| | |
|---|---|
| **Titre:** Title: | Plateforme de tests pour les méthodes d'intégration numérique appliquées aux réseaux électriques |
| **Auteur:** Author: | Serigne Mouhamadou Seye |
| **Date:** | 2018 |
| **Type:** | Mémoire ou thèse / Dissertation or Thesis |
| **Référence:** Citation: | Seye, S. M. (2018). Plateforme de tests pour les méthodes d'intégration numérique appliquées aux réseaux électriques [Mémoire de maîtrise, École Polytechnique de Montréal]. PolyPublie. https://publications.polymtl.ca/3032/ |



| | |
|---|---|
| **URL de PolyPublie:** PolyPublie URL: | https://publications.polymtl.ca/3032/ |
| **Directeurs de recherche:** Advisors: | Jean Mahseredjian |
| **Programme:** Program: | génie électrique |

UNIVERSITÉ DE MONTRÉAL

# PLATEFORME DE TESTS POUR LES MÉTHODES D'INTÉGRATION NUMÉRIQUE APPLIQUÉES AUX RÉSEAUX ÉLECTRIQUES

SERIGNE MOUHAMADOU SEYE

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 2018

UNIVERSITÉ DE MONTRÉAL

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

Ce mémoire intitulé :

# PLATEFORME DE TESTS POUR LES MÉTHODES D'INTÉGRATION NUMÉRIQUE APPLIQUÉES AUX RÉSEAUX ÉLECTRIQUES

présenté par : SEYE Serigne Mouhamadou

en vue de l'obtention du diplôme de : Maîtrise ès Sciences Appliquées

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

M. KARIMI Houshang, Ph. D, président

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

M. SHESHYEKANI Keyhan, Ph. D, membre

# DÉDICACE

*Ce projet de recherche est dédié à feu Saër Fall, mon cher professeur de sciences physiques en Terminale, que le Paradis soit ta destination finale. Tu as toujours cru en moi. Même après mon départ, tu ne ratais jamais l'occasion de me rappeler qui je suis et ce que je dois accomplir. J'imagine encore nos conversations passionnées sur la théorie des cordes, ou encore l'excitation dans ta voix et ton rire, à la suite de l'épreuve de physique du concours général.*

*Tu es sans nul doute le meilleur professeur que j'ai jamais eu et tu le resteras pour toujours. Ton dévouement et le sacrifice pour tes élèves est admirable. Je te remercie de m'avoir rendu meilleur, de m'avoir donné une telle confiance, de m'avoir tant appris sur les sciences, sur la vie. Tu es en grande partie responsable de mon parcours.*

*Repose en paix !*

# REMERCIEMENTS

À toutes les personnes qui m'ont soutenu et encouragé durant mon parcours scolaire, ceux qui de près ou de loin m'ont donné confiance en moi et m'ont accordé la leur, je n'aurai jamais les mots pour vous remercier.

Je commence par adresser mes humbles remerciements et mon plus grand respect à mon directeur de recherche, professeur Jean Mahseredjian, pour votre disponibilité, votre soutien financier, et votre enseignement. La connaissance est un trésor inestimable et à vos côtés, j'ai beaucoup appris et évolué.

Je remercie mes collègues Ming Cai, Anton Stepanov et Xiaopeng Fu, qui m'ont soutenu et aidé à réaliser la plateforme par leur conseils avisés et leur apport dans mon raisonnement mathématique.

Je remercie mes examinateurs pour avoir accepté d'allouer du temps estival à la lecture et l'évaluation de ce travail. Vos suggestions et commentaires sont appréciés.

Je remercie mon frère et mes sœurs de la confiance qu'ils m'ont accordée, et d'être toujours là quand je peine à voir le bout du tunnel.

Je remercie ma femme, Mame Diarra, pour sa patience et sa compréhension de la distance entre nous. C'est difficile, mais tu as toujours été là, pour me soutenir, m'épauler, me compléter et me rendre heureux. Je te remercie du fond du cœur, moitié.

Enfin, je remercie mes parents. Votre priorité à toujours été notre bien-être et notre réussite, vous avez mis à ma disposition tout ce dont un enfant peut rêver pour réussir. Je n'y serai jamais arrivé sans vous, tout le crédit vous revient. Maman, Papa, je vous remercie du fond du cœur.

# RÉSUMÉ

La recherche dans la simulation des réseaux électriques est en plein essor. Les logiciels de simulation ne cessent d'être améliorés afin d'augmenter la précision, la rapidité et la stabilité des simulations. Les méthodes d'intégration numérique jouent un rôle essentiel dans ce cadre. En effet, elles définissent la précision et la stabilité de la simulation. Il est donc important de bien déterminer la méthode à utiliser. Aujourd'hui, la méthode la plus répandue est la méthode trapézoïdale. Elle est simple, facile à coder et précise. Mais elle présente des problèmes de stabilité lorsqu'il y a des discontinuités.

L'objectif premier de ce projet est d'étudier plusieurs méthodes d'intégration numérique afin de déterminer la meilleure solution en termes de précision, de stabilité et de rapidité. Parallèlement, afin d'augmenter la rapidité des simulations de façon générale, nous nous penchons sur l'implémentation d'un algorithme d'intégration numérique à pas variable utilisant la méthode trapézoïdale.

La méthode trapézoïdale présente des oscillations numériques lorsqu'il y a une discontinuité. Pour éviter cela, des logiciels comme EMTP basculent vers la méthode d'Euler régressive dès qu'une discontinuité est détectée puis repassent à la méthode trapézoïdale après deux pas de temps. L'inconvénient de cette solution, c'est la complexité mais aussi la précision faible de la méthode d'Euler régressive qui peut nuire à la précision globale de la simulation. L'enjeu de ce projet est donc de trouver une méthode d'intégration au moins aussi précise que la méthode trapézoïdale et qui est stable durant les discontinuités.

D'abord, nous avons développé une plateforme de simulation avec le logiciel Matlab, qui intègre les modèles tels que les résistances, les inductances, les capacitances, les lignes, les transformateurs, les inductances non-linéaires, etc. La plateforme effectue une simulation dans le domaine temporel, il est donc nécessaire de discrétiser les équations courant-tension des composants tels que les inductances ou les capacitances. La méthode d'intégration numérique intervient à ce niveau. Quatre méthodes d'intégration ont été implémentées : trapézoïdale, Euler régressive, Gear 2$^{ème}$ ordre et Two-Stage Diagonnally Implicit Runge-Kutta (2S-DIRK). Afin de solutionner les équations du réseau à chaque pas de temps, on utilise l'analyse nodale modifiée augmentée, qui permet de trouver les tensions aux nœuds du réseau. Une telle plateforme est très utile pour réaliser des tests car il est facile d'y implémenter de nouveaux modèles ou méthodes.

Les méthodes d'intégration Gear et 2S-DIRK ont, tout comme la méthode trapézoïdale, une précision de deuxième ordre et sont stables durant les discontinuités. Nous avons donc développé des benchmarks afin d'étudier et de comparer toutes ces méthodes.

Dans un premier temps, l'analyse des résultats a permis de conclure que la méthode d'Euler régressive a une précision trop faible (précision de premier ordre) en comparaison aux méthodes trapézoïdale, Gear et 2S-DIRK. La méthode de Gear présente des résultats convenables en termes de stabilité mais sa rapidité ne rentre pas dans les critères fixés (110% de la durée de la simulation avec la méthode trapézoïdale) et l'amortissement des hautes fréquences nuit à sa précision. La méthode 2S-DIRK est très précise mais la résolution se fait en deux étapes ce qui nuit à la rapidité. En outre, la méthode 2S-DIRK a aussi tendance à amortir les hautes fréquences (dans une moindre mesure que Gear), ce qui réduit sa précision dans ce cas.

Ces méthodes n'étant pas satisfaisantes, nous avons développé une combinaison de la méthode trapézoïdale et la méthode 2S-DIRK. L'idée est d'utiliser la méthode trapézoïdale, puis en cas de discontinuité, au lieu de basculer vers Euler régressive, utiliser plutôt 2S-DIRK qui est beaucoup plus précise. Cela permettra de gagner en précision tout en conservant quasiment la même rapidité.

Les résultats obtenus de cette dernière analyse montrent que sur la plupart des circuits testés, la combinaison TRAP et 2S-DIRK augmente la précision de la simulation. Cependant, dans certains cas, 2S-DIRK n'arrive pas à éliminer totalement les oscillations numériques de la méthode trapézoïdale si le pas de temps n'est pas assez faible. Grâce à sa meilleure précision notamment sur des circuits non-linéaires, la méthode trapézoïdale associée à 2S-DIRK présente un léger avantage vis à vis des autres méthodes.

Dans la deuxième partie de ce projet, nous avons développé deux algorithmes d'intégration numérique à pas de temps variable, basées sur la méthode trapézoïdale. Le premier algorithme permet de spécifier un pas de temps différent sur plusieurs intervalles de temps à déterminer. Il est donc possible de choisir, si on connait les régimes de la simulation, le pas de temps que l'on veut utiliser pour chaque régime. Les tests réalisés ont permis de démontrer que l'on peut augmenter la rapidité tout en gardant la même précision.

Le deuxième algorithme permet de déterminer le pas de temps à utiliser en fonction de l'erreur de troncature et de tolérances prédéterminées. Dans ce cas précis, on utilise des tolérances qui ont été déterminées de façon empirique et un nombre fini de pas de temps. Cela a pour avantage de

permettre de calculer les matrices d'admittance relatives aux pas de temps et de les stocker avant le début de la simulation. Ainsi on évite de refactoriser la matrice des équations de réseau à chaque pas de temps comme le font la plupart des logiciels utilisant un pas de temps variable. Les résultats des tests réalisés montrent une grande augmentation de la rapidité des simulations (jusqu'à presque 20 fois dans certains cas) pour une précision très proche.

# ABSTRACT

The recent integration of power electronic converter-based devices such as wind generation, HVDC systems, and photovoltaic (PV) into power systems has introduced significant challenges to power system simulation due to the complexity of modeling these components. There is a growing need for more efficient power system simulation tools. Numerical integration methods play an important role in simulation accuracy and stability. The main goal of this project is to improve the efficiency of power system simulation by developing more efficient integration methods and solution techniques. Specifically, this thesis tackles two main challenges: (i) to improve the accuracy, stability, and speed of existing integration methods; and (ii) to enhance simulation efficiency by developing a variable time-step integration scheme.

Trapezoidal method is the most widely-used integration method because of its second order accuracy and simplicity. The main issue with the trapezoidal method is that it may become unstable when a discontinuity occurs. To overcome this problem, the most common solution is to temporarily switch to the backward Euler (BE) method following a discontinuity. However, due to the first order accuracy of BE, the trapezoidal/BE method may not be sufficiently accurate under repeated discontinuities. To address this challenge, the first part of this thesis proposes a more efficient integration method compared to trapezoidal/BE. To that end, the thesis considers two integration methods, namely, Gear $2^{nd}$ order and 2 Stage Diagnonally Implicit Runge-Kutta (2S-DIRK), and compares their performance in terms of accuracy and simulation speed. The results suggest that a combination of trapezoidal and 2S-DIRK provides a more superior performance compared to the trapezoidal/BE method. However, in some cases, 2S-DIRK was unable to fully eliminate the numerical oscillations resulting from transition from trapezoidal to 2S-DIRK. Nevertheless, due to its superior accuracy specially in regard to non-linear circuits, the proposed trapezoidal/2S-DIRK method seems to have a slight advantage.

Most of the existing simulation algorithms use a fixed time step which is determined based on the highest frequency component of the simulated phenomenon. The representation of a high frequency transient requires a small time-step, even when the high frequencies occurs for a short period of time. However, such a small time-step, which may make the simulation computationally demanding, is not necessary for most of the simulation time. The second part of this thesis proposes a variable time-step integration method which allows changing simulation time-step according to

the phenomenon being simulated at a given time. Tow algorithms have been proposed. The first algorithm allows the user to a priori select different time-steps for different periods of a simulation (e.g., a larger time-steps for a steady state period and a smaller time-step for a transient event). This approach increases simulation speed by using larger time steps during slow events while preserving simulation accuracy. The second proposed algorithm automatically adjusts the time-step during a simulation according to the local truncation error (LTE). To that end, the algorithm uses a look-up table containing a finite set of pre-determined time-steps and empirical tolerances and selects the appropriate time-step by comparing the LTE to the tolerance values. The assumption of a finite number of time-steps is to increase simulation speed through the computation of the corresponding admittance matrixes before the start of the simulation. Simulation results show that this approach can significantly increase simulation speed (up to 20 times in some cases) compared to the existing fixed time-step method.

# TABLE DES MATIÈRES

# LISTE DES TABLEAUX

# LISTE DES FIGURES

# LISTE DES SIGLES ET ABRÉVIATIONS

2S-DIRK : 2 Stage Diagonally Implicit Runge-Kutta

ER : Euler Régressive

GEAR : Gear 2$^{ème}$ ordre

LTE : Erreur de Troncature Locale (*Local Truncation Error* en anglais)

TRAP : Trapézoïdale

TRAP_DIRK : Trapézoïdale associée à 2S-DIRK

TRAP_ER : Trapézoïdale associée à Euler régressive

# CHAPITRE 1 INTRODUCTION

Le projet s'insère dans le cadre de la chaire industrielle de recherche : Simulation multi-échelle des temps des transitoires dans les réseaux électriques de grandes dimensions, dirigé par le professeur Jean Mahseredjian. Les partenaires de cette chaire de recherche sont EDF (Électricité de France), Hydro-Québec, Opal-RT, RTE (Réseau de transport d'électricité), CRNSG, et Polytechnique Montréal. L'objectif de cette chaire est entre autres de développer de nouvelles méthodes performantes pour la simulation des grands réseaux électriques.

Le monde de l'électricité est en constant changement. Cette mutation s'effectue de plus en plus rapidement de nos jours. Qu'il s'agisse des enjeux économiques ou écologiques ou bien de l'évolution technologique pour s'adapter aux exigences de qualité et à la multiplication des sources d'énergie, la simulation joue un rôle fondamental dans la conception et la caractérisation des réseaux électriques [1].

La Figure 1.1 décrit les différents modules d'un programme de simulation. Le module 'Régime permanent' permet de caractériser le réseau dans ces conditions normales de fonctionnement (il peut aussi permettre d'initialiser les calculs en régime transitoire). Le module 'Domaine du temps' permet d'étudier les transitoires, pendant le passage d'un régime permanent à un autre [2].

Pour effectuer un calcul de réseaux électriques, deux étapes importantes sont la modélisation et la simulation. La modélisation consiste à traduire le comportement électrique, magnétique et mécanique de chaque composant mais aussi les interconnexions du réseau par des équations [1]. Dépendant du phénomène étudié, de la taille du réseau étudié, des données et des méthodes mathématiques disponibles, il existe plusieurs modèles. Il est presque impossible de construire un modèle de tous les composants pour toutes les fréquences possibles [3]. La simulation consiste à résoudre simultanément toutes les équations du modèle. Deux méthodes principales se définissent : les équations d'état et l'analyse nodale (basée sur la loi de Kirchhoff ou loi des nœuds). Cette dernière a l'avantage d'être plus facile à formuler de façon automatique lorsque le réseau devient très complexe [2].

Dans ce projet, l'analyse nodale est utilisée afin de déterminer les formes d'ondes des transitoires électromagnétiques. Le modèle de chaque composant est ainsi intégré dans la matrice d'analyse nodale. S'il s'agit d'un composant linéaire, l'équation différentielle est discrétisée en utilisant une

méthode d'intégration numérique. Pour les composants non-linéaires, des méthodes de type Newton permettent de résoudre l'équation non-linéaire (souvent, il s'agit de représenter le modèle comme celui d'un composant linéaire par segment [4].

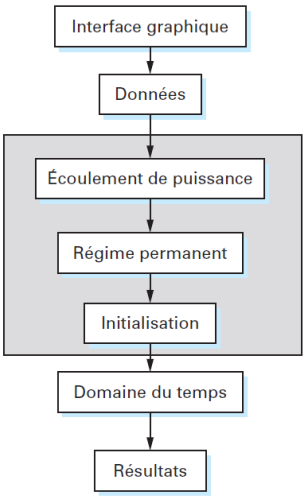


Figure 1.1 : Modules d'un programme de simulation de réseaux électriques

## 1.1 État de l'art des différentes méthodes d'intégration numérique

Les méthodes d'intégration numériques peuvent être classées dans deux catégories : les méthodes explicites et les méthodes implicites. Les méthodes explicites ont l'avantage d'être plus faciles à formuler et moins complexes. Cependant, ces méthodes présentent une stabilité très faible ce qui limite considérablement la gamme du pas de temps [5].

La plupart des réseaux électriques, de par leur large gamme de fréquences, présente des équations différentielles nécessitant une méthode d'intégration A-stable [6]. Une méthode d'intégration est dite A-stable lorsque la solution approximée est stable si la solution exacte l'est, peu importe le pas de temps utilisé [7]. Les méthodes implicites présentent une meilleure stabilité, raison pour laquelle elles sont implémentées dans la plupart des logiciels de simulation de réseaux électriques.

Les méthodes polynomiales, directement dérivées du développement de Taylor, sont les premières à être abordées. Les méthodes d'Euler régressive et trapézoïdale sont très vite implémentées à cause de leur simplicité. Les deux méthodes sont A-stables et relativement rapides. La méthode trapézoïdale admet cependant une meilleure précision et est donc plus utilisée. Elle est implémentée dans EMTP [8] et dans SPICE2 [5].

Une autre méthode polynomiale est la méthode de Gear. Il s'agit d'une méthode présentant plusieurs ordres et qui peut généralement être implémentée jusqu'au sixième. L'ordre 1 de la méthode de Gear correspond à Euler régressive. Plus l'ordre est élevé, plus la méthode est précise mais elle perd en stabilité par la même occasion [5]. En outre, il s'agit d'une méthode à pas multiples (au-delà de l'ordre un), ce qui signifie que des efforts de calcul et de stockage supplémentaires sont nécessaires (il y a plus d'historiques). Aussi, l'initialisation de la méthode est réalisée en utilisant les ordres inférieurs, ce qui rajoute à la complexité du code, surtout quand l'ordre est élevé. Ainsi, le gain réalisé au niveau de la précision (Gear d'ordre 3 ou plus) vis-à-vis de la méthode trapézoïdale, est souvent perdu du point de vue de la rapidité des simulations [6].

Les méthodes de Runge-Kutta implicites admettent aussi plusieurs ordres qui suivent les mêmes règles que les méthodes de Gear. Plus l'ordre de la méthode est élevé, plus elle est précise et moins elle est stable. De façon générale, les méthodes de Runge-Kutta présentent une erreur de troncature qui est plus petite que les méthodes de Gear du même ordre et la méthode trapézoïdale. L'inconvénient de ces méthodes est qu'elles impliquent un ou des pas intermédiaires qu'il faut calculer à chaque itération. Cela rend le calcul plus complexe et la simulation presque 2 fois plus lente que trapézoïdale pour les méthodes de Runge-Kutta d'ordre 2 [5].

Dans la littérature, l'approche la plus répandue de résolution est la méthode **d'intégration trapézoïdale**. En effet, elle est A-stable, et admet une bonne précision lorsque le pas de temps est bien choisi (précision de deuxième ordre) [9]. Cependant, en cas de discontinuité d'une variable d'état (manœuvre d'un interrupteur, changement de pente d'un composant linéaire par segment), des oscillations numériques se produisent. Pour corriger cela, la procédure la plus courante est de passer temporairement à la méthode d'**Euler régressive** qui ne présente pas d'oscillations numériques [10].

L'inconvénient de cette procédure est que la méthode d'Euler est moins précise que la méthode trapézoïdale (précision de premier ordre). Donc si la simulation présente beaucoup de discontinuités (par la présence de beaucoup de composants non-linéaires par exemples), la précision de la simulation en pâtit. Aussi, le passage vers Euler ajoute une complexité de calcul plus importante ainsi qu'une complexité accrue dans le code. C'est donc naturel de se poser la question de savoir comment éviter le passage par la méthode d'Euler rétrograde en cas de discontinuités sans perdre en précision et en rapidité ?

Par ailleurs, afin de réduire le temps des simulations sans détériorer la qualité des ondes, plusieurs types d'algorithme d'intégration à pas variable ont été implémentés. Les plus courants de ces algorithmes, comme dans [5], se basent sur l'erreur de troncature locale afin de déterminer le pas de temps à utiliser. Un contrôle de l'erreur est ainsi réalisé tout au long de la simulation. La méthode trapézoïdale avec contrôle de l'erreur de troncature est par exemple implémentée dans SPICE2. À chaque pas, l'erreur de troncature maximale est évaluée et le pas de temps pour calculer la prochaine itération en est, conditionnellement, déduite. Cela suppose donc qu'à chaque pas, il faut recalculer la matrice des équations du réseau, car cette dernière dépend du pas de temps utilisé. Aussi, comme le souligne [6], le traitement des lignes nécessite le stockage d'historiques permettant de tenir en compte le délai de propagation. Ainsi, le tableau permettant de réaliser ce stockage devra être dynamique et changer de taille constamment. Tout cela entraîne un ralentissement de la simulation. Une alternative proposée dans [6] serait d'utiliser un nombre limité de pas de temps prédéfinis. Ainsi, les matrices peuvent être calculées en amont et la rapidité globale de la simulation ne sera pas affectée.

## 1.2 Objectifs

L'objectif de ce projet de recherche est dans un premier temps de trouver une méthode d'intégration numérique :

- A-stable et qui n'oscille pas en cas de discontinuité
- Au moins aussi précise que la méthode trapézoïdale
- Avec une rapidité similaire (ne dépasse pas 1.1 fois la durée de la simulation avec la méthode trapézoïdale associée à Euler).

Pour cela, nous développons une plateforme de simulation de réseaux électriques en utilisant le logiciel Matlab. Les méthodes d'intégration numérique telles que la méthode Two Stage Diagonally Implicit Runge-Kutta (2S-DIRK) et la méthode de Gear $2^{ème}$ ordre y sont implémentées (aussi bien que les méthodes trapézoïdale et Euler régressive). Nous les étudions et les comparons à la méthode trapézoïdale et d'Euler afin de déterminer la meilleure méthode en termes de précision, de stabilité et de rapidité.

Dans un deuxième temps, afin de réduire la durée globale des simulations, nous implémentons un algorithme de simulation à pas de temps variable, basé sur le contrôle de l'erreur de troncature

locale et un nombre limité de pas de temps prédéfinis. En effet, le pas de temps de la simulation est un facteur important de rapidité. Augmenter le pas de temps en régime permanent et le diminuer en régime transitoire constitue donc une bonne façon de gagner en rapidité sans détériorer la précision.

# CHAPITRE 2 CONCEPTION DE LA PLATEFORME

La plateforme de simulation est construite avec le logiciel Matlab. Le code de la plateforme est orienté objet avec des classes d'objets représentants les éléments du réseau tel que les résistances, les inductances, les lignes, etc. Chaque objet contient en attributs les caractéristiques de l'élément et les valeurs permettant de construire le réseau tels que les numéros de nœuds et la branche.

La liste des éléments du réseaux permet ainsi de construire la matrice d'admittance permettant de résoudre les équations du réseau. En effet la méthode de résolution utilisée est l'analyse nodale modifiée augmentée que nous allons voir un peu plus en détail dans ce chapitre. Auparavant, il faut trouver l'admittance de chaque élément dans le domaine temporel. En ce qui concerne les inductances et les capacitances, cela revient à discrétiser leurs équations différentielles caractéristiques. Enfin, nous verrons plus en détail la documentation de la plateforme.

## 2.1 Modèles Norton

Dans cette partie, il est démontré la discrétisation de l'inductance, de la capacitance et de l'inductance non-linéaire afin de déterminer l'admittance et le courant historique impliqués dans l'analyse nodale.

De façon générale, la discrétisation consiste à utiliser une méthode d'intégration numérique sur une équation différentielle afin de la résoudre pas à pas. En guise d'exemple nous allons utiliser la méthode d'Euler régressive. On parle de modèle Norton car pour chaque composant, la discrétisation mène à l'expression du courant comme étant la somme d'un courant historique et d'un courant à travers une résistance (voir Figure 2.1).

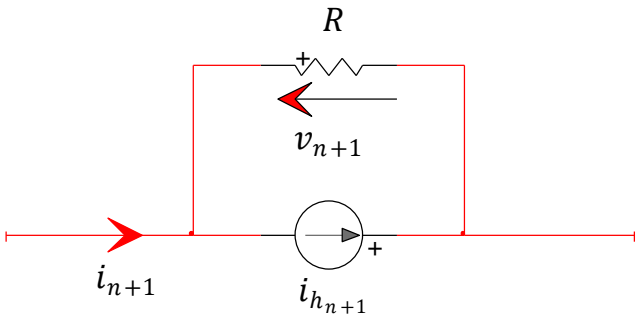


Figure 2.1 : discrétisation d'un élément linéaire

Dans ce qui va suivre, nous allons déterminer la valeur de la résistance et du courant historique pour chaque élément, dans la méthode d'Euler régressive. Afin de présenter le schéma de chaque méthode, nous allons utiliser l'équation générique :

$$\frac{dx}{dt} = f(x) \tag{1}$$

### 2.1.1 Inductance

Le schéma d'Euler régressive pour l'équation (1) s'énonce :

$$x_{n+1} = x_n + h_n f_{n+1} \tag{2}$$

Où $h_n$ est le pas de temps, $f_{n+1} = f(t_{n+1})$ et $x_n$ représente une approximation de $x$ à l'instant $t_n = n * h_n$. Pour garder une uniformité, le pas de temps est représenté avec un indice $n$. Dans la première partie de ce mémoire, le pas de temps utilisé est fixe. Cette représentation devient cependant essentielle lorsque les algorithmes à pas variable sont abordés.

L'équation caractéristique de l'inductance

$$v = L\frac{di}{dt} \tag{3}$$

est donc discrétisée pour donner :

$$i_{n+1} = \frac{h_n}{L} v_{n+1} + i_n \tag{4}$$

Par conséquent, dans la résolution dans le domaine temporel avec Euler régressive, l'inductance est remplacée au pas $n + 1$ par le schéma de la Figure 2.1, avec $\begin{cases} R = R_L = \frac{L}{h_n} \\ i_{h_{n+1}} = i_n \end{cases}$

Cette même procédure est appliquée pour les autres méthodes d'intégration numériques.

### 2.1.2 Capacitance

L'équation caractéristique de la capacitance

$$i = C\frac{dv}{dt} \tag{5}$$

est discrétisée avec Euler régressive pour donner :

$$i_{n+1} = \frac{C}{h_n} v_{n+1} + \frac{C}{h_n} v_n \tag{6}$$

Par conséquent, dans la résolution dans le domaine temporel avec Euler régressive, la capacitance est remplacée au pas $n+1$ par le schéma de la Figure 2.1, avec $\begin{cases} R = R_C = \frac{h_n}{C} \\ i_{h_{n+1}} = \frac{C}{h_n} v_n \end{cases}$

### 2.1.3 Inductance non-linéaire

L'inductance non-linéaire est modélisée comme un composant linéaire par segment. En d'autres termes, en fonction de la plage où se situe le flux (ou le courant), l'inductance ne sera pas la même (Figure 2.2).

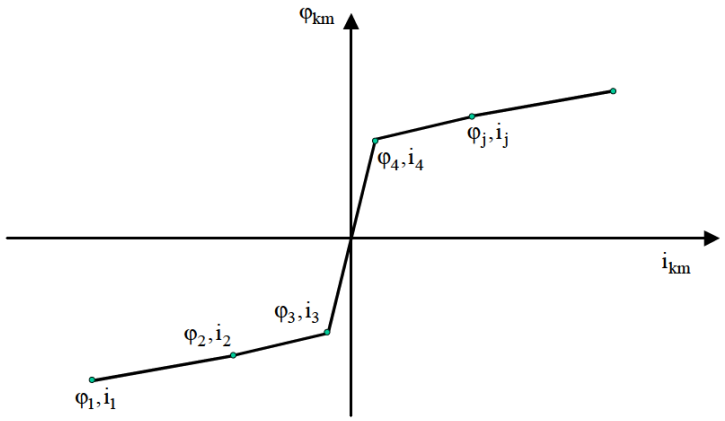


Figure 2.2 : Caractéristique flux-courant d'une inductance non-linéaire

L'équation du flux est représentée par segment et est donnée par :

$$\phi = L_k i + \phi_k \tag{7}$$

Où $L_k$ et $\phi_k$ dépendent du segment k.

Le flux est aussi relié à la tension par l'équation $v = \frac{d\phi}{dt}$.

Donc pour trouver l'équation de Norton, il suffit de trouver le segment sur lequel on se trouve (en calculant le courant ou le flux à travers l'inductance) puis de discrétiser l'équation flux-tension :

$$\phi_{n+1} = h_n v_{n+1} + \phi_n \tag{8}$$

En remplaçant cette expression du flux dans l'équation flux-courant, on trouve l'expression du courant au pas $n+1$ :

$$i_{n+1} = \frac{h_n}{L_k} v_{n+1} + \frac{1}{L_k}(\phi_n - \phi_k) \tag{9}$$

Par conséquent, dans la résolution dans le domaine temporel avec Euler régressive, l'inductance non-linéaire est remplacée au pas $n+1$ par le schéma de la Figure 2.1, avec $\begin{cases} R = R_{L_\phi} = \frac{L_k}{h_n} \\ i_{h_{n+1}} = \frac{1}{L_k}(\phi_n - \phi_k) \end{cases}$

La particularité avec les composants non-linéaire est qu'il faut à chaque pas $n+1$, itérer sur $k$ en utilisant une méthode telle que Newton pour converger. En ce qui concerne les composants linéaires par segment, la méthode de Newton s'exprime simplement : on calcule la solution au pas $n+1$ avec les valeurs du segment obtenu au pas $n$ ; les valeurs obtenues du courant ou du flux permettent de vérifier le segment sur lequel on se trouve. Si le segment a changé, on recommence les calculs avec les nouvelles valeurs de $L_k$ et $\phi_k$. Si le segment est le même, alors on a convergé.

## 2.2 Analyse Nodale Modifiée Augmentée (MANA)

### 2.2.1 Principe de l'analyse

L'analyse nodale modifiée augmentée est inspirée de l'analyse nodale classique. Elle est cependant beaucoup plus générique, représentant en plus des sources, les interrupteurs et les transformateurs dans la même équation matricielle [2]. Le système d'équations se présente sous la forme générique $\boldsymbol{Ax} = \boldsymbol{b}$ ou de façon plus symbolique :

$$\begin{bmatrix} \boldsymbol{Y_n} & \boldsymbol{V_c} & \boldsymbol{D_c} & \boldsymbol{S_c} \\ \boldsymbol{V_r} & \boldsymbol{V_d} & \boldsymbol{V_{VD}} & \boldsymbol{V_{VS}} \\ \boldsymbol{D_r} & \boldsymbol{D_{DV}} & \boldsymbol{D_d} & \boldsymbol{D_{DS}} \\ \boldsymbol{S_r} & \boldsymbol{S_{SV}} & \boldsymbol{S_{SD}} & \boldsymbol{S_d} \end{bmatrix} \begin{bmatrix} \boldsymbol{V_n} \\ \boldsymbol{I_V} \\ \boldsymbol{I_D} \\ \boldsymbol{I_S} \end{bmatrix} = \begin{bmatrix} \boldsymbol{I_n} \\ \boldsymbol{V_b} \\ \boldsymbol{D_b} \\ \boldsymbol{S_b} \end{bmatrix} \tag{10}$$

La partie $\boldsymbol{Y_n}$ de la matrice $\boldsymbol{A}$ représente la matrice des admittances nodales des éléments du réseau. Chaque terme $y_{ii}$ sur la diagonale de la matrice $\boldsymbol{Y_n}$ représente la somme des admittances au nœud $i$, et les termes $y_{ij}$ sont les sommes négatives des admittances entre les nœuds $i$ et $j$. $\boldsymbol{V_n}$ représente

le vecteur des tensions de nœud et $\mathbf{I_n}$ le vecteur des courants (courants historiques et sources de courant).

Si une source de tension est branchée entre les nœuds $k$ et $m$, alors l'équation de la source est donnée par : $V_k - V_m = V_s$.

Donc pour représenter cela dans la matrice d'admittance $\boldsymbol{A}$, en admettant que la source est le $q^{ème}$ éléments dans la liste des sources, on obtient :

$$\begin{aligned} \boldsymbol{V_r}(q,k) &= 1 \\ \boldsymbol{V_r}(q,m) &= -1 \\ \boldsymbol{V_b}(q) &= Vs \end{aligned} \tag{11}$$

Pour tenir compte des courants sortant des sources, on obtient en colonne :

$$\begin{aligned} \boldsymbol{V_c}(k,q) &= 1 \\ \boldsymbol{V_c}(m,q) &= -1 \end{aligned} \tag{12}$$

De la même manière, on peut aussi représenter les interrupteurs en respectant le même principe. L'équation d'un interrupteur fermé entre les nœuds k et m peut être représentée par : $V_k - V_m = 0$.

Dans ce cas, le $q^{ème}$ interrupteur dans la liste des interrupteurs est intégré dans la matrice d'admittance en mettant :

$$\begin{aligned} \boldsymbol{S_r}(q,k) &= 1 \\ \boldsymbol{S_r}(q,m) &= -1 \\ \boldsymbol{S_c}(k,q) &= 1 \\ \boldsymbol{S_c}(m,q) &= -1 \\ \boldsymbol{S_d}(q,q) &= 0 \end{aligned} \tag{13}$$

Dans le cas où l'interrupteur est ouvert, le courant à travers ce dernier (représenté par $I_s$) est nul.

$$\begin{aligned} \boldsymbol{S_r}(q,k) &= 0 \\ \boldsymbol{S_r}(q,m) &= 0 \\ \boldsymbol{S_c}(k,q) &= 0 \\ \boldsymbol{S_c}(m,q) &= 0 \\ \boldsymbol{S_d}(q,q) &= 1 \end{aligned} \tag{14}$$

Enfin, l'équation d'un transformateur idéal entre les nœuds k, m, i et j est donnée par :

$$V_k - V_m - g\left(V_i - V_j\right) = 0 \tag{15}$$

Donc de la même manière, il faut tenir compte de cette équation (à la ligne $q$ de $\boldsymbol{Dr}$ si le transformateur est le $q^{ème}$ dans la liste des transformateurs) mais aussi des courants mis en jeux par le transformateur aux nœuds $k, m, i$ et $j$. On obtient ainsi :

$$\begin{aligned} \boldsymbol{D_r}\,(q,k) &= 1 \\ \boldsymbol{D_r}\,(q,m) &= -1 \\ \boldsymbol{D_r}\,(q,i) &= -g \\ \boldsymbol{D_r}\,(q,j) &= g \end{aligned} \tag{16}$$

et,

$$\begin{aligned} \boldsymbol{D_v}\,(k,q) &= 1 \\ \boldsymbol{D_v}\,(m,q) &= -1 \\ \boldsymbol{D_v}\,(i,q) &= -g \\ \boldsymbol{D_v}\,(j,q) &= g \end{aligned} \tag{17}$$

### 2.2.2 Exemple

En guise d'exemple, nous déterminons la matrice d'admittance ***A*** du circuit de la Figure 2.3 mais aussi les vecteur ***x*** et ***b*** (exemple pris dans [11]). Pour cela, nous utilisons les modèles de Norton afin de déterminer les admittances des éléments linéaires impliqués et les courants historiques.

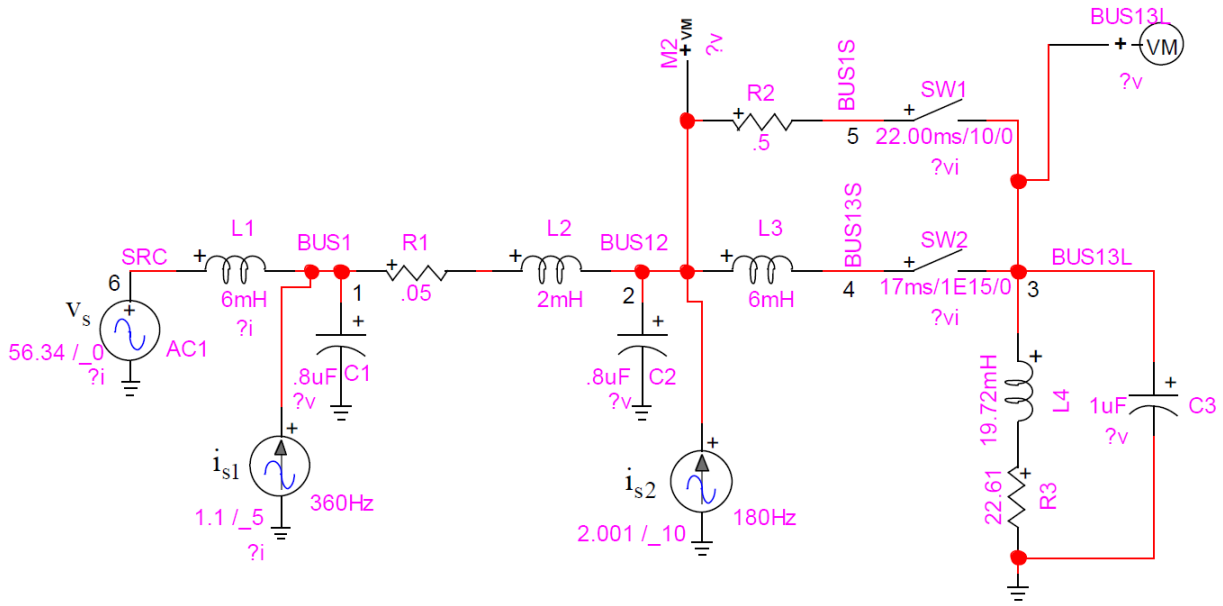


Figure 2.3 : Circuit test pour illustrer l'analyse nodale modifiée augmentée

Les équations dans le domaine temporel sont représentées de la façon suivante :

$$\begin{bmatrix} y_{11} & y_{12} & 0 & 0 & 0 & y_{16} & 0 & 0 & 0 \\ y_{21} & y_{22} & 0 & y_{24} & -2 & 0 & 0 & 0 & 0 \\ 0 & 0 & y_{33} & 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & y_{42} & 0 & y_{44} & 0 & 0 & 0 & 0 & 0 \\ 0 & -2 & 0 & 0 & 2 & 0 & 0 & 1 & 0 \\ y_{61} & 0 & 0 & 0 & 0 & y_{66} & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \\ v_6 \\ i_{v_s} \\ i_{SW1} \\ i_{SW2} \end{bmatrix} = \begin{bmatrix} i_{s1} + i_{h61} - i_{h12} - i_{h10} \\ i_{s2} + i_{h12} - i_{h20} - i_{h24} \\ -i_{h30_1} - i_{h30_2} \\ i_{h24} \\ 0 \\ -i_{h61} \\ v_s \\ 0 \\ 0 \end{bmatrix} \quad (18)$$

Si on utilise la méthode d'Euler régressive, on obtient les admittances suivantes:

Les courants historiques sont calculés comme nous l'avons vu précédemment en fonction des valeurs obtenues aux pas précédents. Dans ce cas de figure, la matrice d'admittance représentée révèle que le premier interrupteur est fermé et que le deuxième est ouvert.

$$y_{11} = \frac{h}{6e-03} + \frac{0.8e-06}{h} + \frac{1}{0.05 + \frac{2e-03}{h}}$$

$$y_{12} = -\frac{1}{0.05 + \frac{2e-03}{h}}$$

$$y_{16} = -\frac{h}{6e-03}$$

$$y_{22} = y_{11} + \frac{1}{0.5}$$

$$y_{21} = y_{12}$$

$$y_{24} = -\frac{h}{6e-03}$$

$$y_{33} = \frac{1e-06}{h} + \frac{1}{22.61 + \frac{19.72e-03}{h}}$$

$$y_{44} = -y_{24}$$

$$y_{66} = -y_{16}$$

$$y_{61} = y_{16}$$

(19)

## 2.3 Traitement des lignes

### 2.3.1 Modélisation

L'équation différentielle d'une ligne de transmission, comme l'expose [3], est obtenue en considérant un élément infinitésimal de la ligne de longueur $\Delta x$ (

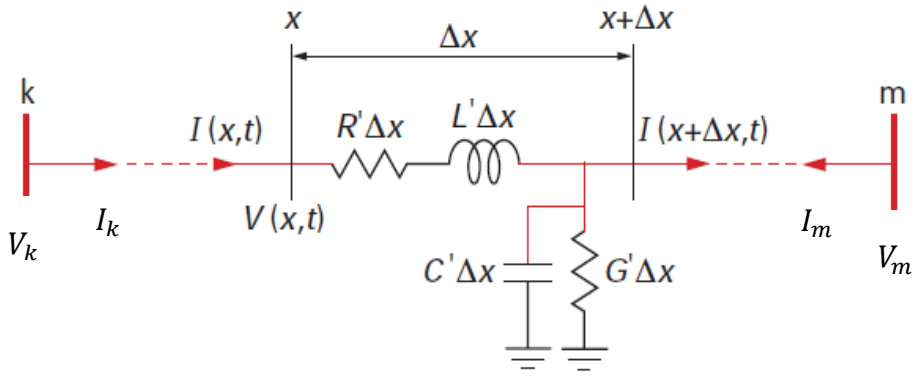


Figure 2.4). Lorsque $\Delta x \to 0$, on obtient les équations :

$$\frac{dV(x,t)}{dx} = -R'I(x,t) - L'\frac{dI(x,t)}{dt}$$
$$\frac{dI(x,t)}{dt} = -G'V(x,t) - C'\frac{dV(x,t)}{dt} \qquad (20)$$

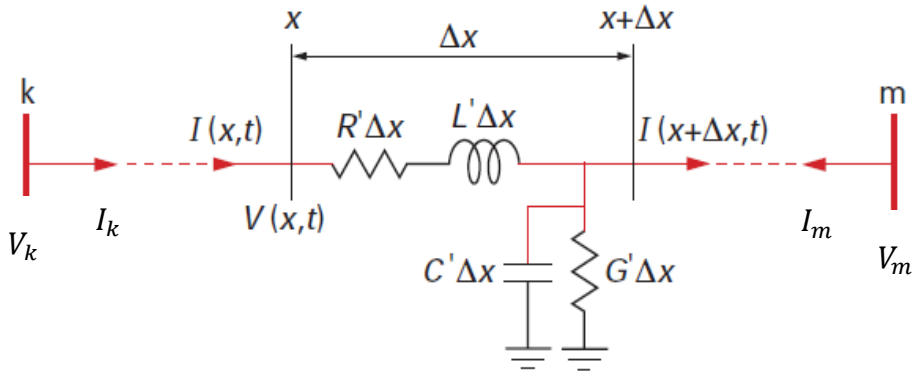


Figure 2.4 : Modélisation d'une ligne à paramètres distribués

En posant $Z' = R' + sL'$ et $Y' = G' + sC'$, la transformée de Laplace permet d'écrire les équations :

$$\frac{d^2V(x,s)}{dx} = Z'Y'V(x,s)$$
$$\frac{d^2I(x,s)}{dx} = Z'Y'I(x,s) \quad (21)$$

La solution générale de ces équations est donnée par :

$$V(x,s) = V^+e^{-\gamma x} + V^-e^{\gamma x}$$
$$I(x,s) = \frac{1}{Z_c}[V^+e^{-\gamma x} - V^-e^{\gamma x}] \quad (22)$$

où $\gamma = \sqrt{Z'Y'}$ et $Z_c = \sqrt{\frac{Z'}{Y'}}$.

On obtient les équations finales de $V_k$ et $V_m$ en appliquant les conditions aux limites :

$$V_k - Z_cI_k = [V_m + Z_cI_m]e^{-\gamma l}$$
$$V_k + Z_cI_k = [V_m - Z_cI_m]e^{\gamma l} \quad (23)$$

Avec $l$ la longueur de la ligne.

Afin d'obtenir les équations dans le domaine temporel, [11] néglige les pertes $R' = G' = 0$ (nous verrons comment inclure les pertes à la fin de cette partie). Ainsi, $\gamma = s\sqrt{L'C'}$, $Z_c = \sqrt{\frac{L'}{C'}}$ et les solutions $V(x,s)$ et $I(x,s)$ peuvent être transformées dans le domaine du temps pour obtenir :

$$v_k(t) - Z_ci_k(t) = v_m(t-\tau) + Z_ci_m(t-\tau)$$
$$v_m(t) - Z_ci_m(t) = v_k(t-\tau) + Z_ci_k(t-\tau) \quad (24)$$

Avec $\tau = l\sqrt{L'C'}$.

À partir de ces équations, on peut construire le circuit de la Figure 2.5, qui est l'équivalent de la ligne sans pertes dans l'analyse nodale, avec :

$$i_{k_h} = \frac{1}{Z_c} V_m(t-\tau) + i_m(t-\tau)$$
$$i_{m_h} = \frac{1}{Z_c} V_k(t-\tau) + i_k(t-\tau) \quad (25)$$

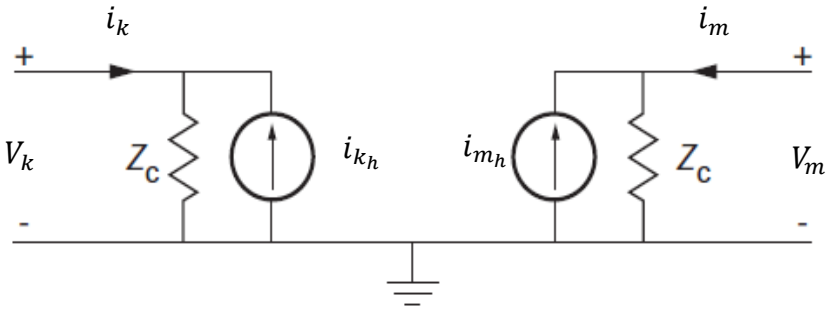


Figure 2.5 : Circuit équivalent de la ligne sans pertes

On constate qu'il y a un découplage dans le temps entre les nœuds k et m. Pour calculer les courants historiques, il faut disposer des valeurs à l'instant $t-\tau$. Si on pose $p = \frac{\tau}{h}$, dans la plupart des cas p est réel. Il faut donc interpoler à l'intérieur des pas $h$ afin de calculer les valeurs à $t-\tau$. Dans un tableau propre à chaque ligne, on stocke en permanence les $E(p)+1$ dernières valeurs des courants et tensions de la ligne, où $E$ est la fonction partie entière. On utilise un pointeur $q$ et à chaque pas, on interpole entre les valeurs $q-1$ et $q$ du tableau pour obtenir les valeurs à $t-\tau$ puis on met à jour les valeurs du tableau se trouvant au rang $q-1$.

Ce modèle fonctionne uniquement si $p \geq 1$ et sa précision augmente avec la réduction du pas de temps $h_n$.

Pour inclure les pertes dues à une résistance $R$, on divise la ligne en deux lignes sans pertes de délai de propagation $\frac{\tau}{2}$ et on insère une résistance de valeur $\frac{R}{4}$ de chaque côté de la ligne et une résistance de $\frac{R}{2}$ au milieu (Figure 2.6). Cette méthode ajoute donc un nœud supplémentaire et est valable seulement lorsque la résistance $R \ll Z_c$ [3].

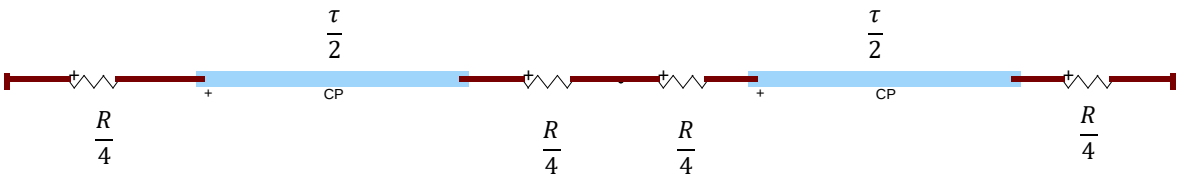


Figure 2.6 : Inclusion des pertes dans le modèle des lignes à paramètres distribués

### 2.3.2 Lignes triphasées équilibrées

Dans cette plateforme, seule les lignes triphasées équilibrées ont été implémentées. Lorsqu'il s'agit de lignes triphasées, le même model reste valable à cela près qu'on a aussi un couplage entre les phases (les nombres sont donc remplacés par des matrices et des vecteurs). Si la ligne est continuellement transposée, la matrice d'impédance dans le domaine fréquentiel est donnée par :

$$\boldsymbol{Z}' = \begin{bmatrix} Z'_S & Z'_M & Z'_M \\ Z'_M & Z'_S & Z'_M \\ Z'_M & Z'_M & Z'_S \end{bmatrix} \tag{26}$$

Comme on peut le voir, il existe des impédances mutuelles qui fait qu'on ne peut pas traiter indépendamment les équations différentielles des courants et tensions. Dans ce cas précis, la ligne est équilibrée et on peut passer au domaine modal afin de découpler ces équations. De façon générale, on utilise des matrices de transformation $T_V$ $et$ $T_I$ telles que :

$$\boldsymbol{V} = \boldsymbol{T_V}\hat{\boldsymbol{V}} \tag{27}$$

$$\boldsymbol{I} = \boldsymbol{T_I}\hat{\boldsymbol{I}} \tag{28}$$

En remplaçant ces expressions dans les équations (21) on obtient :

$$\begin{aligned} \frac{d^2\hat{\boldsymbol{V}}}{dx} &= \hat{\boldsymbol{Z}}\hat{\boldsymbol{Y}}\hat{\boldsymbol{V}} \\ \frac{d^2\hat{\boldsymbol{I}}}{dx} &= \hat{\boldsymbol{Y}}\hat{\boldsymbol{Z}}\hat{\boldsymbol{I}} \end{aligned} \tag{29}$$

Dans le cas d'une ligne équilibrée, on peut utiliser la matrice de transformation de Clarke :

$$\boldsymbol{T}_{\alpha\beta0} = \begin{bmatrix} \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{6}} \\ \frac{1}{\sqrt{3}} & -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{6}} \\ \frac{1}{\sqrt{3}} & 0 & -\frac{2}{\sqrt{6}} \end{bmatrix}$$

La matrice d'impédance modale est alors :

$$\hat{\boldsymbol{Z}}' = \boldsymbol{T}_{\alpha\beta0}^{t} \boldsymbol{Z}' \boldsymbol{T}_{\alpha\beta0} = \begin{bmatrix} Z_0' & 0 & 0 \\ 0 & Z_1' & 0 \\ 0 & 0 & Z_1' \end{bmatrix}$$

Comme on peut le constater, il y a deux modes distincts. Ainsi, dans le domaine modal, on a les équations découplées suivantes :

$$\widehat{\boldsymbol{I_k}} = \widehat{\boldsymbol{Y}}\widehat{\boldsymbol{V}}_{\boldsymbol{k}} - \hat{\boldsymbol{I}}_{\boldsymbol{kh}} \rightarrow \begin{bmatrix} i_{0k} \\ i_{1k} \\ i_{2k} \end{bmatrix} = \begin{bmatrix} \frac{1}{Z_0} & 0 & 0 \\ 0 & \frac{1}{Z_1} & 0 \\ 0 & 0 & \frac{1}{Z_1} \end{bmatrix} \begin{bmatrix} v_{0k} \\ v_{1k} \\ v_{2k} \end{bmatrix} - \begin{bmatrix} i_{0kh} \\ i_{1kh} \\ i_{2kh} \end{bmatrix} \quad (30)$$

Pour passer au domaine des phases, il suffit d'utiliser la matrice de transformation de Clarke $\mathrm{T}_{\alpha\beta0} = T$ :

$$\boldsymbol{I_k} = \left(\boldsymbol{T}\widehat{\boldsymbol{Y}}\boldsymbol{T^t}\right)\boldsymbol{V_k} - \boldsymbol{T}\hat{\boldsymbol{I}}_{\boldsymbol{kh}} \quad (31)$$

Ainsi, on met dans la matrice d'admittance les termes de la matrice $\boldsymbol{T}\widehat{\boldsymbol{Y}}\boldsymbol{T^t}$ en respectant bien les nœuds occupés par les différentes phases de la ligne triphasée.

Concernant les courants historiques, $\boldsymbol{I_{kh}} = \boldsymbol{T}\hat{\boldsymbol{I}}_{\boldsymbol{kh}}$. Il faut faire attention au fait qu'il y a deux modes donc deux délais de propagations. Dans notre plateforme, nous prenons cela en compte en considérant deux tableaux de stockage pour garder les $p_0$ et $p_1$ dernières valeurs des courants et tensions de la ligne avec $p_0 = \frac{\tau_1}{h}$ et $p_1 = \frac{\tau_1}{h}$. Pour l'historique du premier mode, on interpole en utilisant le premier délai de propagation, donc le premier tableau, puis le deuxième tableau pour le deuxième mode. Après avoir interpolé, il suffit donc de calculer les tensions et courants des différents modes en utilisant la matrice de Clarke transposée, puis on peut obtenir les historiques de courant au nœud k : $\boldsymbol{I_{kh}} = \boldsymbol{T}\hat{\boldsymbol{I}}_{\boldsymbol{kh}}$.

Nous avons ici démontré les équations et le raisonnement utilisé pour le nœud $k$; ces derniers restent valables pour le nœud $m$.

## 2.4 Documentation de la plateforme de simulation

La plateforme de simulation est codée sur Matlab. Elle réalise des simulations de réseaux électriques dans le domaine temporel. La méthode utilisée est l'analyse nodale modifiée augmentée qui nécessite la modélisation et la discrétisation des différents composants du réseau en utilisant une méthode d'intégration numérique.

Au total 6 méthodes d'intégration numérique à pas fixe sont implémentées dans la plateforme, et 2 méthode d'intégration trapézoïdale à pas variable. Les méthodes d'intégration à pas fixe regroupe les méthodes trapézoïdale, Euler, Gear 2$^{ème}$ ordre, 2S-DIRK (Two Stage Diagonally Implicit Runge Kutta), une composition de trapézoïdale et Euler et une composition de trapézoïdale et 2S-DIRK.

La plateforme inclue plusieurs modèles tels que les résistances, les inductances, les capacitances, les lignes, les transformateurs, etc.

Dans la suite, nous allons décrire les différentes parties de la plateforme de simulation en commençant par le format des données en entrée représentant les circuits simulés et les classes représentant les modèles des composants du réseau. Puis nous verrons les fonctions permettant de regrouper et de calculer les données de la simulation telle que la matrice d'admittance. Enfin, nous décriront la fonction de résolution, pour un pas fixe et un pas variable.

### 2.4.1 Données en entrée

La plateforme ne comporte pas d'interface graphique. Les données du réseau simulé sont donc fournies sous un format spécifique dans un fichier texte. Chaque ligne du fichier constitue une branche du réseau. Les modèles inclus dans la plateforme sont : les résistances, les inductances linéaires et non-linéaires, les capacitances, les interrupteurs, les sources de tensions, les sources de courant, les lignes, les transformateurs et les charges (modèle impédance constante).

Le premier élément de chaque ligne représente le type de composant. Ensuite il est précisé les nœuds de la branche. En triphasé (type commençant par le chiffre 3), il est simplement communiqué les nœuds de la phase a. Si par exemple une branche admet $k$ comme une de ses

bornes de la phase a, $k+1$ représente la borne de la phase b et $k+2$ celle de la phase c. Il faut donc tenir cela en compte pour numéroter les différents nœuds d'un réseau triphasé.

Après les nœuds, il est précisé les valeurs du composant qui sont essentielles à sa modélisation. Par exemple, pour un transformateur Yy, il est précisé le rapport de transformation, les inductances et résistances au primaire et au secondaire et si celles-ci sont modélisées, la résistance shunt et les données flux-courant de la branche d'inductance non linéaire caractérisant la saturation :

```
3131 k m a[=V2/V1] R1 L1 R2 L2 Rmag phi1 i1 phi2 i2 phi3 i3 ...
```

La Figure 2.8 représente un exemple de données d'un circuit triphasé.

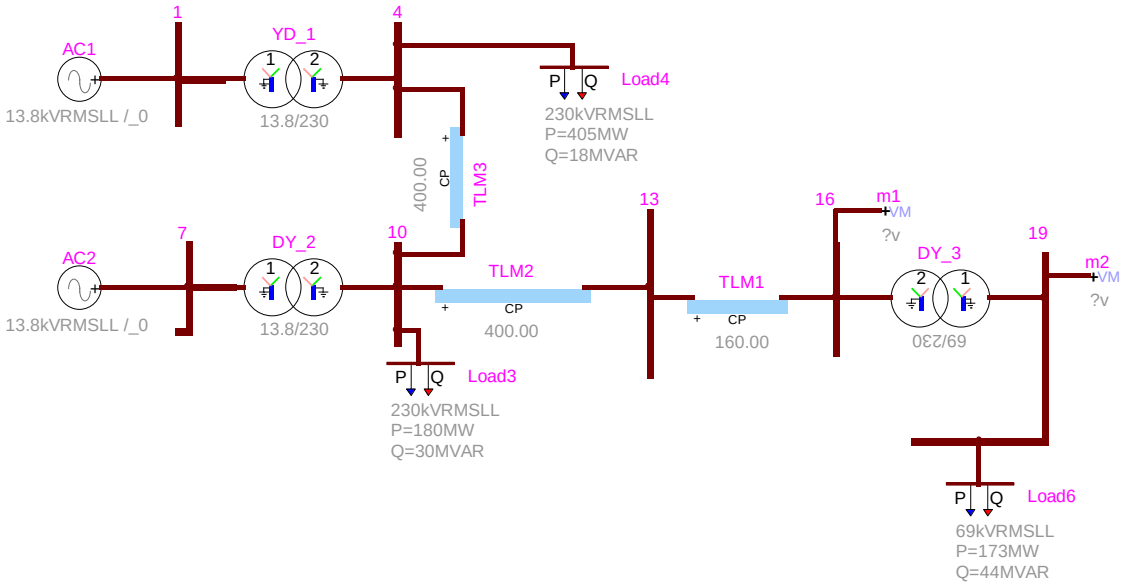


Figure 2.7 : Réseau triphasé

```
34 1 0 13.8e3*sqrt(2/3) 60 0
3131 1 4 230/13.8 0 0.0381/120/pi 0 0.3429/120/pi
312 4 2 405e6 18e6 230e3
34 7 0 13.8e3*sqrt(2/3) 60 0
3131 7 10 230/13.8 0 0.0381/120/pi 0 0.3429/120/pi
312 10 2 180e6 30e6 230e3
39 4 10 0.3 3e-3 0.008e-6 0.02 0.9e-3 0.0126e-6 400
39 10 13 0.3 3e-3 0.008e-6 0.02 0.9e-3 0.0126e-6 400
39 13 16 0.3 3e-3 0.008e-6 0.02 0.9e-3 0.0126e-6 160
3131 19 16 230/69 0 1.9044/120/pi 0 17.1393/120/pi
312 19 2 173e6 44e6 69e3
```

Figure 2.8 : Données représentant le réseau triphasé de la Figure 2.7

### 2.4.2 Classes des modèles

Le code de la plateforme est orienté objet. En d'autres termes, chaque modèle est représenté par une classe. Afin de pouvoir regrouper tous les composants dans une même liste, une classe ***Branche*** qui hérite de la classe ***matlab.mixin.Heterogeneous*** est elle-même héritée par toutes les autres classes. La classe ***Branche*** contient les attributs *nodes* (nœuds) et *value* (valeur) qui sont communs à toutes les classes.

La classe de chaque modèle permet, à partir des données fournies dans le fichier texte, de construire les éléments du réseau en leur attribuant toutes les caractéristiques spécifiées et de les regrouper dans une liste qui permettra de construire entre autres la matrice nodale et de réaliser la simulation.

En exemple, nous montrons ici les classes représentant l'inductance et le transformateur Yy (Figure 2.9 et Figure 2.10). La fonction ***donnees(filename)*** permet de construire ces objets et de les mettre dans la liste *Branches*. Chaque ligne est lue et interprétée en fonction du type du composant. Puis, un appel à la classe du composant est effectué.

```
%classe définissant une Inductance linéaire
classdef Inductor < Branche
    %% attributs
    properties (Constant) %non modifiable
        type = 2; %on attribue un type à chaque composant
    end
    properties
        y %admittance
        brancheNum %numéro de branche
    end
    %% Méthodes
    methods
        %Constructor
        function obj = Inductor(Nodes, Value, BrancheNum)
            obj.nodes = Nodes;
            obj.value = Value;
            obj.brancheNum = BrancheNum;
        end
    end
end
```

Figure 2.9 : Classe d'une inductance linéaire

Appel à la classe de l'inductance dans la fonction ***donnees(filename.)*** :

```
Branches(i) = Inductor([B(2) B(3)], B(4), i);
```

```
%classe définissant un transformateur triphasé YY
classdef TransfoYy < Branche
    %% attributs
    properties (Constant) %non modifiable
        type = 3131; %on attribue un type à chaque composant
    end
    properties
        brancheNum
    end
    %% Méthodes
    methods
        %Constructor
        function obj = TransfoYy(Nodes, Value, BrancheNum)
            obj.nodes = Nodes; %[k m]
            obj.value = Value; %[a=V2/V1 R1 L1 R2 L2 Rmag phi1 i1 phi2 i2 ...]
            obj.brancheNum = BrancheNum;
        end
    end
end
```

Figure 2.10 : Classe d'un transformateur triphasé Yy

Appel à la classe du transformateur triphasé Yy dans la fonction ***donnees(filename)*** :

```
Branches(i) = TransfoYy([B(2) B(3)], B(4:length(B)), i);
```

### 2.4.3 Fonctions de construction du réseau

Ces fonctions permettent de regrouper et de calculer les données nécessaires au déroulement de la simulation. Ces données dépendent donc de la méthode d'intégration numérique utilisée. Il s'agit par exemple de la matrice d'admittance nodale, des données relatives aux composants linéaires et non linéaires mais aussi de fonctions permettant de décomposer une ligne CP en deux demi lignes afin d'inclure les pertes ou alors de décomposer un transformateur en plusieurs branches constituées de résistances, d'inductances et d'un transformateur idéal. Cette dernière fonction, ***TransfoX(branches)***, est donnée en guise d'exemple à la Figure 2.11.

```
function branches = TransfosX(branches)
NONIDEALTRANSFO = 13; NONIDEALTRANSFO3W = 16;
N = length(branches);
n = nodes(branches);
v = 1; u = 1;
for i = 1:N
    if branches(i).type == NONIDEALTRANSFO
        transfo = branches(i);
        node = transfo.nodes; value = transfo.value; q = n+v; p = n+v+1;
        branches(i) = IdealTransfo([q node(2) p node(4)], value(1), i);
        branches(N+u) = NonIdealInductor([node(1) q], [value(2) value(3)], N+u); %inductance primaire
        branches(N+u+1) = NonIdealInductor([p node(3)], [value(4) value(5)], N+u+1); %inductance secondaire
        if length(value) > 6 %on n'exclue pas la branche magnétique
            branches(N+u+2) = NonLinearInductor([q node(2)], value(7:length(value)), N+u+2); %Branche shunt non linéaire
            branches(N+u+3) = Resistor([q node(2)], value(6), N+u+3);
            u = u+4;
        else
            u = u+2;
        end
        v = v+2;
```

Figure 2.11 : Fonction de décomposition d'un transformateur en plusieurs branches

### 2.4.4 Résolution

La fonction ***resolution(****METHODE, dt, T, Donnees, VoutNum****)*** est celle à laquelle on fait appel lorsqu'on veut démarrer une simulation avec un pas de temps fixe. Les différents arguments de la fonction sont :

- *METHODE* : il s'agit d'un nombre entier entre 1 et 6 qui spécifie la méthode d'intégration numérique à utiliser.

1 -> Trapézoïdale

2 -> Euler régressive

3 -> Trapézoïdale associée à Euler

4 -> Gear 2$^{ème}$ ordre

5 -> 2S-DIRK

6 -> Trapézoïdale associée à 2S-DIRK

- *dt* : il s'agit du pas de temps de la simulation
- *T* : représente la durée de la simulation
- *Donnees* : correspond au nom du fichier texte représentant les données du réseau électrique simulé.
- *VoutNum* : Tableau contenant les numéros des nœuds dont on veut mesurer la tension ou bien le courant aux bornes d'un interrupteur, d'une source de tension ou d'un transformateur.

Lorsqu'on effectue une simulation avec la méthode d'intégration trapézoïdale à pas variable, deux autres fonctions de résolution existent :

- ***resolutionLTEm(****dt, T, Donnees, VoutNum, TimeStep****)***
- ***resolutionLTE(****T, Donnees, VoutNum, Tolerance****)***.

Pour la première fonction s'ajoute aux arguments précédents (exceptée la méthode), *TimeStep.* Il s'agit d'un tableau qui précise le pas de temps à utiliser pour chaque intervalle de temps au cours de la simulation. La simulation commence donc avec un pas de temps *dt*, et chaque ligne du tableau *TimeStep* est sous la forme *[ti, dti, ni]*, qui s'interprète ainsi : lorsque *t* atteint *ti*, *dt* devient *dti*.

Pour la deuxième fonction, le changement de pas de temps est basé sur l'erreur de troncature. Le tableau *Tolerance*, spécifie donc des tolérances sur l'erreur de troncature locale associées à des pas de temps. En d'autres termes, sur chaque intervalle délimité par deux tolérances, est spécifié un pas de temps à utiliser.

La troisième colonne de TimeStep et de Tolerance représente le rapport entre le pas de temps de chaque ligne et le plus petit pas de temps. En effet, pour bien effectuer le traitement des lignes, il est impératif que les pas de temps soient toutes des multiples du plus petit pas de temps.

Lorsqu'on appelle la fonction de résolution, cette dernière utilise la fonction ***donnees(filename)*** afin de récupérer les données du réseau sous forme d'une liste d'objets, *Branches*. Cette liste permet ensuite de décomposer certains composants (lignes avec pertes, transformateurs, …). Puis, dépendant de la méthode d'intégration numérique, les données telle que la matrice d'admittance nodale sont calculées avant de faire appel à la méthode d'intégration numérique spécifiée.

Les différentes étapes de la résolution temporelle au sein d'une méthode d'intégration numériques sont spécifiées dans la prochaine partie.

En sortie, la fonction de résolution retourne une matrice constituée de $n + 1$ colonnes, $n$ étant le nombre de signaux mesurés. Les $n$ premières colonnes constituent les vecteurs de tension ou de courant mesurés tandis que la dernière colonne représente le vecteur du temps de la simulation. Pour les simulations à pas de temps variable, la matrice de sortie contient une colonne supplémentaire représentant le vecteur de l'erreur de troncature locale maximale au cours de la simulation.

Un appel à la fonction de résolution sur la console de MATLAB ressemble donc à :

```
V = resolution( 5, 1e-6, 0.05, 'test_1.txt', 19 );
```

qui signifie une simulation utilisant la méthode 2S-DIRK, avec un pas de temps fixe de $1\mu s$ du circuit représenté dans le fichier texte *'test_1.txt'*, et demandant en sortie la forme d'onde de la tension au nœud `19` (ou d'un courant à travers un switch, un transformateur ou une source de tension).

### 2.4.5 Méthodes de résolution

Quelle que soit la méthode utilisée, la résolution est constituée de deux étapes. L'initialisation et les itérations. L'initialisation est ici réalisée sans calcul du régime permanent ; tous les courants et les tensions sont donc nuls à l'instant initial.

L'itération comporte plusieurs étapes :

- D'abord on calcule le vecteur des tensions au pas suivant en utilisant la matrice nodale et les historiques.
- On itère ensuite sur l'ensemble des composants non linéaires jusqu'à ce qu'on atteigne la convergence ou que le nombre de tentatives maximal soit dépassé.
- La solution obtenue est ainsi stockée dans le vecteur de sortie (seulement les nœuds spécifiés dans *VoutNum*) ainsi que le temps de la simulation.
- Les interrupteurs sont traités afin de savoir s'ils sont opérés auquel cas la matrice nodale doit être modifiée (de suite s'il s'agit d'une fermeture, et pour une ouverture d'interrupteur, la modification de la matrice nodale est effectuée lorsque le courant à travers l'interrupteur passe par zéro).
- Les courants historiques sont ensuite mis à jour pour le prochain pas en parcourant les inductances (linéaires et non linéaires) et les capacitances, mais aussi les lignes de transmission pour lesquelles il faut mettre à jour les courants historiques et stocker les nouvelles tensions et courants obtenus.
- Enfin, on met à jour le vecteur des courants grâce aux sources de tension et de courant.

Ces étapes de l'itération (pour une intégration à pas fixe) sont répétées en boucle jusqu'à ce qu'on atteigne le temps de simulation final (*T).*

Pour ce qui concerne l'intégration à pas de temps variable, les étapes sont sensiblement les mêmes. Il faut ajouter à cela une étape de vérification en fonction des tableaux *TimeStep* ou *Tolerance,* si un changement de pas a lieu d'être, puis de l'effectuer en faisant bien attention à la mise à jour des courants historiques. Le Chapitre 4 présente en détail le principe des algorithmes d'intégration à pas variable.

# CHAPITRE 3 COMPARAISON DES MÉTHODES D'INTÉGRATION NUMÉRIQUE

Dans ce chapitre, nous faisons l'étude et la comparaison des méthodes d'intégrations numériques suivantes : trapézoïdale (TRAP), Euler régressive (ER), Gear 2ème ordre (GEAR) et 2 stage diagonally implicit Runge-Kutta (2S-DIRK).

## 3.1 Étude théorique

Il est démontré dans cette partie la construction des différentes méthodes citées ci-haut, mais aussi l'étude de leur précision et de leur stabilité à partir de leur schéma.

### 3.1.1 Schémas et erreurs de troncature locales

La plupart des schémas que nous allons étudier sont tirer de la formule de Taylor :

$$x(t_{n+1}) = x(t_n) + h_n x^{(1)}(t_n) + \frac{h_n^2}{2!}x^{(2)}(t_n) + \frac{h_n^3}{3!}x^{(3)}(t_n) + \cdots \tag{32}$$

#### 3.1.1.1 ER

Considérons l'équation générique (1), et la déclinaison du schéma d'ER (2). Appliquons la formule de Taylor (32) au premier ordre à la solution exacte $x$ en $t = t_n$ :

$$x(t_n) = x(t_{n+1}) - h_n x'(t_{n+1}) + \frac{h_n^2}{2}x''(\tau) \tag{33}$$

En remplaçant $x(t_n)$ par son approximation $x_n$ et la dérivée de $x$ par la fonction $f$ et en utilisant le schéma de Taylor, on obtient :

$$x(t_{n+1}) = x_n + h_n f_{n+1} - \frac{h_n^2}{2}x''(\tau) = x_{n+1} - \frac{h_n^2}{2}x''(\tau), \quad t_n < \tau < t_{n+1} \tag{34}$$

Donc l'erreur de troncature locale est donnée par :

$$\epsilon = x(t_{n+1}) - x_{n+1} = -\frac{h_n^2}{2}x''(\tau), \quad t_n < \tau < t_{n+1} \tag{35}$$

On voit donc que la méthode d'ER a une précision du premier ordre.

### 3.1.1.2 TRAP

Le schéma de TRAP se décline de la manière suivante :

$$x_{n+1} = x_n + \frac{h_n}{2}(f_{n+1} + f_n) \tag{36}$$

Appliquons la formule de Taylor au deuxième ordre à la solution exacte $x$ et à sa dérivée $\frac{dx}{dt}$ en $t = t_{n+1}$ :

$$x(t_{n+1}) = x(t_n) + h_n x'(t_n) + \frac{h_n^2}{2}x''(t_n) + \frac{h_n^3}{6}x^{(3)}(\tau_1) \tag{37}$$

$$x'(t_{n+1}) = x'(t_n) + h_n x''(t_n) + \frac{h_n^2}{2}x^{(3)}(\tau_2) \tag{38}$$

En multipliant (38) par $\frac{h_n}{2}$ et la soustrayant à (37), on obtient :

$$x(t_{n+1}) = x_n + \frac{h_n}{2}(f_n + f_{n+1}) - \frac{h_n^3}{12}\left[2x^{(3)}(\tau_1) - 3x^{(3)}(\tau_2)\right], \quad t_n < \tau < t_{n+1}$$

Pour simplifier cette expression, on peut approximer l'erreur de troncature de TRAP en confondant $\tau_1$ et $\tau_2$ :

$$\epsilon = x(t_{n+1}) - x_{n+1} = -\frac{h_n^3}{12}x^{(3)}(\tau), \quad t_n < \tau < t_{n+1} \tag{39}$$

Le schéma TRAP a une précision du 2ème ordre.

### 3.1.1.3 GEAR

Le schéma de Gear présente plusieurs ordres. Le premier ordre est le schéma d'ER. Pour atteindre les ordres plus élevés, on utilise les ordres inférieurs en guise d'initialisation. Cela implique donc plusieurs pas de temps à stocker et une grande complexité dans le code. Nous nous limiterons donc pour ces raisons à Gear 2ème ordre qui se décline de la manière suivante :

$$x_{n+1} = \frac{4}{3}x_n - \frac{1}{3}x_{n-1} + \frac{2}{3}h_n f_{n+1} \tag{40}$$

Des calculs menés dans [4] permettent d'établir une formule générique pour l'erreur de troncature locale des schémas de Gear. Pour Gear 2ème ordre, cette erreur est donnée par :

$$\epsilon = x(t_{n+1}) - x_{n+1} = -\frac{2}{9}h_n^3 x^{(3)}(\tau), \quad t_n < \tau < t_{n+1} \tag{41}$$

Le schéma GEAR a donc aussi une précision du 2ème ordre. On voit cependant qu'elle est un peu moins précise que TRAP dont le coefficient est $\frac{1}{12}$.

#### 3.1.1.4 2S-DIRK

Les schémas de Runge-Kutta sont aussi construits à partir du développement de Taylor. Il s'agit en fait d'une interpolation de la méthode de Taylor.

La méthode de Taylor est dérivée du développement en série de Taylor (32) en remplaçant les valeurs exactes de $x$ par ses valeurs approchées $x_n$ et $x_{n+1}$. On obtient ainsi,

$$x_{n+1} = x_n + h_n f(t_n) + \frac{h_n^2}{2!}f^{(1)}(t_n) + \frac{h_n^3}{3!}f^{(2)}(t_n) + \cdots + \frac{h_n^p}{p!}f^{(p-1)}(t_n) \tag{42}$$

représentant le schéma de Taylor à l'ordre $p$. Pour $p = 1$, on obtient le schéma d'Euler progressive. Pour $p = 2$, on a :

$$x_{n+1} = x_n + h_n f(t_n) + \frac{h_n^2}{2!}f^{(1)}(t_n) \tag{43}$$

C'est à partir de cette équation que les méthodes explicites de Runge-Kutta d'ordre 2 sont tirées.

Le schéma de Taylor présente des dérivées (dans d'autres cas des dérivées partielles) qui peuvent être difficile à évaluer voire induire des erreurs car dans la plupart des cas la fonction $f$ n'est pas connue explicitement (ce qui est notre cas). Ainsi sont nées les méthodes de Runge-Kutta qui remplacent les dérivées de $f$ par son interpolation en des points intermédiaires sans perdre en précision.

Les schémas de Runge-Kutta implicites sont obtenus en partant du schéma de Taylor implicite d'ordre 2, obtenu en développant cette fois-ci $x(t_n)$ à l'instant $t_{n+1}$ :

$$x_{n+1} = x_n + h_n\left[f(t_{n+1}) - \frac{h_n}{2}f^{(1)}(t_{n+1})\right] = x_n + T_2(t_{n+1}) \tag{44}$$

Les schémas de Runge-Kutta d'ordre 2 se présentent sous la forme générique :

$$\begin{aligned} k_1 &= f(t_n + c_1 h_n, x_n + h_n a_{11} k_1), \\ k_2 &= f\big(t_n + c_2 h_n, x_n + h_n(a_{21} k_1 + a_{22} k_2)\big) \\ x_{n+1} &= x_n + h_n(b_1 k_1 + b_2 k_2) \end{aligned} \tag{45}$$

Dans notre cas, on suppose que la fonction $f$ dépend d'une seule variable $t$ et 2S-DIRK étant implicite, $c_2 = 1$, ce qui donne une formule plus simple :

$$x_{n+1} = x_n + h_n[b_1 f(t_n + c_1 h_n) + b_2 f(t_{n+1})] \tag{46}$$

Pour trouver les constantes $c_1$, $b_1$ et $b_2$ à partir d'une équivalence avec la méthode de Taylor, faisons le développement limité de $f(t_n + ah)$ en $t_{n+1}$ :

$$f(t_n + c_1 h_n) = f(t_{n+1}) - (1 - c_1) h_n f^{(1)}(t_{n+1}) + O(h^2) \tag{47}$$

En remplaçant cette équation dans (46), on obtient :

$$x_{n+1} = x_n + h_n\big[(b_2 + b_1) f(t_{n+1}) - b_1(1 - c_1) h_n f^{(1)}(t_{n+1})\big] \tag{48}$$

Par identification avec le schéma de Taylor implicite du 2ème ordre (44), nous devons obtenir :

$$\begin{cases} b_2 + b_1 = 1 \\ b_1(1 - c_1) = \dfrac{1}{2} \end{cases} \tag{49}$$

Ce système présente un degré de liberté (2 équations et 3 inconnues) ce qui permet de déduire plusieurs types de schémas de Runge-Kutta implicites d'ordre 2. En ce qui concerne notre projet, on implémente le même schéma 2S-DIRK proposé dans [12], où on a posé $c_1 = b_2$, obtenant ainsi :

$$\begin{cases} c_1 = b_2 = 1 - \dfrac{1}{\sqrt{2}} \\ b_1 = \dfrac{1}{\sqrt{2}} \end{cases} \tag{50}$$

Pour simplifier la résolution, posons $a = c_1 = 1 - \frac{1}{\sqrt{2}}$ et $b_1 = a\beta$ où $\beta = 1 + \sqrt{2}$. En pratique, à cause de la valeur intermédiaire entre $t_n$ et $t_{n+1}$, le calcul se fait en deux étapes ressemblant à la méthode ER et une phase de conversion. Posons

$$a = 1 - \frac{1}{\sqrt{2}}; \quad \alpha = -\sqrt{2}; \quad \beta = 1 + \sqrt{2}; \quad \tilde{t}_n = t_{n-1} + ah_n \tag{51}$$

La première étape consiste à calculer $\tilde{x}_n$ qui est la valeur intermédiaire de $x$ :

$$\tilde{x}_n = x_{n-1} + ah_n\tilde{f}_n \tag{52}$$

La conversion permet de retrouver la forme de l'équation (46). Il s'agit de calculer l'historique permettant d'obtenir $x_n$ :

$$\tilde{x}_n^* = \alpha x_{n-1} + \beta\tilde{x}_n \tag{53}$$

Enfin la dernière étape permet de calculer la valeur de $x_n$ grâce à l'historique $\tilde{x}_n^*$ :

$$x_n = \tilde{x}_n^* + ah_n f_n \tag{54}$$

On constate donc que le coefficient multipliant la fonction $f$ est le même dans les deux étapes. Cela permet de conserver la même admittance (et donc la même matrice d'admittance nodale) durant la première et la deuxième étape. Les historiques de courant se calculent cependant différemment pour chaque étape.

L'erreur de troncature locale est donnée par [5] :

$$\epsilon = x(t_{n+1}) - x_{n+1} = -\frac{1}{24}h_n^3 x^{(3)}(\tau), \qquad t_n < \tau < t_{n+1} \tag{55}$$

Le schéma 2S-DIRK a donc aussi une précision du 2ème ordre et est deux fois plus précise que le schéma TRAP selon cette méthode d'analyse.

### 3.1.2 Stabilité

La stabilité étudiée ici concerne la stabilité générale de la méthode (A-stabilité) mais aussi, l'absence ou non d'oscillations numériques en cas de discontinuité.

Pour étudier la stabilité globale, nous allons utiliser l'équation différentielle suivante où $\lambda$ est une constante complexe [9] :

$$\frac{dx}{dt} = \lambda x \tag{56}$$

### 3.1.2.1 ER

La discrétisation de l'équation (56) donne : $x_{n+1} = x_n + \lambda h_n x_{n+1}$

$$x_{n+1} = \frac{1}{1 - \lambda h_n} x_n \tag{57}$$

La stabilité est vérifiée si et seulement si $\left|\frac{1}{1-\lambda h_n}\right| \leq 1$. La fonction $\phi(z) = \frac{1}{1-z}$ est appelée fonction de stabilité du schéma. Le graphe de cette fonction dans le plan complexe permet de définir la région de stabilité du schéma.

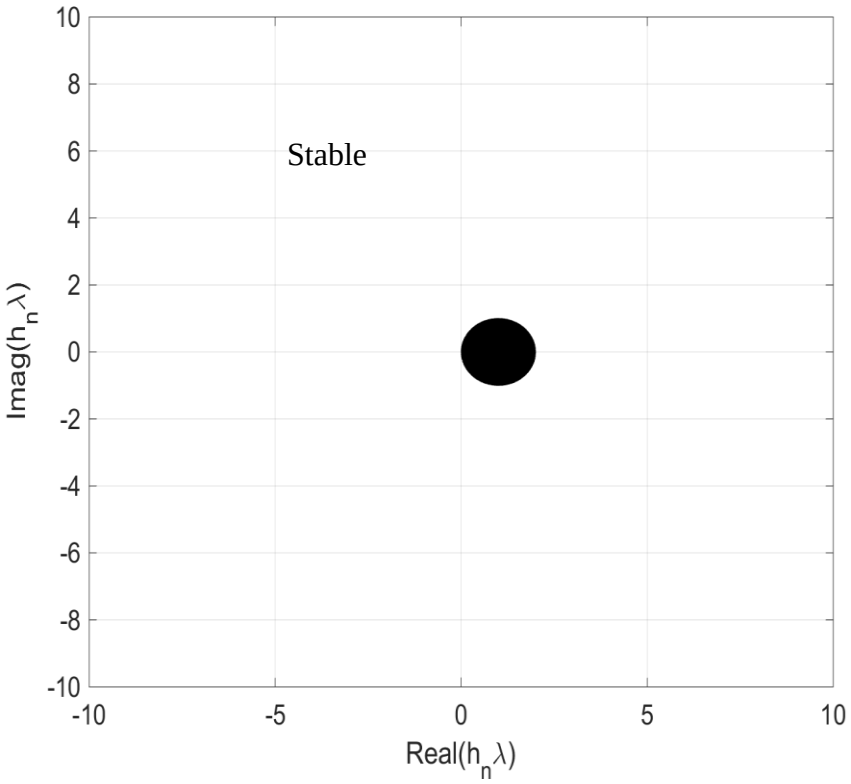


Figure 3.1 : Région de stabilité Euler Régressive – Stable en dehors de la région noire

Le schéma ER est A-stable (c'est-à-dire qu'elle est stable sur toute la partie gauche du plan complexe où Réel(z) < 0).

Pour étudier la présence d'oscillations numériques, considérons l'équation d'une inductance (3). À $t = t_{n+1}$, on impose la discontinuité $i = 0$. Observons le comportement de la tension $v$.

$$\begin{aligned} v_{n+1} &= \frac{L}{h_n}(i_{n+1} - i_n) = \frac{L}{h}(-i_n) \\ v_{n+2} &= \frac{L}{h_n}(i_{n+2} - i_{n+1}) = 0 \end{aligned} \tag{58}$$

Le schéma ER ne présente donc pas d'oscillations numériques en cas de discontinuité.

#### 3.1.2.2 TRAP

La discrétisation de l'équation (56) par le schéma de TRAP (36) donne :

$$x_{n+1} = \left(\frac{1 + \frac{\lambda h_n}{2}}{1 - \frac{\lambda h_n}{2}}\right) x_n \tag{59}$$

La stabilité est vérifiée si et seulement si $\left|\frac{1-\frac{\lambda h_n}{2}}{1+\frac{\lambda h_n}{2}}\right| \leq 1$.

$$\phi(z) = \frac{1 - \frac{z}{2}}{1 + \frac{z}{2}} \tag{60}$$

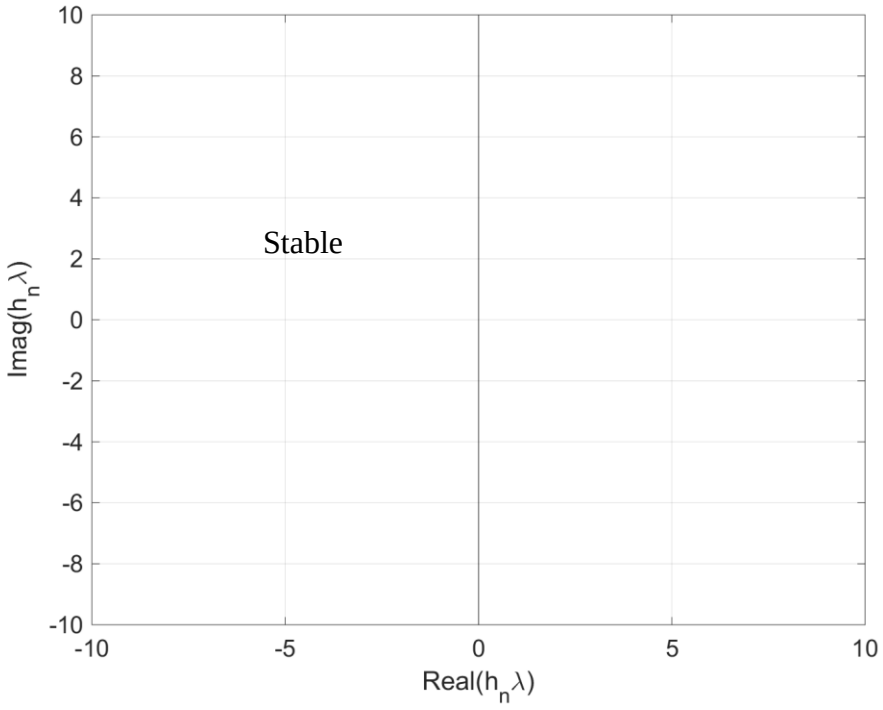


Figure 3.2 : Région de stabilité trapézoïdale – Stable à gauche du plan complexe

Le schéma TRAP est stable sur la partie gauche du plan complexe, elle est donc A-stable.

Considérons la même procédure que précédemment afin de vérifier si la méthode TRAP présente des oscillations numériques. Lorsqu'on annule le courant aux bornes d'une inductance, on obtient :

$$\begin{aligned} v_{n+1} &= \frac{2L}{h_n}(i_{n+1} - i_n) - v_n = \frac{2L}{h_n}(-i_n) - v_n \\ v_{n+2} &= -v_{n+1} \\ v_{n+3} &= -v_{n+2} = v_{n+1} \end{aligned} \tag{61}$$

Le schéma trapézoïdal présente donc des oscillations numériques en cas de discontinuité.

### 3.1.2.3 GEAR

Pour évaluer la stabilité, nous utilisons toujours l'équation (56).

Toujours dans [4], on démontre que la stabilité de Gear 2$^{ème}$ ordre est obtenue en dehors de la région délimitée par :

$$\sigma(\theta) = -\frac{3}{2} + 2e^{-j\theta} - \frac{1}{2}e^{-j2\theta} \tag{62}$$

où $\sigma$ représente le module de $\lambda h_n$ et $\theta$ son argument.

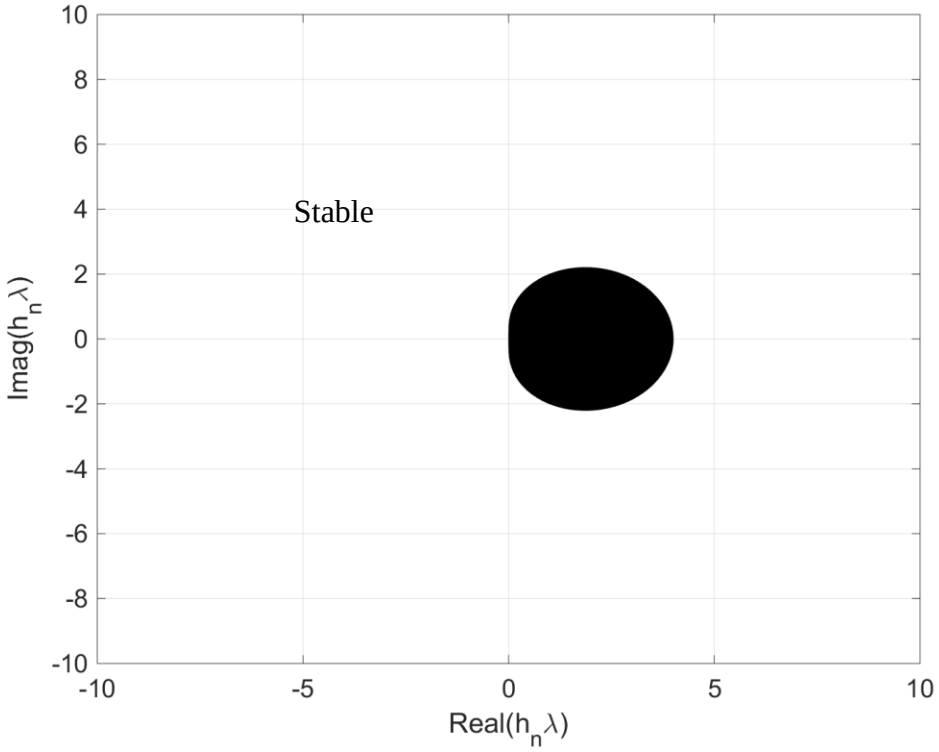


Figure 3.3 : Région de stabilité Gear 2ème ordre – Stable en dehors de la région noire

Le schéma GEAR est donc A-stable.

Lorsqu'on annule le courant aux bornes d'une inductance, on obtient avec le schéma (40) :

$$\begin{aligned} v_{n+1} &= \frac{3L}{2h_n}\left(i_{n+1} - \frac{4}{3}i_n + \frac{1}{3}i_{n-1}\right) = \frac{3L}{2h_n}\left(-\frac{4}{3}i_n + \frac{1}{3}i_{n-1}\right) \\ v_{n+2} &= \frac{3L}{2h_n}\left(\frac{1}{3}i_n\right) \\ v_{n+3} &= 0 \end{aligned} \tag{63}$$

Le schéma de Gear 2ème ordre ne présente pas d'oscillations numériques même si on atteint 0 après deux pas de temps.

#### 3.1.2.4 2S-DIRK

Comme il a été vu dans l'équation (46), la discrétisation de l'équation (56) donne :

$$x_{n+1} = x_n + ah_n\lambda x_{n+1} + a\beta h_n\lambda\tilde{x}_n. \tag{64}$$

$$\text{or } \tilde{x}_n = ah_n\lambda\tilde{x}_n + x_n = \frac{1}{1-ah_n\lambda}x_n \tag{65}$$

Donc :

$$x_{n+1} = \frac{1}{1-ah_n\lambda}\left[1 + \frac{a\beta h_n\lambda}{1-ah_n\lambda}\right]x_n \tag{66}$$

Donc le schéma est stable si $\left|\frac{1}{1-ah_n\lambda}\left[1 + \frac{a\beta h_n\lambda}{1-ah_n\lambda}\right]\right| \leq 1$.

$$\phi(z) = \frac{1}{1-ah_n\lambda}\left[1 + \frac{a\beta h_n\lambda}{1-ah_n\lambda}\right] \tag{67}$$

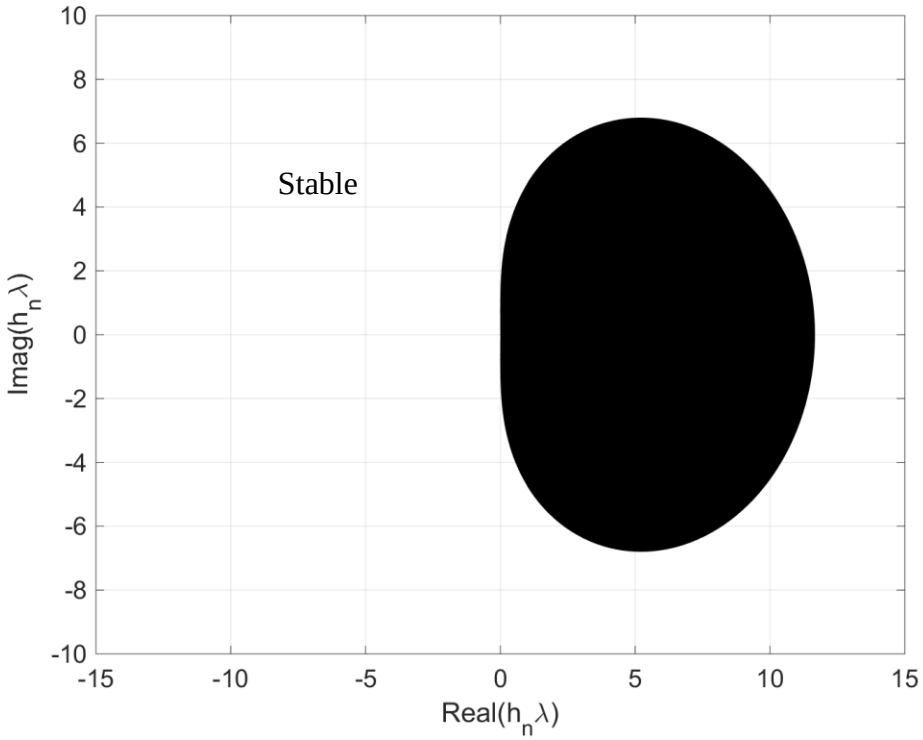


Figure 3.4 : Région de stabilité 2S-DIRK – Stable en dehors de la région noire

Le schéma 2S-DIRK est donc A-Stable.

Vérifions de la même manière si la méthode présente des oscillations numériques en annulant le courant aux bornes d'une inductance ($i_{n+1} = 0$) :

$$\begin{aligned} \tilde{v}_n &= \frac{L}{ah_n}\tilde{\imath}_n - \frac{L}{ah_n}i_n \\ \tilde{\imath}^*{}_n &= \alpha i_n + \beta\tilde{\imath}_n \\ v_{n+1} &= -\frac{L}{ah_n}\tilde{\imath}^*{}_n \\ \tilde{v}_{n+2} &= 0 \\ v_{n+2} &= 0 \end{aligned} \tag{68}$$

La schéma 2S-DIRK ne présente pas d'oscillations numériques en cas de discontinuité.

### 3.1.3 Complexité

La méthode la plus complexe à coder est la méthode 2S-DIRK qui demande deux étapes de calcul et une étape de conversion. Elle est aussi la méthode qui a la plus grande complexité de calcul atteignant deux fois celle de ER, et 1.8 fois celle de TRAP. La méthode de GEAR $2^{ème}$ ordre est une méthode à pas de temps multiples. Elle demande donc de stocker le pas $n-1$ en permanence. L'initialisation doit aussi être réalisée en utilisant ER. GEAR est donc la deuxième méthode la plus complexe, aussi bien au niveau du code que des calculs.

ER et TRAP sont très faciles à coder et ont une rapidité convenable. Nous allons étudier tous ces aspects dans la partie tests et comparaisons.

## 3.2 Traitement des discontinuités par TRAP_DIRK

Comme nous l'avons vu, la méthode TRAP n'est pas stable en cas de discontinuité. Pour y remédier, la méthode la plus courante consiste à passer à la méthode ER sur deux demis pas de temps, puis de repasser vers TRAP s'il n'y a plus de discontinuités. L'utilisation d'un demi pas de temps avec ER permet d'avoir les mêmes admittances et donc la même matrice d'admittance. Aussi, les deux schémas sont très similaires, donc il n'y a pas de grande différence pour le passage d'une méthode à l'autre concernant les courants historiques.

Dans cette plateforme, nous implémentons aussi la combinaison de TRAP avec 2S-DIRK. En effet, cela peut être un bon compromis. 2S-DIRK est beaucoup plus précise que ER, donc en l'utilisant, on devrait augmenter la précision globale de la simulation, comparée à la combinaison TRAP et ER. Dans ce cas-ci, on utilise une matrice d'admittance nodale différente (calculée au début de la simulation au même titre que la matrice d'admittance de TRAP) lorsqu'on passe à 2S-DIRK et le calcul du courant historique est aussi un peu plus délicat.

Nous allons donc démontrer le calcul du courant historique pendant les transitions pour les composants tels que les inductances et les capacitances.

## 3.2.1 Transition de la méthode trapézoïdale vers la méthode 2S-DIRK

### 3.2.1.1 Inductance

Dans le code de la plateforme, les courant historiques sont calculés récursivement. Il est donc important d'actualiser la formule de récurrence lorsqu'on doit passer d'une méthode à l'autre.

L'équation du courant à travers une inductance est donnée par (3) et le schéma trapézoïdal est donné par (36). On obtient ainsi pour l'inductance :

$$i_n = \frac{h_n}{2L} v_n + \frac{h_n}{2L} v_{n-1} + i_{n-1} \tag{69}$$

L'historique de courant est donc égal à :

$$i_{h_n} = \frac{h_n}{2L} v_{n-1} + i_{n-1} \tag{70}$$

Comme présenté dans (52), la première étape de la méthode 2S-DIRK permettant de calculer la valeur intermédiaire est donné par :

$$\tilde{\imath}_n = \frac{a h_n}{L} \tilde{v}_n + i_n \tag{71}$$

Avec $\tilde{x}_n = x(t_n + ah)$.

Le courant historique est donc simplement $i_n$. Donc lorsqu'on passe de la méthode trapézoïdale à la méthode 2S-DIRK, il faut prendre en compte que le courant historique se calcule :

$$i_{h_{n+1}} = i_n = \frac{h_n}{2L} v_n + i_{h_n} \tag{72}$$

#### 3.2.1.2 Capacitance

De la même manière, déterminons la formule de récurrence du courant historique à travers une capacitance lorsqu'on passe de la méthode TRAP à la méthode 2S-DIRK. L'équation du courant à travers une capacitance est donnée par (5).

Le schéma trapézoïdal se décline de la manière suivante :

$$i_n = \frac{2C}{h_n} v_n - \frac{2C}{h_n} v_{n-1} - i_{n-1} \tag{73}$$

L'historique de courant est donc égal à :

$$i_{h_n} = -\frac{2C}{h_n} v_{n-1} - i_{n-1} \tag{74}$$

La première étape de la méthode 2S-DIRK est donné par :

$$\tilde{\imath}_n = \frac{C}{ah_n} \tilde{v}_n - \frac{C}{ah_n} v_n \tag{75}$$

Le courant historique est donc donné par $-\frac{C}{ah_n} v_n$. Donc lorsqu'on passe de la méthode trapézoïdale à la méthode 2S-DIRK, le courant historique se calcule :

$$i_{h_{n+1}} = -\frac{C}{ah_n} v_n \tag{76}$$

### 3.2.2 Transition de la méthode 2S-DIRK vers la méthode trapézoïdale

#### 3.2.2.1 Inductance

Dans cette partie, on passe de la méthode 2S-DIRK, plus précisément de la 2ème étape, vers la méthode TRAP. Le deuxième stage de la méthode 2S-DIRK (54) appliquée à l'inductance donne :

$$i_n = \frac{ah_n}{L} v_n + \tilde{\imath}_n^{\,*} = \frac{ah_n}{L} v_n + i_{h_n} \tag{77}$$

Le schéma trapézoïdal est donné par (69). En remplaçant $i_n$ par son expression dans (77), l'historique de courant pour la méthode TRAP à la suite de 2S-DIRK est donnée par :

$$i_{h_{n+1}} = \frac{h_n}{2L} v_n + i_n = \frac{h_n}{2L} v_n + \frac{ah_n}{L} v_n + i_{h_n} \tag{78}$$

#### 3.2.2.2 Capacitance

Le deuxième stage de la méthode 2S-DIRK est donné par :

$$i_n = \frac{C}{ah} v_n - \frac{C}{ah} \widetilde{v_n}^* = \frac{C}{ah} v_n + ih_n \tag{79}$$

Le schéma trapézoïdal (73) permet de déterminer la formule de récurrence du courant historique :

$$ih_{n+1} = -\frac{2C}{h} v_n - i_n = -\frac{2C}{h} v_n - \frac{C}{ah} v_n - ih_n \tag{80}$$

## 3.3 Tests et comparaisons

### 3.3.1 Validation des méthodes

Afin de valider les méthodes et de mesurer précisément leurs erreurs, nous allons utiliser un circuit RLC série Figure 3.5. En effet, il nous est possible de calculer analytiquement la valeur exacte du courant circulant dans un tel circuit. Nous utilisons pour cela la méthode de Laplace qui permet de trouver l'expression du courant pour une source de tension cosinusoïdale :

$$I(s) = \frac{1}{L} E \frac{s^2}{s^2 + \omega^2} \frac{1}{\left[s^2 + \frac{R}{L} s + \frac{1}{LC}\right]} \tag{81}$$

où $s$ est la variable de Laplace, $\omega$ représente la pulsation (c'est-à-dire $2\pi f$) et $R$, $L$, $C$ et $E$ sont respectivement les paramètres de la résistance, de l'inductance, de la capacitance et de la source de tension. Nous utilisons ensuite Matlab afin de convertir ce courant dans le domaine temporel. L'erreur absolue de chaque méthode est définie comme étant la valeur absolue de la différence entre la valeur exacte du courant calculée ci-dessus, et la valeur du courant obtenu par intégration numérique avec un **pas de temps de $1\mu s$**.

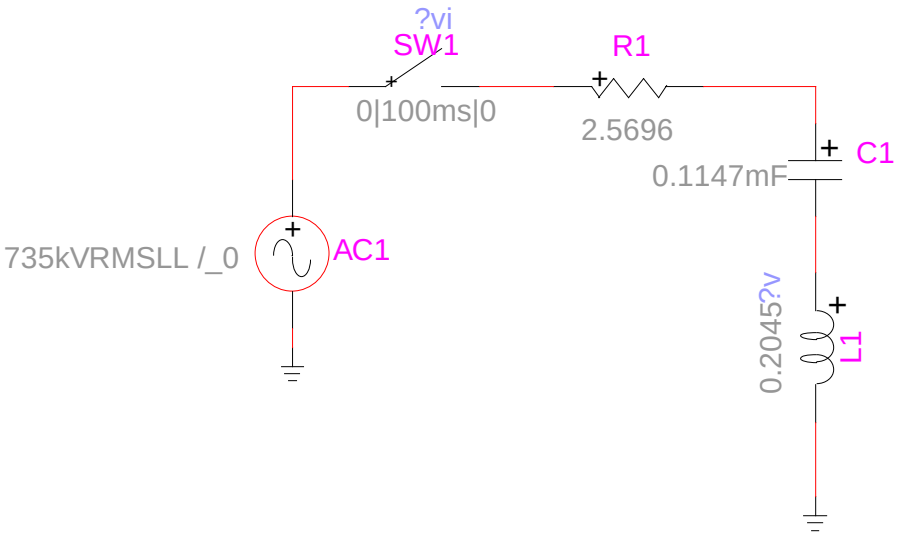


Figure 3.5 : Circuit RLC série

La représentation du courant à travers SW1 est très proche de la référence exacte, quelle que soit la méthode considérée. C'est la raison pour laquelle il est important de regarder l'erreur absolue de ces différentes méthodes (Figure 3.7). On observe ainsi que ER est la moins précise des méthodes, ce qui confirme bien la théorie. Cependant, TRAP est anormalement très peu précise comparée aux autres méthodes. Cela est liée à l'initialisation, qui est très mauvaise pour la méthode TRAP comme on peut le voir sur le zoom de la Figure 3.6. En effet, dès le premier pas de temps, le signal s'écarte de la référence. C'est la raison pour laquelle TRAP_ER et TRAP_DIRK utilisent ER et 2S-DIRK pour initialiser.

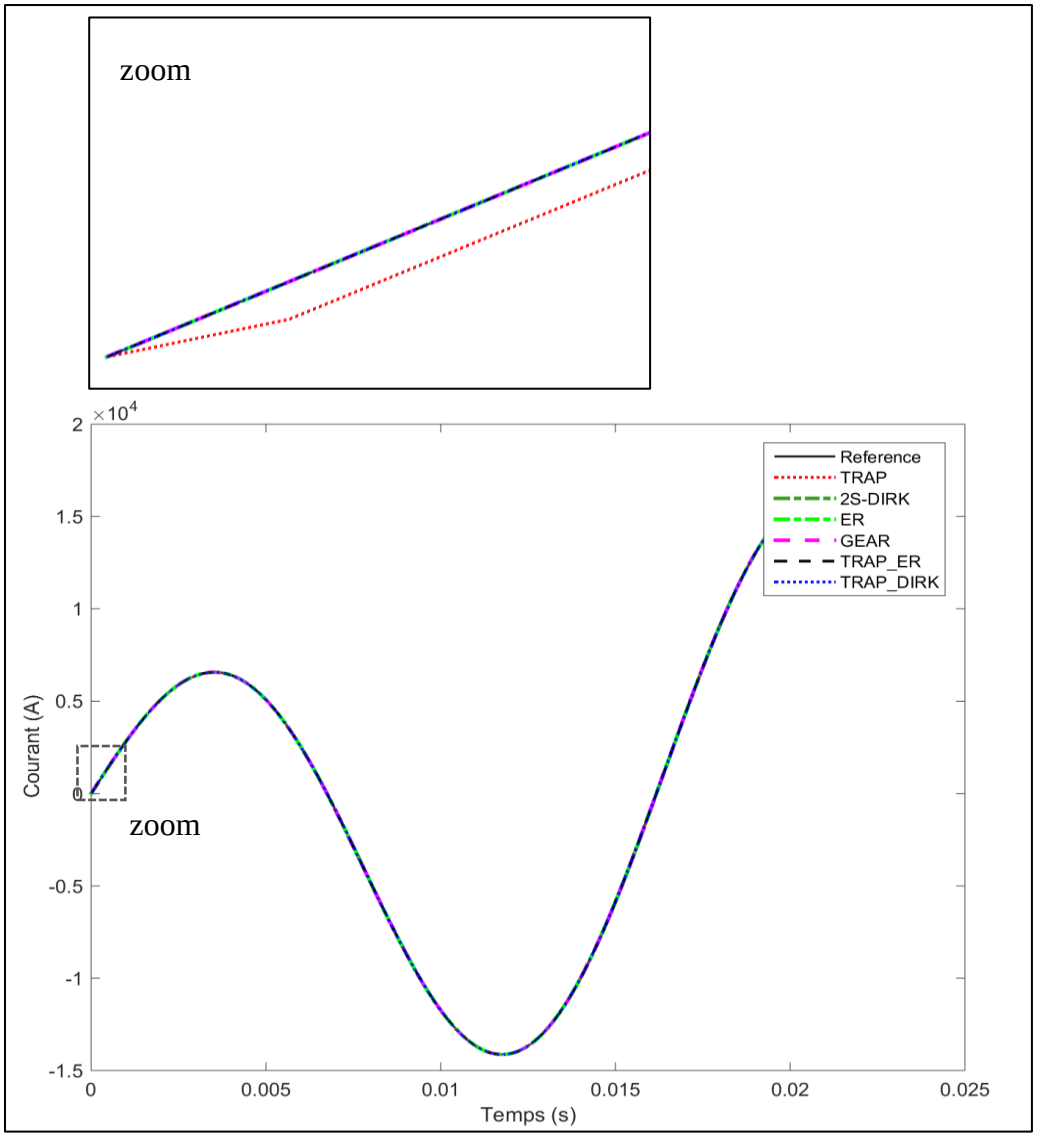


Figure 3.6 : Comparaison des différentes méthodes avec la solution exacte - courant à travers SW1

Lorsqu'on compare les erreurs des autres méthodes, on constate que les méthodes TRAP_DIRK et TRAP_ER ont quasiment la même précision, ce qui est normal sachant que les méthodes ER et 2S-DIRK ne sont utilisées dans chaque cas que sur un pas de temps pour l'initialisation. Ces deux dernières sont plus précises que la méthode de GEAR, et la méthode 2S-DIRK est la plus précise. Cela confirme bien la théorie, lorsqu'on compare les erreurs de troncature locale des différentes méthodes.

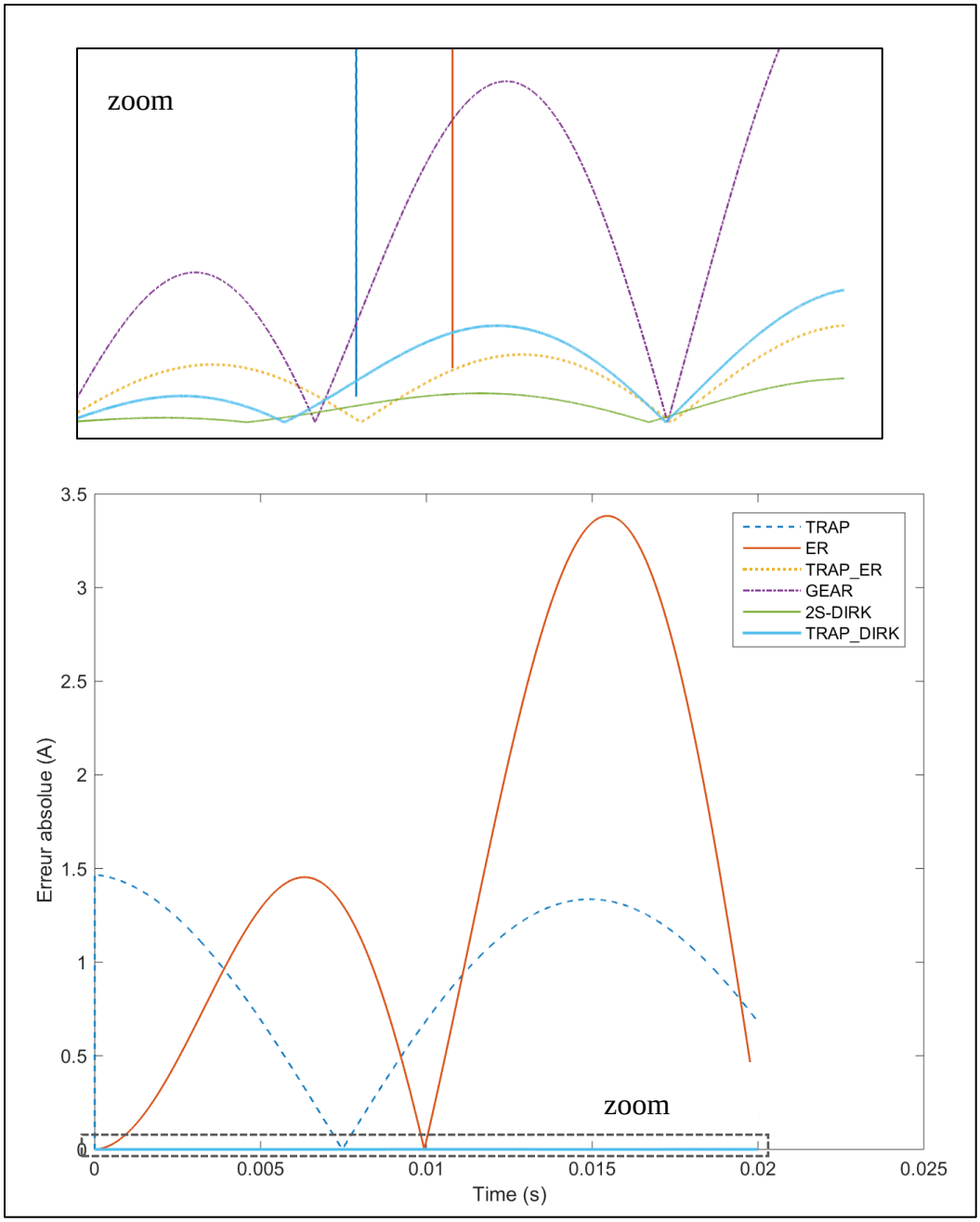


Figure 3.7 : Erreur absolue des différentes méthodes sur le courant à travers SW1

## 3.3.2 Circuit 1 - harmoniques

### 3.3.2.1 Présentation du circuit de test

Le premier circuit de test, tiré d'un cas réel, présente des harmoniques représentées ici par deux sources de courant AC3 et AC2 qui ont respectivement des fréquences de 360 et 180 Hz (6$^{ème}$ et 3$^{ème}$ harmoniques). L'expérience consiste à fermer le disjoncteur SW2 à 100ms puis de l'ouvrir à 300ms. Ces manœuvres mènent à des transitoires avec des fréquences élevées.

On s'intéresse à la tension aux bornes du condensateur C2 (le nœud M2 sur la Figure 3.8). Les différentes simulations sont réalisées avec un **pas de temps de $50\mu s$**. Pour avoir la référence, nous utilisons le logiciel EMTP [8] qui utilise une combinaison de TRAP et ER, avec un pas de temps de $1\mu s$.

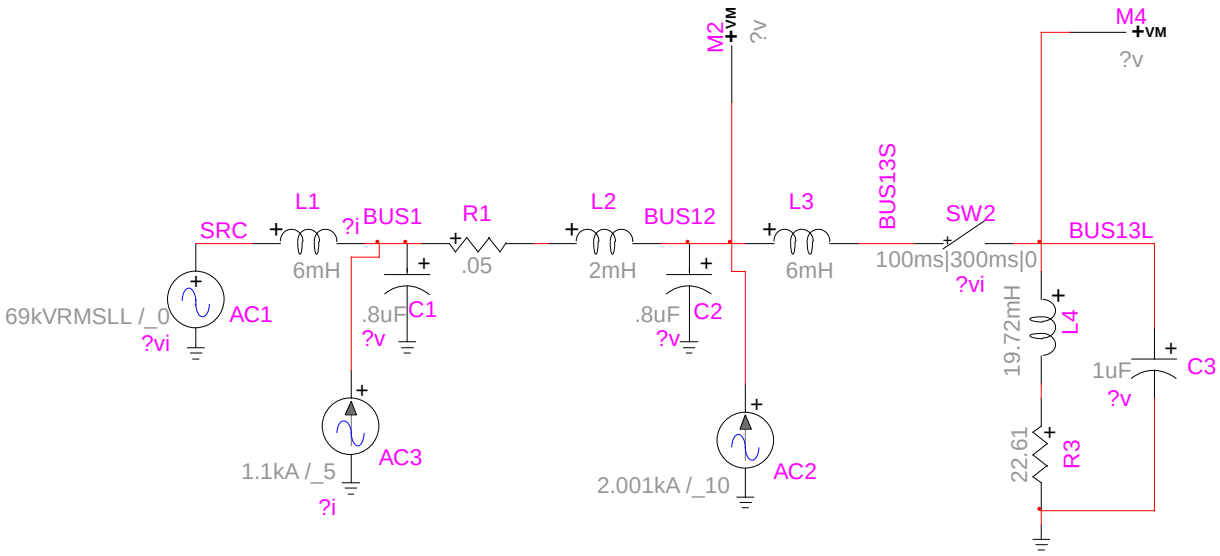


Figure 3.8 : Circuit de test avec des harmoniques

### 3.3.2.2 Présentation et analyse des résultats

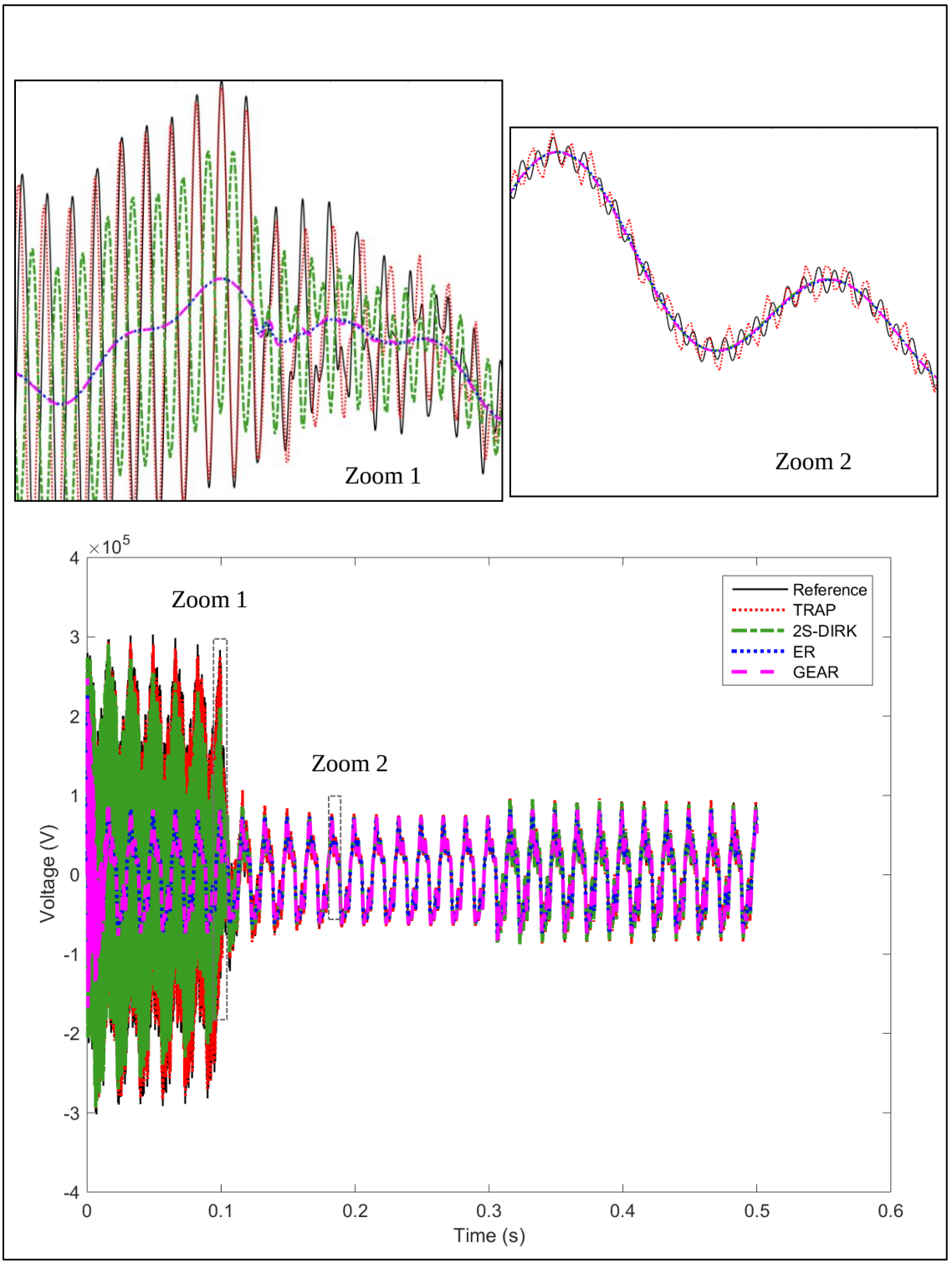


Figure 3.9 : Comparaison de TRAP, ER, GEAR et 2S-DIRK – Tension aux bornes de C2

La Figure 3.9 permet de tirer plusieurs conclusions sur les différentes méthodes. D'abord, on observe que les méthodes ER et GEAR (zoom 1), mais aussi 2S-DIRK de façon moins sévère, amortissent excessivement le signal en présence de hautes fréquences. On constate ce même phénomène lorsqu'on observe le zoom 2 aux alentours de 180ms. Cela diminue la précision de ces méthodes lorsque le signal présente des hautes fréquences.

Afin de démontrer l'amortissement de ces méthodes, nous allons nous référer à la procédure utilisée dans [13].

Considérons l'équation (56) qui nous a permis de démontrer la stabilité des méthodes. Soit $\lambda = \sigma + i\omega$ le facteur complexe dans cette équation. Le facteur d'amortissement relatif est défini comme étant le rapport :

$$\frac{a_n(\lambda)}{a(\lambda)} = \frac{|\frac{x_{n+1}}{x_n}|}{|\frac{x(t_{n+1})}{x(t_n)}|} \tag{82}$$

La solution exacte de (56) étant $x_0\exp(\lambda t)$, on sait facilement calculer $a(\lambda)$ qui est égal à $\exp(\sigma)$. Donc, pour calculer et tracer l'amortissement des différentes méthodes en fonction de la fréquence $\omega$, il suffit de trouver $a_n(\lambda)$ pour chaque méthode.

Les coefficients d'amortissement $a_n(\lambda)$ des méthodes ER, TRAP et 2S-DIRK ont déjà été évalués avec (57), (59) et (66). Cependant, GEAR étant une méthode à pas de temps multiple, est un peu plus compliqué à évaluer.

Tout d'abord, la discrétisation de (56) par GEAR donne l'équation :

$$\left(1 - \frac{2}{3}h_n\lambda\right)x_{n+1} = \frac{4}{3}x_n - \frac{1}{3}x_{n-1} \tag{83}$$

Pour trouver le coefficient d'amortissement de GEAR, il faut d'abord trouver sa matrice d'amplification [14]. Ensuite, le coefficient d'amortissement est donné par la plus grande, en valeur absolue, des valeurs propres de la matrice d'amplification.

Commençons donc par calculer la matrice d'amplification de l'équation (83). Pour cela, il faut le réécrire sous forme matricielle de sorte à avoir uniquement deux vecteurs $X_{n+1} = \begin{pmatrix} x_n \\ x_{n+1} \end{pmatrix}$ et

$X_n = \begin{pmatrix} x_{n-1} \\ x_{n+1} \end{pmatrix}$. On obtient ainsi l'expression : $A_1 X_{n+1} = A_2 X_n$ ou encore $X_{n+1} = A_1^{-1} A_2 X_n$, où les matrices $A_1 = \begin{pmatrix} 1 & 0 \\ 0 & 1 - \frac{2}{3} h_n \lambda \end{pmatrix}$ et $A_2 = \begin{pmatrix} 0 & 1 \\ -\frac{1}{3} & \frac{4}{3} \end{pmatrix}$. La matrice d'amplification est égale à $A_1^{-1} A_2$.

Ainsi, le coefficient d'amortissement de GEAR est donné par :

$$a_n(\lambda) = \rho[A_1^{-1}(\lambda) A_2] \tag{84}$$

où $\rho$ représente la plus grande, en valeur absolue, des valeurs propres de la matrice $A_1^{-1}(\lambda) A_2$.

Nous pouvons enfin tracer et comparer les amortissements des différentes méthodes en fonction de la fréquence, pour des valeurs de $\sigma$ différents.

La Figure 3.10 représente les coefficients d'amortissement relatif des différentes méthodes pour deux valeurs différentes de $\sigma$. Ces figures confirment ce qui a été vu précédemment. En effet, pour une valeur nulle de $\sigma$ (amortissement réel égal à 1), la méthode TRAP admet un amortissement égal à 1. ER, GEAR et 2S-DIRK ont cependant tendance à trop amortir les hautes fréquences, 2S-DIRK présentant un amortissement relatif moins élevé que les deux premières méthodes. Pour une valeur de $\sigma$ inférieur à zéro, le constat reste quasiment le même.

Dans la Figure 3.11, nous comparons les tensions aux bornes de C2 (Figure 3.8) obtenues avec les méthodes TRAP_ER et TRAP_DIRK. On constate en zoomant dans les hautes fréquences que la méthode TRAP combinée à 2S-DIRK présente des pics qui s'apparentent à des oscillations numériques. Cela viendrait du fait que 2S-DIRK n'est pas en mesure d'annuler totalement les oscillations numériques qui découlent de la méthode trapézoïdale. En effet, ER admet un coefficient d'amortissement relatif plus faible que 2S-DIRK, elle est donc plus efficace que 2S-DIRK dans la suppression des oscillations numériques qui découlent de la méthode TRAP.

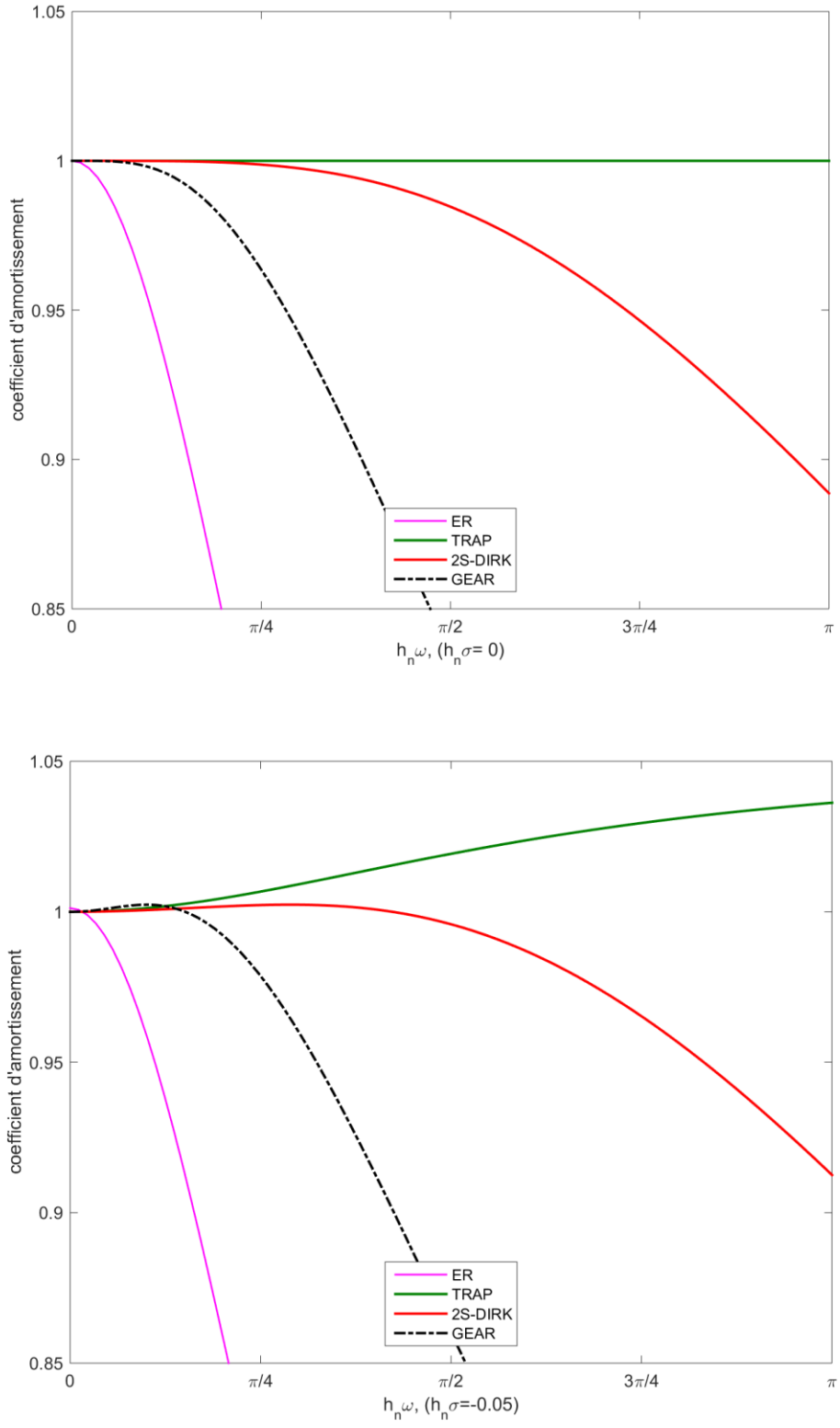


Figure 3.10 : Amortissement relatif des différentes méthodes

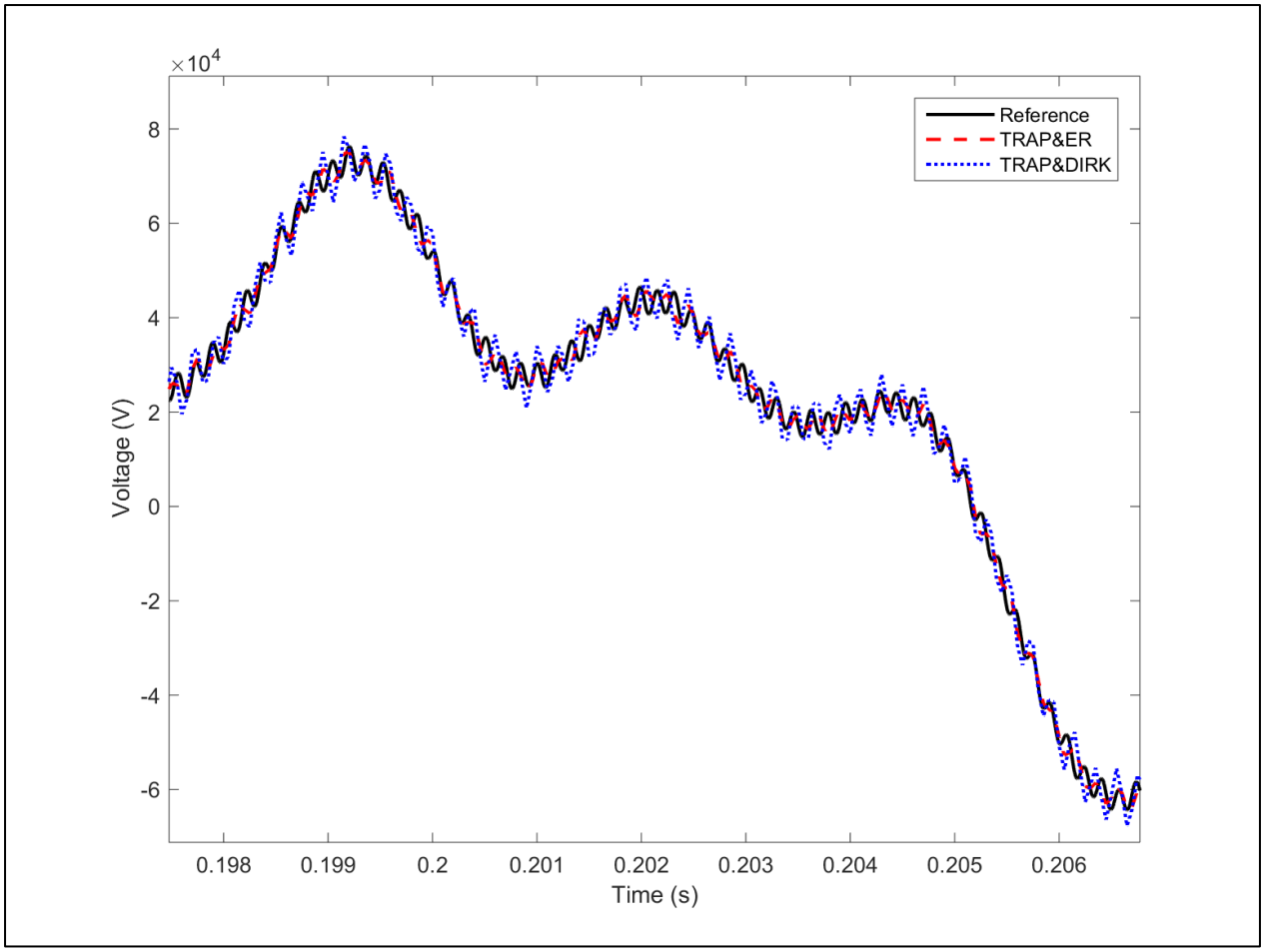


Figure 3.11 : TRAP_ER vs TRAP_DIRK – Tension aux bornes de C2

### 3.3.3 Circuit 2 – Bancs de condensateurs

#### 3.3.3.1 Présentation du circuit de test

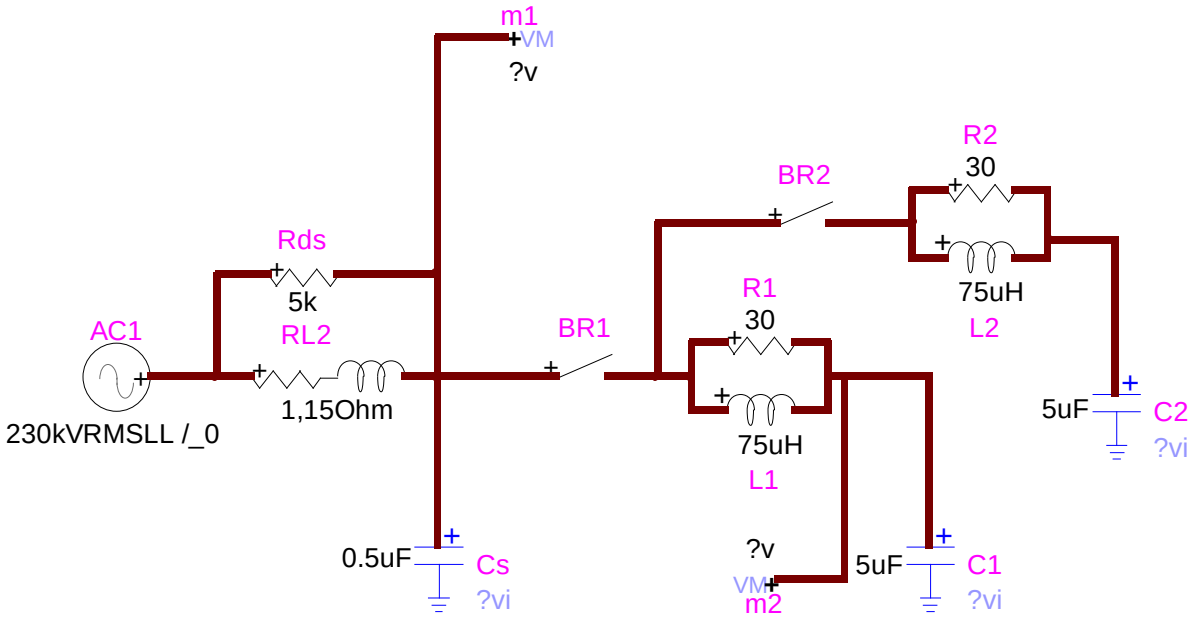


Figure 3.12 : Circuit de test avec bancs de condensateurs

Le deuxième circuit de test, aussi tiré d'un cas réel, est un exemple d'enclenchement de bancs de condensateurs. L'expérience consiste à fermer le disjoncteur BR1 à 20ms, de l'ouvrir à 125ms puis de l'enclencher de nouveau à 200ms. Ensuite, le disjoncteur BR2 est enclenché à 225ms. Ces manœuvres mènent à des transitoires avec des hautes et très hautes fréquences naturelles.

On s'intéresse à la tension aux bornes du condensateur Cs. Les différentes simulations sont réalisées avec un **pas de temps de $50\mu s$**. Pour avoir la référence, nous utilisons le logiciel EMTP qui utilise une combinaison de TRAP et ER, avec un pas de temps de $1\mu s$.

### 3.3.3.2 Présentation et analyse des résultats

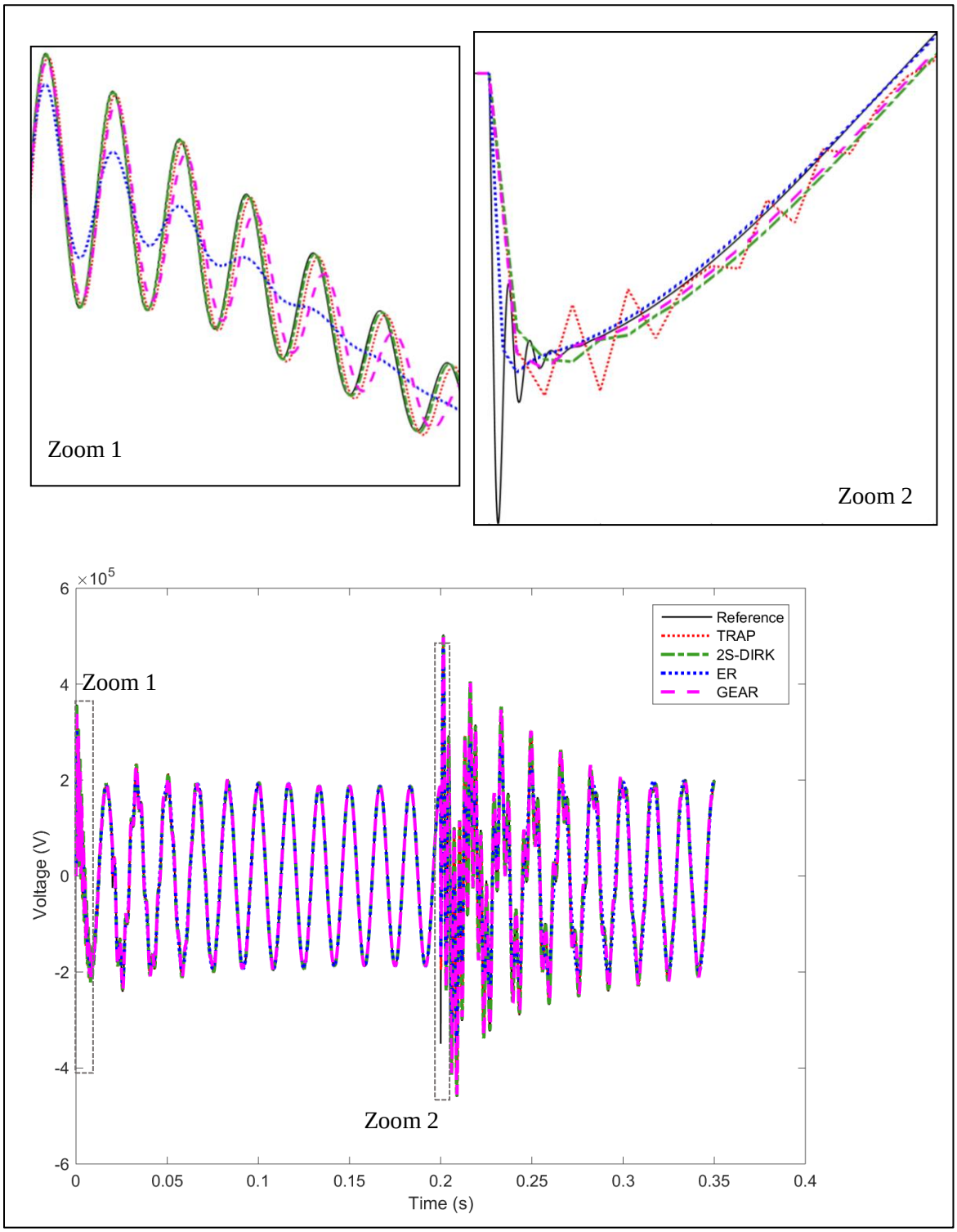


Figure 3.13 : Comparaison de TRAP, ER, GEAR et 2S-DIRK – Tensions aux bornes de Cs

La Figure 3.13 confirme le constat réalisé dans le sous-chapitre précédent : ER et GEAR ont tendance à amortir les hautes fréquences de façon excessive (zoom 1). 2S-DIRK a un amortissement qui est moindre que ces deux dernières méthodes. Sur cette expérience, 2S-DIRK présente une précision similaire à TRAP, mais a l'avantage de ne pas présenter d'oscillations numériques, contrairement à TRAP (zoom 2). 2S-DIRK présente donc une meilleure réponse.

Lorsqu'on compare TRAP_DIRK et TRAP_ER (Figure 3.14), on constate que si ER arrive à totalement supprimer les oscillations de TRAP après l'enclenchement de BR1, ce n'est pas le cas de 2S-DIRK. Cela peut être dû à l'amortissement de 2S-DIRK qui est plus faible que celui de ER. Ce phénomène n'apparait pas à toutes les discontinuités, et dépend surtout des fréquences mises en jeu.

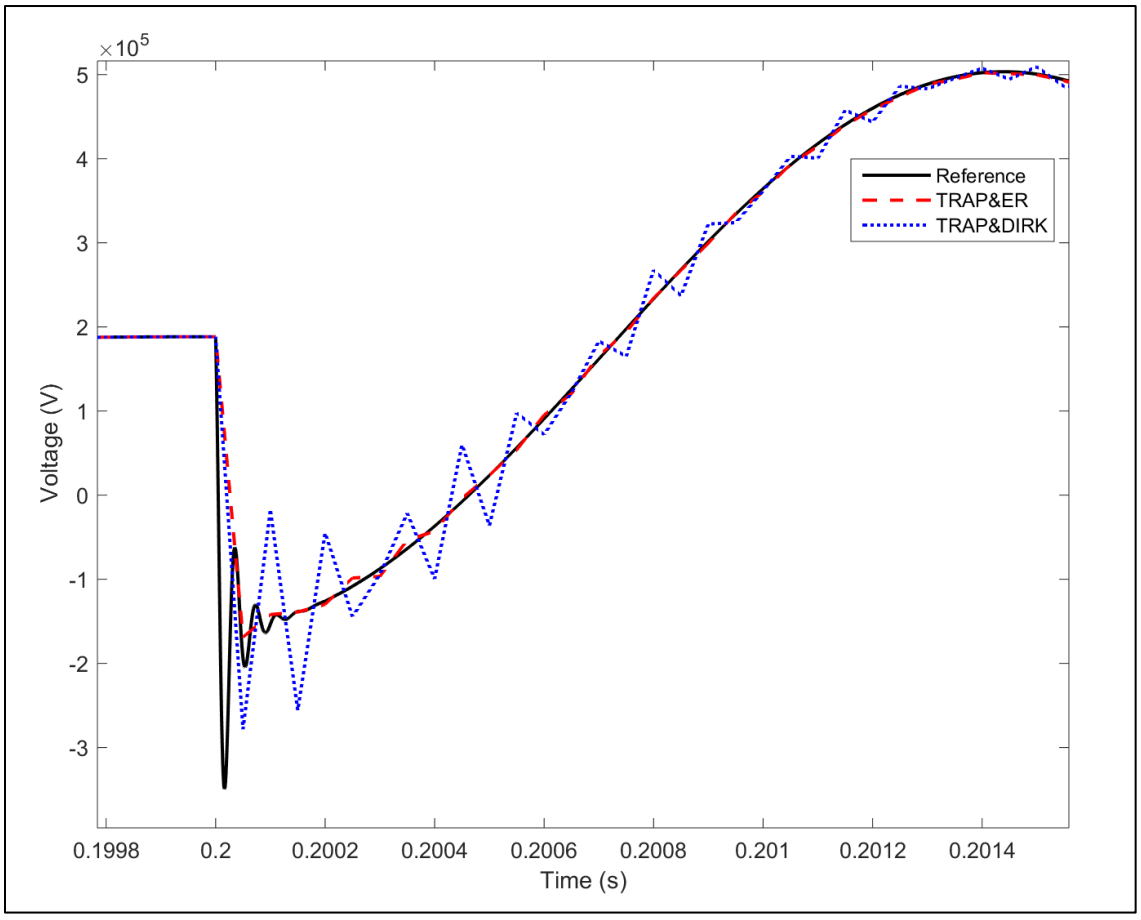


Figure 3.14 : TRAP_ER vs TRAP_DIRK – Tensions aux bornes de Cs

On observe également le courant d'appel à travers l'interrupteur BR1. Les différentes simulations sont réalisées cette fois-ci avec un **pas de temps de $10\mu s$** puis avec un pas de **$1\mu s$**, la référence étant obtenue de la même manière. En effet, la fréquence naturelle maximale à travers BR1 est d'environ 28 kHz. Or dans la simulation, afin de s'assurer de bien représenter les ondes, on essaie de prendre un pas de temps 10 fois plus petit que l'inverse de la plus grande fréquence. Le pas de temps doit donc être plus petit que $35\mu s$.

La Figure 3.15 confirme le comportement de TRAP_DIRK, qui présente des oscillations d'amplitude trop élevée et qui durent trop longtemps, similaires à celles de la méthode TRAP. Les autres méthodes cependant représentent un peu mieux cette haute fréquence à l'exception de ER et GEAR qui ont un amortissement trop élevé.

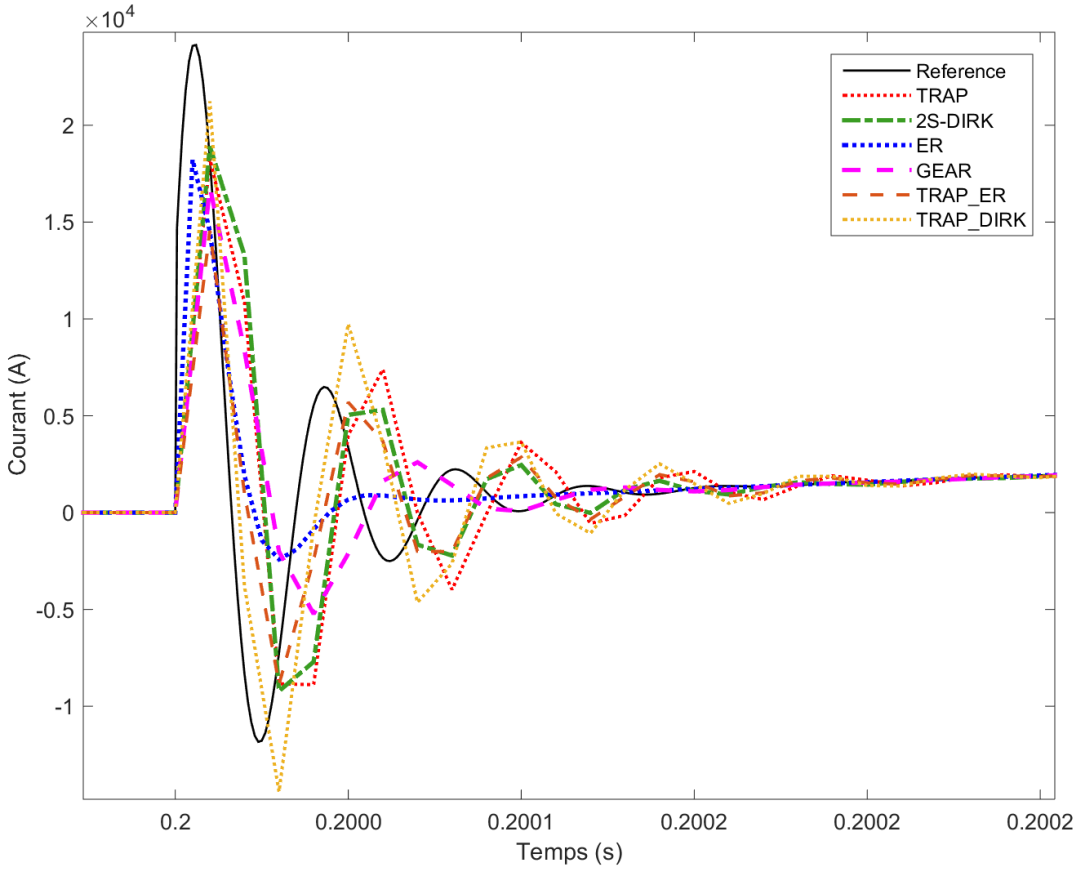


Figure 3.15 : Comparaison des différentes méthodes ($h_n = 10\mu s$)– Courant d'appel à travers BR1

La Figure 3.16 présente les formes d'ondes obtenues avec un pas de temps de $1\mu s$. À l'exception de ER dont l'amortissement des hautes fréquences se fait encore sentir malgré le petit pas de temps, les autres méthodes représentent toutes fidèlement le courant d'appel.

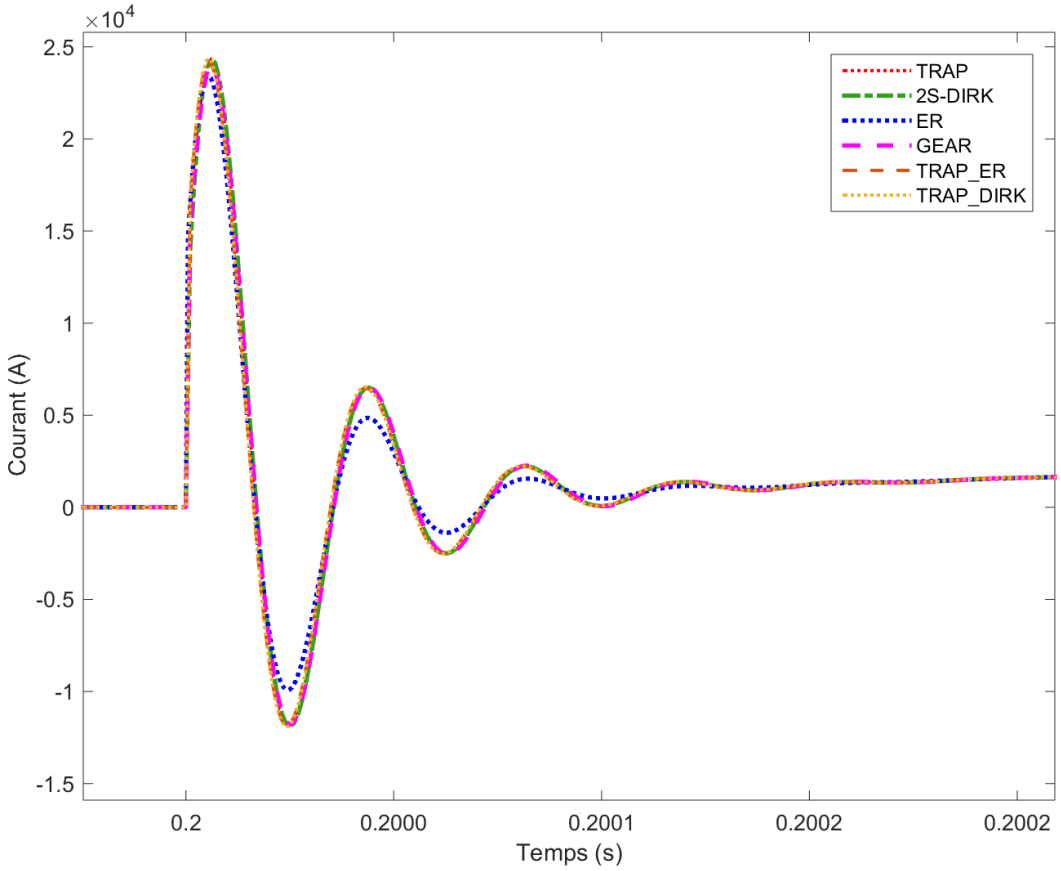


Figure 3.16 : Comparaison des différentes méthodes ($h_n = 1\mu s$)– Courant d'appel à travers BR1

### 3.3.4 Circuit 3 – défaut phase a terre

#### 3.3.4.1 Présentation du circuit de test

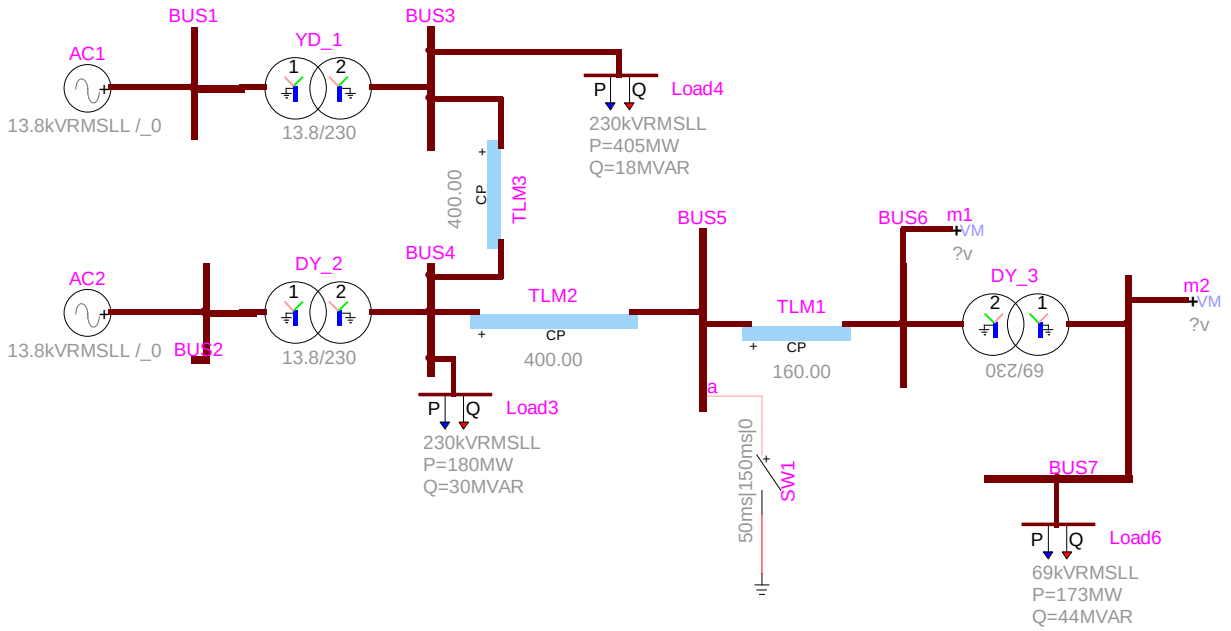


Figure 3.17 : Circuit de test avec défaut phase-terre

Le troisième circuit de test, est un exemple de défaut phase-terre sur un petit réseau. Le défaut apparait à la barre BUS5 au niveau de la phase a et dure 6 cycles. Cela entraîne une baisse de la tension de phase a à la barre BUS7 puis un recouvrement lorsque le défaut disparaît. Nous observons en particulier le comportement de chaque méthode durant les transitoires.

On s'intéresse au courant de défaut juste après la fermeture de SW1. Les différentes simulations sont réalisées avec un **pas de temps de $5\mu s$**. Pour avoir la référence, nous utilisons le logiciel EMTP qui utilise une combinaison de TRAP et ER, avec un pas de temps de $1\mu s$.

#### 3.3.4.2 Présentation et analyse des résultats

Le réseau de la Figure 3.18 permet entre autres de tester le bon fonctionnement de toutes les méthodes avec un réseau triphasé un peu plus complet. En effet, ce réseau présente des lignes, des charges et des transformateurs triphasés avec une branche de magnétisation.

Les 4 méthodes ainsi que les deux combinaisons (Figure 3.19) représentent bien le régime permanent, mais aussi le régime transitoire juste après le défaut qui est quasiment pareil pour toutes les méthodes. Les très hautes fréquences et perturbations transitoires sont bien représentées. S'il permet de valider les différentes méthodes sur un réseau triphasé contenant des lignes équilibrées, ce circuit ne permet pas de les départager en matière de précision et de stabilité.

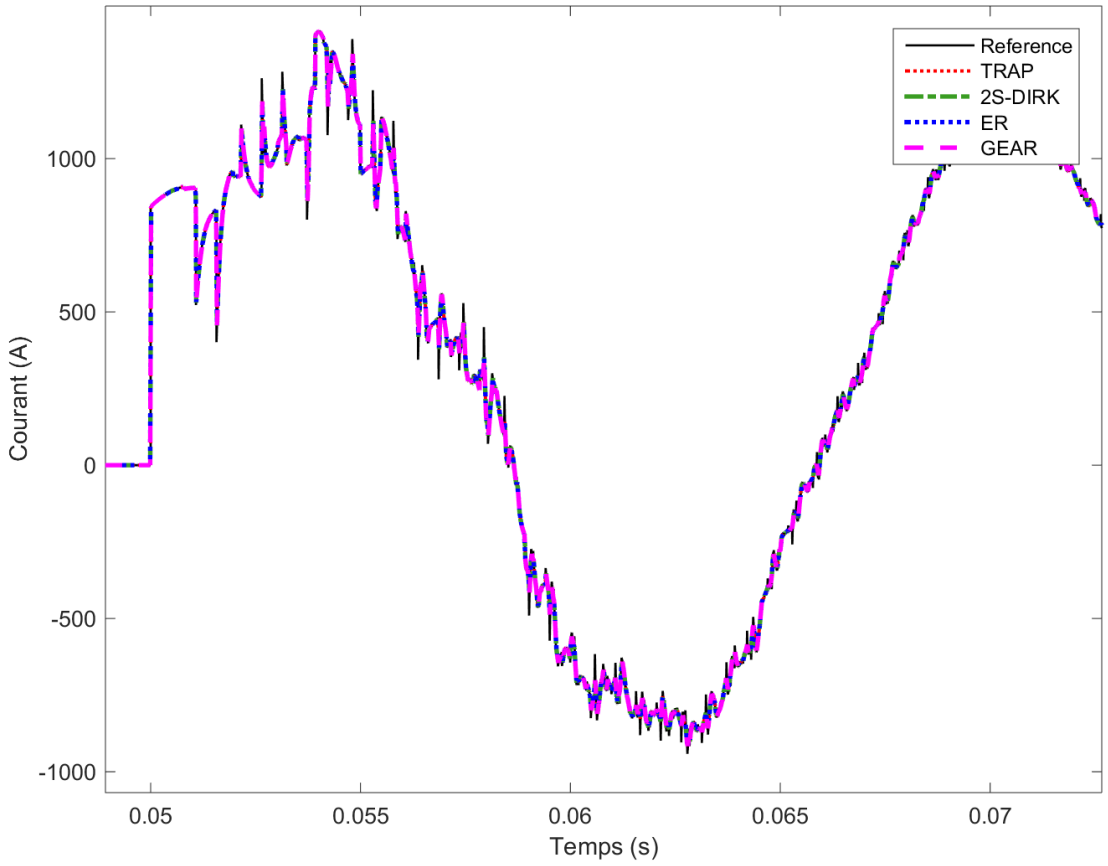


Figure 3.18 : Comparaison de TRAP, ER, GEAR et 2S-DIRK – courant de défaut

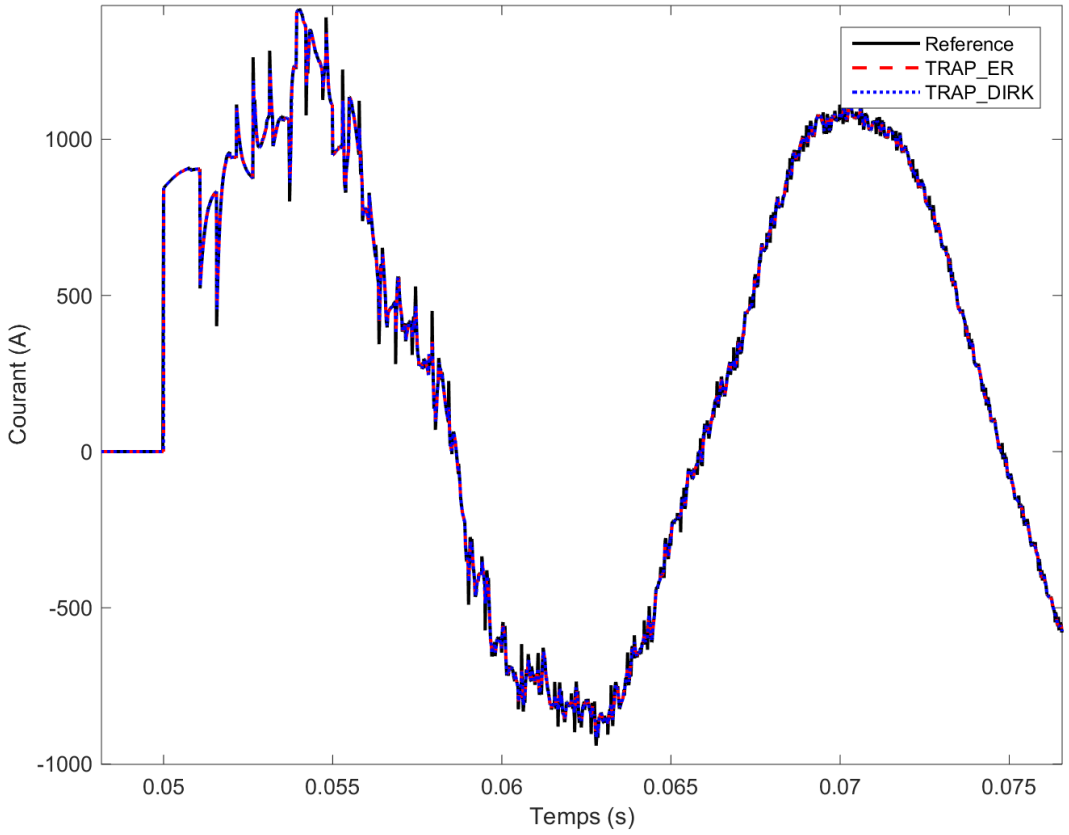


Figure 3.19 : Comparaison de TRAP_ER et TRAP_DIRK – courant de défaut

## 3.3.5 Circuit 4 – Ferro-résonance

### 3.3.5.1 Présentation du circuit de test

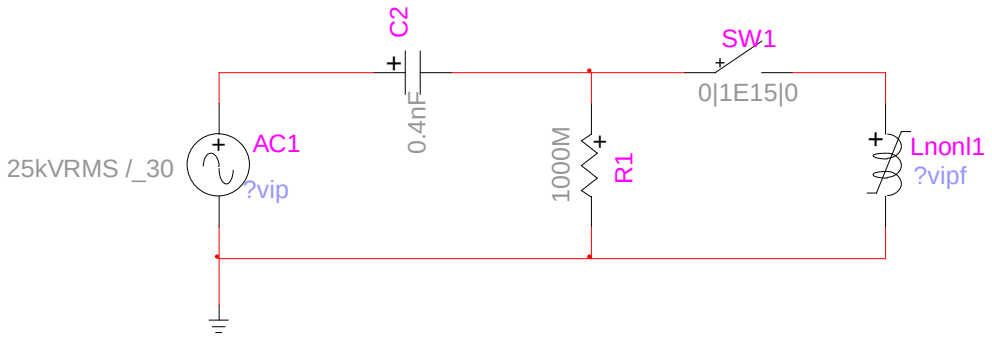


Figure 3.20 : Circuit de test avec non-linéarités

Le quatrième circuit de test met en évidence les non-linéarités et les phénomènes de Ferro-résonance grâce à l'inductance non-linéaire. Le bon comportement d'une méthode dans le traitement d'un réseau non-linéaire est essentiel étant donné que la quasi-totalité des réseaux existants présentent des non-linéarités.

On s'intéresse à la tension aux bornes de l'inductance non-linéaire Lnonl1. Les différentes simulations sont réalisées avec un **pas de temps de $50\mu s$**. Pour avoir la référence, nous utilisons le logiciel EMTP qui utilise une combinaison de TRAP et ER, avec un pas de temps de $1\mu s$. Les paramètres de la simulation sont choisis de sorte à avoir des phénomènes de Ferro-résonance. La source à un déphasage de 30° et la caractéristique flux-courant est donnée à la Figure 3.21.

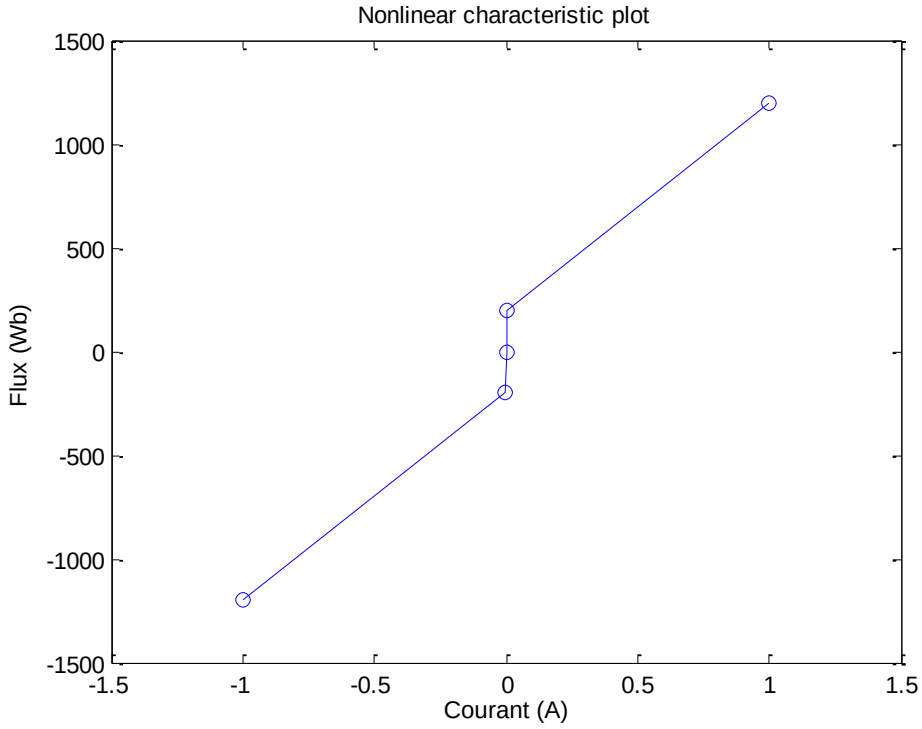


Figure 3.21 : Caractéristique flux-courant de l'inductance non-linéaire

### 3.3.5.2 Présentation et analyse des résultats

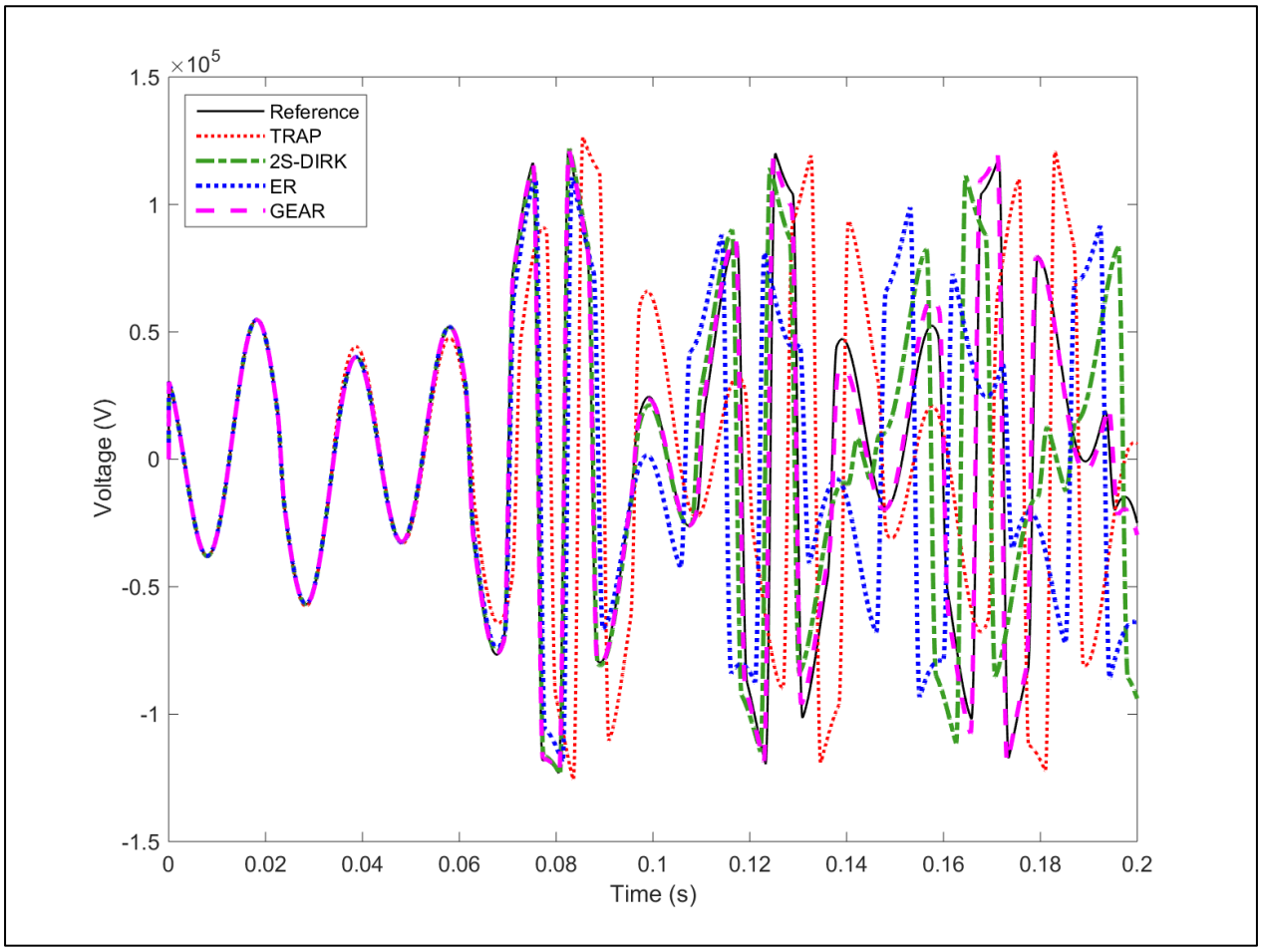


Figure 3.22 : Comparaison de TRAP, ER, GEAR et 2S-DIRK - Tension aux bornes de l'inductance non-linéaire Lnonl1

En présence des non-linéarités, la méthode GEAR est celle qui est plus proche de la référence. On peut aussi observer que 2S-DIRK est plus précise que les méthodes TRAP et ER qui s'écartent assez rapidement de la référence dès le début des non-linéarités Ceci est normal car TRAP devient instable (oscillations numériques) lorsqu'il y a des discontinuités (changement de pente de l'inductance non-linéaire) et ER a une faible précision. On peut donc conclure ici que GEAR et 2S-DIRK ont le meilleur comportement en Ferro-résonance.

La comparaison des combinaisons TRAP_ER et TRAP_DIRK permet de confirmer le constat réalisé à la Figure 3.22. En effet, la Figure 3.23 montre que TRAP_DIRK est plus précis que TRAP_ER. À chaque changement de pente au niveau de l'inductance non-linéaire, la méthode bascule vers ER qui a une précision de 1$^{er}$ ordre. Cela explique donc la faible précision de TRAP_ER lorsqu'on est en Ferro-résonance car on utilise trop souvent la méthode ER qui a une faible précision. Par contre, 2S-DIRK a une précision de 2$^{ème}$ ordre. Sa combinaison avec TRAP conserve donc une bonne précision.

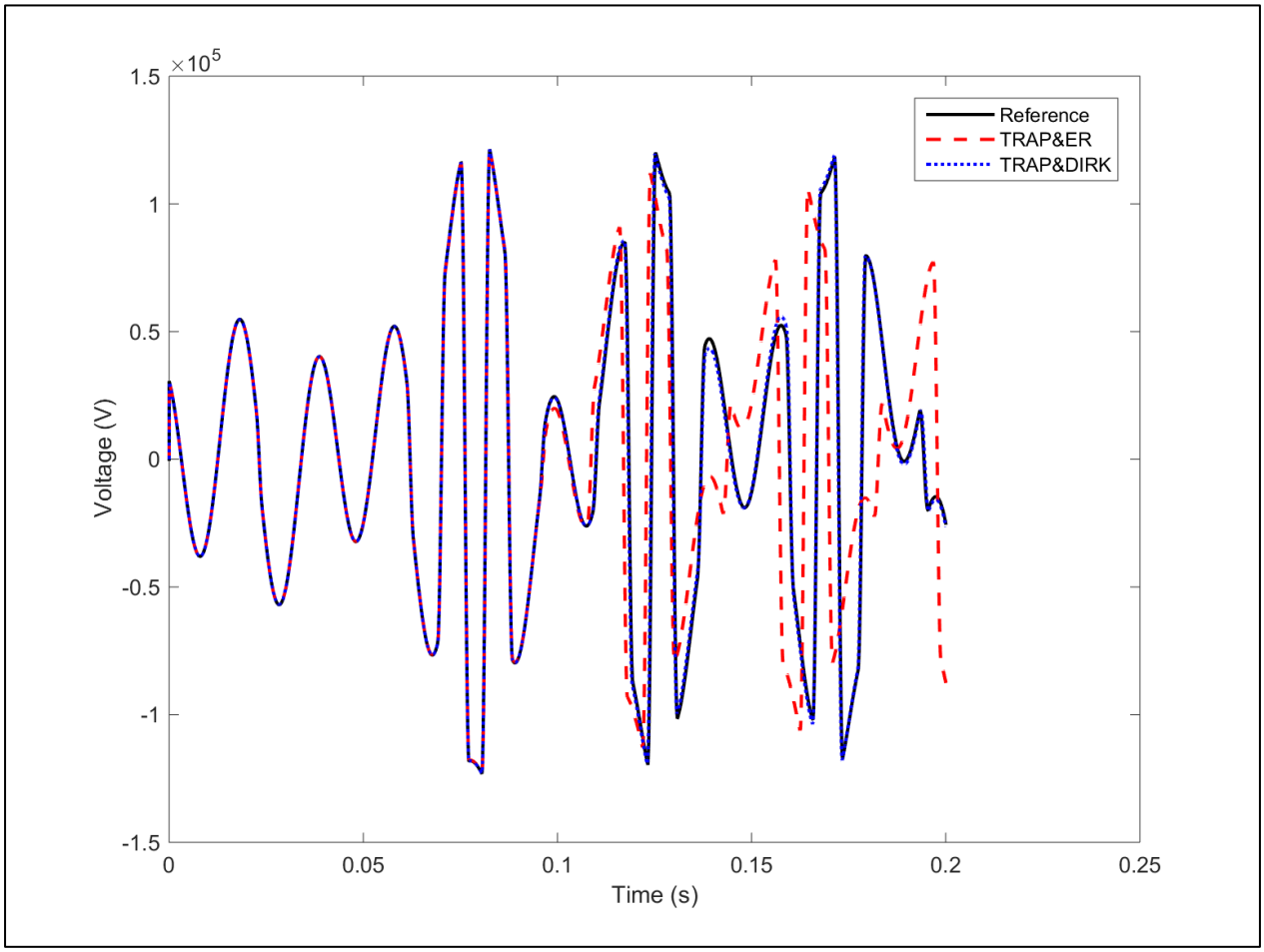


Figure 3.23 : TRAP_ER vs TRAP_DIRK ($h_n = 50\mu s$)– Tension aux bornes de l'inductance non-linéaire Lnonl1

Pour confirmer ce résultat, nous réalisons la même simulation avec un **pas de temps de $10\mu s$** et comparons de nouveau les méthodes TRAP_ER et TRAP_DIRK à la référence ($h_n = 1\mu s$). On constate de nouveau que la courbe obtenue avec TRAP_DIRK reste toujours plus proche de la référence que celle obtenue avec TRAP_ER (Figure 3.24).

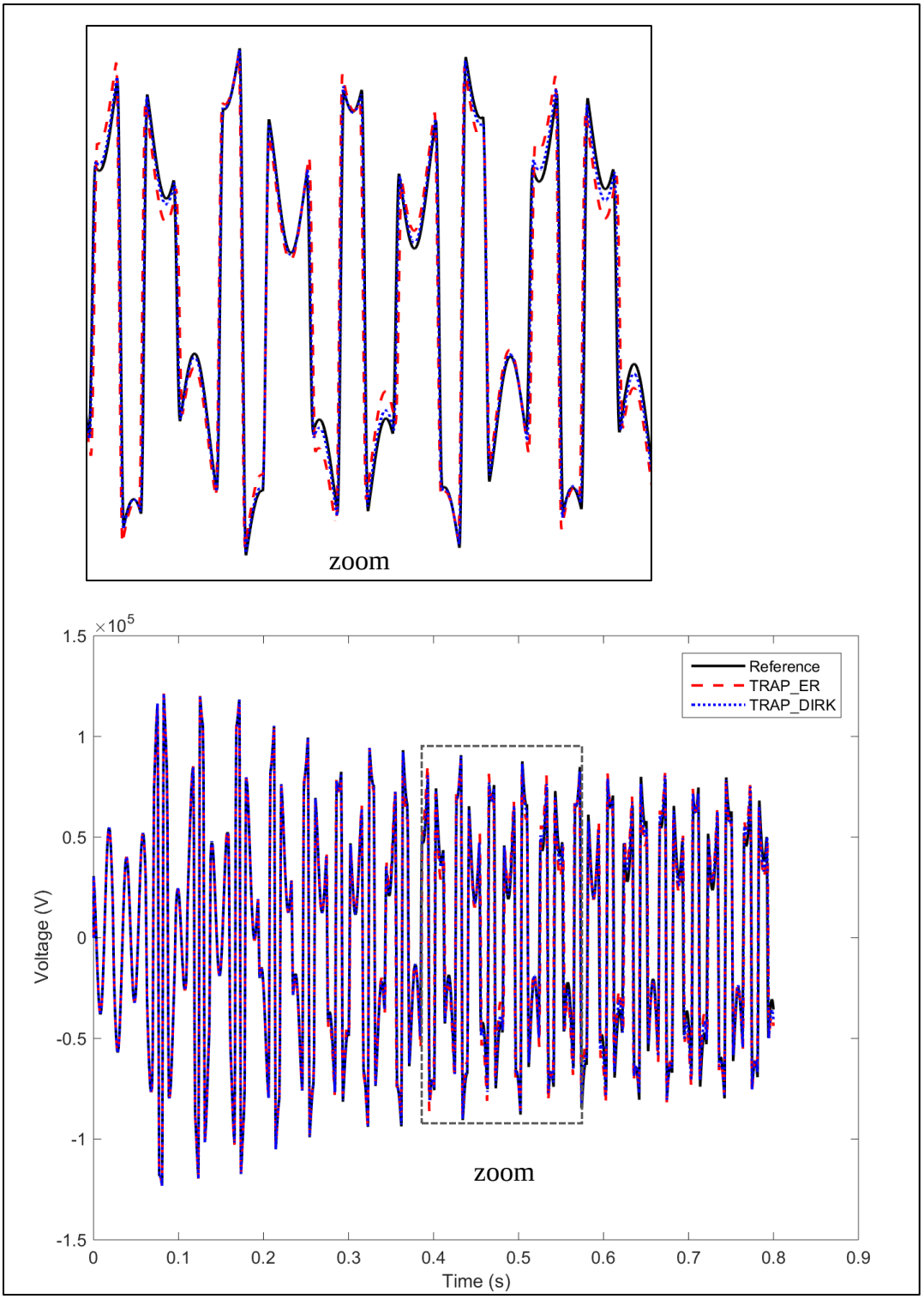


Figure 3.24 : TRAP_ER vs TRAP_DIRK ($h_n = 10\mu s$) – Tension aux bornes de l'inductance non-linéaire Lnonl1

### 3.3.6 Complexités et conclusion partielle

Tableau 3-1 : Comparaison de la complexité des différentes méthodes

| Temps de résolution relatif | | | | | | |
|---|---|---|---|---|---|---|
| | Méthode | | | | | |
| **Circuits** | **TRAP** | **ER** | **TRAP_ER** | **GEAR** | **2S-DIRK** | **TRAP_DIRK** |
| Circuit 1 - Harmoniques | 1 | 0.9480 | 1.0013 | 1.0541 | 1.6778 | 1.0101 |
| Circuit 2 - Bancs de condensateurs | 1 | 0.9691 | 1.0070 | 1.0721 | 1.5502 | 1.0167 |
| Circuit 3 - Défaut phase terre | 1 | 0.8990 | 1.0712 | 1.1295 | 1.7576 | 1.0878 |
| Circuit 4 - Ferro-résonance | 1 | 0.9758 | 1.0771 | 1.3557 | 1.7679 | 1.1862 |
| **Moyenne** | **1** | **0.9480** | **1.0392** | **1.1528** | **1.6884** | **1.0752** |

La rapidité de la méthode est un critère important afin de faire un choix. Comme énoncé dans les hypothèses, la méthode à valider doit avoir une rapidité en deçà de 1.1 fois celle de la méthode TRAP_ER, pour un même pas de temps. Lorsqu'on regarde le tableau 3-1, on conste que cela est vrai uniquement pour les méthodes TRAP (0.96 fois), ER (0,91 fois) et TRAP_DIRK (1.03)

La méthode GEAR est très juste, atteignant 1.1 fois la rapidité de TRAP_ER. De plus, son amortissement trop élevé diminue grandement sa précision en présence de hautes fréquences. Elle n'est donc pas convenable de ce point de vue. 2S-DIRK, à cause des deux étapes que cela implique est aussi trop lente. Elle présente aussi un amortissement, moins sévère que GEAR, sur les hautes fréquences.

Pour conclure sur le choix de méthode, nous allons donc considérer les méthodes TRAP_ER et TRAP_DIRK. En effet, la précision de ER est trop faible et TRAP est instable en cas de discontinuités. En termes de précision, TRAP_DIRK est mieux que TRAP_ER, car la méthode 2S-DIRK est bien plus précise que la méthode ER. Cependant, 2S-DIRK a aussi moins d'amortissement que ER, donc dans certains cas, elle a du mal à éliminer totalement les oscillations qui proviennent de TRAP, à moins de baisser le pas de temps.

Aucune des deux méthodes n'est parfaite pour tous les cas. Cependant, TRAP_DIRK est légèrement meilleure grâce à la précision accrue, surtout en présence de non-linéarités.

# CHAPITRE 4 INTÉGRATION NUMÉRIQUE À PAS VARIABLE

La plupart des logiciels de simulation de type EMTP utilisent un pas fixe. En effet, l'utilisation d'un pas variable peut présenter beaucoup d'intérêts mais son implémentation efficace est compliquée. Souvent, elle est basée sur le contrôle de l'erreur de troncature [6]. Quand il s'agit de signaux avec une fréquence très élevée, le changement de pas devient souvent abusif ce qui rend la simulation très lente.

Pourtant, il y a un grand intérêt à pouvoir utiliser plusieurs pas de temps. En effet, sur certaines simulations, les transitoires à hautes fréquences se présentent seulement sur une durée très courte, et pour bien les représenter, il faut utiliser un pas de temps assez petit. Il serait donc intéressant de pouvoir utiliser un petit pas de temps seulement durant cette durée et augmenter le pas pour le reste de la simulation.

## 4.1 Principe général

L'idée est de réaliser un algorithme qui permet de spécifier des pas de temps différents sur des tranches de la simulation. Nous avons utilisé la méthode TRAP pour illustrer cela. Le premier algorithme implémenté permet de spécifier le pas de temps manuellement tandis que le deuxième se base sur l'erreur de troncature et sur des tolérances prédéfinies pour déterminer un pas de temps parmi une liste de pas donnée.

Avant d'étudier ces deux algorithmes, nous démontrons d'abord en exemple le calcul du courant historique à travers une inductance lors d'un changement de pas et le traitement du délai de propagation des lignes. Nous évaluons aussi de façon plus détaillée l'erreur de troncature de la méthode TRAP et notamment sa discrétisation.

### 4.1.1 Gestion des transitions

Au moment du changement de pas, il faut ajuster le code car la discrétisation change. Pour l'expliquer plus clairement, considérons un exemple avec une inductance. L'équation du courant est donnée par (3) et le schéma trapézoïdal se décline (69).

L'historique de courant est donc égal à :

$$i_{h_{n+1}} = \frac{h_{n+1}}{2L} v_n + i_n \tag{85}$$

Lorsque le pas ne change pas, l'historique de courant pour l'inductance est mis à jour en utilisant la formule de récurrence :

$$i_{h_{n+1}} = i_{h_n} + 2\frac{h_{n+1}}{2L} v_n \tag{86}$$

Supposons maintenant que l'on effectue un changement de pas à l'instant t. C'est-à-dire que le pas $h_n \neq h_{n+1}$. Le courant historique ne peut plus être calculé à partir de la formule de récurrence dans ce cas. On doit retourner à la formule générale (85). Pour cela, il faut recalculer $i_n$ en utilisant le pas $h_n$ avant de calculer $i_h$ :

$$i_n = \frac{h_n}{2L} v_n + i_{h_n} \tag{87}$$

$$i_{h_{n+1}} = \frac{h_{n+1}}{2L} v_n + i_n = \frac{h_{n+1} + h_n}{2L} v_n + i_{h_n} \tag{88}$$

Par la suite, la formule de récurrence (86) permet de mettre à jour le courant historique jusqu'au prochain changement de pas.

Pour ce qui concerne le condensateur et l'inductance non-linéaire, le principe reste le même. Cependant, les lignes présentent un délai de propagation et demandent un traitement plus particulier.

### 4.1.2 Traitement des lignes

Afin de stocker les valeurs permettant de calculer les courants historiques des lignes, un tableau de taille $E\left(\frac{\tau}{h_{min}}\right) + 1$ est utilisé, où E est la fonction partie entière, $\tau$ représente le délai de propagation et $h_{min}$ le plus petit pas de temps spécifié. En effet, dans cet algorithme d'intégration à pas variable, la mémoire de stockage est fixe et le pas le plus petit est utilisé pour de déterminer la taille du tableau. Aussi, il est imposé que tous les pas soient des multiples du plus petit pas de temps.

Ainsi, la première étape est le stockage des tensions et courants au niveau des lignes. Ici, on utilise l'interpolation pour stocker toutes les valeurs entre $t$ et $t + h_n$ dans le tableau (il y a $\frac{h_n}{h_{min}}$ valeurs à mettre à jour). Ensuite on déplace le pointeur du tableau convenablement afin de trouver les

valeurs à $t - \tau$ : le pointeur est déplacé de $\frac{h_n}{h_{min}}$ rangs. Enfin, on calcule les tensions et courants à $t - \tau$ par interpolation.

La différence entre le traitement des lignes avec un pas fixe et un pas variable réside donc uniquement dans le stockage des données historiques (en pas variable il faut stocker plus d'une valeur et cela est fait par interpolation) et le déplacement du pointeur (qui dépend du rapport entre le pas actuel et la pas minimum).

### 4.1.3 Erreur de troncature locale

L'erreur de troncature locale est définie par $\epsilon = |x(t_{n+1}) - x_{n+1}|$ en considérant $x(t_n) = x_n$.

Elle est souvent utilisée pour déterminer la précision d'une méthode d'intégration numérique car c'est la somme des erreurs de troncature locale qui donne l'erreur globale.

Nous avons vu que l'erreur de troncature locale pour la méthode TRAP est donnée par (39):

Si on veut l'implémenter dans un algorithme, il faut pouvoir calculer la dérivée troisième de la grandeur $x$. Cette dérivée discrétisée est donnée dans [5] par :

$$\frac{d^3x}{dt^3} = \frac{\frac{\frac{x_{n+1} - x_n}{h_{n+1}} - \frac{x_n - x_{n-1}}{h_n}}{\langle h_{n+1} \rangle} - \frac{\frac{x_n - x_{n-1}}{h_n} - \frac{x_{n-1} - x_{n-2}}{h_{n-1}}}{\langle h_n \rangle}}{\frac{h_{n+1} + h_n + h_{n-1}}{3}} \tag{89}$$

avec $\langle h_n \rangle = \frac{h_n + h_{n-1}}{2}$

Il faut donc stocker les valeurs de $x$ aux instants $t_{n-2}$, $t_{n-1}$ et $t$ aussi bien que les pas de temps utilisés (le pas de temps n'étant pas fixe).

## 4.2 Pas variable par intervalle

### 4.2.1 Présentation de l'algorithme

L'algorithme est très simple en principe. En entrée, il faut spécifier dans un même tableau et par couple, les valeurs de chaque pas de temps et l'instant de changement de pas (Figure 4.1). Par exemple, comme on le fera dans les tests qui vont suivre, on peut vouloir baisser le pas de temps

avant une ouverture ou fermeture d'interrupteur donnant lieu à des transitoires avec des fréquences élevées puis augmenter le pas de temps une fois le régime permanent atteint.

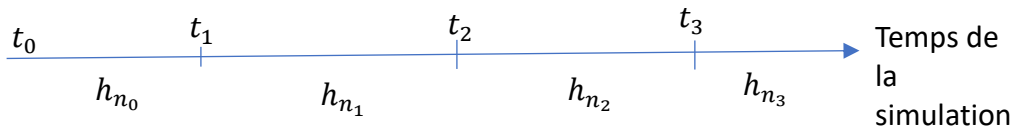


Figure 4.1 : Schéma explicatif de l'algorithme à pas variable par intervalle

Afin d'alléger les calculs, comme le nombre de pas de temps est limité, on peut calculer certaines données en avance telles que la matrice d'admittance nodale. Cela évite d'avoir à la recalculer à chaque changement de pas comme dans les algorithmes avec un changement de pas continu [5]. On peut aussi stocker les admittances des différents composants (inductances linéaires et non-linéaires et capacitances).

### 4.2.2 Tests et comparaisons

- Présentation du circuit

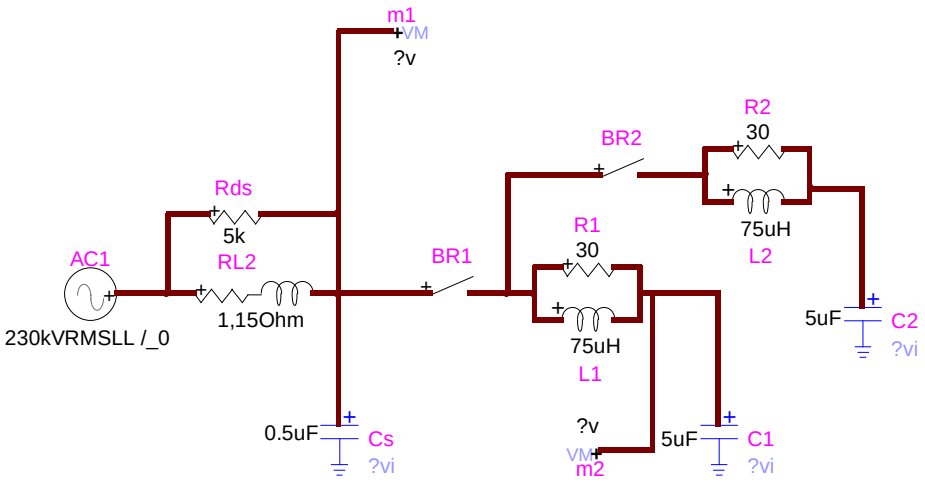


Figure 4.2 : Circuit de test avec bancs de condensateurs

Nous utilisons ici le même circuit que dans le sous-chapitre 3.3.2. Pour rappel, l'expérience consiste à fermer le disjoncteur BR1 à 20ms, de l'ouvrir à 125ms puis de l'enclencher de nouveau à 200ms. Ensuite, le disjoncteur BR2 est enclenché à 225ms. Ces manœuvres mènent à des transitoires avec des hautes et très hautes fréquences naturelles.

On s’intéresse à la tension aux bornes du condensateur Cs. Pour avoir la référence, nous utilisons la méthode TRAP avec un pas de temps de $1\mu s$. La simulation à pas variable par intervalle utilise les paramètres ci-dessous :

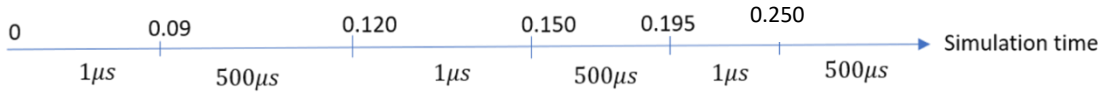


Figure 4.3 : Paramètres de l'algorithme à pas variable par intervalle pour le circuit 1

Le principe général ici est d’observer le moment des transitoires et d’utiliser un petit pas de temps.

- Présentation et analyse des résultats

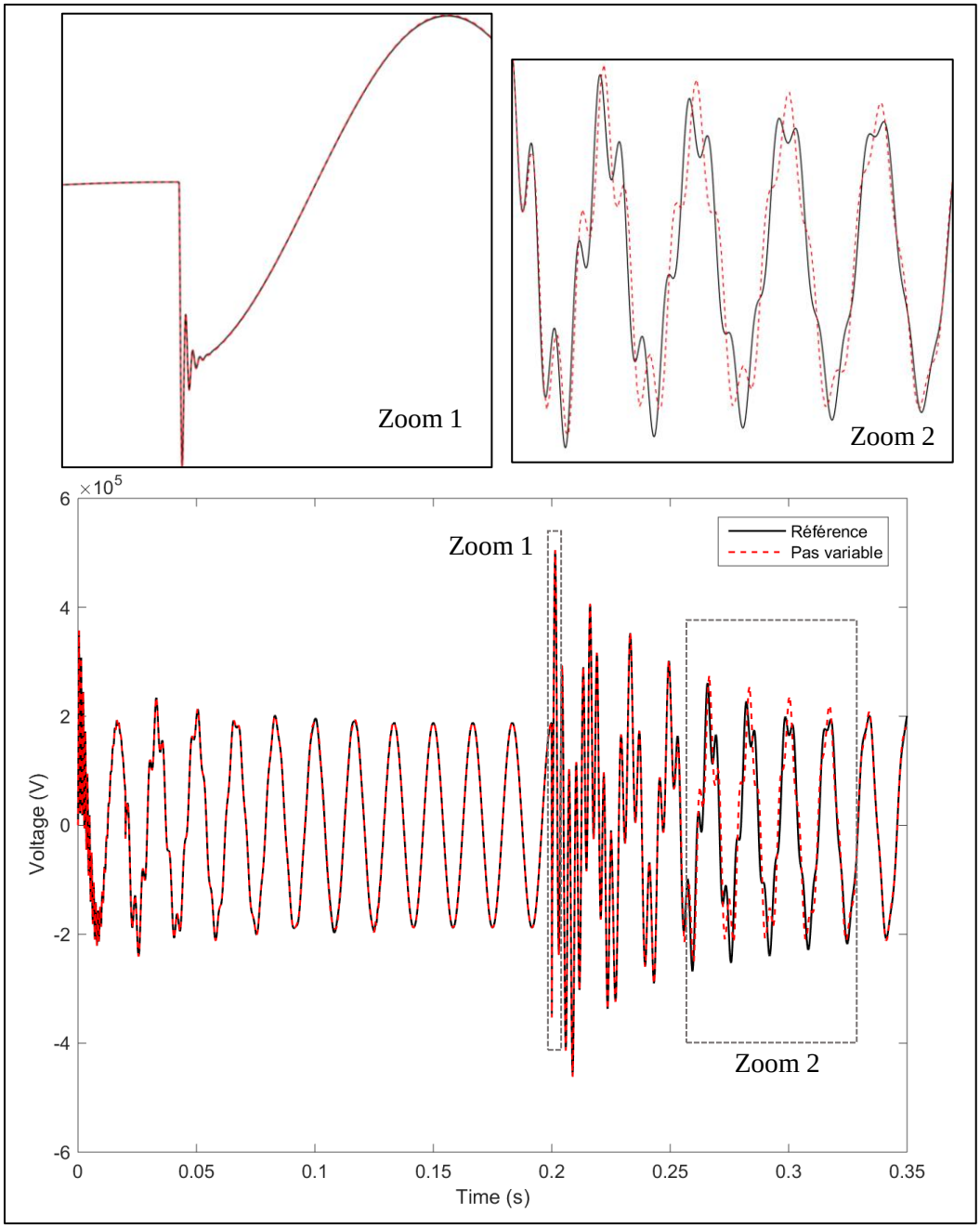


Figure 4.4 : Comparaison : pas variable par intervalle et pas fixe ($1\mu s$) – Tension aux bornes de Cs

La Figure 4.4 montre une comparaison des courbes de tension obtenues avec l'intégration à pas fixe (référence) et l'intégration à pas variable par intervalle. On observe que la qualité de l'onde est conservée sur la quasi-totalité de la simulation, même pour les très hautes fréquences. Vers 250ms cependant (zoom 2), on observe quelques différences dues aux transitoires encore présentes

lorsqu'on augmente le pas de temps. Pour corriger cela, il suffit de changer de pas de temps un peu plus tard à 300ms par exemple. Au niveau du temps de simulation, cet algorithme est **2 fois plus rapide** que la référence.

En somme, l'intégration à pas variable par intervalle offre de nouvelles possibilités de simulation. Elle permet de réduire sensiblement le temps de simulation tout en restant fidèle à la représentation des ondes. Implémentée dans un logiciel de simulation, elle engendrerait beaucoup d'ouvertures.

Cependant, dans ce premier algorithme, il faut connaitre en avance les intervalles de temps des transitoires et y spécifier un pas de temps faible pour obtenir de bons résultats. Cela n'est pas toujours facile surtout lorsqu'il y a beaucoup de non-linéarités. C'est la raison pour laquelle nous avons réalisé un algorithme automatique basé sur l'erreur de troncature pour choisir le pas de temps à utiliser.

## 4.3 Pas variable avec contrôle de l'erreur de troncature locale (LTE)

### 4.3.1 Présentation de l'algorithme

Dans cette section, on essaie de déterminer le pas de temps en fonction de l'erreur de troncature. Cependant, au lieu de l'exprimer directement à partir de cette dernière, ce qui entrainerait une variation continue du pas et des simulations interminables, on choisit plusieurs pas de temps, chacun associé à un intervalle de tolérance sur l'erreur de troncature. Ainsi, comme dans le cas du premier algorithme, les matrices d'admittance nodale et toutes les autres données relatives au pas de temps sont calculées et stockées en avance. L'erreur de troncature locale est évaluée tous les cents pas et sa valeur maximale en valeur absolue (sur l'ensemble des variables d'état du circuit) est comparée aux tolérances pour déterminer le pas de temps à utiliser. En plus de ces tolérances, à chaque fois qu'il y a une discontinuité (manœuvre d'un interrupteur, changement de pente d'un composant non-linéaire), on passe au pas de temps le plus petit (pendant 1000 pas pour les manœuvres d'interrupteur et 100 pas pour les non-linéarités), afin de bien représenter les courbes d'ondes pendant les transitoires. Aussi, une particularité de l'algorithme est que lorsqu'on augmente ou diminue le pas de temps, on passe au pas de temps le plus proche du pas actuel. En effet, l'erreur de troncature oscille. Si pendant l'évaluation on se situe au moment où elle croise le zéro, on est amené à croire qu'elle est faible quand bien même son amplitude est élevée. On

passerait ainsi au pas le plus grand sans raison valable. Afin d'éviter cela, l'augmentation ou la diminution du pas se fait de façon progressive.

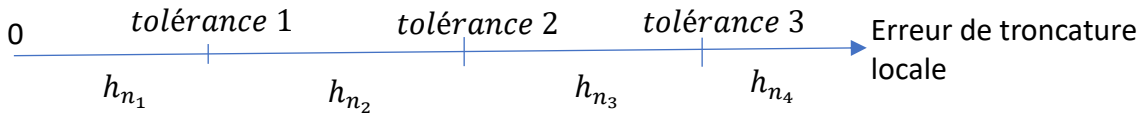


Figure 4.5 : Schéma explicatif de l'intégration à pas variable avec contrôle du LTE

On contrôle donc l'erreur de troncature mais de façon beaucoup moins sévère (pas de façon continue). Pour un pas fixe, on peut par exemple se douter que lors des transitoires, l'erreur de troncature est plus élevée que lors du régime permanent. En utilisant cet algorithme à pas variable, on joue donc sur l'erreur de troncature en fixant des tolérances qui permettent d'avoir un pas de temps plus petit lors des transitoires et un pas raisonnablement élevé en régime permanent.

### 4.3.2 Tests et comparaisons

Nous utilisons les mêmes circuits que dans la partie 3.3 afin de réaliser les tests.

La référence est toujours donnée par la méthode TRAP avec un **pas de temps de $1\mu s$**. La simulation à pas variable avec contrôle de l'erreur de troncature (LTE) utilise les paramètres ci-dessous :


Figure 4.6 : Paramètres de l'algorithme à pas variable avec contrôle du LTE

L'avantage est qu'ici on utilise les mêmes paramètres de tolérances pour tous les circuits. Ces tolérances ont été déterminées empiriquement. Pour ce faire, nous avons observé l'erreur de troncature locale durant la simulation du circuit avec les bancs de condensateurs (Figure 4.7) pour un pas fixe de $10\mu s$. Nous avons ensuite testé plusieurs séries de tolérances afin d'obtenir un bon ratio précision/rapidité. Ces valeurs (Figure 4.6) ne sont donc pas optimales, et sont basées sur un seul circuit. Dans l'avenir, il serait intéressant de les déterminer de façon plus générale en trouvant par exemple une façon de rendre l'erreur de troncature relative. En effet, si on considère un circuit mettant en jeu des valeurs plus petites que 1, alors l'erreur risque de ne pas dépasser 1, et la simulation se déroulerait entièrement avec le pas de temps le plus grand.

#### 4.3.2.1 Circuit 1 – bancs de condensateurs

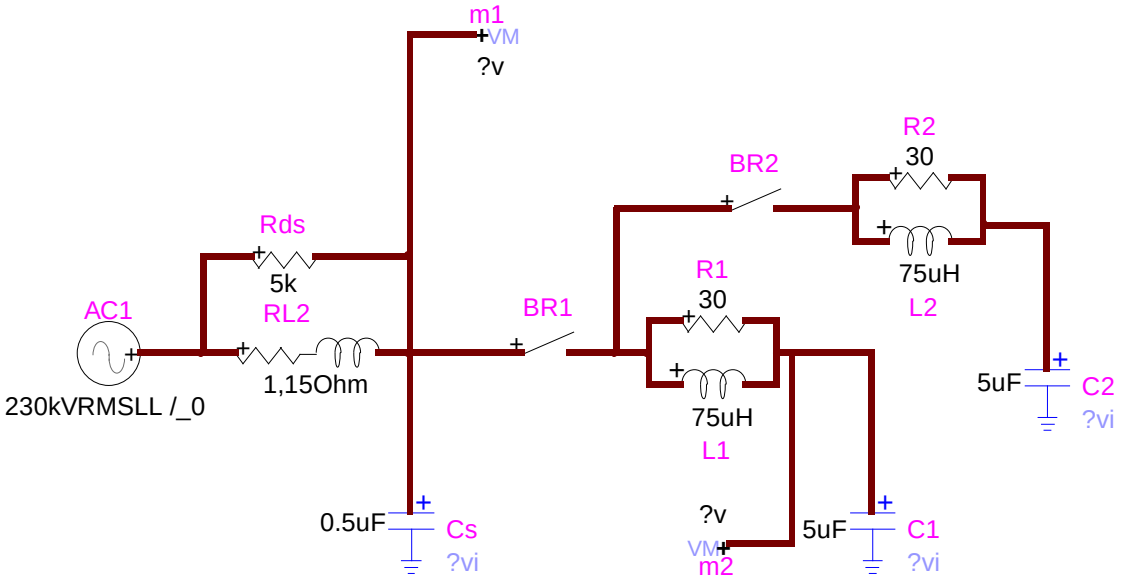


Figure 4.7 : Circuit de test avec bancs de condensateurs

La Figure 4.8 montre que la représentation des ondes est très fidèle comparée à la référence. Aussi bien les hautes fréquences que les très hautes fréquences sont très bien représentées. En outre, la simulation est **13.28 fois plus rapide** que la simulation à pas fixe (6.8069s vs 90.4267s). C'est donc encore plus rapide que l'algorithme à pas variable par intervalle, et plus précis aussi.

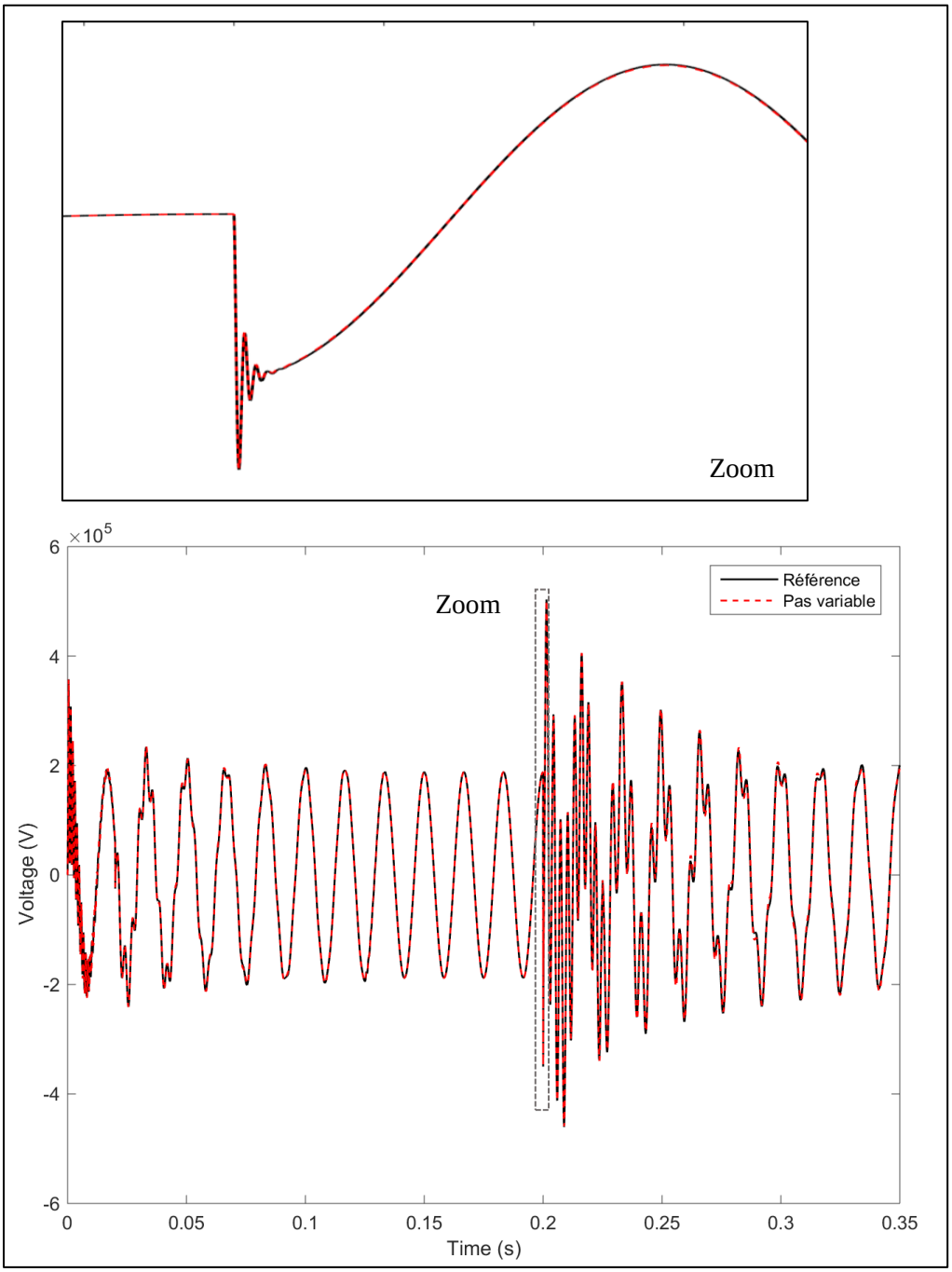


Figure 4.8 : Comparaison Pas variable avec contrôle du LTE et pas fixe ($1\mu s$) – bancs de condensateurs

Il est aussi réalisé une comparaison de la courbe obtenue avec un pas variable à celle obtenue avec un pas fixe de $10\mu s$ (Figure 4.9). Cela révèle une précision bien plus grande de la simulation à pas

variable, qui est par ailleurs **1.32 fois** plus rapide. L'algorithme à pas variable implémenté est donc très performant.

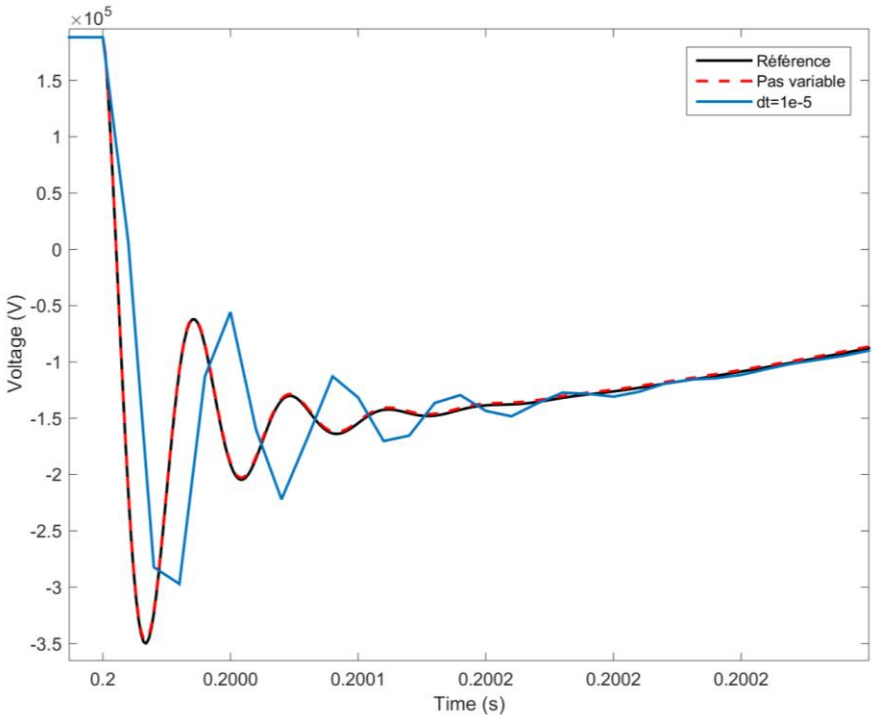


Figure 4.9 : comparaison des hautes fréquences de la tension aux bornes de Cs - pas variable avec contrôle du LTE et pas fixe de $10\mu s$

#### 4.3.2.2 Circuit 2 – défaut phase a terre

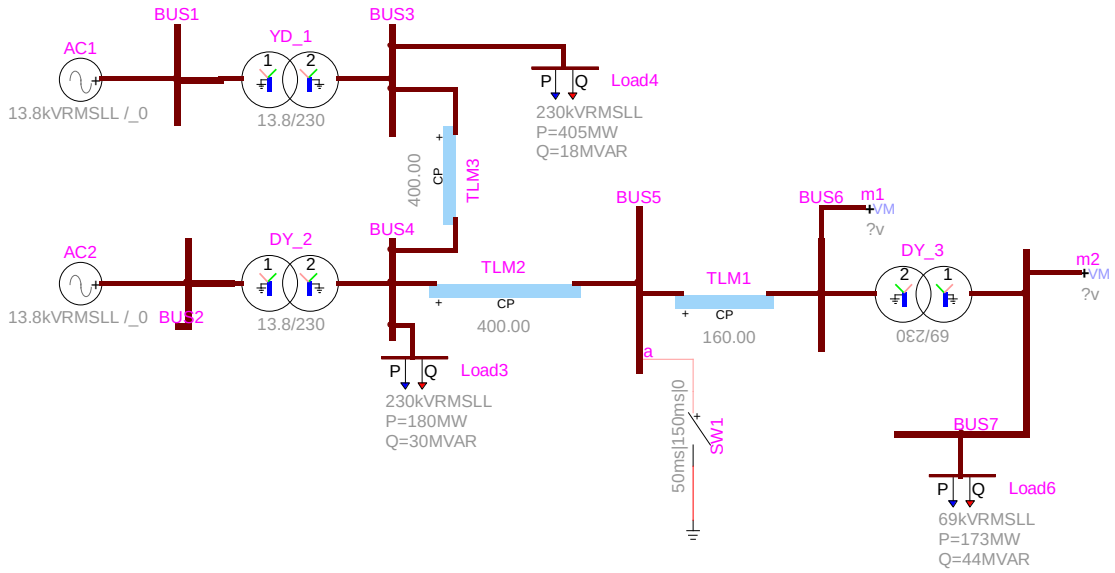


Figure 4.10 : Circuit de test avec défaut phase-terre

La Figure 4.11 montre que la simulation à pas variable intègre très bien les lignes. La qualité des ondes est conservée et le traitement des lignes se fait beaucoup plus rapidement ce qui entraîne un gain de temps considérable en régime permanent. En effet, la simulation à pas variable est **19.34 fois** (36.20s vs 700.04s) plus rapide que la référence. Cependant, lorsque les hautes fréquences ont une faible amplitude et durent longtemps, elles ont tendance à être amorties comme c'est le cas ici.

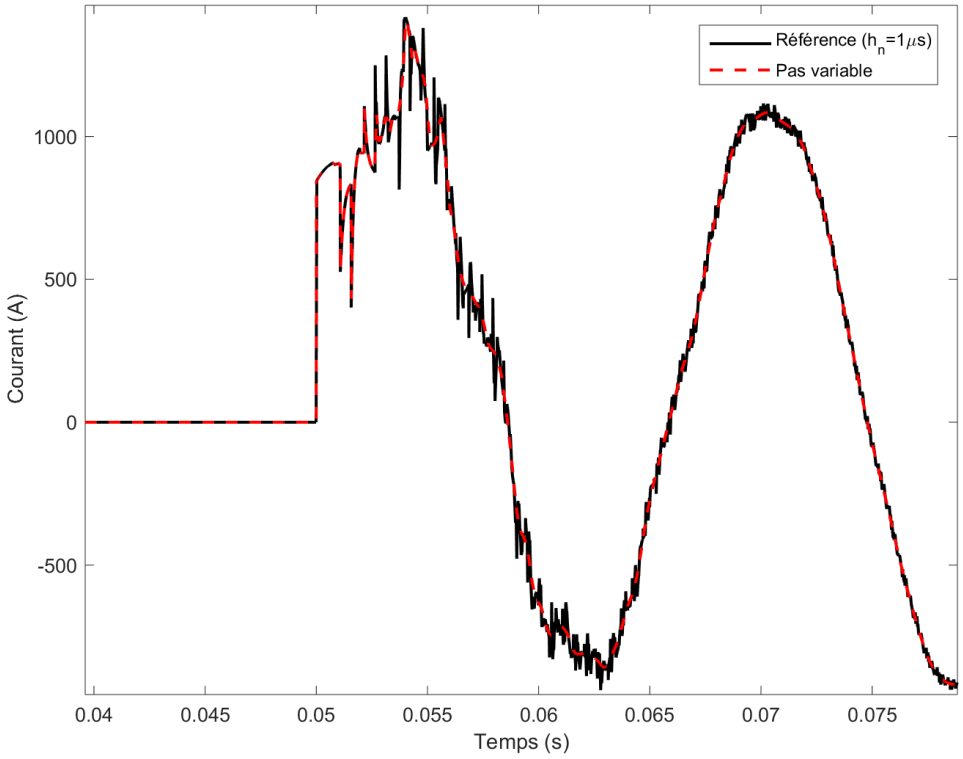


Figure 4.11 : Comparaison : pas fixe ($1\mu s$) et pas variable avec contrôle du LTE – Courant de défaut

#### 4.3.2.3 Circuit 3 – Ferro-résonance

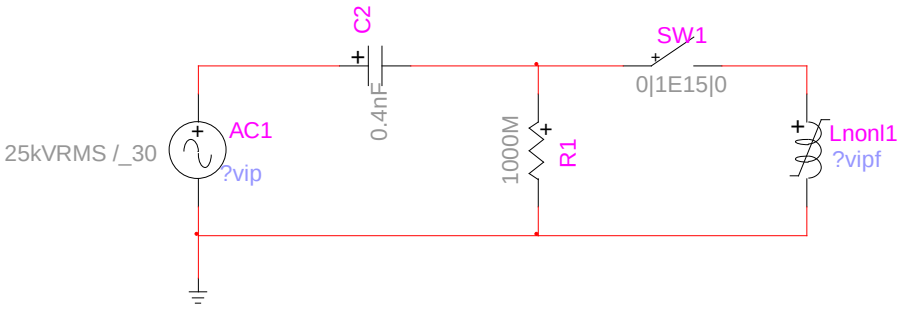


Figure 4.12 : Circuit de test avec non-linéarités

La simulation à pas variable est dans cet exemple **14.47 fois** (1.99s vs 28.80s) plus rapide que la référence ($1\mu s$). La qualité de l'onde est cependant quasiment la même. Cela signifie que la simulation est environ **1.45 fois** plus rapide que la simulation à pas fixe avec un pas de $10\mu s$ et beaucoup plus précise que cette dernière comme en témoigne la Figure 4.13. Cela montre encore une fois que la simulation à pas variable est une solution qui allie rapidité et précision. En présence

de très hautes fréquences sur de courtes durées ou encore de non-linéarités, ses performances sont encore plus notables.

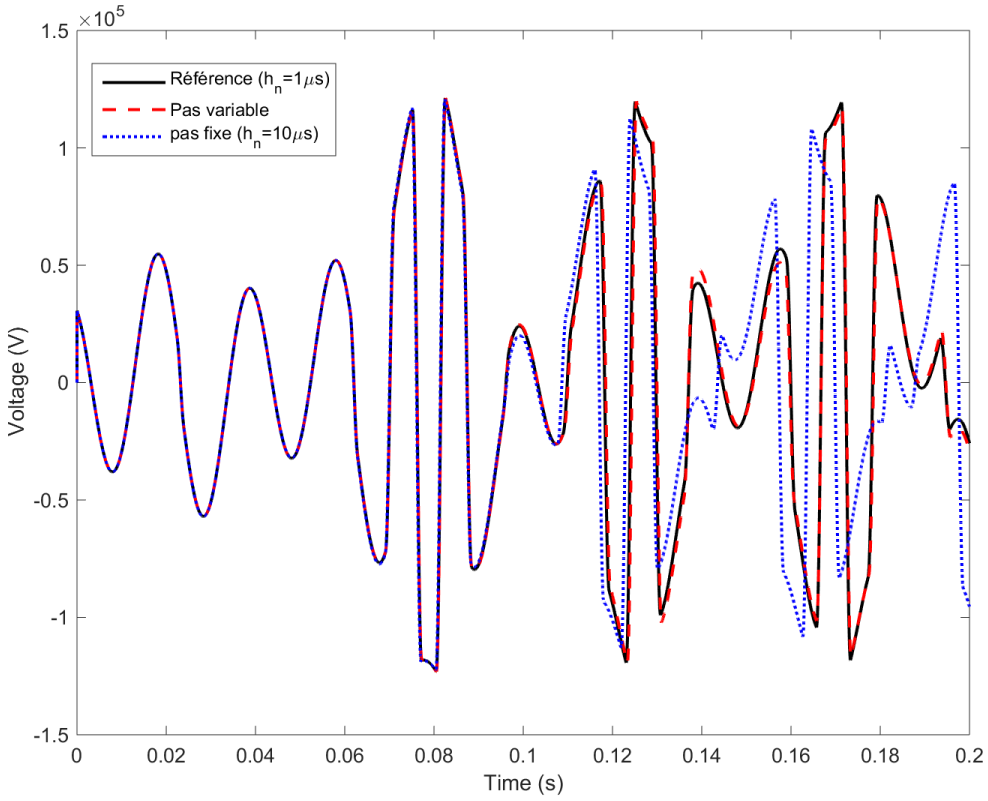


Figure 4.13 : Comparaison pas fixes $1\mu s$ et $10\mu s$ et pas variable

#### 4.3.2.4 Conclusion partielle

Globalement, grâce à des tolérances définies sur l'erreur de troncature locale, l'intégration à pas variable avec contrôle du LTE arrive à représenter fidèlement les courbes d'ondes sans perdre en précision, et en augmentant considérablement la rapidité de la simulation. Plus il y a de transitoires ou de hautes fréquences durables, plus ce rapport est diminué comme nous avons pu le constater entre les circuits 1 et 2.

Il s'agit donc d'un moyen efficace d'augmenter la rapidité générale des simulations. La prochaine étape à réaliser est l'intégration des modèles plus sophistiqués de contrôle tels que les différents composants des machines synchrones. Aussi, une étude plus poussée sur les tolérances sur l'erreur de troncature locale permettrait d'optimiser encore plus le ratio précision/rapidité de l'algorithme.

# CHAPITRE 5 CONCLUSIONS ET RECOMMANDATIONS

En définitive, nous avons développé une plateforme de simulation de réseaux électriques afin de tester et de comparer différentes méthodes d'intégration numérique. La plateforme a été développée sur Matlab et utilise l'analyse nodale modifiée augmentée pour résoudre les équations du réseau. Nous avons implémenté les méthodes trapézoïdale, Euler régressive, Gear 2$^{ème}$ ordre et 2S-DIRK, mais aussi des combinaisons de la méthode trapézoïdale avec Euler et trapézoïdale avec 2S-DIRK.

La méthode trapézoïdale est simple à implémenter et admet une bonne précision. Cependant, elle devient instable (oscillations numériques) lorsqu'il y a une discontinuité. C'est la raison pour laquelle elle est combinée à Euler dans EMTP. L'inconvénient de cette méthode est que Euler a une faible précision, donc la précision de la simulation est affectée s'il y a beaucoup de discontinuités.

Nous avons donc implémenté d'autres méthodes afin de trouver une méthode sans oscillations numériques, plus précise que la combinaison trapézoïdale et Euler, et avec une rapidité acceptable (en dessous de 1.1 fois la vitesse de simulation avec trapézoïdale et Euler).

La méthode trapézoïdale associée à 2S-DIRK répond à tous ces critères de précision et de rapidité. En effet, 2S-DIRK est beaucoup plus précise que Euler et est aussi stable. On constate à cause de son amortissement moins élevé que celui d'Euler, que dans certains cas, les oscillations numériques persistent sur quelques pas de temps si le pas n'est pas assez faible. Cependant, de façon générale, cette combinaison est plus précise que trapézoïdale associée à Euler, surtout en présence de non-linéarités.

Dans la deuxième partie de ce projet, nous avons implémenté dans la plateforme deux algorithmes d'intégration à pas variable, basés sur la méthode trapézoïdale. La première est basée sur une variation du pas par intervalles spécifiés par l'utilisateur. Elle permet de réduire sensiblement la durée de la simulation mais exige que l'utilisateur connaisse les périodes des transitoires afin d'y spécifier un petit pas de temps pour éviter de détériorer la qualité des ondes.

Le deuxième algorithme est automatique. Il se base sur l'erreur de troncature locale, et des tolérances définies empiriquement, afin de déterminer le pas de temps à utiliser au cours de la simulation. Nous avons vu que cette méthode permet de réduire considérablement la vitesse de la simulation (jusqu'à 15 fois la vitesse de l'intégration à pas fixe de $1\mu s$) et est très précise.

Les travaux de ce projet permettent d'une part de faire un état de l'art sur les principales méthodes d'intégration existantes, et de discuter leurs performances (rapidité-précision-stabilité) en les comparant sur des cas réels. Une telle plateforme permet en outre d'ajouter facilement des modèles supplémentaires, mais aussi d'implémenter de nouvelles méthodes d'intégration. Cela ouvre donc plein de possibilités quant à l'avenir des logiciels de simulation de réseaux électriques.

D'autre part, le développement d'algorithmes à pas variable qui diminuent sensiblement la durée des simulations sans détériorer la précision constitue aussi un grand pas pour les logiciels de simulation. Des travaux de recherche plus poussés doivent cependant être accomplis dans ce domaine afin d'inclure tous les modèles et de réaliser des analyses plus poussées sur des réseaux plus grands.

# BIBLIOGRAPHIE

[1] B. de METZ-NOBLAT, (2004). – Les calculs sur les réseaux électriques BT et HT, Extrait du Cahier Technique Schneider Electric n°23.

[2] J. Mahseredjian, A. Xémard & B. Khodabakhchian, (2008). Simulation des régimes transitoires dans les réseaux électriques. Éditions Les techniques de l'Ingénieur, 10.

[3] J. Mahseredjian, (2007). Régimes transitoires dans les réseaux électriques. Ed. Techniques Ingénieur.

[4] L. O. Chua & P. Y. Lin, (1975). Computer-aided analysis of electronic circuits: algorithms and computational techniques. Prentice Hall Professional Technical Reference.

[5] L. W. Nagel, (1975). "SPICE2: A computer program to simulate semiconductor circuits." ERL Memo ERL-M520.

[6] J. Mahseredjian, V. Dinavahi, & J. A. Martinez, (2009). Simulation tools for electromagnetic transients in power systems: Overview and challenges. IEEE Transactions on Power Delivery, 24(3), 1657-1669.

[7] G. G. Dahlquist, (1963). A special stability problem for linear multistep methods. BIT Numerical Mathematics, 3(1), 27-43.

[8] Mahseredjian, J., Dennetière, S., Dubé, L., Khodabakhchian, B., & Gérin-Lajoie, L. (2007). On a new approach for the simulation of transients in power systems. Electric power systems research, 77(11), 1514-1520.

[9] G. Allaire, (2005). Analyse numérique et optimisation, Editions de l'Ecole Polytechnique, Palaiseau.

[10] B. Kulicke, (1981). Simulationsprogramm NETOMAC: Differenzleitwertverfahren bei kontinuierlichen und diskontinuierlichen Systemen. Siemens Forschungs-und Entwicklungsberichte, 10(5), 299-302.

[11] J. Mahseredjian, (2017). Notes du cours ELE8457, Génie Électrique, École polytechnique de Montréal

[12] T. Noda, K. Takenaka & T. Inoue, (2009). Numerical integration by the 2-stage diagonally implicit Runge-Kutta method for electromagnetic transient simulations. IEEE Transactions on Power Delivery, 24(1), 390-399.

[13] X. Fu, S. M. Seye, J. Mahseredjian, M. Cai & C. Dufour, (2018). A Comparison of Numerical Integration methods and Discontinuity Treatment for EMT Simulations, en cours de publication.

[14] E. A. Celaya, & J. J. Anza, (2013). BDF-α: A multistep method with numerical damping control. Universal Journal of Computational Mathematics, 1(3), 96-108.

[15] M. Zou, J. Mahseredjian, G. Joos, B. Delourme & L. Gérin-Lajoie, (2006). Interpolation and reinitialization in time-domain simulation of power electronic circuits. Electric Power Systems Research, 76(8), 688-694.

[16] R. Alexander, (1977). Diagonally implicit Runge–Kutta methods for stiff ODE's. SIAM Journal on Numerical Analysis, 14(6), 1006-1021.

[17] T. Noda, T. Kikuma, & R. Yonezawa, (2014). Supplementary techniques for 2S-DIRK-based EMT simulations. Electric Power Systems Research, 115, 87-93.

[18] W. Gao, E. Solodovnik, R. Dougal, G. Cokkinides & A. S. Meliopoulos, (2003, February). Elimination of numerical oscillations in power system dynamic simulation. In Applied Power Electronics Conference and Exposition, 2003. APEC'03. Eighteenth Annual IEEE (Vol. 2, pp. 790-794). IEEE.

[19] N. S. Dattani, (2008). Linear multistep numerical methods for ordinary differential equations. arXiv preprint arXiv:0810.4965.

[20] F. L. Alvarado, R. H. Lasseter, & J. J. Sanchez, (1983). Testing of trapezoidal integration with damping for the solution of power transient problems. IEEE Transactions on Power Apparatus and Systems, (12), 3783-3790.

[21] R. Carbone, H. W. Dommel, R. Langella & A. Testa, (2002). Analysis and estimation of truncation errors in modeling complex resonant circuits with the EMTP. International journal of electrical power & energy systems, 24(4), 295-304.

[22] E. Gad, M. Nakhla, R. Achar & Y. Zhou, (2009). A-stable and L-stable high-order integration methods for solving stiff differential equations. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 28(9), 1359-1372.

[23] R. J. LeVeque, (2007). Finite difference methods for ordinary and partial differential equations: steady-state and time-dependent problems. Society for Industrial and Applied Mathematics.

[24] M. Zou, J. Mahseredjian, G. Joos, B. Delourme & L. Gérin-Lajoie, (2006). Interpolation and reinitialization in time-domain simulation of power electronic circuits. Electric Power Systems Research, 76(8), 688-694.

[25] J. R. Marti, & J. Lin, (1989). Suppression of numerical oscillations in the EMTP power systems. IEEE Transactions on Power Systems, 4(2), 739-747.

[26] U. M. Ascher, & L. R. Petzold, (1998). Computer methods for ordinary differential equations and differential-algebraic equations (Vol. 61). Siam.