \begin{bmatrix} \boldsymbol{C^{-1}}(\boldsymbol{GC^{-1}C_k} - \boldsymbol{G_k}) \\ \boldsymbol{C^{-1}}(\boldsymbol{\Gamma C^{-1}C_k} - \boldsymbol{\Gamma_k}) \end{bmatrix} \tag{2.19}$$

$$\boldsymbol{F} = [\boldsymbol{U} \quad \boldsymbol{0}] \tag{2.20}$$

$$\boldsymbol{D} = -\boldsymbol{C^{-1}C_k} \tag{2.21}$$

2.2.1.1 Inclusión de pérdidas en serie

El modelo descrito en la sección anterior se replantea para agregar las pérdidas en serie. Utilizando un procedimiento similar al descrito en [32], las ecuaciones de estado quedan como las ecuaciones (2.11) y (2.12) con las matrices de coeficientes definidas de la siguiente forma:

$$\boldsymbol{A} = \begin{bmatrix} \boldsymbol{K_i} & \boldsymbol{K_e} \\ \boldsymbol{P_i} & \boldsymbol{0} \end{bmatrix} \tag{2.22}$$

$$\boldsymbol{B} = \begin{bmatrix} \boldsymbol{K_u} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{P_u} \end{bmatrix} \tag{2.23}$$

$$\boldsymbol{F} = [\boldsymbol{0} \quad \boldsymbol{U}] \tag{2.24}$$

$$\boldsymbol{D} = \begin{bmatrix} \boldsymbol{0} \\ \boldsymbol{0} \end{bmatrix} \tag{2.25}$$

donde:

$$\boldsymbol{K_i} = -\boldsymbol{\Gamma R} \tag{2.26}$$

$$\boldsymbol{K_e} = \boldsymbol{\Gamma_2}(i, j+1) \quad \forall\, i\,;\ 1 \leq j \leq n-1 \tag{2.27}$$

$$\boldsymbol{K_u} = \boldsymbol{\Gamma_2}(i,1) \qquad \forall\, i \tag{2.28}$$

$$\boldsymbol{P_i} = -\boldsymbol{C_4}^{-1}\boldsymbol{Q_a} \tag{2.29}$$

$$\boldsymbol{P_u} = -\boldsymbol{C_4}^{-1}\boldsymbol{C_3} \tag{2.30}$$

además:

$$\boldsymbol{R} = \begin{bmatrix} r_1 & 0 & \dots & 0 \\ 0 & r_2 & 0 & \vdots \\ \vdots & 0 & \ddots & 0 \\ 0 & \dots & 0 & r_n \end{bmatrix} \tag{2.31}$$

$$[\boldsymbol{\Gamma_2}(i,j)] = \begin{cases} \boldsymbol{\Gamma}(i,j) & \forall\, i\,;\ j = 1 \\ \boldsymbol{\Gamma}(i,j) - \boldsymbol{\Gamma}(i,j-1) & \forall\, i\,;\ 1 \le j \le n \end{cases} \tag{2.32}$$

$$\boldsymbol{C_3}(i,1) = \boldsymbol{C_2}(i,1) \quad \forall\, i \tag{2.33}$$

$$\boldsymbol{C_4}(i,1) = \boldsymbol{C_2}(i,j+1) \quad \forall\, i\,;\ 1 \le j \le n-1 \tag{2.34}$$

en lo cual:

$\boldsymbol{C_2}$ = Matriz de capacitancia nodales eliminando la primera columna de la matriz $\boldsymbol{C}$

$\boldsymbol{Q_a}$ = Matriz de incidencias de corriente eliminando la primera columna de la matriz $\boldsymbol{Q_i}$

$$\boldsymbol{Q_i} = \begin{bmatrix} 1 & 0 & \cdots & 0 & 0 \\ -1 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & -1 & 1 \end{bmatrix} \tag{2.35}$$

y la señal de entrada es un vector de la forma:

$$\boldsymbol{v(t)} = \begin{bmatrix} v_1 \\ \dfrac{dv_1}{dt} \end{bmatrix} \tag{2.36}$$

2.2.2 Modelo de parámetros concentrados basado en análisis de redes

La ecuación (2.3) expresada en el dominio de Laplace es la siguiente:

$$s\boldsymbol{C}\boldsymbol{V}(s) + \boldsymbol{G}\boldsymbol{V}(s) + \frac{\boldsymbol{\Gamma}}{s}\boldsymbol{V}(s) = -s\boldsymbol{C_k}U(s) - \boldsymbol{G_k}U(s) - \frac{\boldsymbol{\Gamma_k}}{s}U(s) \tag{2.37}$$

Lo cual puede escribirse en forma compacta como:

$$\boldsymbol{I}(s) = \boldsymbol{Y}(s)\boldsymbol{V}(s) \tag{2.38}$$

donde:

$\boldsymbol{Y}(s)$ = Matriz de admitancias nodales del circuito
$\boldsymbol{I}(s)$ = Vector de corrientes nodales, dados por:

$$\boldsymbol{Y}(s) = s\boldsymbol{C} + \boldsymbol{G} + \frac{\boldsymbol{\Gamma}}{s} \tag{2.39}$$

$$\boldsymbol{I}(s) = -s\boldsymbol{C_k}U(s) - \boldsymbol{G_k}U(s) - \frac{\boldsymbol{\Gamma_k}}{s}U(s) \tag{2.40}$$

La propagación de tensión a lo largo del devanado se calcula resolviendo la ecuación (2.38) para $\boldsymbol{V}(s)$. La respuesta en el tiempo del circuito puede obtenerse por medio de un algoritmo de transformación numérica frecuencia-tiempo, o por un procedimiento de aproximación racional para describir la matriz de admitancias [1].

2.3 MODELOS DE PARÁMETROS DISTRIBUIDOS

El fenómeno de propagación de una onda a lo largo del devanado sólo puede reproducirse correctamente mediante un modelo de parámetros distribuidos. Sin embargo, debido a la naturaleza propia de una línea de transmisión, la representación de inductancias mutuas puede ocasionar dificultades. Esto puede resolverse simulando al devanado como una línea de transmisión multiconductora en donde cada conductor representa una vuelta del devanado y se encuentra conectada en zigzag para preservar continuidad entre vueltas.

2.3.1 Modelo basado en la Teoría de la Línea de Transmisión Monofásica (LTM)

Al considerar el devanado del transformador como una línea de transmisión monofásica, el modelo se puede obtener por medio de las ecuaciones del telegrafista [1], que definidas en el dominio del tiempo son:

$$-\frac{\partial v(x,t)}{\partial x} = Ri(x,t) + L\frac{\partial i(x,t)}{\partial t} \tag{2.41}$$

$$-\frac{\partial i(x,t)}{\partial x} = Gv(x,t) + C\frac{\partial v(x,t)}{\partial t} \tag{2.42}$$

donde $v(x,t)$ e $i(x,t)$ son la tensión y la corriente en un punto x del devanado. Diversas soluciones numéricas para las ecuaciones (2.41) y (2.42) han sido propuestas, entre las que se incluyen técnicas en el dominio del tiempo basadas en el método de Bergeron o en métodos de diferencias finitas, así como su solución en el dominio de la frecuencia, y su posterior transformación numérica al dominio de tiempo.

2.3.1.1 Método de Bergeron para línea monofásica

Una solución exacta del comportamiento de una línea de transmisión ideal bajo transitorios electromagnéticos puede obtenerse por medio del método de Bergeron. Su principal ventaja es que puede ser implementado fácilmente de forma computacional.

Aunque el método de Bergeron es aplicable a líneas con pérdidas, las ecuaciones diferenciales ordinarias que se generan no son integrables de forma directa [33], es decir, no son dependientes de una sola variable. Por lo tanto, considérese una línea de transmisión sin pérdidas con inductancia L y capacitancia C por unidad de longitud. En un punto x a lo largo de la línea la tensión y la corriente están relacionados por:

$$-\frac{\partial v}{\partial x} = L\frac{\partial i}{\partial t} \tag{2.43}$$

$$-\frac{\partial i}{\partial x} = C\frac{\partial v}{\partial t} \tag{2.44}$$

La solución general establecida por D'Alembert es [34], [2]:

$$i(x,t) = f_1(x - vt) + f_2(x + vt) \tag{2.45}$$

$$v(x,t) = Z_0 f_1(x - vt) - Z_0 f_2(x + vt) \tag{2.46}$$

La interpretación física de $f_1(x - vt)$ es la de una onda viajando a una velocidad v en dirección positiva en x y de $f_2(x + vt)$ es de una onda viajando en

una dirección contraria a la primera. El término Z_0 en la ecuación (2.46) es la impedancia característica de la línea de transmisión, dada por

$$Z_0 = \sqrt{\frac{L}{C}} \tag{2.47}$$

mientras que v es su velocidad de fase. Siguiendo el procedimiento establecido en [34], se deduce la ecuación de dos puertos para $i_{k,m}$.

$$\begin{aligned} i_{k,m}(t) &= (1/Z_0)v_k(t) + I_k(t-\tau) \\ i_{m,k}(t) &= (1/Z_0)v_m(t) + I_m(t-\tau) \end{aligned} \tag{2.48}$$

Con las fuentes de corriente equivalentes I_k y I_m, que son conocidas en el tiempo t por medio de la historia en el tiempo $t-\tau$

$$\begin{aligned} I_k(t-\tau) &= -(1/Z_0)v_m(t-\tau) - i_{m,k}(t-\tau) \\ I_m(t-\tau) &= -(1/Z_0)v_m(t-\tau) - i_{k,m}(t-\tau) \end{aligned} \tag{2.49}$$

La Figura 2.6 muestra la red de impedancia correspondiente, la cual describe completamente la línea de transmisión sin pérdidas, en sus terminales. Las terminales no se encuentran conectadas topológicamente; las condiciones en una terminal con respecto a la otra son vistas indirectamente y con un retraso de tiempo τ a través de la fuente equivalente I.

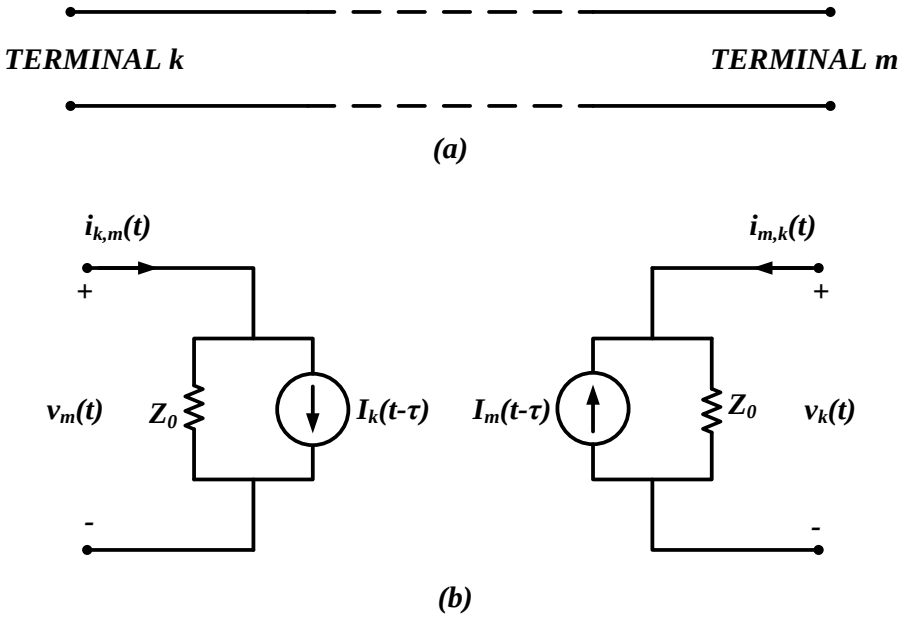


Figura 2.6 (a) Línea sin pérdidas. (b) Red de impedancia equivalente [34].

Para cualquier tipo de red con n nodos, un sistema de n ecuaciones puede ser formado de la manera siguiente:

$$\boldsymbol{Y}\boldsymbol{v}(t) = \boldsymbol{i}(t) - \boldsymbol{I} \tag{2.50}$$

donde:

$\boldsymbol{Y}$ = Matriz de admitancias nodales
$\boldsymbol{e}(t)$ = Vector de tensiones nodales en el tiempo t
$\boldsymbol{i}(t)$ = Vector de corrientes nodales inyectadas en el tiempo t
$\boldsymbol{I}$ = Vector de términos de historia conocidos

La formación de la matriz $\boldsymbol{Y}$ sigue las reglas establecidas para la formación de una matriz de admitancias nodal en el análisis de estado estacionario [34].

En algunas ocasiones, ciertos nodos tendrán tensiones conocidas, ya sea porque se encuentran conectadas fuentes de tensión a ellos o porque se encuentran aterrizados. En este caso, los nodos se subdividen en un conjunto de nodos A con tensiones desconocidas y un conjunto de nodos B con tensiones conocidas. Subdividiendo las matrices y vectores de forma correspondiente, de la ecuación (2.50) se obtiene:

$$\begin{bmatrix} \boldsymbol{Y_{AA}} & \boldsymbol{Y_{AB}} \\ \boldsymbol{Y_{BA}} & \boldsymbol{Y_{BB}} \end{bmatrix} \begin{bmatrix} \boldsymbol{v_A}(t) \\ \boldsymbol{v_B}(t) \end{bmatrix} = \begin{bmatrix} \boldsymbol{i_A}(t) \\ \boldsymbol{i_B}(t) \end{bmatrix} - \begin{bmatrix} \boldsymbol{I_A} \\ \boldsymbol{I_B} \end{bmatrix} \tag{2.51}$$

Las tensiones desconocidas se encuentran resolviendo:

$$\boldsymbol{Y_{AA}}\boldsymbol{v_A}(t) = \boldsymbol{i_A}(t) - \boldsymbol{I_A} - \boldsymbol{Y_{AB}}\boldsymbol{v_B}(t) \tag{2.52}$$

En cada paso de tiempo, los vectores del lado derecho de la ecuación (2.55) se construyen por medio de los términos de historia y las fuentes conocidas de tensión y corriente. Posteriormente, el sistema de ecuaciones lineales se resuelve para $\boldsymbol{v_A}(t)$. Por último se actualizan los términos de historia para su uso en pasos de tiempo posteriores.

Para el caso de la representación del devanado del transformador por medio de una línea de transmisión monofásica como la que se muestra en la **¡Error! No se encuentra el origen de la referencia.** , las variables de la ecuación (2.52), tendrán la siguiente forma

$$\boldsymbol{Y_{AA}} = \begin{bmatrix} Y_0 + 1/R_F & 0 \\ 0 & Y_0 \end{bmatrix};\ \boldsymbol{Y_{AB}} = \begin{bmatrix} -1/R_F \\ 0 \end{bmatrix}$$

$$\boldsymbol{v_A}(t) = \begin{bmatrix} v_k(t) \\ v_m(t) \end{bmatrix};\ \boldsymbol{v_B}(t) = v_F(t) \tag{2.53}$$

$$\boldsymbol{i_A}(t) = \begin{bmatrix} 0 \\ 0 \end{bmatrix};\ \boldsymbol{I_A} = \begin{bmatrix} I_k(t-\tau) \\ I_m(t-\tau) \end{bmatrix}$$

donde:

Y_0 = Admitancia característica de la línea $(1/Z_0)$
$v_F(t)$ = Fuente de alimentación de tensión
R_F = Resistencia interna de la fuente de alimentación

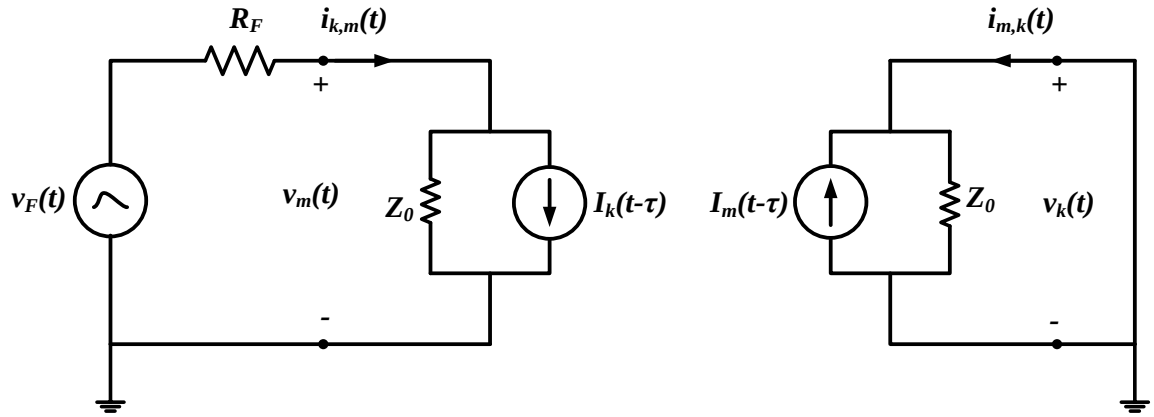


Figura 2.7 Representación del devanado del transformador utilizando la teoría de la línea de transmisión monofásica

2.3.1.2 Inclusión de pérdidas en serie en el método de Bergeron para LTM

Las pérdidas en serie distribuidas a lo largo de una línea de transmisión y con una conductancia en paralelo despreciable $(G = 0)$, pueden modelarse con una exactitud razonable tratando a la línea de transmisión como si se tratase de una línea ideal y agregando las pérdidas en serie de forma concentrada en ambos extremos de la línea. Tales pérdidas en serie concentradas pueden insertarse en muchos lugares a lo largo de la línea, cuando la longitud de ésta se divide en varios segmentos de línea. Se ha encontrado que no existe una diferencia notable entre insertar las pérdidas en serie concentradas en pocos lugares o en muchos, siempre que el valor de las pérdidas en serie sea mucho menor al valor de la impedancia característica de la línea [8].

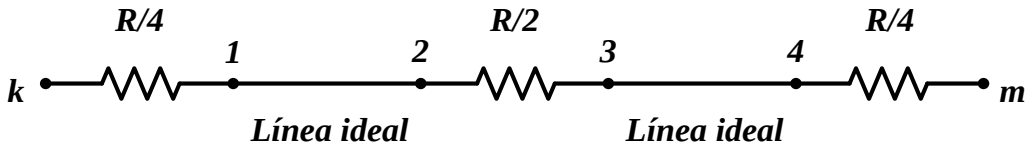


Figura 2.8 Representación de la línea de transmisión con pérdidas en serie concentradas [8].

El modelo utilizado por el programa EMTP concentra las pérdidas en serie en 3 puntos de la línea, de forma específica $R/4$ en ambos extremos y $R/2$ en la parte media como se muestra en la Figura 2.8. Esta aproximación se utiliza debido a que permite conservar las ecuaciones (2.48), excepto que:

$$i_{k,m}(t) = \left(1/Z_{modificada}\right)v_k(t) + I_k(t-\tau)$$

donde:

$$Z_{modificada} = Z_0 + \frac{R}{4} \tag{2.54}$$

R son las pérdidas en serie totales. Los términos de historia ahora contienen condiciones de ambos extremos de la línea en el tiempo $t-\tau$:

$$\begin{aligned} I_k(t-\tau) &= -\left(\frac{Z_0}{Z_{modificada}^2}\right)\left[v_m(t-\tau) + \left(Z_0 - \frac{R}{4}\right)i_{m,k}(t-\tau)\right] \\ &\quad - \frac{R}{4Z_{modificada}^2}\left[v_k(t-\tau) + \left(Z_0 - \frac{R}{4}\right)i_{k,m}(t-\tau)\right] \\ I_m(t-\tau) &= -\left(\frac{Z_0}{Z_{modificada}^2}\right)\left[v_k(t-\tau) + \left(Z_0 - \frac{R}{4}\right)i_{k,m}(t-\tau)\right] \\ &\quad - \frac{R}{4Z_{modificada}^2}\left[v_m(t-\tau) + \left(Z_0 - \frac{R}{4}\right)i_{m,k}(t-\tau)\right] \end{aligned} \tag{2.55}$$

2.3.2 Modelo basado en la Línea de Transmisión Multiconductora (LTMC)

El problema que presenta el modelo anterior radica en la dificultad de tomar en cuenta las inductancias mutuas entre vueltas del devanado, las cuales pueden ser de gran importancia cuando se calculan las tensiones entre vueltas debidas a la aplicación de una onda de frente rápido.

Una forma de sortear este problema es considerar un modelo que represente al devanado del transformador como una línea de transmisión multiconductora, como se muestra en la Figura 2.9 [1].

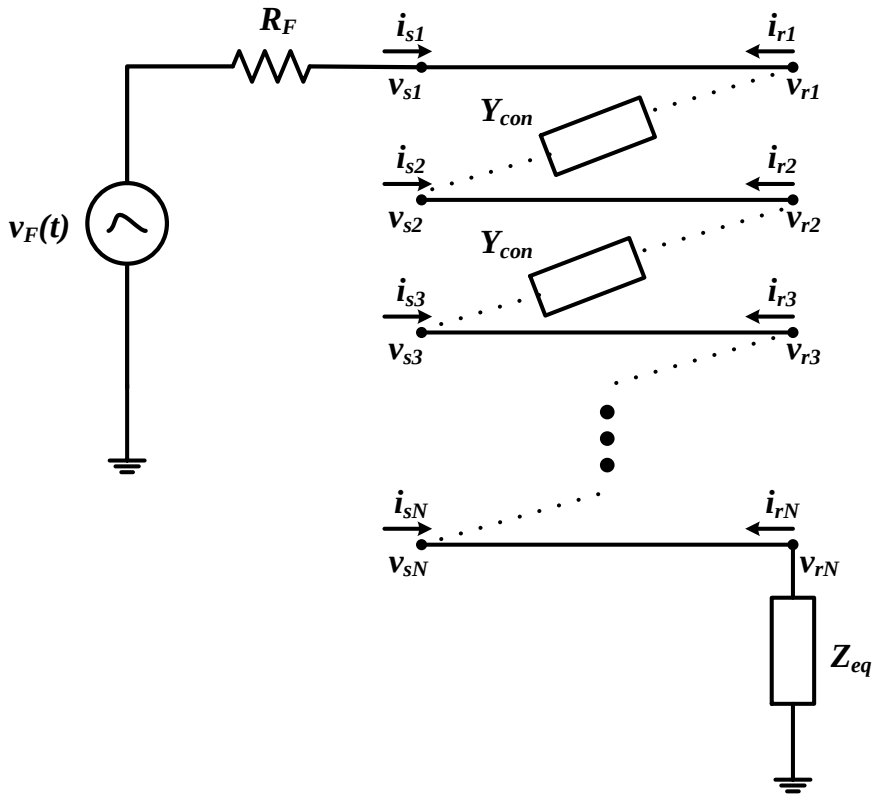


Figura 2.9. Modelo de línea de transmisión multiconductora para el devanado del transformador

En este modelo cada conductor representa una sección del devanado (disco o vuelta). Para conservar la continuidad del devanado, el final de cada conductor se conecta con el principio del siguiente, lo que resulta en una conexión en forma de zigzag. Esto puede definirse mediante el uso de las siguientes ecuaciones:

$$v_{ri} = v_{s(i+1)} \qquad i_{ri} = -i_{s(i+1)}, \qquad i = 1 \ldots n-1 \tag{2.56}$$

El modelo del devanado se obtiene por medio de las ecuaciones del telegrafista para líneas de transmisión multiconductoras en el dominio del tiempo:

$$-\frac{\partial \boldsymbol{v}(x,t)}{\partial x} = \boldsymbol{R}\boldsymbol{i}(x,t) + \boldsymbol{L}\frac{\partial \boldsymbol{i}(x,t)}{\partial t} \tag{2.57}$$

$$-\frac{\partial \boldsymbol{i}(x,t)}{\partial x} = \boldsymbol{G}\boldsymbol{v}(x,t) + \boldsymbol{C}\frac{\partial \boldsymbol{v}(x,t)}{\partial t} \tag{2.58}$$

En la Figura 2.9, la impedancia equivalente Z_{eq} que se encuentra conectada al final del *n*-ésimo conductor puede usarse para representar el resto del devanado cuando se desea modelar en forma detallada sólo una sección del mismo. Z_{eq} también puede representar la impedancia de conexión a tierra. La solución de las ecuaciones que representan este modelo puede obtenerse por medio de los métodos numéricos mencionados para el modelo de línea de transmisión monofásica.

2.3.2.1 Método de Bergeron para LTMC

Las ecuaciones (2.43) y (2.44) también son válidas para líneas multiconductoras si las cantidades escalares se remplazan por vectores $\boldsymbol{v}$, $\boldsymbol{i}$ y matrices $\boldsymbol{L}$, $\boldsymbol{C}$ [34]. Realizando la segunda derivada de dichas ecuaciones, una de las variables puede eliminarse, quedando de la siguiente manera:

$$\frac{\partial^2 \boldsymbol{v}}{\partial x^2} = \boldsymbol{LC}\frac{\partial^2 \boldsymbol{v}}{\partial t^2} \tag{2.59}$$

$$\frac{\partial^2 \boldsymbol{i}}{\partial x^2} = \boldsymbol{CL}\frac{\partial^2 \boldsymbol{i}}{\partial t^2} \tag{2.60}$$

La solución de las ecuaciones (2.59) y (2.60) es complicada debido a la presencia de elementos fuera de la diagonal principal en las matrices, los cuales son causados por los acoplamientos mutuos que existen entre cada una de las fases. Este problema puede simplificarse si las n ecuaciones acopladas se transforman en n ecuaciones desacopladas; de esa manera, las ecuaciones pueden resolverse como si se tratase de n sistemas monofásicos [34], [35], [7].

Para poder realizar el desacoplamiento de las ecuaciones es necesario utilizar transformaciones de similaridad que conviertan las variables en el dominio de fases a variables en el dominio modal. Estas transformaciones producen matrices diagonales derivadas de la teoría de eigenvalores y eigenvectores. Cada una de las ecuaciones independientes en el dominio modal puede ser resuelta con el algoritmo utilizado para la línea de transmisión monofásica utilizando su tiempo de viaje e impedancia característica, ambos en el dominio modal.

Para explicar esta teoría, se define un cambio de variables como:

$$\boldsymbol{v}(x,t) = \boldsymbol{T_v}\boldsymbol{v_m}(x,t) \tag{2.61}$$

$$\boldsymbol{i}(x,t) = \boldsymbol{T_i}\boldsymbol{i_m}(x,t) \tag{2.62}$$

Las matrices complejas $\boldsymbol{T_v}$ y $\boldsymbol{T_i}$, de dimensiones $n \times n$, se conocen como matrices de transformación de similaridad entre el fasor real de tensiones y corrientes, $\boldsymbol{v}$ e $\boldsymbol{i}$, y el modo de tensiones y corrientes, $\boldsymbol{v_m}$ e $\boldsymbol{i_m}$. Para que esto sea válido, las matrices de transformación deben de ser no singulares, es decir, $\boldsymbol{T_v^{-1}}$ y $\boldsymbol{T_i^{-1}}$ deben de existir. Sustituyendo las ecuaciones (2.61) y (2.62) en las ecuaciones (2.57) y (2.58) considerando una línea sin pérdidas, se tiene:

$$\frac{\partial \boldsymbol{v_m}(x,t)}{\partial x} = -\boldsymbol{T_v^{-1} L T_i} \frac{\partial \boldsymbol{i_m}(x,t)}{\partial t} \tag{2.63}$$

$$\frac{\partial \boldsymbol{i_m}(x,t)}{\partial z} = -\boldsymbol{T_i^{-1} C T_v} \frac{\partial \boldsymbol{v_m}(x,t)}{\partial t} \tag{2.64}$$

Si se pudieran elegir unas matrices $\boldsymbol{T_v}$ y $\boldsymbol{T_i}$ de tal forma que las ecuaciones anteriores fueran desacopladas, se tendrían la forma general de la soluciones en forma modal. Si se supone que se encuentra una matriz $\boldsymbol{T_v}$ y una matriz $\boldsymbol{T_i}$ que puedan diagonalizar de forma simultánea las matrices $\boldsymbol{L}$ y $\boldsymbol{C}$ como:

$$\boldsymbol{T_v^{-1} L T_i} = \boldsymbol{L_m} \tag{2.65}$$

$$\boldsymbol{T_i^{-1} C T_v} = \boldsymbol{C_m} \tag{2.66}$$

donde $\boldsymbol{L_m}$ y $\boldsymbol{C_m}$ son matrices diagonales de la forma:

$$\boldsymbol{L_m} = \begin{bmatrix} l_{m1} & 0 & \cdots & 0 \\ 0 & l_{m2} & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & l_{mn} \end{bmatrix} \tag{2.67}$$

$$\boldsymbol{C_m} = \begin{bmatrix} c_{m1} & 0 & \cdots & 0 \\ 0 & c_{m2} & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & c_{mn} \end{bmatrix} \tag{2.68}$$

Entonces las ecuaciones modales (2.63) y (2.64) toman la siguiente forma desacoplada:

$$\frac{\partial \boldsymbol{v_m}(x,t)}{\partial x} = -\boldsymbol{L_m} \frac{\partial \boldsymbol{i_m}(x,t)}{\partial t} \tag{2.69}$$

$$\frac{\partial \boldsymbol{i_m}(x,t)}{\partial z} = -\boldsymbol{C_m} \frac{\partial \boldsymbol{v_m}(x,t)}{\partial t} \tag{2.70}$$

O en forma expandida:

$$\left.\begin{aligned} \frac{\partial v_{m1}(x,t)}{\partial x} &= -l_{m1}\frac{\partial i_{m1}(x,t)}{\partial t} \\ \frac{\partial i_{m1}(x,t)}{\partial x} &= -c_{m1}\frac{\partial v_{m1}(x,t)}{\partial t} \\ &\vdots \\ \frac{\partial v_{mn}(x,t)}{\partial x} &= -l_{mn}\frac{\partial i_{mn}(x,t)}{\partial t} \\ \frac{\partial i_{mn}(x,t)}{\partial x} &= -c_{mn}\frac{\partial v_{mn}(x,t)}{\partial t} \end{aligned}\right\} \tag{2.71}$$

La impedancia característica de cada uno de los modos es de la forma:

$$Z_{Cmi} = \sqrt{\frac{l_{mi}}{c_{mi}}} \tag{2.72}$$

y la velocidad de propagación se calcula de la siguiente manera:

$$v_{mi} = \frac{1}{\sqrt{l_{mi}c_{mi}}} \tag{2.73}$$

Una vez que las ecuaciones de línea han sido resueltas en el dominio modal, es necesario realizar una transformación inversa que permita obtener los resultados en el dominio de fases y que puedan interactuar con el resto de la red. Esto se muestra de forma esquemática en la Figura 2.10.

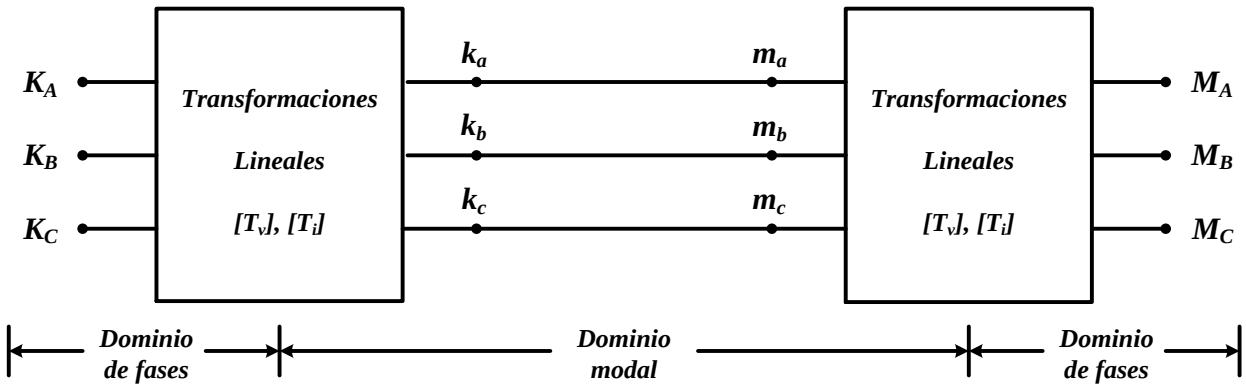


Figura 2.10 Transformación entre el dominio de fases y el dominio modal en una línea trifásica [8].

Por simplicidad, se considera que la línea tiene únicamente tres fases. Con la notación que se muestra en la Figura 2.10, cada uno de los modos está descrito por una ecuación de forma similar a la ecuación (2.48), es decir:

$$\begin{aligned} i_{ka,ma}(t) &= (1/Z_{0a})v_{ka}(t) + I_{ka}(t-\tau_a) \\ i_{kb,mb}(t) &= (1/Z_{0b})v_{kb}(t) + I_{kb}(t-\tau_b) \\ i_{kc,mc}(t) &= (1/Z_{0c})v_{kc}(t) + I_{kc}(t-\tau_c) \end{aligned} \tag{2.74}$$

donde cada término de historia ha sido calculado con anterioridad. Para cada uno de los modos, el término de historia está dado por:

$$\begin{aligned} I_{ka}(t-\tau_a) &= -(1/Z_{0a})v_{ma}(t-\tau_a) - i_{ma,ka}(t-\tau_a) \\ I_{kb}(t-\tau_b) &= -(1/Z_{0b})v_{mb}(t-\tau_b) - i_{mb,kb}(t-\tau_b) \\ I_{kc}(t-\tau_c) &= -(1/Z_{0c})v_{mc}(t-\tau_c) - i_{mc,kc}(t-\tau_c) \end{aligned} \tag{2.75}$$

Los términos de historia se calculan para ambos extremos de la línea tan pronto como la solución ha sido obtenida en el instante t, para utilizarse en un paso de tiempo posterior.

Las ecuaciones (2.74) y (2.75) pueden incluirse en la ecuación nodal (2.50) de forma similar a como se hace para la línea de transmisión monofásica, modificando las dimensiones de cada una de las matrices en forma proporcional al número de líneas de transmisión.

En el caso de la representación del devanado del transformador por medio de una línea de transmisión multiconductora como se muestra en la Figura 2.9, las variables de la ecuación (2.52), tendrán la siguiente forma:

$$\boldsymbol{Y_{AA}} = \begin{bmatrix} \boldsymbol{Y_k} & \boldsymbol{Y_c} \\ \boldsymbol{Y_c^t} & \boldsymbol{Y_m} \end{bmatrix};\ \boldsymbol{Y_k} = \begin{bmatrix} Y_{01}+1/R_F & 0 & \cdots & 0 \\ 0 & Y_{02}+Y_{con} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & Y_{0n}+Y_{con} \end{bmatrix}$$

$$\boldsymbol{Y_c} = \begin{bmatrix} 0 & 0 & 0 & \cdots & 0 \\ -Y_{con} & 0 & 0 & \cdots & 0 \\ 0 & -Y_{con} & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & -Y_{con} & \cdots & 0 \end{bmatrix} \tag{2.76}$$

$$\boldsymbol{Y_m} = \begin{bmatrix} Y_{01} + Y_{con} & 0 & \cdots & 0 \\ 0 & Y_{02} + Y_{con} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & Y_{0n} + Y_{fin} \end{bmatrix}$$

$$\boldsymbol{Y_{AB}} = \begin{bmatrix} -1/R_F \\ 0 \\ \vdots \\ 0 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}; \ \boldsymbol{v_A}(t) = \begin{bmatrix} v_{k1}(t) \\ v_{k2}(t) \\ \vdots \\ v_{kn}(t) \\ v_{m1}(t) \\ v_{m2}(t) \\ \vdots \\ v_{mn}(t) \end{bmatrix}$$

$$\boldsymbol{v_B}(t) = v_F(t)$$

$$\boldsymbol{i_A}(t) = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}; \ \boldsymbol{I_A} = \begin{bmatrix} I_{k1}(t - \tau_{k1}) \\ I_{k2}(t - \tau_{k2}) \\ \vdots \\ I_{kn}(t - \tau_{kn}) \\ I_{m1}(t - \tau_{m1}) \\ I_{m2}(t - \tau_{m2}) \\ \vdots \\ I_{mn}(t - \tau_{mn}) \end{bmatrix}$$

donde:

Y_{con}	=	Admitancia de conexión entre el extremo final de una vuelta y el inicio de la siguiente
Y_{fin}	=	Admitancia que representa la conexión al final del devanado $(1/Z_{eq})$
$v_F(t)$	=	Fuente de alimentación de tensión
R_F	=	Resistencia interna de la fuente de alimentación

La inclusión de pérdidas en serie en forma concentrada para una línea de transmisión multiconductora se realiza de forma similar al caso monofásico; es decir, las ecuaciones (2.54) y (2.55) deberán utilizarse para cada uno de los n modos involucrados en el fenómeno.

2.3.3 Modelo basado en la combinación de las teorías de la LTM y LTMC

Para el caso de transitorios electromagnéticos muy rápidos, todas las vueltas y bobinas del devanado del transformador deben considerarse en el estudio. En

consecuencia, un modelo basado en la teoría de la línea multiconductora resultaría en un costo computacional muy grande debido a las operaciones matriciales que conlleva. Este problema puede solucionarse realizando una combinación de los modelos basados en las teorías de LTM y LTMC, como se muestra en la Figura 2.11. El procedimiento se describe a continuación: Primero, cada bobina se representa por un modelo de LTM, obteniéndose las tensiones en los extremos de cada bobina. Después, cada bobina se representa por medio de un modelo de LTMC para calcular la distribución de tensión en cada una de las vueltas independientemente de las otras bobinas y usando las tensiones obtenidos en el paso anterior como entradas. Debido a que normalmente las primeras bobinas son expuestas al mayor esfuerzo dieléctrico, el modelo de la LTMC puede ser utilizado únicamente para estas bobinas, de tal forma que la distribución de tensión pueda ser estudiada a detalle [1].

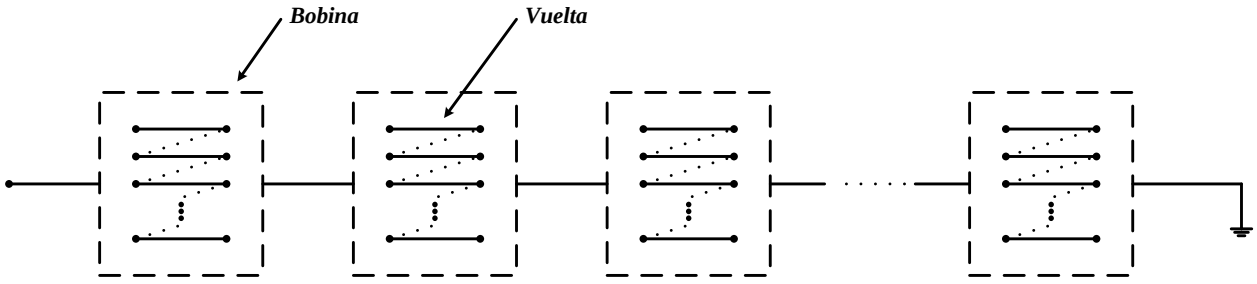


Figura 2.11 Modelo del devanado del transformador basado en la combinación de las teorías de la LTM y LTMC [1].

En el presente trabajo se eligieron dos modelos a implementar: uno de parámetros concentrados basado en ecuaciones de estado y otro de parámetros distribuidos basado en el la teoría de la línea multiconductora, ambos en el dominio del tiempo.

La razón de que ambos se encuentren en el dominio del tiempo es debido a la necesidad de obtener las tensiones transitorias de cada vuelta del devanado del transformador de forma directa; contrario a lo que sucedería al utilizar modelos en el dominio de la frecuencia, en los que se debe utilizar un algoritmo de transformación numérica para obtener las tensiones transitorias en el dominio del tiempo.

Se seleccionaron específicamente estos dos modelos puesto que se considera que existe suficiente información acerca de ellos y de la forma en que se les incluyen las pérdidas en serie.

CAPÍTULO 3. DETERMINACIÓN DE PARÁMETROS PARA TRANSITORIOS ELECTROMAGNÉTICOS RÁPIDOS

En este capítulo se presentan algunos de los métodos de cálculo de parámetros para modelos de transformador de alta frecuencia. Además, se explica un algoritmo alternativo de cálculo de capacitancia basado en el método de imágenes.

3.1 INTRODUCCIÓN

El cálculo de parámetros para modelos del transformador para transitorios electromagnéticos de alta frecuencia se encuentra basado en aproximaciones, entre las cuales se pueden distinguir básicamente tres metodologías [1]:

- Aplicación de fórmulas: Ecuaciones derivadas de configuraciones geométricas simplificadas o ecuaciones empíricas obtenidas por medio de mediciones.

- Determinación experimental por medio de pruebas de laboratorio.

- Simulaciones de campos electromagnéticos utilizando métodos numéricos, siendo el Método de Elemento Finito (MEF) el más común.

Un problema importante en la determinación de parámetros de los modelos del transformador para transitorios de alta frecuencia es que se requiere de información muy detallada de la configuración geométrica interna del transformador, la cual generalmente sólo se encuentra disponible para los fabricantes.

Sin importar el modelo empleado para la simulación, los elementos inductivos, capacitivos y las componentes de pérdidas son los parámetros más importantes para la correcta descripción del comportamiento del devanado del transformador cuando se somete a fenómenos transitorios de alta frecuencia.

3.2 CAPACITANCIA

Para poder hacer una estimación correcta de la distribución de tensión dentro del devanado de un transformador cuando se encuentra bajo el efecto de un

fenómeno transitorio de alta frecuencia, es necesario conocer los valores de capacitancia en serie y capacitancia a tierra [30].

En el caso de un modelo de parámetros concentrados, el devanado del transformador se divide en segmentos (vueltas o grupos de vueltas). Cada uno de estos segmentos contiene un nodo de entrada y un nodo de salida. Entre estos dos nodos se encuentra un elemento capacitivo denominado capacitancia en serie C_s.

De forma adicional, cada segmento del devanado del transformador se encontrará asociado con un elemento capacitivo entre grupos de vueltas y tierra o, grupos de vueltas y un blindaje; éste se denomina comúnmente como capacitancia a tierra C_g, y en la mayoría de los casos se divide en 2 partes, cada una de las cuales se conecta en cada extremo del segmento correspondiente.

Debido a que los segmentos en los que se divide el devanado del transformador tienen dimensiones radiales y axiales pequeñas y radios grandes, puede utilizarse la fórmula de placas paralelas para calcular tanto las capacitancias en serie como las capacitancias a tierra [36]:

$$C_s = \frac{\varepsilon_0 \varepsilon_r h}{d_s} \tag{3.1}$$

donde:

ε_0 = Permitividad del espacio libre
ε_r = Permitividad relativa del material dieléctrico entre vueltas
h = Altura del conductor rectangular
d_s = Distancia entre vueltas

$$C_g = \frac{\varepsilon_0 \varepsilon_r w}{d_g} \tag{3.2}$$

donde:

w = Ancho del conductor rectangular
d_g = Distancia entre una vuelta y el plano de tierra

En un capacitor de placas paralelas, el campo eléctrico no se encuentra contenido completamente entre ellas debido a que cerca de los bordes de las placas las líneas del campo eléctrico no son rectas. Las ecuaciones (3.1) y (3.2)

no toman en cuenta este fenómeno conocido como efecto borde, por lo que se han obtenido expresiones para realizar el cálculo de capacitancias entre vueltas y entre discos que tomen en consideración este aspecto [29]:

$$C_{ev} = \frac{\varepsilon_0 \varepsilon_{ev}(w + d_{ev})}{d_{ev}} \tag{3.3}$$

$$C_{ed} = \varepsilon_0 \left(\frac{k}{d_{ev}/\varepsilon_{ev} + d_{ed}/\varepsilon_{ac}} + \frac{1-k}{d_{ev}/\varepsilon_{ev} + d_{ed}/\varepsilon_{ed}} \right) (R + d_{ed}) \tag{3.4}$$

donde:

C_{ev} = Capacitancia entre vueltas
C_{ed} = Capacitancia entre discos
ε_{ev} = Permitividad relativa del aislamiento entre vueltas
ε_{ed} = Permitividad relativa del aislamiento entre discos
ε_{ac} = Permitividad relativa del aceite aislante
d_{ev} = Distancia entre vueltas
d_{ed} = Distancia entre discos
k = Fracción del espacio circunferencial ocupado por el aceite
R_d = Profundidad radial del devanado

Un cálculo más preciso de la capacitancia del devanado del transformador puede obtenerse utilizando un programa de simulación de campos electromagnéticos basado en el MEF [37], [38], [39]. Considerando N vueltas del devanado, el potencial del j-ésimo segmento se ajusta a un valor de $U = 1\,V$, mientras que el potencial de las vueltas restantes se fija en cero. La energía electrostática cuando el j-ésimo segmento se encuentra energizado está definida como:

$$W_j = \frac{1}{2} \sum_{i=1}^{N} C_{ij} \Delta U_{ij}^2 \tag{3.5}$$

donde:

C_{ij} = Capacitancia entre las vueltas i y j (no es un elemento de la matriz $\boldsymbol{C}$)
ΔU_{ij} = Diferencia de potencial entre dichas vueltas
W_j = Energía electrostática obtenida por medio del MEF

La matriz de capacitancia se forma de manera similar a una matriz de admitancia nodal: los elementos de la diagonal principal están dados por la suma

de todas las capacitancias que convergen a ese nodo, mientras que los elementos fuera de la diagonal principal se encuentran dados por la capacitancia mutua existente entre los nodos i y j.

La principal desventaja de utilizar esta herramienta es la gran cantidad de tiempo que puede llegar a consumir al generar el modelo y realizar las simulaciones necesarias; esto se incrementa con la complejidad de la geometría [1].

Una alternativa para el cálculo de las capacitancias es utilizar un algoritmo basado en el método de imágenes, el cual ha sido empleado para calcular capacitancias en líneas de transmisión utilizando conductores simétricos o imágenes que ocupan una posición equipotencial desde un plano de interés [40].

El método de imágenes establece que una configuración de carga dada sobre un plano conductor perfecto e infinito conectado a tierra puede reemplazarse por la propia configuración de carga, su imagen y una superficie equipotencial en sustitución del plano conductor [41].

Como se muestra en la Figura 3.1(a), el análisis se desarrolla dentro de una ventana (estructura metálica) que posee un conductor a de carga +Q. Para el análisis del método de imágenes se coloca una imagen b de carga –Q. Para obtener la capacitancia es necesario calcular el potencial eléctrico mediante la integración del campo eléctrico a lo largo de la línea horizontal que une a los conductores y el núcleo [42].

La capacitancia mutua entre los conductores a y b puede ser obtenida integrando las componentes del campo eléctrico del conductor a y su imagen b a lo largo del contorno correspondiente al conductor a'. (Figura 3.1(b)).

Para el cálculo de capacitancia propia, cada una de las cuatro paredes de la ventana se sustituye por su respectiva imagen. Las imágenes que se encuentran en las esquinas contienen carga eléctrica positiva, con el fin de representar de mejor forma el campo eléctrico en esa zona [42].

El potencial eléctrico a lo largo de una trayectoria se calcula de acuerdo con la siguiente ecuación:

$$V = -\int \boldsymbol{E} \cdot \boldsymbol{dl} \tag{3.6}$$

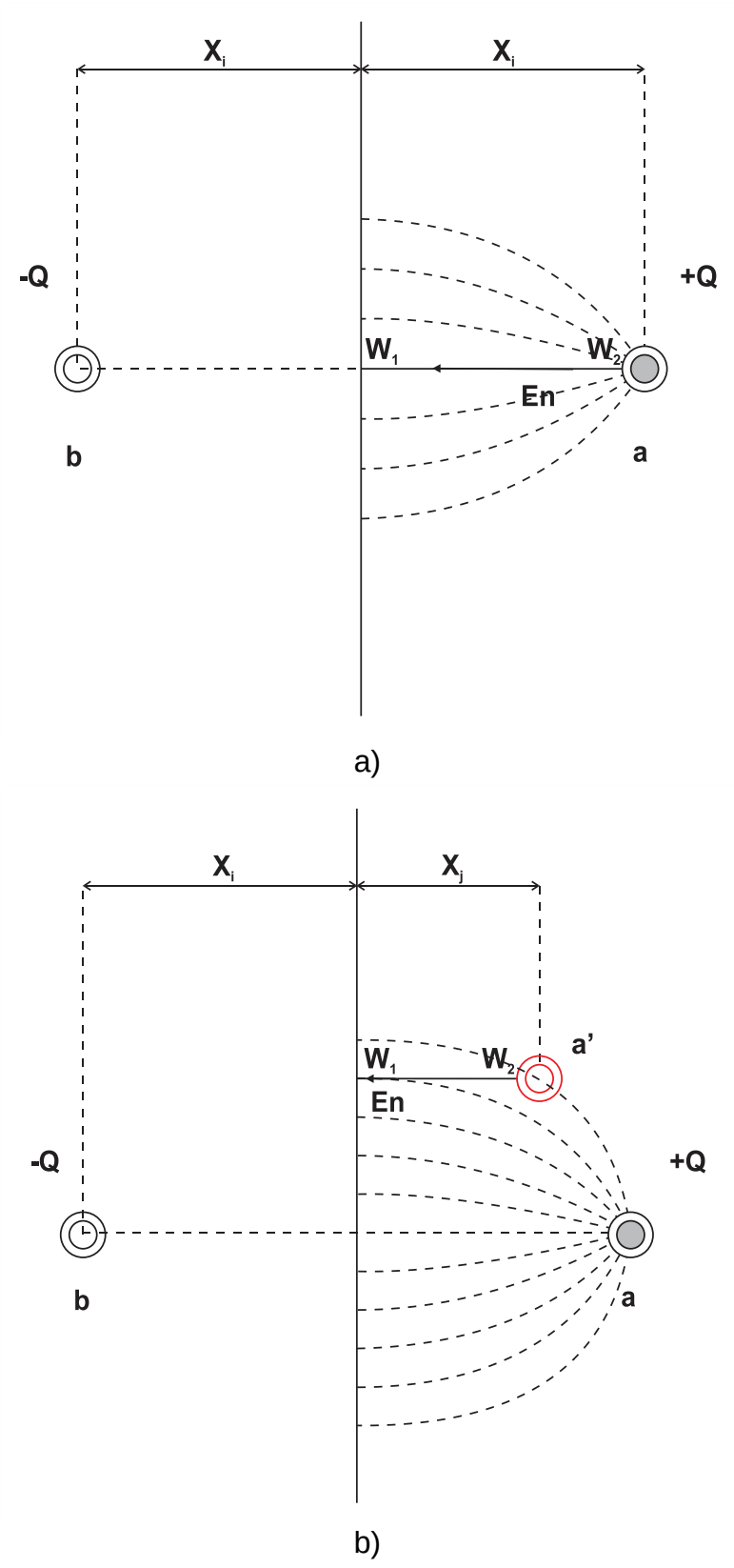


Figura 3.1 Método de imágenes para calcular (a) capacitancia propia y (b) capacitancia mutua

El campo eléctrico normal a la pared de la ventana y debido a un conductor circular de radio r se obtiene con la ecuación:

$$\boldsymbol{E_n} = \frac{Q}{2\pi\varepsilon_0 r}\boldsymbol{a_n} \tag{3.7}$$

Sustituyendo la ecuación (3.7) en (3.6) y aplicando los límites de integración necesarios se obtiene el potencial eléctrico debido a los conductores con carga eléctrica positiva, dado por:

$$V^+ = -\frac{Q}{2\pi\varepsilon_0}\ln\left(\frac{r_{W1}}{r_{W2}}\right) \tag{3.8}$$

donde:

$$\begin{aligned} r_{W1} &= \sqrt{(d_{xW1})^2 + \left(d_{yW1}\right)^2} \\ r_{W2} &= \sqrt{(d_{xW2})^2 + \left(d_{yW2}\right)^2} \end{aligned} \tag{3.9}$$

d_{xW1} = Distancia en x del conductor al punto W1
d_{yW1} = Distancia en y del conductor al punto W1
d_{xW2} = Distancia en x del conductor al punto W2
d_{yW2} = Distancia en y del conductor al punto W2

El potencial eléctrico debido a los conductores con carga negativa se obtiene de forma análoga:

$$V^- = \frac{Q}{2\pi\varepsilon_0}\ln\left(\frac{r_{W2}}{r_{W1}}\right) \tag{3.10}$$

El potencial eléctrico total a lo largo de la línea que une los puntos *W*1 y *W*2 será igual a la contribución de los conductores con carga positiva más la contribución de los conductores con carga negativa, teniéndose:

$$V_T = -\frac{Q}{2\pi\varepsilon_0}\sum\ln\left(\frac{r_{W2}}{r_{W1}}\right)^+ + \frac{Q}{2\pi\varepsilon_0}\sum\ln\left(\frac{r_{W2}}{r_{W1}}\right)^- \tag{3.11}$$

La capacitancia propia por unidad de longitud es:

$$C = \frac{Q}{V} = \frac{2\pi\varepsilon_0}{\left[\sum \ln\left(\frac{r_{W2}}{r_{W1}}\right)^{-} - \sum \ln\left(\frac{r_{W2}}{r_{W1}}\right)^{+}\right]} \tag{3.12}$$

Para el caso de capacitancia mutua, se realiza un procedimiento similar al caso de capacitancia propia, sustituyendo las cuatro paredes de la ventana por el respectivo sistema de imágenes que deformarán el campo eléctrico de manera equivalente a la ventana, como se muestra en la Figura 3.2 [42]. Sin embargo, se debe considerar el potencial eléctrico debido al conductor a (Figura 3.3)

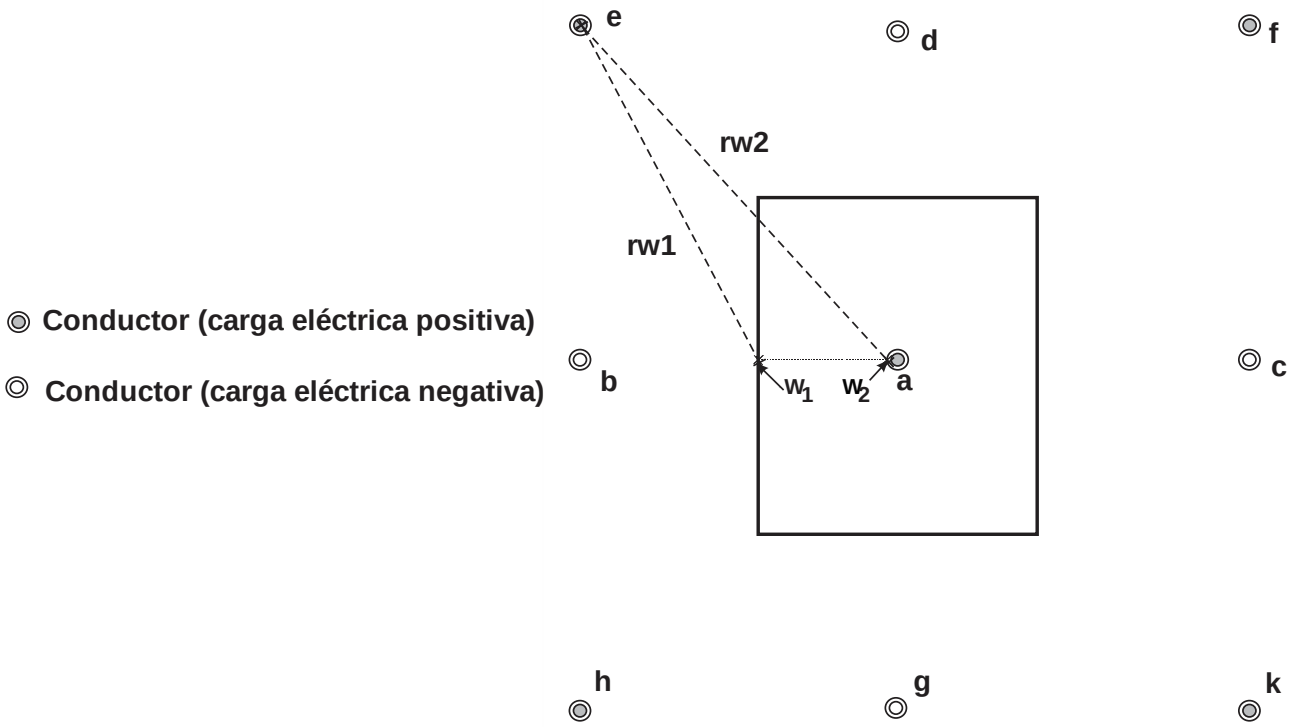


Figura 3.2 Sistema de imágenes para e cálculo de capacitancia propia

Las ecuaciones utilizadas para el cálculo de capacitancia propia se utilizan de igual forma para realizar el cálculo de capacitancia mutua, realizando las modificaciones correspondientes a las distancias debidas al nuevo contorno de integración.

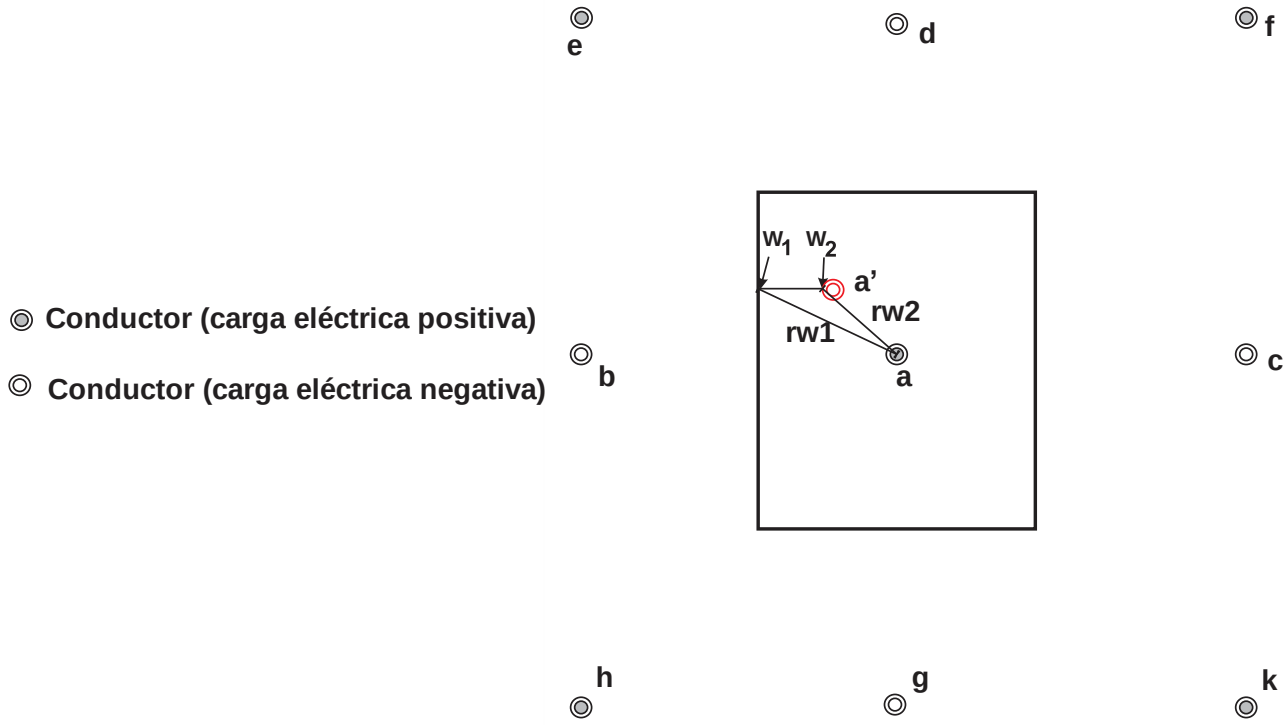


Figura 3.3 Sistema de imágenes para el cálculo de capacitancia mutua

3.3 INDUCTANCIA

Para calcular la inductancia en un devanado con núcleo de hierro para el análisis de transitorios rápidos, generalmente se realiza la suposición de que el flujo magnético se encuentra confinado al espacio de aire, debido a que el tiempo requerido para que el flujo magnético penetre el material ferromagnético es mayor al tiempo de duración del transitorio [7]. Por lo tanto, es práctica común reemplazar al núcleo de hierro por un núcleo de aire en el análisis de transitorios rápidos. Sin embargo, recientemente se ha demostrado que esta suposición introduce errores en el cálculo, ya que debido a las corrientes de eddy las paredes internas del núcleo se comportan como una barrera contra el flujo magnético en altas frecuencias, lo cual no es equivalente a considerar un núcleo de aire [42].

Antes de la introducción de la computadora, no existían muchas fórmulas analíticas para calcular las inductancias propias y mutuas de bobinas con núcleo de hierro [36]. Varios autores han planteado diversos métodos para realizar el cálculo de la inductancia de una bobina, algunos de los cuales se mencionan a continuación.

Una expresión para las inductancias propias y mutuas de filamentos circulares fue desarrollada por Maxwell. Para la Figura 3.4, la inductancia mutua entre los filamentos A y B está dada por [1], [43], [14], [44]:

$$L_{AB} = \mu_0 (R_1 R_2)^{1/2} \left[\left(\frac{2}{k} - k \right) K(k) - \frac{2}{k} E(k) \right] \tag{3.13}$$

donde:

$$k = \left[\frac{4R_1 R_2}{(R_1 + R_2)^2 + d^2} \right]^{1/2} \tag{3.14}$$

μ_0 es la permeabilidad del vacío y $K(k)$ y $E(k)$ son las integrales elípticas completas de primer y segundo orden, respectivamente.

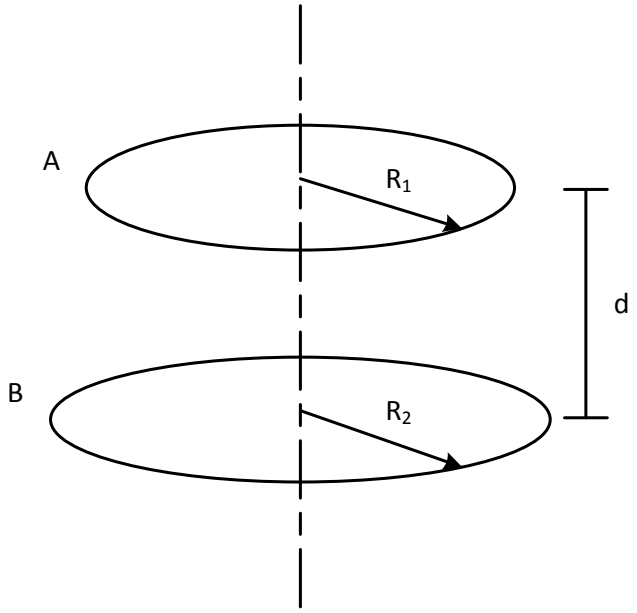


Figura 3.4 Inductancia mutua entre dos filamentos circulares concéntricos [14]

Para el caso de bobinas circulares con sección transversal rectangular, el método de Lyle [43], [44] consiste en reemplazar cada bobina por dos filamentos equivalentes, como se muestra en la Figura 3.5.

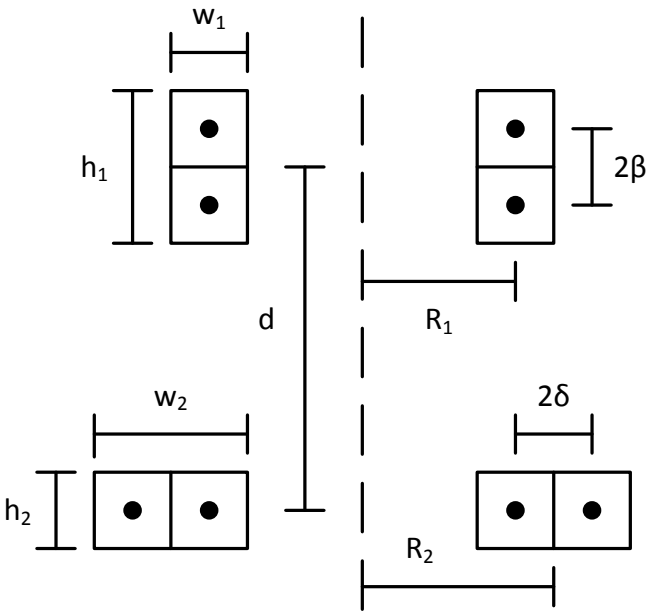


Figura 3.5 Representación de bobinas de dimensiones finitas por el método de Lyle [1]

Para $b > c$, se reemplaza la bobina con dos filamentos 11' y 22' de radios

$$r_1 = R_1\left(1 + \frac{w_1^2}{24R_1^2}\right) \tag{3.15}$$

Ambos filamentos se encuentran espaciados a cada lado del plano medio una distancia β, dada por:

$$\beta = \sqrt{\frac{h_1^2 - w_1^2}{12}} \tag{3.16}$$

Para el caso $c > b$, el método de Lyle reemplaza la bobina con dos filamentos 33' y 44' que yacen en el plano medio con radios $(r_2 + \delta)$y $(r_2 - \delta)$, donde:

$$r_2 = R_2\left(1 + \frac{h_2^2}{24R_2^2}\right) \tag{3.17}$$

$$\delta = \sqrt{\frac{w_2^2 - h_2^2}{12}} \tag{3.18}$$

La inductancia mutua de las bobinas se encuentra dada por:

$$L_{AB} = \frac{L_{13} + L_{14} + L_{23} + L_{24}}{4} \tag{3.19}$$

con cada término de la ecuación (3.19) calculado de acuerdo con la ecuación (3.13).

La inductancia propia de una bobina circular de una sola vuelta y sección transversal cuadrada, con un radio promedio de a y longitud de uno de sus lados c se encuentra dada como [1]:

$$L_{AA} = \mu_0 a \left[\frac{1}{2}\left(1 + \frac{1}{6}\left(\frac{c}{2a}\right)^2\right) ln\left(\frac{8}{(c/2a)^2}\right) - 0.84834 + 0.2041\left(\frac{c}{2a}\right)^2\right] \tag{3.20}$$

La ecuación (3.20) es aplicable para secciones transversales pequeñas, tal que $(c/2a) < 0.2$. Si la sección transversal no es cuadrada, debe dividirse en un número de secciones transversales cuadradas y utilizar en conjunto las ecuaciones (3.13) y (3.20) para calcular el valor de la inductancia propia.

Para una bobina tipo disco como la que se muestra en la Figura 3.6, la inductancia propia puede calcularse por la siguiente ecuación propuesta por Gray [45], [44]:

$$L_{AA} = \mu_0 R N^2 \left[ln\left(\frac{8R}{DMG}\right) - 2\right] \tag{3.21}$$

donde:

$$\begin{aligned} ln(DMG) = \frac{1}{2} ln(a^2 + b^2) - \frac{b^2}{12a^2} ln\left(1 + \frac{a^2}{b^2}\right) - \frac{a^2}{12b^2} ln\left(1 + \frac{b^2}{a^2}\right) \\ + \frac{2b}{3a} tan^{-1}\left(\frac{a}{b}\right) + \frac{2a}{3b} tan^{-1}\left(\frac{b}{a}\right) - \frac{25}{12} \end{aligned} \tag{3.22}$$

N es el número de vueltas.

Una forma alternativa para realizar el cálculo de la inductancia, basada en la teoría de la línea multiconductora, es definiendo una matriz de inductancia por unidad de longitud, dividida en una matriz de inductancia geométrica $\boldsymbol{L_g}$ y una matriz de inductancia de los conductores $\boldsymbol{L_c}$, tal que:

$$\boldsymbol{L} = \boldsymbol{L_g} + \boldsymbol{L_c} \tag{3.23}$$

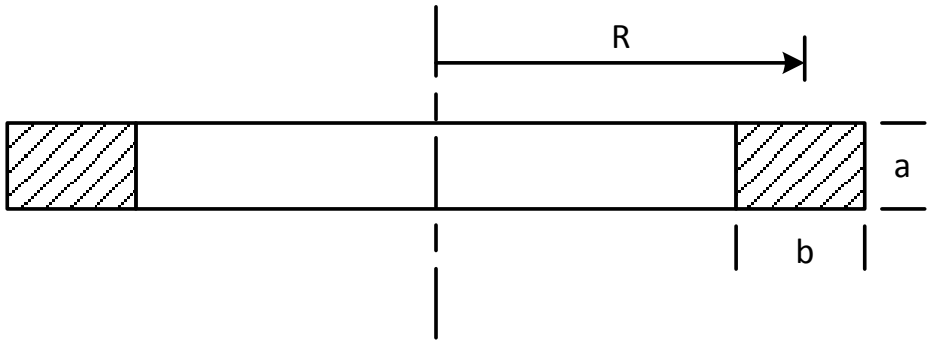


Figura 3.6 Bobina tipo disco con sección transversal rectangular [14]

La matriz de inductancia geométrica puede obtenerse con facilidad directamente de la matriz de capacitancia haciendo uso de la siguiente ecuación:

$$\boldsymbol{L_g} = \frac{\varepsilon_r}{c^2}\boldsymbol{C^{-1}} \tag{3.24}$$

donde:

ε_r = Permitividad relativa del material dieléctrico
c = Velocidad de la luz en el espacio libre

La matriz de inductancia de los conductores se calcula de la siguiente forma [1]:

$$\boldsymbol{L_c} = \frac{Im(Z_c)}{\omega}\boldsymbol{U} \tag{3.25}$$

donde:

Z_c = Impedancia del conductor debida al efecto piel
ω = Frecuencia angular

De manera similar a la matriz de capacitancia, la matriz de inductancia puede calcularse por medio del MEF. Utilizando el método de la energía, la inductancia propia L_{ii} puede calcularse por medio de la energía magnética obtenida al aplicar una corriente I_i a la i-ésima sección (vuelta o grupo de vueltas) del devanado:

$$W_{mag,i} = \frac{1}{2} L_{ii} I_i^2 \tag{3.26}$$

La inductancia mutua L_{ij} se calcula por medio de la energía magnética obtenida al aplicar corriente a ambos elementos i y j:

$$W_{mag,ij} = L_{ij} I_i I_j + \frac{1}{2}\left(L_{ii} I_i^2 + L_{jj} I_j^2\right) \tag{3.27}$$

Los elementos de la diagonal principal deben ser calculados primeramente por medio de la ecuación (3.26), para después obtener los elementos fuera de la diagonal principal por medio de la ecuación (3.27).

De manera similar al cálculo de capacitancia, para arreglos de devanados reales de transformadores de mayor tamaño, realizar el cálculo de la matriz de inductancia por medio de un programa de simulación de campos electromagnéticos tendría un alto costo computacional; por ello, es preferible utilizar un algoritmo de cálculo de la matriz de inductancia basado en el método de imágenes, tal como el que se menciona en [42].

3.4 PÉRDIDAS

En estado estacionario, las pérdidas son características indeseables y costosas para los sistemas físicos. Sin embargo, para fenómenos de alta frecuencia, las pérdidas producen un efecto positivo en el transformador al reducir la magnitud de las oscilaciones de tensión transitoria [36].

3.4.1 PÉRDIDAS EN LOS CONDUCTORES

Las pérdidas causadas por la circulación de corriente en los conductores de los devanados del transformador tienen tres componentes: resistencia de corriente directa, efecto piel y efecto de proximidad.

3.4.1.1 Resistencia de corriente directa

La resistencia de corriente directa de un conductor por unidad de longitud está dada por:

$$R_{dc} = \frac{\rho}{A} \tag{3.28}$$

donde:

ρ = Resistividad del conductor
A = Área de la sección transversal del conductor

La variable ρ se encuentra en función del material del conductor y de su temperatura.

3.4.1.2 Efecto piel y efecto de proximidad

Las pérdidas en los conductores debidas al efecto piel y al efecto de proximidad pueden ser calculadas por medio de la siguiente expresión [46]:

$$\boldsymbol{R} = \frac{1}{d}\sqrt{\frac{2\omega}{\sigma_c \mu_c}}\boldsymbol{L} \tag{3.29}$$

donde:

d = Distancia entre capas
ω = Frecuencia
σ_c = Conductividad del conductor del devanado
μ_c = Permeabilidad del conductor del devanado

Una forma alternativa de realizar el cálculo de las pérdidas en los conductores tomando en cuenta únicamente el efecto piel, y considerando una sección transversal rectangular se calcula por medio de la resistencia de corriente directa R_{cd} y la impedancia a alta frecuencia Z_{hf}:

$$Z_c = \sqrt{R_{cd}^2 + Z_{hf}^2} \tag{3.30}$$

donde:

$$Z_{hf} = \frac{\rho_c}{2p(w+h)} \tag{3.31}$$

ρ_c = Resistividad del material del conductor
w = Anchura del conductor
h = Altura del conductor
p = Profundidad de penetración compleja debida al efecto piel, definida como:

$$p = \sqrt{\frac{\rho_c}{j\omega\mu_c}} \tag{3.32}$$

donde μ_c es la permeabilidad del material del conductor

3.4.2 PÉRDIDAS EN EL NÚCLEO

Las pérdidas en el núcleo debido a corrientes eddy pueden representarse de la siguiente forma [1]:

$$Z_{nucleo} = \frac{4N^2A}{ld^2\sigma} x tanh(x) \tag{3.33}$$

donde:

$$x = \frac{d\sqrt{j\omega\mu\sigma}}{2} \tag{3.34}$$

l = Longitud de la trayectoria magnética
d_l = Anchura de la laminación
μ = Permeabilidad del material
N = Número de vueltas en la bobina
A = Área total de la sección transversal de todas las laminaciones
ω = Frecuencia

Esta fórmula representa la impedancia equivalente de una bobina arrollada alrededor de un núcleo de hierro laminado; fue obtenida al resolver las ecuaciones de Maxwell suponiendo que la distribución del campo electromagnético es idéntica en todas las laminaciones.

3.4.3 PÉRDIDAS CAPACITIVAS

Las pérdidas capacitivas en el material aislante pueden calcularse directamente de la matriz de capacitancia haciendo uso del factor de pérdidas $\tan(\delta)$, y se encuentran definidas en términos de una matriz de conductancia:

$$\boldsymbol{G} = \omega \tan(\delta) \boldsymbol{C} \tag{3.35}$$

De la Figura 2.1, G_{ii} corresponde a la suma de los elementos $1/R_s$ y $1/R_g$ que convergen en el nodo i, mientras que G_{ij} se encuentra dado por el elemento $1/R_s$ conectado entre los nodos i y j con signo negativo. Puede observarse en la ecuación (3.35) que este elemento es una función lineal de la frecuencia.

Aunque en el presente trabajo los parámetros eléctricos se introducen directamente en los modelos implementados, se pretende que el método de imágenes sea incluído posteriormente como una opción de cálculo de parámetros.

CAPÍTULO 4. IMPLEMENTACIÓN DE MODELOS DE TRANSFORMADOR EN MATLAB-SIMULINK®

En este capítulo se explica de forma detallada el proceso realizado para la implementación de modelos de transformador de parámetros concentrados y parámetros distribuidos como parte de una biblioteca dentro del programa de simulación MATLAB-Simulink®.

4.1 INTRODUCCIÓN

Se eligió el programa de simulación MATLAB-Simulink® para realizar la implementación de los modelos debido a la facilidad que presenta este lenguaje para la programación de los mismos y contar además con las herramientas necesarias para la creación de modelo propios.

4.2 IMPLEMENTACIÓN DEL MODELO DEL TRANSFORMADOR DE PARÁMETROS CONCENTRADOS BASADO EN ECUACIONES DE ESTADO

Dentro de las opciones que provee MATLAB-Simulink® para expandir sus capacidades de modelado y simulación, existe aquella que permite al usuario desarrollar bloques personalizados. Está opción tiene como finalidad proporcionar una alternativa para el modelado de fenómenos para los cuales MATLAB-Simulink® no contiene un modelo propio.

De forma general, para diseñar un bloque personalizado se debe seguir el siguiente proceso [47]:

1. Definir el comportamiento requerido para el bloque personalizado.
2. Decidir el tipo de bloque personalizado a usar.
3. Determinar si el bloque residirá en una biblioteca.
4. Agregar una interfaz gráfica del usuario con el bloque.

4.2.1 Definiendo el comportamiento del bloque personalizado

Primero se definen las características y limitaciones del bloque personalizado. Para el caso del modelo del transformador de parámetros concentrados, el bloque debe tener las siguientes características:

- Ingresar los parámetros del modelo por medio de una variable definida en el espacio de trabajo de MATLAB® o por medio de un archivo con extensión .mat o .txt.
- Ingresar el número de la vuelta a la que le será aplicada la señal de entrada
- Elegir el número de vueltas con el que cuenta el devanado.
- Seleccionar las vueltas a través de las cuales podrá medirse la señal de salida.
- Activar y desactivar la inclusión de pérdidas en serie del devanado.
- Permitir el guardado de un archivo de salida en el espacio de trabajo de MATLAB®.
- Presentar una gráfica de contorno de la distribución de tensión de cada una de las vueltas del devanado contra el tiempo.
- Interactuar con otros elementos prestablecidos de la biblioteca de MATLAB-Simulink®.

También debe contar con las siguientes limitaciones:

- Limitar el tipo y las dimensiones de las variables ingresadas en el bloque.
- Prohibir la ausencia de alguna de las variables necesarias para el bloque.

4.2.2 Definiendo el tipo de bloque personalizado

En base a las características del bloque, el bloque implementado debe soportar lo siguiente:

- Un puerto de entrada y múltiples puertos de salida que cambien con los parámetros definidos por el usuario.
- Un algoritmo relativamente simple.
- Variables de estado continuas.
- Diferentes tipos de datos.

Por lo tanto, se implementará un bloque personalizado utilizando una Level-2 M-File S-Function.

4.2.2.1 Definiendo los parámetros de la M-File S-Function

El modelo del transformador de parámetros concentrados requiere de los siguientes parámetros:

1. Un parámetro que indica la forma en la cual se introducen las matrices de inductancia y capacitancia. Esto puede realizarse mediante una variable

definida en el espacio de trabajo de MATLAB® o por medio de un archivo con extensión .mat o .txt.

2. Un parámetro que es el número de vueltas que tiene el devanado del transformador.
3. Un parámetro que es el número de vuelta a la cual se le aplica la señal de entrada.
4. Un parámetro que es el valor de la matriz de capacitancia, el cual se ingresa con el nombre de una variable definida en el espacio de trabajo de MATLAB® o por medio de un archivo con extensión .mat.
5. Un parámetro que es el valor de la matriz de inductancia, el cual se ingresa con el nombre de una variable definida en el espacio de trabajo de MATLAB® o por medio de un archivo con extensión .mat.
6. Un parámetro que permite el ingreso de las pérdidas en serie del devanado
7. Un parámetro que es el valor de las pérdidas en serie del devanado. Este parámetro se utiliza solo si se activa la opción de ingresar las pérdidas en serie.
8. Un parámetro que permite elegir el número de vueltas específicas de las cuales se desea medir la señal de salida.
9. Un parámetro que es el vector que contiene los números de vueltas específicas de las cuales se desea medir la señal de salida. Este parámetro se utiliza solo si se activa la opción de elegir el número de vueltas específicas de las cuales se desea medir la señal de salida.
10. Un parámetro que permite el guardado de un archivo de salida con las tensiones de todas las vueltas del devanado.
11. Un parámetro que es el nombre con el cual se guarda el archivo de salida generado. Este parámetro se utiliza solo si se activa la opción de guardado de un archivo de salida.
12. Un parámetro que permite la realización de una gráfica de contorno de la distribución de tensión de cada una de las vueltas del devanado contra el tiempo.
13. Cuatro parámetros que representan los valores de las matrices $\boldsymbol{A}$, $\boldsymbol{B}$, $\boldsymbol{F}$ y $\boldsymbol{D}$, respectivamente.

4.2.2.2 Escritura de la M-File S-Function

Una Level-2 M-File S-Function es un archivo M que define las propiedades y comportamiento del bloque personalizado. Un archivo M contiene un conjunto de "Callback Methods" que Simulink llama cuando realiza la actualización o simulación del modelo. Los Callback Methods realizan la inicialización de las variables y el cálculo de las salidas del bloque definido por la S-Function. Al crear una S-Function con un conjunto apropiado de Callback Methods, se puede definir

un bloque que cumpla con los requerimientos de la aplicación que se desea realizar [47].

Las M-File S-Functions deben incluir obligatoriamente los siguientes Callback Methods:

- Una función **setup** para incializar las características básicas de la S-Function.
- Una función **Outputs** para calcular la salida de la S-Function.

Las S-Functions pueden contener otros Callback Methods dependiendo de los requerimientos del bloque.

Definidos los parámetros de la S-Function y su funcionalidad, se procede a escribir la S-Function. Se puede utilizar una plantilla prestablecida como punto de inicio que puede consultarse en [47].

El código fuente se muestra por partes con el objetivo de poder explicar con detalle las modificaciones realizadas y su significado. El código completo del modelo de parámetros concentrados se puede consultar en el **ANEXO A**.

- Primero, dentro de la función **setup** se especifica el número de parámetros que la S-Function soporta usando la instrucción **block.NumDialogPrms**

```
function setup(block)
% Register parameters
block.NumDialogPrms = 13;
```

- Posteriormente, se asignan algunos de estos parámetros a variables; para este caso específico, se asigna los parámetros 3 y 10 a las variables *n*, que representa el número de vueltas del devanado, y *NumOut*, que representa el número de vueltas específicas de las que se quiere obtener la señal de salida, respectivamente. Esta acción se realiza con la finalidad de simplificar la escritura del código dentro de la función **setup**; cuando se desee utilizar alguno de los parámetros mencionados, bastará con utilizar la variable en lugar de escribir la instrucción completa **block.DialogPrm(*number*).Data**. Cabe mencionar que lo anterior debe realizarse dentro de cada uno de los Callback Methods contenidos en la M-File S-Function

```
n = block.DialogPrm(3).Data;
NumOut = block.DialogPrm(10).Data;
```

- La instrucción **get_param** en conjunción con la propiedad **UserData** se utiliza para obtener información del usuario asociada con un bloque y guardarla en una variable de cualquier tipo, incluyendo arreglos, estructuras y objetos. En este caso se asigna la información del usuario asociada con la manipulación del bloque **block.BlockHandle** a la variable tipo estructura *ud* la cual se definirá más adelante.

  Al tratarse de una variable tipo estructura, *ud* cuenta con diferentes campos. La variable *n* que representa el número de vueltas del devanado del transformador se asigna al campo de la variable *ud* designado como *nodo.*

  Se asocian los datos recién ingresados en *ud* con la información del usuario que se relaciona con la manipulación del bloque por medio de la instrucción **set_param** en conjunción con la propiedad **UserData**.

```
ud = get_param(block.BlockHandle,'UserData');
ud.nodo = n;
set_param(block.BlockHandle,'UserData',ud)
```

- A continuación, se especifican el número de puertos de entrada y de salida a través de las instrucciones **block.NumInputPorts** y **block.NumOutputPorts**, respectivamente. Para el caso del modelo de parámetros concentrados sólo se hace uso de un puerto de entrada en el cual se aplicará la señal y que puede representar el inicio de cualquiera de las vueltas del devanado del transformador. El número de puertos de salida se encuentra definido por la variable *NumOut* que define el número de vueltas de las que se requiere obtener la señal de salida; su valor puede ser el total del número de vueltas del devanado o, en el caso de seleccionar vueltas específicas, el número de éstas.

```
% Register number of ports
block.NumInputPorts  = 1;
block.NumOutputPorts = NumOut;
```

- La instrucción **block.SetPreCompInpPortInfoToDynamic** se utiliza cuando todos los puertos de entrada heredan sus propiedades (tipo de datos, dimensiones, complejidad y modo de muestreo) de sus señales de entrada. Esto se realiza de manera similar para los puertos de salida mediante la instrucción **block.SetPreCompOutPortInfoToDynamic**

```
% Setup port properties to be inherited or dynamic
block.SetPreCompInpPortInfoToDynamic;
block.SetPreCompOutPortInfoToDynamic;
```

- Para cada puerto de entrada, el método **setup** puede especificar [47]:

  1. Las dimensiones del puerto de entrada por medio de la instrucción **block.InputPort(*number*).Dimensions**.
  2. Si el Puerto tiene retroalimentación directa usando la instrucción **block.InputPort(*number*).DirectFeedthrough**. Un puerto tiene retroalimentación directa si la entrada se utiliza para calcular la salida en el método **Outputs** o para calcular el siguiente tiempo de muestreo para una S-Function con tiempo de muestreo variable. La bandera de retroalimentación directa puede establecerse a *true* o *false*.
  3. El tipo de datos del puerto de entrada utilizando la instrucción **block.InputPort(*number*).DatatypeID**. Si se quiere que el tipo de datos del puerto dependa del tipo de datos del puerto al que se encuentra conectado, se especifica que el tipo de datos es -1.
  4. El tipo numérico del puerto de entrada, si el puerto acepta señales con valores complejos, usando la instrucción **block.InputPort(*number*).Complexity**. Si se desea que el tipo numérico del puerto dependa del tipo numérico del puerto al que se encuentra conectado, se especifica el tipo numérico como *'Inherited'*.
  5. El modo de muestreo del puerto de entrada, usando la instrucción **block.InputPort(*number*).SamplingMode**. Si se quiere que el modo de muestreo del puerto dependa del modo de muestreo del puerto al que se encuentra conectado, se especifica el modo de muestreo como *'Inherited'*. Si la S-Function tiene múltiples puertos de salida, se debe implementar el método **SetInputPortComplexSignal** si cualquiera de los puertos tiene un modo de muestreo *'Inherited'*.

  Estas propiedades pueden establecerse de manera similar para los puertos de salida cambiando la palabra **Input** por **Output** en cada una de las instrucciones anteriores

```
%  Override input port properties
block.InputPort(1).Dimensions  = 1;
block.InputPort(1).DatatypeID  = 0;  % double
block.InputPort(1).Complexity  = 'Real';
block.InputPort(1).DirectFeedthrough = true;
```

- Se especifica el tiempo de muestreo al cual el bloque produce las señales de salida por medio de la instrucción **block.SampleTimes**. En este caso se

tiene un tiempo de muestreo continuo. Para mayor información del establecimiento del tiempo de muestreo, véase [47].

```
block.SampleTimes = [0 0];
```

- Si la S-Function necesita variables de estado continuas, se inicializa el número de estas variables en el método **setup** por medio de la instrucción **block.NumContStates**. Para este modelo se necesitan $2n-1$ variables debido a la formulación realizada de las ecuaciones de estado

```
% Set up the continuous states.
block.NumContStates = ((2*n)-1);
```

- Por medio de la instrucción **block.RegBlockMethod** se registran los Callback Methods necesarios para el bloque personalizado, cada uno de los cuales realiza una función específica durante la simulación.

```
block.RegBlockMethod('CheckParameters', @CheckPrms);
block.RegBlockMethod('InitializeConditions',
                     @InitializeConditions);
block.RegBlockMethod('SetInputPortSamplingMode',
                     @SetInpPortFrameData);
block.RegBlockMethod('Derivatives', @Derivatives);
block.RegBlockMethod('Outputs',@Outputs);
block.RegBlockMethod('Terminate', @Terminate); % Required
```

- El segundo método utilizado es **CheckParameters**, el cual tiene la función de revisar la validez de los parámetros de la S-Function. Para el modelo de parámetros concentrados se utiliza para verificar que los datos ingresados en los campos de la interfaz con el usuario sean del tipo y dimensiones correctas, así como que ninguno de los campos quede vacío.

  De manera similar al método **setup** se realiza la asignación de parámetros del bloque a variables que serán utilizadas dentro del método. La variable *Node* representa la vuelta a la que le será aplicada la señal de entrada; *Param* es el modo por el cual se ingresarán las matrices de inductancia y capacitancia al modelo; *CB* es la matriz de capacitancia; *LB* es la matriz de inductancia; *Outurns* es el vector que contiene los números de vueltas específicas de las que se requiere obtener la señal de salida; *Rtot* es el valor de las pérdidas en serie y *File* es el nombre del archivo de salida

```
function CheckPrms(block)
Node  = block.DialogPrm(1).Data;
Param = block.DialogPrm(2).Data;
n     = block.DialogPrm(3).Data;
CB    = block.DialogPrm(4).Data;
```

```
LB    = block.DialogPrm(5).Data;
Outurns = block.DialogPrm(11).Data;
Rtot = block.DialogPrm(12).Data;
File = block.DialogPrm(13).Data;
```

- Se definen cuatro variables, a las cuales se les asignan los números de filas y de columnas de las matrices *CB* y *LB*, respectivamente

```
[mc,nc] = size(CB);
[ml,nl] = size(LB);
```

- Se establecen mensajes de error en caso de que los parámetros no sean introducidos correctamente en los campos de la interfaz del usuario. Cuando se ingresan las matrices de inductancia y capacitancia por medio de una variable definida en el espacio de trabajo de MATLAB®, se tiene que el primero de ellos se despliega en caso de que el campo de número de vueltas del devanado se deje vacío.

```
if isequal(Param,1)
    if isempty(n)
        error('Introduce the number of turns');
    end
```

- El segundo mensaje de error se despliega en caso de que el número de vueltas ingresado en el campo no sea un número entero.

```
if mod(n,1)~=0
    error('Number of turns must be an integer type');
end
```

- El siguiente mensaje de error se refiere a que el número de vueltas ingresado sea igual a cero o un número negativo.

```
if n<1
    error('Number of turns must be positive or nonzero')
end
```

- Los mensajes de error anteriores se aplican de manera similar al número ingresado como vuelta a la que será aplicada la señal de entrada; además, se agrega una restricción en el caso de que el número ingresado sea mayor al número de vueltas del devanado.

```
if isempty(Node)
    error('Introduce the input node');
end
if mod(Node,1)~=0
    error('Input node must be an integer type');
```

```
end
if Node<1
    error('Input node must be positive or nonzero')
end
if Node>n
    error('Input node does not exist');
end
```

- Las siguientes restricciones se encuentran relacionadas con las matrices de inductancia y capacitancia: los campos donde se ingresan los nombres de las variables que representan esas matrices no pueden estar vacíos; el contenido de las matrices debe ser del tipo numérico; las matrices deben ser cuadradas y la dimensión de las matrices debe ser igual al número de vueltas del devanado.

```
if isempty(CB)
    error('Introduce the Capacitance Matrix Name');
end
if ~isa(CB,'numeric')
    error('Capacitance matrix must be numeric type');
end
if mc ~= nc
    error('Capacitance Matrix must be square');
end
if ~isequal(mc,n)
    error('Dimension of Capacitance Matrix must be equal to the
number of turns');
end
if isempty(LB)
    error('Introduce the Inductance Matrix Name');
end
if ~isa(LB,'numeric')
    error('Inductance matrix must be numeric type');
end
if ml ~= nl
    error('Inductance Matrix must be square');
end
if ~isequal(ml,n)
    error('Dimension of Inductance Matrix must be equal to the
number of turns');
end
```

- En el caso del valor de las pérdidas en serie, también se despliega un mensaje de error en caso de que el campo quede vacío, cuando el valor ingresado no es del tipo numérico y cuando el valor ingresado es un número negativo. En esta última restricción, debido a que las pérdidas en serie forman una matriz diagonal, únicamente se toma en consideración el primer valor para el desplegado del mensaje de error.

```
if isempty(Rtot)
    error('Introduce Series Losses Value');
end
if ~isa(Rtot,'numeric')
    error('Series Losses Value must be numeric type');
end
if Rtot(1,1)<0
    error('Series Losses Value must be positive or zero');
end
```

- Para el parámetro que representa el vector que contiene el número de las vueltas de las que se desea obtener la señal de salida, también se establecen restricciones en caso de que el campo se encuentre vacío y que el valor ingresado no sea del tipo numérico; sin embargo, se añade la restricción de que los números de vueltas específicas sean ingresados en forma de vector.

```
if isempty(Outurns)
    error('Introduce Number(s) of Specific Turn(s)');
end
if ~isa(Outurns,'numeric')
    error('Number(s) of Specific Turn(s) must be numeric type');
end
if ~isvector(Outurns)
    error('Number(s) of Specific Turn(s) must be a vector');
end
```

- Se definen dos variables que representan las dimensiones del vector que contiene los números de vueltas específicas. Con estas dos variables se verifica que cada uno de los elementos contenidos en el vector no sea igual a cero o que tenga un valor negativo y que no sea mayor al número de vueltas del devanado.

```
[mout,nout] = size(Outurns);
for i=1:mout
    for j=1:nout
        if Outurns(i,j)<1
            error('Number(s) of Specific Turn(s) must be
            positive or nonzero');
            break
        end
        if Outurns(i,j)>n
            error('Number(s) of Specific Turn(s) does(do) not
            exist');
            break
        end
    end
end
```

- Por último, se establece una restricción para el caso cuando el campo del nombre del archivo de salida se encuentre vacío.

```
if isempty(File)
    error('Introduce Output File Name');
end
```

- Las restricciones anteriores se establecen de forma similar cuando los parámetros se ingresan por medio de archivo .mat o .txt.

- El tercer método utilizado es **InitializeConditions**, el cual tiene la función de inicializar los vectores de variables de estado de la S-Function. Para el modelo de parámetros concentrados se inicializa con un valor igual a cero un vector de variables de estado de dimensiones $(2n-1)\times 1$ por medio de la instrucción **block.ContStates.Data**

```
function InitializeConditions(block)
n = block.DialogPrm(3).Data;
block.ContStates.Data= zeros(((2*n)-1),1);
```

- El cuarto método utilizado es **SetInputPortSamplingMode**, el cual se utiliza en una S-Function en la que sus puertos heredan su método de muestreo de los bloques a los que se encuentran conectados. En este caso se establece un ciclo para que cada uno de los puertos de salida y el puerto de entrada tengan un método de muestreo tipo **sample** por medio de las instrucciones **block.InputPort(*number*).SamplingMode**, para el caso de los puertos de entrada y **block.OutputPort(*number*).SamplingMode**, para el caso de los puertos de salida

```
function SetInpPortFrameData(block,~,~)
NumOut = block.DialogPrm(10).Data;
block.InputPort(1).SamplingMode = 0;
for i=1:NumOut
    block.OutputPort(i).SamplingMode  = 0;
end
```

- El quinto método utilizado es **Derivatives**, el cual resuelve el sistema de ecuaciones diferenciales ordinarias propias del modelo que se desea implementar.

  Como primer paso dentro del método, se asignan los parámetros 6 y 7 a las variables *A* y *B*, que representan a las matrices de las ecuaciones de estado con las que se resuelve el sistema.

Se obtiene de nueva cuenta la información del usuario asociada con la manipulación del bloque; el campo de la variable *ud* designado como *entrada* se asigna a una variable con el mismo nombre con el propósito de realizar operaciones con ella. Esta acción se realiza también para el campo *deriv*.

```
function Derivatives(block)
A = block.DialogPrm(6).Data;
B = block.DialogPrm(7).Data;
ud = get_param(block.BlockHandle,'UserData');
entrada = ud.entrada;
deriv = ud.deriv;
```

- Se realiza el cálculo de las derivadas de las variables de estado continuas, por medio de la ecuación (2.11) y la instrucción **block.Derivatives.Data**

```
block.Derivatives.Data =  A*block.ContStates.Data+
                          B*[entrada(end);deriv(end)];
```

- El sexto método utilizado es **Outputs**, el cual calcula las salidas de la S-Function en el correspondiente paso de tiempo y almacena los resultados en los puertos de salida del bloque. Para S-Functions que tienen tiempo de muestreo variable, este método calcula el siguiente tiempo de muestreo en el que se realizará el cálculo.

  De forma similar al método anterior, se asignan los parámetros a las variables correspondientes. En este caso los parámetros 8 y 9 se asignan a las variables *F* y *D* que representan a las matrices de las ecuaciones de estado con las que se resuelve el sistema.

```
function Outputs(block)
Node     = block.DialogPrm(1).Data;
n        = block.DialogPrm(3).Data;
F        = block.DialogPrm(8).Data;
D        = block.DialogPrm(9).Data;
Outurns  = block.DialogPrm(11).Data;
ud = get_param(block.BlockHandle,'UserData');
```

- Como se muestra en la ecuación (2.36), el vector de entrada utilizado en las ecuaciones de estado consiste de dos elementos: uno que representa a la señal de entrada y otro que representa la derivada de dicha señal. El cálculo de la derivada se realiza por medio de la siguiente ecuación (diferencias finitas hacia atrás):

$$\frac{dv_1}{dt} \approx \frac{v_1(t) - v_1(t - \Delta t)}{\Delta t} \tag{4.1}$$

Sin embargo, en $t = 0$ la derivada de la señal de entrada no puede calcularse por medio de la ecuación anterior, por lo que se supone que su valor es igual a cero

Debido a lo anterior, se establecen dos condiciones: una para el caso de que el tiempo de simulación sea igual a cero, por medio de la instrucción **block.CurrentTime**, y otra para un tiempo de simulación diferente. Además, se agrega una restricción dada por la instrucción **block.IsMajorTimeStep** debido a que MATLAB-Simulink®, para modelos que contienen variables de estado continuas, realiza pasos de tiempo menores que permiten al método numérico utilizado para la solución del problema, alcanzar la exactitud deseada en el cálculo de las variables de estado; sin embargo, para este caso solo es necesaria la solución obtenida una vez alcanzada la exactitud requerida, lo que corresponde a un paso de tiempo mayor.

Para el caso en el que el tiempo de simulación es igual a cero se almacena el tiempo de simulación actual en el campo *tiempo*, la señal de entrada en el campo *entrada* y el valor de la derivada de la señal de entrada en el campo *derivada*, todos de la estructura *ud*, para posteriormente utilizar estos vectores en la generación del archivo y gráfica de salida. Estos vectores se asignan a variables con el mismo nombre, que servirán para realizar los cálculos necesarios para el modelo de parámetros concentrados. Todos los datos ingresados en *ud* se asocian nuevamente con la información del usuario que se encarga de la manipulación del bloque.

Debido a que existe la posibilidad de seleccionar vueltas específicas, los resultados se guardan en la variable *OUT* para su posterior manipulación.

```
if  block.CurrentTime==0 && block.IsMajorTimeStep
    ud.tiempo = [ud.tiempo block.CurrentTime];
    ud.entrada = [ud.entrada block.InputPort(1).Data];
    ud.deriv = [ud.deriv 0.0];
    entrada = ud.entrada;
    deriv = ud.deriv;
    set_param(block.BlockHandle,'UserData',ud)
    OUT = F*block.ContStates.Data+D*[entrada(end);deriv(end)];
```

- Cuando el tiempo de simulación es diferente de cero, el procedimiento es similar, exceptuando el cálculo del valor de la derivada de la señal de entrada que se lleva a cabo por medio de la ecuación (4.1).

```
elseif block.IsMajorTimeStep
    ud.tiempo = [ud.tiempo block.CurrentTime];
    tiempo2 = ud.tiempo;
    ud.entrada = [ud.entrada block.InputPort(1).Data];
    entrada = ud.entrada;
    ud.deriv = [ud.deriv (entrada(end)-
                entrada(end-1))/(tiempo(end)-tiempo(end-1))];
    deriv = ud.deriv;
    set_param(block.BlockHandle,'UserData',ud)
    OUT = F*block.ContStates.Data+D*[entrada(end);deriv(end)];
end
```

- Si se trata de un paso de simulación mayor, las señales de salida almacenadas en la variable *OUT* se asignan a los puertos que representan las vueltas específicas seleccionadas.

```
if block.IsMajorTimeStep
    k = 1;
    for i=Outurns
        if i==Node
            block.OutputPort(k).Data = block.InputPort(1).Data;
        elseif i<Node
            block.OutputPort(k).Data = OUT(i,:);
        else
            block.OutputPort(k).Data = OUT(i-1,:);
        end
        k = k+1;
    end
```

- Si, además, la opción de archivo de salida o la opción de graficado se encuentran seleccionadas, se ingresan las tensiones de salida en el campo *salida* de la estructura *ud* y se asocian de nueva cuenta los valores ingresados con la información del usuario que se encarga de la manipulación del bloque.

```
if strcmp(ud.opcion,'on')||strcmp(ud.opcion2,'on')
    for i=1:n
        if i==Node
            ud.salida = [ud.salida block.InputPort(1).Data];
        elseif i<Node
            ud.salida = [ud.salida OUT(i,:)];
        else
            ud.salida = [ud.salida OUT(i-1,:)];
        end
    end
    set_param(block.BlockHandle,'UserData',ud)
end
```

- El último método utilizado es **Terminate**. Este es un método que debe incluirse de forma obligatoria ya que realiza acciones necesarias para MATLAB-Simulink® cuando se termina una simulación, tales como la liberación de memoria.

```
function Terminate(~)
```

4.2.3 Colocación del bloque personalizado en una biblioteca

Las bibliotecas permiten compartir los bloques personalizados con otros usuarios, actualizar fácilmente la funcionalidad de copias del bloque personalizado y agrupar bloques para un proyecto en particular en un mismo lugar. Para colocar el bloque personalizado en una biblioteca se tienen los siguientes pasos [47]:

1. Abrir una nueva biblioteca de MATLAB-Simulink®.
2. Añadir un nuevo bloque Level-2 M-File S-Function desde la biblioteca Simulink User-Defined Functions dentro de la nueva biblioteca.
3. Dar doble clic en el bloque para abrir el cuadro de diálogo de parámetros del bloque. Ingresar el nombre de la S-Function en el campo con el título **M-File Name**. Para este caso, el nombre es *trans_concen*.
4. Ingresar los nombres de las variables a utilizar por el bloque en el campo **Parameters**: *Node,Param,n,CB,LB,A,B,F,D,NumOut,Outurns,Rtot,File.*
5. Dar clic en **OK** en el cuadro de diálogo.
6. Guardar la biblioteca en el directorio de trabajo, en este caso como *transformers.mdl.*

4.2.4 Adición de una interfaz con el usuario a un bloque personalizado

Se puede crear un cuadro de diálogo sencillo utilizando la opción de enmascaramiento incluida en MATLAB-Simulink®. Para enmascarar el bloque se tienen los siguientes pasos:

1. Dar clic derecho en el bloque personalizado en la biblioteca *transformers.mdl* y seleccionar **Mask M-File S-Function** desde el menú desplegable para abrir el editor de máscara.

2. En el panel **Icon & Ports** se ingresa el siguiente código dentro de la sección **Drawing Commands**.

```
image(imread('Trans_conc.jpg'))
[mOut,nOut] = size(Outurns);
for i=1:length(Outurns)
    if mOut<=nOut
        Outurns2 = num2str(Outurns(1,i));
    else
        Outurns2 = num2str(Outurns(i,1));
    end
    port_label('output',i,Outurns2)
end
```

La instrucción **image** sirve para desplegar la imagen que representa al modelo de parámetros concentrados.

El vector utilizado para ingresar el número de vueltas específicas puede ser tipo fila o tipo columna. Se comparan sus dimensiones para determinar de qué forma se leerán cada uno de sus elementos y así poder asignarlos a la variable *Outurns2*. Para mostrar el número de cada una de las vueltas del bloque se utiliza la instrucción **port_label**.

Si se desea mayor información de los paneles del editor de máscara y los comandos de dibujo, véase [47].

3. En el panel **Parameters** se añaden los parámetros correspondientes a la S-Function, estableciendo sus propiedades como se muestra en la Tabla 4.1

Tabla 4.1 Propiedades de los parámetros de la S-Function para el modelo de parámetros concentrados

Prompt	Variable	Type	Evaluate	Enable parameter	Show parameter
Parameters Calculation	Param	popup	Sí	Sí	Sí
Number of Turns	n	edit	Sí	Sí	Sí
Input Node	Node	edit	Sí	Sí	Sí
Capacitance Matrix Name [F]	CB	edit	Sí	Sí	Sí
Inductance Matrix Name [H]	LB	edit	Sí	Sí	Sí
Capacitance Matrix File Name [F]	CBF	edit	No	Sí	No

Inductance Matrix File Name [H]	LBF	edit	No	Sí	No
Add Losses	Loss	checkbox	No	Sí	Sí
Series Losses per turn [ohm]	Rtot	edit	Sí	Sí	No
Select Specific Output Turn(s)	Spout	checkbox	No	Sí	Sí
Output Turn(s)	Outurns	edit	Sí	Sí	No
Save Output File	Fileopt	checkbox	No	Sí	Sí
Output File Name	File	edit	No	Sí	No
Potential Distribution Contour Plot	Meshopt	checkbox	No	Sí	Sí
Matrix1	A	edit	Sí	No	No
Matrix2	B	edit	Sí	No	No
Matrix3	F	edit	Sí	No	No
Matrix4	D	edit	Sí	No	No

Debido a que la variable *Param* tiene un control de edición del tipo **popup**, se deben ingresar los valores del control en el campo **Popups**, siendo estos los modos por los cuales podrán ser ingresadas las matrices de capacitancia e inductancia al modelo implementado:

```
Predefined Matrices (Workspace)
From File (.mat)
```

Para lograr que el cuadro de diálogo cambie su apariencia dependiendo de los ajustes de control hechos por el usuario, es necesario ingresar código en el campo denominado **Dialog callback**.

Dentro de MATLAB-Simulink® se tiene una serie de parámetros que describen los bloques enmascarados. Pueden usarse estos parámetros descriptivos en combinación con las instrucciones **get_param** y **set_param** para obtener y especificar las propiedades de la máscara de un bloque.

Para la variable *Param* se asignan los valores de los parámetros del cuadro de diálogo a un vector denominado *maskStr* utilizando el parámetro descriptivo **MaskValues**; el parámetro **gcb** se refiere a que los parámetros se obtendrán del bloque actualmente seleccionado. De forma similar se obtienen las visibilidades de cada uno de los parámetros del cuadro de diálogo por medio del parámetro descriptivo **MaskVisibilities** y estas se asignan a un vector denominado *vis*.

Posteriormente, se utiliza la instrucción **switch** para determinar cuál de los dos modos de inserción de parámetros se encuentra activo y de esta forma poder modificar la visibilidad de los parámetros correspondientes; en el caso de ingresar las matrices por medio de variables predefinidas en el espacio de trabajo de MATLAB®, los campos *Inductance Matrix Name* y *Capacitance Matrix Name* se activarán y los campos *Inductance Matrix File Name* y *Capacitance Matrix File Name* serán desactivados; esto funcionará de forma opuesta cuando el modo de inserción de parámetros sea por medio de un archivo con extensión .mat o .txt.

```
maskStr = get_param(gcb,'MaskValues');
vis = get_param(gcb,'MaskVisibilities');
switch maskStr{1}
    case 'Predefined Matrices (Workspace)'
        set_param(gcb,'MaskVisibilities',[vis(1:11);{'on'};{'on'};
                  {'off'};{'off'};vis(16:26)]);
    case 'From File (.mat)'
      set_param(gcb,'MaskVisibilities',[vis(1:11);{'off'};{'off'};
                  {'on'};{'on'};vis(16:26)]);
end
```

Para las variables *Loss* y *Spout* se utilizan líneas de código similares a las anteriores con la diferencia de éstas se utilizan, en el primer caso, para activar o desactivar el campo de inclusión de pérdidas en serie, y en el segundo, para especificar los números de vueltas específicas de las cuales se desea obtener la señal de salida.

```
maskStr = get_param(gcb,'MaskValues');
vis = get_param(gcb,'MaskVisibilities');
if strcmp(maskStr{16},'on')
set_param(gcb,'MaskVisibilities',[vis(1:16);{'on'};vis(18:26)]);
else
set_param(gcb,'MaskVisibilities',[vis(1:16);{'off'};vis(18:26)]);
end
maskStr = get_param(gcb,'MaskValues');
vis = get_param(gcb,'MaskVisibilities');
if strcmp(maskStr{18},'on')
```

```
set_param(gcb,'MaskVisibilities',[vis(1:18);{'on'};vis(20:26)]);
else
set_param(gcb,'MaskVisibilities',[vis(1:18);{'off'};vis(20:26)]);
end
```

Al ingresar las líneas de código de la variable *Fileopt* se realiza la declaración de la estructura *ud* cuyos campos se utilizan en el código de la S-Function; la asignación de los valores de los parámetros 20 y 21 a los campos *opcion* y *nombre* de dicha estructura, permitirá o no el guardado de un archivo de salida con el nombre escrito en el campo correspondiente.

```
maskStr = get_param(gcb,'MaskValues');
vis = get_param(gcb,'MaskVisibilities');
ud = struct('nombre',[],'opcion',[],'opcion2',[],
            'tiempo',[],'salida',[],'nodo',[],
            'entrada',[],'deriv',[]);
if strcmp(maskStr{20},'on')
set_param(gcb,'MaskVisibilities',[vis(1:20);{'on'};vis(22:26)]);
    ud.nombre = maskStr{21};
    ud.opcion = maskStr{20};
else
set_param(gcb,'MaskVisibilities',[vis(1:20);{'off'};vis(22:26)]);
end
set_param(gcb,'UserData',ud);
```

En el caso de la variable *Meshopt*, el valor del parámetro 22 se asigna al campo *opcion2* que permitirá o no la realización de una gráfica de contorno de la distribución de tensión de cada una de las vueltas del devanado contra el tiempo.

```
maskStr = get_param(gcb,'MaskValues');
if strcmp(maskStr{22},'on')
    ud.opcion2 = maskStr{22};
end
set_param(gcb,'UserData',ud);
```

4. En el panel **Initialization** se escriben los comandos de inicialización del bloque, con lo que se logrará realizar el cálculo de variables que serán utilizadas dentro de la S-Function. Primeramente se asigna una matriz identidad de dimensión igual al número de vueltas del devanado del transformador a la variable *I*.

```
I = eye(n);
```

Si la opción de agregación de pérdidas en serie se encuentra desactivada, la variable *Rtot* que las representa será una matriz cuadrada de ceros de dimensión igual al número de vueltas del devanado. En caso contrario, el

valor ingresado en el campo correspondiente del cuadro de diálogo será multiplicado por la matriz identidad.

```
if strcmp(Loss,'off')
    Rtot = zeros(n);
else
    Rtot = Rtot*I;
end
```

Cuando la opción de selección de vueltas específicas se encuentre desactivada, la variable *NumOut*, que representa el número de puertos de salida que tendrá el bloque, será igual que el número de vueltas del devanado; la variable *Outurns*, que es el vector que contiene los números de las vueltas específicas, tendrá todos los números de las vueltas dentro de él. En caso contrario, el número de puertos de salida estará relacionado con el número de elementos de los que conste el vector *Outurns*.

```
if strcmp(Spout,'off')
    NumOut = n;
    Outurns = [1:n];
else
    NumOut = length(Outurns);
end
```

En caso de que la inserción de parámetros sea por medio de un archivo con extensión .mat, las variables *CB* y *LB* que representan a las matrices de capacitancia e inductancia respectivamente, obtendrán sus valores de importar los archivos con los nombres guardados en las variables *CBF* y *LBF*.

```
if isequal(Param,2)
    CB = importdata(CBF);
    LB = importdata(LBF);
end
```

Las siguientes líneas de código realizan los cálculos necesarios para formar las matrices dadas por las ecuaciones (2.22), (2.23), (2.24) y (2.25).

```
Qi=zeros(n);
Qi(1,1)=1;
for i=2:n
    Qi(i,i)= 1; Qi(i,i-1)=-1;
end
QiT= Qi'; T1=(LB^-1);
Ki=-T1*Rtot; T2=zeros(n,n);
for l=1:n
   for m=1:n
       if m==1
```

```
            T2(l,m)=T1(l,m);
        else
            T2(l,m)=T1(l,m)-T1(l,m-1);
        end
    end
end
Ke=T2; Ke(:,Node)=[];
Ku=T2(:,Node);
C4=CB; C4(Node,:)=[]; C4k=C4; C4(:,Node)=[];
Qa=Qi; Qa(Node,:)=[]; df=-(C4^-1); Pi=df*Qa;
C3=C4k(:,Node);
Pu=df*C3;
a=zeros((n-1),(n-1));
A=[Ki,Ke;Pi,a];
b1=zeros(n,1); b2=zeros(n-1,1);
B=[Ku,b1;b2,Pu];
c=zeros(n-1,n); uno=eye(n-1,m-1);
F=[c,uno]; D=zeros(n-1,2);
```

5. En el panel **Initialization** se selecciona la opción **Allow library block to modify its contents**. Esta opción permitirá a la S-Function cambiar el número de puertos del bloque [47].

6. Dentro del panel **Documentation**, en el campo **Mask type** se ingresa el nombre del bloque personalizado: *Lumped-Parameter Transformer Model Based on State-Space Equations*.

7. Dar clic en **OK** en el editor de máscara para completar el cuadro de diálogo.

4.2.5 Agregando funcionalidad al bloque usando callbacks

El bloque personalizado provee una opción para guardar un archivo de salida y una opción para realizar una gráfica de contorno de la distribución de tensión cuando la simulación termina. Los siguientes pasos describen el proceso para su implementación:

1. Escribir el M-File para el guardado del archivo de salida y el graficado de la distribución de tensión. Para el bloque personalizado el nombre de este archivo es: *file_generation.m.*

   Primero se asignan los elementos guardados en los campos *tiempo* y *salida* de la estructura *ud* en las variables *tiempo* y *tensiones*, que serán utilizadas para poder realizar las operaciones de guardado y graficado.

```
function file_generation(block)
ud = get_param(block,'UserData');
```

```
tiempo = ud.tiempo;
tensiones = ud.salida;
OUT = zeros(1,length(tensiones)/ud.nodo);
```

Si el campo que contiene el tiempo de simulación no se encuentra vacío y además la opción de guardado de un archivo de salida o la opción de graficado se encuentra activada, se guardan tanto el tiempo de simulación como las tensiones de cada una de las vueltas del devanado dentro de la variable *OUT*.

```
if ~isempty(ud.tiempo)
    if strcmp(ud.opcion,'on')||strcmp(ud.opcion2,'on')

        k = 1;
        for i=1:(length(tensiones)/ud.nodo)
            OUT(1,i)=tiempo(i);
            for j=2:ud.nodo+1
                OUT(j,i) = tensiones(k);
                k = k +1;
            end
        end
    end
```

Nuevamente se establecen las condiciones anteriores, pero en esta ocasión de manera individual; al cumplirse la primera, la variable *OUT* se guarda en un archivo con el nombre especificado en el campo correspondiente del cuadro de diálogo; al cumplirse la segunda, se realiza una gráfica de contorno de la distribución de tensión de cada una de las vueltas del devanado del transformador contra el tiempo de simulación.

```
if strcmp(ud.opcion,'on')
    save(ud.nombre,'OUT');
end
if strcmp(ud.opcion2,'on')
    TIME = OUT(1,:);
    OUT(1,:)=[];
    TURNS = (1:1:ud.nodo);
    figure(1);
    contourf(TIME,TURNS,OUT);
    title('Potential Distribution');
    xlabel('Time [seconds]');
    ylabel('Turns');
    colorbar('EastOutside');
end
```

2. Dar clic en el bloque Level-2 M-File S-Function y seleccionar la opción **Block Properties**. En el panel **Callbacks** se debe modificar **StopFcn** para que llame al archivo de guardado y graficado, como se muestra en la Figura 4.1. Posteriormente, dar clic en **OK**.

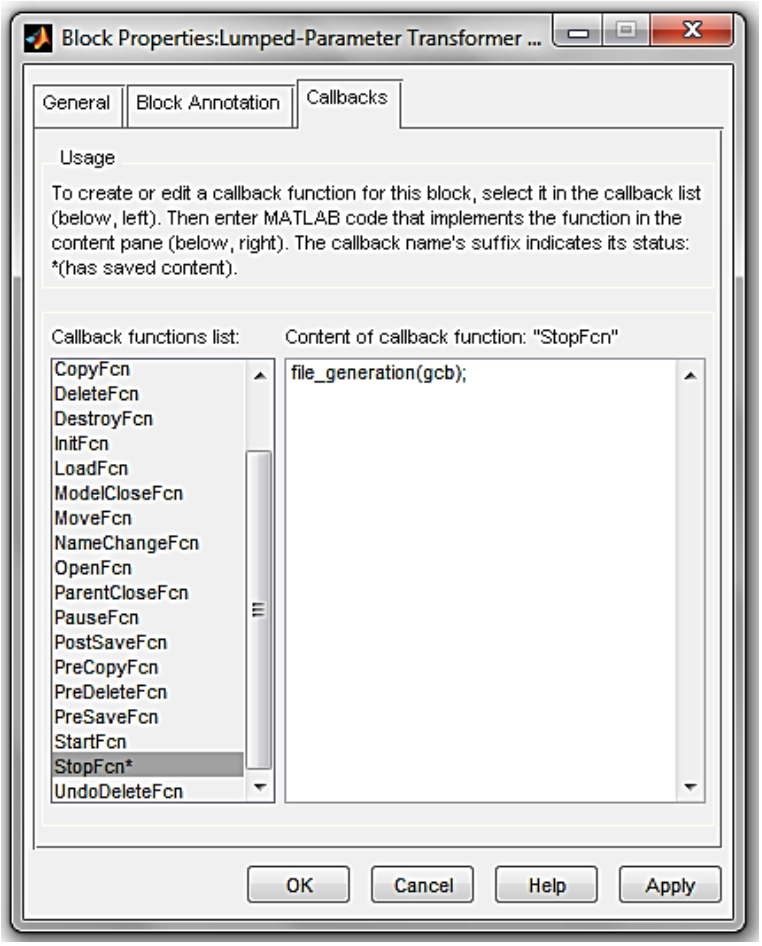
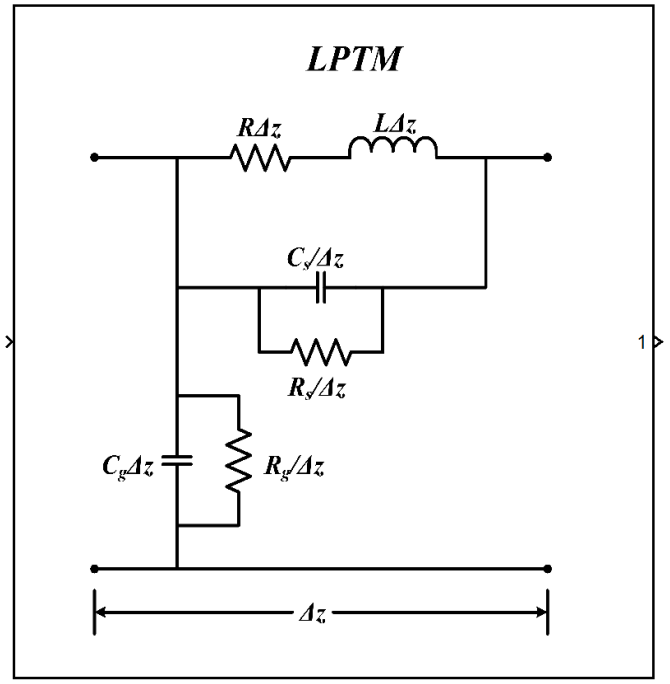


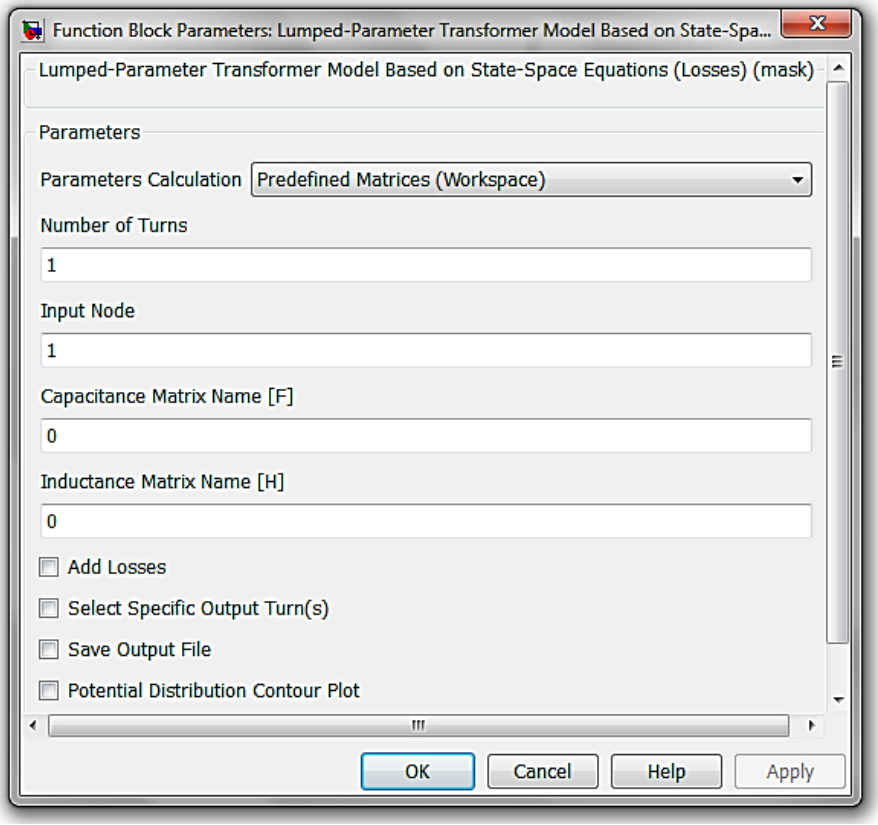
Figura 4.1 Modificación de Callback para guardado de un archivo y graficado

En la Figura 4.2 se muestra el bloque implementado que representa el modelo del transformador de parámetros concentrados basado en ecuaciones de espacio estado, así como la interfaz con el usuario que permite el ingreso de sus parámetros.

Cabe mencionar que los campos que se muestran en las interfaces con el usuario, así como las gráficas de contorno desplegadas al finalizar las simulaciones se encuentran en el idioma inglés; esto se debe a que se tiene contemplado su futuro uso a nivel internacional.



a)



b)

Figura 4.2 Bloque del modelo del transformador de parámetros concentrados basado en ecuaciones de estado: a) Símbolo, b) Interfaz

4.3 IMPLEMENTACIÓN DEL MODELO DEL TRANSFORMADOR DE PARÁMETROS DISTRIBUÍDOS BASADO EN LA TEORÍA DE LA LÍNEA DE TRANSMISIÓN MULTICONDUCTORA

Para realizar la implementación de este modelo se debe seguir un proceso similar al utilizado para la implementación del modelo de parámetros concentrados.

4.3.1 Definiendo el comportamiento del bloque personalizado

Las características y limitaciones de este bloque personalizado son idénticas a las anteriores con excepción de lo siguiente:

- Ingresar la longitud de una vuelta del devanado.

4.3.2 Definiendo el tipo de bloque personalizado

El bloque que contenga el modelo de parámetros distribuidos deberá soportar las mismas características que el bloque del modelo de parámetros concentrados; por lo tanto, se utilizará una Level-2 M-File S-Function para su implementación.

4.3.2.1 Definiendo los parámetros de la M-File S-Function

El modelo del transformador de parámetros distribuidos requiere de los mismos parámetros que el modelo de parámetros concentrados, más algunos adicionales:

1. Un parámetro que es la longitud promedio de una vuelta del devanado del transformador.
2. Un parámetro que representa las veces que cabe el paso de tiempo seleccionado en el tiempo de viaje.
3. Un parámetro que es la matriz de vectores propios de la matriz de inductancia.
4. Un parámetro que es la longitud del vector que contiene el tiempo de simulación.
5. Un parámetro que es la matriz de impedancias características modales de cada una de las líneas que representan el devanado del transformador.
6. Un parámetro que representa la matriz anterior modificada al agregarle las pérdidas en serie de cada una de las vueltas de forma concentrada.

7. Un parámetro que es la matriz de admitancias características modales de cada una de las vueltas del devanado.
8. Un parámetro que es la resistencia interna de la fuente que se utiliza para aplicar el pulso de entrada.
9. Un parámetro que representa la matriz de impedancias obtenida invirtiendo la matriz de admitancias nodales del circuito.

4.3.2.2 Escritura de la M-File S-Function

De manera similar al modelo de parámetros concentrados, el código fuente se muestra por partes con el objetivo de poder explicar con detalle las modificaciones realizadas a la plantilla prestablecida y su significado. El código completo del modelo de parámetros distribuidos se puede consultar en el **ANEXO A**.

- Las primeras instrucciones contenidas dentro de la función **setup** son muy similares a las utilizadas en la implementación de la sección anterior; la diferencia más importante es que para este modelo no se hará uso del método **Derivatives**.

- El segundo método utilizado es **CheckParameters**. Al igual que para el modelo de parámetros concentrados, se utiliza para verificar que los datos ingresados en los campos de la interfaz con el usuario sean del tipo y dimensiones correctas, así como que ninguno de los campos quede vacío, extendiéndose para los parámetros adicionales propios del modelo de parámetros distribuidos.

- El método **SetInputPortSamplingMode** se utiliza del mismo modo que en la implementación del modelo de parámetros concentrados.

- En el método **Outputs** se incluyen dos condiciones: una para el caso de que el tiempo de simulación sea igual a cero y otra para un tiempo de simulación diferente. Esto se hace con la finalidad de separar dos procesos: la primera condición permite inicializar los valores de tensiones nodales y corrientes de rama que serán utilizados en pasos de tiempo posteriores a $t = 0$; la segunda condición permite realizar los cálculos necesarios para obtener las tensiones en el extremo receptor de cada una de las vueltas una vez inicializadas todas las variables.

  Para el caso en el que el tiempo de simulación es igual a cero se establece el valor de la primera iteración en cero, se guarda en el campo *itera* de la estructura *ud* y se asigna a la variable *j*.

A continuación, se establece su valor al sumarle las veces que cabe el paso de tiempo dentro del tiempo de viaje. La razón fundamental del cálculo anterior se encuentra en que es necesario conocer el valor de las tensiones de los extremos emisor y receptor de cada una de las vueltas del devanado en $t = t - \tau$, donde τ es el tiempo de viaje.

```
if block.CurrentTime==0 && block.IsMajorTimeStep
    ud = get_param(block.BlockHandle,'UserData');
    ud.itera = [ud.itera 0.0];
    j = ud.itera;
    j = j+s+1;
```

- Se inicializan los valores de las tensiones nodales de entrada de cada una de las vueltas del devanado con un valor igual a cero, se guardan en el campo *entrada* y se asignan a la matriz *E*; el número de fila de esta matriz es de $2n$ porque incluye las tensiones de ambos extremos de cada una de las vueltas.

```
ud.entrada = [ud.entrada zeros(2*n,k+s)];
E = ud.entrada;
```

- El valor de la señal de entrada se asigna a la fila de la matriz *E* que corresponde a la vuelta en la que se aplica. Utilizando la matriz de vectores propios se realiza la transformación modal de la matriz de tensiones.

```
E(Node,j-s) = block.InputPort(1).Data;
Em = [(I/Tl)*E(1:n,:);(I/Tl)*E(n+1:2*n,:)];
```

- Las matrices de tensiones modales de los extremos emisor y receptor, y de corrientes de historia modales, se inicializan con valores iguales a cero. De forma correspondiente se guardan en los campos *tension0*, *tensionL*, *corriente0* y *corrienteL*; posteriormente se asignan a las variables *v0m*, *vLm*, *I0m* y *ILm*.

```
ud.tension0 = [ud.tension0 zeros(n,k+s)];
v0m = ud.tension0;
ud.tensionL = [ud.tensionL zeros(n,k+s)];
vLm = ud.tensionL;
ud.corriente0 = [ud.corriente0 zeros(n,k+s)];
I0m = ud.corriente0;
ud.corrienteL = [ud.corrienteL zeros(n,k+s)];
ILm = ud.corrienteL;
```

- Las ecuaciones (2.55), (2.52) y (2.48) se aplican al caso de línea multiconductora para calcular las tensiones modales de los extremos emisor y receptor de cada vuelta del devanado del transformador.

```
H0m = -(Z0m/(Z0m2^2))*(vLm(:,j-s)+(Z0m-Rtot)*(ILm(:,j-s)))-
      (Rtot/(Z0m2^2))*(v0m(:,j-s)+(Z0m-Rtot)*(I0m(:,j-s)));
HLm = -(Z0m/(Z0m2^2))*(v0m(:,j-s)+(Z0m-Rtot)*(I0m(:,j-s)))-
      (Rtot/(Z0m2^2))*(vLm(:,j-s)+(Z0m-Rtot)*(ILm(:,j-s)));
Hm = [-H0m;-HLm];
Vm = ZBUS*(Hm-(-1/Rf)*(Em(:,j-s)));
v0m(:,j) = Vm(1:n,1);
vLm(:,j) = Vm(n+1:2*n,1);
I0m(:,j) = (Y0m)*(v0m(:,j))+H0m;
ILm(:,j) = (Y0m)*(vLm(:,j))+HLm;
```

- Los resultados de las tensiones modales de ambos extremos y los valores de las corrientes de rama se guardan de nueva cuenta en los campos correspondientes de la estructura *ud*.

```
ud.tension0 = (v0m);
ud.tensionL = (vLm);
ud.corriente0 = (I0m);
ud.corrienteL = (ILm);
```

- Las tensiones modales del extremo emisor se transforman de vuelta al dominio de fases y almacenados en la variable *OUT* para su posterior manipulación.

```
OUT = (Tl)*vLm;
```

- Las señales almacenadas en la variable *OUT* se asignan a los puertos que representan las vueltas específicas seleccionadas.

```
q =1;
for i=Outurns
     block.OutputPort(q).Data = OUT(i,j(end));
     q = q+1;
end
```

- Cuando el tiempo de simulación es diferente de cero, el procedimiento es similar, excepto que se utilizan los valores almacenados un paso de tiempo anterior de cada una de las variables.

```
elseif block.IsMajorTimeStep
    ud = get_param(block.BlockHandle,'UserData');
    ud.itera = [ud.itera j(end)+1];
    j = ud.itera;
    j = j+s+1;
```

```
    E = ud.entrada;
    E(Node,j(end)-s) = block.InputPort(1).Data;
    Em = [(I/Tl)*E(1:n,:);(I/Tl)*E(n+1:2*n,:)];
    v0m = ud.tension0;
    vLm = ud.tensionL;
    I0m = ud.corriente0;
    ILm = ud.corrienteL;
    H0m = -(Z0m/(Z0m2^2))*(vLm(:,j(end)-s)+
          (Z0m-Rtot)*(ILm(:,j(end)-s)))-
          (Rtot/(Z0m2^2))*(v0m(:,j(end)-s)+
          (Z0m-Rtot)*(I0m(:,j(end)-s)));
    HLm = -(Z0m/(Z0m2^2))*(v0m(:,j(end)-s)+
          (Z0m-Rtot)*(I0m(:,j(end)-s)))-
          (Rtot/(Z0m2^2))*(vLm(:,j(end)-s)+
          (Z0m-Rtot)*(ILm(:,j(end)-s)));
    Hm = [-H0m;-HLm];
    Vm = ZBUS*(Hm-(-1/Rf)*(Em(:,j(end)-s)));
    v0m(:,j(end)) = Vm(1:n,1);
    vLm(:,j(end)) = Vm(n+1:2*n,1);
    I0m(:,j(end)) = (Y0m)*(v0m(:,j(end)))+H0m;
    ILm(:,j(end)) = (Y0m)*(vLm(:,j(end)))+HLm;
    ud.tension0 = (v0m);
    ud.tensionL = (vLm);
    ud.corriente0 = (I0m);
    ud.corrienteL = (ILm);
    set_param(block.BlockHandle,'UserData',ud)
    OUT = (Tl)*vLm;
    q=1;
    for i=Outurns
         block.OutputPort(q).Data = OUT(i,j(end));
         q=q+1;
    end
end
```

- Si se trata de un paso de simulación mayor y, además, la opción de archivo de salida o la opción de graficado se encuentran seleccionadas, se ingresan las tensiones de salida y el tiempo de simulación en los campos *salida* y *tiempo* de la estructura *ud*, respectivamente.

```
if block.IsMajorTimeStep
    if strcmp(ud.opcion,'on')||strcmp(ud.opcion2,'on')
         ud = get_param(block.BlockHandle,'UserData');
         ud.tiempo = [ud.tiempo block.CurrentTime];
         for i=1:n
             ud.salida = [ud.salida OUT(i,j(end))];
         end
    end
    set_param(block.BlockHandle,'UserData',ud)
end
```

- Por último se utiliza el método **Terminate**.

4.3.3 Colocación del bloque personalizado en una biblioteca

Los pasos mostrados para el caso del modelo de parámetros concentrados también se utilizan para el caso del modelo de parámetros distribuidos; sin embargo, el nombre de la S-Function que se ingresa en el campo **M-File Name** es *trans_distrib* y las variables a utilizar por el bloque en el campo **Parameters** son:

Node,Param,n,CB,LB,s,Tl,k,Z0m,Z0m2,Y0m,Rf,ZBUS,NumOut,Outurns,Rtot,File, long

4.3.4 Adición de una interfaz con el usuario a un bloque personalizado

Al igual que con el modelo de parámetros concentrados se deben seguir una serie de pasos para enmascarar el bloque:

1. Las instrucciones utilizadas para manipular la imagen del bloque y su número de puertos de salida son idénticas a las utilizadas anteriormente.

2. En el panel **Parameters** se añaden los parámetros correspondientes al modelo de parámetros distribuidos, estableciendo sus propiedades como se muestra en la Tabla 4.2.

Tabla 4.2 Propiedades de los parámetros de la S-Function para el modelo de parámetros distribuidos

Prompt	Variable	Type	Evaluate	Enable parameter	Show parameter
Parameters Calculation	Param	popup	Sí	Sí	Sí
Number of Turns	n	edit	Sí	Sí	Sí
Turn Length	long	edit	Sí	Sí	Sí
Input Node	Node	edit	Sí	Sí	Sí
Capacitance Matrix Name [F]	CB	edit	Sí	Sí	Sí
Inductance Matrix Name [H]	LB	edit	Sí	Sí	Sí
Capacitance Matrix File Name [F]	CBF	edit	No	Sí	No

Inductance Matrix File Name [H]	LBF	edit	No	Sí	No
Add Losses	Loss	checkbox	No	Sí	Sí
Series Losses per turn [ohm]	Rtot	edit	Sí	Sí	No
Select Specific Output Turn(s)	Spout	checkbox	No	Sí	Sí
Output Turn(s)	Outurns	edit	Sí	Sí	No
Save Output File	Fileopt	checkbox	No	Sí	Sí
Output File Name	File	edit	No	Sí	No
Potential Distribution Contour Plot	Meshopt	checkbox	No	Sí	Sí
Times tau	s	edit	Sí	No	No
Eigenmatrix	Tl	edit	Sí	No	No
Time Length	k	edit	Sí	No	No
Surge Impedance	Z0m	edit	Sí	No	No
Surge Impedance Modified	Z0m2	edit	Sí	No	No
Surge Admittance	Y0m	edit	Sí	No	No
Source Impedance	Rf	edit	Sí	No	No
Impedance Matrix	ZBUS	edit	Sí	No	No

El procedimiento para modificar la apariencia de la interfaz con el usuario es idéntico al utilizado en la implementación del modelo anterior.

Se escriben los comandos que inicializan las variables del modelo de parámetros distribuidos dentro del panel **Initialization.**
Después de especificar las variables de pérdidas en serie, número de puertos de salida y vueltas específicas de la forma en que se hizo para el

modelo anterior, se establecen el valor de la impedancia de la fuente y los valores de la admitancia de conexión entre vueltas y de conexión al final del devanado.

```
Rf = 0.0000001
Ycon = 1e6;
Yfin = 1e6;
```

Se inicializan las matrices que se utilizarán durante el cálculo: matriz de admitancias nodales, matriz de resistencias de fuente y matriz de admitancias de conexión.

```
YBUS = zeros(2*n);
Res = zeros(n);
Yc = zeros(n);
```

Se obtiene la matriz de valores propios de la matriz de inductancia, se calcula con ello la matriz de vectores propios de la matriz de capacitancia y ambas se utilizan para realizar las correspondientes transformaciones modales. Las operaciones de diagonalización de las matrices producen valores muy pequeños fuera de la diagonal principal que ocasionan errores en los cálculos posteriores; la aplicación doble de la instrucción **diag** resuelve este problema.

```
[Tl,Kl] = eig(LB);
Tc = (I/Tl)';
Lgm = (I/Tl)*LB*(I/Tl)';
Lgm = diag(diag(Lgm));
Cm = (I/Tc)*CB*(Tc);
Cm = diag(diag(Cm));
```

Las matrices modales de inductancia y capacitancia son diagonales, por lo que la ecuación (2.72) puede aplicarse de forma directa para obtener una matriz diagonal que contiene las impedancias características de cada vuelta del transformador. A estas impedancias se les agregan las pérdidas en serie de forma concentrada. Finalmente, la matriz se invierte para obtener la matriz de admitancias características modales.

```
Z0m = sqrt(Lgm/Cm);
Z0m2 = Z0m + Rtot;
Y0m = (I/Z0m2);
```

La solución obtenida será correcta siempre que el tiempo de viaje sea un múltiplo entero del paso de tiempo [8]. Para lograrlo, se obtiene primero el tiempo de viaje modal de cada una de las vueltas del devanado.

```
tau = diag(long.*sqrt(Lgm.*Cm));
```

Posteriormente, se obtiene el tiempo máximo de simulación y el paso de tiempo máximo ingresados en los parámetros de configuración de MATLAB-Simulink®. Se divide el tiempo de viaje entre el paso de tiempo máximo y se redondea el resultado al entero más próximo; de esta manera se obtiene el número de veces que cabe el paso de tiempo en un tiempo de viaje.

```
tmax = get_param(gcs,'StopTime');
tmax = str2num(tmax);
dt = get_param(gcs,'MaxStep');   %Ajuste del paso de tiempo y
tiempo maximo de simulacion
dt = str2num(dt);
s = round(tau(1,1)/dt);
```

Con el número entero obtenido se calcula un nuevo paso de tiempo y se convierte a una cadena de caracteres para poder ser asignado nuevamente a los parámetros de configuración.

```
dt = tau(1,1)/s;
dt = num2str(dt);
```

El tiempo máximo de simulación se divide entre el tiempo de viaje y se redondea el resultado al número entero más próximo. Con este número, se calcula de nueva cuenta el tiempo máximo de simulación, multiplicándolo por el tiempo de viaje. El resultado obtenido se convierte a una cadena de caracteres para ser asignado a los parámetros de configuración.

```
N = round(tmax/tau(1,1));
tmax = N*tau(1,1);
tmax = num2str(tmax);
%keyboard
set_param(gcs,'StopTime',tmax,'MaxStep',dt);
```

Con los nuevos valores de tiempo máximo de simulación y paso de tiempo, se establece un vector de tiempo y se obtiene su número de elementos.

```
dt = str2num(dt);
tmax = str2num(tmax);
tiempo = 0:dt:tmax;
k = length(tiempo);
```

Se realiza el armado de la matriz de admitancias nodales de acuerdo a lo establecido en la ecuación (2.76) y se realiza su inversión.

```
Y01 = Tc*Y0m*(I/Tc);
Y02 = Y01;
Res(Node,Node) = 1/Rf;
for i = 2:n
    Y01(i,i) = Y01(i,i)+Ycon;
end
Y01 = Y01+Res;
Y0m1 = (I/Tc)*Y01*Tc;
for i= 1:n-1
    Y02(i,i) = Y02(i,i)+Ycon;
end
Y02(end,end) = Y02(end,end) + Yfin;
Y0m2 = (I/Tc)*Y02*Tc;
for i=1:n-1
    Yc(i+1,i)=-Ycon;
end
Ycm = (I/Tc)*Yc*Tc;
YBUS(1:n,1:n) = Y0m1;
YBUS(1:n,n+1:2*n) = Ycm;
YBUS(n+1:2*n,1:n) = Ycm';
YBUS(n+1:2*n,n+1:2*n) = Y0m2;
ZBUS = inv(YBUS);
```

3. En el panel **Initialization** se selecciona la opción **Allow library block to modify its contents**.

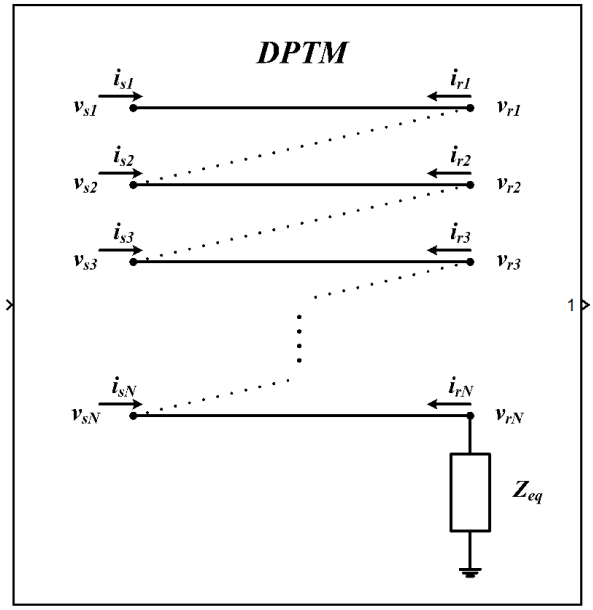
4. Dentro del panel **Documentation**, en el campo **Mask type** se ingresa el nombre del bloque personalizado: *Distributed-Parameter Transformer Model Based on Multiconductor Transmission Line Theory*.

5. Dar clic en **OK** en el editor de máscara para completar el cuadro de diálogo.

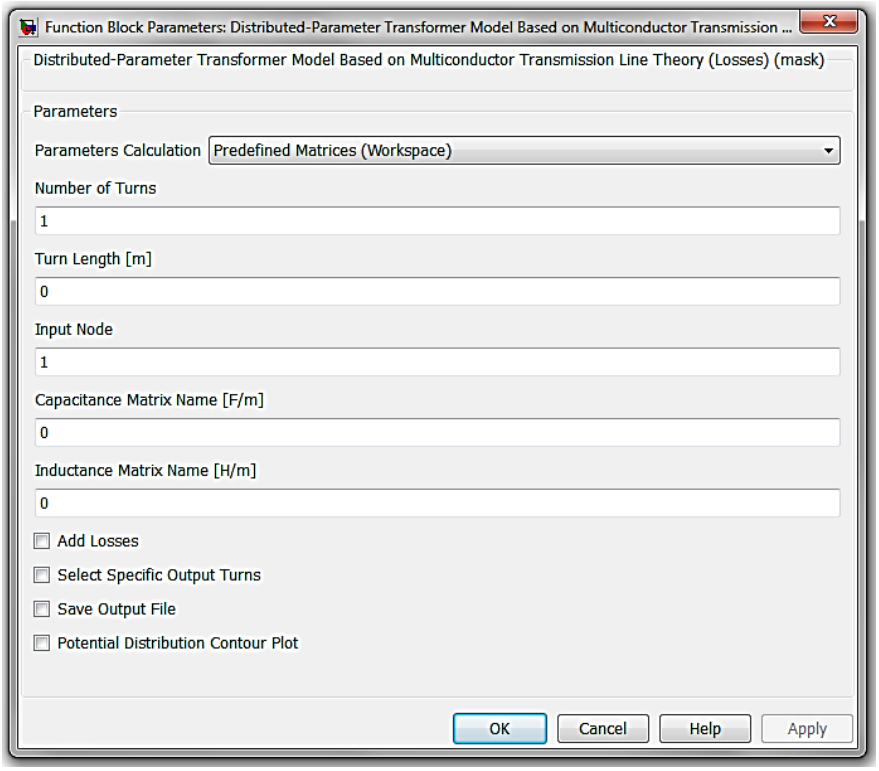
4.3.5 Agregando funcionalidad al bloque usando callbacks

El modelo de parámetros distribuidos también provee una opción para guardar un archivo de salida y una opción para realizar una gráfica de contorno de la distribución de tensión cuando la simulación termina. Para su implementación se debe utilizar el M-File: file_generation.m, y cambiar las propiedades como se describió en el modelo de parámetros concentrados.

En la Figura 4.3 se muestra el bloque implementado que representa el modelo del transformador de parámetros distribuidos basado en la teoría de la línea de transmisión multiconductora, así como la interfaz con el usuario que permite el ingreso de sus parámetros.



a)



b)

Figura 4.3 Bloque del modelo del transformador de parámetros distribuidos basado en la teoría de la línea de transmisión multiconductora. a) Símbolo. b) Interfaz

CAPÍTULO 5. PRUEBAS Y ANÁLISIS DE RESULTADOS

En este capítulo se realizan algunas pruebas de funcionamiento de las características de los modelos implementados. Ambos se utilizan en la simulación de un fenómeno transitorio rápido en el devanado de un transformador.

5.1 INTERFAZ CON EL USUARIO

Con estas pruebas se verifica que los cuadros de diálogo de ambos modelos realicen las acciones para las que fueron diseñados.

5.1.1 Selección del modo de cálculo de parámetros

Dependiendo del modo que sea elegido para el cálculo de parámetros, la interfaz con el usuario debe mostrar los campos correspondientes para cada opción. En la Figura 5.1 se muestran los cambios que ocurren para el bloque de parámetros concentrados.

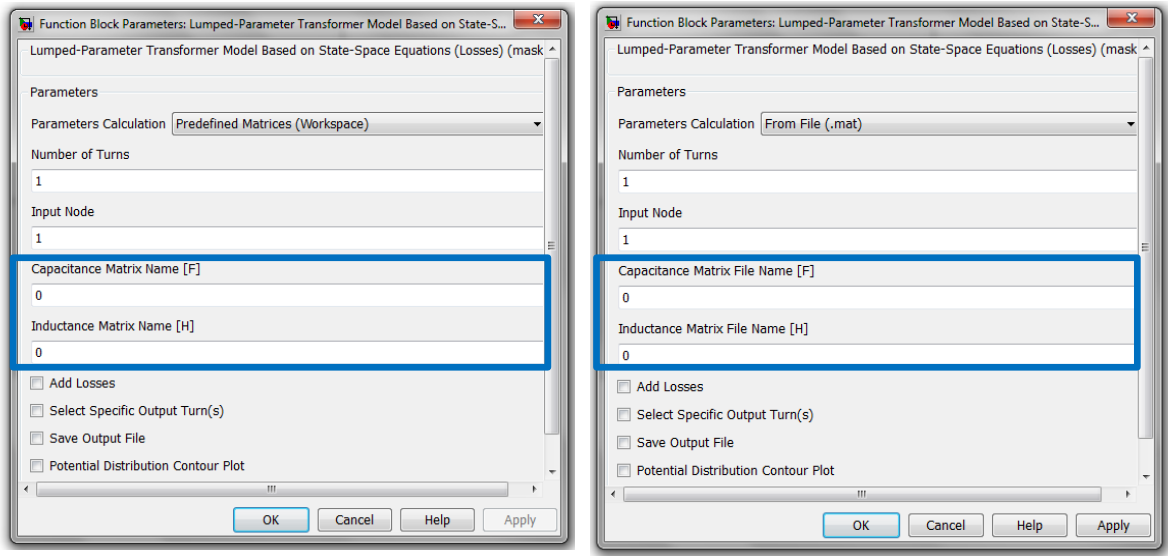


Figura 5.1 Selección del modo de cálculo de parámetros

5.1.2 Ingreso del número de vueltas

Cuando el número de vueltas del devanado se ingresa en el campo correspondiente y la opción de elegir vueltas específicas no ha sido seleccionada, el número de puertos de salida debe corresponder al total de número de vueltas.

En la Figura 5.2 se muestra el cambio que ocurre para el modelo de parámetros distribuidos.

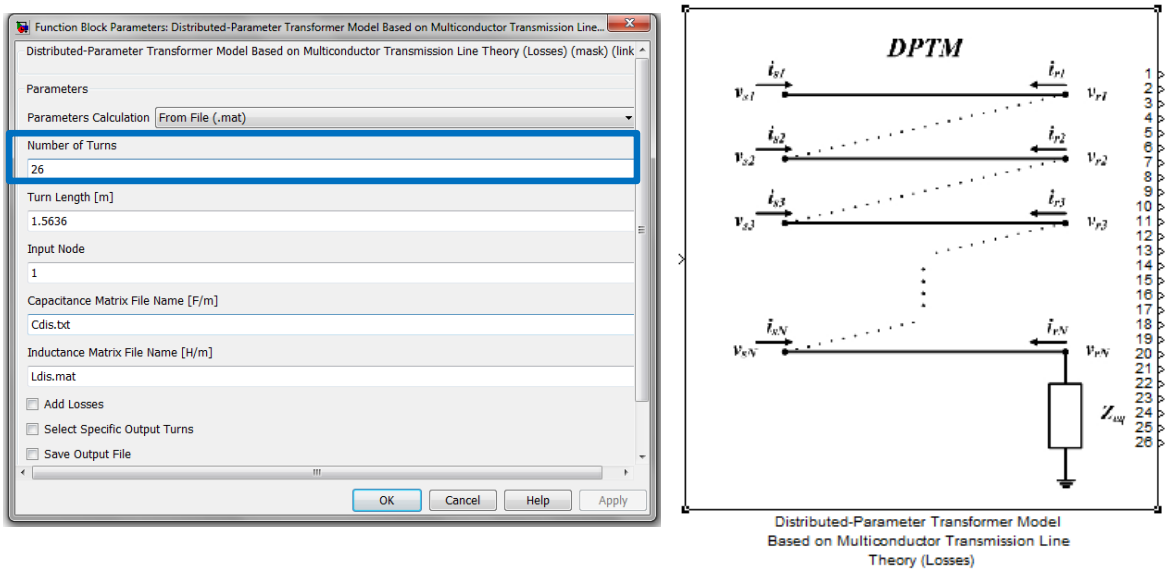


Figura 5.2 Ingreso del número de vueltas sin seleccionar la opción de vueltas específicas

Si la opción de vueltas específicas se encuentra seleccionada, el número de puertos de salida se encontrará en función de la cantidad de vueltas contenidas en el vector ingresado en el campo correspondiente, como se muestra en la Figura 5.3.

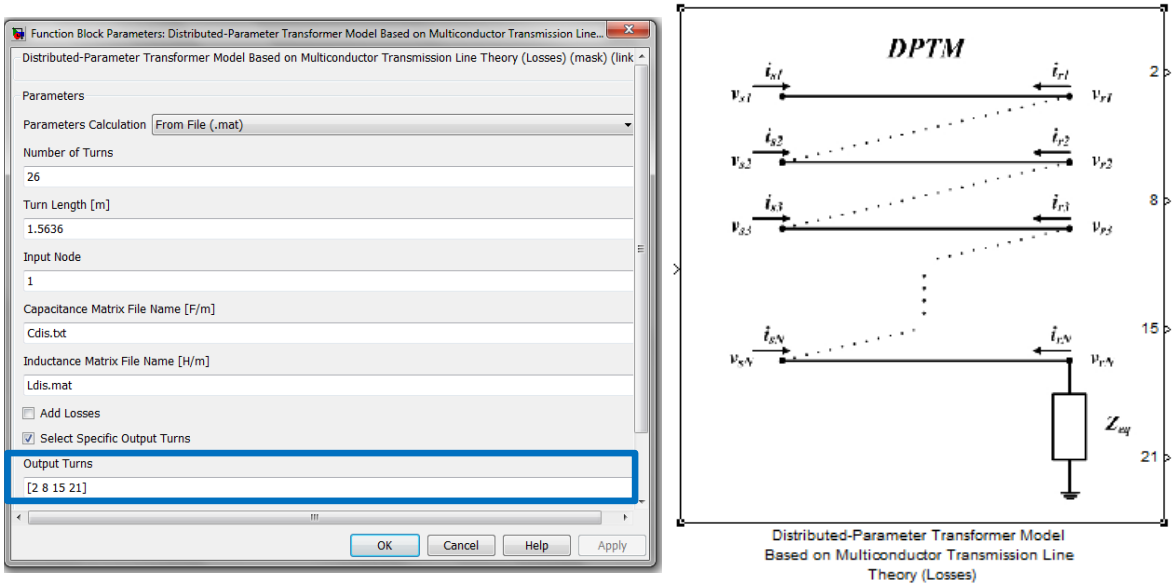


Figura 5.3 Ingreso del número de vueltas seleccionando la opción de vueltas específicas

5.1.3 Desplegado de errores

Si en alguno de los campos de la interfaz con el usuario de ambos modelos se ingresa un parámetro de forma incorrecta o se deja vacío, los modelos muestran un cuadro con un mensaje de error, como se muestra en la Figura 5.4.

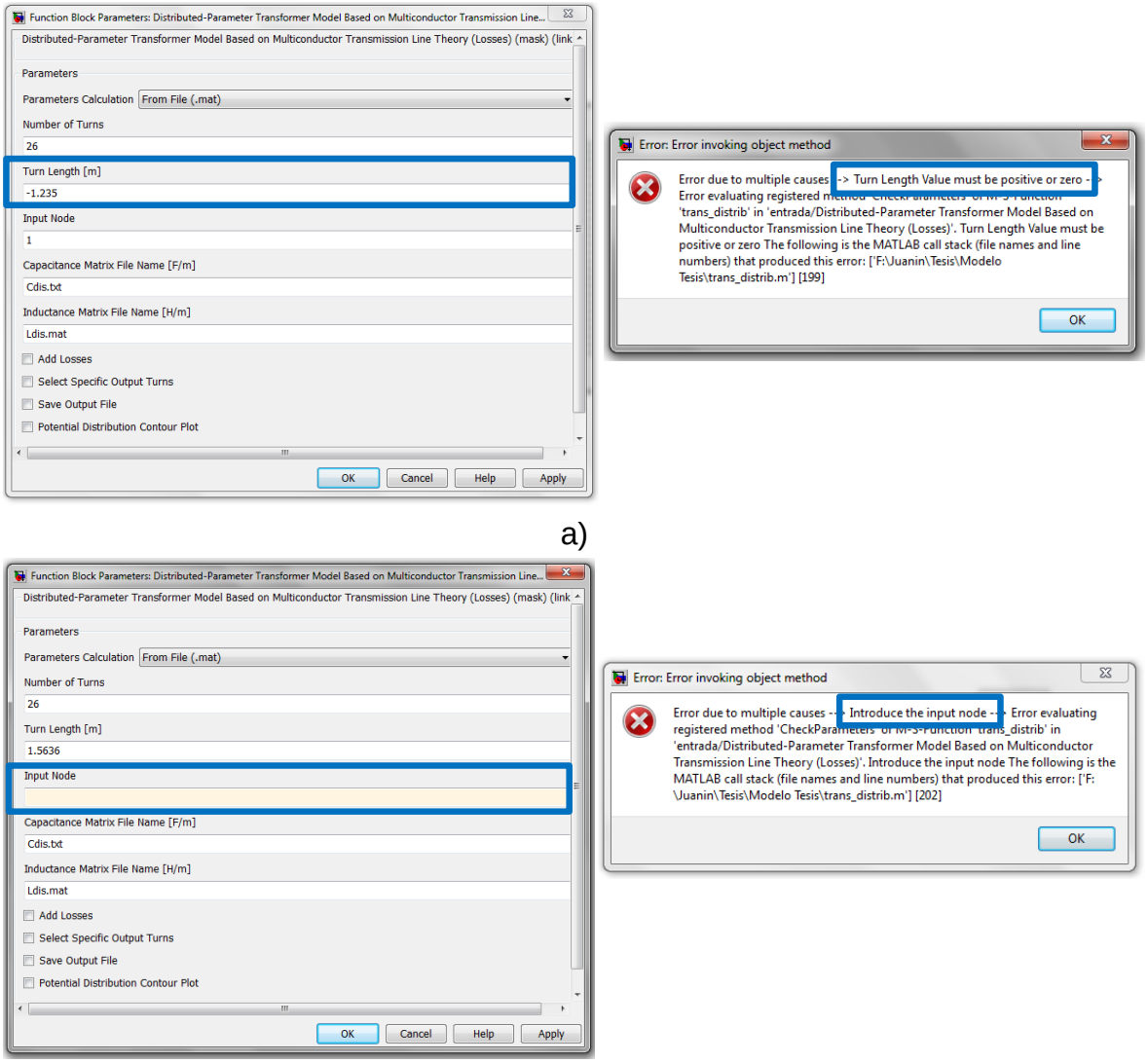


Figura 5.4 Desplegado de errores. a) Parámetro ingresado de forma incorrecta. b) Campo vacío

5.2 FUNCIONAMIENTO

Para verificar el funcionamiento correcto de ambos modelos implementados se utiliza un transformador de 22 discos, cada uno con 13 vueltas. La representación esquemática de los primeros dos discos del transformador se muestra en la Figura

5.5. La configuración geométrica y parámetros de este ejemplo se tomaron de [48].

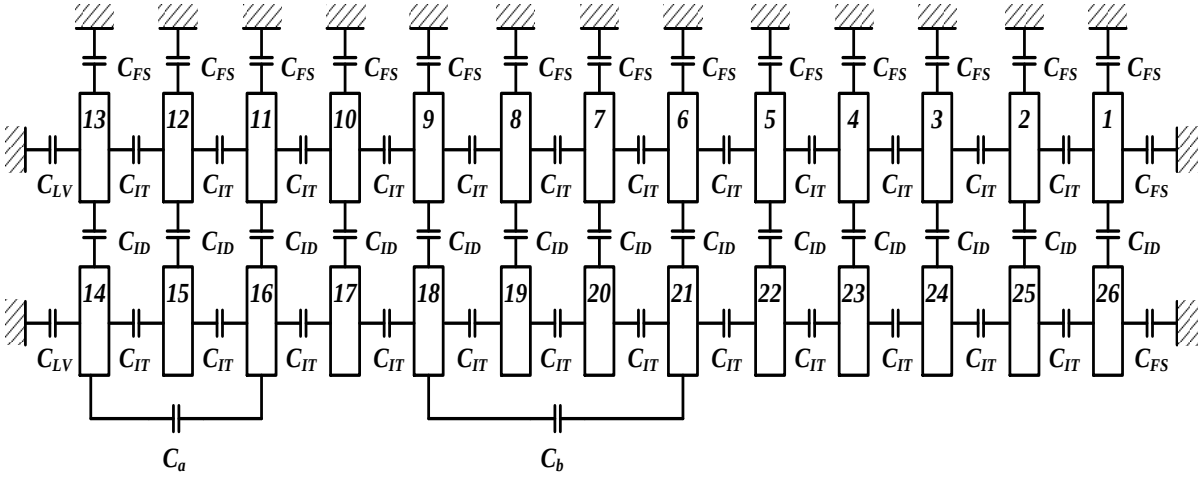


Figura 5.5 Representación esquemática de los primeros dos discos del transformador de prueba [48]

De la figura se tiene:

C_{IT} = Capacitancia entre vueltas
C_a = Capacitancia entre dos vueltas separadas por una vuelta entre ellas
C_b = Capacitancia entre dos vueltas separadas por dos vueltas entre ellas
C_{ID} = Capacitancia entre dos discos adyacentes
C_{FS} = Capacitancia entre una vuelta y tierra
C_{LV} = Capacitancia entre los devanados de alta y baja tensión

La matriz de capacitancias puede obtenerse por el mismo procedimiento utilizado para el cálculo de una matriz de admitancias. Para el transformador utilizado, los elementos de $[\boldsymbol{C}]$ están dados por:

$$C_{1,1} = C_{286,286} = 2C_{FS} + C_a + C_b + C_{ID} + C_{IT} \quad (5.1)$$

$$C_{2,2} = C_{12,12} = C_{275,275} = C_{285,285} = C_{FS} + C_a + C_b + C_{ID} + 2C_{IT} \quad (5.2)$$

$$C_{3,3} = C_{11,11} = C_{276,276} = C_{284,284} = C_{2,2} + C_a \quad (5.3)$$

$$C_{4,4} = \cdots = C_{10,10} = C_{277,277} = \cdots = C_{283,283} = C_{FS} + 2C_a + 2C_b + C_{ID} + 2C_{IT} \quad (5.4)$$

$$C_{13,13} = C_{274,274} = C_{FS} + C_a + C_b + C_{ID} + C_{IT} + C_{LV} \quad (5.5)$$

$$C_{14,14} = \cdots = C_{273,273} = C_{IT} + 2C_{ID} + C_a + C_b + C_{LV} \quad (5.6)$$

$$C_{26,26} = \cdots = C_{261,261} = C_{FS} + C_{IT} + 2C_{ID} + C_a + C_b \quad (5.7)$$

$$C_{15,15} = \cdots = C_{272,272} = C_{25,25} = \cdots = C_{262,262} = 2C_{IT} + 2C_{ID} + C_a + C_b \quad (5.8)$$

$$C_{16,16} = \cdots = C_{271,271} = C_{24,24} = \cdots = C_{263,263} = 2C_{IT} + 2C_{ID} + 2C_a + C_b \quad (5.9)$$

$$C_{17,17} = \cdots = C_{270,270} = C_{23,23} = \cdots = C_{264,264} = 2C_{IT} + 2C_{ID} + 2C_a + 2C_b \quad (5.10)$$

$$C_{1,2} = C_{2,3} = \cdots = C_{25,26} = -C_{IT} \quad (5.11)$$

$$C_{1,3} = C_{2,4} = \cdots = C_{24,26} = -C_a \quad (5.12)$$

$$C_{1,4} = C_{2,5} = \cdots = C_{10,13} = -C_b \quad (5.13)$$

$$C_{1,26} = C_{2,25} = \cdots = C_{26,1} = -C_{ID} \quad (5.14)$$

$$C_{12,14} = C_{11,14} = C_{13,15} = C_{13,16} = 0 \quad (5.15)$$

Los valores de las capacitancias utilizadas en las simulaciones se muestran en la Tabla 5.1.

Tabla 5.1 Valores de capacitancia utilizados para el arreglo esquemático de la Figura 5.5 [48].

$C_{ID} = 4.7928\ [\mathrm{pF/m}]$	$C_a = 284.34\ [\mathrm{pF/m}]$
$C_{FS} = 20.943\ [\mathrm{pF/m}]$	$C_b = 189.56\ [\mathrm{pF/m}]$
$C_{IT} = 568.68\ [\mathrm{pF/m}]$	$C_{LV} = 4.3690\ [\mathrm{pF/m}]$

La matriz de inductancias se calcula utilizando la siguiente ecuación:

$$[L] = L_0[C]^{-1}\ [\mathrm{H/m}] \quad (5.16)$$

donde:

$$L_0 = (1/v_0)^2 = 2.1697 \times 10^{-17} \quad (5.17)$$

La matriz de pérdidas en serie se encuentra conformada de la siguiente manera:

$$[R] = 0.3049[U]\ [\Omega/\mathrm{m}] \quad (5.18)$$

donde $[U]$ es la matriz identidad.

Para las simulaciones realizadas se aplicó una onda con forma de doble rampa lineal, la cual se encuentra definida en [7] como:

$$A(t) = \alpha_1 t u(t) - \alpha_2 (t - t_f) u(t - t_f) \quad (5.19)$$

donde:

$$\alpha_1 = \frac{A_0}{t_f}, \alpha_2 = \frac{2t_h - t_f}{2t_f(t_h - t_f)} A_0 \tag{5.20}$$

$t_f =$ tiempo de frente de onda $(60\ \text{ns})$; $t_h =$ tiempo de decaimiento al valor medio $(25\ \mu\text{s})$, y $A_0 =$ valor pico $(1\ \text{V})$.

Las matrices de inductancia y capacitancia utilizadas por ambos modelos se guardaron en archivos con extensiones .mat y .txt, respectivamente, en la carpeta de trabajo de MATLAB®

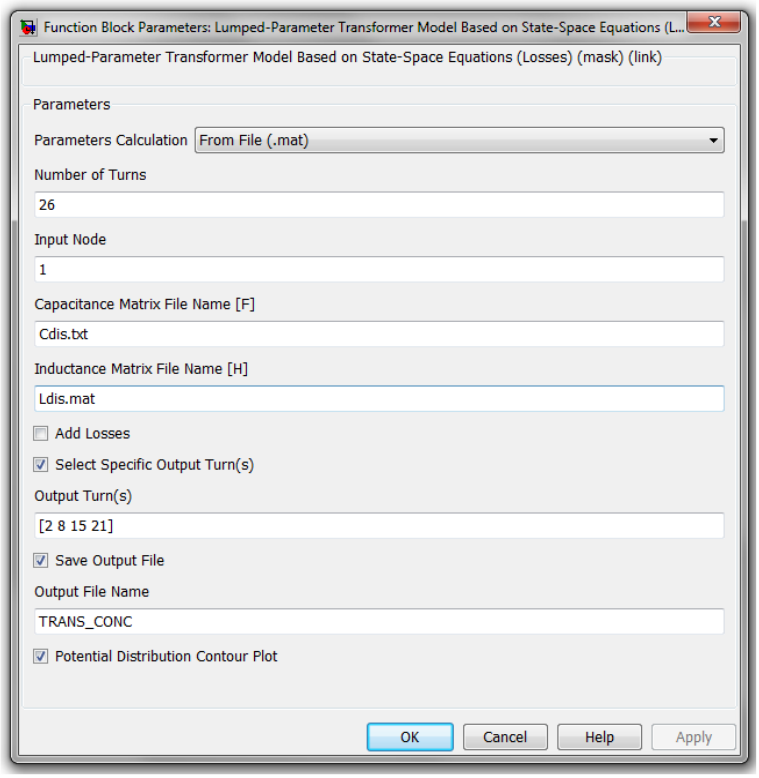
Se realizaron dos simulaciones por medio de las cuales se compararon ambos modelos. En la primera de ellas, las pérdidas en serie se desprecian. Se considera una longitud de vuelta media de $1.5636\ \text{m}$.

Las interfaces con el usuario de cada uno de los modelos, una vez ingresados los parámetros necesarios para la simulación, se muestran en la Figura 5.6. Los resultados que contienen las tensiones de todas las vueltas se guardan en los archivos *TRANS_CONC* y *TRANS_DIS*. Además, se obtienen las tensiones específicas de las vueltas 2, 8, 15 y 21 para su comparación con los resultados almacenados en los archivos.

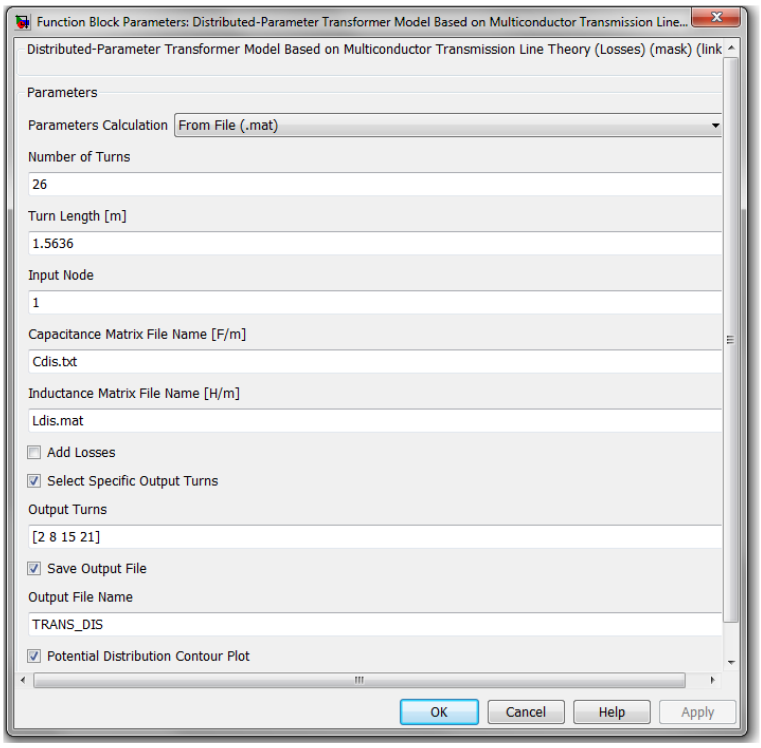
El diagrama utilizado para la simulación dentro de MATLAB-Simulink® se muestra en la Figura 5.7. Las vueltas específicas de cada uno de los modelos se guardan en archivos de salida con extensión .mat por medio del bloque que se encuentra a la derecha. El bloque de la izquierda contiene el vector de valores que representa a la señal con forma de doble rampa lineal.

Los valores de las admitancias de conexión se consideran con un valor de $1 \times 10^6\ \text{S/m}$, con el propósito de conservar la continuidad del devanado, y el final del devanado se encuentra sólidamente aterrizado. El método numérico utilizado para la solución es regla trapezoidal, con un tiempo de observación total de $4.37\ \mu\text{s}$ y un paso de tiempo máximo de $0.72833\ \text{ns}$.

Una vez terminada la simulación se puede acceder a los archivos de salida desde la carpeta de trabajo de MATLAB®.



a)



b)

Figura 5.6 Interfaces con el usuario con los parámetros necesarios para la simulación. a) Modelo de parámetros concentrados. b) Modelo de parámetros distribuidos

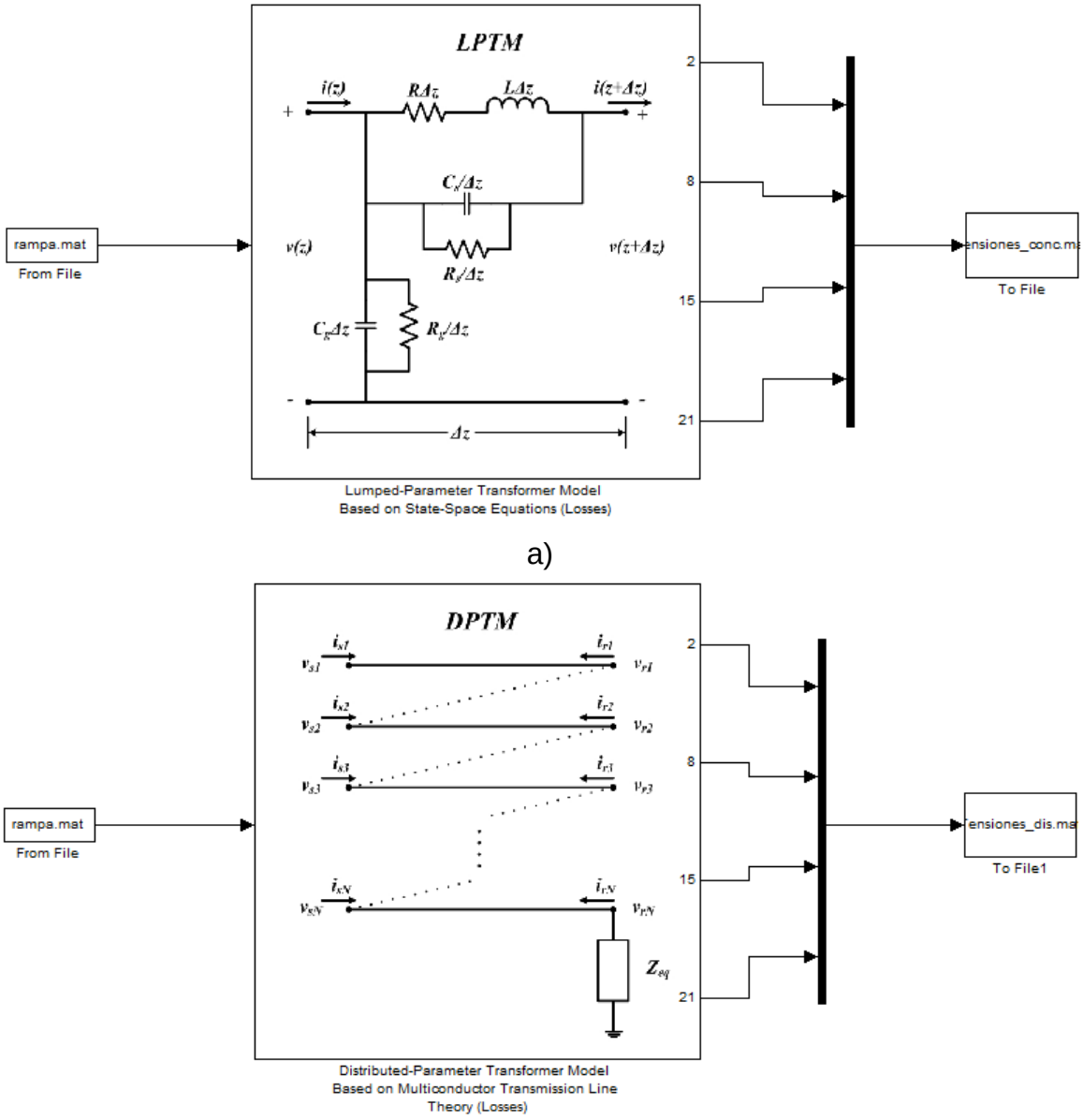


Figura 5.7 Diagrama utilizado para la simulación dentro de MATLAB-Simulink®. a) Modelo de parámetros concentrados. b) Modelo de parámetros distribuidos

En la Figura 5.8 se muestran las tensiones transitorias en cinco vueltas diferentes del devanado. Se observa una gran similitud entre los resultados de los modelos, aunque el modelo de parámetros concentrados no es capaz de reproducir algunos de las oscilaciones de tensión relacionadas con las mayores frecuencias presentes en el fenómeno, como se muestra en el acercamiento de la Figura 5.9.

En la Figura 5.10 se muestran las tensiones transitorias para las mismas cinco vueltas del devanado, tomando en consideración las pérdidas en serie.

Comparando con la Figura 5.8, es difícil apreciar la atenuación en las formas de onda obtenidas, por lo que en la Figura 5.11 se muestra la tensión transitoria de las cinco vueltas del devanado considerando ambos casos, para los dos modelos utilizados. Se observa que, al incluir las pérdidas, los resultados de ambos modelos son prácticamente iguales.

En la Tabla 5.2 se muestran los valores máximos de tensiones transitorias en el devanado del transformador para cada una de las simulaciones realizadas

Tabla 5.2 Valores máximos de tensiones transitorias en el devanado del transformador

Modelo de parámetros concentrados		**Modelo de parámetros distribuidos**	
Ideal	Pérdidas en serie	Ideal	Pérdidas en serie
1.3860 [V]	1.3575 [V]	1.3920 [V]	1.3532 [V]

Los resultados obtenidos muestran que existen tensiones transitorias que exceden el valor máximo de la señal de entrada principalmente en las primeras vueltas del devanado del transformador. Estos valores proporcionan información importante en la etapa de diseño del transformador, ya que permiten conocer y localizar el valor máximo de tensión al que se encontrará expuesto el devanado, y con ello realizar las modificaciones necesarias para disminuir el esfuerzo dieléctrico entre vueltas ocasionado por fenómenos transitorios rápidos.

Como lo muestran los datos de la Tabla 5.2, el modelo de parámetros concentrados proporciona valores de tensión que se encuentran ligeramente por debajo de aquellos obtenidos por medio del modelo de parámetros distribuidos. En los casos donde no sea necesario realizar un análisis muy detallado de las sobretensiones presentes en el devanado del transformador, el modelo de parámetros concentrados representa una ventaja considerable en cuanto al tiempo de cómputo necesario para obtener los resultados de las simulaciones.

Aún cuando los tiempos de cómputo se encuentran relacionados con el equipo de cómputo utilizado, cada uno de los modelos requiere realizar procesos que extienden el tiempo de simulación. Para el modelo de parámetros distribuidos esto es provocado por el cálculo de matrices de vectores propios y por la cantidad de pasos de tiempo contenidos en el tiempo de viaje. En el modelo de parámetros concentrados lo anterior se encuentra determinado por el método de integración utilizado para resolver el sistema de ecuaciones diferenciales.

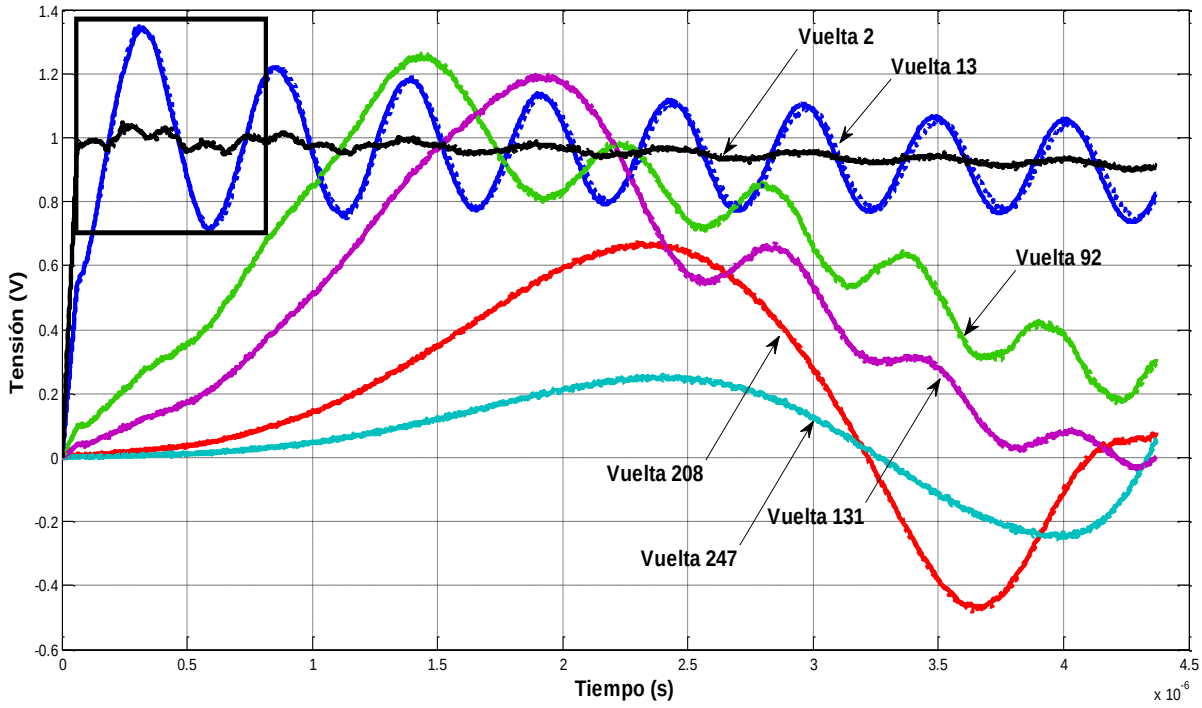


Figura 5.8 Comparación de la tensión transitoria en cinco vueltas diferentes del devanado del transformador sin pérdidas. Modelo de parámetros concentrados (línea sólida). Modelo de parámetros distribuidos (línea discontinua)

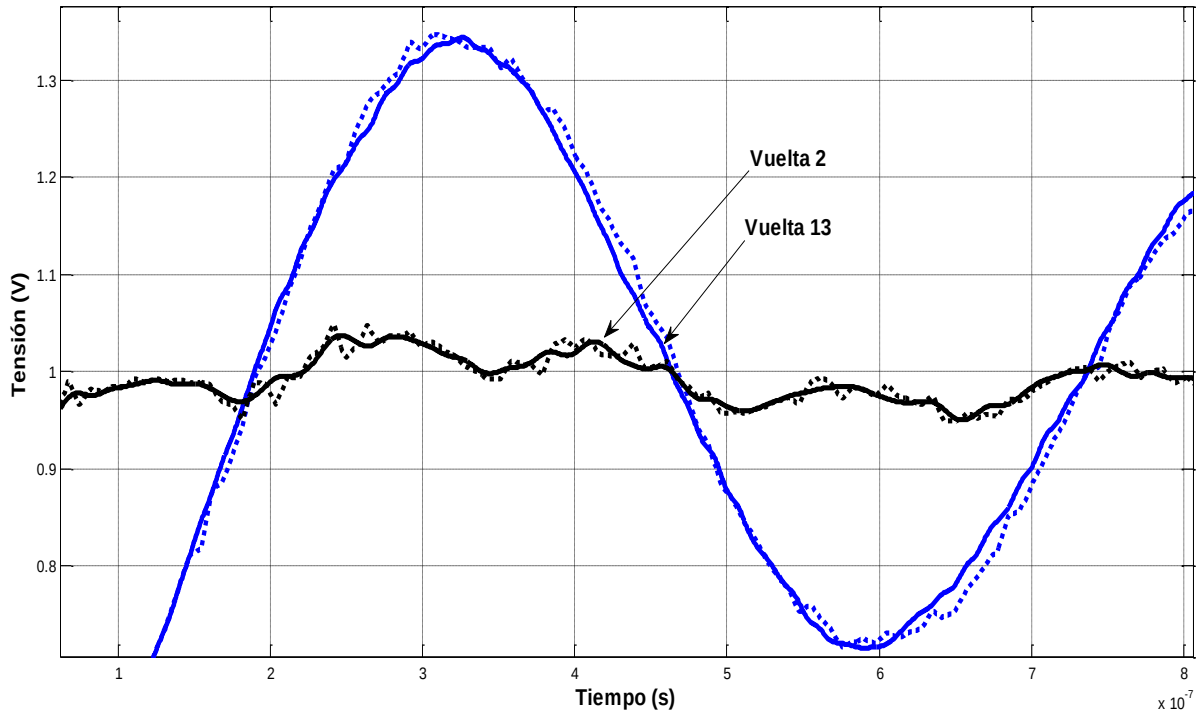


Figura 5.9 Acercamiento de la comparación de la tensión transitoria en las vueltas 2 y 13 del devanado del transformador sin pérdidas. Modelo de parámetros concentrados (línea sólida). Modelo de parámetros distribuidos (línea discontinua)

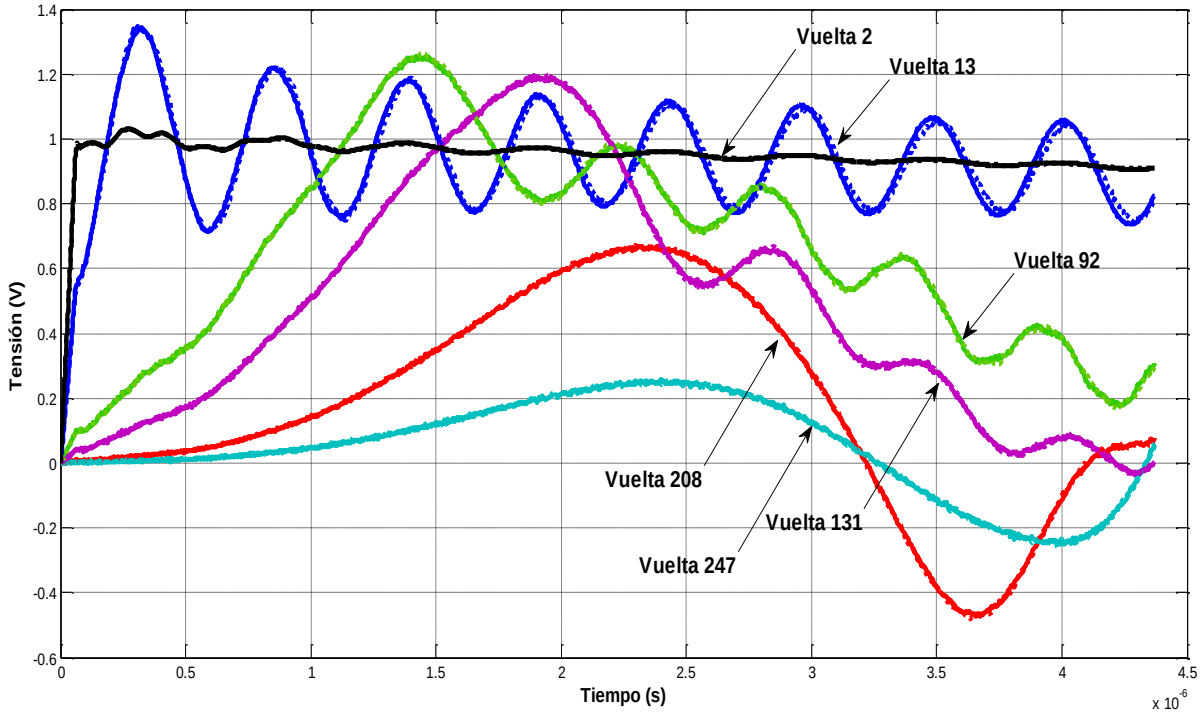
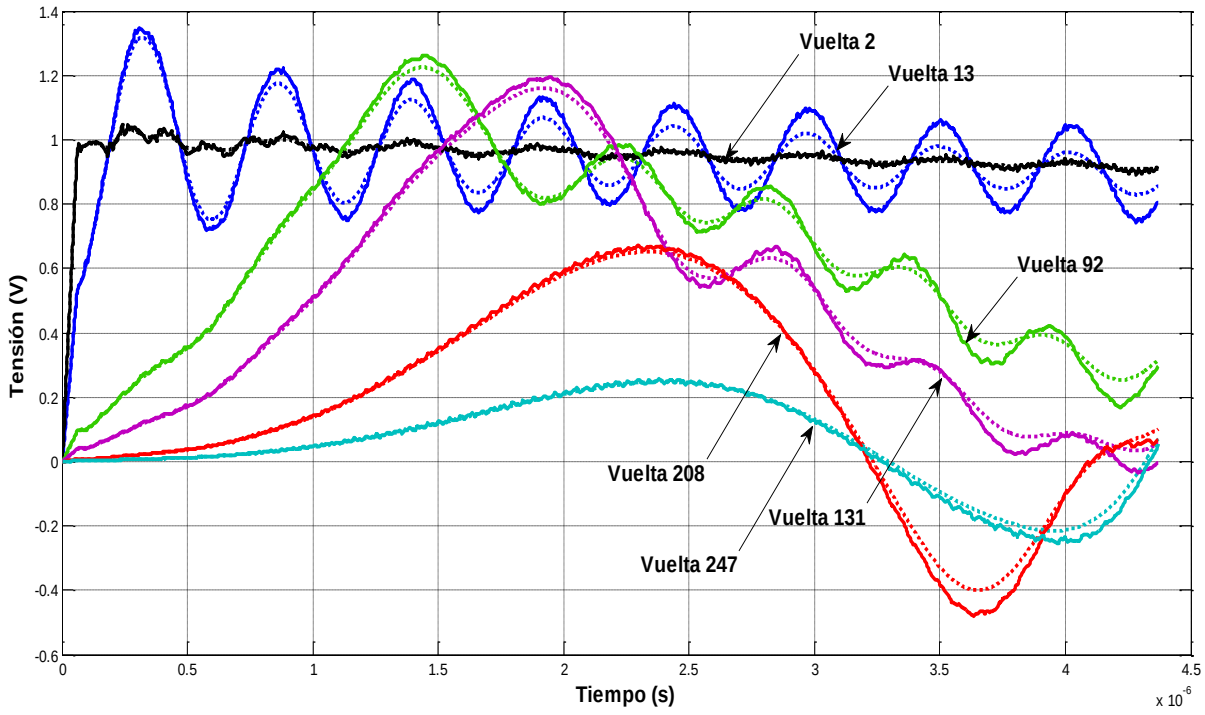
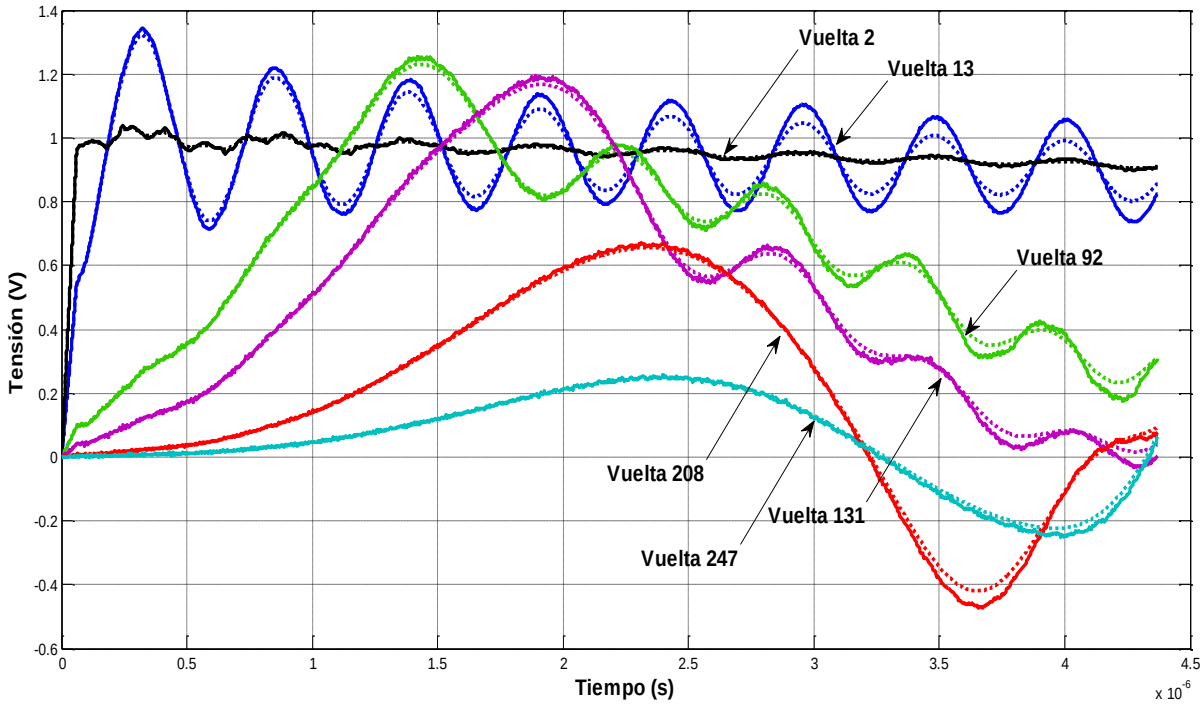


Figura 5.10 Comparación de la tensión transitoria en cinco vueltas diferentes del devanado del transformador con pérdidas. Modelo de parámetros concentrados (línea sólida). Modelo de parámetros distribuidos (línea discontinua)



a)



b)

Figura 5.11 Comparación de la tensión transitoria en cinco vueltas del devanado del transformador considerando las pérdidas en serie. a) Modelo de parámetros distribuidos. b) Modelo de parámetros concentrados. Caso ideal (línea sólida). Caso con pérdidas (línea discontinua)

En la Tabla 5.3 se muestran los tiempos de cómputo utilizados por cada uno de los modelos implementados. Las simulaciones se realizaron en una computadora con un procesador Intel Core i3-2120 3.30 GHz y una memoria RAM de 2 GB.

Tabla 5.3 Tiempos de cómputo de los modelos implementados

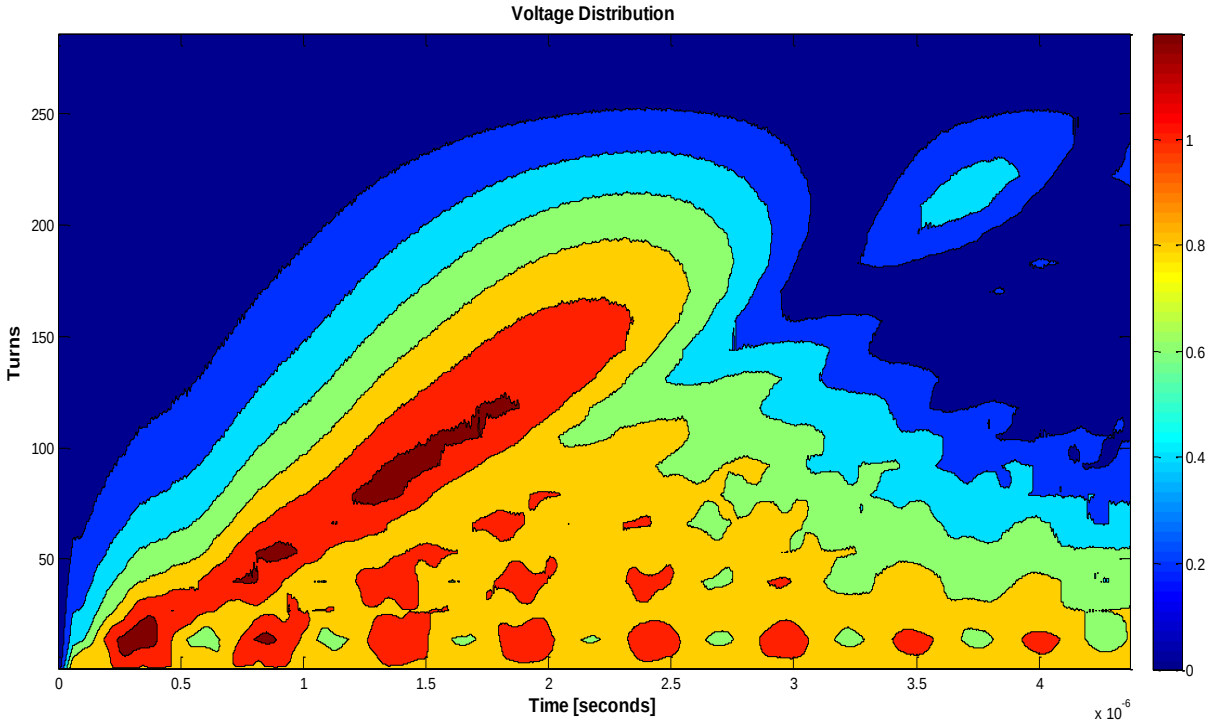
Modelo de parámetros concentrados	Modelo de parámetros distribuidos
2 horas 8 minutos	2 horas 44 minutos

Cabe mencionar que para las simulaciones realizadas, el número de muestras obtenido en el resultado del modelo de parámetros concentrados no coincide con el del modelo de parámetros distribuidos. Esto se debe principalmente a que el modelo de parámetros concentrados, al resolver un sistema de ecuaciones diferenciales por medio de un método de integración de paso variable, necesita ajustar el paso de tiempo de acuerdo al grado de oscilación que presenta la curva de solución, es decir, requerirá un paso de tiempo más pequeño en las zonas donde la curva se encuentre caracterizada por oscilaciones rápidas y un paso de tiempo mayor para las zonas donde la curva sea más suave. Por otro lado, en el modelo de parámetros distribuidos el método de integración no se utiliza para

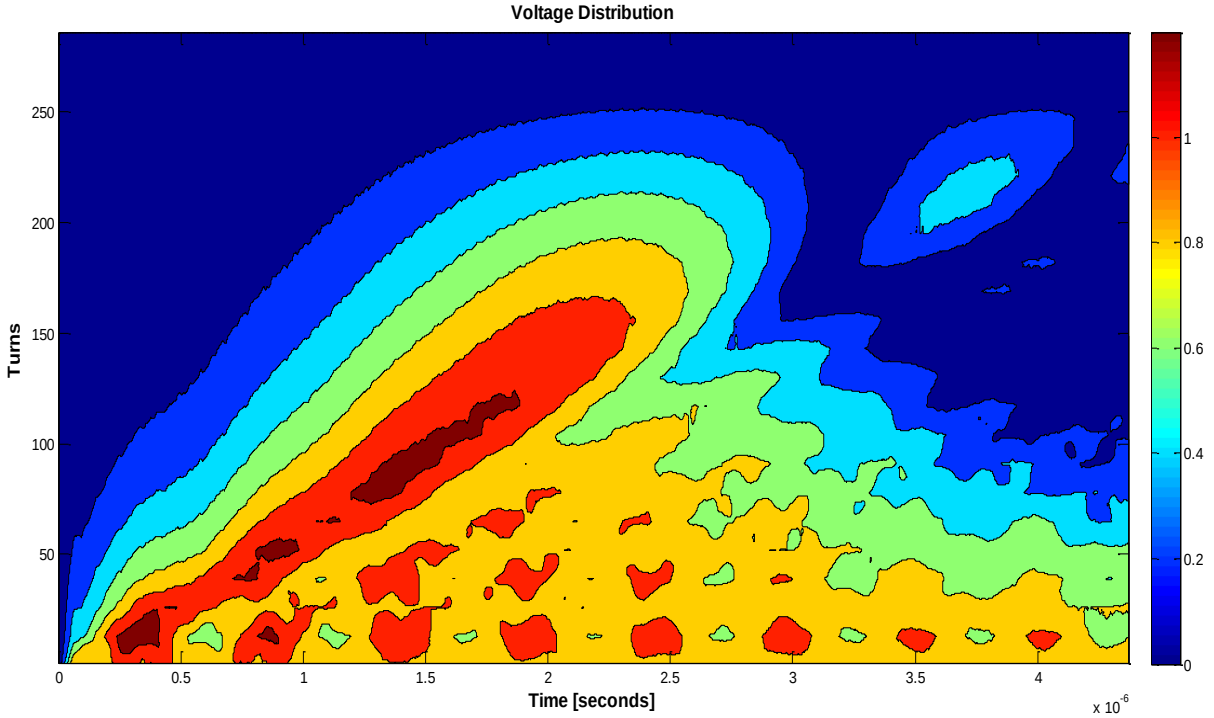
obtener la solución debido a que se encuentra definido por ecuaciones algebraicas y, por lo tanto, el valor del paso de tiempo se mantiene constante.

Finalmente, en las Figura 5.12 y Figura 5.13 se muestran las gráficas de contorno obtenidas para ambas simulaciones. En ellas se muestran los valores de tensión presentes en cada una de las vueltas del transformador durante el tiempo de simulación. Puede observarse que para ambos casos y ambos modelos existen sobretensiones transitorias en las primeras vueltas del devanado en un tiempo cercano al inicio de la simulación. También se observa que para un tiempo mayor a $2\ \mu s$ así como en las vueltas finales del devanado, las tensiones no sobrepasan el valor máximo de la señal de entrada.

Adicionalmente, en estas gráficas se aprecia la atenuación que existe en las tensiones de cada una de las vueltas durante el tiempo de simulación al incluir la componente de pérdidas en serie. Esto provoca que las sobretensiones transitorias aparezcan en menos vueltas del devanado y que disminuyan su valor en un periodo de tiempo más corto. Además, se puede verificar que al no contener las pérdidas en serie, los modelos presentan un mayor número de oscilaciones en los resultados obtenidos.

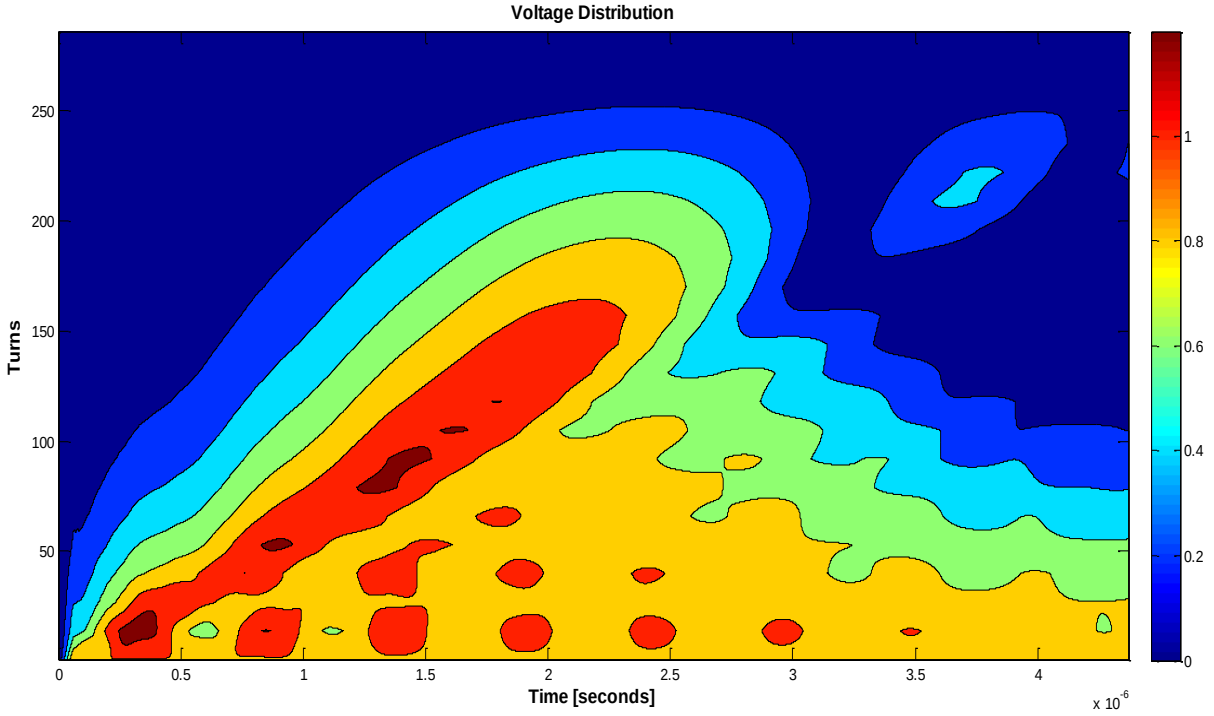


a)

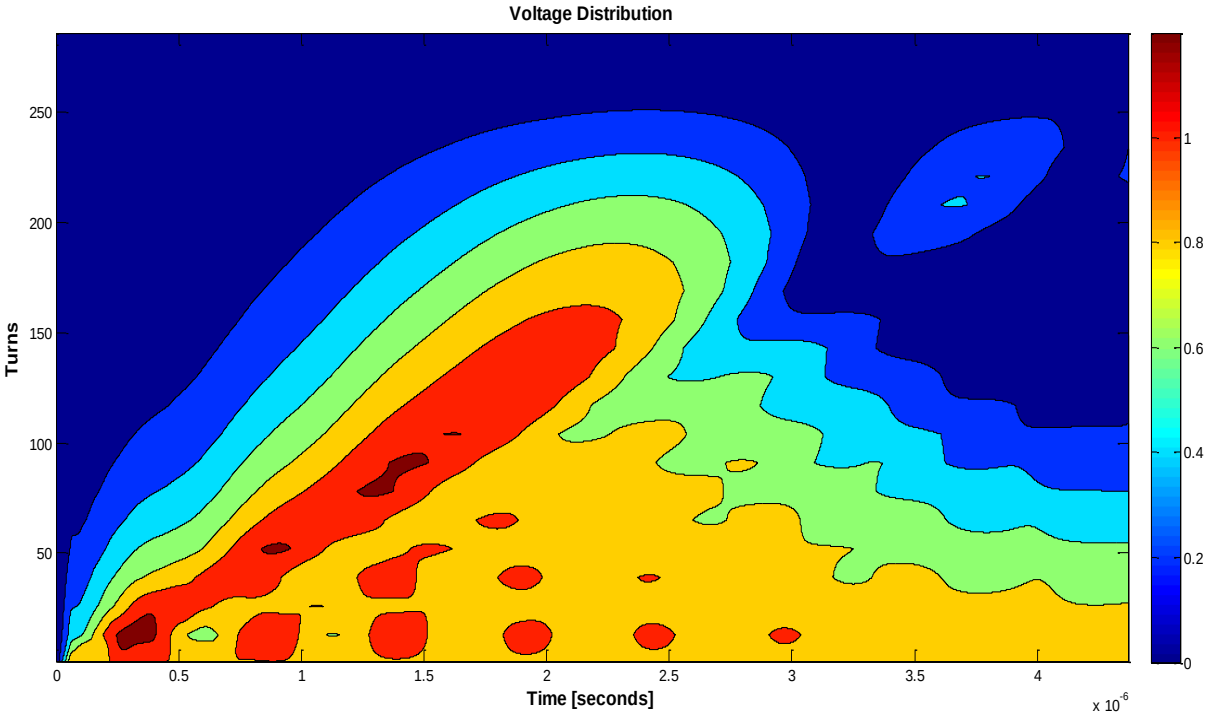


b)

Figura 5.12 Gráficas de contorno de la tensión transitoria en las vueltas del devanado del transformador sin pérdidas. a) Modelo de parámetros concentrados. b) Modelo de parámetros distribuidos



a)



b)

Figura 5.13 Gráficas de contorno de la tensión transitoria en las vueltas del devanado del transformador con pérdidas. a) Modelo de parámetros concentrados. b) Modelo de parámetros distribuidos

CAPÍTULO 6. CONCLUSIONES

A continuación se presentan las conclusiones obtenidas de la realización del presente trabajo, así como las recomendaciones para trabajos futuros.

6.1 CONCLUSIONES

Se realizó la implementación directa de dos modelos del transformador para transitorios electromagnéticos rápidos en el programa comercial de simulación MATLAB-Simulink®. El primero de ellos es un modelo de parámetros concentrados basado en ecuaciones de estado; el segundo es un modelo de parámetros distribuidos basado en la teoría de la línea de transmisión multiconductora.

Ambos modelos fueron utilizados en el análisis de un transitorio electromagnético de alta frecuencia aplicado en el devanado de un transformador. Para ello se realizaron dos simulaciones: la primera de ellas considerando el caso ideal y la segunda tomando en consideración la componente de pérdidas en serie de cada vuelta del devanado.

Observando los resultados obtenidos, puede constatarse que los dos modelos implementados cumplen satisfactoriamente todas las funciones para las que fueron diseñados, tales como:

- Ingreso de parámetros por medio de una variable definida en el espacio de trabajo de MATLAB® o por medio de un archivo con extensión .mat o .txt.
- Modificación del número de puertos de salida en forma proporcional al número de vueltas con las que cuenta el devanado.
- Inclusión la componente de pérdidas en serie.
- Selección de vueltas específicas de las que se requiere obtener la señal de salida.
- Guardado de un archivo de salida.
- Realización de una gráfica de contorno de la distribución de tensión de las vueltas del devanado a lo largo del tiempo de simulación.

Cabe mencionar que las restricciones establecidas en el caso del ingreso incorrecto de alguno de los parámetros necesarios para la simulación funcionan adecuadamente, desplegando mensajes de error que indican cuál es la falla.

Comparando las formas de tensión obtenidas por cada uno de los modelos implementados, se puede afirmar que existe una gran similitud entre ambos resultados, aunque para el caso ideal, el modelo de parámetros concentrados no es capaz de reproducir algunos picos de tensión relacionados con las mayores frecuencias presentes en el fenómeno. Para el caso en el que se consideran las pérdidas en serie, los resultados son prácticamente iguales.

En los resultados obtenidos por ambos modelos existen tensiones transitorias que sobrepasan el valor máximo de la señal de entrada; estas sobretensiones pueden producir esfuerzos eléctricos considerables en el sistema de aislamiento del transformador y originar una falla dentro del mismo. El correcto análisis de los resultados propocionados por las simulaciones provee información importante para el diseño y construcción de transformadores.

6.2 APORTACIONES

- Se obtienen dos nuevos bloques en MATLAB-Simulink® que representan modelos de transformador para transitorios electromagnéticos rápidos en el dominio del tiempo.

- Se obtienen gráficas de contorno de la distribución de tensión, las cuales constituyen una herramienta importante en la localización de esfuerzos dieléctricos a lo largo del devanado del transformador.

- Se obtiene una metodología básica para la inclusión a futuro de nuevos modelos desarrollados por los usuarios de MATLAB-Simulink®.

6.3 RECOMENDACIONES PARA TRABAJOS FUTUROS

- Incluir en los modelos la componente de pérdidas en derivación y los parámetros de pérdidas dependientes de la frecuencia: pérdidas por efecto piel y pérdidas por efecto de proximidad.

- Incluir el método de imágenes como una opción para el cálculo de parámetros en los modelos implementados, ya que la obtención de estos por medio de programas de simulación de campos electromagnéticos basados en el método de elemento finito es computacionalmente costosa.

- Realizar la validación de los resultados de los modelos implementados mediante comparaciones con resultados obtenidos experimentalmente.

REFERENCIAS

[1] J. A. Martínez-Velasco, Ed., Power System Transients: Parameter Determination, CRC Press, 2010.

[2] A. Greenwood, Electrical Transients in Power Systems, John Wiley & Sons, Inc., 1991.

[3] N. Watson y J. Arrillaga, Power Systems Electromagnetic Transients Simulation, The Institution of Engineering and Technology, 2007.

[4] J. Mahseredjian, V. Dinavahi y J. A. Martinez, «Simulation Tools for Electromagnetic Transients in Power Systems: Overview and Challenges,» *IEEE Transactions on Power Delivery,* vol. 24, nº 3, pp. 1657-1669, 2009.

[5] H. W. Dommel y W. Scott, «Computation of Electromagnetic Transients,» *Proceedings of the IEEE,* vol. 62, nº 7, pp. 983-993, 1974.

[6] W. Derek Humpage y K.-P. Wong, «Electromagnetic Transient Analysis in EHV Power Networks,» *Proceedings of the IEEE,* vol. 70, nº 4, pp. 379-402, 1982.

[7] P. Chowdhuri, Electromagnetic Transients in Power Systems, Hertfordshire: Research Studies Press Ltd., 2004.

[8] B. P. Administration, Electro-Magnetic Transients Program Theory Book, Portland: EMTP User Group, 1995.

[9] N. Nagaoka y A. Ametani, «A Development of a Generalized Frequency-Domain Transient Program,» *IEEE Transactions on Power Delivery,* vol. 3, nº 4, pp. 1996-2004, 1988.

[10] P. Moreno y A. Ramírez, «Implementation of the Numerical Laplace Transform: A Review,» *IEEE Transactions on Power Delivery,* vol. 23, nº 4, pp. 2599-2609, 2008.

[11] T. Noda y A. Ramírez, «z-Transform-Based Methods for Electromagnetic Transient Simulations,» *IEEE Transactions on Power Delivery,* vol. 22, nº 3, pp. 1799-1805, 2007.

[12] L. Rabins, «A New Approach to the Analysis of Impulse Voltages and Gradients in Transformer Windings,» *Power Apparatus and Systems, Part III. Transactions of the American Institute of Electrical Engineers,* vol. 78, nº 4, pp. 1784-1791, 1959.

[13] J. L. Guardado y K. J. Cornick, «A Computer Model for Calculating Steep-Fronted Surge Distribution in Machine Windings,» *IEEE Transactions on Energy Conversion,* vol. 4, nº 1, pp. 95-101, 1989.

[14] A. Miki, T. Hosoya y K. Okuyama, «A Calculation Method for Impulse Voltage

Distribution and Transferred Voltage in Transformer Windings,» *IEEE Transactions on Power Apparatus and Systems,* vol. PAS-97, nº 3, pp. 930-939, 1978.

[15] Y. Shibuya, S. Fujita y N. Hosokawa, «Analysis of Very Fast Transient Overvoltage in Transformer Winding,» *IEE Proceedings-Generation, Transmission and Distribution,* vol. 144, nº 5, pp. 461-468, 1997.

[16] B. Gustavsen y A. Semlyen, «Application of Vector Fitting to State Equation Representation of Transformers for Simulation of Electromagnetic Tranients,» *IEEE Transactions on Power Delivery,* vol. 13, nº 3, pp. 834-842, 1998.

[17] Y. Shibuya, S. Fujita y E. Tamaki, «Analysis of Very Fast Transients in Transformers,» *IEE Proceedings-Generation, Transmission and Distribution ,* vol. 148, nº 5, pp. 377-383, 2001.

[18] A. S. AlFuhaid, «Frequency Characteristics of Single-Phase Two-Winding Transformers Using Distributed-Parameter Modeling,» *IEEE Transactions on Power Delivery,* vol. 16, nº 4, pp. 637-642, 2001.

[19] B. Gustavsen, «Wide Band Modeling of Power Transformers,» *IEEE Transactions on Power Delivery,* vol. 19, nº 1, pp. 414-422, 2004.

[20] G. Liang, H. Sun, X. Zhang y X. Cui, «Modeling of Tranformer Windings Under Very Fast Transient Overvoltages,» *IEEE Transactions on Electromagnetic Compatibility,* vol. 48, nº 4, pp. 621-627, 2006.

[21] M. Popov, L. van der Sluis, R. P. P. Smeets, J. Lopez-Roldan y V. V. Terzija, «Modelling, Simulation and Measurement of Fast Transients in Transformer Windings with Consideration of Frequency-Dependent Losses,» *IET Electric Power Applications,* vol. 1, nº 1, pp. 29-35, 2007.

[22] P. Zhang, Y.-h. Wang, X.-p. Nie, W.-l. Yan y H.-j. Zhang, «A Modelling for Transformer Windings under Very Fast Transient Over-voltages,» de *International Conference on Electrical Machines and Systems*, Wuhan, 2008.

[23] X. Zhu, . H. Dong, G. Liang y C. Ji, «A New Hybrid Model of Transformer Winding under Very Fast Transient Overvoltages,» de *International Conference on Electrical Machines and Systems*, Wuhan, 2008.

[24] D. Juárez Aguilar, Análisis de la Distribución de la Tensión de Impulso en Devanados de Transformadores Tipo Columna, México D. F.: SEPI ESIME Zacatenco, 2002.

[25] J. A. De León Brito, Estudio del Efecto de las Tensiones Tipo PWM en los Sistemas de Aislamiento de los Transformadores de Media Tensión Alimentados por Variadores de Velocidad, México D. F.: SEPI ESIME Zacatenco, 2009.

[26] D. Soto Meza, Técnicas Computacionales para el Diseño Dieléctrico de

Transformadores de Potencia, México D. F.: SEPI ESIME Zacatenco, 2012.

[27] W. G. Nájera Gutiérrez, Análisis de Esfuerzos Dieléctricos en Transformadores Debidos a Excitaciones No Sinusoidales, México D. F.: SEPI ESIME Zacatenco, 2013.

[28] W. J. McNutt, T. J. Blalock y R. A. Hinton, «Response of Transfomer Windings to System Transient Voltages,» de *IEEE PES Summer Meeting & EHV/UHV Conference*, Vancouver, 1973.

[29] S. M. Hassan Hosseini, M. Vakilian y G. B. Gharehpetian, «Comparison of Transformer Detailed Models for Fast and Very Fast Transient Studies,» *IEEE Transactions on Power Delivery,* vol. 23, nº 2, pp. 733-741, 2008.

[30] S. Kulkarni y S. Khaparde, Transformer Engineering: Design and Practice, Marcel Dekker, Inc., 2004.

[31] P. I. Fergestad y H. T., «Transient Oscillations in Multiwinding Transformers,» de *IEEE PES Summer Meeting & EHV/UHV Conference*, Vancouver, 1973.

[32] K. Ragavan y L. Satish, «An Efficient Method to Compute Transfer Function of a Transformer From its Equivalent Circuit,» *IEEE Transactions on Power Delivery,* vol. 20, nº 2, pp. 780-788, 2005.

[33] F. H. Branin, «Transient Analysis of lossless transmission lines,» *Procedings of the IEEE,* vol. 55, pp. 2012-2013, 1967.

[34] H. W. Dommel, «Digital Computer Solution of Electromagnetic Transients in Single and Multiphase Networks,» *IEEE Transactions on Power Apparatus and Systems,* vol. PAS-88, nº 4, pp. 388-399, 1969.

[35] C. R. Paul, Analysis of Multiconductor Transmission Lines, John Wiley & Sons, Inc., 2008.

[36] J. H. Harlow, Ed., Electric Power Transformer Engineering, CRC Press LLC, 2004.

[37] Z. Azzouz, A. Foggia, L. Pierrat y G. Meunier, «3D Finite Element Computation of the High Frequency Parameters of Power Transformer Windings,» *IEEE Transactions on Magnetics,* vol. 29, nº 2, pp. 1407-1410, 1993.

[38] G. Liang, H. Sun, X. Zhang y X. Cui, «Modeling of Transformer Windings Under Very Fast Transient Overvoltages,» *IEEE Transactions on Electromagnetic Compatibility,* vol. 48, nº 4, pp. 621-627, 2006.

[39] C. Zhao, J. Ruan, Z. Du, S. Liu, Y. Yu y Y. Zhang, «Calculation of parameters in transformer winding based on the model of multi-conductor transmission line,» de *International Conference on Electrical Machines and Systems*, Wuhan, 2008.

[40] W. D. Stevenson, Análisis de Sistemas Eléctricos de Potencia, McGraw-Hill, 1979.

[41] M. N. O. Sadiku, Elementos de Electromagnetismo, Oxford Press, 2003.

[42] P. Gómez y F. de León, «Accurate an Efficient Computation of the Inductance Matrix of Transformer Windings for the Simulation of Very Fast Transients,» *IEEE Transactions on Power Delivery,* vol. 26, nº 3, pp. 1423-1431, 2011.

[43] K. A. Wirgau, «Inductance Calculation of an Aire-Core Disk Winding,» *IEEE Transactions on Power Apparatus and Systems,* vol. PAS-95, nº 1, pp. 394-400, 1976.

[44] E. B. Rosa y F. W. Grover, Formulas and Tables for the Calculation of Mutual and Self-Inductance [Revised], Department of Commerce Bureau of Standards, 1948.

[45] A. Gray, Absolute Measurements in Electricity and Magnetism, Macmillan and Co., Limited, 1921.

[46] P. Gómez, F. de León y I. A. Hernández, «Impulse-Response Analysis of Toroidal Core Distribution Transformers for Dielectric Design,» *IEEE Transactions on Power Delivery,* vol. 26, nº 2, pp. 1231-1238, 2011.

[47] The MathWorks, Inc., Simulink User's Guide, 2013.

[48] F. J. Quiñónez, P. Moreno, A. Chávez y J. L. Naredo, «Analysis of Fast Transient Overvoltages in Transformers Windings Using The Method of Characteristics,» de *North American Power Symposium (NAPS)*, Texas, 2001.

ANEXO

MODELO DE PARÁMETROS CONCENTRADOS

```
function trans_concen(block)

setup(block);


function setup(block)

% Register parameters
block.NumDialogPrms     = 13;

n = block.DialogPrm(3).Data;
NumOut = block.DialogPrm(10).Data;
ud = get_param(block.BlockHandle,'UserData');
ud.nodo = n;
set_param(block.BlockHandle,'UserData',ud)


% Register number of ports
block.NumInputPorts  = 1;
block.NumOutputPorts = NumOut;

% Setup port properties to be inherited or dynamic
block.SetPreCompInpPortInfoToDynamic;
block.SetPreCompOutPortInfoToDynamic;

%  Override input port properties

block.InputPort(1).Dimensions  = 1;
block.InputPort(1).DatatypeID  = 0;  % double
block.InputPort(1).Complexity  = 'Real';
block.InputPort(1).DirectFeedthrough = true;

block.SampleTimes = [0 0];

% Set up the continuous states.
block.NumContStates = ((2*n)-1);

block.RegBlockMethod('CheckParameters', @CheckPrms);
block.RegBlockMethod('InitializeConditions', @InitializeConditions);
block.RegBlockMethod('SetInputPortSamplingMode', @SetInpPortFrameData);
block.RegBlockMethod('Derivatives', @Derivatives);
block.RegBlockMethod('Outputs',@Outputs);
block.RegBlockMethod('Terminate', @Terminate); % Required

% end setup

function CheckPrms(block)
```

```
Node  = block.DialogPrm(1).Data;
Param = block.DialogPrm(2).Data;
n     = block.DialogPrm(3).Data;
CB    = block.DialogPrm(4).Data;
LB    = block.DialogPrm(5).Data;
Outurns = block.DialogPrm(11).Data;
Rtot = block.DialogPrm(12).Data;
File = block.DialogPrm(13).Data;

[mc,nc] = size(CB);
[ml,nl] = size(LB);

if isequal(Param,1)
    if isempty(n)
        error('Introduce the number of turns');
    end
    if mod(n,1)~=0
        error('Number of turns must be an integer type');
    end
    if n<1
        error('Number of turns must be positive or nonzero')
    end
    if isempty(Node)
        error('Introduce the input node');
    end
    if mod(Node,1)~=0
        error('Input node must be an integer type');
    end
    if Node<1
        error('Input node must be positive or nonzero')
    end
    if Node>n
        error('Input node does not exist');
    end
    if isempty(CB)
        error('Introduce the Capacitance Matrix Name');
    end
    if ~isa(CB,'numeric')
        error('Capacitance matrix must be numeric type');
    end
    if mc ~= nc
        error('Capacitance Matrix must be square');
    end
    if ~isequal(mc,n)
        error('Dimension of Capacitance Matrix must be equal to the
number of turns');
    end
    if isempty(LB)
        error('Introduce the Inductance Matrix Name');
    end
    if ~isa(LB,'numeric')
        error('Inductance matrix must be numeric type');
    end
    if ml ~= nl
        error('Inductance Matrix must be square');
    end
    if ~isequal(ml,n)
```

```
        error('Dimension of Inductance Matrix must be equal to the number
of turns');
    end
    if isempty(Rtot)
        error('Introduce Series Losses Value');
    end
    if ~isa(Rtot,'numeric')
        error('Series Losses Value must be numeric type');
    end
    if Rtot(1,1)<0
        error('Series Losses Value must be positive or zero');
    end
    if isempty(Outurns)
        error('Introduce Number(s) of Specific Turn(s)');
    end
    if ~isa(Outurns,'numeric')
        error('Number(s) of Specific Turn(s) must be numeric type');
    end
    if ~isvector(Outurns)
        error('Number(s) of Specific Turn(s) must be a vector');
    end
    [mout,nout] = size(Outurns);
    for i=1:mout
        for j=1:nout
            if Outurns(i,j)<1
                error('Number(s) of Specific Turn(s) must be positive or
nonzero');
                break %#ok<UNRCH>
            end
            if Outurns(i,j)>n
                error('Number(s) of Specific Turn(s) does(do) not
exist');
                break %#ok<UNRCH>
            end
        end
    end
    if isempty(File)
        error('Introduce Output File Name');
    end
end

if isequal(Param,2)
    if isempty(n)
        error('Introduce the number of turns');
    end
    if mod(n,1)~=0
        error('Number of turns must be an integer type');
    end
    if n<1
        error('Number of turns must be positive or nonzero')
    end
    if isempty(Node)
        error('Introduce the input node');
    end
    if mod(Node,1)~=0
        error('Input node must be an integer type');
    end
```

```
    if Node<1
        error('Input node must be positive or nonzero')
    end
    if Node>n
        error('Input node does not exist');
    end
    if isempty(CB)
        error('Introduce the Capacitance Matrix File Name');
    end
    if mc ~= nc
        error('Capacitance Matrix must be square');
    end
    if ~isequal(mc,n)
        error('Dimension of Capacitance Matrix must be equal to the
number of turns');
    end
    if isempty(LB)
        error('Introduce the Inductance Matrix File Name');
    end
    if ml ~= nl
        error('Inductance Matrix must be square');
    end
    if ~isequal(ml,n)
        error('Dimension of Inductance Matrix must be equal to the number
of turns');
    end
    if isempty(Rtot)
        error('Introduce Series Losses Value');
    end
    if ~isa(Rtot,'numeric')
        error('Series Losses Value must be numeric type');
    end
    if Rtot(1,1)<0
        error('Series Losses Value mut be positive or zero');
    end
    if isempty(Outurns)
        error('Introduce Number(s) of Specific Turn(s)');
    end
    if ~isa(Outurns,'numeric')
        error('Number(s) of Specific Turn(s) must be numeric type');
    end
    if ~isvector(Outurns)
        error('Number(s) of Specific Turn(s) must be a vector');
    end
    [mout,nout] = size(Outurns);
    for i=1:mout
        for j=1:nout
            if Outurns(i,j)<1
                error('Number(s) of Specific Turn(s) must be positive or
nonzero');
                break %#ok<UNRCH>
            end
            if Outurns(i,j)>n
                error('Number(s) of Specific Turn(s) does(do) not
exist');
                break %#ok<UNRCH>
            end
```

```
        end
    end
    if isempty(File)
        error('Introduce Output File Name');
    end
end

function InitializeConditions(block)

n = block.DialogPrm(3).Data;
block.ContStates.Data= zeros(((2*n)-1),1);

function SetInpPortFrameData(block,~,~)

% n = block.DialogPrm(3).Data;
NumOut = block.DialogPrm(10).Data;
block.InputPort(1).SamplingMode = 0;
for i=1:NumOut
    block.OutputPort(i).SamplingMode  = 0;
end

function Derivatives(block)

A = block.DialogPrm(6).Data;
B = block.DialogPrm(7).Data;
ud = get_param(block.BlockHandle,'UserData');
% tiempo2 = ud.tiempo2;
entrada = ud.entrada;
deriv = ud.deriv;

block.Derivatives.Data =
A*block.ContStates.Data+B*[entrada(end);deriv(end)];

%end Derivatives

function Outputs(block)

Node     = block.DialogPrm(1).Data;
n        = block.DialogPrm(3).Data;
F        = block.DialogPrm(8).Data;
D        = block.DialogPrm(9).Data;
Outurns  = block.DialogPrm(11).Data;

ud = get_param(block.BlockHandle,'UserData');
;

if  block.CurrentTime==0 && block.IsMajorTimeStep
    ud.tiempo = [ud.tiempo block.CurrentTime];
    ud.entrada = [ud.entrada block.InputPort(1).Data];
    ud.deriv = [ud.deriv 0.0];
    entrada = ud.entrada;
    deriv = ud.deriv;
    set_param(block.BlockHandle,'UserData',ud)
    OUT =   F*block.ContStates.Data+D*[entrada(end);deriv(end)];
```

```
elseif block.IsMajorTimeStep
    ud.tiempo = [ud.tiempo block.CurrentTime];
    tiempo = ud.tiempo;
    ud.entrada = [ud.entrada block.InputPort(1).Data];
    entrada = ud.entrada;
    ud.deriv = [ud.deriv (entrada(end)-entrada(end-1))/(tiempo(end)-
tiempo(end-1))];
    deriv = ud.deriv;
    set_param(block.BlockHandle,'UserData',ud)
    OUT =   F*block.ContStates.Data+D*[entrada(end);deriv(end)];
end

if block.IsMajorTimeStep

 k = 1;
 for i=Outurns
     if i==Node
         block.OutputPort(k).Data = block.InputPort(1).Data;
     elseif i<Node
         block.OutputPort(k).Data = OUT(i,:);
     else
         block.OutputPort(k).Data = OUT(i-1,:);
     end
     k = k+1;
 end

 if strcmp(ud.opcion,'on')||strcmp(ud.opcion2,'on')
     for i=1:n
         if i==Node
             ud.salida = [ud.salida block.InputPort(1).Data];
         elseif i<Node
             ud.salida = [ud.salida OUT(i,:)];
         else
             ud.salida = [ud.salida OUT(i-1,:)];
         end
     end
     set_param(block.BlockHandle,'UserData',ud)
 end
end

%end Outputs

function Terminate(~)

%end Terminate
```

MODELO DE PARÁMETROS DISTRIBUIDOS

```
function trans_distrib(block)

setup(block);

function setup(block)

% Register parameters
block.NumDialogPrms     = 18;

n = block.DialogPrm(3).Data;
NumOut = block.DialogPrm(14).Data;
ud = get_param(block.BlockHandle,'UserData');
ud.nodo = n;
set_param(block.BlockHandle,'UserData',ud)


% Register number of ports
block.NumInputPorts  = 1;
block.NumOutputPorts = NumOut;

% Setup port properties to be inherited or dynamic
block.SetPreCompInpPortInfoToDynamic;
block.SetPreCompOutPortInfoToDynamic;

block.SampleTimes = [0 0];

% Set up the continuous states.
block.NumContStates = ((2*n)-1);

block.RegBlockMethod('CheckParameters', @CheckPrms);
block.RegBlockMethod('InitializeConditions', @InitializeConditions);
block.RegBlockMethod('SetInputPortSamplingMode', @SetInpPortFrameData);
% block.RegBlockMethod('Derivatives', @Derivatives);
block.RegBlockMethod('Outputs',@Outputs);
block.RegBlockMethod('Terminate', @Terminate); % Required

% end setup

function CheckPrms(block)

Node  = block.DialogPrm(1).Data;
Param = block.DialogPrm(2).Data;
n     = block.DialogPrm(3).Data;
CB    = block.DialogPrm(4).Data;
LB    = block.DialogPrm(5).Data;
Outurns = block.DialogPrm(15).Data;
Rtot = block.DialogPrm(16).Data;
File = block.DialogPrm(17).Data;
long = block.DialogPrm(18).Data;

[mc,nc] = size(CB);
[ml,nl] = size(LB);
```

```
if isequal(Param,1)
    if isempty(n)
        error('Introduce the number of turns');
    end
    if mod(n,1)~=0
        error('Number of turns must be an integer');
    end
    if n<1
        error('Number of turns must be positive')
    end
    if isempty(long)
        error('Introduce Turn Length Value');
    end
    if ~isa(long,'numeric')
        error('Turn Length Value must be numeric type');
    end
    if long<0
        error('Turn Length Value must be positive or zero');
    end
    if isempty(Node)
        error('Introduce the input node');
    end
    if mod(Node,1)~=0
        error('Input node must be an integer');
    end
    if Node<1
        error('Input node must be positive')
    end
    if Node>n
        error('Input node does not exist');
    end
    if isempty(CB)
        error('Introduce the Capacitance Matrix Name');
    end
    if ~isa(CB,'numeric')
        error('Capacitance matrix must be numeric type');
    end
    if mc ~= nc
        error('Capacitance Matrix must be square');
    end
    if ~isequal(mc,n)
        error('Dimension of Capacitance Matrix must be equal to the
number of turns');
    end
    if isempty(LB)
        error('Introduce the Inductance Matrix Name');
    end
    if ~isa(LB,'numeric')
        error('Inductance matrix must be numeric type');
    end
    if ml ~= nl
        error('Inductance Matrix must be square');
    end
    if ~isequal(ml,n)
        error('Dimension of Inductance Matrix must be equal to the number
of turns');
```

```
    end
    if isempty(Rtot)
        error('Introduce Series Losses Value');
    end
    if ~isa(Rtot,'numeric')
        error('Series Losses Value must be numeric type');
    end
    if Rtot(1,1)<0
        error('Series Losses Value must be positive or zero');
    end
    if isempty(Outurns)
        error('Introduce Number(s) of Specific Turn(s)');
    end
    if ~isa(Outurns,'numeric')
        error('Number(s) of Specific Turn(s) must be numeric type');
    end
    if ~isvector(Outurns)
        error('Number(s) of Specific Turn(s) must be a vector');
    end
    [mout,nout] = size(Outurns);
    for i=1:mout
        for j=1:nout
            if Outurns(i,j)<1
                error('Number(s) of Specific Turn(s) must be positive or
nonzero');
                break %#ok<UNRCH>
            end
            if Outurns(i,j)>n
                error('Number(s) of Specific Turn(s) does(do) not
exist');
                break %#ok<UNRCH>
            end
        end
    end
    if isempty(File)
        error('Introduce Output File Name');
    end
end

if isequal(Param,2)
    if isempty(n)
        error('Introduce the number of turns');
    end
    if mod(n,1)~=0
        error('Number of turns must be an integer');
    end
    if n<1
        error('Number of turns must be positive')
    end
    if isempty(long)
        error('Introduce Turn Length Value');
    end
    if ~isa(long,'numeric')
        error('Turn Length Value must be numeric type');
    end
    if long<0
        error('Turn Length Value must be positive or zero');
```

```
    end
    if isempty(Node)
        error('Introduce the input node');
    end
    if mod(Node,1)~=0
        error('Input node must be an integer');
    end
    if Node<1
        error('Input node must be positive')
    end
    if Node>n
        error('Input node does not exist');
    end
    if isempty(CB)
        error('Introduce the Capacitance Matrix File Name');
    end
    if mc ~= nc
        error('Capacitance Matrix must be square');
    end
    if ~isequal(mc,n)
        error('Dimension of Capacitance Matrix must be equal to the
number of turns');
    end
    if isempty(LB)
        error('Introduce the Inductance Matrix File Name');
    end
    if ml ~= nl
        error('Inductance Matrix must be square');
    end
    if ~isequal(ml,n)
        error('Dimension of Inductance Matrix must be equal to the number
of turns');
    end
    if isempty(Rtot)
        error('Introduce Series Losses Value');
    end
    if ~isa(Rtot,'numeric')
        error('Series Losses Value must be numeric type');
    end
    if Rtot(1,1)<0
        error('Series Losses Value mut be positive or zero');
    end
    if isempty(Outurns)
        error('Introduce Number(s) of Specific Turn(s)');
    end
    if ~isa(Outurns,'numeric')
        error('Number(s) of Specific Turn(s) must be numeric type');
    end
    if ~isvector(Outurns)
        error('Number(s) of Specific Turn(s) must be a vector');
    end
    [mout,nout] = size(Outurns);
    for i=1:mout
        for j=1:nout
            if Outurns(i,j)<1
                error('Number(s) of Specific Turn(s) must be positive or
nonzero');
```

```
                break %#ok<UNRCH>
            end
            if Outurns(i,j)>n
                error('Number(s) of Specific Turn(s) does(do) not
exist');
                break %#ok<UNRCH>
            end
        end
    end
    if isempty(File)
        error('Introduce Output File Name');
    end
end

function InitializeConditions(block)

n = block.DialogPrm(3).Data;
block.ContStates.Data= zeros(((2*n)-1),1);

%
function SetInpPortFrameData(block,~,~)

% n = block.DialogPrm(3).Data;
NumOut = block.DialogPrm(14).Data;
block.InputPort(1).SamplingMode = 0;
for i=1:NumOut
    block.OutputPort(i).SamplingMode  = 0;
end

function Outputs(block)

Node     = block.DialogPrm(1).Data;
n        = block.DialogPrm(3).Data;
s        = block.DialogPrm(6).Data;
Tl       = block.DialogPrm(7).Data;
k        = block.DialogPrm(8).Data;
Z0m      = block.DialogPrm(9).Data;
Z0m2     = block.DialogPrm(10).Data;
Y0m      = block.DialogPrm(11).Data;
Rf       = block.DialogPrm(12).Data;
Rtot     = block.DialogPrm(16).Data;
ZBUS     = block.DialogPrm(13).Data;
Outurns  = block.DialogPrm(15).Data;


I = eye(n);
ud = get_param(block.BlockHandle,'UserData');
j = ud.itera;

if block.CurrentTime==0 && block.IsMajorTimeStep
    ud = get_param(block.BlockHandle,'UserData');
    ud.itera = [ud.itera 0.0];
    j = ud.itera;
    j = j+s+1;
    ud.entrada = [ud.entrada zeros(2*n,k+s)];
```

```
    E = ud.entrada;
    E(Node,j-s) = block.InputPort(1).Data;
    Em = [(I/Tl)*E(1:n,:);(I/Tl)*E(n+1:2*n,:)];
    ud.tension0 = [ud.tension0 zeros(n,k+s)];
    v0m = ud.tension0;
    ud.tensionL = [ud.tensionL zeros(n,k+s)];
    vLm = ud.tensionL;
    ud.corriente0 = [ud.corriente0 zeros(n,k+s)];
    I0m = ud.corriente0;
    ud.corrienteL = [ud.corrienteL zeros(n,k+s)];
    ILm = ud.corrienteL;

    H0m = -(Z0m/(Z0m2^2))*(vLm(:,j-s)+(Z0m-Rtot)*(ILm(:,j-s)))-
(Rtot/(Z0m2^2))*(v0m(:,j-s)+(Z0m-Rtot)*(I0m(:,j-s)));
    HLm = -(Z0m/(Z0m2^2))*(v0m(:,j-s)+(Z0m-Rtot)*(I0m(:,j-s)))-
(Rtot/(Z0m2^2))*(vLm(:,j-s)+(Z0m-Rtot)*(ILm(:,j-s)));

    Hm = [-H0m;-HLm];

    Vm = ZBUS*(Hm-(-1/Rf)*(Em(:,j-s)));

    v0m(:,j) = Vm(1:n,1);
    vLm(:,j) = Vm(n+1:2*n,1);

    I0m(:,j) = (Y0m)*(v0m(:,j))+H0m;
    ILm(:,j) = (Y0m)*(vLm(:,j))+HLm;

    ud.tension0 = (v0m);
    ud.tensionL = (vLm);
    ud.corriente0 = (I0m);
    ud.corrienteL = (ILm);

    set_param(block.BlockHandle,'UserData',ud)

    OUT = (Tl)*vLm;

    q =1;
    for i=Outurns  %1:n
            block.OutputPort(q).Data = OUT(i,j(end));
            q = q+1;
    end

elseif block.IsMajorTimeStep
    ud = get_param(block.BlockHandle,'UserData');
    ud.itera = [ud.itera j(end)+1];
    j = ud.itera;
    j = j+s+1;
    E = ud.entrada;
    E(Node,j(end)-s) = block.InputPort(1).Data;
    Em = [(I/Tl)*E(1:n,:);(I/Tl)*E(n+1:2*n,:)];
    v0m = ud.tension0;
    vLm = ud.tensionL;
    I0m = ud.corriente0;
    ILm = ud.corrienteL;
```

```
    H0m = -(Z0m/(Z0m2^2))*(vLm(:,j(end)-s)+(Z0m-Rtot)*(ILm(:,j(end)-s)))-
(Rtot/(Z0m2^2))*(v0m(:,j(end)-s)+(Z0m-Rtot)*(I0m(:,j(end)-s)));
    HLm = -(Z0m/(Z0m2^2))*(v0m(:,j(end)-s)+(Z0m-Rtot)*(I0m(:,j(end)-s)))-
(Rtot/(Z0m2^2))*(vLm(:,j(end)-s)+(Z0m-Rtot)*(ILm(:,j(end)-s)));

    Hm = [-H0m;-HLm];

    Vm = ZBUS*(Hm-(-1/Rf)*(Em(:,j(end)-s)));

    v0m(:,j(end)) = Vm(1:n,1);
    vLm(:,j(end)) = Vm(n+1:2*n,1);

    I0m(:,j(end)) = (Y0m)*(v0m(:,j(end)))+H0m;
    ILm(:,j(end)) = (Y0m)*(vLm(:,j(end)))+HLm;

    ud.tension0 = (v0m);
    ud.tensionL = (vLm);
    ud.corriente0 = (I0m);
    ud.corrienteL = (ILm);

    set_param(block.BlockHandle,'UserData',ud)

    OUT = (Tl)*vLm;

    q=1;
    for i=Outurns  %1:n
            block.OutputPort(q).Data = OUT(i,j(end));
            q=q+1;
    end

end

if block.IsMajorTimeStep
    if strcmp(ud.opcion,'on')||strcmp(ud.opcion2,'on')
         ud = get_param(block.BlockHandle,'UserData');
         ud.tiempo = [ud.tiempo block.CurrentTime];
         for i=1:n
             ud.salida = [ud.salida OUT(i,j(end))];
         end
    end
    set_param(block.BlockHandle,'UserData',ud)
 end

%end Outputs

function Terminate(~)

%end Terminate
```