Titre: Title:	Nouvelle approche de modélisation de la branche de magnétisation pour la simulation des transitoires électromagnétiques
Auteur: Author:	Mathieu Lambert
Date:	2009
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	Lambert, M. (2009). Nouvelle approche de modélisation de la branche de magnétisation pour la simulation des transitoires électromagnétiques [Mémoire de maîtrise, École Polytechnique de Montréal]. PolyPublie. https://publications.polymtl.ca/8506/

UNIVERSITÉ DE MONTRÉAL

NOUVELLE APPROCHE DE MODÉLISATION DE LA BRANCHE DE MAGNÉTISATION POUR LA SIMULATION DES TRANSITOIRES ÉLECTROMAGNÉTIQUES

MATHIEU LAMBERT
DÉPARTEMENT DE GÉNIE ÉLECTRIQUE
ÉCOLE POLYTECHNIQUE DE MONTRÉAL

MÉMOIRE PRÉSENTÉ EN VUE DE L'OBTENTION
DU DIPLÔME DE MAÎTRISE ÈS SCIENCES APPLIQUÉES
(GÉNIE ÉLECTRIQUE)
AVRIL 2009

Your file Votre référence
ISBN: 978-0-494-69166-3
Our file Notre référence
ISBN: 978-0-494-69166-3

UNIVERSITÉ DE MONTRÉAL

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

Ce mémoire intitulé:

NOUVELLE APPROCHE DE MODÉLISATION DE LA BRANCHE DE MAGNÉTISATION POUR LA SIMULATION DES TRANSITOIRES ÉLECTROMAGNÉTIQUES

présenté par: LAMBERT Mathieu

en vue de l'obtention du diplôme de: Maîtrise ès sciences appliquées

a été dûment accepté par le jury d'examen constitué de:

M. SIROIS Frédéric, Ph.D., président

M. MAHSEREDJIAN Jean, Ph.D., membre et directeur de recherche

M. DESSAINT Louis-A., Ph.D., membre et codirecteur de recherche

M. SAAD Omar, M.Sc.A., membre

À Amélie et Vincent

REMERCIEMENTS

La réalisation de cet ouvrage n'aurait pu être possible sans la contribution de plusieurs partenaires. Dans un premier temps, je tiens à remercier mon directeur de recherche, Dr. Jean Mahseredjian, pour son éternelle patience et ses explications sur les rouages du calcul numérique appliqué à la simulation des réseaux électriques. Malgré les obstacles rencontrés en cours de projet, il a toujours su que je parviendrais à mes fins. Deuxièmement, j'aimerais remercier la *Chaire TransÉnergie sur la simulation et la commande des réseaux électriques* pour son soutien accordé au projet, ainsi qu'à son titulaire, Dr. Louis-A. Dessaint. D'autre part, la validation du modèle A(x) n'aurait pu être possible sans les mesures fournies par l'Institut de recherche d'Hydro-Québec. À cet effet, je voudrais remercier André Gaudreau, Silvano Casoria et Omar Saad pour leurs contributions respectives au projet. De surcroît, je tiens à remercier Dr. Afshin Rezaei pour sa collaboration ainsi que pour ses éclaircissements apportés sur un sujet aussi complexe que passionnant. Finalement, j'aimerais remercier mes parents pour leur soutien et leurs encouragements.

RÉSUMÉ

Cet ouvrage présente un nouveau modèle de la branche de magnétisation pour simuler les phénomènes d'hystérésis et de saturation dans un transformateur. Ce nouveau modèle, le modèle A(x), a été implémenté dans un logiciel de simulation des transitoires électromagnétiques, le logiciel EMTP-RV. Le modèle A(x) permet d'obtenir des résultats plus précis et d'avoir plusieurs degrés de liberté pour mieux reproduire une large gamme de courbes expérimentales, tout en restant efficace en terme de temps de calcul. Ce modèle peut être classé comme un modèle scalaire statique d'hystérésis ferromagnétique avec une mémoire non locale.

Les objectifs de ce projet consistaient à implémenter un nouveau modèle facilement paramétrisable et de le comparer, ainsi que de le valider, par rapport aux modèles existants. Pour ce faire, plusieurs circuits de test ont été élaborés dans le but de vérifier la validité du modèle soumis à différents scénarios. Les résultats de ces tests sont concluants ; le modèle A(x) a été capable de mieux reproduire les pertes en régime permanent que les autres modèles antisymétriques et un régime ferrorésonant a été correctement amorcé lorsqu'un terne couplé a été ouvert sur une ligne biterne à haute tension connectée à un transformateur sans charge. Aussi, le modèle a montré qu'il était rapide en temps de calcul et qu'il requiert seulement les mesures de la boucle majeure, qui sont plus facilement disponibles que les données des boucles mineures, ainsi que les données sur les courbes de renversement du premier ordre.

De plus, la théorie entourant le modèle A(x) a été expliquée de fond en comble et les détails concernant son implémentation dans un contexte numérique ont été explicités. D'autre part, certains cas spéciaux de divergence, par exemple les oscillations numériques, ont été examinés et des solutions ont été proposées.

ABSTRACT

This work presents a new magnetizing branch model to simulate the hysteresis and saturation phenomena in a transformer. This new model, named the A(x) model, has been implemented for an electromagnetic transients simulation program, named EMTP-RV. The capabilities of the A(x) model include better precision and extended degrees of freedom to fit a large variety of curves, while maintaining computational efficiency. This model can be categorized as a rate-independent scalar model of ferromagnetic hysteresis with non-local memory.

The objectives of this project are to implement a new model that would be easily definable and to compare and validate it with other existing models. To do so, a few benchmarks are created, in order to assess the validity of the model under different scenarios. The results obtained from these tests are conclusive; the A(x) model is able to better reproduce the losses under steady-state conditions than other antisymmetric models and it is able to initiate ferroresonnance correctly when an open coupled circuit from a high-voltage double-circuit line is connected to an unloaded transformer. It is also computationally efficient and requires only the major loop data, which is more readily available data than the minor loops or first-order reversal curves.

Also, the theory surrounding the A(x) model is thoroughly explained and the details regarding its implementation in a numerical environment are refined. Furthermore, special cases of divergence, numerical oscillations, for instance, are examined and solutions are proposed.

TABLE DES MATIÈRES

LISTE DES TABLEAUX

LISTE DES FIGURES

LISTE DES SIGLES ET ABRÉVIATIONS

ϕ :	Flux instantanné
i :	Courant instantanné
v :	Tension instantannée
$\Phi(i)$:	Trajectoires de la boucle majeure dans le plan $\phi - i$
Δt :	Pas d'intégration
τ :	Point arbitraire dans le temps
C :	Paramètre de translation pour les trajectoires des boucles mineures
f :	Fonction à solutionner pour la méthode itérative de Newton

INDICES

ss :	Régime permanent
$coer$:	Coercitif
$+$:	Trajectoire ascendante
$-$:	Trajectoire descendante
N :	Norton
km :	Terminaux de la branche de magnétisation
$hist$:	Historique
n :	Ordre de renversements
q :	Point quiescent

EXPOSANTS

(n) :	Numéro de l'itération de Newton pour trouver le courant de la branche à partir du flux
(k) :	Numéro de l'itération de Newton de l'élément non-linéaire

LISTE DES ANNEXES

INTRODUCTION

Le phénomène d'hystérésis est connu depuis fort longtemps. La racine éthymologique du terme est grecque, *hysteros*, et signifie *plus tard* ou *être en retard*, du verbe *hysterein*. C'est Ewing qui introduisit le terme en 1881, après avoir constaté un retard entre le changement de polarisation d'un fil de fer soumis à un champ magnétique solénoïdal et une variation de la torsion du fil [1]. D'autre part, il remarqua qu'il pouvait exister deux valeurs différentes de qualité thermoélectrique pour une même valeur de stress mécanique, dépendamment de la manière dont l'on approche la valeur de stress donnée [2]. Dès lors, il avait découvert la propriété fondamentale de tout hystérésis : la ramification. On connait surtout l'hystérésis magnétique, mais en réalité ce phénomène est présent dans plusieurs autres domaines que ce soit dans l'élasticité mécanique qui ne suit pas exactement la loi de Hooke, en passant par l'hystérésis d'absorbtion ou encore dans le comportement des neurones soumis à un stimulus. D'autre part, on croit souvent à tort que l'hystérésis résulte de la formation de boucles dans la réponse, par rapport à l'excitation. Cependant, les boucles représentent une condition particulière de ramification, lorsque l'excitation est en régime permanent et qu'elle subit les mêmes variations cycliques. De surcroît, on associe fréquemment le phénomène d'hystérésis magnétique à la saturation magnétique, mais ce sont en réalité deux phénomènes distincts ; la saturation résulte en l'incapacité de la réponse d'augmenter, malgré l'acroissement de l'excitation. La caractéristique de saturation anhystérétique est une fonction monotone et réversible, où aucune ramification n'est possible. Même s'il est possible de représenter les deux phénomènes séparément, le présent ouvrage présente un nouveau modèle, le modèle A(x), pour représenter les deux phénomènes simultanément à l'intérieur d'une même branche, appelée *branche de magnétisation*. Celle-ci remplace la branche de magnétisation linéaire composant le modèle classique de transformateur pour l'étude de transitoires dans la simulation des réseaux électriques et il a été implémenté en Fortran 95, sous forme de DLL, dans le logiciel EMTP-RV [3]. Le modèle A(x) peut être classé comme un modèle d'ordre phénoménologique scalaire et indépendant du taux de

variation de l'excitation. L'objectif premier de ce projet consiste à présenter un modèle générique facilement paramétrisable qui permet de tenir compte correctement de l'ensemble de l'effet hystérétique et de saturation et qui permet de représenter correctement des régimes ferrorésonants. En second lieu, il s'agit de vérifier et de valider les modèles existants ainsi que de les comparer au nouveau modèle à l'aide de résultats d'essais disponibles.

Ce travail débute en exposant les différentes contraintes inhérentes reliées à la modélisation de l'hystérésis et de la saturation. Par la suite, une brève revue littéraire des principaux modèles est proposée. Ensuite, la théorie entourant le nouveau modèle est explicitée et les détails de son implémentation dans un contexte de calcul numérique sont exposés. Finalement, plusieurs scénarios de simulation sont testés et les résultats démontrent l'intérêt, ainsi que la plage de validité du modèle proposé.

CHAPITRE 1

PROBLÉMATIQUE

Malgré les percées récentes dans les fondements du ferromagnétisme grâce à l'imagerie des domaines magnétiques avec les rayons X polarisés [4], le mystère de la dynamique magnétique ne reste que partiellement élucidé et aucun modèle ne peut prétendre reproduire le phénomène de façon exacte. Par contre, on sait que l'hystérésis magnétique est un phénomène thermodynamique métastable [5], où le système tente de se diriger vers l'état où l'énergie libre est moindre. De plus, on a remarqué que le mouvement des parois des domaines magnétiques, pour passer d'un minimum local à un autre, se fait sous forme de sauts discrets reliant des états magnétiques distincts, phénomène appelé *effet Barkhausen* [6][7]. De ces connaissances, a emergé la première catégorie de modèles : les modèles fondamentaux. Ceux-ci se basent sur la physique du phénomène et sont, en général, définis par un système d'équations différentielles non-linéaires qui tentent, autant que faire se peut, de décrire le comportement micromagnétique du matériau et ils sont lourds en temps de calcul [8]. Malgré la complexité des modèles, certaines approximations et hypothèses sont nécessaires, car toutes les variables ne peuvent être connues et, plus souvent qu'autrement, ces approximations viennent obscurcir la beauté inhérente de la théorie dans les cas pratiques. À l'opposé, les modèles de type phénoménologique tentent de reproduire les résultats expérimentaux au moyen de fonctions mathématiques et penchent davantage du côté pratique que du côté théorique. Ceux-ci ne tentent pas d'expliquer le phénomène au niveau microscopique, mais caractérisent le comportement macroscopique de l'élément : c'est le principe de la boîte noire. Dans un contexte de simulation de réseaux électriques, la deuxième approche est davantage intéressante pour modéliser la branche de magnétisation des transformateurs, car c'est le comportement global de l'appareil avec le réseau que l'on cherche à modéliser et non son comportement interne. D'autre part, les modèles

fondamentaux sont lourds en temps de calcul de par la présence d'équations différentielles non-linéaires à résoudre, comparé aux modèles phénoménologiques. Ainsi, la deuxième catégorie de modèles est plus intéressante dans le contexte actuel, car le réseau étudié peut contenir plusieurs transformateurs et le temps de calcul doit rester dans les limites du raisonnable.

1.1 Trajectoires du phénomène d'hystérésis avec saturation

Dans un premier temps, il importe de définir quelques termes, qui seront utilisés au cours de cet ouvrage. Une caractéristique typique de la branche de magnétisation est présentée à la figure 1.1, dans le plan flux-courant. La courbe constituant l'enveloppe externe de ces trajectoires est appelée boucle majeure ; c'est la boucle asymptotique, obtenue en théorie en effectuant un cycle avec l'excitation de $-\infty$ à $+\infty$. Toutes autres trajectoires refermées sur elles-mêmes sont appelées boucles mineures et sont contenues à l'intérieur de la boucle majeure, pour une fréquence donnée de l'excitation. Cependant, en pratique, on considère par abus de langage qu'une boucle mineure suffisamment grande, pour un flux de 1.5 pu, par exemple, représente la boucle majeure, car celle-ci n'est physiquement pas mesurable. De toute façon, cette boucle mineure se rapproche suffisamment de l'asymptote pour être considérée comme telle. De plus, puisque le phénomène d'hystérésis est un phénomène thermodynamique de pertes, les trajectoires sont antihoraires : la trajectoire ascendante est toujours celle à droite du point de renversement et la trajectoire descendante est toujours celle de gauche. Les points de renversements représentent les extrema de l'excitation, donc les points où la dérivée change de signe.

À la figure 1.2, on peut apercevoir un phénomène intéressant qui apparaît sur les caractéristiques expérimentales des transformateurs. La courbe s'évase en approchant de la saturation et donne lieu à une silhouette dite en col d'oie. La courbe pointillée représente la rotation de 180 degrés de la partie inférieure de la trajectoire ascendante par rapport à la coercivité. De facto, on constate que la partie supérieure n'est pas une rotation de 180

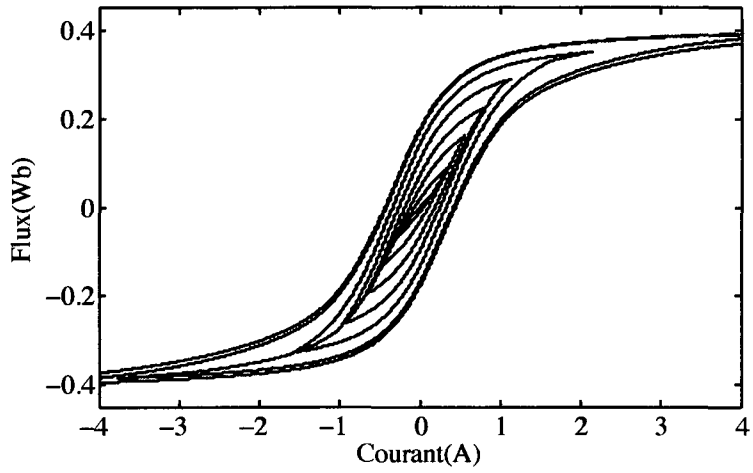


Figure 1.1 Trajectoires hystérétiques typiques

degrés de la partie inférieure et donc que la trajectoire n'est pas antisymétrique. Cependant, la courbe descendante représente l'antisymétrie de la courbe ascendante par rapport à l'origine. Ces constatations sont primordiales pour la suite, lors du choix de la fonction du modèle phénoménologique pour reproduire les mesures expérimentales. En fait, l'hypothèse d'antisymétrie de la courbe expérimentale est adoptée par la grande majorité des modèles phénoménologiques et c'est en général la source d'incapacité de ces modèles de reproduire précisément les boucles majeures expérimentales. Clairement, on voit sur la figure 1.2 que l'utilisation de fonctions antisymétriques vient modifier l'allure de la courbe et par le fait même, la surface de la boucle. L'aire sous la courbe d'hystérésis représente les pertes et celles-ci agissent sur l'amortissement des transitoires, au même titre qu'une résistance. Par le fait même, on voit toute l'importance de modéliser précisément les cycles d'hystérésis dans un contexte de simulation de transitoires électromagnétiques dans les réseaux électriques.

D'après la figure 1.1, on constate qu'il peut exister une infinité de trajectoires contenues à l'intérieur de la boucle majeure, dépendamment des états magnétiques précédents, donc de l'historique de l'excitation. L'infinité de solutions complexifie la modélisation du phénomène et certaines hypothèses doivent être émises. Des axiomes, propositions qui sont

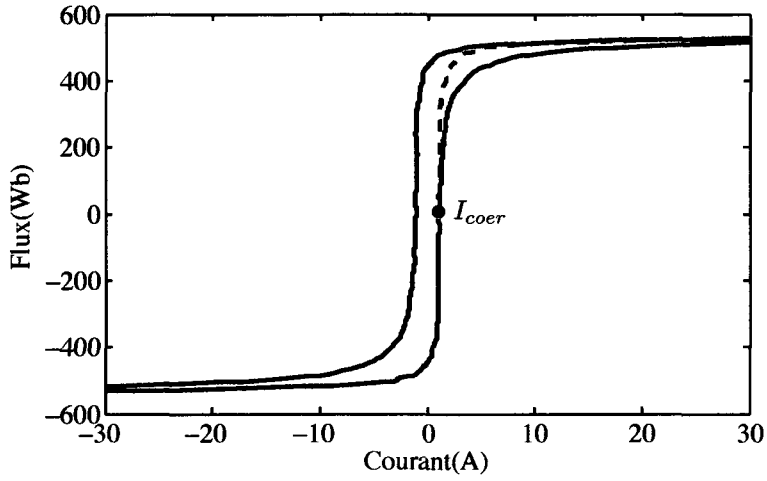


Figure 1.2 Boucle majeure expérimentale avec la trajectoire antisymétrique

généralement acceptées et qui sont en accord avec les observations en pratique, servent de pierre angulaire pour la modélisation des trajectoires. Les principaux axiomes utilisés peuvent se résumer ainsi [8][9][10] :

Axiome 1 *Toute trajectoire doit rester à l'intérieur de la boucle majeure.*

Axiome 2 *Si l'excitation dépasse en valeur absolue la valeur d'un extremum passé, l'effet de cet extremum intermédiaire sur les états magnétiques futurs est effacé. On appelle généralement cette propriété la propriété d'effacement.*

Axiome 3 *Les boucles résultantes de variations cycliques entre les deux mêmes valeurs d'excitation sont congruentes. Il découle de cette propriété que les trajectoires se referment pour former des boucles et que la trajectoire active passe nécessairement par l'avant-dernier extremum.*

Finalement, puisque le nombre de trajectoires possibles est infini, il faudrait en réalité une infinité de mesures pour modéliser toutes les situations possibles. Cependant, il n'est pas

possible en pratique d'effectuer une infinité d'observations et par ailleurs, les données expérimentales disponibles sur le transformateur à modéliser sont généralement peu nombreuses. En général, seule la boucle majeure est disponible. Ainsi, certaines hypothèses seront nécessaires, quant à la forme que prendront les trajectoires à l'intérieur de la boucle majeure.

CHAPITRE 2

REVUE LITTÉRAIRE

Le présent chapitre offre une brève revue des modèles aujourd'hui considérés comme classiques. La liste n'est pas exhaustive et n'a pas l'ambition d'être complète. Le lecteur intéressé à approfondir davantage les différents modèles mathématiques d'hystérésis développés au fil des âges peut se référer à [11][12].

2.1 Modèle de Preisach

Le modèle de Preisach est certainement l'un des modèles d'hystérésis le plus ancien et le plus étudié. Élaboré par F. Preisach en 1935 [13], il était au départ considéré comme un modèle fondamental et se basait sur l'hypothèse que le phénomène d'hystérésis magnétique était dû à la contribution globale de plusieurs dipôles ou domaines magnétiques élémentaires. Ces dipôles peuvent être vus comme des opérateurs à bascule, dont la caractéristique est présentée à la figure 2.1. La réponse de ces opérateurs, notée β, peut seulement prendre deux valeurs : $\beta = +B_{sat}$ ou $\beta = -B_{sat}$. L'état dans lequel se trouve le dipôle dépend bien sûr de l'excitation. Pour passer de l'état négatif à l'état positif, le champ magnétique doit dépasser la valeur de bascule H_u et pour passer de l'état positif à l'état négatif, il doit diminuer en deçà de la valeur de bascule H_d, où évidemment on a $H_u \geq H_d$. La transition d'un état à un autre se fait instantannément et aucun état intermédiaire n'est possible. D'autre part, les valeurs de H_u et de H_d varient d'un dipôle à un autre, pour tenir compte de l'intéraction entre les domaines magnétiques. Le modèle classique de Preisach peut-être défini comme :

$$B(t) = \iint_{H_u \geq H_d} \omega(H_u, H_d)\beta(H(t))\mathrm{d}H_u\mathrm{d}H_d \tag{2.1}$$

où chaque opérateur est multiplié par une fonction de poids, $\omega(H_u, H_d)$, et où l'intégrale

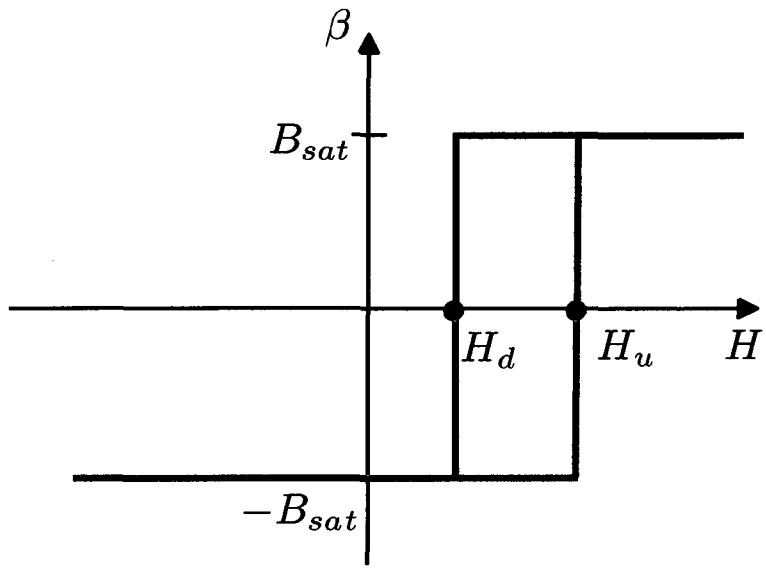


Figure 2.1 Caractéristique du dipôle élémentaire de Preisach

de la contribution de chaque domaine donne la réponse globale du système $B(t)$. Pour simplifier le fonctionnement du modèle classique de Preisach, on peut observer son comportement géométriquement dans le plan $H_u - H_d$, appelé le plan de Preisach. Puisqu'on a la condition $H_u \geq H_d$, le plan est en fait un triangle séparé par la droite $H_u = H_d$, tel qu'illustré à la figure 2.2. Puisque chaque opérateur ne peut prendre que deux valeurs, la réponse correspond donc à la somme de l'intégrale de la contribution des opérateurs dans l'état positif à celle des opérateurs dans l'état négatif. Dans le plan de Preisach, cela se traduit en deux surfaces $S^+(t)$ et $S^-(t)$, séparées par une courbe $L(t)$ en forme d'escalier qui varie dans le temps, tel qu'illustré à la figure 2.2. À l'aide du plan de Preisach, on peut réécrire l'équation (2.1) comme :

$$B(t) = \iint_{S^+(t)} \omega(H_u, H_d)\beta(H(t))\mathrm{d}H_u\mathrm{d}H_d + \iint_{S^-(t)} \omega(H_u, H_d)\beta(H(t))\mathrm{d}H_u\mathrm{d}H_d \quad (2.2)$$

D'autre part, puisque $\beta = +B_{sat}$ dans S^+ et que $\beta = -B_{sat}$ dans S^-, on obtient :

$$B(t) = B_{sat} \iint_{S^+(t)} \omega(H_u, H_d)\mathrm{d}H_u\mathrm{d}H_d - B_{sat} \iint_{S^-(t)} \omega(H_u, H_d)\mathrm{d}H_u\mathrm{d}H_d \quad (2.3)$$

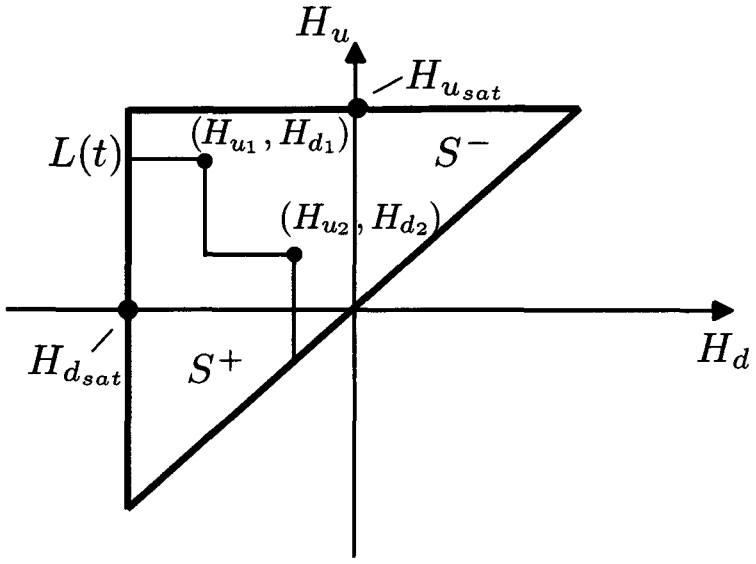


Figure 2.2 Plan de Preisach

Évidemment, les surfaces S^+ et S^- sont finies et les cas limites correspondent à la saturation $B = +B_{sat}$ ou $B = -B_{sat}$ obtenus à partir des valeurs de champ $H_{u_{sat}}$ ou $H_{d_{sat}}$, respectivement. La variation de l'excitation $H(t)$ aura pour effet de modifier l'interface $L(t)$ dans le plan de Preisach où les extrema représentent les sommets de l'escalier. Le principe est illustré à la figure 2.2 pour l'excitation de la figure 2.3. Au départ, le transformateur est saturé négativement, alors la surface correspond à S^-. Ensuite, on augmente l'excitation jusqu'à la valeur H_{u_1} ; cela aura pour effet de créer un nouveau segment de droite à l'interface $L(t)$ qui balaye de bas en haut le plan de Preisach jusqu'à $H_u = H_{u_1}$. À ce moment, l'excitation change de direction et diminue jusqu'au minimum H_{d_1}. En diminuant, un nouveau segment vertical s'ajoute à l'interface $L(t)$, balayant le plan de droite à gauche jusqu'au minimum $H_d = H_{d_1}$. Le processus continue en fonction des extrema de $H(t)$ et les sommets de l'interface $L(t)$ correspondent ainsi à l'historique ou à la mémoire du modèle de Preisach. Cependant, tel qu'il a été observé en pratique par Madelung en 1905 [10], l'augmentation de l'excitation à la saturation a pour effet d'effacer la contribu-

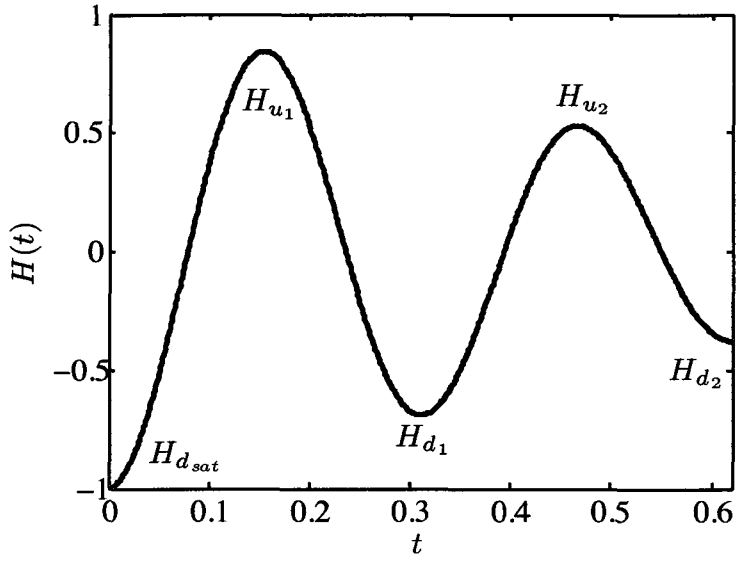


Figure 2.3 Excitation correspondant à la formation de $L(t)$

tion des extrema passés. Cette propriété est appelée propriété d'effacement et elle se traduit dans le plan de Preisach à l'effacement d'un sommet de l'interface $L(t)$ lorsque le segment de droite le dépasse.

En pratique, le problème consiste à identifer la fonction de poids $\omega(H_u, H_d)$, statistiquement, à l'aide des courbes de premier renversement. De surcroît, l'intégrale double de l'équation (2.1) est coûteuse en temps de calcul et c'est alors que D.H. Everett proposa en 1952 [14] une nouvelle approche en utilisant directement le résultat de l'intégrale double par rapport à une surface triangulaire, soit la fonction d'Everett :

$$\Omega(h_u, h_d) = \iint_{T(h_u,h_d)} \omega(H_u, H_d)\mathrm{d}H_u\mathrm{d}H_d \tag{2.4}$$

où $T(h_u, h_d)$ est le triangle défini par les limites $h_u \leq H_{u_{sat}}$, $h_d \leq H_{d_{sat}}$ et $H_d \leq H_u$. D'autre part, grâce aux généralisations mathématiques du modèle de Preisach par M.A. Krasnosel'skii et A.V. Pokrovskii [15], ainsi que plus récemment par I.D. Mayergoyz [8], le modèle de Preisach est de nos jours un modèle phénoménologique appliqué à plusieurs autres types d'hystérésis. Ainsi, malgré ses fondements dans la physique du comportement

des matériaux magnétiques, le modèle est maintenant davantage un artifice mathématique qu'un outil pour expliquer les fondements de l'hystérésis.

2.2 Modèle de Stoner et Wolhfarth

Le modèle Stoner-Wolhfarth a été élaboré en 1948 par E.C. Stoner et E.P. Wohlfarth [16]. Il est composé de particules ellipsoïdes, plus petites que les domaines magnétiques et qui n'intéragissent pas entre elles. Ce modèle est un modèle fondamental qui se base sur la minimisation de l'énergie magnétocristaline dans un matériau anisotrope en fonction du champ magnétique donné, ainsi que l'orientation de la magnétisation de chaque particule par rapport à leur axe préférentiel, tel qu'illusté à la figure 2.4. Cette énergie est donnée, pour les sphéroïdes oblate et prolate, par :

$$E = \frac{1}{4}(N_b + N_a)M_0^2 - \frac{1}{4}(N_b - N_a)M_0^2 \cos(2\psi) - HM_0 \cos(\phi) \tag{2.5}$$

où N_a et N_b sont les coefficients de démagnétisation par rapport aux axes polaire a et équatoriale b et où M_0 est le vecteur de magnétisation de la particule. Pour obtenir la magnétisation globale de la contribution de ces particules, on considère que ces particules sont distribuées uniformément dans le matériau et que l'angle entre l'axe préférentiel et le champ magnétique suit une loi gaussienne [17] :

$$F(\theta) = \frac{1}{\sqrt{2\pi}\sigma} \exp \frac{-(\theta - \overline{\theta})^2}{2\sigma^2} \tag{2.6}$$

où $\overline{\theta}$ est l'espérance et σ est l'écart type. Le modèle Stoner-Wolhfarth présente l'avantage de générer les parties réversible et irréversible de la magnétisation à l'aide d'un seul modèle. Cependant, ce modèle présente une mémoire locale et le phénomène d'hystérésis est causé seulement par l'anisotropie du matériau, alors qu'en réalité, il y a une partie liée à l'intéraction entre les particules, ainsi qu'à l'effet de blocage [17].

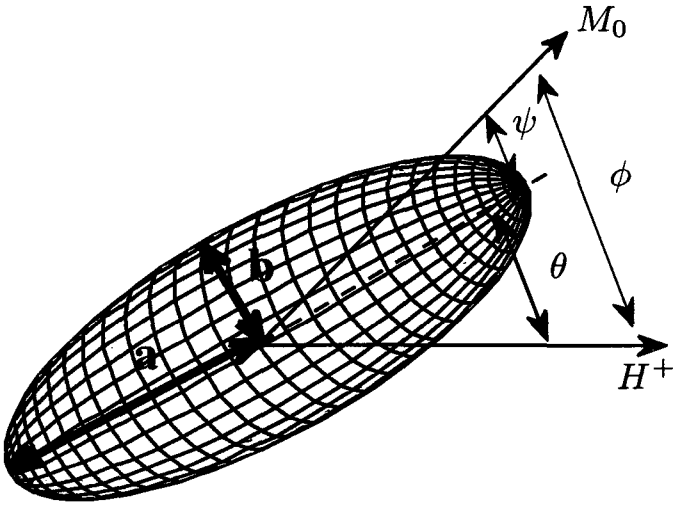


Figure 2.4 Particule du modèle de Stoner-Wolhfarth

2.3 Modèle de Jiles et Atherton

Le modèle Jiles-Atherton a été développé par D.C. Jiles et D.L. Atherton en 1983 [18]. C'est un modèle phénoménologique basé sur l'isotropie globale de plusieurs domaines et où la magnétisation est causée principalement par le mouvement des murs des domaines. L'énergie par unité de volume d'un domaine est donnée par :

$$E = -\mu_0 m H \tag{2.7}$$

où μ_0 est la perméabilité du vide, m est le moment magnétique par unité de volume et H est le champ. De plus, il faut considérer l'intéraction entre les domaines du matériau et l'on obtient :

$$E = -\mu_0 m(H + \alpha M) \tag{2.8}$$

où α est un paramètre qui représente le couplage entre les différents domaines. Pour représenter la magnétisation globale du matériau à partir du champ effectif $H_e = H + \alpha M$, le

modèle de Jiles et Atherton emploi l'équation de Langevin [19] :

$$M = M_{sat}\left(\coth\frac{\mu_0 m H_e}{k_B T} - \frac{k_B T}{\mu_0 m H_e}\right) \tag{2.9}$$

où M_{sat} est la magnétisation à la saturation, k_B est la constante de Boltzmann et T est la température en Kelvin. Bien sûr, l'équation précédente ne convient pas tout à fait pour l'étude de la magnétisation des matériaux ferromagnétiques, car elle ne tient pas compte de l'effet d'épinglage qui bloque le mouvement des murs de domaines. Pour en tenir compte, on introduit un terme différentiel à (2.9) :

$$M = M_{sat}\left(\coth\frac{\mu_0 m H_e}{k_B T} - \frac{k_B T}{\mu_0 m H_e}\right) - \delta k\left(\frac{\mathrm{d}M}{\mathrm{d}B_e}\right) \tag{2.10}$$

où le paramètre $\delta = 1$ quand le champ augmente et $\delta = -1$ quand le champ diminue. Le paramètre k peut être constant pour simplifier le problème, mais en pratique il varie en fonction de M et H.

Le problème de ce modèle est qu'il est nécessaire de connaître d'avance les valeurs d'extrema H_{r_n} et $H_{r_{n+1}}$ pour calculer la trajectoire d'une boucle mineure. Ce désavantage considérable pour la simulation dans un contexte de réseaux électriques peut être contourné en adoptant une approche géométrique, en utilisant la trajectoire de la boucle majeure avec un décalage et en la réduisant à la bonne échelle [20].

CHAPITRE 3

REVUE DES MODÈLES NUMÉRIQUES ACTUELS

Le présent chapitre traite des modèles numériques actuels appliqués plus spécifiquement à la simulation des transitoires électromagnétiques des transformateurs. La liste n'est pas exhaustive, mais les principaux modèles de type EMTP y sont explicités.

3.1 Modèle Type 96

Le modèle Type 96 est un modèle d'hystérésis implémenté dans EMTP-V3. La théorie entourant la base de ce modèle a été originellement élaborée par S. N. Talukdar et J. R. Bailey en 1976 [21], pour être par la suite améliorée en 1982 par J. G. Frame, N. Mohan et T.-H. Liu. [22]. Au départ, l'idée était de créer un modèle simple et efficace. Ainsi, les branches de la boucle majeure sont définies par des fonctions linéaires par partie et l'on assume que les trajectoires des boucles mineures sont contenues à l'intérieur de la boucle majeure. Celle-ci est définie par des segments de la forme :

$$\Phi = b_k i + a_k \tag{3.1}$$

où k est le numéro du segment, b_k est sa pente et a_k son ordonnée à l'origine. D'autre part, on assume que les deux branches de la boucle majeure se rejoignent à un point de saturation (i_{sat}, Φ_{sat}), à partir duquel la relation non-linéaire devient une fonction. Pour définir les boucles mineures, on assume que la trajectoire courante vise l'avant dernier point de renversement noté $(i_{r_{n-1}}, \phi_{r_{n-1}})$. La distance verticale dans le plan $\phi - i$, entre la boucle majeure et le point de renversement actuel, (i_{r_n}, ϕ_{r_n}) est notée D_{r_n} et celle avec l'avant dernier point est notée $D_{r_{n-1}}$. On assume que la distance verticale entre la boucle

majeure et la trajectoire varie linéairement. On aura alors :

$$D(\phi) = D_s\phi + D_0 \tag{3.2}$$

$$D_s = \frac{D_{r_n} - D_{r_{n-1}}}{\phi_{r_n} - \phi_{r_{n-1}}} \tag{3.3}$$

$$D_0 = \phi_{r_{n-1}} - D_s\phi_{r_n} \tag{3.4}$$

où D_s est la pente et D_0 est l'ordonnée à l'origine de cette droite. On peut alors formuler l'expression pour calculer les boucles mineures :

$$\phi = \Phi - D(\phi) \tag{3.5}$$

Pour représenter les éléments non-linéaires dans EMTP, il est nécessaire de linéariser localement la relation $\phi(i)$. Alors, le modèle doit retourner un équivalent Norton au réseau pour qu'il soit solutionné. Cependant, le Type 96 est un modèle pseudo non-linéaire, i.e. qu'il n'est pas solutionné simultannément avec le réseau. L'équivalent Norton pour le modèle est donné par :

$$R_N = \frac{2b'_k}{\Delta t} \tag{3.6}$$

$$I_N = \frac{\phi(t-\Delta t) - a'_k + \frac{\Delta t}{2}v(t-\Delta t)}{b'_k} \tag{3.7}$$

où a'_k et b'_k sont donnés par :

$$a'_k = b'_k(\frac{D_0 + a_k}{b_k}) \tag{3.8}$$

$$b'_k = \frac{b_k}{1 - D_s} \tag{3.9}$$

Enfin, plusieurs cas spéciaux d'opération demandent des traitements séparés. Par exemple, lors du calcul du point cible de la trajectoire courante après un renversement, il est possible que la trajectoire sorte de la boucle majeure si le point de renversement précédent était dans la saturation. On choisira alors le point de saturation comme avant dernier point de renversement.

3.1.1 Avantages et inconvénients

Dans un premier temps, on peut aisément conclure que le modèle est très rapide et simple, car il utilise des fonctions linéaires par partie pour représenter le phénomène non-linéaire. Cependant, plusieurs défauts, tant au niveau théorique que numérique, sont apparents [23]. En effet, la première lacune du modèle est qu'il ne possède pas de pile pour contenir les extrema. Cela entraîne un problème lorsqu'une boucle mineure est fermée et qu'il y a un dépassement, tel qu'illustré à la figure 3.1. Après avoir fermé la boucle entre les points 3 et 4, si l'excitation dépasse le point 3 le nouveau point visé devient le point de saturation, alors qu'en réalité, il aurait dû viser le point 1. Le Type 96 est donc un modèle à mémoire locale, ce qui n'est pas valide en pratique [8] et la boucle formée par les points 1 et 2 n'est pas fermée. D'autre part, puisque la distance entre la boucle majeure et une boucle mineure diminue linéairement, les boucles mineures sont toutes de la même forme, peu importe le niveau d'excitation et peu importe l'historique de magnétisation. On dit alors que les boucles sont congruentes, ce qui n'est pas vérifié en pratique [8]. De plus, il n'existe pas en réalité de point de saturation à proprement dit, car les pertes hystérétiques seraient fixes dans la saturation, peu importe le niveau d'excitation. Le comportement est plutôt asymptotique et les pertes augmentent avec l'excitation. Finalement, l'autre problème du Type-96 est qu'il soit pseudo non-linéaire. Lorsque l'élément non-linéaire n'est pas solutionné simultannément avec le réseau, des conditions illégales d'opérations peuvent survenir, par exemple s'il y a une discontinuité ou un changement brusque de pente entre deux segments linéaires du modèle. L'équivalent Norton peut être corrigé au pas de temps suivant, mais le point illégal est enregistré dans le temps. Pour remédier au problème, on peut utiliser un grand nombre de segments pour représenter la boucle majeure et diminuer le pas d'intégration significativement, mais on perd alors toute la force du modèle qui était sa rapidité.

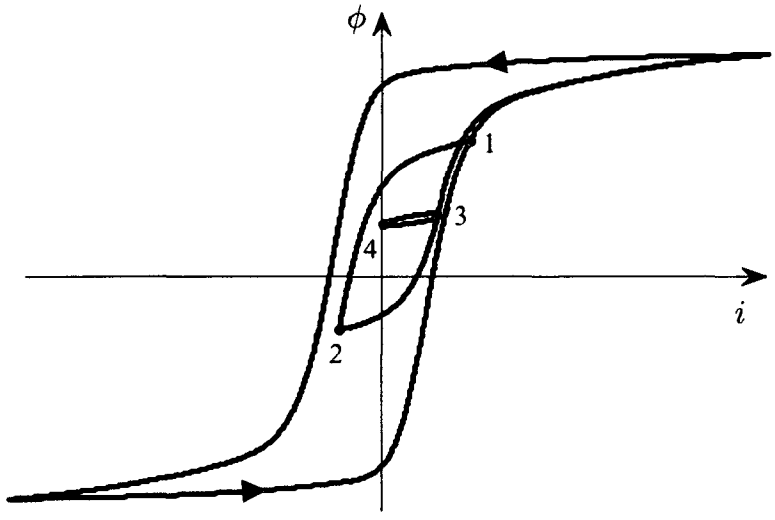
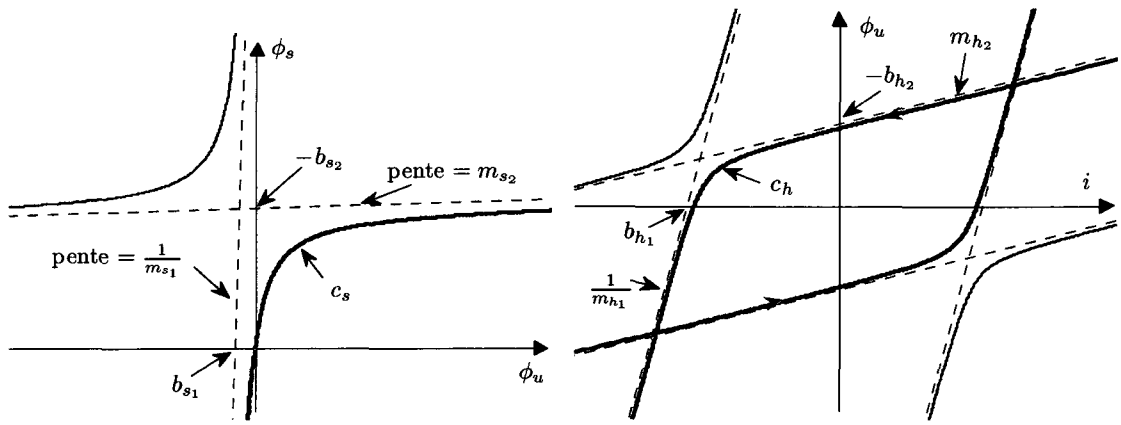


Figure 3.1 Boucle mineure ouverte

3.2 Modèle Type 92

Le modèle Type 92 a initialement été élaboré pour EMTP-V3 en 1996 par A. Narang, E. P. Dick et R. C. Cheung [24] et il est basé sur les théories trouvées dans [25][26]. Par la suite, le modèle a été amélioré par S. Dennetière en 2003 et il a été implémenté dans EMTP-RV [23]. Au départ, les défauts du Type 96 avaient été soulignés et une méthode de compensation existait dans EMTP-V3 pour solutionner les éléments non-linéaires de façon simultanée avec le réseau linéaire [27]. Alors il fût décidé d'avoir recours à un modèle non-linéaire continu, plutôt que linéaire par partie, ainsi que d'utiliser la méthode de compensation pour solutionner simultanément le modèle et le réseau. Dans ce modèle, le flux instantanné, appelé flux de saturation ϕ_s, est séparé en deux contributions ; la première est une fonction de saturation réversible et la deuxième est la relation hystérétique irréversible. La sortie de la fonction de saturation donne le flux insaturé ϕ_u à partir d'un flux saturé ϕ_s. Par la suite, le flux insaturé est utilisé par la relation hystérétique pour calculer le courant. Pour représenter l'hystérésis et la saturation, le Type 92 a recours à des hyperboles, définies par :

$$(\phi_u - m_{s_1}\phi_s - b_{s_1})(m_{s_2}\phi_u - \phi_s - b_{s_2}) = c_s \tag{3.10}$$



(a) Fonction de saturation (b) Relation hystérétique

Figure 3.2 Relations hyperboliques et leurs paramètres

pour la fonction de saturation et par :

$$(i - m_{h_1}\phi_u - b_{h_1})(m_{h_2}i - \phi_u - b_{h_2}) = c_h \tag{3.11}$$

pour la relation hystérétique. L'indice s représente la saturation, tandis que l'indice h représente la partie hystérétique. Les paramètres c règlent la courbure des fonctions hyperboliques, les paramètres m modifient la pente des asymptotes et les paramètres b représentent leurs ordonnées à l'origine. Ces paramètres sont déterminés à partir d'un régresseur non-linéaire utilisant la méthode des moindres carrés non-linéaire. Les deux hyperboles avec leurs paramètres respectifs sont présentées aux figures 3.2a et 3.2b. Les équations à résoudre (3.10) et (3.11) sont quadratiques, alors elles peuvent admettre deux solutions. La solution choisie dépendra de si la trajectoire est ascendante ou descendante. Pour obtenir les boucles mineures, il suffit d'effectuer une translation des asymptotes de la relation hystérétique, en modifiant les ordonnées à l'origine b_{h_1} et b_{h_2}. Cela aura pour effet de modifier la position de l'hyperbole hystérétique, pour la faire coïncider avec le dernier point de renversement. Contrairement au Type 96, le Type 92 mémorise les extrema passés dans une pile, de façon à ce que le problème de boucle mineure ouverte soit résolu.

3.2.1 Avantages et inconvénients

Tout d'abord, le Type 92 a bien sûr résolu les problèmes associés à l'implémentation pseudo non-linéaire du Type 96. De surcroît, l'ajout d'une pile pour l'historique des extrema a réglé le problème de boucle mineure ouverte et l'utilisation de fonctions asymptotiques a réglé le problème de pertes fixes à la saturation. D'autre part, le Type 92 est un modèle rapide, efficace et plus précis que son prédécesseur. Cependant, l'antisymétrie des fonctions utilisées est contradictoire avec les résultats expérimentaux présentés dans [24], qui présentent des courbes hystérétiques avec une silhouette en col d'oie. Enfin, les résultats de régressions dictés par ces fonctions hyperboliques sont questionnables, tel qu'illustré dans [28]. Ainsi, pour bien reproduire un large éventail de courbes expérimentales, plus de degrés de liberté seraient nécessaires.

3.3 Modèle de Preisach modifié

Un modèle de Preisach modifié, appliqué pour la simulation des transitoires électromagnétiques, a été élaboré pour PSCAD/EMTDC [29] en 2007 par A. Rezaei [30][31]. L'implémentation d'un modèle de Preisach dans un contexte numérique se heurte à plusieurs problèmes. L'évaluation des intégrales doubles de (2.3) est coûteuse en temps de calcul et la fonction de poids $\omega(H_u, H_d)$ est difficile à obtenir à partir des données expérimentales, car elle est trouvée à partir de dérivées partielles de mesures expérimentales, amplifiant ainsi le bruit de mesure [8]. Pour surmonter ces difficultés, on peut utiliser l'approche proposée par D. H. Everett, présentée à la section 2.1. On peut alors utiliser directement le résultat de l'intégrale double, plutôt que de la calculer à chaque itération. Dans un premier temps, la densité de flux B du transformateur peut être séparée en deux contributions : celle du noyau ferromagnétique B_{hys}, qui sature à une valeur de B_{sat}, et une partie pour représenter la perméabilité du milieu ambiant B_0, qui ne sature pas et représente la pente

de la courbe $B(H)$ dans la région de la saturation :

$$B(H) = B_{hys}(H) + B_0(H) \tag{3.12}$$

La contribution du milieu ambiant peut être considérée linéaire et proportionnelle à l'inductance de l'air. Pour la partie non-linéaire B_{hys}, elle peut-être calculée à partir du modèle de Preisach comme étant :

$$B_{hys}(t) = -B_{sat} + 2\sum_{k=1}^{n(t)-1}\left[\Omega(H_{u_k}, H_{d_{k-1}}) - \Omega(H_{u_k}, H_{d_k})\right] + 2\Omega(H_e(t), H_{d_{n-1}}) \tag{3.13}$$

pour une trajectoire ascendante et :

$$\begin{aligned} B_{hys}(t) \quad = \quad & -B_{sat} + 2\sum_{k=1}^{n(t)-1}\left[\Omega(H_{u_k}, H_{d_{k-1}}) - \Omega(H_{u_k}, H_{d_k})\right] \\ & + 2\left[\Omega(H_{u_n}, H_{d_{n-1}}) - \Omega(H_{u_n}, H_e(t))\right] \end{aligned} \tag{3.14}$$

pour la trajectoire descendante. Les termes H_{u_k} et H_{d_k} représentent les valeurs de champ magnétique aux extrema, tel qu'illustré à la figure 2.3. Puisque le nombre d'extrema dans la pile varie dans le temps, n est fonction du temps et H_e est le champ magnétique effectif. D'autre part, la fonction $\Omega(H_u, H_d)$ est la fonction d'Everett présentée à la section 2.1. Dans le modèle de Preisach modifié, une fonction non-linéaire est utilisée pour reproduire les courbes de premier renversement expérimentales et le résultat de l'intégrale double d'Everett sera donné par :

$$\Omega\left(x(H_u, H_d)\right) = G(x)\left(1 - \sum_{j=1}^{n} k_{bj}\exp\left(-k_{pj}x\right)\right) \tag{3.15}$$

où la somme des k_{bj} doit être comprise entre 0 et 1 et où les paramètres k_{pj} sont positifs. La variable x correspond aux valeurs de champ magnétique H_u ou H_d. Les termes exponentiels servent à modifier les boucles mineures pour qu'elles ne soient pas congruentes et la boucle

majeure est donnée par la fonction $G(x)$:

$$G(x) = \frac{1}{2} \sum_{i=1}^{q} B_i \left[\tanh\left(P_i x\right) + C_i \operatorname{sech}^2(P_i x) \right] \tag{3.16}$$

où q est ajusté pour avoir le nombre de degrés de liberté désiré lors de la régression, B_i sont les paramètres d'amplitude dont la somme doit être égale à B_{sat}, les paramètres P_i ajustent la pente des fonctions et les paramètres C_i ajustent l'amplitude relative de la sécante hyperbolique par rapport à la tangente hyperbolique. De plus, les paramètres P_i doivent être strictement positifs et les paramètres C_i doivent être compris entre 0 et 0.5 pour la trajectoire ascendante et entre -0.5 et 0 pour la trajectoire descendante. Tous ces paramètres sont obtenus à l'aide de la méthode de régression non-linéaire par région de confiance à l'aide de la trajectoire ascendante de la boucle majeure.

3.3.1 Avantages et inconvénients

De prime abord, l'utilisation de la fonction d'Everett simplifie grandement les calculs et rend le modèle de Preisach numérique possible. La précision d'un tel modèle est impressionnante et la rapidité de calcul se rapproche des performances observées avec le Type 92. D'autre part, l'ajout de la sécante hyperbolique dans (3.16) aura pour effet de briser l'antisymétrie de la fonction et permettre, par le fait même, de reproduire la forme évasée des boucles majeures expérimentales. Cependant, le modèle de Preisach requiert les courbes de premier renversement expérimentales qui sont rarement disponibles. De plus, la coercivité de la boucle majeure est obtenue par l'addition des sécantes hyperboliques de (3.16), car les tangentes hyperboliques passent par l'origine. Ainsi, pour augmenter la coercivité, il faut augmenter la valeur des C_i, mais celle-ci est limitée à 0.5, en valeur absolue et cela aura aussi pour effet de modifier la courbure de la boucle majeure. On peut aussi ajuster les P_i pour modifier la largeur de la sécante hyperbolique, mais cela modifiera du même coup la pente de la tangente hyperbolique et donc la perméabilité au point coercitif. Cela aura pour effet de diminuer la qualité de la régression pour les boucles majeures avec de plus grandes

coercivités. Il faudrait donc modifier la fonction (3.16) pour découpler les deux termes et ajouter un déphasage à ceux-ci, pour donner plus de flexibilité lors de la régression.

3.4 Modèle de MATLAB/Power System Blockset

Le modèle d'inductance hystérétique de MATLAB/Power System Blockset [32] fait partie intégrante du modèle de transformateur saturable et il a été implémenté par S. Casoria, P. Brunelle et G. Sybille en 2003 [33]. La théorie sous-jacente est basée sur le modèle Type 96, sauf que l'équivalent Norton du Type 96 est remplacé par une source de courant contrôlée par la tension à ses bornes. De plus, la boucle majeure est donnée par l'expression analytique :

$$\Phi = -a \arctan(-bI + c) + \alpha I - e \tag{3.17}$$

pour la branche ascendante et :

$$\Phi = a \arctan(bI + c) + \alpha I + e \tag{3.18}$$

pour la branche descendante. Le paramètre a représente l'amplitude de l'arctangente, le paramètre b modifie la pente au point coercitif, le paramètre c change le déphasage de la fonction, le paramètre α représente l'inductance de l'air et le paramètre e est une translation verticale utilisée pour que les courbes se rejoignent au point de saturation (i_{sat}, Φ_{sat}). D'autre part, puisque c'est le flux qui est considéré comme étant l'excitation lors de la simulation de transitoires électromagnétiques, la relation inverse $I(\Phi)$ est nécessaire. De surcroît, pour accélérer le calcul du courant à chaque pas de temps, les valeurs sont précalculées et enregistrées à l'intérieur d'un tableau. Ainsi, le modèle utilise des fonctions linéaires par partie pour représenter la boucle majeure, comme pour le Type 96. D'ailleurs, le modèle de MATLAB/Power System Blockset est aussi un modèle non simultanné et les boucles mineures sont aussi calculées avec la translation $D(\phi)$. Cependant, contrairement au Type 96, ce modèle possède une pile pour mémoriser les extrema et deux tolérances

additionnelles permettent de simplifier les calculs. La première tolérance élimine le besoin de recalculer la trajectoire d'une boucle mineure fermée si elle est suffisamment proche de la boucle mineure fermée précédente. La deuxième tolérance évite de calculer la trajectoire d'une boucle mineure si le point de renversement est proche de l'avant dernier point de renversement. On assume alors que l'évolution entre les deux points est linéaire et que la surface de la boucle fermée est négligeable.

3.4.1 Avantages et inconvénients

L'avantage principal du modèle, comme pour le Type 96, est qu'il soit simple et rapide. D'autre part, l'inconvénient de mémoire locale du Type 96 a été éliminé en ajoutant une pile pour mémoriser les extrema et l'ajout de tolérances pour le calcul des boucles mineures en fait un candidat idéal pour la simulation en temps réel. Cependant, comme pour le Type 96, l'existance d'un point de saturation limite l'augmentation des pertes hystérétiques dans la saturation et l'utilisation d'un modèle non simultanné pourrait créer des conditions d'opération illégales. Aussi, comme dans le cas du Type 96, les boucles mineures sont congruentes et cette propriété n'est pas vérifiée en pratique.

3.5 Modèle de résistance de magnétisation instantannée

Le modèle de résistance instantanné a été élaboré par A. Gaudreau, P. Picher, L. Bolduc et A. Coutu en 2002 [34] pour EMTP-V3. L'utilisation d'une résistance hystérétique plutôt qu'une inductance hystérétique est logique, car la surface sous la courbe introduite par l'hystérésis dans le plan $\phi - i$ représente les pertes. Ainsi, une résistance qui varie en fonction du flux est fondamentalement plus appropriée pour représenter ces pertes. Si l'on assume que la courbe anhystérétique passe au milieu de la boucle majeure et par le fait même que les branches sont antisymétriques, on peut exprimer la résistance instantannée

comme :

$$r_m = \left| \frac{v}{(i_+ - i_-)/2} \right| \tag{3.19}$$

pour les trajectoires ascendantes et :

$$r_m = \left| \frac{v}{(i_- - i_+)/2} \right| \tag{3.20}$$

pour les trajectoires descendantes. Les courants i_+ et i_- sont les courants instantannés des trajectoires ascendantes et descendantes de la boucle majeure expérimentale à flux nominal et v est la tension instantannée. On peut alors déterminer à l'aide des courbes expérimentales une fonction $r_m(\phi)$ linéaire par partie et la multiplier par le coefficient :

$$0.21 \left(\frac{\hat{V}}{\hat{V}_{1.779T}} \right)^2 + 0.79 \tag{3.21}$$

pour différents niveaux de surexcitation crête, par rapport à un niveau de tension crête pour une densité de flux de 1.779 T. Le modèle a été ensuite implémenté dans EMTP-RV à l'aide des modules de contrôle pour choisir la résistance de magnétisation instantannée en fonction du flux instantanné en pu. Une interpolation linéaire est utilisée pour trouver la résistance à partir de la fonction linéaire par partie.

3.5.1 Avantages et inconvénients

Comme pour le Type 96 et le modèle de MATLAB/Power System Blockset, l'utilisation d'une fonction linéaire par partie simplifie grandement le calcul. Cependant, le modèle ne tient aucunement compte de l'historique de magnétisation du transformateur et il convient pour des études en régime permanent où le flux varie de façon symétrique. Pour les trajectoires de boucles mineures asymétriques, la résistance r_m n'est plus symétrique par rapport à l'ordonnée, comme le confirme les résultats de courant d'appel présentés dans [34]. Par conséquent, le modèle de résistance de magnétisation instantannée est moins intéressant d'un point de vue de la simulation des transitoires électromagnétiques.

CHAPITRE 4

NOUVEAU MODÈLE

Puisque le modèle proposé est un modèle phénoménologique, il importe de trouver une fonction ou un ensemble de fonctions qui reproduit adéquatement la courbe hystérétique expérimentale. Dans un premier temps, la fonction doit être asymptotique dans la région de saturation, où les deux branches de la boucle majeure se rejoignent. Deuxièmement, pour tenir compte de l'inductance de l'air ou du milieu environnant, la pente de l'asymptote dans cette région doit être ajustable, car cet inductance est généralement non négligeable. Troisièmement, le nouveau modèle doit permettre de reproduire la surface plus large de la courbe d'hystérésis sous la saturation. Cette condition implique que la portion inférieure de la branche ascendante de la boucle majeure ne soit pas une rotation de 180 degrés de la partie supérieure et donc que la courbe ne soit pas antisymétrique. Quatrièmement, la fonction requiert un décalage horizontal, afin de tenir compte de la coercivité ainsi que de la rémanence. Enfin, le modèle doit avoir assez de degrés de liberté pour obtenir de bonnes régressions non-linéaires à partir d'un large éventail de courbes expérimentales [35].

4.1 Boucle majeure

Pour illustrer les conditions précédentes, une boucle majeure expérimentale typique est présentée à la figure 4.1. On remarque que la courbe s'évase sous la région de la saturation et que la pente de l'asymptote en saturation est finie, i.e. que l'inductance est non nulle. Maintenant, il faut trouver des fonctions reproduisant cette caractéristique. Pour y arriver, on utilise depuis longtemps les fonctions hyperboliques, car elles sont, par définition, asymptotiques [19][36][37]. Plus particulièrement, il y a deux fonctions très intéressantes qui méritent d'être examinées de plus près, soit la tangente hyperbolique et la sécante hyperbolique.

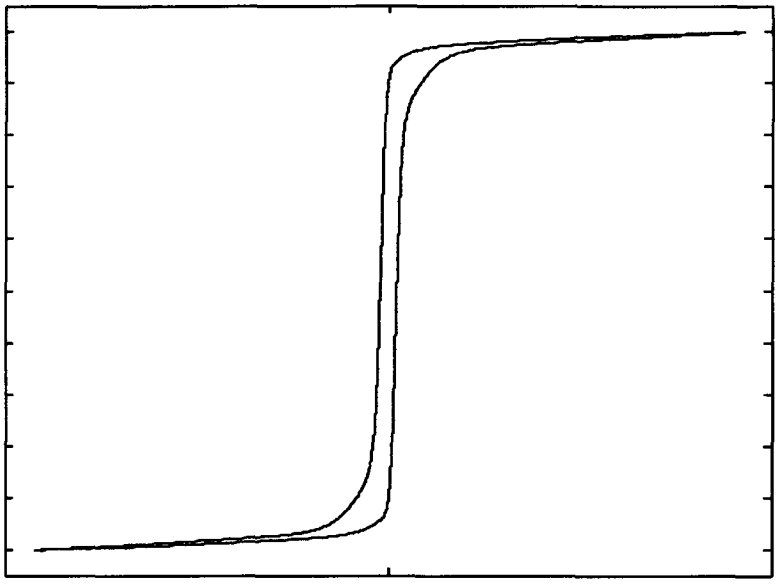


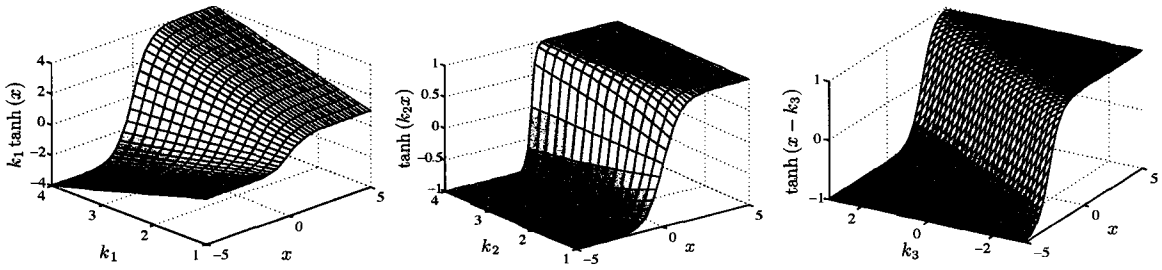
Figure 4.1 Boucle majeure expérimentale typique

Dans un premier temps, on peut définir :

$$f_1(x) = k_1 \tanh(k_2 x - k_3) \tag{4.1}$$

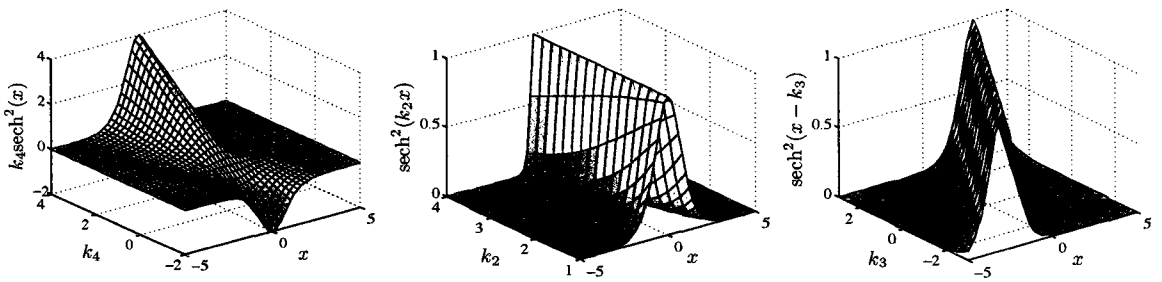
$$f_2(x) = k_4 \operatorname{sech}^2(k_2 x - k_3) \tag{4.2}$$

où les paramètres k_1 et k_4 définissent l'amplitude de ces fonctions, le paramètre k_2 défini le facteur d'échelle horizontal et le paramètre k_3 représente la translation horizontale. La sécante hyperbolique est élevée au carré pour que la fonction soit plus abrupte. On peut constater l'effet de la variation de ces paramètres aux figures 4.2 et 4.3. De ce fait, on voit que la tangente hyperbolique est très ressemblante à l'aspect sigmoïdal de la boucle majeure. À l'aide du paramètre k_3, on peut ajuster la fonction à la bonne coercivité et le paramètre k_1 à la bonne amplitude. Ensuite, le paramètre k_2 peut être ajusté afin d'obtenir la bonne perméabilité initiale. Cependant, comme la courbe expérimentale n'est pas antisymétrique et pour permettre de reproduire correctement la courbure de la boucle majeure, on ajoute à la fonction précédente la sécante hyperbolique élevée au carrée. De plus, dans le but d'effectuer de bonnes régressions sur une large gamme de courbes, il importe d'ajouter



(a) Variations avec k_1 (b) Variations avec k_2 (c) Variations avec k_3

Figure 4.2 La fonction tangente hyperbolique en relation avec ses paramètres



(a) Variations avec k_4 (b) Variations avec k_2 (c) Variations avec k_3

Figure 4.3 La fonction sécante hyperbolique en relation avec ses paramètres

quelques degrés de liberté supplémentaires. Pour ce faire, on peut ajouter quelques tangentes et sécantes hyperboliques additionnelles avec leurs paramètres distincts. Enfin, pour tenir compte de l'inductance du milieu ambiant à la saturation, on ajoute un terme linéaire à la somme précédente. On obtient alors le modèle A(x), inspiré des travaux de [9] et [31], dont la branche ascendante de la boucle majeure est définie par la fonction :

$$\begin{aligned} A(x) \quad = \quad & k_1 \tanh(k_2 x - k_3) - k_1 k_4 \text{sech}^2(k_2 x - k_3) \\ & + k_5 \tanh(k_6 x - k_7) - k_5 k_8 \text{sech}^2(k_6 x - k_7) \\ & + k_9 \tanh(k_{10} x - k_{11}) - k_9 k_{12} \text{sech}^2(k_{10} x - k_{11}) \\ & + k_{13} x \end{aligned} \tag{4.3}$$

Si l'on traduit le modèle précédent dans le contexte de l'hystérésis magnétique, on a alors :

$$\begin{aligned}\Phi_+(i) &= k_1 \tanh(k_2 i - k_3) - k_1 k_4 \text{sech}^2(k_2 i - k_3) \\ &\quad + k_5 \tanh(k_6 i - k_7) - k_5 k_8 \text{sech}^2(k_6 i - k_7) \\ &\quad + k_9 \tanh(k_{10} i - k_{11}) - k_9 k_{12} \text{sech}^2(k_{10} i - k_{11}) \\ &\quad + k_{13} i \end{aligned} \tag{4.4}$$

$$\begin{aligned}\Phi_-(i) &= k_1 \tanh(k_2 i + k_3) + k_1 k_4 \text{sech}^2(k_2 i + k_3) \\ &\quad + k_5 \tanh(k_6 i + k_7) + k_5 k_8 \text{sech}^2(k_6 i + k_7) \\ &\quad + k_9 \tanh(k_{10} i + k_{11}) + k_9 k_{12} \text{sech}^2(k_{10} i + k_{11}) \\ &\quad + k_{13} i \end{aligned} \tag{4.5}$$

Le flux de la partie ascendante de la boucle majeure est notée $\Phi_+(i)$, tandis que la partie descendante est notée $\Phi_-(i)$, où i est le courant de la branche de magnétisation. Les termes k_1 à k_{13} sont des constantes déterminées à l'aide d'une méthode de régression non-linéaire avec des données expérimentales constituant la partie ascendante de la boucle majeure. Puisque la fonction 4.4 n'est pas antisymétrique, il est préférable d'utiliser les données complètes de la branche ascendante et non seulement les données qui figurent dans le premier quadrant pour la régression. Enfin, les trajectoires des boucles mineures et majeure sont illustrées à la figure 4.4.

4.2 Boucles mineures

Maintenant, pour définir les trajectoires des boucles mineures, il faut se rappeler que toute trajectoire doit être contenue à l'intérieur de la boucle majeure. De plus, le flux des trajectoires ascendante et descendante doit être égal au point de renversement de courant i_{r_n}, où n est l'ordre de renversement. Ainsi, il faut effectuer une translation des ordonnées pour que les courbes se rencontrent au point (i_{r_n}, ϕ_{r_n}) [9]. Cela équivaut à introduire une constante

dans les équations 4.4 et 4.5. Cependant, cette translation fait en sorte que les courbes ascendante et descendante ne se rejoignent plus à la saturation. Ainsi, il faut utiliser un paramètre, plutôt qu'une constante, qui égalise le flux des trajectoires ascendante et descendante au point de retournement et qui devient graduellement nul lorsque la trajectoire s'approche de la saturation. On obtient alors les équations générales du modèle :

$$\phi_+(i) = \Phi_+(i) + C_{n+}(i) \tag{4.6}$$

$$\phi_-(i) = \Phi_-(i) + C_{n-}(i) \tag{4.7}$$

où $C_{n+}(i)$ et $C_{n-}(i)$ sont les paramètres qui modifient respectivement les trajectoires ascendante et descendante de la boucle majeure en fonction du courant. On calcule ces paramètres à partir des équations suivantes :

$$C_{n+}(i) = C_{nu+}\left(\frac{a_+(i_{r_{n-1}}) - a_+(i)}{a_+(i_{r_{n-1}}) - a_+(i_{r_n})}\right) + C_{nd+}\left(\frac{a_+(i_{r_n}) - a_+(i)}{a_+(i_{r_n}) - a_+(i_{r_{n-1}})}\right) \tag{4.8}$$

$$C_{n-}(i) = C_{nu-}\left(\frac{a_-(i_{r_{n-1}}) - a_-(i)}{a_-(i_{r_{n-1}}) - a_-(i_{r_n})}\right) + C_{nd-}\left(\frac{a_-(i_{r_n}) - a_-(i)}{a_-(i_{r_n}) - a_-(i_{r_{n-1}})}\right) \tag{4.9}$$

Seuls les termes $a_+(i)$ et $a_-(i)$ varient en fonction du courant, les autres sont constants et calculés à chaque renversement de courant avec le point actuel i_{r_n} et le point précédent $i_{r_{n-1}}$. Les constantes C_{nu+} et C_{nu-} font en sorte qu'au point de renversement (i_{r_n}, ϕ_{r_n}), les équations 4.6 et 4.7 sont égales et le paramètre $C_n(i_{r_n})$ est égal à C_{nu}. Ainsi, ces constantes correspondent à la différence de flux entre la boucle majeure et le dernier point de renversement. D'autre part, puisque la trajectoire doit nécessairement rejoindre le point $(i_{r_{n-1}}, \phi_{r_{n-1}})$ si l'amplitude de l'excitation repasse par cet extremum, il faut calculer les constantes C_{nd+} et C_{nd-} pour obtenir la même translation de flux en ce point. Le principe pour ces différentes constantes est illustré à la figure 4.4, où l'ordre de renversement n est égal à trois. Pour ce qui est des fonctions $a_+(i)$ et $a_-(i)$, elles servent de levier entre le point

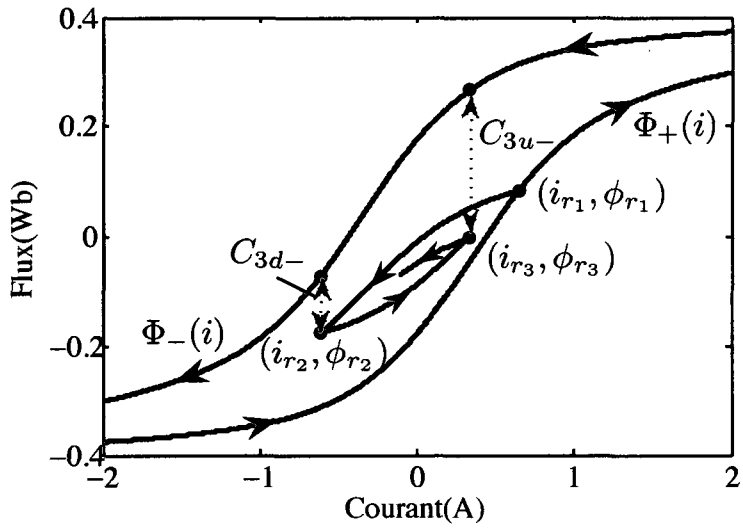


Figure 4.4 Trajectoires des boucles mineures

de renversement actuel i_{r_n} et le point précédent $i_{r_{n-1}}$. De plus, elles doivent être de la même forme que la boucle majeure pour ne pas sortir de celle-ci et l'on choisira préférablement des fonctions saturables, car si la trajectoire se situe dans la région de la saturation, les paramètres $C_{n+}(i)$ et $C_{n-}(i)$ devront être pratiquement nuls. Ainsi, on choisira les fonctions suivantes :

$$\begin{aligned} a_+(i) &= k_1 \tanh(k_2 i - k_3) - k_1 k_4 \mathrm{sech}^2(k_2 i - k_3) \\ &\quad + k_5 \tanh(k_6 i - k_7) - k_5 k_8 \mathrm{sech}^2(k_6 i - k_7) \\ &\quad + k_9 \tanh(k_{10} i - k_{11}) - k_9 k_{12} \mathrm{sech}^2(k_{10} i - k_{11}) \end{aligned} \tag{4.10}$$

$$\begin{aligned} a_-(i) &= k_1 \tanh(k_2 i + k_3) + k_1 k_4 \mathrm{sech}^2(k_2 i + k_3) \\ &\quad + k_5 \tanh(k_6 i + k_7) + k_5 k_8 \mathrm{sech}^2(k_6 i + k_7) \\ &\quad + k_9 \tanh(k_{10} i + k_{11}) + k_9 k_{12} \mathrm{sech}^2(k_{10} i + k_{11}) \end{aligned} \tag{4.11}$$

Toutefois, il faut être prudent en utilisant ces fonctions, car on remarque, d'après 4.8 et 4.9, qu'il y a une discontinuité lorsque $a_+(i_{r_n}) = a_+(i_{r_{n-1}})$ ou $a_-(i_{r_n}) = a_-(i_{r_{n-1}})$. En théorie, ce cas est seulement possible pour $i \longrightarrow \pm\infty$, car 4.10 et 4.11 sont asymptotiques et se rejoignent à l'infini. Cependant, en pratique, la précision numérique fait en sorte que ces fontions sont numériquement équivalentes pour des excitations beaucoup plus faibles. Ce cas correspond à un double renversement dans la région de la saturation. Pour résoudre ce problème, il suffit de poser $C_n = 0$, $C_{nd} = 0$, $C_{nu} = 0$ et $a(i_{r_{n-1}}) = 0$ lorsque le double renversement dans la saturation est détecté.

4.3 Courbe de première magnétisation

Enfin, un dernier aspect mérite d'être abordé, soit la courbe de première magnétisation. Lorsque le transformateur est dans l'état démagnétisé, la trajectoire est différente de celle qui résulterait d'un renversement de l'excitation au point $(0, 0)$ dans le plan $\phi - i$. On appelle généralement cette trajectoire la courbe de première magnétisation ou la courbe de magnétisation vierge et l'on peut apercevoir cette trajectoire particulière à la figure 4.5. Pour représenter cette courbe, il faut alors utiliser une fonction différente que l'on peut définir comme :

$$\begin{aligned} \phi_{vierge}(i) &= (k_1 \tanh(k_2 i) + k_5 \tanh(k_6 i) + k_9 \tanh(k_{10} i) \\ &\quad + k_{13} i)(1 - 2k_{14}\mathrm{sech}^2(k_{15} i)) \end{aligned} \tag{4.12}$$

où le paramètre k_{14} représente la courbure initiale, comprise entre zéro et une demie, inclusivement. Plus ce paramètre est élevé, plus la pente à l'origine sera faible. Quant à lui, le paramètre k_{15} permet de modifier la pente de la trajectoire sous la saturation et on l'ajuste pour que la courbe de première magnétisation soit comprise à l'intérieur de la boucle majeure. En général, on choisira comme valeur le maximum entre les paramètres k_2, k_6 et k_{10}, mais on peut évidemment l'ajuster, au besoin. On peut constater l'effet de la variation de ces paramètres à la figure 4.6.

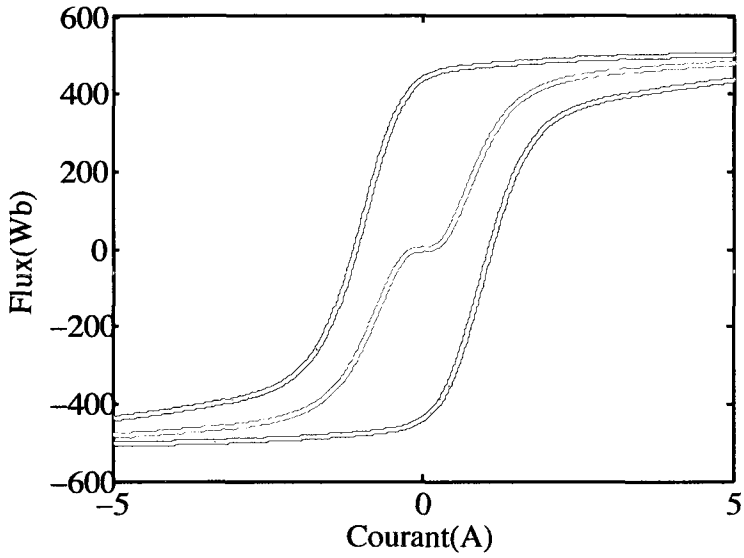
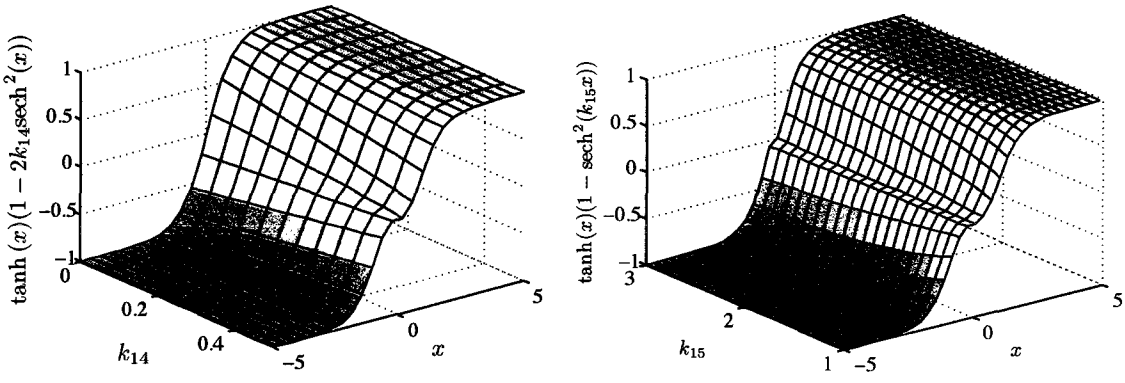


Figure 4.5 Courbe de première magnétisation



(a) Variations avec k_{14}

(b) Variations avec k_{15}

Figure 4.6 La courbe de première magnétisation en relation avec ses paramètres

CHAPITRE 5

CONTEXTE DE PROGRAMMATION

Le modèle A(x), élaboré au chapitre 4, a été implémenté dans EMTP-RV sous forme d'un DLL écrit en Fortran en utilisant les lignes directrices de [38]. D'autre part, son implémentation a été inspirée des travaux de [28] et [23]. Dans EMTP-RV, les éléments non-linéaires sont solutionnés simultannément avec le système d'équations linéaires. À chaque incrément de temps et de façon itérative, l'élément non-linéaire est linéarisé et remplacé par un équivalent Norton. Puisque la tension est l'inconnue à résoudre pour le système d'équations, le flux est considéré comme étant l'excitation de la branche de magnétisation dans ce contexte, plutôt que le courant.

5.1 Domaine du temps

Dans un premier temps, le processus itératif, pour solutionner cet élément non-linéaire à un point τ dans le temps, débute en calculant le flux. Celui-ci est calculé à partir de la tension de la branche de magnétisation, trouvée d'après l'équivalent Norton de l'itération précédente :

$$\phi_{km}(\tau) = \int_{\tau-\Delta t}^{\tau} v_{km}(t)\mathrm{d}t + \phi_{km}(\tau - \Delta t) \tag{5.1}$$

où k et m représentent les noeuds de la branche de magnétisation, v_{km} est la tension de l'élément et Δt est le pas d'intégration. L'équation (5.1) est solutionnée numériquement à l'aide de la méthode d'intégration trapézoïdale, ou dans le cas où il y aurait une discontinuité, avec la méthode d'Euler implicite (*Backward Euler*). Ainsi, on peut réécrire (5.1)

comme étant :

$$\phi_{km}(\tau) = \frac{\Delta t}{2} v_{km}(\tau) + \phi_{km\text{hist}}(\tau) \tag{5.2}$$

où le terme $\phi_{km\text{hist}}(\tau)$ dépend de la méthode d'intégration utilisée. Dans le cas de la méthode trapézoïdale, on aura :

$$\phi_{km\text{hist}}(\tau) = \frac{\Delta t}{2} v_{km}(\tau - \Delta t) + \phi_{km}(\tau - \Delta t) \tag{5.3}$$

et pour la méthode d'Euler implicite :

$$\phi_{km\text{hist}}(\tau) = \phi_{km}(\tau - \Delta t) \tag{5.4}$$

Ensuite, on trouve à l'aide de la méthode de Newton le courant $i_{km}(\tau)$ qui correspond au flux trouvé avec (5.2). En effet, puisque dans les équations (4.6) et (4.7) le courant est considéré comme étant l'excitation et puisque l'inverse analytique $i(\phi)$ n'existe pas, on doit trouver l'inverse de la fonction numériquement. Pour se faire, on utilise la méthode de Newton définie comme :

$${i_{km}}^{(n+1)}(\tau) = {i_{km}}^{(n)}(\tau) - \frac{f({i_{km}}^{(n)}(\tau))}{\mathrm{d}f} \tag{5.5}$$

où (n) représente le numéro de l'itération, f est la fonction à solutionner et $\mathrm{d}f$ est sa dérivée, qui sont données par :

$$f_+({i_{km}}^{(n)}(\tau)) = \Phi_+({i_{km}}^{(n)}(\tau)) + C_{n+}({i_{km}}^{(n)}(\tau)) - \phi(\tau) \tag{5.6}$$

$$\mathrm{d}f_+({i_{km}}^{(n)}(\tau)) = \mathrm{d}\Phi_+({i_{km}}^{(n)}(\tau)) + \mathrm{d}C_{n+}({i_{km}}^{(n)}(\tau)) \tag{5.7}$$

pour une trajectoire ascendante et :

$$f_-({i_{km}}^{(n)}(\tau)) = \Phi_-({i_{km}}^{(n)}(\tau)) + C_{n-}({i_{km}}^{(n)}(\tau)) - \phi(\tau) \tag{5.8}$$

$$\mathrm{d}f_-({i_{km}}^{(n)}(\tau)) = \mathrm{d}\Phi_-({i_{km}}^{(n)}(\tau)) + \mathrm{d}C_{n-}({i_{km}}^{(n)}(\tau)) \tag{5.9}$$

pour une trajectoire descendante. Les dérivées $\mathrm{d}\Phi(i)$ et $\mathrm{d}C_n(i)$ sont données par :

$$\begin{aligned}
\mathrm{d}\Phi_+(i) \quad = \quad & k_1k_2\mathrm{sech}^2(k_2i - k_3)(1 + 2k_4\tanh(k_2i - k_3)) \\
& + k_5k_6\mathrm{sech}^2(k_6i - k_7)(1 + 2k_8\tanh(k_6i - k_7)) \\
& + k_9k_{10}\mathrm{sech}^2(k_{10}i - k_{11})(1 + 2k_{12}\tanh(k_{10}i - k_{11})) + k_{13}
\end{aligned} \tag{5.10}$$

$$\begin{aligned}
\mathrm{d}\Phi_-(i) \quad = \quad & k_1k_2\mathrm{sech}^2(k_2i + k_3)(1 - 2k_4\tanh(k_2i + k_3)) \\
& + k_5k_6\mathrm{sech}^2(k_6i + k_7)(1 - 2k_8\tanh(k_6i + k_7)) \\
& + k_9k_{10}\mathrm{sech}^2(k_{10}i + k_{11})(1 - 2k_{12}\tanh(k_{10}i + k_{11})) + k_{13}
\end{aligned} \tag{5.11}$$

$$\begin{aligned}
\mathrm{d}C_{n+}(i) \quad = \quad & [k_1k_2\mathrm{sech}^2(k_2i - k_3)(1 + 2k_4\tanh(k_2i - k_3)) \\
& + k_5k_6\mathrm{sech}^2(k_6i - k_7)(1 + 2k_8\tanh(k_6i - k_7)) \\
& + k_9k_{10}\mathrm{sech}^2(k_{10}i - k_{11})(1 + 2k_{12}\tanh(k_{10}i - k_{11}))] \\
& * \left(\frac{C_{nd+} - C_{nu+}}{a_+(i_{r_{n-1}}) - a_+(i_{r_n})}\right)
\end{aligned} \tag{5.12}$$

$$\begin{aligned} \mathrm{d}C_{n-}(i) &= [k_1 k_2 \mathrm{sech}^2(k_2 i + k_3)(1 - 2k_4 \tanh(k_2 i + k_3)) \\ &\quad + k_5 k_6 \mathrm{sech}^2(k_6 i + k_7)(1 - 2k_8 \tanh(k_6 i + k_7)) \\ &\quad + k_9 k_{10} \mathrm{sech}^2(k_{10} i + k_{11})(1 - 2k_{12} \tanh(k_{10} i + k_{11}))] \\ &\quad * \left(\frac{C_{nd-} - C_{nu-}}{a_-(i_{r_{n-1}}) - a_-(i_{r_n})} \right) \end{aligned} \tag{5.13}$$

Après avoir trouvé le point d'opération $(i_{km}(\tau), \phi_{km}(\tau))$, on peut linéariser la fonction localement, en supposant que le pas d'intégration est suffisamment petit et la droite passant par le point d'opération aura pour équation :

$$\phi_{km}(\tau) = K_q i_{km}(\tau) + \phi_q \tag{5.14}$$

où K_q est la pente de la droite et ϕ_q est l'ordonné à l'origine de celle-ci. Pour trouver la pente au point de linéarisation, on pourrait utiliser la méthode par perturbation, car on ne connait pas explicitement la fonction $i(\phi)$ et encore moins sa dérivée. Cependant, on peut profiter du fait que le flux aux bornes de la branche de magnétisation ne peut varier instantanément, ainsi que le courant qui la traverse. Si le pas de temps est suffisamment petit, on peut considérer que la pente est donnée approximativement par :

$$K_q \approx \frac{\phi_{km}(\tau) - \phi_{km}(\tau - \Delta t)}{i_{km}(\tau) - i_{km}(\tau - \Delta t)} \tag{5.15}$$

Ensuite, on peut calculer l'ordonnée à l'origine en solutionnant pour ϕ_q à partir de l'équation (5.14). En combinant les équations (5.2) et (5.14), on obtient le courant de la branche de magnétisation linéarisée :

$$i_{km}(\tau) = \frac{\Delta t}{2K_q} v_{km}(\tau) + \frac{\phi_{km\mathrm{hist}}(\tau) - \phi_q}{K_q} \tag{5.16}$$

À partir de l'équation (5.16), on trouve l'équivalent Norton de la branche de magnétisation

au temps τ :

$$Y_N = \frac{\Delta t}{2K_q} \tag{5.17}$$

$$I_N = \frac{\phi_{kmhist}(\tau) - \phi_q}{K_q} \tag{5.18}$$

Enfin, cet équivalent Norton est retourné au système d'équations dans EMTP-RV pour qu'il soit solutionné de nouveau et ce processus itératif est répété jusqu'à ce que tous les éléments non-linéaires du réseau aient convergé.

5.2 Domaine fréquentiel et régime permanent

Pour ce qui est du domaine fréquentiel, puisqu'il est actuellement impossible dans EMTP-RV de solutionner les éléments non-linéaires en régime permanent, on remplace pour l'instant la branche de magnétisation par l'équivalent linéaire classique, soit une inductance en parallèle avec une résistance. Dans le plan $\phi - i$, l'inductance est la pente d'une droite passant par l'origine et puisqu'on ne peut pas représenter la saturation ou les boucles mineures, on considère que l'inductance est celle de la boucle majeure à la coercivité. Ainsi, on débute en calculant le courant coercitif du modèle en solutionnant (4.4) pour $\Phi_+ = 0$. Ensuite, on le remplace dans (5.10) pour trouver la dérivée qui représente la pente, ou l'inductance, en ce point :

$$L_{ss} = \mathrm{d}\Phi_+(I_{coer}) \tag{5.19}$$

Pour la résistance, elle est représentée par une ellipse dans le plan $\phi - i$ et l'on peut trouver sa valeur en profitant du fait que l'inductance passe par l'origine. Ainsi, pour un flux nul, la tension sera maximale et le courant sera nul dans l'inductance, mais maximal dans la résistance. Alors, la résistance correspond au rapport entre la tension crête des mesures

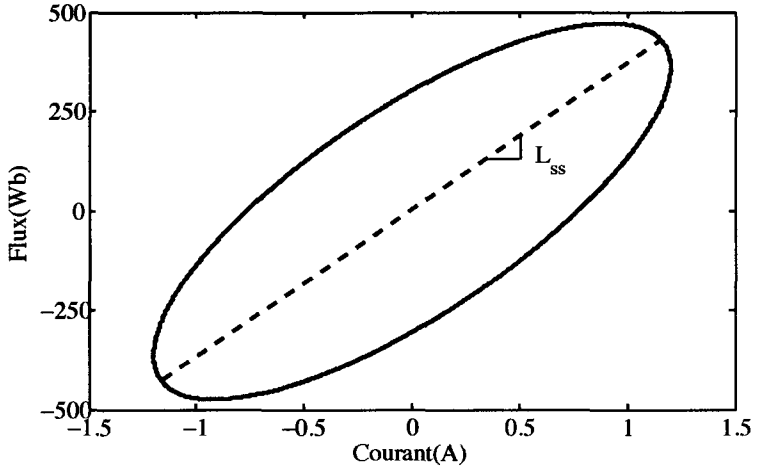


Figure 5.1 Caractéristique flux-courant pour l'admittance en régime permanent

expérimentales utilisées pour la régression non-linéaire, et du courant coercitif calculé précédemment :

$$R_{ss} = \frac{V_{max}}{I_{coer}} \tag{5.20}$$

Enfin, on peut calculer l'admittance équivalente en régime permanent de la branche de magnétisation :

$$Y_{ss} = \frac{1}{R_{ss}} + \frac{1}{j\omega L_{ss}} \tag{5.21}$$

où ω est la fréquence angulaire qui dépend des sources et à l'aide du principe de superposition, on peut trouver la solution au système d'équations pour chaque fréquence. La caractéristique équivalente de cette branche de magnétisation dans le plan $\phi - i$ est présentée à la figure 5.1. C'est cette admittance qui est insérée dans le système d'équations d'EMTP-RV, soit pour solutionner le réseau en régime permanent ou lors d'un balayage en fréquence si le paramètre d'initialisation `init` du modèle est égal à 1, donc s'il participe au régime permanent. Sinon, elle est remplacée par une faible valeur d'admittance, typiquement 10^{-12} S, pour éviter qu'une partie du réseau soit flottante.

5.3 Initialisation

L'initialisation de la branche de magnétisation est d'une importance capitale, puisque les trajectoires dépendent de l'historique de l'excitation et qu'à partir d'un point donné dans le plan $\phi - i$, il existe une infinité de trajectoires possibles. Puisque le concept de temps initial, $t = 0$, est purement théorique, il importe de se questionner sur le comportement du transformateur avant cette référence de temps, soit pour $t = 0_-$. L'hypothèse de départ pour les études transitoires est que le réseau se trouve en régime permanent pour $t < 0$ et que l'évènement perturbateur, s'il y a lieu, arrive pour $t \geq 0$. D'autre part, le transformateur dans ce réseau peut être initialement démagnétisé, donc déconnecté pour $t < 0$, ou encore connecté en régime permanent harmonique linéaire, ou il est tout simplement déconnecté du réseau pour $t < 0$ mais avec un flux rémanent. Pour choisir l'option d'initialisation appropriée pour la branche de magnétisation dans EMTP-RV, il suffit de modifier le paramètre `init` du modèle aux valeurs suivantes :

- `init` $= 1$: initialisation en régime permanent harmonique linéaire
- `init` $= 2$: initialisation avec un flux rémanent
- `init` $= 3$: initialisation à l'état démagnétisé

Les différentes méthodes d'initialisation sont explicitées dans les sous-sections qui suivent.

5.3.1 Initialisation en régime permanent harmonique linéaire

Puisqu'on initialise à partir du régime permanent, il faut passer du domaine fréquentiel au domaine du temps pour $t = 0$. Les tensions du système d'équation sont obtenues par superposition des phaseurs aux différentes fréquences et la tension aux bornes de la branche de magnétisation est de la forme suivante :

$$v(t) = V_1 \cos(\omega_1 t + \theta_1) + V_2 \cos(\omega_2 t + \theta_2) + V_3 \cos(\omega_3 t + \theta_3) + \ldots \quad (5.22)$$

D'autre part, considérant que le flux est l'intégrale de la tension donnée par l'équation (5.22), on obtient :

$$\phi(t) = \frac{V_1}{\omega_1}\sin(\omega_1 t + \theta_1) + \frac{V_2}{\omega_2}\sin(\omega_2 t + \theta_2) + \frac{V_3}{\omega_3}\sin(\omega_3 t + \theta_3) + \ldots \quad (5.23)$$

Or, les phaseurs de tensions sont donnés par :

$$\tilde{V} = V\cos(\theta) + jV\sin(\theta) \quad (5.24)$$

De plus, à $t = 0$ le flux sera :

$$\phi(0) = \frac{V_1}{\omega_1}\sin(\theta_1) + \frac{V_2}{\omega_2}\sin(\theta_2) + \frac{V_3}{\omega_3}\sin(\theta_3) + \ldots \quad (5.25)$$

Alors, en terme de phaseurs, on aura :

$$\phi(0) = \frac{\Im\{\tilde{V}_1\}}{\omega_1} + \frac{\Im\{\tilde{V}_2\}}{\omega_2} + \frac{\Im\{\tilde{V}_3\}}{\omega_3} + \ldots \quad (5.26)$$

Par conséquent, on peut initialiser l'historique de flux à cette valeur, mais la direction du flux n'est pas connue, a priori. Il faut donc évaluer le flux à un instant précédent, pour savoir s'il augmente, ou s'il diminue. On peut choisir par exemple de calculer le flux à l'instant $t = -\Delta t$, supposant que le pas d'intégration est suffisamment petit par rapport aux périodes des sources :

$$\begin{aligned}\phi(-\Delta t) = {} & \frac{V_1}{\omega_1}\sin(\omega_1(-\Delta t) + \theta_1) + \frac{V_2}{\omega_2}\sin(\omega_2(-\Delta t) + \theta_2) \\ & + \frac{V_3}{\omega_3}\sin(\omega_3(-\Delta t) + \theta_3) + \ldots \end{aligned} \quad (5.27)$$

Ainsi, la trajectoire sera ascendante pour $\phi(0) > \phi(-\Delta t)$ et descendante dans le cas contraire. Si $\phi(0) = \phi(-\Delta t)$, soit cela signifie que le pas d'intégration n'est pas suffisamment petit, soit le transformateur est déconnecté. S'il est déconnecté, il faut changer le

paramètre d'initialisation `init` à la valeur 2 ou 3.

Ensuite, à partir du flux trouvé avec l'équation (5.26), on doit trouver le courant de la branche de magnétisation à l'instant $t = 0$. Puisqu'il existe une infinité de courants possibles pour ce flux, il faut faire une hypothèse concernant la trajectoire initiale. De plus, puisque le régime permanent harmonique linéaire est une approximation, le flux initial trouvé n'est pas exact, surtout si le transformateur est en saturation. Il convient alors de supposer, pour simplifier, que le transformateur est initialement sur la trajectoire de la boucle majeure. On peut alors trouver le courant $i_{km}(0)$ correspondant au flux $\phi(0)$ à l'aide de l'équation (5.5), en considérant que $C_n(i) = 0$ pour la boucle majeure. Finalement, il est important de noter que l'option de simulation *Find steady-state solution and start from steady-state* doit être sélectionnée dans EMTP-RV afin que l'initialisation en régime permanent de la branche de magnétisation se fasse adéquatement.

5.3.2 Initialisation avec un flux rémanent

Dans certaines simulations, par exemple la mise sous tension d'un transformateur, il peut être intéressant d'initialiser la branche de magnétisation avec un flux rémanent. On considère que le transformateur est déconnecté pour $t < 0$ et l'on initialise habituellement le réseau environnant à partir du régime permanent. La principale difficulté pour cette initialisation vient du fait que l'historique de l'excitation amenant le transformateur au flux rémanent, n'est pas connue. Il faut donc émettre quelques hypothèses pour simplifier le problème. Dans un permier temps, on considère que le transformateur était en régime permanent avant son ouverture et que le cycle d'hystérésis ainsi formé était centré à l'origine. Ainsi, il faut trouver quel cycle engendrera le flux rémanent exigé. Plusieurs solutions sont possibles. On peut considérer qu'avant la fermeture le transformateur était sur la boucle majeure et il suffit de trouver le renversement du permier ordre qui passe par le flux rémanent. Cependant, cette solution est peu réaliste et représente moins bien la réalité. Une autre solution consiste à démagnétiser lentement le transformateur, à partir de la boucle

majeure, pour trouver le bon renversement. Cette option est plus réaliste, car la branche de magnétisation possèderait une historique d'excitation avant l'enclenchement du transformateur. D'autre part, on pourrait choisir de magnétiser lentement le transformateur à partir de l'origine, mais le résultat serait similaire à la démagnétisation et il faudrait procéder à l'inverse, i.e. remplir initialement tout l'historique amenant à la démagnétisation complète et effacer les extrema au fur et à mesure qu'on magnétise le transformateur, ce qui est plus compliqué que la procédure inverse. Alors, on choisira la seconde méthode pour initialiser la branche de magnétisation avec un flux rémanent.

Tout d'abord, avec les hypothèses précédentes, on considère qu'un flux rémanent positif découle d'une trajectoire descendante et qu'un flux rémanent négatif provient d'une trajectoire ascendante, car le cycle est centré à l'origine. De plus, le flux rémanent ne peut évidemment pas dépasser le flux rémanent de la boucle majeure. Puisqu'on connait le flux rémanent de la boucle mineure et celui de la boucle majeure, on peut réécrire les équations (4.4) et (4.5) :

$$C_{n+}(0) = \phi_{+}(0) - \Phi_{+}(0) \tag{5.28}$$

$$C_{n-}(0) = \phi_{-}(0) - \Phi_{-}(0) \tag{5.29}$$

Le problème consiste alors à trouver le bon historique qui amènera le paramètre C_n, pour $i = 0$ à la valeur calculée avec (5.28) ou (5.29). Ensuite, il faut démagnétiser la branche de magnétisation à partir de la boucle majeure. Puisque ce calcul se fait en boucle ouverte et indépendamment du réseau, on peut considérer que l'excitation est le courant, pour simplifier les calculs. Ainsi, on commence par trouver une règle pour démagnétiser le transformateur à partir de la boucle majeure. Cependant, puisque le courant du point de renversement de la boucle majeure est théoriquement infini, il faut trouver une règle qui ramène rapidement le courant des prochains renversements à un intervale plus intéressant

et qui diminue graduellement la différence entre chaque renversement pour avoir plus de précision. Par conséquent, la règle suivante a été choisie pour définir les points de renversement de courant :

$$i_{r_{n+2}} = \begin{cases} \sqrt{i_{r_n}} & \text{si } i_{r_n} > 2 \\ \frac{i_{r_n}}{2} & \text{sinon} \end{cases} \tag{5.30}$$

$$i_{r_{n+1}} = -i_{r_n} \tag{5.31}$$

Par la suite, il faut calculer les paramètres $C_n(0)$ pour chaque renversement à l'aide des équations (4.8) et (4.9) pour $i = 0$. Au fur et à mesure que le transformateur est démagnétisé, on ajoute les extrema à l'historique. Lorsque le critère de convergence (5.28) ou (5.29) est à l'intérieur de la tolérance, on a correctement initialisé l'historique de la branche de magnétisation. Par contre, il est évidemment presque impossible de converger directement sur le bon point de flux rémanent du premier coup, à moins d'avoir une tolérance assez élevée. Ainsi, lorsque la trajectoire du dernier renversement dépasse le flux rémanent, il faut recalculer la valeur du dernier minimum et maximum. Pour se faire, on utilise l'interpolation linéaire suivante :

$$i_{r_n} = i_{r_{n-2}} - (i_{r_{n-2}} - i_{r_n}) \left(\frac{\Phi_+(0) + C_{(n-2)+}(0) - \phi_+(0)}{C_{(n-2)+}(0) - C_{n+}(0)} \right) \tag{5.32}$$

lorsque le flux rémanent est négatif et :

$$i_{r_n} = i_{r_{n-2}} - (i_{r_{n-2}} - i_{r_n}) \left(\frac{\Phi_-(0) + C_{(n-2)-}(0) - \phi_-(0)}{C_{(n-2)-}(0) - C_{n-}(0)} \right) \tag{5.33}$$

lorsque le flux rémanent est positif. Il ne s'agit pas ici de formules récursives : le courant i_{r_n} dans le membre de droite représente le renversement qui a engendré le dépassement du flux rémanent à l'itération précédente et $C_n(0)$ est son paramètre. Ensuite, on répète le processus avec les équations (5.30) et (5.31), jusqu'à ce que l'on converge aux alentours du flux rémanent.

Finalement, il est important de mentionner que puisque le modèle A(x) est un modèle sta-

tique, la validité de celui-ci pour représenter la mise sous tension d'un transformateur avec un courant d'appel élevé n'est pas assurée, car le contenu fréquentiel du courant d'appel est riche en harmoniques.

5.3.3 Initialisation à l'état démagnétisé

Tel que mentionné au chapitre 4, lorsque le transformateur est à l'état démagnétisé, la trajectoire résultante est différente d'une trajectoire qui résulterait d'un renversement passant par l'origine dans le plan $\phi - i$. Cette trajectoire est appelée courbe de première magnétisation et on peut l'apercevoir à la figure 4.5. Puisque la trajectoire vise toujours le dernier extremum pour fermer la boucle, il faut alors initialiser l'historique avec les points constituant cette courbe, définie par l'équation (4.12) et initialiser le flux à zéro. D'autre part, pour sauver du temps et éviter de calculer ces points à chaque simulation, on inclura ce calcul à l'intérieur du régresseur qui sauvegardera les points dans l'attribut `ModelData` de la branche de magnétisation dans l'interface d'EMTP-RV. Enfin, il est possible, dépendamment du réseau, que le transformateur ne soit pas sous tension à $t = 0$ et que $v_{km}(0) = 0$. Il faut alors remplacer la branche de magnétisation par une faible valeur d'admittance, typiquement 10^{-12} S, jusqu'à ce qu'une tension apparaisse à ses bornes afin de déterminer le sens de la trajectoire dans le plan $\phi - i$.

5.4 Gestion de la pile des extrema

Les trajectoires des boucles mineures dépendent de l'historique de l'excitation et plus précisément des différents extrema du passé. Selon l'axiome d'effacement, lorsque l'excitation dépasse la valeur du dernier extremum, l'effet du dernier minimum et maximum est annulé. D'autre part, puisque les boucles mineures doivent être fermées, il faut toujours que la trajectoire vise l'avant-dernier extremum. On constate ainsi l'importance de créer une pile pour mémoriser l'historique de l'excitation. De plus, il faut toujours s'assurer que la pile n'est pas vide pour ne pas engendrer des problèmes d'accès de mémoire. Pour se faire,

on initialise le fond de la pile avec un courant très élevé et physiquement impossible, par exemple 10^{22} A, avec le flux correspondant. Enfin, deux conditions peuvent permettre de modifier le contenu de la pile et elles sont explicitées aux sous-sections suivantes.

5.4.1 Retournement

Un retournement ou renversement arrive lorsque l'excitation change de sens, formant ainsi un nouvel extremum. Cependant, puisque le processus de résolution est itératif, il faut attendre que la solution ait convergé vers cette valeur avant d'ajouter l'extremum sur le dessus de la pile. Ensuite, on doit itérer de nouveau pour trouver le nouveau point d'opération avec le réseau correspondant à cette nouvelle trajectoire. Toutefois, une exception possible mérite une attention particulière : les points de selle. Par la présence de bruit numérique dans la solution du système d'équations du réseau, un renversement pourrait être détecté lors d'un point de selle, alors qu'il ne devrait pas y en avoir normalement, ce qui engendrerait une mauvaise trajectoire. De plus, le bruit numérique pourrait causer beaucoup de renversements et ainsi, des itérations inutiles. Pour résoudre ce problème, on considère qu'il y a un renversement seulement si l'excitation a changé de direction et qu'il y a une différence de ε avec l'extremum, typiquement de 10^{-8}, en valeur absolue.

5.4.2 Dépassement

Le dépassement survient lorsque l'excitation dépasse la valeur de l'avant-dernier extremum, i.e. du dernier maximum, si la trajectoire est ascendante et du dernier minimum, si la trajectoire est descendante. D'après l'axiome d'effacement, il faut alors effacer les deux derniers extrema locaux, situés sur le dessus de la pile. D'autre part, car le processus de résolution est itératif, il faut s'assurer d'effacer les extrema intermédiaires seulement lorsque la solution a convergé vers la nouvelle valeur de dépassement. Après les avoir effacés, il faut recalculer les trajectoires et recommencer les itérations avec le système d'équations du réseau pour trouver la véritable solution. De plus, pour éviter d'avoir à calculer des dépas-

sements si le signal est en régime permanent, par exemple s'il y a du bruit numérique dans la solution, on considère qu'il y a dépassement seulement si l'excitation surpasse l'avant-dernier extremum d'une valeur ε.

5.5 Méthode de modification d'estimation de flux

Lors du processus itératif entre le réseau et les éléments non-linéaires, il est possible, sous certaines conditions, que le système d'équations converge difficilement vers la solution. Par exemple, dans la région de la saturation, un faible incrément du flux entraîne une grande différence dans le courant et par conséquent, dans l'équivalent Norton de l'élément. Cette situation est illustrée à la figure 5.2. Le réseau, de nouveau solutionné, retourne une valeur plus faible de tension, ce qui diminue le flux et par le fait même le courant. Le processus itératif continue et peut prendre beaucoup de temps à converger, voire même diverger, dans les cas de débordements numériques. Ainsi, pour converger plus rapidement et éviter les débordements, il faut appliquer une correction sur la prédiction du flux. La méthode employée est celle de la modification d'estimation de flux, basée sur la théorie trouvée dans [23] et [39]. Il s'agit en fait de rester sur la linéarisation de la fonction $\phi_{km}(i_{km})$ de l'itération (k) pour déterminer l'estimation de flux de l'itération $(k+1)$. L'équation (5.16) devient alors :

$$i_{km_c}^{(k+1)}(\tau) \quad = \quad Y_N^{(k)} v_{km}^{(k+1)}(\tau) + I_N^{(k)} \tag{5.34}$$

À partir de cette estimation de courant, on calcule l'estimation de flux $\phi_{km_c}^{(k+1)}$ à l'aide des équations (4.6) et (4.7). Dans certaines conditions, il est possible que l'estimation accélère la solution, plutôt que la décélérer. Cette situation arrive, par exemple, lorsque la trajectoire descend de la saturation positive pour passer dans la région à plus forte perméabilité ou dans le cas contraire, où la trajectoire monte de la saturation négative pour transiter dans cette région, tel qu'illustré à la figure 5.3. Ainsi, il faut choisir l'estimation de flux $\phi_{km_c}^{(k+1)}$ seulement si elle est plus proche du flux de l'itération précédente $\phi_{km}^{(k)}$ que le flux calculé

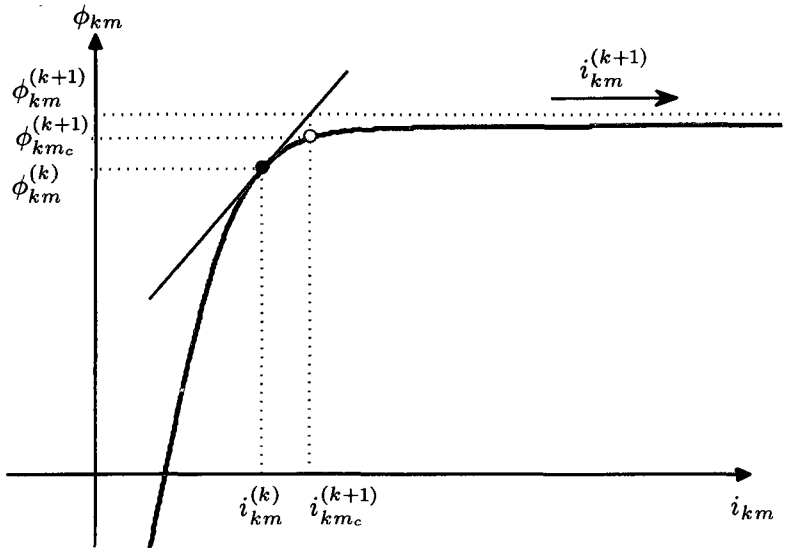


Figure 5.2 Méthode de modification d'estimation de flux

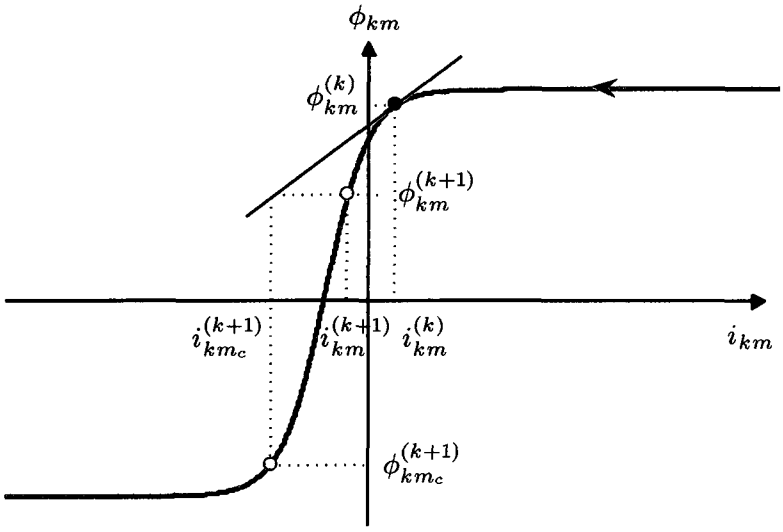


Figure 5.3 Situation où l'estimation accélère la solution

à partir de la tension du réseau $\phi_{km}^{(k+1)}$. De toute évidence, pour appliquer cette méthode, il faut d'abord avoir effectué une itération au temps τ sans appliquer le facteur de correction.

5.6 Méthode de la bissection et oscillations numériques

Il y a une autre exception qui pourrait empêcher la convergence de la branche de magnétisation : les oscillations numériques à l'intérieur de la boucle pour trouver $i(\phi)$, définie par l'équation (5.5). En général, le problème est causé par une mauvaise valeur initiale, tel qu'illustré à la figure 5.4. Pour bien initialiser $i_{km}^{(0)}(\tau)$, on présume que le courant est dans les environs du courant du pas de temps précédent, soit $i_{km}(\tau - \Delta t)$. Cependant, si le courant précédent était dans la région de la saturation et que la nouvelle solution se situait aux alentours du point d'inflexion et que, d'autre part, la perméabilité initiale était assez élevée, alors la solution pourrait osciller entre les deux niveaux de saturation. Pour remédier à ce problème, il suffit de choisir une valeur initiale différente, située au milieu des deux plateaux, lorsque les conditions d'oscillations numériques sont détectées, ou que le courant converge difficilement. Ainsi, un choix conservateur serait d'utiliser le courant coercitif comme valeur initiale de courant, car la dérivée y est la plus élevée dans cette région et que les incréments de courant à chaque itération seront plus modestes. Bien sûr, si la pente de l'asymptote dans la saturation est nulle ou extrêmement petite, le courant divergera, plutôt que d'osciller. C'est pourquoi il faut changer la valeur initiale dès qu'il y a un débordement numérique.

5.7 Méthode d'itération panique

Dans la majorité des cas, quand le pas d'intégration est suffisamment petit et qu'il n'y a pas de discontinuité, la branche de magnétisation converge très rapidement, en deux ou trois itérations seulement. Cependant, advenant une discontinuité du réseau, par exemple une manoeuvre de disjoncteur, il est alors plus difficile pour les éléments non-linéaires de converger. Pour améliorer la convergence dans ces conditions et éviter que le maximum d'itérations soit atteint, on augmentera la tolérance de convergence d'un certain facteur `ktol` après un certain nombre d'itérations réglé par le paramètre `iter_panic`, typiquement de 15.

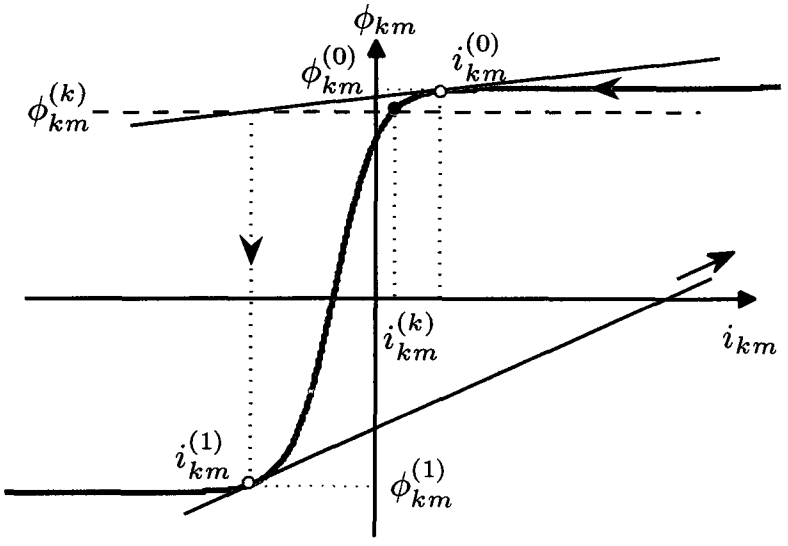


Figure 5.4 Situation d'oscillations de la méthode Newton

5.8 Régresseur non-linéaire

Pour obtenir les paramètres du modèle A(x) à partir de données expérimentales, un régresseur non-linéaire est nécessaire. D'autre part, les données de la boucle majeure ne sont pas toujours disponibles, c'est pourquoi un mode simplifié a été implémenté qui utilise des données facilement disponibles, soit `Phi_sat` le flux de saturation, `dPhi_dI_0` la pente au point coercitif, `Coer` le courant coercitif et `Lsat` l'inductance à la saturation. Cependant, puisqu'il n'existe pas de mesures pour l'élargissement au niveau du point d'inflexion de la courbe, certaines modifications sont nécessaires aux équations (4.4) et (4.5) : premièrement, les sécantes hyperboliques doivent être nulles et deuxièmement, pour simplifer le modèle, seule la première tangente hyperbolique est utilisée. Ainsi, on obtient `K1=Phi_sat`, `K2=dPhi_dI_0`, `K3=Coer`, `K13=Lsat` et tous les autres paramètres sont nuls. Le code source du régresseur est présenté à l'annexe II. Ce programme a été implémenté dans MATLAB [40] et il fournit le paramètre `ModelData` du modèle A(x) à copier dans les paramètres de l'élément dans EMTP-RV. Les paramètres à fournir au régresseur sont : `simplified_mode`, indiquant l'utilisation du mode simplifié (`true`) ou avancé (`false`), `init`, correspondant au mode d'initialisation de la branche de magnéti-

sation, `fluxinit` qui correspond au flux initial pour l'initialisation avec un flux manuel, `nomvolt` correspond à la tension nominale et `fitting.mat` est la matrice contenant les courants et les flux de la trajectoire ascendante de la boucle majeure pour le mode avancé. La matrice doit être nommée `fitting`. La première colonne contient les courants en ordre croissant et la deuxième contient les flux en ordre croissant.

La méthode de régression non-linéaire utilisée dans le mode avancé est la méthode de région de confiance [41]. Pour améliorer la régression, le régresseur effectue une deuxième régression non-linéaire en excluant les données proches du courant coercitif, car celles-ci augmentent le résidu et auront pour effet, en général, d'obtenir une régression très proche des données avant le point d'inflexion et à la saturation, mais avec un dépassement important au point d'inflexion. En comparant le coefficient de détermination ajusté des deux régressions, on prend la régression ayant le coefficient le plus proche de 1, correspondant au meilleur des cas. De surcroît, le nombre de points pour la courbe de magnétisation normale peut-être ajustée en modifiant `x2` et les paramètres `K14` et `K15` de l'équation (4.12) peuvent être modifiés manuellement.

5.8.1 Procédure de régression

Voici la procédure détaillée pour obtenir les paramètres du modèle A(x) en mode simplifié :

1. Assigner la variable `simplified_mode` du script à la valeur `true`.
2. Choisir l'option d'initialisation `init` : une valeur de 1 signifie l'initialisation en régime permanent, une valeur de 2 signifie l'initialisation avec un flux manuel et requiert la valeur de flux initial dans la variable `fluxinit` et une valeur de 3 signifie une initialisation à l'état démagnétisé.
3. Inscrire la tension nominale dans la variable `nomvolt`.
4. Choisir le nombre maximal d'éléments de la pile d'extrema et le placer dans la variable `n_stack`. La valeur par défaut de 1000 points devrait être suffisante.

5. Entrer les valeurs du modèle simplifié, soient le courant coercitif `Coer`, le flux de saturation `Phi_sat`, l'inductance au point coercitif `dPhi_dI_0` et l'inductance du milieu ambiant `Lsat`.
6. Il est aussi possible de modifier manuellement `K14` et `K15` pour obtenir la courbe de magnétisation normale désirée. D'autre part, les points de courant de la courbe de première magnétisation sont donnés par une fonction exponentielle dans `x2`, afin d'avoir plus de points proches de l'origine et moins de points dans la saturation. Cependant, la définition des points peut être modifiée manuellement pour obtenir la courbe désirée.
7. Sauvegarder le fichier et l'exécuter dans MATLAB.
8. Copier la valeur de `ModelData` obtenue.
9. Coller cette valeur dans l'attribut `ModelData` de la branche de magnétisation dans EMTP-RV. Les attributs sont accessibles par le menu contextuel de l'élément.

Pour le mode avancé, voici la procédure à suivre :

1. Assigner la variable `simplified_mode` du script à la valeur `false`.
2. Choisir l'option d'initialisation `init` : une valeur de 1 signifie l'initialisation en régime permanent, une valeur de 2 signifie l'initialisation avec un flux manuel et requiert la valeur de flux initial dans la variable `fluxinit` et une valeur de 3 signifie une initialisation à l'état démagnétisé.
3. Inscrire la tension nominale dans la variable `nomvolt`.
4. Choisir le nombre maximal d'éléments de la pile d'extrema et le placer dans la variable `n_stack`. La valeur par défaut de 1000 points devrait être suffisante.
5. Préparer une matrice contenant les données expérimentales de la trajectoire ascendante de la boucle majeure. Le fichier doit être nommé `fitting.mat` et la matrice doit porter le nom `fitting`. Le fichier doit se situer dans le même répertoire que le régresseur. La première colonne doit contenir les courants et la deuxième, les flux.

La caractéristique doit être monotone croissante et l'aire sous la courbe de la boucle majeure dans la saturation doit tendre vers zéro.

6. Au besoin, les paramètres des régressions non-linéaires peuvent être modifiés manuellement. Ils se trouvent dans la variable `s`. Les valeurs fournies par défaut produisent généralement de bonnes régressions.
7. Il est aussi possible de modifier manuellement `K14` et `K15` pour obtenir la courbe de magnétisation normale désirée. D'autre part, les points de courant de la courbe de première magnétisation sont donnés par une fonction exponentielle dans `x2`, afin d'avoir plus de points proches de l'origine et moins de points dans la saturation. Cependant, la définition des points peut être modifiée manuellement pour obtenir la courbe désirée.
8. Sauvegarder le fichier et l'exécuter dans MATLAB.
9. Copier la valeur de `ModelData` obtenue.
10. Coller cette valeur dans l'attribut `ModelData` de la branche de magnétisation dans EMTP-RV. Les attributs sont accessibles par le menu contextuel de l'élément.

CHAPITRE 6

SCÉNARIOS DE SIMULATION

Dans le but d'étudier la validité du modèle élaboré, ainsi que de tester ses limites, plusieurs scénarios de simulation ont été construits dans EMTP-RV. Par la suite, les résultats sont comparés avec les observations expérimentales, ainsi qu'avec les résultats de d'autres modèles existants.

6.1 Scénario 1

Le premier circuit, présenté à la figure 6.1, sert à vérifier le comportement de la branche de magnétisation du transformateur à vide en régime permanent. Les résultats sont comparés au modèle d'inductance hystérétique d'EMTP-RV [28], ainsi qu'à des mesures expérimentales. Le transformateur étudié est un auto-transformateur d'une puissance nominale de 370 MVA et il est présenté dans [34]. Les mesures utilisées pour la régression non-linéaire sont celles de la boucle majeure à 1,4 pu, pour être le plus près possible de la boucle majeure théorique. La source a un courant de court-circuit de 24 kA et pour l'étude, elle varie de 1,0 pu à 1,4 pu, donc de 126 $kV_{(rms)}$ à 177 $kV_{(rms)}$. Puisque l'essai se fait en conditions statiques, la résistance de magnétisation est incluse dans la branche de magnétisation et une résistance série effective a été ajoutée pour tenir compte des pertes quand le transformateur est saturé. La résistance série a été ajustée par tâtonnement pour reproduire la surface sous la courbe à 1,4 pu présentée à la figure 6.2, ce qui correspond environ à 950 mΩ.

6.1.1 Résultats

Les résultats des simulations sont présentés aux figures 6.3 à 6.6. Les courbes bleues représentent les mesures expérimentales et les courbes rouges sont les résultats des simulations.

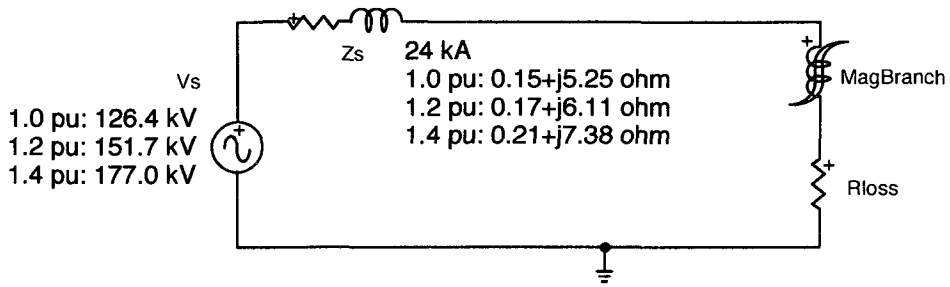


Figure 6.1 Circuit pour tester le régime permanent

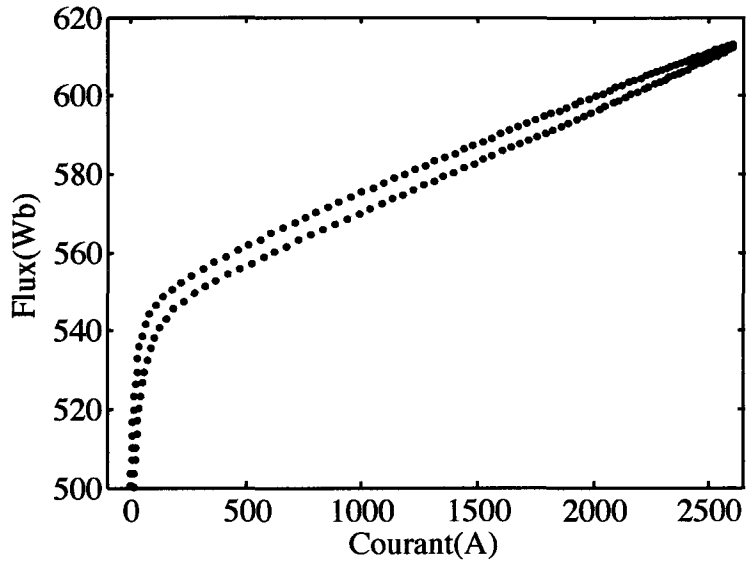
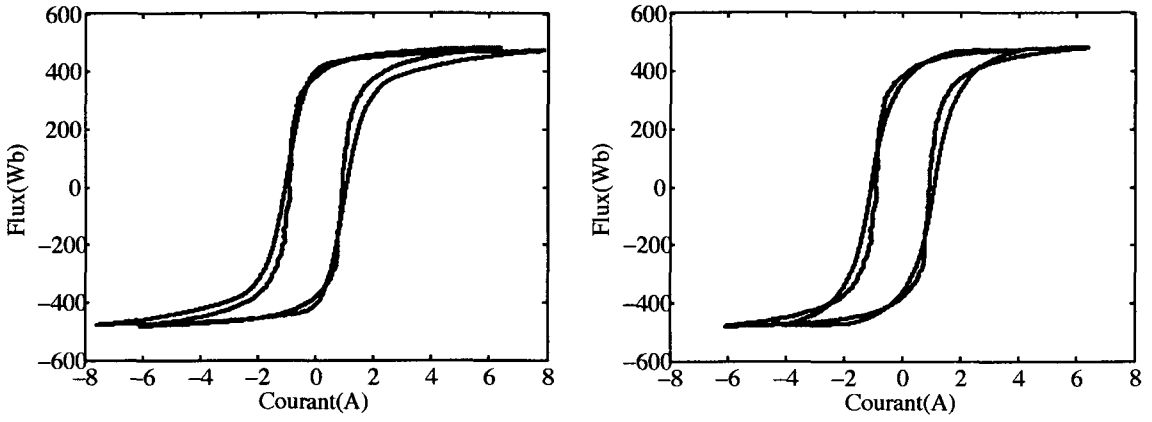


Figure 6.2 Mesures expérimentales à 1,4 pu illustrant les pertes à la saturation

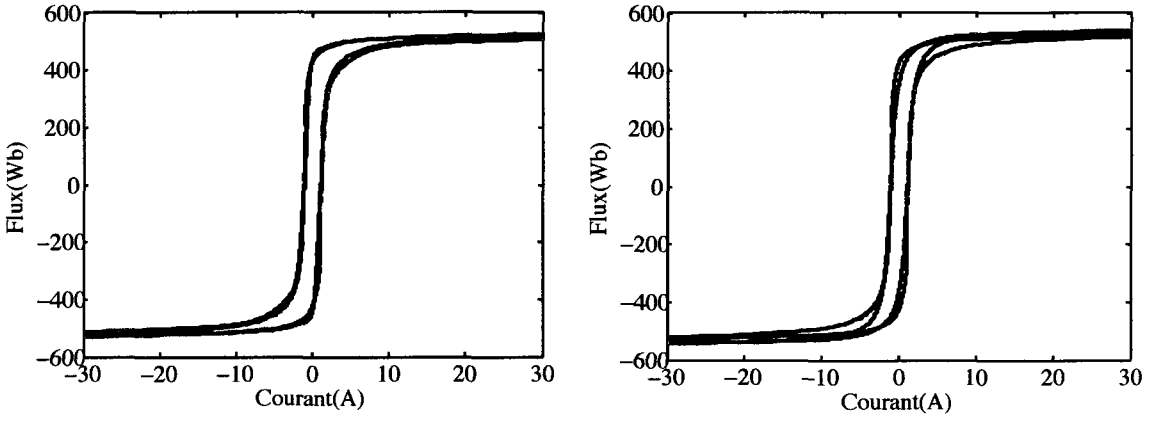
6.1.2 Discussion

Dans un premier temps, on peut constater la nécessité de la résistance en série de 950 mΩ pour simuler la surface sous la courbe dans la région de la saturation, telle que présentée à la figure 6.6. Sans elle, les branches ascendante et descendante de la trajectoire seraient pratiquement confondues et dictées par l'inductance en saturation L_{sat}. Puisque les paramètres des deux modèles ont été ajustés lors de la régression non-linéaire pour reproduire le



(a) Modèle A(x) (b) Inductance hystérétique d'EMTP-RV

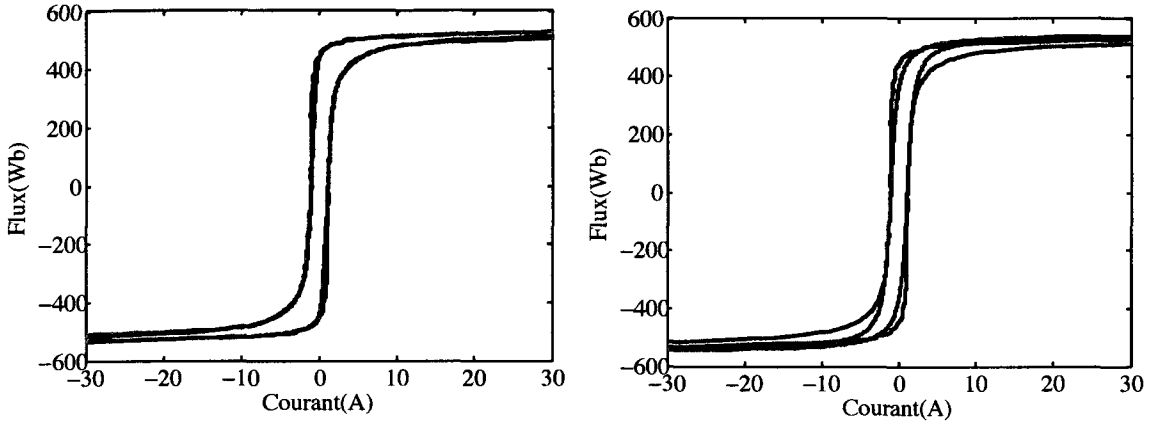
Figure 6.3 Résultats pour le régime permanent à 1,0 pu



(a) Modèle A(x) (b) Inductance hystérétique d'EMTP-RV

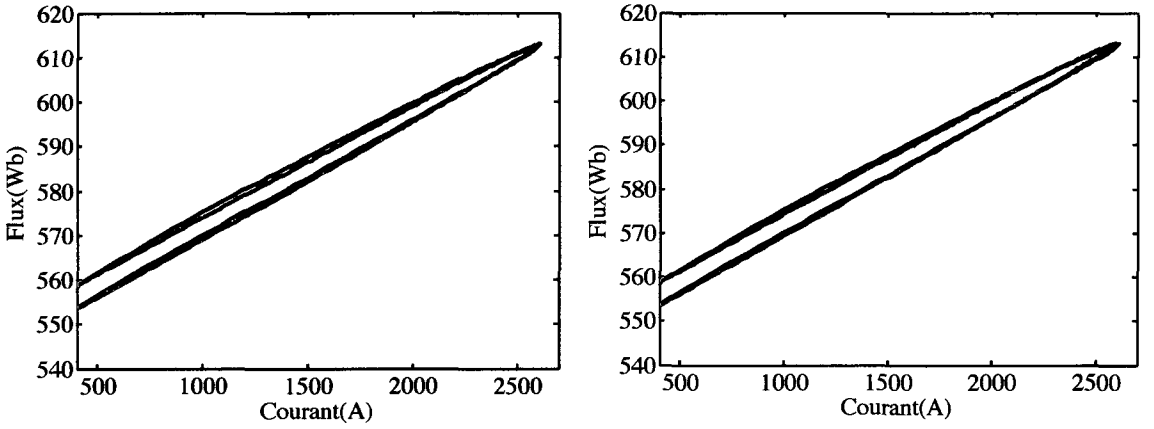
Figure 6.4 Résultats pour le régime permanent à 1,2 pu

bon niveau de saturation et la bonne pente, les résultats obtenus en saturation sont, à toutes fins pratiques, identiques. Deuxièmement, si l'on regarde les résultats du régime permanent aux figures 6.3 à 6.5, on constate la force du modèle A(x) par rapport à un modèle utilisant une fonction antisymétrique : plus la courbe s'évase en saturant le transformateur, plus la différence de surface entre le modèle antisymétrique et les mesures expérimentales est importante au point d'inflexion. Évidemment, lorsqu'il est question d'aire sous la courbe d'hystérésis, il s'agit en fait de pertes et si celles-ci sont modélisées incorrectement,



(a) Modèle A(x) (b) Inductance hystérétique d'EMTP-RV

Figure 6.5 Résultats pour le régime permanent à 1,4 pu



(a) Modèle A(x) (b) Inductance hystérétique d'EMTP-RV

Figure 6.6 Résultats pour le régime permanent à 1,4 pu illustrant les pertes à la saturation

l'amortissement des transitoires électromagnétiques sera affecté [24]. Il est donc important de choisir un modèle phénoménologique adéquat pour représenter le phénomène étudié.

6.2 Scénario 2

Pour le deuxième scénario, on s'intéresse à vérifier le comportement du transformateur lors de son réenclenchement et plus précisément son courant d'appel. Le circuit est présenté à

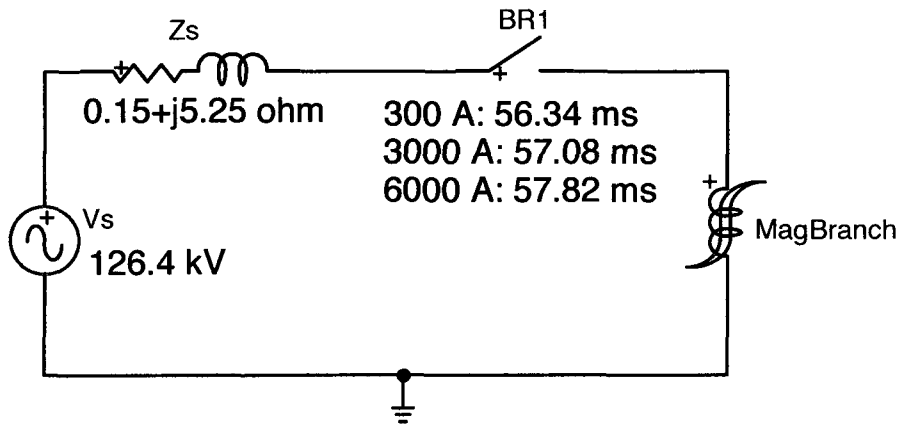
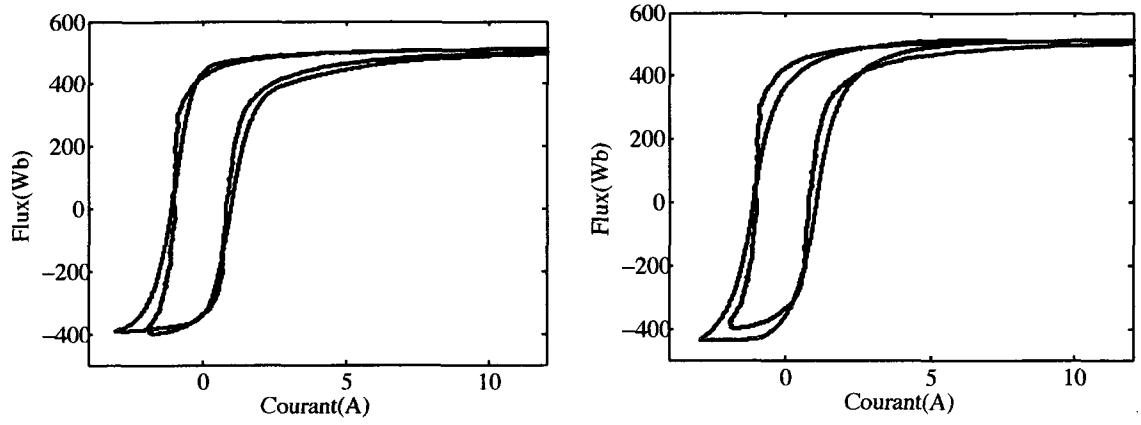


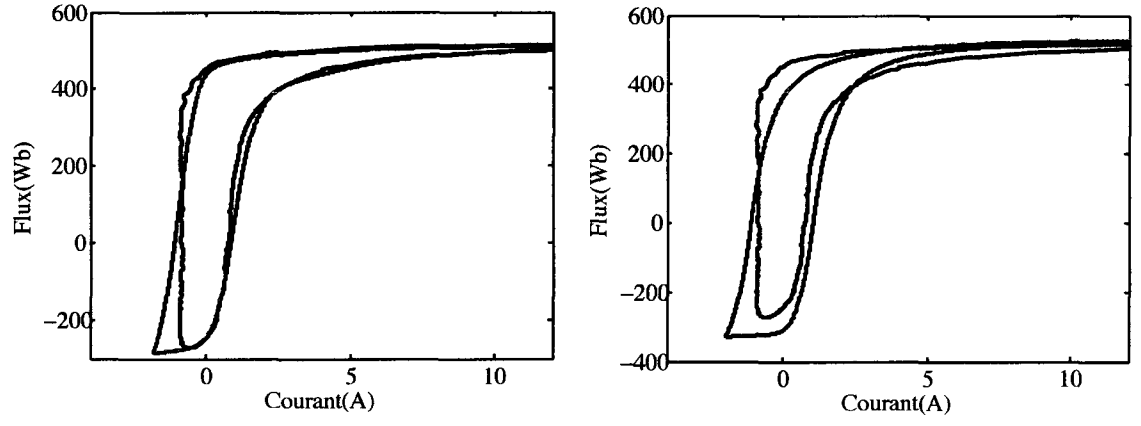
Figure 6.7 Circuit pour tester le courant d'appel du transformateur

la figure 6.7. La source utilisée est la même que le circuit précédent à 1,0 pu, mais on enlève la résistance série R_{loss}, car le courant d'appel élevé va modifier significativement la tension de la branche de magnétisation et par le fait même, le flux. Dans un premier temps, on met sous tension le transformateur et l'on ouvre le disjoncteur BR1 lors d'un passage par zéro du courant pour obtenir le flux rémanent. Pour minimiser la réponse forcée lors du réenclenchement, il faut que le flux en amont du disjoncteur BR1 soit égal au flux rémanent du transformateur, donc que le flux calculé aux bornes du disjoncteur soit nul. À partir de ce point dans le temps, on peut varier l'angle de fermeture pour changer l'amplitude du courant d'appel. Les résultats du modèle A(x) sont comparés au modèle d'inductance hystérétique d'EMTP-RV, ainsi qu'aux mesures expérimentales des courants d'appel de [34]. Trois cas de courants d'appel sont simulés, soient à 300 A, 3000 A et 6000 A, respectivement. La manoeuvre d'ouverture du disjoncteur se fait à $t = 21$ ms, afin d'avoir le flux rémanent positif correspondant à l'excitation nominale. Finalement, la manoeuvre de fermeture de BR1 se fait à $t = 56.34$ ms, pour obtenir un courant d'appel de 300 A, à $t = 57.08$ ms, pour avoir un courant d'appel de 3000 A et à $t = 57.82$ ms, pour un courant d'appel de 6000 A.



(a) Modèle A(x) (b) Inductance hystérétique d'EMTP-RV

Figure 6.8 Courant d'appel de 300A



(a) Modèle A(x) (b) Inductance hystérétique d'EMTP-RV

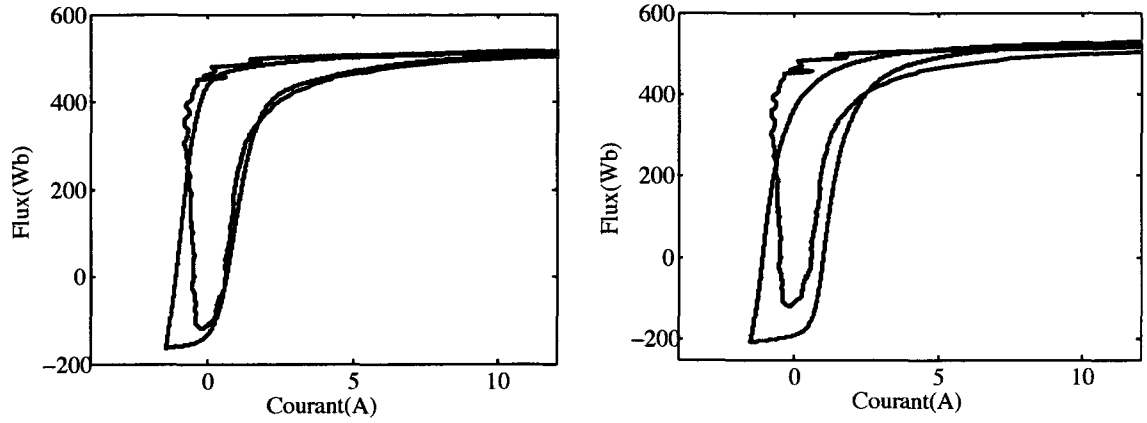
Figure 6.9 Courant d'appel de 3000A

6.2.1 Résultats

Les résultats des simulations sont présentés aux figures 6.8 à 6.10.

6.2.2 Discussion

On peut constater d'emblée que les résultats des simulations sont plus près des observations expérimentales pour les faibles courants d'appels. On pourrait attribuer la différence



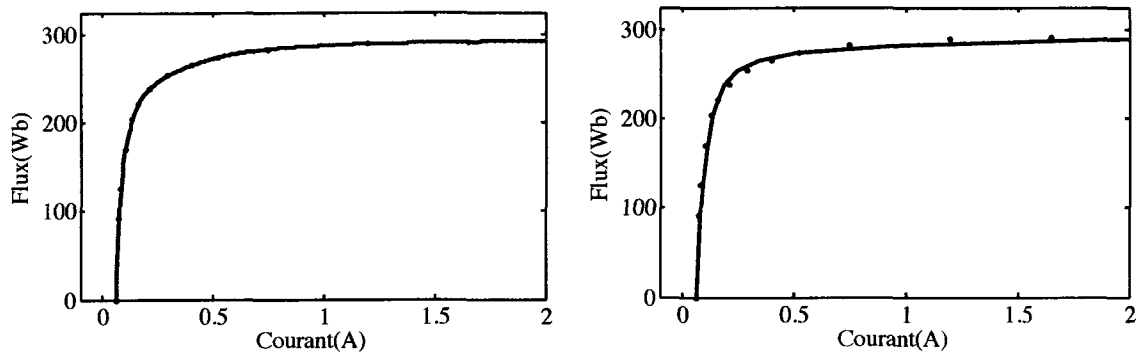
(a) Modèle A(x) (b) Inductance hystérétique d'EMTP-RV

Figure 6.10 Courant d'appel de 6000A

au contenu fréquentiel plus important lors de courants d'appel élevés, comme en témoigne la figure 6.10. En effet, les courants de Foucault et l'effet pelliculaire peuvent modifier substentiellement les pertes lorsque la fréquence augmente et l'effet des capacités parasites devient moins négligeable. De surcroît, on remarque que le modèle d'inductance hystérétique d'EMTP-RV reproduit avec moins d'exactitude l'aire sous la courbe lorsque le courant d'appel augmente. Enfin, pour mieux représenter le comportement du transformateur soumis à de forts courants d'appel, il faudrait utiliser un modèle dynamique et inclure la capacité shunt des enroulements du transformateur de [34].

6.3 Scénario 3

Le troisième scénario représente le cas de ferrorésonance quasi-périodique [42][43] présenté dans [44]. Le réseau est illustré à la figure 6.15. Il est causé par l'ouverture suite à un défaut d'un terne d'une longue ligne biterne à haute tension avec couplage électrostatique, connectée à un transformateur sans charge. Le circuit fait partie des exemples dans EMTP-RV et les inductances hystérétiques sont remplacées par des branches du modèle A(x). La charge de 6.8 MW et 1.2 Mvar au poste Silver est délestée à $t = 50$ ms et le terne de la ligne



(a) Modèle A(x) (b) Inductance hystérétique d'EMTP-RV

Figure 6.11 Régressions non-linéaires pour le cas de ferrorésonance

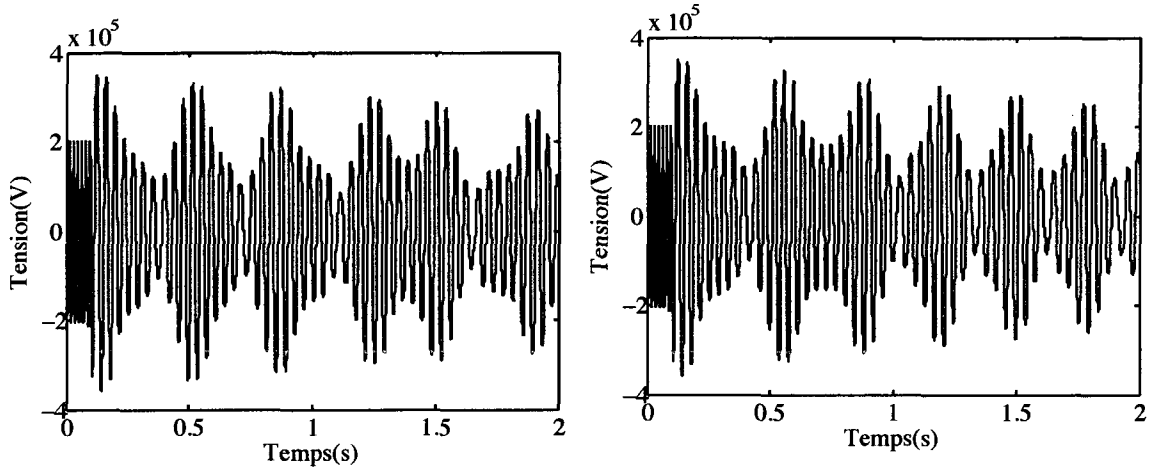
alimentant cette sous-station est ouvert aux deux extrémités à $t = 80$ ms. De plus, le défaut à la barre ROSAS qui force l'ouverture de la ligne est éliminé à $t = 100$ ms. Les données utilisées pour la régression non-linéaire du modèle sont les mêmes que celles utilisées pour l'inductance hystérétique d'EMTP-RV, sauf que la partie inférieure de la trajectoire ascendante a été ajoutée, en assumant que les données étaient antisymétriques, faute d'avoir les mesures expérimentales. La régression obtenue est présentée à la figure 6.11.

6.3.1 Résultats

Les résultats des simulations sont présentés aux figures 6.12 à 6.14.

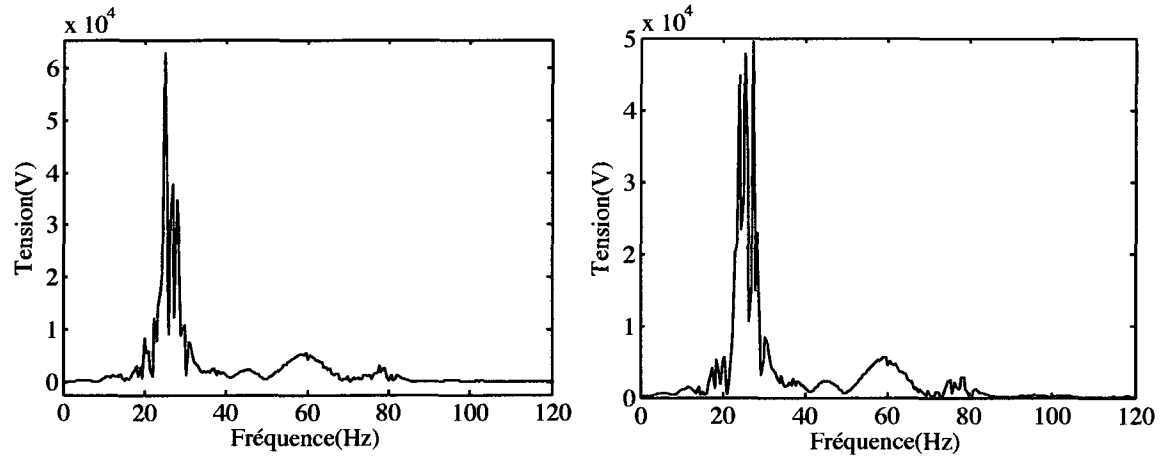
6.3.2 Discussion

En premier lieu, on peut constater, d'après la figure 6.11, l'avantage d'utiliser un modèle possédant beaucoup de degrés de liberté pour avoir de meilleures régressions non-linéaires et ce, sur un large éventail de courbes expérimentales différentes. Dans cet exemple, la différence est minime pour l'inductance hystérétique d'EMTP-RV, mais tout de même plus importante que dans le cas du modèle A(x). Deuxièmement, puisque les boucles majeures des deux modèles sont pratiquement identiques, on pourrait s'attendre à obtenir des résul-



(a) Modèle A(x) (b) Inductance hystérétique d'EMTP-RV

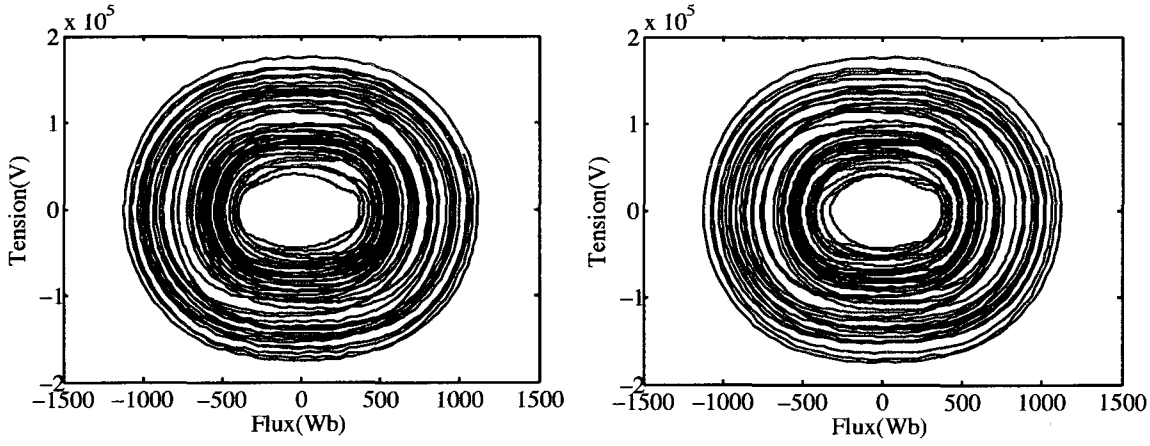
Figure 6.12 Tension de la phase A à la barre SILVB



(a) Modèle A(x) (b) Inductance hystérétique d'EMTP-RV

Figure 6.13 Réponse fréquentielle de la tension de la phase A à la barre SILVB

tats sensiblement équivalents. C'est ce que l'on constate aux figures 6.12 à 6.14. Un régime ferrorésonant quasi-périodique s'amorce dans les deux cas et la fréquence sous-synchrone observée est similaire, de l'ordre de 25 Hz. Les surtensions transitoires observées sont de l'ordre de 1,7 pu et les trajectoires quasi-périodiques observées dans le plan tension-flux sont, à toutes fins utiles, identiques.



(a) Modèle A(x) (b) Inductance hystérétique d'EMTP-RV

Figure 6.14 Plan tension-flux du transformateur

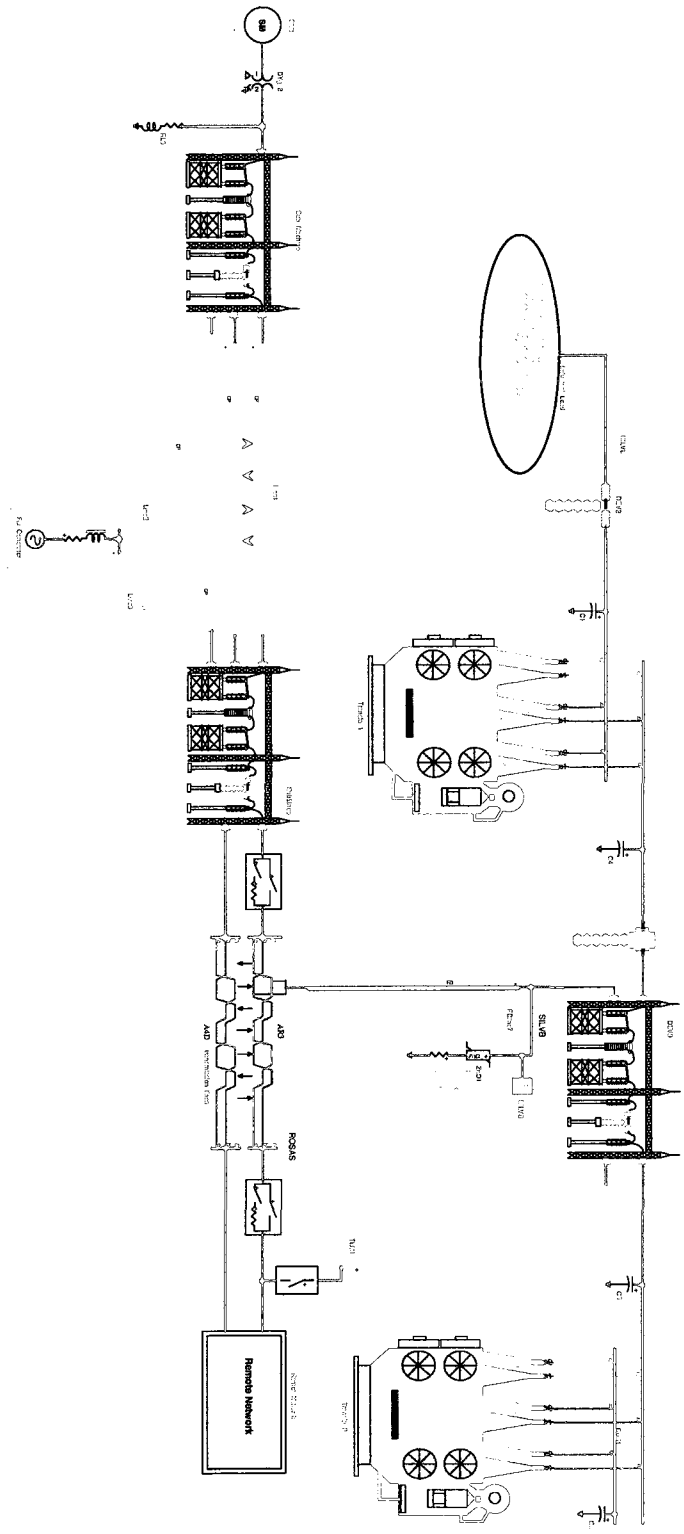


Figure 6.15 Schéma du réseau ferrorésonant

CONCLUSION

Dans ce mémoire, un nouveau modèle phénoménologique de la branche de magnétisation d'un transformateur a été présenté, permettant de reproduire avec une plus grande fidélité la surface sous la courbe, représentant les pertes. De surcroît, il a été démontré que les modèles phénoménologiques employant des fonctions antisymétriques ne peuvent fournir des régressions non-linéaires adéquates lorsque la boucle majeure présente une silhouette typique en forme de col d'oie. Une nouvelle approche a été abordée pour obtenir des résultats plus précis, tout en restant réaliste en terme de temps de calculs. D'autre part, le formalisme théorique entourant le modèle A(x) a été développé et son implémentation numérique dans un contexte de simulation des réseaux électriques a été dévoilé. Par la suite, puisqu'il s'avérait nécessaire de vérifier et de comparer le modèle, plusieurs scénarios de simulation ont été élaborés. Le nouveau modèle a clairement su surclasser les modèles antisymétriques lors des tests en régime permanent et il a aussi été possible d'observer un régime ferrorésonant quasi-périodique en utilisant la branche de magnétisation. De plus, le modèle A(x) ne requiert aucune donnée sur les boucles mineures, ce qui le rend plus facilement paramétrisable, comparé à la plupart des modèles. Cependant, il est important de mentionner que le nouveau modèle a ses limites. C'est un modèle statique, ce qui implique qu'il est indépendant de la variation du taux de l'excitation et que les pertes, ainsi que l'inductance, sont constants en fréquence. Alors, le modèle représentera adéquatement l'amortissement autour de la fréquence de caractérisation, mais le comportement ne sera plus valide plus on augmente en fréquence, car les courants de Foucault deviennent beaucoup plus importants et l'effet pelliculaire cause aussi une augmentation dans la résistance ainsi qu'une diminution de l'inductance du transformateur. Somme toute, pour tenir compte de ces effets et permettre à la branche de magnétisation de reproduire convenablement la gamme complète de transitoires électromagnétiques dans un réseau électrique, il faudrait inclure une dépendance en fréquence dans le calcul de la réponse, ce qui représenterait un ajout fort intéressant au modèle A(x). De plus, il sera souhaitable d'implémenter une interface graphique pour faciliter la régression, ainsi que l'entrée des paramètres du modèle.

RÉFÉRENCES

[1] J. A. Ewing, "On the production of transient electric currents in iron and steel conductors by twisting them when magnetised or by magnetising them when twisted," in *Proc. Roy. Soc.*, vol. XXXIII, 1882, pp. 21–23.

[2] ——, "Effects of stress on the thermoelectric quality of metals, part I," in *Proc. Roy. Soc.*, vol. XXXII, 1881, pp. 399–402.

[3] J. Mahseredjian, S. Dennetière, L. Dubé, B. Khodabakhchian, and L. Gérin-Lajoie, "On a new approach for the simulation of transients in power systems," *Electric Power Systems Research*, vol. 77, no. 11, pp. 1514–1520, 2007.

[4] A. Rogalev, F. Wilhelm, N. Jaouen, J. Goulon, and J.-P. Kappler, *Magnetism : A synchrotron radiation approach.* Springer Lecture Notes in Physics, 2006, vol. 697, pp. 71–93.

[5] G. Bertotti, *Hysteresis in magnetism*, 1st ed. Academic Press, 1998.

[6] H. Barkhausen, "Zwei mit Hilfe der neuen Verstärker entdeckte Erscheinungen," *Physik. Zeitschr.*, vol. XX, pp. 401–403, 1919.

[7] L. Callegaro, E. Puppin, and M. Zani, "Barkhausen jumps and metastability," *J. Phys. D : Appl. Phys.*, vol. 36, pp. 2036–2040, 2003.

[8] I. Mayergoyz, *Mathematical models of hysteresis and their applications*, 2nd ed. Elsevier, 2003.

[9] J. Takács, *Mathematics of hysteretic phenomena*, 1st ed. Wiley VCH, 2003.

[10] E. Madelung, "Über magnetisierung durch schnellverlaufende ströme und die wirkungsweise des rutherford-marconischen magnetdetektors," *Ann. Phys.*, vol. 322, no. 10, pp. 861–890, 1905.

[11] G. Bertotti and I. Mayergoyz, *The science of hysteresis.* Elsevier, 2005.

[12] A. Visintin, *Differential models of hysteresis*, ser. Applied Mathematical Sciences. Springer, Berlin, 1994.

[13] F. Preisach, "Über die magnetische Nachwirkung," *Zeitschrift für Physik*, vol. 94, pp. 277–302, 1935.

[14] D. Everett and W. Whitton, "A general approach to hysteresis," *Trans. Faraday Soc.*, vol. 48, pp. 749–757, 1952.

[15] M. Krasnosel'skii and A. Pokrovskii, *Systems with Hysteresis*. Springer Verlag, 1989.

[16] E. Stoner and W. E.P., "A mechanism of magnetic hysteresis in heterogeneous alloys," *Phil. Trans. Roy. Soc.*, vol. 240, pp. 599–642, 1948.

[17] F. Liorzou, B. Phelps, and D. Atherton, "Macroscopic models of magnetization," *Magnetics, IEEE Transactions on*, vol. 36, no. 2, pp. 418–428, March 2000.

[18] D. Jiles and D. Atherton, "Ferromagnetic hysteresis," *IEEE Transactions on Magnetics*, vol. 19, no. 5, pp. 2183–2185, 1983.

[19] P. Langevin, "Magnétisme et théorie des électrons," *Ann. Chim. Phys.*, vol. V, pp. 70–127, 1905.

[20] K. Carpenter, "A differential equation approach to minor loops in the jiles-atherton hysteresis model," *IEEE Transactions on Magnetics*, vol. 27, no. 6, pp. 4404–4406, 1991.

[21] S. N. Talukdar and J. R. Bailey, "Hysteresis models for system studies," *IEEE Transactions on Power Apparatus and Systems*, vol. PAS-95, no. 4, pp. 1429–1434, July/August 1976.

[22] J. Frame, N. Mohan, and T.-H. Liu, "Hysteresis modeling in an electro-magnetic transients program," *IEEE Transactions on Power Apparatus and Systems*, vol. PAS-101, no. 9, pp. 3403–3412, Sept. 1982.

[23] S. Dennetière, "Modélisation du phénomène d'hystérésis sous EMTP-RV," Master's thesis, École Polytechnique de Montréal, 2003.

[24] A. Narang, E. P. Dick, and R. C. Cheung, "Transformer model for electromagnetic transient studies," Ontario Hydro Technologies, CEA Report No. 175 T 331G, December 1997.

[25] D. N. Ewart, "Digital computer simulation model of a steel-core transformer," *IEEE Transactions on Power Delivery*, vol. PWRD-1, no. 3, pp. 174–183, July 1986.

[26] E. P. Dick and W. Watson, "Transformer models for transient studies based on field measurements," *IEEE Transactions on Power Apparatus and Systems*, vol. PAS-100, no. 1, pp. 409–419, Jan. 1981.

[27] H. W. Dommel, *Electromagnetic Transients Program Reference Manual (EMTP Theory Book)*, Bonneville Power Administration, Oregon, August 1986.

[28] S. Dennetière, J. Mahseredjian, M. Martinez, M. Rioual, A. Xémard, and P. Bastard, "On the implementation of a hysteretic reactor model in EMTP," in *IPST-2003, International Conference on Power Systems Transients*, New Orleans, USA, September 2003.

[29] PSCAD/EMTDC Version 4, Manitoba HVDC Reseach Centre Inc., 2004.

[30] A. Rezaei-Zare, M. Sanaye-Pasand, H. Mohseni, S. Farhangi, and R. Iravani, "Analysis of ferroresonance modes in power transformers using preisach-type hysteretic magnetizing inductance," *Power Delivery, IEEE Transactions on*, vol. 22, no. 2, pp. 919–929, April 2007.

[31] A. Rezaei-Zare, R. Iravani, M. Sanaye-Pasand, H. Mohseni, and S. Farhangi, "An accurate current transformer model based on Preisach theory for the analysis of electromagnetic transients," *IEEE Transactions on Power Delivery*, vol. 23, no. 1, pp. 233–242, Jan. 2008.

[32] MATLAB SimPowerSystems 2.3, The MathWorks, Inc., 2002.

[33] S. Casoria, P. Brunelle, and G. Sybille, "Hysteresis modeling in the MATLAB/Power System Blockset," in *Electrimacs 2002*, École de technologie supérieure, Montréal, 2002.

[34] A. Gaudreau, P. Picher, L. Bolduc, and A. Coutu, "No-load losses in transformer under overexcitation/inrush-current conditions : Tests and a new model," *IEEE Transactions on Power Delivery*, vol. 17, no. 4, pp. 1009–1017, October 2002.

[35] M. Lambert, J. Mahseredjian, L.-A. Dessaint, and A. Gaudreau, "Implementation of a new magnetizing branch in EMTP-RV using the A(x) model," in *IPST-2009, International Conference on Power Systems Transients*, Kyoto, Japan, June 2009.

[36] P. Weiss, "L'hypothèse du champ moléculaire et la propriété ferromagnétique," *J. de Phys. Radium*, vol. VI, pp. 661–690, 1907.

[37] L. Brillouin, "Les moments de rotation et le magnétisme dans la mécanique ondulatoire," *J. de Phys. Radium*, vol. VIII, pp. 74–84, 1927.

[38] J. Mahseredjian, *DLL programming, EMTP-RV documentation*, 2007.

[39] ——, "EMTPRV : The design of version 0A," Hydro-Québec, Tech. Rep., 2000.

[40] MATLAB v7.5.0, The MathWorks, Inc., 2007.

[41] M. R. Celis, J. E. Dennis, and R. A. Tapia, "A trust region strategy for nonlinear equality constrained optimization," *SIAM Numerical Optimization*, pp. 71–82, 1985.

[42] C. Kieny and A. Sbaï, "Ferrorésonance dans les réseaux," Techniques de l'ingénieur, Tech. Rep., March 1996.

[43] P. Ferracci, "Ferroresonance," Schneider, Tech. Rep. 190, March 1998.

[44] D. A. N. Jacobson, L. Martí, and R. W. Menzies, "Modeling ferroresonance in a 230 kV transformer-terminated double-circuit transmission line," in *IPST-1999, International Conference on Power Systems Transients*, Budapest, Hungary, June 1999, pp. 451–456.

ANNEXE I

CODE SOURCE DE LA BRANCHE DE MAGNÉTISATION

```
!*******************************************************************************
!*  NewMagBranch.f90                                                           *
!*  Copyright(c) Hydro-Quebec TransEnergie                                     *
!*                                                                             *
!*  Created: 2008-05-08 09:46:27                                               *
!*  Author : Mathieu Lambert                                                   *
!*  Last change: ML 2009-01-28 17:59:23                                        *
!*  Version: 1.17                                                              *
!*                                                                             *
!*  This program is the new model of magnetizing branch for EMTP-RV            *
!*  It uses parameters K1 to K13 found from the fitting program                *
!*  The major loop is based on a summation of tanh and sech^2 functions        *
!*******************************************************************************

MODULE NewMagBranch
  USE sizelimits,        ONLY: max_len_comp_id
  USE default_precision, ONLY: krealhp, zero, onehalf
  USE simulation_data
  USE variable,          ONLY: jz, halfpi

  IMPLICIT NONE

  TYPE, PUBLIC :: data_holder                !Data structure for each magnetizing branch
    CHARACTER(LEN=max_len_comp_id) :: myname !Unique name for this device from EMTP
    INTEGER                        :: idev   !Unique device number for all DLL devices

    !Major loop parameters found from fitting
    REAL(krealhp)     :: K1
    REAL(krealhp)     :: K2
    REAL(krealhp)     :: K3
    REAL(krealhp)     :: K4
    REAL(krealhp)     :: K5
    REAL(krealhp)     :: K6
    REAL(krealhp)     :: K7
    REAL(krealhp)     :: K8
    REAL(krealhp)     :: K9
    REAL(krealhp)     :: K10
    REAL(krealhp)     :: K11
    REAL(krealhp)     :: K12
    REAL(krealhp)     :: K13

    REAL(krealhp)     :: nomvolt  = zero !Peak voltage of measurements in fitting
    REAL(krealhp)     :: remflux  = zero !Remanent flux of major loop
    INTEGER           :: init            !1 means initialize from SS
                                         !2 means manual flux
                                         !3 means no initial conditions
    REAL(krealhp)     :: fluxinit = zero !Manual flux initialization
    INTEGER           :: n_stack         !The number of rows in stack to be allocated
    INTEGER           :: iter     = 0    !Local iteration counter
    INTEGER           :: ktol     = 1    !Tolerance acceleration factor
    REAL(krealhp)     :: coer     = zero !Coercivity

    COMPLEX(krealhp)  :: Yss      = 0    !Steady-state admittance
    COMPLEX(krealhp)  :: Iss      = 0    !Steady-state current

    REAL(krealhp)     :: Iq       = zero !Norton current source
    REAL(krealhp)     :: y        = zero !Norton conductance
    REAL(krealhp)     :: fluxkm   = zero !Instantaneous flux
    REAL(krealhp)     :: fluxold  = zero !Flux at t-dt
    REAL(krealhp)     :: iold     = zero !Current at t-dt
    REAL(krealhp)     :: h        = zero !Flux history term
    REAL(krealhp)     :: h_dt     = zero !Flux at t=-dt for SS initialization
    REAL(krealhp)     :: cur      = zero !=-Phi_Q/k_Q for each iteration
    REAL(krealhp)     :: ikm      = zero !Instantaneous current
```

```
    REAL(krealhp)     :: vkm      = zero !Instantaneous voltage
    !Constants used for minor loop trajectories,
    !(1) is for FIND_CURRENT and (2) is for FIND_FLUXGUESS
    REAL(krealhp)     :: Cnu(2)   = zero
    REAL(krealhp)     :: Cnd(2)   = zero
    REAL(krealhp)     :: Cn(2)    = zero
    REAL(krealhp)     :: Aold(2)  = zero
    REAL(krealhp)     :: Anew(2)  = zero

    REAL(krealhp), DIMENSION(:,:), ALLOCATABLE :: max_stack !Maxima points stack
    REAL(krealhp), DIMENSION(:,:), ALLOCATABLE :: min_stack !Minima points stack
    INTEGER                       :: n=0     !Number of maxima
    INTEGER                       :: m=0     !Number of minima

    INTEGER,POINTER,DIMENSION(:) :: power_signal_nodes !The actual global node numbers
    LOGICAL     :: converged     = .FALSE. !Indicates which device did not converge
    LOGICAL     :: upward        = .FALSE. !Indicates if the flux is going upward
    LOGICAL     :: reversal      = .FALSE. !Indicates if there is a reversal
    LOGICAL     :: reversalguess = .FALSE. !Indicates if there is a reversal for guess
    LOGICAL     :: demagnetized  = .TRUE.  !Indicates if the branch is demagnetized
    LOGICAL     :: overtake      = .FALSE. !Indidates if there is an overtaking

    TYPE(data_holder),POINTER  :: next => NULL() !Pointer to the next magnetizing branch
  END TYPE
  TYPE(data_holder),POINTER,PUBLIC::MagBranch=>NULL()  !Pointer to the current element
  TYPE(data_holder),POINTER,PUBLIC::MagBranch_first=>NULL()!Pointer to the first element

  LOGICAL,PUBLIC :: EXISTENCE=.FALSE. !To indicate the existence of a magnetizing branch
  INTEGER,PUBLIC       :: Total_number_of_devices=0    !To keep the local count
  REAL(krealhp),PUBLIC :: epsilon=1.e-08_krealhp       !Overtaking, reversal tolerance
  REAL(krealhp),PUBLIC :: lowadmittance=1.e-12_krealhp !A small admittance value
  REAL(krealhp),PUBLIC :: SatCurrent=1.e22_krealhp     !A high current value
  INTEGER,PUBLIC       :: ktol_min=1                   !Minimum value of ktol
  INTEGER,PUBLIC     :: ktol_max=5 !ktol can take this value when iter_panic is exceeded
  INTEGER,PUBLIC     :: iter_panic=15    !Number of iterations after which the iterative
                                         !procedure should try lowering tolerance

  CONTAINS
    PURE ELEMENTAL FUNCTION DSECH(x) !Computes the hyperbolic secant
      REAL(krealhp)              :: DSECH
      REAL(krealhp), INTENT(IN)  :: x
      DSECH=2.0_krealhp/(DEXP(x)+DEXP(-x))
    END FUNCTION

    PURE ELEMENTAL FUNCTION is_NE(x, ref, myrelative_tolerance)
      LOGICAL                  :: is_NE                    !Comparison test result
      REAL(krealhp),INTENT(IN) :: x                        !Compared real number
      REAL(krealhp),INTENT(IN) :: ref                      !Comparison reference
      REAL(krealhp)            :: dref                     !Margin of granularity
      REAL(krealhp),INTENT(IN) :: myrelative_tolerance     !Given relative tolerance
      dref=get_margin_r_myreltol(ref,myrelative_tolerance) !Get margin using given tol
      is_NE = (x < ref-dref .OR. x > ref+dref)             !Test x < ref- or x > ref+
    END FUNCTION

    PURE ELEMENTAL FUNCTION get_margin_r_myreltol( x, myreltol ) RESULT(margin)
      REAL(krealhp)            :: margin             !Calculated margin
      REAL(krealhp),INTENT(IN) :: x                  !Reference real number
      REAL(krealhp),INTENT(IN) :: myreltol           !Given relative tolerance
      margin = MAX(ABS(x)*myreltol,&                 !Calc size of granularity margin
                   equality_precision)               !with floor at equality_precision
    END FUNCTION

    PURE ELEMENTAL FUNCTION A(x, pos) !The a-(i) and a+(i) functions
      REAL(krealhp)            :: A
      LOGICAL,INTENT(IN)       :: pos                 !The upward function if true
      REAL(krealhp),INTENT(IN) :: x                   !Current

      IF(pos) THEN
        A=(MagBranch%K1*((DTANH(MagBranch%K2*x-MagBranch%K3))&
              -(MagBranch%K4*(DSECH(MagBranch%K2*x-MagBranch%K3)**2))))&
              +(MagBranch%K5*((DTANH(MagBranch%K6*x-MagBranch%K7))&
              -(MagBranch%K8*(DSECH(MagBranch%K6*x-MagBranch%K7)**2))))&
              +(MagBranch%K9*((DTANH(MagBranch%K10*x-MagBranch%K11))&
              -(MagBranch%K12*(DSECH(MagBranch%K10*x-MagBranch%K11)**2))))

      ELSE
        A=(MagBranch%K1*((DTANH(MagBranch%K2*x+MagBranch%K3))&
```

```
            +(MagBranch%K4*(DSECH(MagBranch%K2*x+MagBranch%K3)**2))))&
            +(MagBranch%K5*((DTANH(MagBranch%K6*x+MagBranch%K7))&
            +(MagBranch%K8*(DSECH(MagBranch%K6*x+MagBranch%K7)**2))))&
            +(MagBranch%K9*((DTANH(MagBranch%K10*x+MagBranch%K11))&
            +(MagBranch%K12*(DSECH(MagBranch%K10*x+MagBranch%K11)**2))))

  END IF
END FUNCTION

PURE ELEMENTAL FUNCTION dA(x, pos) !The derivative of a(i) functions
  REAL(krealhp)              :: dA
  LOGICAL,INTENT(IN)         :: pos                  !The upward function if true
  REAL(krealhp),INTENT(IN)   :: x                    !Current

  IF(pos) THEN
    dA=(MagBranch%K1*MagBranch%K2*(DSECH(MagBranch%K2*x-MagBranch%K3)**2)&
          *(1+(2*MagBranch%K4*DTANH(MagBranch%K2*x-MagBranch%K3))))&
          +(MagBranch%K5*MagBranch%K6*(DSECH(MagBranch%K6*x-MagBranch%K7)**2)&
          *(1+(2*MagBranch%K8*DTANH(MagBranch%K6*x-MagBranch%K7))))&
          +(MagBranch%K9*MagBranch%K10*(DSECH(MagBranch%K10*x-MagBranch%K11)**2)&
          *(1+(2*MagBranch%K12*DTANH(MagBranch%K10*x-MagBranch%K11))))
  ELSE
    dA=(MagBranch%K1*MagBranch%K2*(DSECH(MagBranch%K2*x+MagBranch%K3)**2)&
          *(1-(2*MagBranch%K4*DTANH(MagBranch%K2*x+MagBranch%K3))))&
          +(MagBranch%K5*MagBranch%K6*(DSECH(MagBranch%K6*x+MagBranch%K7)**2)&
          *(1-(2*MagBranch%K8*DTANH(MagBranch%K6*x+MagBranch%K7))))&
          +(MagBranch%K9*MagBranch%K10*(DSECH(MagBranch%K10*x+MagBranch%K11)**2)&
          *(1-(2*MagBranch%K12*DTANH(MagBranch%K10*x+MagBranch%K11))))
  END IF
END FUNCTION

PURE ELEMENTAL FUNCTION PhiMaj(x, pos) !The major loop flux
  REAL(krealhp)              :: PhiMaj
  LOGICAL,INTENT(IN)         :: pos                  !The upward function if true
  REAL(krealhp),INTENT(IN)   :: x                    !Current

  IF(pos) THEN
    PhiMaj=(MagBranch%K1*((DTANH(MagBranch%K2*x-MagBranch%K3))&
          -(MagBranch%K4*(DSECH(MagBranch%K2*x-MagBranch%K3)**2))))&
          +(MagBranch%K5*((DTANH(MagBranch%K6*x-MagBranch%K7))&
          -(MagBranch%K8*(DSECH(MagBranch%K6*x-MagBranch%K7)**2))))&
          +(MagBranch%K9*((DTANH(MagBranch%K10*x-MagBranch%K11))&
          -(MagBranch%K12*(DSECH(MagBranch%K10*x-MagBranch%K11)**2))))&
          +(MagBranch%K13*x)
  ELSE
    PhiMaj=(MagBranch%K1*((DTANH(MagBranch%K2*x+MagBranch%K3))&
          +(MagBranch%K4*(DSECH(MagBranch%K2*x+MagBranch%K3)**2))))&
          +(MagBranch%K5*((DTANH(MagBranch%K6*x+MagBranch%K7))&
          +(MagBranch%K8*(DSECH(MagBranch%K6*x+MagBranch%K7)**2))))&
          +(MagBranch%K9*((DTANH(MagBranch%K10*x+MagBranch%K11))&
          +(MagBranch%K12*(DSECH(MagBranch%K10*x+MagBranch%K11)**2))))&
          +(MagBranch%K13*x)
  END IF
END FUNCTION

PURE ELEMENTAL FUNCTION dPhiMaj(x, pos) !Major loop derivative dPhi
  REAL(krealhp)              :: dPhiMaj
  LOGICAL,INTENT(IN)         :: pos                  !The upward function if true
  REAL(krealhp),INTENT(IN)   :: x                    !Current

  IF(pos) THEN
    dPhiMaj=(MagBranch%K1*MagBranch%K2*(DSECH(MagBranch%K2*x-MagBranch%K3)**2)&
          *(1+(2*MagBranch%K4*DTANH(MagBranch%K2*x-MagBranch%K3))))&
          +(MagBranch%K5*MagBranch%K6*(DSECH(MagBranch%K6*x-MagBranch%K7)**2)&
          *(1+(2*MagBranch%K8*DTANH(MagBranch%K6*x-MagBranch%K7))))&
          +(MagBranch%K9*MagBranch%K10*(DSECH(MagBranch%K10*x-MagBranch%K11)**2)&
          *(1+(2*MagBranch%K12*DTANH(MagBranch%K10*x-MagBranch%K11))))&
          +(MagBranch%K13)
  ELSE
    dPhiMaj=(MagBranch%K1*MagBranch%K2*(DSECH(MagBranch%K2*x+MagBranch%K3)**2)&
          *(1-(2*MagBranch%K4*DTANH(MagBranch%K2*x+MagBranch%K3))))&
          +(MagBranch%K5*MagBranch%K6*(DSECH(MagBranch%K6*x+MagBranch%K7)**2)&
          *(1-(2*MagBranch%K8*DTANH(MagBranch%K6*x+MagBranch%K7))))&
          +(MagBranch%K9*MagBranch%K10*(DSECH(MagBranch%K10*x+MagBranch%K11)**2)&
          *(1-(2*MagBranch%K12*DTANH(MagBranch%K10*x+MagBranch%K11))))&
          +(MagBranch%K13)
  END IF
```

```
    END FUNCTION
END MODULE

!Data reading and allocation of pointers for each device
SUBROUTINE DLL_INITIALIZE_NEW(myname,idev,Data_Section,DLL_NAME)
  USE NewMagBranch
  !DEC$ ATTRIBUTES DLLEXPORT:: DLL_INITIALIZE_NEW
  CHARACTER(LEN=*), INTENT(IN) :: myname    !Unique name for this device coming from EMTP
  INTEGER,          INTENT(IN) :: idev      !Unique device for all DLL type devices
  CHARACTER(LEN=*), INTENT(IN) :: Data_Section(*)  !ModelData field of the device
  CHARACTER(LEN=*), INTENT(IN) :: DLL_NAME !The name of this DLL as referenced in EMTP

  INTEGER                      :: mlines
  INTEGER                      :: ii
  REAL(krealhp)                :: flux
  REAL(krealhp)                :: current

  !Create a chained list of magnetizing branches
  IF(EXISTENCE) THEN !If one magnetizing branch already exists, allocate a new one
    ALLOCATE(MagBranch%next)
    MagBranch=>MagBranch%next
    MagBranch%next=>MagBranch_first
  ELSE !If it's the first magnetizing branch, allocate it as the first of the list
    ALLOCATE(MagBranch)
    MagBranch_first=>MagBranch
    EXISTENCE=.TRUE.
  END IF

  Total_number_of_devices=Total_number_of_devices+1 !Increment the count of mag branches
  MagBranch%myname = myname !Assign the name to the device
  MagBranch%idev   = idev   !Assign the device number

  !Reading of ModelData attributes is done here
  READ(Data_Section(1),*,ERR=999,END=999)MagBranch%K1,MagBranch%K2,MagBranch%K3,&
                                         MagBranch%K4
  READ(Data_Section(2),*,ERR=999,END=999)MagBranch%K5,MagBranch%K6,MagBranch%K7,&
                                         MagBranch%K8
  READ(Data_Section(3),*,ERR=999,END=999)MagBranch%K9,MagBranch%K10,MagBranch%K11,&
                                         MagBranch%K12
  READ(Data_Section(4),*,ERR=999,END=999)MagBranch%K13,MagBranch%init,MagBranch%fluxinit
  READ(Data_Section(5),*,ERR=999,END=999)MagBranch%nomvolt,MagBranch%n_stack,mlines

  !Data checking
  IF((MagBranch%init < 1) .OR. (MagBranch%init > 3)) THEN
    CALL device_error_(MagBranch%myname,'Initial status is wrong',.TRUE.)
  END IF
  IF(MagBranch%n_stack <= 0) THEN
    CALL device_error_(MagBranch%myname,'Increase the number of stack elements',.TRUE.)
  END IF

  ALLOCATE(MagBranch%max_stack(MagBranch%n_stack,2), STAT=istat)
  IF(istat.NE.0) CALL device_error_(MagBranch%myname,'Internal bug',.TRUE.)
  ALLOCATE(MagBranch%min_stack(MagBranch%n_stack,2), STAT=istat)
  IF(istat.NE.0) CALL device_error_(MagBranch%myname,'Internal bug',.TRUE.)

  !Push high saturation values into the stacks
  CALL PUSH_STACK(.TRUE.,SatCurrent,PhiMaj(SatCurrent,.TRUE.))
  CALL PUSH_STACK(.FALSE.,-SatCurrent,PhiMaj(-SatCurrent,.FALSE.))

  !If zero initial conditions is chosen,
  !fill the stack with the normal magnetization curve
  IF(MagBranch%init==3) THEN
    DO ii=6,mlines  !The first five lines have been read
      READ(Data_Section(ii),*,ERR=999,END=999)current,flux !The extrema are in ModelData
      CALL PUSH_STACK(flux>zero,current,flux)
    END DO
  END IF

  CALL FIND_COER()

  RETURN
  999 CALL device_error_(MagBranch%myname,'Data is wrong',.TRUE.)
END SUBROUTINE DLL_INITIALIZE_NEW

INCLUDE 'dll_data_pointers.f90'

SUBROUTINE POP_STACK(max_stack) !To remove an extrema from the stack
```

```
  USE NewMagBranch
  LOGICAL,       INTENT(IN) :: max_stack !If the selected stack is the maxima stack

  IF(MagBranch%n>0 .AND. max_stack)         MagBranch%n=MagBranch%n-1
  IF(MagBranch%m>0 .AND. (.NOT. max_stack)) MagBranch%m=MagBranch%m-1
END SUBROUTINE POP_STACK

SUBROUTINE PUSH_STACK(max_stack,current,flux) !To add an extrema to the stack
  USE NewMagBranch
  LOGICAL,       INTENT(IN) :: max_stack      !If the selected stack is the maxima stack
  REAL(krealhp), INTENT(IN) :: current        !Current value for stack
  REAL(krealhp), INTENT(IN) :: flux           !Flux value for stack

  IF((MagBranch%n .EQ. MagBranch%n_stack).OR.(MagBranch%m .EQ. MagBranch%n_stack)) THEN
    CALL device_error_(MagBranch%myname,'Maximum number of stack elements reached,&
                                        increase n_stack',.TRUE.)
  END IF

  IF (max_stack) THEN
    MagBranch%n=MagBranch%n+1   !To keep the count of maxima
    MagBranch%max_stack(MagBranch%n,1)=current
    MagBranch%max_stack(MagBranch%n,2)=flux
  ELSE
    MagBranch%m=MagBranch%m+1   !To keep the count of minima
    MagBranch%min_stack(MagBranch%m,1)=current
    MagBranch%min_stack(MagBranch%m,2)=flux
  END IF
END SUBROUTINE PUSH_STACK

SUBROUTINE FIND_CONSTANTS() !To find the constants for the minor loops after overtaking
  USE NewMagBranch

  IF(MagBranch%upward) THEN
    MagBranch%Cnu(1)=MagBranch%max_stack(MagBranch%n,2)&
                     -PhiMaj(MagBranch%max_stack(MagBranch%n,1),.TRUE.)
    MagBranch%Aold(1)=A(MagBranch%max_stack(MagBranch%n-1,1),.TRUE.)
    MagBranch%Anew(1)=A(MagBranch%max_stack(MagBranch%n,1),.TRUE.)
    MagBranch%Cnd(1)=MagBranch%max_stack(MagBranch%n-1,2)&
                     -PhiMaj(MagBranch%max_stack(MagBranch%n-1,1),.TRUE.)
  ELSE
    MagBranch%Cnu(1) =MagBranch%min_stack(MagBranch%m,2)&
                     -PhiMaj(MagBranch%min_stack(MagBranch%m,1),.FALSE.)
    MagBranch%Aold(1)=A(MagBranch%min_stack(MagBranch%m-1,1),.FALSE.)
    MagBranch%Anew(1)=A(MagBranch%min_stack(MagBranch%m,1),.FALSE.)
    MagBranch%Cnd(1) =MagBranch%min_stack(MagBranch%m-1,2)&
                      -PhiMaj(MagBranch%min_stack(MagBranch%m-1,1),.FALSE.)
  END IF
  MagBranch%Cnu(2) =MagBranch%Cnu(1)
  MagBranch%Cnd(2) =MagBranch%Cnd(1)
  MagBranch%Anew(2)=MagBranch%Anew(1)
  MagBranch%Aold(2)=MagBranch%Aold(1)
END SUBROUTINE FIND_CONSTANTS

SUBROUTINE FIND_COER() !To find the coercivity
  USE NewMagBranch
  REAL(krealhp)    :: inew     !The current at operating point Q
  REAL(krealhp)    :: iold(2)  !The last iteration current, used to iterate for Newton
  REAL(krealhp)    :: iref     !In case of oscillations
  REAL(krealhp)    :: f        !phi(iold)
  REAL(krealhp)    :: df       !phi'(iold)
  INTEGER :: ii
  INTEGER :: ktol

  ktol=1
  !The initial guess, a good guess might be around K3+K7+K11
  iold(1)=MagBranch%K3+MagBranch%K7+MagBranch%K11
  iref=iold(1)
  iold(2)=zero
  DO ii=1,SolMet%MaxNumberIter
    f=PhiMaj(iold(1),.TRUE.)
    df=dPhiMaj(iold(1),.TRUE.)
    inew=iold(1)-(f/df)
    !If convergence is not met, update iold
    IF(is_NE(inew,iold(1),ktol*SolMet%NonlConvergeTol) .OR. (inew .NE. inew)) THEN
      !After 10 iterations or numerical oscillations are detected,
      !reset iold to zero to help converge
      IF(inew==iold(2) .OR. ii==10) THEN
```

```
        iold(2)=iold(1)
        iref=iref/2.0_krealhp
        iold(1)=iref
        ktol=5
      ELSE
        iold(2)=iold(1)
        iold(1)=inew
      END IF
    ELSE   !Convergence met, return the coercivity
      MagBranch%coer=inew
      RETURN
    END IF
  END DO
  CALL convergence_problem_(MagBranch%myname) !In case the maximum iteration is reached
END SUBROUTINE FIND_COER

!Subroutine used to find the corresponding device
!in the chained list of magnetizing branches
SUBROUTINE FIND_MYDEV(idev)
  USE NewMagBranch
  INTEGER, INTENT(IN) :: idev !Unique device number for all DLL type devices
  INTEGER :: ii

  !Find the device with number idev and assign
  !the current pointer to the corresponding device
  DO ii=1,Total_number_of_devices
    IF( MagBranch%idev == idev) THEN
      RETURN
    ELSE
      MagBranch=>MagBranch%next
    END IF
  END DO
END SUBROUTINE FIND_MYDEV

SUBROUTINE FIND_FLUXGUESS(iguess,fluxnew,fluxguess)
  USE NewMagBranch
  REAL(krealhp), INTENT(IN)  :: iguess
  REAL(krealhp), INTENT(IN)  :: fluxnew
  REAL(krealhp), INTENT(OUT) :: fluxguess
  LOGICAL  :: upward
  LOGICAL  :: demagnetized

  upward=MagBranch%upward
  demagnetized=MagBranch%demagnetized
  IF(.NOT. MagBranch%upward) THEN  !For the downward major loop curve
    IF(MagBranch%reversalguess) THEN
      IF(iguess .GT. MagBranch%max_stack(MagBranch%n,1)) THEN
        upward=.TRUE.              !There is no reversal change trajectory
      ELSE
        MagBranch%Cnu(2)=MagBranch%Cnu(1)
        MagBranch%Aold(2)=MagBranch%Aold(1)
        MagBranch%Anew(2)=MagBranch%Anew(1)
        MagBranch%Cnd(2)=MagBranch%Cnd(1)
      END IF
      MagBranch%reversalguess=.FALSE.
    END IF

    MagBranch%Cn(2)=MagBranch%Cnu(2)*((MagBranch%Aold(2)-A(iguess,upward))&
                                      /(MagBranch%Aold(2)-MagBranch%Anew(2)))&
                    +MagBranch%Cnd(2)*((MagBranch%Anew(2)-A(iguess,upward))&
                                      /(MagBranch%Anew(2)-MagBranch%Aold(2)))

    IF(MagBranch%Aold(2)==MagBranch%Anew(2)) THEN
        MagBranch%Cn(2)=zero
        MagBranch%Cnd(2)=zero
        MagBranch%Cnu(2)=zero
        MagBranch%Aold(2)=zero
    END IF

  ELSE  !For the upward major loop curve
    IF(MagBranch%reversalguess) THEN
      IF(iguess .LT. MagBranch%min_stack(MagBranch%m,1)) THEN
        upward=.FALSE.             !There is no reversal change trajectory
      ELSE
        MagBranch%Cnu(2)=MagBranch%Cnu(1)
        MagBranch%Aold(2)=MagBranch%Aold(1)
        MagBranch%Anew(2)=MagBranch%Anew(1)
```

```
          MagBranch%Cnd(2)=MagBranch%Cnd(1)
        END IF
        MagBranch%reversalguess=.FALSE.
      END IF

      MagBranch%Cn(2)=MagBranch%Cnu(2)*((MagBranch%Aold(2)-A(iguess,upward))&
                                        /(MagBranch%Aold(2)-MagBranch%Anew(2)))&
                     +MagBranch%Cnd(2)*((MagBranch%Anew(2)-A(iguess,upward))&
                                        /(MagBranch%Anew(2)-MagBranch%Aold(2)))

      IF(MagBranch%Aold(2)==MagBranch%Anew(2)) THEN
          MagBranch%Cn(2)=zero
          MagBranch%Cnd(2)=zero
          MagBranch%Cnu(2)=zero
          MagBranch%Aold(2)=zero
      END IF
    END IF

    fluxguess=PhiMaj(iguess,upward)+MagBranch%Cn(2)

    IF(DABS(fluxnew-MagBranch%fluxkm) <= DABS(fluxguess-MagBranch%fluxkm)) THEN
      MagBranch%fluxkm=fluxnew
    ELSE
      MagBranch%fluxkm=fluxguess
      MagBranch%ikm=iguess
      IF(MagBranch%upward .NEQV. upward) THEN
        MagBranch%upward=upward
        MagBranch%demagnetized=demagnetized
        MagBranch%Cnu(1)=MagBranch%Cnu(2)
        MagBranch%Aold(1)=MagBranch%Aold(2)
        MagBranch%Anew(1)=MagBranch%Anew(2)
        MagBranch%Cnd(1)=MagBranch%Cnd(2)
        CALL POP_STACK(upward)
      END IF
    END IF

  END SUBROUTINE FIND_FLUXGUESS

  SUBROUTINE FIND_SS_CURRENT(flux,inew)
    USE NewMagBranch
    REAL(krealhp), INTENT(IN)  :: flux  !The flux operating point
    REAL(krealhp), INTENT(OUT) :: inew  !The current at operating point
    REAL(krealhp)  :: iold(2)  !The last iteration current, used to iterate for Newton
    REAL(krealhp)  :: f        !phi(iold)
    REAL(krealhp)  :: df       !phi'(iold)
    INTEGER :: ii

    iold(1)=MagBranch%coer !The initial guess
    iold(2)=zero
    DO ii=1,SolMet%MaxNumberIter
      IF(.NOT. MagBranch%upward) THEN  !For the downward major loop curve
        f=PhiMaj(iold(1),.FALSE.)-flux
        df=dPhiMaj(iold(1),.FALSE.)
        inew=iold(1)-(f/df)
      ELSE  !For the upward major loop curve
        f=PhiMaj(iold(1),.TRUE.)-flux
        df=dPhiMaj(iold(1),.TRUE.)
        inew=iold(1)-(f/df)
      END IF
      IF(is_NE(inew,iold(1),ktol*SolMet%NonlConvergeTol) .OR. (inew .NE. inew)) THEN
        !If convergence is not met, update iold
        !After 10 iterations or numerical oscillations are detected,
        !reset iold to coer to help converge
        IF(inew==iold(2) .OR. ii==10) THEN
          iold(2)=iold(1)
          iold(1)=((-1)**(.NOT. MagBranch%upward))*MagBranch%coer
          ktol=5
        ELSE
          iold(2)=iold(1)
          iold(1)=inew
        END IF
      ELSE   !Convergence met, return the current operating point
        RETURN
      END IF
    END DO

    CALL convergence_problem_(MagBranch%myname)  !In case the maximum iteration is reached
```

```
END SUBROUTINE FIND_SS_CURRENT

!This subroutine uses the Newton method to solve
!for the current until tolerance is reached
SUBROUTINE FIND_CURRENT(flux,inew)
  USE NewMagBranch
  REAL(krealhp), INTENT(IN)  :: flux  !The flux operating point
  REAL(krealhp), INTENT(OUT) :: inew  !The current at operating point
  REAL(krealhp)      :: iold(2)  !The last iteration current, used to iterate for Newton
  REAL(krealhp)      :: f        !phi(iold)
  REAL(krealhp)      :: df       !phi'(iold)
  INTEGER :: ii
  INTEGER :: ktol

  MagBranch%overtake=.FALSE.
  ktol=1
  !The initial guess, a good guess might be the last current operating point
  iold(1)=MagBranch%iold
  iold(2)=zero
  DO ii=1,SolMet%MaxNumberIter
    IF(.NOT. MagBranch%upward) THEN  !For the downward major loop curve
      IF(MagBranch%reversal) THEN !There is a reversal, recalculate the constants
        MagBranch%Cnu(1)=MagBranch%max_stack(MagBranch%n,2)&
                         -PhiMaj(MagBranch%max_stack(MagBranch%n,1),.FALSE.)

        MagBranch%Aold(1)=A(MagBranch%min_stack(MagBranch%m,1),.FALSE.)
        MagBranch%Anew(1)=A(MagBranch%max_stack(MagBranch%n,1),.FALSE.)

        MagBranch%Cnd(1)=MagBranch%min_stack(MagBranch%m,2)&
                         -PhiMaj(MagBranch%min_stack(MagBranch%m,1),.FALSE.)

        MagBranch%reversal=.FALSE.
        MagBranch%reversalguess=.TRUE.
        MagBranch%demagnetized=.FALSE.
      END IF

      MagBranch%Cn(1)=MagBranch%Cnu(1)*((MagBranch%Aold(1)-A(iold(1),.FALSE.))&
                                        /(MagBranch%Aold(1)-MagBranch%Anew(1)))&
                     +MagBranch%Cnd(1)*((MagBranch%Anew(1)-A(iold(1),.FALSE.))&
                                        /(MagBranch%Anew(1)-MagBranch%Aold(1)))

      !In case we are in saturation and Aold=Anew with (reversal in saturation)
      IF(MagBranch%Aold(1)==MagBranch%Anew(1)) THEN
        MagBranch%Cn(1)  = zero
        MagBranch%Cnd(1) = zero
        MagBranch%Cnu(1) = zero
        MagBranch%Aold(1)= zero
      END IF

      f=PhiMaj(iold(1),.FALSE.)+MagBranch%Cn(1)-flux
      df=dPhiMaj(iold(1),.FALSE.)+dA(iold(1),.FALSE.)&
                                  *((MagBranch%Cnd(1)-MagBranch%Cnu(1))&
                                    /(MagBranch%Aold(1)-MagBranch%Anew(1)))
      inew=iold(1)-(f/df)
    ELSE  !For the upward major loop curve
      IF(MagBranch%reversal) THEN !There is a reversal, recalculate the constants
        MagBranch%Cnu(1)=MagBranch%min_stack(MagBranch%m,2)&
                         -PhiMaj(MagBranch%min_stack(MagBranch%m,1),.TRUE.)

        MagBranch%Aold(1)=A(MagBranch%max_stack(MagBranch%n,1),.TRUE.)
        MagBranch%Anew(1)=A(MagBranch%min_stack(MagBranch%m,1),.TRUE.)

        MagBranch%Cnd(1)=MagBranch%max_stack(MagBranch%n,2)&
                         -PhiMaj(MagBranch%max_stack(MagBranch%n,1),.TRUE.)

        MagBranch%reversal=.FALSE.
        MagBranch%reversalguess=.TRUE.
        MagBranch%demagnetized=.FALSE.
      END IF

      MagBranch%Cn(1)=MagBranch%Cnu(1)*((MagBranch%Aold(1)-A(iold(1),.TRUE.))&
                                        /(MagBranch%Aold(1)-MagBranch%Anew(1)))&
                     +MagBranch%Cnd(1)*((MagBranch%Anew(1)-A(iold(1),.TRUE.))&
                                        /(MagBranch%Anew(1)-MagBranch%Aold(1)))

      !In case we are in saturation and Aold=Anew(reversal in saturation)
      IF(MagBranch%Aold(1)==MagBranch%Anew(1)) THEN
```

```
        MagBranch%Cn(1)  = zero
        MagBranch%Cnd(1) = zero
        MagBranch%Cnu(1) = zero
        MagBranch%Aold(1)= zero
      END IF

      f=PhiMaj(iold(1),.TRUE.)+MagBranch%Cn(1)-flux
611   df=dPhiMaj(iold(1),.TRUE.)+dA(iold(1),.TRUE.)&
                                *((MagBranch%Cnd(1)-MagBranch%Cnu(1))&
                                  /(MagBranch%Aold(1)-MagBranch%Anew(1)))
      inew=iold(1)-(f/df)
    END IF
    IF(is_NE(inew,iold(1),ktol*SolMet%NonlConvergeTol) .OR. (inew .NE. inew)) THEN
      !If convergence is not met, update iold
      !After 10 iterations or numerical oscillations are detected,
      !reset iold to coer to help converge
      IF(inew==iold(2) .OR. ii==10) THEN
621     iold(2)=iold(1)
        iold(1)=((-1)**(.NOT. MagBranch%upward))*MagBranch%coer
        ktol=5
      ELSE
        iold(2)=iold(1)
        iold(1)=inew
      END IF
    ELSE   !Convergence met, return the current operating point
      RETURN
    END IF
631 END DO

  CALL convergence_problem_(MagBranch%myname)  !In case the maximum iteration is reached
END SUBROUTINE FIND_CURRENT

!This subroutine is used to find the Norton slope, which is i'(phi)
SUBROUTINE FIND_NSLOPE(flux,ynew)
  USE NewMagBranch
  REAL(krealhp), INTENT(IN)  :: flux   !Actual flux operating point
  REAL(krealhp), INTENT(OUT) :: ynew   !The Norton slope found
641
  IF((flux-MagBranch%fluxold) .NE. zero) THEN
    ynew=(MagBranch%ikm-MagBranch%iold)/(flux-MagBranch%fluxold)
  END IF
END SUBROUTINE FIND_NSLOPE

SUBROUTINE REMFLUX_INIT()
  USE NewMagBranch
  REAL(krealhp)              :: Cnm(2)
  REAL(krealhp)              :: Cnmold
651 REAL(krealhp)              :: Cnpold
  REAL(krealhp)              :: iold
  REAL(krealhp)              :: Cnp(2)
  REAL(krealhp)              :: Cnum
  REAL(krealhp)              :: Cnup
  REAL(krealhp)              :: Cndm
  REAL(krealhp)              :: Cndp
  REAL(krealhp)              :: Cnupold
  REAL(krealhp)              :: Cndpold
  REAL(krealhp)              :: Cnumold
661 REAL(krealhp)              :: Cndmold
  REAL(krealhp)              :: inew
  REAL(krealhp)              :: sol
  INTEGER :: ii
  INTEGER :: ktol

  MagBranch%upward=MagBranch%fluxinit<=zero
  MagBranch%remflux=PhiMaj(zero,MagBranch%upward)
  ktol=100

671 Cnum=zero
  Cnup=zero
  Cnupold=zero
  Cndpold=zero
  Cnumold=zero
  Cndmold=zero
  Cndp=zero
  Cndm=zero
  Cnp(2)=zero
  Cnm(2)=zero
```

```
    iold=zero
    sol=MagBranch%fluxinit-MagBranch%remflux

    IF(MagBranch%upward) THEN
      Cnp(1)=MagBranch%remflux
      inew=-SQRT(SatCurrent)
      Cnum=PhiMaj(inew,.TRUE.)-PhiMaj(inew,.FALSE.)
      Cnpold=Cnp(1)
      DO ii=2,MagBranch%n_stack
        Cnup=PhiMaj(inew,.FALSE.)-PhiMaj(inew,.TRUE.)+Cnm(2)
        Cndp=PhiMaj(MagBranch%max_stack(MagBranch%n,1),.FALSE.)&
             -PhiMaj(MagBranch%max_stack(MagBranch%n,1),.TRUE.)+Cnum
        CALL PUSH_STACK(.FALSE.,inew,PhiMaj(inew,.TRUE.)+Cnup)
        Cnp(1)=Cnup*((A(MagBranch%max_stack(MagBranch%n,1),.TRUE.)-A(zero,.TRUE.))&
                     /(A(MagBranch%max_stack(MagBranch%n,1),.TRUE.)&
                      -A(MagBranch%min_stack(MagBranch%m,1),.TRUE.)))&
              +Cndp*((A(MagBranch%min_stack(MagBranch%m,1),.TRUE.)-A(zero,.TRUE.))&
                     /(A(MagBranch%min_stack(MagBranch%m,1),.TRUE.)&
                      -A(MagBranch%max_stack(MagBranch%n,1),.TRUE.)))
        inew=-inew
        Cnp(2)=Cnup*((A(MagBranch%max_stack(MagBranch%n,1),.TRUE.)-A(inew,.TRUE.))&
                     /(A(MagBranch%max_stack(MagBranch%n,1),.TRUE.)&
                      -A(MagBranch%min_stack(MagBranch%m,1),.TRUE.)))&
              +Cndp*((A(MagBranch%min_stack(MagBranch%m,1),.TRUE.)-A(inew,.TRUE.))&
                     /(A(MagBranch%min_stack(MagBranch%m,1),.TRUE.)&
                      -A(MagBranch%max_stack(MagBranch%n,1),.TRUE.)))
        Cnum=PhiMaj(inew,.TRUE.)-PhiMaj(inew,.FALSE.)+Cnp(2)
        Cndm=PhiMaj(MagBranch%min_stack(MagBranch%m,1),.TRUE.)&
             -PhiMaj(MagBranch%min_stack(MagBranch%m,1),.FALSE.)+Cnup
        CALL PUSH_STACK(.TRUE.,inew,PhiMaj(inew,.FALSE.)+Cnum)
        IF(inew>2.0_krealhp) THEN
          inew=-SQRT(inew)
        ELSE
          inew=-inew/2
        END IF
        Cnm(2)=Cnum*((A(MagBranch%min_stack(MagBranch%m,1),.FALSE.)-A(inew,.FALSE.))&
                     /(A(MagBranch%min_stack(MagBranch%m,1),.FALSE.)&
                      -A(MagBranch%max_stack(MagBranch%n,1),.FALSE.)))&
              +Cndm*((A(MagBranch%max_stack(MagBranch%n,1),.FALSE.)-A(inew,.FALSE.))&
                     /(A(MagBranch%max_stack(MagBranch%n,1),.FALSE.)&
                      -A(MagBranch%min_stack(MagBranch%m,1),.FALSE.)))
        IF(is_NE(Cnp(1),sol,ktol*SolMet%NonlConvergeTol)) THEN
          IF(Cnp(1)>sol) THEN
            inew=iold-((iold-MagBranch%min_stack(MagBranch%m,1))&
                       *((MagBranch%remflux+Cnpold-MagBranch%fluxinit)/(Cnpold-Cnp(1))))
            Cnum=Cnumold
            Cndm=Cndmold
            CALL POP_STACK(.FALSE.)
            CALL POP_STACK(.TRUE.)
            Cnm(2)=Cnum*((A(MagBranch%min_stack(MagBranch%m,1),.FALSE.)-A(inew,.FALSE.))&
                         /(A(MagBranch%min_stack(MagBranch%m,1),.FALSE.)&
                          -A(MagBranch%max_stack(MagBranch%n,1),.FALSE.)))&
                  +Cndm*((A(MagBranch%max_stack(MagBranch%n,1),.FALSE.)-A(inew,.FALSE.))&
                         /(A(MagBranch%max_stack(MagBranch%n,1),.FALSE.)&
                          -A(MagBranch%min_stack(MagBranch%m,1),.FALSE.)))
          ELSE
            Cnpold=Cnp(1)
            iold=MagBranch%min_stack(MagBranch%m,1)
            Cnumold=Cnum
            Cndmold=Cndm
          END IF
        ELSE
          CALL POP_STACK(.TRUE.)
          MagBranch%Cnu(1)=Cnup
          MagBranch%Cnd(1)=Cndp
          MagBranch%Aold(1)=A(MagBranch%max_stack(MagBranch%n,1),.TRUE.)
          MagBranch%Anew(1)=A(MagBranch%min_stack(MagBranch%m,1),.TRUE.)
          RETURN
        END IF
      END DO
    ELSE
      Cnm(1)=MagBranch%remflux
      inew=SQRT(SatCurrent)
      Cnup=PhiMaj(inew,.FALSE.)-PhiMaj(inew,.TRUE.)
      Cnmold=Cnm(1)
      DO ii=2,MagBranch%n_stack
        Cnum=PhiMaj(inew,.TRUE.)-PhiMaj(inew,.FALSE.)+Cnp(2)
```

```
      Cndm=PhiMaj(MagBranch%min_stack(MagBranch%m,1),.TRUE.)&
           -PhiMaj(MagBranch%min_stack(MagBranch%m,1),.FALSE.)+Cnup
      CALL PUSH_STACK(.TRUE.,inew,PhiMaj(inew,.FALSE.)+Cnum)
      Cnm(1)=Cnum*((A(MagBranch%min_stack(MagBranch%m,1),.FALSE.)-A(zero,.FALSE.))&
                  /(A(MagBranch%min_stack(MagBranch%m,1),.FALSE.)&
                   -A(MagBranch%max_stack(MagBranch%n,1),.FALSE.)))&
            +Cndm*((A(MagBranch%max_stack(MagBranch%n,1),.FALSE.)-A(zero,.FALSE.))&
                  /(A(MagBranch%max_stack(MagBranch%n,1),.FALSE.)&
                   -A(MagBranch%min_stack(MagBranch%m,1),.FALSE.)))
      inew=-inew
      Cnm(2)=Cnum*((A(MagBranch%min_stack(MagBranch%m,1),.FALSE.)-A(inew,.FALSE.))&
                  /(A(MagBranch%min_stack(MagBranch%m,1),.FALSE.)&
                   -A(MagBranch%max_stack(MagBranch%n,1),.FALSE.)))&
            +Cndm*((A(MagBranch%max_stack(MagBranch%n,1),.FALSE.)-A(inew,.FALSE.))&
                  /(A(MagBranch%max_stack(MagBranch%n,1),.FALSE.)&
                   -A(MagBranch%min_stack(MagBranch%m,1),.FALSE.)))
      Cnup=PhiMaj(inew,.FALSE.)-PhiMaj(inew,.TRUE.)+Cnm(2)
      Cndp=PhiMaj(MagBranch%max_stack(MagBranch%n,1),.FALSE.)&
           -PhiMaj(MagBranch%max_stack(MagBranch%n,1),.TRUE.)+Cnum
      CALL PUSH_STACK(.FALSE.,inew,PhiMaj(inew,.TRUE.)+Cnup)
      IF(inew<-2.0_krealhp) THEN
        inew=SQRT(-inew)
      ELSE
        inew=-inew/2
      END IF
      Cnp(2)=Cnup*((A(MagBranch%max_stack(MagBranch%n,1),.TRUE.)-A(inew,.TRUE.))&
                  /(A(MagBranch%max_stack(MagBranch%n,1),.TRUE.)&
                   -A(MagBranch%min_stack(MagBranch%n,1),.TRUE.)))&
            +Cndp*((A(MagBranch%min_stack(MagBranch%m,1),.TRUE.)-A(inew,.TRUE.))&
                  /(A(MagBranch%min_stack(MagBranch%m,1),.TRUE.)&
                   -A(MagBranch%max_stack(MagBranch%n,1),.TRUE.)))
      IF(is_NE(Cnm(1),sol,ktol*SolMet%NonlConvergeTol)) THEN
        IF(Cnm(1)<sol) THEN
          inew=iold-((iold-MagBranch%max_stack(MagBranch%n,1))&
                     *((MagBranch%remflux+Cnmold-MagBranch%fluxinit)/(Cnmold-Cnm(1))))
          Cnup=Cnupold
          Cndp=Cndpold
          CALL POP_STACK(.FALSE.)
          CALL POP_STACK(.TRUE.)
          Cnp(2)=Cnup*((A(MagBranch%max_stack(MagBranch%n,1),.TRUE.)-A(inew,.TRUE.))&
                      /(A(MagBranch%max_stack(MagBranch%n,1),.TRUE.)&
                       -A(MagBranch%min_stack(MagBranch%n,1),.TRUE.)))&
                +Cndp*((A(MagBranch%min_stack(MagBranch%m,1),.TRUE.)-A(inew,.TRUE.))&
                      /(A(MagBranch%min_stack(MagBranch%m,1),.TRUE.)&
                       -A(MagBranch%max_stack(MagBranch%n,1),.TRUE.)))
        ELSE
          Cnmold=Cnm(1)
          iold=MagBranch%max_stack(MagBranch%n,1)
          Cnupold=Cnup
          Cndpold=Cndp
        END IF
      ELSE
        CALL POP_STACK(.FALSE.)
        MagBranch%Cnu(1)=Cnum
        MagBranch%Cnd(1)=Cndm
        MagBranch%Aold(1)=A(MagBranch%min_stack(MagBranch%m,1),.FALSE.)
        MagBranch%Anew(1)=A(MagBranch%max_stack(MagBranch%n,1),.FALSE.)
        RETURN
      END IF
    END DO
  END IF

  CALL convergence_problem_(MagBranch%myname)  !In case the maximum iteration is reached
END SUBROUTINE REMFLUX_INIT

!This request is used to save the global node numbers given by EMTP for each device
SUBROUTINE DLL_POST_INITIALIZE_NEW(myname,idev,power_signal_nodes,n_nodes)
  USE NewMagBranch
  !DEC$ ATTRIBUTES DLLEXPORT:: DLL_POST_INITIALIZE_NEW
  CHARACTER(LEN=*), INTENT(IN) :: myname  !Unique name for this device coming from EMTP
  INTEGER,          INTENT(IN) :: idev    !Unique device number for all DLL type devices
  INTEGER,          INTENT(IN) :: power_signal_nodes(*) !The actual global node numbers
  INTEGER,          INTENT(IN) :: n_nodes !Number of actual nodes

  INTEGER istat !Local, for allocation

  MagBranch=>MagBranch_first
```

```
   CALL FIND_MYDEV(idev)  !To find the current device

   !Save the global node numbers for later
   ALLOCATE(MagBranch%power_signal_nodes(n_nodes), STAT=istat)
   IF(istat.NE.0) CALL device_error_(MagBranch%myname,'Internal bug',.TRUE.)
   MagBranch%power_signal_nodes=power_signal_nodes(1:n_nodes)

END SUBROUTINE DLL_POST_INITIALIZE_NEW

!Called to fill the Yn part of the A matrix in A*x=b
!Steady-state solution
SUBROUTINE DLL_PUT_IN_YN_SS(idev,w)
  USE NewMagBranch
  !DEC$ ATTRIBUTES DLLEXPORT:: DLL_PUT_IN_YN_SS
  INTEGER,          INTENT(IN)    :: idev !Unique device number for all DLL type devices
  REAL(krealhp),    INTENT(IN)    :: w    !Solution frequency

  CALL FIND_MYDEV(idev)  !To find the current device

  IF(MagBranch%init == 1) THEN !If this device participates in the steady-state solution
    CALL FIND_COER()
    MagBranch%Yss=1/(jz*w*dPhiMaj(MagBranch%coer,.TRUE.))&
                  +1/(MagBranch%nomvolt/MagBranch%coer) !Y=1/jwL+1/R
  ELSE
    MagBranch%Yss=lowadmittance !Low admittance value to avoid floating network warnings
  END IF

  !We give the nodal admittance values to EMTP through the fill_ request
  CALL fill_(MagBranch%power_signal_nodes(1),MagBranch%power_signal_nodes(1),&
             MagBranch%Yss) !k,k
  CALL fill_(MagBranch%power_signal_nodes(1),MagBranch%power_signal_nodes(2),&
            -MagBranch%Yss) !k,m
  CALL fill_(MagBranch%power_signal_nodes(2),MagBranch%power_signal_nodes(1),&
            -MagBranch%Yss) !m,k
  CALL fill_(MagBranch%power_signal_nodes(2),MagBranch%power_signal_nodes(2),&
             MagBranch%Yss) !m,m
END SUBROUTINE DLL_PUT_IN_YN_SS

!Frequency scan solution
SUBROUTINE DLL_PUT_IN_YN_SS_FREQSCAN(idev,w)
  USE NewMagBranch
  !DEC$ ATTRIBUTES DLLEXPORT:: DLL_PUT_IN_YN_SS_FREQSCAN
  INTEGER,       INTENT(IN) :: idev !Unique device number for all DLL type devices
  REAL(krealhp), INTENT(IN) :: w    !Solution frequency

  CALL DLL_PUT_IN_YN_SS(idev,w) !For this device, it is the same request
END SUBROUTINE DLL_PUT_IN_YN_SS_FREQSCAN

!The network solution phasors must be superposed for t=0
!This is equivalent to setting t=0 in a Fourier series
!It is assumed that phasors are based on the cosine function
SUBROUTINE DLL_SUPERPOSE_SS_AT_W(idev,w)
  USE NewMagBranch
  !DEC$ ATTRIBUTES DLLEXPORT:: DLL_SUPERPOSE_SS_AT_W
  INTEGER,       INTENT(IN)     :: idev !Unique device number for all DLL type devices
  REAL(krealhp), INTENT(IN)     :: w    !Solution frequency
  REAL(krealhp)                 :: flux_amplitude
  REAL(krealhp)                 :: flux_angle
  REAL(krealhp)                 :: flux_ini
  REAL(krealhp)                 :: flux_ini_dt

  CALL FIND_MYDEV(idev)

  IF(MagBranch%init == 1) THEN  !If this device participates to the steady-state
    !Superposition at each frequency
    MagBranch%h=AIMAG(SimData%vector_xc(MagBranch%power_signal_nodes(1))-&
                SimData%vector_xc(MagBranch%power_signal_nodes(2)))/w+MagBranch%h

    MagBranch%h_dt=ABS(SimData%vector_xc(MagBranch%power_signal_nodes(1))&
                   -SimData%vector_xc(MagBranch%power_signal_nodes(2)))/w&
                   *SIN((-SimTime%Dt)*w&
                   +ATAN2(AIMAG(SimData%vector_xc(MagBranch%power_signal_nodes(1))&
                   -SimData%vector_xc(MagBranch%power_signal_nodes(2))),&
                   REAL(SimData%vector_xc(MagBranch%power_signal_nodes(1))&
                   -SimData%vector_xc(MagBranch%power_signal_nodes(2)))))+MagBranch%h_dt
  END IF
```

```
911 END SUBROUTINE DLL_SUPERPOSE_SS_AT_W

    !This call is for printing the steady-state solution in the global steady-state web
    SUBROUTINE DLL_PRINT_SS(myname,idev,w,Current,Spower)
      USE NewMagBranch
      !DEC$ ATTRIBUTES DLLEXPORT:: DLL_PRINT_SS
      CHARACTER(LEN=*), INTENT(IN)  :: myname !Unique name for this device coming from EMTP
      INTEGER,          INTENT(IN)  :: idev   !Unique device number for all DLL type devices
      REAL(krealhp),    INTENT(IN)  :: w      !Solution frequency
      COMPLEX(krealhp), INTENT(OUT) :: Current(*)  !Current in ss for this device
921   COMPLEX(krealhp), INTENT(OUT) :: Spower(*)   !Complex power in ss for this device

      CALL FIND_MYDEV(idev)  !Find the current device

      MagBranch%Iss=(SimData%vector_xc(MagBranch%power_signal_nodes(1))&
            -SimData%vector_xc(MagBranch%power_signal_nodes(2)))*MagBranch%Yss !Iss=Vkm*Yss

      Current(1)=MagBranch%Iss   !Ik is Iss
      Spower(1)=SimData%vector_xc(MagBranch%power_signal_nodes(1))*CONJG(MagBranch%Iss)/2

931   Current(2)=-MagBranch%Iss  !Im is -Iss
      Spower(2)=SimData%vector_xc(MagBranch%power_signal_nodes(2))*CONJG(-MagBranch%Iss)/2
    END SUBROUTINE DLL_PRINT_SS

    !This function is called when we have zero initial conditions
    SUBROUTINE DLL_ZERO_INITIAL_CONDITIONS(idev)
      USE NewMagBranch
      !DEC$ ATTRIBUTES DLLEXPORT:: DLL_ZERO_INITIAL_CONDITIONS
      INTEGER, INTENT(IN) :: idev !Unique device number for all DLL type devices

941   CALL FIND_MYDEV(idev) !Find the current device

      MagBranch%h=zero    !Initialize the flux history to zero
      MagBranch%fluxkm=zero !Initialize flux to zero
      MagBranch%ikm=zero  !Initialize the current to zero
    END SUBROUTINE DLL_ZERO_INITIAL_CONDITIONS

    !Return signals on pins o1, o2 and o3 for observables,
    !they are sent to the control system
    SUBROUTINE DLL_LOAD_OBSERVABLES_T0(idev,Returned_obs_array)
951   USE NewMagBranch
      !DEC$ ATTRIBUTES DLLEXPORT:: DLL_LOAD_OBSERVABLES_T0
      INTEGER, INTENT(IN) :: idev   !Unique device number for all DLL type devices
      REAL(krealhp), INTENT(OUT)  :: Returned_obs_array(*) !Return all observables

      CALL FIND_MYDEV(idev) !Find the current device
      MagBranch%vkm=SimData%vector_x(MagBranch%power_signal_nodes(1))-&
                    SimData%vector_x(MagBranch%power_signal_nodes(2))
      IF(MagBranch%init==1) THEN
        MagBranch%fluxkm=MagBranch%h
961     IF(MagBranch%h .GT. MagBranch%h_dt) THEN
          MagBranch%upward=.TRUE. !upward trajectory
        ELSE
          MagBranch%upward=.FALSE. !downward trajectory
        END IF
        CALL FIND_SS_CURRENT(MagBranch%h,MagBranch%ikm)
      ELSE IF(MagBranch%init==2) THEN
        MagBranch%fluxkm=MagBranch%fluxinit
      END IF
      Returned_obs_array(1)=MagBranch%vkm  !Pin o1 returns the voltage of the device
971   Returned_obs_array(2)=MagBranch%ikm  !Pin o2 returns the current of the device
      Returned_obs_array(3)=MagBranch%fluxkm  !Pin o3 returns the flux of the device
    END SUBROUTINE DLL_LOAD_OBSERVABLES_T0

    !This function is called for initialization
    !with Trapezoidal integration for first time-step
    SUBROUTINE DLL_INIT_AT_T0(idev)
      USE NewMagBranch
      !DEC$ ATTRIBUTES DLLEXPORT:: DLL_INIT_AT_T0
      INTEGER, INTENT(IN) :: idev !Unique device number for all DLL type devices
981
      CALL FIND_MYDEV(idev) !Find the current device

      MagBranch%vkm=SimData%vector_x(MagBranch%power_signal_nodes(1))-&
                    SimData%vector_x(MagBranch%power_signal_nodes(2))
      IF(MagBranch%init == 1 .AND. ABS(MagBranch%fluxkm)>zero) THEN
```

```
      !If the device is initialized from steady-state
      MagBranch%h=SimTime%Dton2*MagBranch%vkm+MagBranch%h
      MagBranch%demagnetized=.FALSE.
    ELSE IF(MagBranch%init == 2) THEN
      MagBranch%h=MagBranch%fluxinit
      CALL REMFLUX_INIT()
    END IF
    MagBranch%fluxold=MagBranch%h
    MagBranch%iold=MagBranch%ikm
  END SUBROUTINE DLL_INIT_AT_T0

  !This function is called for initialization with EBA integration for first time-step
  SUBROUTINE DLL_EBA_INIT_AT_T0(idev)
    USE NewMagBranch
    !DEC$ ATTRIBUTES DLLEXPORT:: DLL_EBA_INIT_AT_T0
    INTEGER, INTENT(IN) :: idev !Unique device number for all DLL type devices

    CALL FIND_MYDEV(idev) !Find the current device

    IF(MagBranch%init == 1) THEN
      MagBranch%demagnetized=.FALSE.
    ELSE IF(MagBranch%init == 2) THEN
      MagBranch%h=MagBranch%fluxinit
      CALL REMFLUX_INIT()
    END IF
    MagBranch%fluxold=MagBranch%h
    MagBranch%iold=MagBranch%ikm
  END SUBROUTINE DLL_EBA_INIT_AT_T0

  !This function provides the location of the nonlinear device in the Yn matrix
  SUBROUTINE DLL_PUT_NODES_IN_YNONLIN(idev)
    USE NewMagBranch
    !DEC$ ATTRIBUTES DLLEXPORT:: DLL_PUT_NODES_IN_YNONLIN
    INTEGER, INTENT(IN) :: idev !Unique device number for all DLL type devices

    CALL FIND_MYDEV(idev) !Find the current device

    CALL put_nonl_(MagBranch%power_signal_nodes(1),MagBranch%power_signal_nodes(1)) !k,k
    CALL put_nonl_(MagBranch%power_signal_nodes(1),MagBranch%power_signal_nodes(2)) !k,m
    CALL put_nonl_(MagBranch%power_signal_nodes(2),MagBranch%power_signal_nodes(1)) !m,k
    CALL put_nonl_(MagBranch%power_signal_nodes(2),MagBranch%power_signal_nodes(2)) !m,m
  END SUBROUTINE DLL_PUT_NODES_IN_YNONLIN

  !Allows to make a prediction for the first iteration process for nonlinear devices
  !vector_nonl_b is set to 0 in the Core before entering this code
  !SimData%rebuild_for_nonl is forced by default in the Core at the end of ITER0
  SUBROUTINE DLL_ITER0(idev)
    USE NewMagBranch
    !DEC$ ATTRIBUTES DLLEXPORT:: DLL_ITER0
    INTEGER, INTENT(IN) :: idev !Unique device number for all DLL type devices
    REAL(krealhp) :: rev_cur

    CALL FIND_MYDEV(idev) !Find the current device

    MagBranch%vkm=SimData%vector_x(MagBranch%power_signal_nodes(1))&
                 -SimData%vector_x(MagBranch%power_signal_nodes(2)) !Instantaneous voltage
    MagBranch%fluxkm=MagBranch%vkm*SimTime%Dton2 + MagBranch%h

    IF(MagBranch%vkm>zero) THEN
      MagBranch%upward=.TRUE.  !If dphi/dt is positive (vkm positive), flux is upward
      IF(MagBranch%init==2 .AND. MagBranch%fluxinit>zero) THEN
        CALL PUSH_STACK(.FALSE.,zero,MagBranch%fluxinit)
      END IF
    ELSE IF(MagBranch%vkm<zero) THEN
      MagBranch%upward=.FALSE.
      IF(MagBranch%init==2 .AND. MagBranch%fluxinit<zero) THEN
        CALL PUSH_STACK(.TRUE.,zero,MagBranch%fluxinit)
      END IF
    END IF

    MagBranch%reversal=.TRUE.

    IF(ABS(MagBranch%vkm)>zero) THEN
      CALL FIND_CURRENT(MagBranch%fluxkm,MagBranch%ikm)  !Find the current
      CALL FIND_NSLOPE(MagBranch%fluxkm,MagBranch%y)     !1/Kq, temporary storage in y
      MagBranch%cur=-(MagBranch%y*MagBranch%fluxkm)+MagBranch%ikm  !-phiq/Kq
```

```
      MagBranch%Iq=(MagBranch%h*MagBranch%y)+MagBranch%cur  !The Norton current source
      MagBranch%y=MagBranch%y*SimTime%Dton2          !The Norton conductance
    ELSE
      MagBranch%ikm=zero
      MagBranch%cur=zero
      MagBranch%Iq=zero
      MagBranch%y=lowadmittance
    END IF

    !Provide the new admittance to EMTP
    CALL fill_(MagBranch%power_signal_nodes(1),MagBranch%power_signal_nodes(1),&
               MagBranch%y) !k,k
    CALL fill_(MagBranch%power_signal_nodes(1),MagBranch%power_signal_nodes(2),&
              -MagBranch%y) !k,m
    CALL fill_(MagBranch%power_signal_nodes(2),MagBranch%power_signal_nodes(1),&
              -MagBranch%y) !m,k
    CALL fill_(MagBranch%power_signal_nodes(2),MagBranch%power_signal_nodes(2),&
               MagBranch%y) !m,m

    !Update the Inonlin vector with the contribution of the Norton current source
    !Contribution to k
    SimData%vector_nonl_b(MagBranch%power_signal_nodes(1))=&
        SimData%vector_nonl_b(MagBranch%power_signal_nodes(1))-MagBranch%Iq

    !Contribution to m
    SimData%vector_nonl_b(MagBranch%power_signal_nodes(2))=&
        SimData%vector_nonl_b(MagBranch%power_signal_nodes(2))+MagBranch%Iq
  END SUBROUTINE DLL_ITER0

  !Iterative process for nonlinear devices
  SUBROUTINE DLL_ITER(idev,convergence_flag)
    USE NewMagBranch
    !DEC$ ATTRIBUTES DLLEXPORT:: DLL_ITER
    INTEGER, INTENT(IN)  :: idev !Unique device number for all DLL type devices
    LOGICAL, INTENT(OUT) :: convergence_flag !To tell EMTP if the device has converged
    REAL(krealhp) :: fluxnew
    REAL(krealhp) :: iguess
    REAL(krealhp) :: fluxguess

    CALL FIND_MYDEV(idev)  !Find the current device

    IF(SimTime%t >= 0.000166) THEN
      dummyx=1
    END IF

    MagBranch%converged=.FALSE.
    MagBranch%iter=MagBranch%iter+1

    !Acceleration factor for convergence
    IF(MagBranch%iter > iter_panic) THEN
      MagBranch%ktol=ktol_max
    END IF

    MagBranch%vkm=(SimData%vector_x(MagBranch%power_signal_nodes(1))-&
                   SimData%vector_x(MagBranch%power_signal_nodes(2))) !Instantaneous voltage
    fluxnew=MagBranch%vkm*SimTime%Dton2  !The new flux increment

    !When the flux starts at 0, we can't know if the trajectory will be upward or not
    !If the branch is deenergized, we put a low admittance value and wait for flux
    !to know if it is upward or not
    IF(MagBranch%demagnetized) THEN
      IF(MagBranch%vkm>zero) THEN
        MagBranch%upward=.TRUE.
        CALL PUSH_STACK(.FALSE.,MagBranch%iold,MagBranch%fluxold)
      ELSE IF(MagBranch%vkm<zero) THEN
        MagBranch%upward=.FALSE.
        CALL PUSH_STACK(.TRUE.,MagBranch%iold,MagBranch%fluxold)
      END IF
    END IF

    IF(ABS(fluxnew)>zero) THEN
      fluxnew=fluxnew+MagBranch%h  !The new flux
      !If tolerance not met for convergence
      IF(is_NE(MagBranch%fluxkm,fluxnew,MagBranch%ktol*SolMet%NonlConvergeTol)&
               .OR. MagBranch%reversal .OR. MagBranch%overtake) THEN
        SimData%rebuild_for_nonl=.TRUE. !We must rebuild since the device did not converge
        convergence_flag=.FALSE.        !Tell EMTP this device did not converge
```

```
      CALL FIND_CURRENT(fluxnew,MagBranch%ikm)    !Find the current from fluxnew
      IF(MagBranch%iter .NE. 1) THEN
        iguess=MagBranch%y*MagBranch%vkm+MagBranch%Iq
        CALL FIND_FLUXGUESS(iguess,fluxnew,fluxguess)  !Find the flux from iguess
      ELSE
        MagBranch%fluxkm=fluxnew
      END IF
      CALL FIND_NSLOPE(MagBranch%fluxkm,MagBranch%y) !1/Kq, temporary storage in y
      MagBranch%cur=-(MagBranch%y*MagBranch%fluxkm)+MagBranch%ikm  !-phiq/Kq
      MagBranch%Iq=(MagBranch%h*MagBranch%y)+MagBranch%cur  !The Norton current source
      MagBranch%y=MagBranch%y*SimTime%Dton2           !The Norton conductance
    ELSE  !Converged
      !If there's a reversal
      IF(MagBranch%upward .AND. ((MagBranch%fluxkm-MagBranch%fluxold)<-epsilon)) THEN
        MagBranch%upward=.FALSE.
        CALL PUSH_STACK(.TRUE.,MagBranch%iold,MagBranch%fluxold) !Save the last maximum
        convergence_flag=.FALSE.
        MagBranch%reversal=.TRUE.
        SolMet%can_relax=.FALSE.
      ELSE IF((.NOT. MagBranch%upward) .AND. &
              ((MagBranch%fluxkm-MagBranch%fluxold)>epsilon)) THEN
        MagBranch%upward=.TRUE.
        CALL PUSH_STACK(.FALSE.,MagBranch%iold,MagBranch%fluxold) !Save the last minimum
        convergence_flag=.FALSE.
        MagBranch%reversal=.TRUE.
        SolMet%can_relax=.FALSE.
      !If there's overtaking
      ELSE IF(MagBranch%upward .AND. &
                ((MagBranch%fluxkm-MagBranch%max_stack(MagBranch%n,2))>epsilon)) THEN
        convergence_flag=.FALSE.
        MagBranch%overtake=.TRUE.
        CALL FIND_CONSTANTS()
        CALL POP_STACK(.TRUE.)
        CALL POP_STACK(.FALSE.)
        SolMet%can_relax=.FALSE.
      ELSE IF((.NOT. MagBranch%upward) .AND. &
                ((MagBranch%fluxkm-MagBranch%min_stack(MagBranch%m,2))<-epsilon)) THEN
        convergence_flag=.FALSE.
        MagBranch%overtake=.TRUE.
        CALL FIND_CONSTANTS()
        CALL POP_STACK(.TRUE.)
        CALL POP_STACK(.FALSE.)
        SolMet%can_relax=.FALSE.
      ELSE
        MagBranch%converged =.TRUE.
        convergence_flag    =.TRUE.  !Tell EMTP this device has converged
      END IF
    END IF
  ELSE
    SimData%rebuild_for_nonl=.TRUE.
    convergence_flag=.FALSE.
    MagBranch%Iq=zero
    MagBranch%cur=zero
    MagBranch%y=lowadmittance
    MagBranch%fluxkm=MagBranch%h
  END IF

  !Update the admittance matrix with the new conductance
  CALL fill_(MagBranch%power_signal_nodes(1),MagBranch%power_signal_nodes(1),&
             MagBranch%y)
  CALL fill_(MagBranch%power_signal_nodes(1),MagBranch%power_signal_nodes(2),&
            -MagBranch%y)
  CALL fill_(MagBranch%power_signal_nodes(2),MagBranch%power_signal_nodes(1),&
            -MagBranch%y)
  CALL fill_(MagBranch%power_signal_nodes(2),MagBranch%power_signal_nodes(2),&
             MagBranch%y)

  !Update the Inonlin vector with the contribution of the Norton current source
  SimData%vector_nonl_b(MagBranch%power_signal_nodes(1))=&
      SimData%vector_nonl_b(MagBranch%power_signal_nodes(1))-MagBranch%Iq

  SimData%vector_nonl_b(MagBranch%power_signal_nodes(2))=&
      SimData%vector_nonl_b(MagBranch%power_signal_nodes(2))+MagBranch%Iq
END SUBROUTINE DLL_ITER

!Called only when something went wrong during the iterative process
!It is needed to find which device failed
```

```
SUBROUTINE DLL_CONVERGENCE_MESSAGE(idev)
  USE NewMagBranch
  !DEC$ ATTRIBUTES DLLEXPORT:: DLL_CONVERGENCE_MESSAGE
  INTEGER, INTENT(IN) :: idev !Unique device number for all DLL type devices
1221
  CALL FIND_MYDEV(idev)  !Find the current device

  !Send an error message when atleast one nonlinear device did not converge
  IF(.NOT. MagBranch%converged) THEN
    CALL convergence_problem_(MagBranch%myname)
  END IF
END SUBROUTINE

!Mandatory request, but not used for this device
1231 SUBROUTINE DLL_PUT_IN_IAUG(idev)
  USE NewMagBranch
  !DEC$ ATTRIBUTES DLLEXPORT:: DLL_PUT_IN_IAUG
  INTEGER, INTENT(IN) :: idev !Unique device number for all DLL type devices
END SUBROUTINE DLL_PUT_IN_IAUG

!Send observables to controls using Returned_obs_array
SUBROUTINE DLL_LOAD_OBSERVABLES(idev,Returned_obs_array)
  USE NewMagBranch
  !DEC$ ATTRIBUTES DLLEXPORT:: DLL_LOAD_OBSERVABLES
1241 INTEGER, INTENT(IN) :: idev !Unique device number for all DLL type devices
  REAL(krealhp), INTENT(OUT)  :: Returned_obs_array(*)

  CALL FIND_MYDEV(idev)  !Find the current device
  Returned_obs_array(1)=MagBranch%vkm     !Pin o1 corresponds to the device's voltage
  Returned_obs_array(2)=MagBranch%ikm     !Pin o2 corresponds to the device's current
  Returned_obs_array(3)=MagBranch%fluxkm  !Pin o3 corresponds to the device's flux
END SUBROUTINE DLL_LOAD_OBSERVABLES

!This function is used to request a discontinuity treatment with SolMet%discon_found
1251 !Not used for this device
SUBROUTINE DLL_UPDATE_STATUS_AT_T(idev)
  USE NewMagBranch
  !DEC$ ATTRIBUTES DLLEXPORT:: DLL_UPDATE_STATUS_AT_T
  INTEGER :: idev  !Unique device number from the list of all DLL type devices
END SUBROUTINE DLL_UPDATE_STATUS_AT_T

!Integration when using Trapezoidal method
SUBROUTINE DLL_UPDATE_AT_T(idev)
  USE NewMagBranch
1261 !DEC$ ATTRIBUTES DLLEXPORT:: DLL_UPDATE_AT_T
  INTEGER, INTENT(IN)          :: idev !Unique device number for all DLL type devices

  CALL FIND_MYDEV(idev)  !Find the current device

  MagBranch%vkm=(SimData%vector_x(MagBranch%power_signal_nodes(1))&
         -SimData%vector_x(MagBranch%power_signal_nodes(2))) !Save the device's voltage
  MagBranch%h=MagBranch%vkm*SimTime%Dton2+MagBranch%fluxkm   !Update the flux history

  !Make a prediction on Iq
1271 MagBranch%Iq=(MagBranch%h*MagBranch%y/SimTime%Dton2)+MagBranch%cur

  MagBranch%fluxold=MagBranch%fluxkm !Save the last value of flux for the next time step
  MagBranch%iold=MagBranch%ikm    !Save the last value of current for the next time step

  MagBranch%ktol=ktol_min
  MagBranch%iter=0

  !Update the Inonlin vector with the contribution of the prediction on Iq
  !Contribution to k
1281 SimData%vector_nonl_b(MagBranch%power_signal_nodes(1))=&
      SimData%vector_nonl_b(MagBranch%power_signal_nodes(1))-MagBranch%Iq

  !Contribution to m
  SimData%vector_nonl_b(MagBranch%power_signal_nodes(2))=&
      SimData%vector_nonl_b(MagBranch%power_signal_nodes(2))+MagBranch%Iq
END SUBROUTINE DLL_UPDATE_AT_T

!Integration when using EBA method
SUBROUTINE DLL_EBA_UPDATE_AT_T(idev)
1291 USE NewMagBranch
  !DEC$ ATTRIBUTES DLLEXPORT:: DLL_EBA_UPDATE_AT_T
```

```
   INTEGER, INTENT(IN)          :: idev !Unique device number for all DLL type devices

   CALL FIND_MYDEV(idev)  !Find the current device

   MagBranch%vkm=(SimData%vector_x(MagBranch%power_signal_nodes(1))&
         -SimData%vector_x(MagBranch%power_signal_nodes(2)))
!Save the device's voltage
   MagBranch%h=MagBranch%fluxkm       !Update the flux history

   MagBranch%fluxold=MagBranch%fluxkm !Save the last value of flux for the next time step
   MagBranch%iold=MagBranch%ikm    !Save the last value of current for the next time step

   MagBranch%ktol=ktol_min
   MagBranch%iter=0

   !Make a prediction on Iq
   MagBranch%Iq=(MagBranch%h*MagBranch%y/SimTime%Dton2)+MagBranch%cur

   !Contribution to k
   SimData%vector_nonl_b(MagBranch%power_signal_nodes(1))=&
       SimData%vector_nonl_b(MagBranch%power_signal_nodes(1))-MagBranch%Iq

   !Contribution to m
   SimData%vector_nonl_b(MagBranch%power_signal_nodes(2))=&
       SimData%vector_nonl_b(MagBranch%power_signal_nodes(2))+MagBranch%Iq
END SUBROUTINE DLL_EBA_UPDATE_AT_T

!Integration when moving from EBA to TRAP
SUBROUTINE DLL_EBATOTRAP_UPDATE_AT_T(idev)
  USE NewMagBranch
  !DEC$ ATTRIBUTES DLLEXPORT:: DLL_EBATOTRAP_UPDATE_AT_T
  INTEGER, INTENT(IN)          :: idev !Unique device number for all DLL type devices

  CALL DLL_UPDATE_AT_T(idev)  !Same function
END SUBROUTINE DLL_EBATOTRAP_UPDATE_AT_T

!Integration when moving from TRAP to EBA
SUBROUTINE DLL_TRAPTOEBA_UPDATE_AT_T(idev)
  USE NewMagBranch
  !DEC$ ATTRIBUTES DLLEXPORT:: DLL_TRAPTOEBA_UPDATE_AT_T
  INTEGER, INTENT(IN)          :: idev !Unique device number for all DLL type devices

  CALL DLL_EBA_UPDATE_AT_T(idev) !Same function
END SUBROUTINE DLL_TRAPTOEBA_UPDATE_AT_T

!Statistical option
!The DLL must save all time-dependent data
!to be able to restart the simulation correctly
SUBROUTINE DLL_SAVE_ME(idev)
  USE NewMagBranch
  !DEC$ ATTRIBUTES DLLEXPORT :: DLL_SAVE_ME
  INTEGER, INTENT(IN)          :: idev !Unique device number for all DLL type devices

  CALL FIND_MYDEV(idev)  !Find the current device

  !Save all time-dependant data
  CALL save_data_into_bin_(MagBranch%Iq)
  CALL save_data_into_bin_(MagBranch%fluxkm)
  CALL save_data_into_bin_(MagBranch%h)
  CALL save_data_into_bin_(MagBranch%y)
  CALL save_data_into_bin_(MagBranch%cur)
  CALL save_data_into_bin_(MagBranch%ikm)
  CALL save_data_into_bin_(MagBranch%vkm)
  CALL save_data_into_bin_(MagBranch%max_stack)
  CALL save_data_into_bin_(MagBranch%min_stack)
END SUBROUTINE DLL_SAVE_ME

!Statistical option
!The DLL must load all time-dependent data
!to be able to restart the simulation correctly
SUBROUTINE DLL_LOAD_ME(idev)
  USE NewMagBranch
  !DEC$ ATTRIBUTES DLLEXPORT :: DLL_LOAD_ME
  INTEGER, INTENT(IN)          :: idev !Unique device number for all DLL type devices

  CALL FIND_MYDEV(idev)  !Find the current device
```

```
  !Load all time-dependant data
  CALL load_data_from_bin_(MagBranch%Iq)
  CALL load_data_from_bin_(MagBranch%fluxkm)
  CALL load_data_from_bin_(MagBranch%h)
  CALL load_data_from_bin_(MagBranch%y)
  CALL load_data_from_bin_(MagBranch%cur)
  CALL load_data_from_bin_(MagBranch%ikm)
  CALL load_data_from_bin_(MagBranch%vkm)
  CALL load_data_from_bin_(MagBranch%max_stack)
  CALL load_data_from_bin_(MagBranch%min_stack)
END SUBROUTINE DLL_LOAD_ME

!This request is called for ending the DLL
SUBROUTINE DLL_END(idev)
  USE NewMagBranch
  !DEC$ ATTRIBUTES DLLEXPORT  :: DLL_END
  INTEGER, INTENT(IN)         :: idev !Unique device number for all DLL type devices

  CALL DEALLOCATE_MAGBRANCH() !To deallocate pointers
END SUBROUTINE DLL_END

!This function is called to make some garbage collect
SUBROUTINE DEALLOCATE_MAGBRANCH()
  USE NewMagBranch
  INTEGER :: ii
  IF(EXISTENCE) THEN
    DO ii=1,Total_number_of_devices-1
      MagBranch=>MagBranch_first%next
      DEALLOCATE(MagBranch_first%power_signal_nodes)
      DEALLOCATE(MagBranch_first%max_stack)
      DEALLOCATE(MagBranch_first%min_stack)
      DEALLOCATE(MagBranch_first)
      MagBranch_first=>MagBranch
    END DO
    DEALLOCATE(MagBranch_first%power_signal_nodes)
    DEALLOCATE(MagBranch_first%max_stack)
    DEALLOCATE(MagBranch_first%min_stack)
    DEALLOCATE(MagBranch_first)
    NULLIFY(MagBranch)
    EXISTENCE=.FALSE.
  END IF
END SUBROUTINE DEALLOCATE_MAGBRANCH
```

ANNEXE II

CODE SOURCE DU RÉGRESSEUR

```
%*****************************************************************************
%*  FitterAx.m                                                               *
%*  Copyright(c) Hydro-Quebec TransEnergie                                   *
%*                                                                           *
%*  Created: 2009-02-11 15:03:00                                             *
%*  Author : Mathieu Lambert                                                 *
%*  Last change: ML 2009-03-04 21:34:41                                      *
%*  Version: 1.2                                                             *
%*                                                                           *
%*  This is the curve fitter for the A(x) model used to generate the ModelData *
%*  attribute for the device                                                 *
%*****************************************************************************

clear
clc

simplified_mode=true; %To choose the simplified or advanced mode

%The initialization parameter: 1 means connected in SS
%                              2 means initialize from manual conditions
%                              3 means no initial conditions
init=1;
fluxinit=0.0; %Manual flux if init=2
nomvolt=126439.71; %Nomimal rms voltage to find Rss
n_stack=1000; %Maximum number of extrema in stack

%For simplified mode only
Phi_sat=400.0; %Saturation flux value
Lsat=0.0257;   %Air core inductance
dPhi_dI_0=1.5; %Initial inductance at coercivity
Coer=1.085;    %Coercive current

if (simplified_mode) %In simplified mode we use only K1,K2,K3,K13,K14,K15
    K1=Phi_sat;
    K2=dPhi_dI_0;
    K3=Coer;
    K4=0.0;
    K5=0.0;
    K6=0.0;
    K7=0.0;
    K8=0.0;
    K9=0.0;
    K10=0.0;
    K11=0.0;
    K12=0.0;
    K13=Lsat;

    K14=0.45;
    K15=K2;

    x=[-100*Coer:Coer/10:100*Coer]; %Current points
    x2=[10:-0.5:-10];
    x2=[-exp(x2) exp(x2)]; %Current points for the virgin curve

    %The ascending branch of the major loop
    flux_up=K1*tanh(K2*x-K3)-K1*K4*((sech(K2*x-K3)).^2)+...
            K5*tanh(K6*x-K7)-K5*K8*((sech(K6*x-K7)).^2)+...
            K9*tanh(K10*x-K11)-K9*K12*((sech(K10*x-K11)).^2)+K13*x;
    %The descending branch of the major loop
    flux_down=K1*tanh(K2*x+K3)+K1*K4*((sech(K2*x+K3)).^2)+...
              K5*tanh(K6*x+K7)+K5*K8*((sech(K6*x+K7)).^2)+...
              K9*tanh(K10*x+K11)+K9*K12*((sech(K10*x+K11)).^2)+K13*x;
    %The virgin curve
    flux_virgin=(K1*(tanh(K2*x2))+K5*(tanh(K6*x2))+K9*(tanh(K10*x2))+K13*x2).*...
                (1-2*K14*((sech(K15*x2)).^2));
```

```
    n_virgin=length(flux_virgin); %The number of points

    %The fitting result is plotted
    plot(x,flux_up,'-b',x,flux_down,'-b',x2,flux_virgin,'.b')
    xlabel('Current(A)')
    ylabel('Flux(Wb)')
    title('Fitting results for the simplified A(x) model')
else
    load fitting.mat %upward part of the major loop, x is current, y is flux
    xdata=fitting(:,1); %Current values, sorted from lowest to greatest
    ydata=fitting(:,2); %Flux values, sorted from lowest to greatest

    %The air core inductance
    sat_slope=(ydata(end)-ydata(end-1))/(xdata(end)-xdata(end-1));
    ymax=ydata(end); %The maximum flux is assumed to be at the end

    %All fitting options are included here
    s = fitoptions('Method','NonlinearLeastSquares',...
     'Lower',[0 0 0 -0.5 0 0 0 -0.5 0 0 0 -0.5 0.99*sat_slope],...
     'Upper',[Inf Inf Inf 0.5 Inf Inf Inf 0.5 Inf Inf Inf 0.5 1.01*sat_slope],...
  'Startpoint',[0.6*ymax 0.5 1 0.1 0.16*ymax 1 0.5 0.05 0.2*ymax 2 1 0.3 sat_slope],...
     'MaxIter',400000,...
     'MaxFunEvals',600000,...
     'TolFun',5E-8,...
     'TolX',1E-5,...
     'DiffMinChange',1E-12,...
     'DiffMaxChange',1E-8,...
     'Robust','LAR');

    %The fitting function is defined here
    g1 = fittype('K1*tanh(K2*xdata-K3)-K1*K4*((sech(K2*xdata-K3))^2)+K5*tanh(K6*xdata
                 -K7)-K5*K8*((sech(K6*xdata-K7))^2)+K9*tanh(K10*xdata-K11)-K9*K12
                 *((sech(K10*xdata-K11))^2)+K13*xdata',...
                 'coefficients',{'K1','K2','K3','K4','K5','K6','K7','K8','K9','K10',
                 'K11','K12','K13'},...
                 'independent','xdata',...
                 'options',s);

    %The first curve fitting is done here
    [cfun1,gof1]=fit(xdata,ydata,g1);

    [diff,index]=min(abs(ydata)); %To return the index of the coercivity (nearest)
    xdata_coer=xdata(index); %The coercivity
    %The second fit excludes data near coercivity to try to get a better fit
    mod_data=excludedata(xdata,ydata,'domain',[0.5*xdata_coer,0.9999*xdata_coer])...
            &excludedata(xdata,ydata,'domain',[1.0001*xdata_coer,2*xdata_coer]);
    xdata2=xdata(mod_data); %The modified xdata
    ydata2=ydata(mod_data); %The modified ydata

    %The second fit with the modified data
    g2 = fittype('K1*tanh(K2*xdata2-K3)-K1*K4*((sech(K2*xdata2-K3))^2)+K5*tanh(K6*xdata2
                 -K7)-K5*K8*((sech(K6*xdata2-K7))^2)+K9*tanh(K10*xdata2-K11)-K9*K12
                 *((sech(K10*xdata2-K11))^2)+K13*xdata2',...
                 'coefficients',{'K1','K2','K3','K4','K5','K6','K7','K8','K9','K10',
                 'K11','K12','K13'},...
                 'independent','xdata2',...
                 'options',s);
    %Second fit is performed here
    [cfun2,gof2]=fit(xdata2,ydata2,g2);

    %If the second fit is better, replace the parameters with the new ones
    if (gof2.adjrsquare>gof1.adjrsquare)
        K1=cfun2.K1;
        K2=cfun2.K2;
        K3=cfun2.K3;
        K4=cfun2.K4;
        K5=cfun2.K5;
        K6=cfun2.K6;
        K7=cfun2.K7;
        K8=cfun2.K8;
        K9=cfun2.K9;
        K10=cfun2.K10;
        K11=cfun2.K11;
        K12=cfun2.K12;
        K13=cfun2.K13;
    else %Else keep the old ones
        K1=cfun1.K1;
```

```
        K2=cfun1.K2;
        K3=cfun1.K3;
        K4=cfun1.K4;
        K5=cfun1.K5;
        K6=cfun1.K6;
        K7=cfun1.K7;
        K8=cfun1.K8;
        K9=cfun1.K9;
        K10=cfun1.K10;
        K11=cfun1.K11;
        K12=cfun1.K12;
        K13=cfun1.K13;
    end;

    %Parameters for the virgin curve
    K14=0.45;
    K15=max([K2 K6 K10])/2;

    x=[min(xdata):(max(xdata)-min(xdata))/50000:max(xdata)]; %Current points
    x2=[10:-0.5:-10];
    x2=[-exp(x2) exp(x2)]; %Current points for the virgin curve

    %The ascending branch of the major loop
    flux_up=K1*tanh(K2*x-K3)-K1*K4*((sech(K2*x-K3)).^2)+...
            K5*tanh(K6*x-K7)-K5*K8*((sech(K6*x-K7)).^2)+...
            K9*tanh(K10*x-K11)-K9*K12*((sech(K10*x-K11)).^2)+K13*x;
    %The descending branch of the major loop
    flux_down=K1*tanh(K2*x+K3)+K1*K4*((sech(K2*x+K3)).^2)+...
            K5*tanh(K6*x+K7)+K5*K8*((sech(K6*x+K7)).^2)+...
            K9*tanh(K10*x+K11)+K9*K12*((sech(K10*x+K11)).^2)+K13*x;
    %The virgin curve
    flux_virgin=(K1*(tanh(K2*x2))+K5*(tanh(K6*x2))+K9*(tanh(K10*x2))+K13*x2).*...
                (1-2*K14*((sech(K15*x2)).^2));
    n_virgin=length(flux_virgin); %Number of points

    %The fitting results are plotted, along with experimental data
    plot(x,flux_up,'-b',x,flux_down,'-b',x2,flux_virgin,'.b',xdata,ydata,'.r')
    xlabel('Current(A)')
    ylabel('Flux(Wb)')
    title('Fitting results for the A(x) model')
end;

%Modeldata contains all parameters needed in EMTP-RV
%Copy this string into the ModelData attribute of the device in EMTP-RV
modeldata=sprintf('NewMagBranch,\n%f %f %f %f\n%f %f %f %f\n%f %f %f %f\n%f %d %f\n%f
                  %d %d\n',...
   K1,K2,K3,K4,K5,K6,K7,K8,K9,K10,K11,K12,K13,init,fluxinit,nomvolt,n_stack,n_virgin+4);
for k=1:n_virgin %To save the points of the virgin curve for the stack
   modeldata=sprintf('%s%f %f\n',modeldata,x2(k),flux_virgin(k));
end
modeldata %Display the result
```