Use of sinusoidal predictors for time domain simulation of AC power systems

Pierre-Marie Gibert

N°d'ordre NNT :
2018LYSE1275

THESE de DOCTORAT DE L'UNIVERSITE DE LYON
opérée au sein de
l'Université Claude Bernard Lyon 1
à
l'Institut Camille Jordan

Ecole Doctorale N° 512
Infomaths

Spécialité de doctorat : Mathématiques Appliquées

Soutenue publiquement le 30/11/2018, par :
Pierre-Marie GIBERT

Utilisation de prédicteurs sinusoïdaux pour la simulation temporelle de systèmes électriques en courant alternatif

---

Devant le jury composé de :

Bonnet, Catherine	Directrice de Recherche	INRIA Saclay Île-de-France	Présidente
Magoulès, Frédéric	Professeur des Universités	CentraleSupélec	Rapporteur
Van Cutsem, Thierry	Directeur de Recherche	FNRS	Rapporteur
Debit, Naïma	Maître de Conférence HDR	Université Lyon 1	Examinatrice
Bonnet, Catherine	Directrice de Recherche	INRIA Saclay Île-de-France	Examinatrice
Tromeur-Dervout, Damien	Professeur des Universités	Université Lyon 1	Directeur de thèse
Erhel, Jocelyne	Directrice de Recherche	INRIA Rennes	Co-directrice de thèse
Losseau, Romain	Ingénieur R&D	RTE	Invité

À mes proches

Remerciements

Je tiens tout d'abord à remercier mon directeur de thèse, Damien Tromeur-Dervout, pour son encadrement, son soutien et - il faut le dire - ses multiples relectures tout au long de ce travail de recherche. Des prémisses de ce dernier, lors de mon stage de fin d'étude à Polytech'Lyon, jusqu'à la soutenance, il a toujours su faire preuve d'une incroyable disponibilité et m'a ainsi permis de réaliser ce travail dans de bonnes conditions.
J'adresse également ma gratitude à ma co-directrice de thèse, Jocelyne Erhel, pour nos discussions très intéressantes et notamment ses conseils précieux lors de la phase de rédaction du présent manuscrit.

Mes remerciements vont ensuite à l'ensemble des membres du jury qui m'ont fait l'honneur d'évaluer mon travail de recherche. Je tiens ainsi à exprimer toute ma reconnaissance à Madame Catherine Bonnet qui a accepté de présider le jury et d'y participer en tant qu'examinatrice, Messieurs Frédéric Magoulès et Thierry Van Cutsem, rapporteurs, ainsi que Madame Naïma Debit, examinatrice. J'ai été particulièrement touché par l'attention que vous avez bien voulu porter à l'égard de mon travail, comme en ont témoigné, dans un premier temps, les rapports d'analyse du manuscrit puis, dans un second temps, nos échanges extrêmement enrichissants lors de la soutenance.

J'exprime également toute ma gratitude à mes tuteurs industriels, Romain et Adrien mais également François, pour leur immense implication dans l'encadrement de mon travail de thèse. Je ne saurai jamais suffisamment vous dire merci pour tout ce que vous avez fait pour moi qu'il s'agisse de conseils, de relectures, d'encouragements ou tout simplement de moments et discussions échangés ensemble.
Je tiens aussi à remercier Patrick Panciatici, mon responsable scientifique au sein de RTE, qui est à l'origine de ce sujet de thèse et avec qui les discussions, très riches, ont souvent été à l'origine d'avancées considérables ainsi que Florent et son prédécesseur Jean-Baptiste, responsables du pôle Intégration des Nouvelles Technologies de RTE, pour leur grande

attention et leur accompagnement afin que ma mission se déroule dans les meilleures conditions possibles.
J'en viens désormais à l'équipe des Dynawoïstes avec laquelle j'ai eu la chance de travailler au sein de RTE. Un énorme merci à tous, Marianne, Sébastien, Florentine, Gautier, Youssef et Pengbo pour votre aide, vos retours et plus généralement pour tous ces moments partagés ensemble. Mes pensées vont également aux Mickaël, au deuxième Youssef, à Medhi, Françoise ainsi qu'à l'ensemble des membres de RTE que je n'ai pas cités mais que j'ai eu la chance de côtoyer au cours de mon stage de fin d'études puis de cette thèse.

Pour les mêmes raisons, je tiens maintenant à exprimer ma gratitude à mes collègues de Polytech'Lyon, en particulier Fabien, Anissa, Christine, Stéphane, Ahmed, Fouzia, ainsi qu'Emmanuel Perrin, Jean-Louis Lefèvre, Fabienne Oudin, Jonathan Laroue, Jean-Pierre Benedetto et Marie-Thérèse Tchangodei.
Un grand merci à Amjad, pour nos discussions sérieuses (et moins sérieuses). Je te souhaite également le meilleur pour ta soutenance et la suite.
Je tiens également à remercier mes anciens professeurs de Polytech'Lyon, en particulier Anne-Laure et Thibault ainsi que Daniel Le Roux, Céline Vial et bien entendu Naïma Debit, pour les discussions très enrichissantes que j'ai pu avoir avec eux lors de mon cycle ingénieur et notamment pour leurs encouragements à me lancer dans cette aventure puis pour leur soutien pendant ladite aventure !
De même, je tiens à faire part de ma reconnaissance à Messieurs Luca Zamboni et Simon Masnou ainsi que Madame Véronique Maume-Deschamps, pour leur participation au suivi de mon travail de recherche.

Pour finir, je souhaite redire toute mon affection à mes proches. Parce que vous êtes toujours là pour moi, je ne saurai jamais vous remercier suffisamment. Pour votre amour, votre amitié, pour vos félicitations lorsque je réussis, pour vos encouragements lorsque je doute, pour votre tendresse, votre soutien lorsque j'ai l'impression de ne pas avancer mais aussi pour la force de vos mots lorsque j'ai besoin d'aide pour me relever. Supporter un thésard n'est pas de tout repos... surtout un comme moi ! Cette thèse, je vous la dédie :
À mon incroyable famille, à mes grands-parents, Pierre et Denise ainsi que Jean et Denise, à mes parents, Françoise et Philippe, à mes grandes sœurs, Cécile, Camille et Élise, à mon beau-frère, Jérémy, à mes super nièces et neveu, Aude, Perrine et Arthur, ainsi qu'à mon quasi-beau-frère, Florent.
À ma formidable compagne, Justine. À ma belle-famille, et en particulier ma belle-maman, Sylvie, et son conjoint, Denis, ainsi que toute la famille du Sud-Ouest.

Aux lyonnais les plus géniaux que je connaisse, Alexandre, Thomas, Valentin, Mathieu, Kévin, Adeline, et Philippe. Aux copains de HCE. À mes amis d'Orsay, Elvis, Maxime, Léo et Loan. À mes amis de toujours, Maryn, Hugo, Valentin et Nico. Aux sédélociens, Estelle, Sylvain, Mélanie et Jean-Mi.
MERCI À TOUS !

Je tiens également à remercier l'Association Nationale Recherche Technologie qui a soutenue financièrement cette thèse dans le cadre d'un contrat CIFRE (n°2015/0885).

Résumé

Dans les systèmes électriques modernes, une part croissante de l'électricité est produite à partir de sources d'énergies renouvelables, ce qui nécessite une adaptation des réseaux de transport d'électricité. L'introduction d'équipements d'électronique de puissance y est notamment de plus en plus marquée. Afin de permettre ces évolutions tout en garantissant le niveau requis de sûreté de fonctionnement à leurs clients, les gestionnaires de réseaux de transport d'électricité réalisent de nombreuses simulations dynamiques, visant à évaluer le niveau de robustesse du système électrique face à un certain nombre d'aléas. Historiquement, deux grandes classes de simulations étaient réalisées selon le type de phénomènes transitoires étudiés. Les simulations dites TS (Transient Stability) permettaient d'étudier les phénomènes transitoires électromécaniques, caractérisés par des constantes de temps et une propagation spatiale plus élevées. Au contraire, afin d'étudier les phénomènes transitoires électromagnétiques, caractérisés par des constantes de temps et une propagation spatiale bien plus faibles, on avait recours aux simulations dites EMT (ElectroMagnetic Transient). Dans la mesure où les évolutions actuelles du système électrique rendent poreuse la frontière entre ces deux grands domaines d'application, les simulations EMT ont finalement vocation à se généraliser à l'étude de systèmes de grande taille sur le moyen-voire le long-terme.

Cependant, ce type de simulation est pour le moment bien trop coûteux pour envisager de telles applications tout en conservant un contrôle fin sur la précision de la solution calculée, de par la raideur et la dimension des systèmes d'équations différentielles algébriques à résoudre. De plus, et il s'agit de la problématique de base de notre travail de recherche, le pas de calcul utilisé pour intégrer ces équations est fortement contraint par la fréquence d'oscillation des systèmes oscillants, ces derniers étant représentés à partir de leur forme d'onde complète dans les simulations EMT. Pour accélérer ces simulations, la plupart des approches actuelles se focalisent alors sur la modélisation, modifiant ainsi intégralement (e.g. phaseurs dynamiques) ou partiellement (simulations hybrides) le système d'EDA à résoudre à partir d'hypothèses fortes sur la solution. C'est pourquoi, dans la plupart

des outils de simulation, les aspects liés à la modélisation et à la résolution numérique se retrouvent souvent imbriqués.

Au contraire, l'approche proposée dans cette thèse, ayant pour objectif de contrôler finement l'erreur de calcul, repose sur un environnement de simulation découplant distinctement ces deux aspects. De plus, nous proposons l'utilisation d'un schéma d'intégration à pas adaptatif visant, d'une part, à garantir à l'utilisateur un certain niveau de précision et, d'autre part, à ouvrir la possibilité d'optimiser le coût de calcul des simulations. En effet, le pas de calcul étant directement lié à la dynamique de la solution simulée, un tel mécanisme d'ajustement permet alors de le raffiner lorsque le système est en régime transitoire, de sorte à capter les dynamiques étudiées, puis de l'augmenter de manière significative en régime permanent, afin de réduire le nombre d'itérations nécessaires à la simulation. L'objectif de la méthode des prédicteurs sinusoïdaux (SPM) proposée dans cette thèse est donc de tirer parti du comportement attendu des composantes oscillantes de la solution directement au sein du schéma d'intégration numérique afin d'en améliorer les performances, notamment en régime permanent où elles sont censées être proches de sinusoïdes à enveloppes et phases constantes oscillant à la fréquence nominale du système. En définitive, notre approche a pour vocation de tirer pleinement profit de l'utilisation de schémas d'intégration à pas adaptatif dans le cadre des simulations EMT.

D'un point de vue méthodologique, l'idée de base de la SPM est de décomposer la solution de manière additive en la somme d'une partie périodique et d'un terme de correction pour chaque intervalle d'intégration considéré. La première correspond au comportement sinusoïdal attendu de la solution en régime permanent. Il s'agit d'une fonction paramétrique, ne dépendant que du temps lorsque ses paramètres sont fixés. Ces derniers sont alors les coefficients de Fourier associés au mode fondamental, qui sont mis à jour par estimation paramétrique. La deuxième partie, i.e. le terme de correction, est quant à elle calculée à l'aide d'un schéma d'intégration à pas adaptatif. À chaque itération en temps, la méthode consiste tout d'abord à injecter la valeur locale des coefficients de Fourier afin de déduire le problème local sur le terme de correction à partir de l'EDA originale et de la fonction périodique locale. Ce problème est ensuite intégré à l'aide d'un schéma d'intégration classique, la mise du pas de calcul étant effectuée sur le terme de correction au lieu de la solution globale. Cette dernière est alors déduite en additionnant le terme de correction intégré et l'évaluation de la fonction périodique locale au pas de temps suivant. Pour finir, les coefficients de Fourier sont mis à jour par estimation paramétrique. Au final, l'objectif de la SPM est donc de capter autant que possible le comportement sinusoïdal attendu des

composantes oscillantes de la solution au sein de la partie périodique de sorte à amortir le terme de correction et ainsi pouvoir utiliser de grands pas de calcul pour l'intégration numérique, d'autant plus en régime permanent. C'est pourquoi le choix de l'estimateur a fait l'objet d'une attention toute particulière.

Afin de valider son potentiel industriel, la SPM a par ailleurs été implémentée au sein du solveur de référence SUNDIALS IDA puis interfacée avec Dynawo, un environnement de simulation développé par RTE étant basé sur le cadre de travail découplant le modeleur du solveur. Il nous a alors été possible d'implémenter des cas tests à l'aide du langage Modelica afin de valider puis comparer notre solveur IDASPM avec l'implémentation classique d'IDA. Les résultats obtenus nous ont montré que l'utilisation de la SPM permet d'importants gains de performances.

Dans la première partie de cette thèse, le problème mathématique ainsi que quelques approches existantes pour le résoudre sont introduits. Le chapitre 2 rappelle les systèmes d'EDA à résoudre dans le cadre des simulations dynamiques de systèmes électriques et leurs propriétés en régime permanent. La limitation du pas de calcul due à la présence de variables oscillantes est présentée de manière plus détaillée. Nous proposons notamment une estimation du pas de calcul maximum utilisable par la méthode des trapèzes en fonction de la tolérance et de la fréquence des variables simulées. Les propriétés attendues de la méthodologie développée sont également évoquées. Ensuite, dans le chapitre 3, nous présentons une revue bibliographique des méthodes existantes visant à accélérer la simulation de systèmes électriques en courant alternatif. Cet état de l'art couvre les approches consistant à reformuler le système d'EDA dans le domaine fréquentiel ainsi que les méthodes type simulation hybride permettant de coupler les simulations EMT et TS. Nous y présentons également quelques méthodes numériques.

La seconde partie de la thèse présente les aspects théoriques de la SPM. Dans le chapitre 4, la méthode est présentée de manière très détaillée. Nous y introduisons tout d'abord les concepts de notre méthode puis les deux grandes étapes de chaque itération en temps, à savoir l'intégration numérique du terme de correction et l'estimation paramétrique des coefficients de Fourier. Le choix de l'estimateur étant déterminant pour les performances globales de notre schéma numérique, nous y revenons de façon plus approfondie dans le chapitre 5. Ce dernier synthétise le cheminement suivi ayant permis le développement de l'estimateur retenu finalement. Cette partie se conclut avec le chapitre 6 qui présente les résultats obtenus à partir de notre analyse théorique de la SPM sur un cas scalaire

simple. Une attention particulière est portée sur le comportement attendu de la SPM en régime transitoire et permanent.

La dernière partie présente notre implémentation industrielle de la SPM et quelques résultats. Le chapitre 7 détaille le travail fait pour implémenter la SPM au sein du solveur de référence SUNDIALS IDA. L'interface avec le moteur de simulation développé par RTE, qui est basé sur l'environnement OpenModelica, est ensuite exposée dans le chapitre 8. Pour finir, le chapitre 9 présente les résultats obtenus sur des cas tests implémentés à l'aide du langage Modelica. En particulier, nous y comparons les performances obtenues avec et sans l'utilisation de la SPM en confrontant notre solveur IDASPM avec l'implémentation classique d'IDA. Le chapitre 10 conclut le présent manuscrit et propose quelques perspectives.

Contents

Introduction

1

Introduction

Contents

1.1 Motivations

Modern power systems are characterized by the smart-grids development and the renewable energy sources increasing proportion in the energy mix. In 2030, 40% of the French electricity should come from renewable energy sources in order to meet the European energy targets. As the underlying technologies are radically different from historically present electrical components, the transmission system has to be consequently adapted. Therefore, more and more new smart controls and power electronics devices are introduced into the transmission grid. In order to support these technological changes while providing the security of supply required level to their customers, transmission system operators (TSOs) have to assess the system reliability by performing numerous time-domain simulations. Time-domain simulations generally consist in studying the power system dynamical behavior during scenarios, which can be roughly seen as sequences of discrete events associated to perturbations and remedial actions (controls and human operations). By this way, the transmission grids robustness and especially the remedial actions efficiency in response to these perturbations can be evaluated.

In power systems, electromagnetic transients (EMT) phenomena have always been investigated. Such phenomena are mainly studied for designing system protections in order to enhance the system resilience to strong events, for example lightning strikes causing transient over-voltages. As most of the simulated phenomena are associated to very fast dynamics, these studies have been historically performed on short time intervals, i.e.

around 1 second. However, with the increasing penetration of intermittent renewable energy sources interfaced through power electronics into the existing infrastructures, more and more complex fast dynamics are to be anticipated. In addition, as the existing infrastructure contains many electromechanical components, whose inertia introduce slower dynamical behaviors, modeling fast dynamics by performing long-term EMT simulations is becoming crucial for assessing the entire system security.

However, this kind of simulation currently requires too large computational resources for considering long simulation duration, for instance during several minutes, while controlling the solution accuracy. Indeed, as power systems contain a wide variety of dynamics associated to different ranges of time scales, the differential algebraic equations (DAE) systems that arise when modeling power systems in the time-domain are particularly stiff. Indeed, on the one hand, modern components introduce electromagnetic transients phenomena, whose typical time range varies from the microsecond to a few milliseconds. On the other hand, historically present electrical components such as synchronous machines lead to electromechanical transients, whose typical time range varies from the millisecond to the minute. In addition, the study of these two kinds of phenomena can be less and less separated since electromechanical components can be dramatically affected by electromagnetic phenomena. In addition, as EMT models take into account all the system dynamics (a larger number of state variables must be considered, ideally in unbalanced conditions), the underlying equations systems dimension can be roughly at least one order of magnitude larger. As a result, each time step of the integration process potentially has an important computational cost. To finish, most of the existing EMT simulation tools are based on fixed step size strategies for integrating the DAE system. In order to catch accurately the fastest dynamics, the step size is generally fixed to a very small value, such as 10μs. In the current implementations, the computation time is reduced using decomposition approaches and parallel computing taking advantage of the propagation delays on long power lines, but these approaches cannot offer any guarantees on the solution accuracy; they must be properly customized using trials and errors in order to find the proper fixed time step and how to decompose the system (what is a long powerline?). This could be considered as an acceptable practice for protection design studies but certainly not for operational decisions making processes close to real time or to study future complex hypothetical systems. Henceforth, solving DAE systems associated to the EMT models of large-scale power systems with some guarantees on the solutions accuracy requires a considerable number time steps whose individual computational cost is important.

As the time and space propagation of EMT through the transmission grid has been

historically assumed to be very limited, EMT simulations are generally done on a small part of the system and during very short time intervals. This approximation is justified by the fact that the system is a low pass filter and damps the fast dynamics, in particular via the electromechanical components inertia. The fast dynamics are not traveling very far in the system, if they are stable and if they don't trigger protection devices actions. On the contrary, as the space and time propagation of slow electromechanical transients is very important (e.g. inter-zone oscillations), the associated studies are conducted on large-scale systems. Then, this currently results to the use of different models and so of different simulation tools depending on the dynamics to study and implicitly assuming that the neglected dynamics are stable:

- Electromechanical transient phenomena are simulated from so-called Transient Stability (TS) models. These models are based on numerous assumptions which enable faster computations. In particular, they generally represent three-phase systems in balanced conditions and in the frequency domain, focusing on their envelope. In other words, inner-cycle dynamics are neglected as they are considered as stable and instantaneous (singular perturbation).
- EMT tools are used to study very fast phenomena. They model electrical components with a much finer accuracy than TS models and, especially, do not make the above-mentioned assumptions. As the underlying models represent the three-phase systems in full-waveform, inner-cycle dynamics can be considered. This leads to the use of much smaller time steps.

Then, as the penetration of new technologies is getting important, choosing a priori the appropriate tool and relevant modeling level (which part of the system to consider, which time window, which boundary conditions, ...) depending on the phenomenon to study is getting harder and harder. Hence, as EMT models are much more detailed and could thus enable to simulate all the investigated dynamics, our objective is to perform mid-to-long-term large-scale power systems simulations with full-EMT models.

Furthermore, in most of the existing simulation tools, the modeling and solving aspects are combined. For instance, mathematical equations defining the components dynamical behavior are often implemented in discretized form: trapezoidal formula for smooth dynamics and backward Euler formula for discontinuities [43]. Although it enables faster computations, such an approach leads to numerous heuristics and the presence of solver parameters in physical models. It is consequently impossible to change the solving strategy and to assess or compare the simulations accuracy, which leads to question the simulation results reliability. More precisely, the DAE systems are generally solved with fixed step size integration schemes which do not really enable to control the computational error.

Therefore, in order to meet some accuracy requirements, the time step is fixed to very small values (e.g. 10 μs), which completely prevents from long-term simulations. To tackle this constraint, most of the existing approaches consist in simplifying the DAE system to solve by reducing either the simulated solution frequency or the EMT system dimension. However, as such approaches directly affect the model by modifying the original equations to solve in a non-equivalent way, controlling the resulting computational error seems difficult.

The approach proposed in this thesis is based on two cornerstones:

1. Completely separating the modeler from the solver. As shown in figure 1.1, from a mathematical point of view, the idea is that the modeler only builds the DAE system, which is solved in a second time by the numerical integrator. In particular, the model does not contain any solver data. On the contrary, the solver can contain model-related data such as the oscillating variables index and pulsation, which is provided by the modeler. However, the solver should not interfere at all with the modeling process, i.e. the corresponding data flux is unidirectional. As the model is not affected by the solver choice, state-of-the-art methods can more easily be tested within the solver. To finish, such an approach simplifies the simulation results analysis as the modeler and solver respective roles can be clearly distinguished.
2. Integrating DAE systems with an adaptive step size solver, in which the time step size is adjusted so that the integration error meets a tolerance target. By this way, the simulation results reliability is reinforced. Furthermore, the step size adjustment makes the computational cost optimization possible. Indeed, as the step size is directly linked to the solution dynamics (the faster its variations, the smaller the step size), it enables to catch fast transient phenomena when the system is subject to perturbations while reducing the computational cost when the system returns to steady-state.

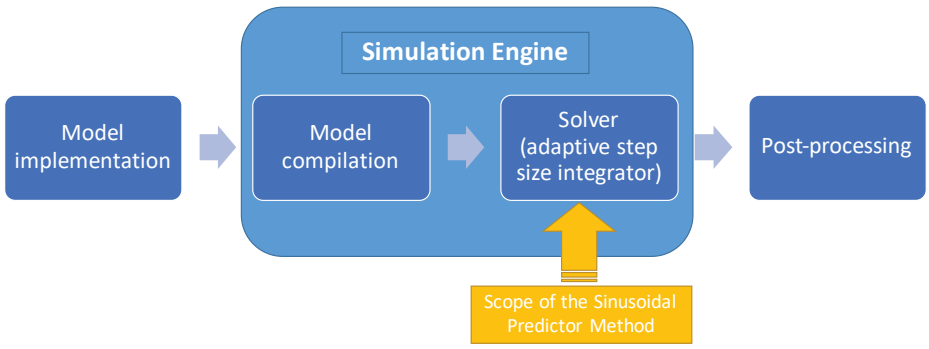

Figure 1.1: Proposed approach: fully separated modeler/solver framework

However, as the three-phase voltages and currents full-waveform is simulated in EMT simulations, using an adaptive step size strategy is generally inefficient because of their sinusoidal behavior. More precisely, their frequency limits the time step size that is used for the integration. Paradoxically, this limitation is finally due to an expected behavior which could be taken into account. That's why the proposed solver aims at exploiting this property in order to increase the step size and so the performances in steady-state, while being able to finely catch transients phenomena. In other words, our objective is to exploit the full potential of adaptive step size schemes in the EMT simulations context.

Hence, in the developed Sinusoidal Predictor Method (SPM), the solution is decomposed as the sum of a periodic part and a correction term. The former corresponds to the expected steady-state oscillating solution, i.e. a sinusoid oscillating at the nominal frequency. It is a parametered function of the time, whose parameters are the Fourier coefficients associated to this fundamental mode. It is updated using parametric estimation. The latter, i.e. the correction term, is integrated by an adaptive step size solver. At each time step, the DAE system is rewritten on the correction term by injecting the solution periodic part and solved with a predictor-corrector scheme. The step size adjustment is then performed on the correction term. Thus, the SPM aims at catching as much as possible the solution smooth sinusoidal behavior into the periodic part. That's why a special attention has been given on the estimator choice. By this way, as the solution oscillating parts should tend toward sinusoids with constant Fourier coefficients in steady-state, the associated correction term would be damped. As a result, the simulated variable would be reduced to a constant term (zero for the solution oscillating parts and their asymptotic values for its non-oscillating parts) and much larger time steps could be used by the solver.

To asses its industrial potential, this method has been implemented into the reference solver SUNDIALS IDA [34] and interfaced with RTE's simulation engine for time-domain simulations [25]. The latter is based on the OpenModelica [19] environment and enables to set our simulations. Thus, using the simulation engine, it was possible to implement EMT models in Modelica language [20] from an internally-implemented library. Finally, our customized solver has been validated and compared with IDA's basic implementation.

1.2 Contents

In this thesis first part, the mathematical problem and some existing approaches for solving it are introduced. Hence, chapter 2 presents the DAE systems to solve in time-domain simulations of power systems and their steady-state properties. It also further

develops the step size limitation by the oscillating variables frequency and some expected properties of our methodology. Then, in chapter 3, we present a bibliographic review of existing approaches for accelerating the AC power systems simulation when the frequency is known a priori. This covers methods modifying the entire model by rewriting it in the frequency domain and some current methods for hybrid simulations coupling EMT and TS simulations.

Then, in the second part, different SPM theoretical aspects are introduced. First, we present the Sinusoidal Predictor Method (SPM) in chapter 4, which details the method concept and different steps, i.e. the correction term integration by an adaptive step size solver and the periodic term update by a parametric estimator. The estimator choice is more precisely discussed in chapter 5. In particular, it presents the developments made to design the finally retained estimator. This part is concluded with chapter 6 which presents some SPM theoretical validation results on a simple ODE. A special attention is given on the method behavior in steady-state and during transients.

To finish, the third part focuses on the SPM implementation and obtained results. Chapter 7 details our work for integrating the SPM into the reference solver SUNDIALS IDA. The interface with the OpenModelica-based RTE's simulation engine for performing time-domain simulations is presented in chapter 8. Then, the chapter 9 shows results obtained with the SPM on several test cases that have been implemented with Modelica and simulated with this implementation. In particular, we compare our customized solver IDASPM performances with those of the classical solver IDA.

1.3 Publications

1. Gibert P.-M., Panciatici P., Tromeur-Dervout D., Beaude F., Wang P. and Erhel J. (2017). A Generic Customized Predictor Corrector Approach for accelerating EMTP-like simulations. IEEE PES PowerTech 2017.
2. Gibert P.-M., Panciatici P., Losseau R., Guironnet A., Tromeur-Dervout D. and Erhel J. (2018). Speedup of EMT simulations by using an integration scheme enriched with a predictive Fourier coefficients estimator. IEEE PES ISGT 2018
3. Gibert P.-M., Losseau R., Guironnet A., Panciatici P., Tromeur-Dervout D. and Erhel J. (2018). Use of the Sinusoidal Predictor Method within a fully separated modeler/solver framework for fast and flexible EMT simulations. SIMULTECH 2018

Part I

Problem and existing approaches

2

Mathematical problem

Contents

As presented in the general introduction, this research work objective is to accelerate the time-domain simulations of AC power systems in order to pave the way for long-term EMT simulations. This chapter presents the corresponding mathematical problem. To begin, DAE systems arising in EMT simulations are presented. As our methodology mainly aims at enhancing the integration method performances by using larger time steps in steady-state, the system properties in these conditions are then discussed. In the third section, the step size limitation affecting classical numerical schemes due to the presence of oscillating variables is explained in a more formal way with its mathematical justification. To finish, we present the expected properties and behavior of our method.

2.1 Time-domain simulation of AC power systems

In the time-domain simulation of power systems, the mathematical problem to solve is generally a set of differential algebraic equations [63]. In its most general form, this system can be written as a full-implicit equation [29], i.e.

$$F(t, X(t), \dot{X}(t)) = 0 \tag{2.1}$$

Let d be the system dimension. In this equation $t \in \mathbf{R}$ refers to the time, $X : \mathbf{R} \to \mathbf{R}^d$ is the solution and $F : \mathbf{R} \times \mathbf{R}^d \times \mathbf{R}^d \to \mathbf{R}^d$ is the DAE system residual function. In most of

the power system application, this system can be rewritten in semi-explicit form [4] i.e.

$$\begin{cases} \dot{x}(t) & = f(t, x(t), y(t)) \\ 0 & = g(t, x(t), y(t)) \\ (x(t_0), y(t_0)) & = (x_0, y_0) \end{cases} \tag{2.2}$$

where

- $x : \mathbf{R} \to \mathbb{R}^{d_x}$ are the solution differential state components. Let $\mathcal{I}_x$ be the associated index set.
- $y : \mathbf{R} \to \mathbb{R}^{d_y}$ are the solution algebraic state components. Let $\mathcal{I}_y$ be the associated index set.
- $f : \mathbb{R} \times \mathbb{R}^{d_x} \times \mathbb{R}^{d_y} \to \mathbb{R}^{d_x}$ is the differential equations Right-Hand-Side function.
- $g : \mathbb{R} \times \mathbb{R}^{d_x} \times \mathbb{R}^{d_y} \to \mathbb{R}^{d_y}$ is the algebraic constraints function.
- (x_0, y_0) are consistent initial conditions. It is generally computed from a dedicated initialization procedure and corresponds to the DAE system steady-state solution.

In this system, some variables that we will denote by the s subscript are oscillating. They are thus assumed to be sinusoidal in standard operations, i.e. when the system is not subject to transients. Let d_s be the number of oscillating variables and $\mathcal{I}_s$ the associated index set. On the contrary, let d_{ns} be the number of non-oscillating variables that we will denote by the ns subscript. The index set $\mathcal{I}_s$ is known a priori and generally corresponds to network electrical quantities such as three-phase currents and voltages. Their frequency is supposed to be close to the system nominal frequency.

2.2 System steady-state properties

In steady-state, the transmission system is designed for the three-phase currents and voltages to be as close as possible to balanced three-phase sinusoids. In other words, each three-phase signal should be a system of three centered sinusoids oscillating at the nominal frequency (for instance 50Hz for the Continental Europe) with equal amplitudes and phases with a $\frac{2\pi}{3}$ offset, as illustrated in 2.1:

$$X^{\infty}_{s,abc}(t) = \begin{bmatrix} A_\infty \cos(\omega t + \phi_\infty) \\ A_\infty \cos(\omega t + \phi_\infty - 2\pi/3) \\ A_\infty \cos(\omega t + \phi_\infty + 2\pi/3) \end{bmatrix} \tag{2.3}$$

The s, abc subscript means that we consider a three-phase oscillating component and the ∞ superscript refers to the steady-state solution. A_∞ and ϕ_∞ are respectively the steady-state amplitude and phase of $X_{s,abc}$.

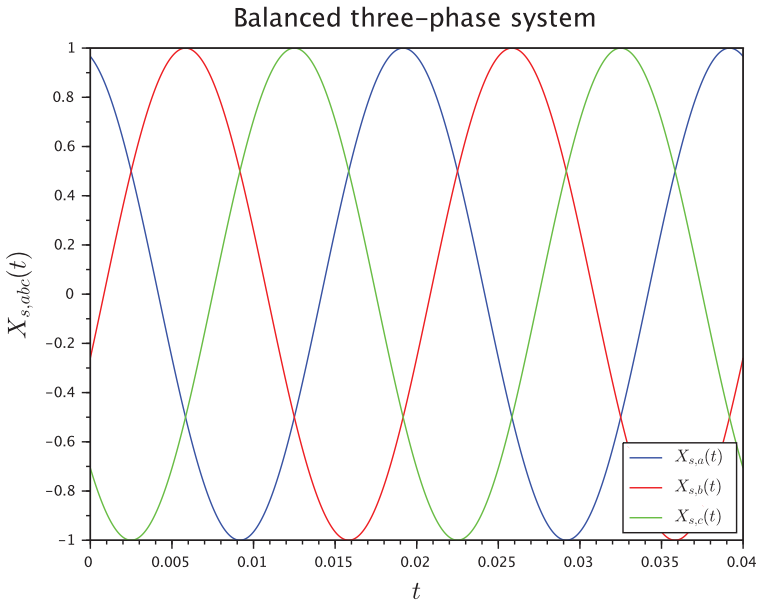

Figure 2.1: Illustration of a balanced three-phase system (on the abscissa: time in seconds, on the ordinate: three-phase system in arbitrary units).

Then, each component a, b or c should be in the form:

$$X_s^\infty(t) = u_\infty \sin(\omega t) + v_\infty \cos(\omega t) \tag{2.4}$$

where u_∞ and v_∞ are the steady-state Fourier coefficients associated to the three-phase signal fundamental mode. Hence, the power system is assumed to not contain second or higher order harmonics in steady-state. As these asymptotic Fourier coefficients are constant, the oscillating solutions time-derivative is given by

$$\dot{X}_s^\infty(t) = \omega(u_\infty \cos(\omega t) - v_\infty \sin(\omega t)) \tag{2.5}$$

In addition, the non-oscillating components should be constant, i.e.

$$X_{ns}^\infty(t) = X_{ns}^\infty \tag{2.6}$$

thus

$$\dot{X}_{ns}^\infty(t) = 0 \tag{2.7}$$

Finally, by injecting (2.4), (2.5), (2.6) and (2.7) into (2.1), we see that the DAE system should respect the following equation in steady-state:

$$F\left(t, \begin{bmatrix} u_{s,\infty} \sin(\omega t) + v_{s,\infty} \cos(\omega t) \\ X_{ns}^\infty \end{bmatrix}, \begin{bmatrix} \omega(u_{s,\infty} \cos(\omega t) - v_{s,\infty} \sin(\omega t)) \\ 0 \end{bmatrix}\right) = 0 \tag{2.8}$$

2.3 Step size limitation with classical EMT methods

Even when the system is in steady-state, the time step used by numerical integration algorithms is limited by the presence of oscillating variables and, more precisely, by the alternating currents frequency [13] (the higher their frequency, the lower the numerical integration performances). As precised in the previous section, the power system is designed such as there is no harmonic of the nominal frequency in standard operations. It means that, in steady-state, this step size limitation is finally due to the presence of expected sinusoidal variables whose frequency is equal or close to the system nominal frequency. Despite the high predictability of this constraint and especially in steady-state when using larger step sizes should be possible, classical EMT methods do not enable to get around it.

In order to illustrate this point, let us consider the equation below which corresponds to the simplest strategy for controlling the step size [47]

$$h_{new} = k_s \sqrt[q+1]{\frac{TOL}{e_{n+1}}} h_{old} \tag{2.9}$$

where $k_s \in]0,1[$ is a security coefficient, q is the integration scheme order (for instance $q = 2$ for the trapezoidal formula), TOL is the tolerance on the local truncation error that is set by the user and e_{n+1} is an estimation of the local truncation error. h_{old} and h_{new} are respectively the step size before and after the adjustment. In other words, the estimated local truncation error should be at most equal to the tolerance for the step size to be validated.

Now, if we consider a second-order method (e.g. the trapezoidal formula), its local truncation error is given by

$$e_{n+1} = \mathcal{O}(h_n^3) \approx \frac{h_n^3}{12} ||X_n^{(3)}|| \text{ for the trapezoidal formula} \tag{2.10}$$

where $X_n^{(3)}$ refers to the third time-derivative of X_n. Then, if we simulate a per united sinusoidal variable (i.e. whose amplitude is equal to $||X_n|| = 1$) oscillating at the Continental Europe reference frequency (i.e. $\omega = 100\pi rad/s$), an approximation of the maximum step size with the trapezoidal formula is

$$h_n^{max} = \sqrt[3]{\frac{12TOL}{\omega^3 ||X_n||}} = \sqrt[3]{\frac{12TOL}{10^6 \pi^3}} \tag{2.11}$$

Tolerance	1.e-3	1.e-4	1.e-5	1.e-6
Estimated maximum step size	7.29e-4	3.38e-4	1.57e-4	7.29e-5

Table 2.1: Estimated maximum step sizes for several tolerances on the local truncation error.

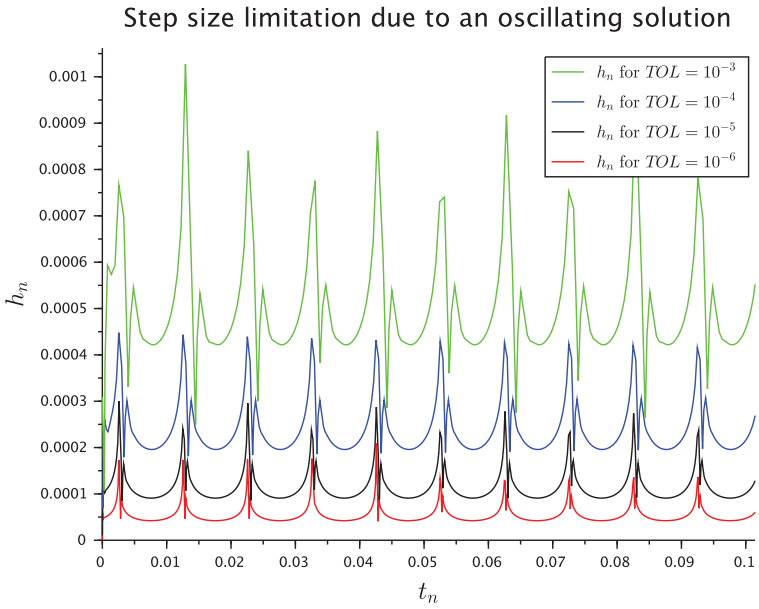

Figure 2.2: Illustration of the step size limitation due to the integration of an oscillating solution (on the abscissa: time in seconds, on the ordinate: step size in seconds).

In table 2.1 and figure 2.2, some corresponding maximum step size estimations and simulation curves are presented for different solver tolerances. Such step sizes prevent from simulation durations comparable to those of transient stability simulations. In classical TS studies, investigated time scales are generally in the range of minutes [63], while the used step sizes in EMT simulations do not exceed $100\mu s$. It means that several millions of time steps would be necessary for conducting mid-term EMT studies. In addition, as EMT models are much more detailed than TS models, the individual computational cost of each time step is dramatically greater [39].

2.4 Implemented method expected properties

Even if our objective is to accelerate time-domain simulations of EMT phenomena, such simulations require the use of a global methodology that enables a fine control of the computational error [58]. The latter results from, on the one hand, the approximations

made on the model representing the simulated system and phenomena and, on the other hand, the solver numerical errors [65]. Thus, from the users point-of-view, controlling the error should be feasible by modifying the model assumptions or the solver tolerance. The only counterpart would be a possibly significant increase of the required computational time and resources. That's why the entire simulation process has to be as transparent as possible.

In order to finely control the integration error, the implemented method should respect some requirements that are further developed in chapter 6:

- Solving the original DAE system (2.1) (or (2.2)) and particularly not modifying it in order to accelerate the simulations: if the DAE system is rewritten in a non-equivalent way at the continuous level, the reference to the basic equations is lost. Then, nothing can ensure that the error control process is applied to the actual solution.
- Using an adaptive step size strategy. By this way, the integration time step is adjusted so that the error remains close to a chosen tolerance [58]. Indeed, as the integration error depends on the simulated variables dynamics [28], choosing an appropriate fixed step size is complicated. It is all the more difficult as power systems lead to very stiff DAE systems [4]. In addition, thanks to the adaptive step size, the integration process can be significantly optimized [58] since the step size is adjusted so that fast dynamics are accurately approximated during transients by using small time steps while the computational cost can be drastically reduced in steady-state by using large time steps.
- A-stability [29] and, in particular, A-instability in order to respect as much as possible the simulated signals dynamics. Indeed, as time-domain simulations are mainly used for security assessments in our context, unstable phenomena should not be damped by the solver.

3

Review of the relevant literature

Contents

In order to accelerate the EMT simulations of power systems, existing approaches mainly focus on the modeling aspects. Two strategies are generally implemented:

1. Using larger integration step sizes by reducing the simulated oscillating variables frequency. In order to do this, the DAE system associated to these variables waveform is rewritten on their envelope. In this chapter, we briefly present dynamic phasors and some related approaches. Then, we highlight the Time Domain Transformation which could almost be seen as the corresponding solver approach, initially proposed for purely oscillating ordinary differential equations, with our extension to differential algebraic equations with both oscillating and non-oscillating components.
2. Reducing the EMT model dimension by coupling it with a TS model, also referred as hybrid simulations. The two kinds of coupling are presented i.e. with a geographical slicing, which is the most investigated in the literature, and with a temporal slicing, which has been more recently proposed. In particular, we will show that choosing

a priori an appropriate interface between the EMT and TS model is the common difficulty of such approaches.

3.1 Frequency-domain modeling

As above mentioned, the first strategy consists in rewriting the DAE system on the oscillating variables waveform into a system on their envelope for tackling the step size limitation [13]. Such approaches are currently the most investigated within the power system community as they really focus on simulation modeling aspects. Especially, as most of the used simulators were based on phasor models for TS studies, their implementation would only consist in extending existing models. However, as we will see, the problem is that these kinds of approaches rely on high hypotheses on the solution which may be inappropriate in the system design context.

3.1.1 Dynamic Phasors

Such kind of approach has been initially implemented within classical TS simulation tools using so-called phasor models. In phasor models, oscillating solutions are represented by centered complex variables with piecewise-constant amplitude and phase, i.e.

$$X_s(t) = \Re\left\{Ae^{j(\omega_s t+\phi)}\right\} = \Re\left\{\underbrace{Ae^{j\phi}}_{\stackrel{def}{=}\bar{X}} e^{j\omega_s t}\right\} \tag{3.1}$$

where A and ϕ are respectively the phasor $\bar{X}$ amplitude and phase. ω_s is the power system reference pulsation. As a consequence, if the frequency of $X_s(t)$ is close to those of the system, the simulated phasor $\bar{X}$ frequency is close to zero. This approach is illustrated in figure 3.1. In addition, as this modeling is used for quasi-steady-state studies, fast variations are commonly neglected, i.e. $\dot{A} = 0$ and $\dot{\phi} = 0$. This assumption was acceptable for TS studies but not for EMT studies. That's why, the idea of dynamic phasors has been to extend this approach by adding the phasor time variation and higher harmonics to the modeling [13].

In general, a real or complex valued periodic signal $\tilde{X}_s$ with period T can be expressed with a Fourier series representation given by (3.2) :

$$\tilde{X}_s(\tau) = \sum_{k=-\infty}^{\infty} \bar{X}_k e^{jk\omega_s\tau} \tag{3.2}$$

where $\omega_s = \dfrac{2\pi}{T}$ and $\bar{X}_k$ is the k-th Fourier coefficient which is invariant in time. This Fourier coefficients can be referred to as k-th order phasors. In [56] Sanders & Al extended

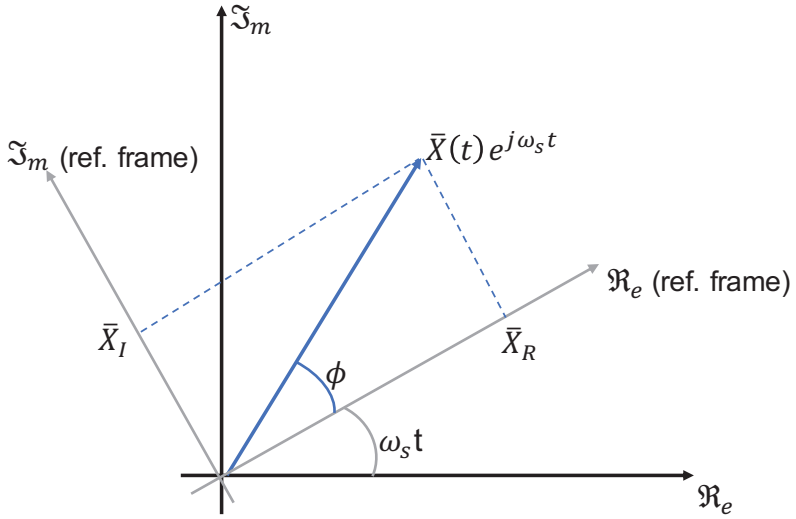

Figure 3.1: Example of phasor diagram. The reference frame is a rotating plan whose angle is $\omega_s t$ with respect to the real axis. The complex signal defined by $\bar{X}(t)e^{j\omega_s t}$ can thus be projected into this plan to define the low-frequency real part $\bar{X}_R$ and imaginary part $\bar{X}_I$.

this approach and approximated $\tilde{X}_s(\tau)$ in the interval $\tau \in]t-T, t]$ with a Fourier series representation given by (3.3) :

$$\tilde{X}_s(\tau) = \sum_{k=-\infty}^{\infty} \bar{X}_k(t) e^{jk\omega_s \tau} \tag{3.3}$$

where the phasors coefficients are time varying and are referred to as dynamic phasors. They are expressed as :

$$\bar{X}_k(t) = \frac{1}{T} \int_{t-T}^{t} \tilde{X}_s(\tau) e^{-jk\omega_s \tau} d\tau \tag{3.4}$$

In transient simulations, the oscillating variables are real signals and can be represented as the real part of dynamic phasors with a centered complex exponential, i.e.

$$X_s(t) = \sqrt{2}\Re\left\{\bar{X}(t)e^{j\omega_s t}\right\} \tag{3.5}$$

Let us illustrate this approach with a signal containing only the fundamental mode. The dynamic phasor of $X_s(t)$ is the complex number $\bar{X}(t)$ which can thus be represented as

$$\bar{X}(t) = A(t)e^{j\phi(t)} \tag{3.6}$$

This exponential representation is very convenient for electrotechnical purposes since the two parameters will play an important role : for instance, in synchronous machines modeling, the modulus A can be used as input for machine controls while the argument ϕ can

be used for measuring its relative angle. However, for time-domain simulations, complex numbers are generally implemented in additive representation:

$$\bar{X}(t) = \bar{X}_R(t) + j\ \bar{X}_I(t) \tag{3.7}$$

Indeed, as the real and the imaginary axes are orthogonal, the equations on the real part $\bar{X}_R$ can be distinguished from those on the imaginary part $\bar{X}_I$. We recall the following equation giving the relation between these two phasor representations:

$$\bar{X}(t) = A(t)e^{j\phi(t)} = \underbrace{A(t)\cos(\phi(t))}_{\bar{X}_R(t)} + j\ \underbrace{A(t)\sin(\phi(t))}_{\bar{X}_I(t)} \tag{3.8}$$

As previously mentioned, if the pulsation of a 3-phase system is ω, the phasor oscillates at the pulsation $|\omega_s - \omega|$. That's why this approach, which focuses only on the envelope and so substitutes the original oscillating signal by an almost constant variable in steady-state, is very efficient. Finally, as ω_s is known, the complete waveform corresponding to $\bar{X}(t)e^{j\omega_s t}$ can be computed from the phasor $\bar{X}(t)$.

In this representation, the time derivative is computed as follows:

$$\frac{d}{dt}(\bar{X}(t)e^{j\omega_s t}) = \frac{d}{dt}(\left[\bar{X}_R(t) + j\ \bar{X}_I(t)\right] e^{j\omega_s t}) \tag{3.9}$$

$$= (\left[\dot{\bar{X}}_R(t) + j\ \dot{\bar{X}}_I(t)\right] e^{j\omega_s t}) + j\omega_s \left[\bar{X}_R(t) + j\ \bar{X}_I(t)\right] e^{j\omega_s t} \tag{3.10}$$

$$= \left[(\dot{\bar{X}}_R(t) - \omega_s \bar{X}_I(t)) + j\ (\dot{\bar{X}}_I(t) + \omega_s \bar{X}_R(t))\right] e^{j\omega_s t} \tag{3.11}$$

To illustrate this approach, let us consider the example of a linear model corresponding to electrical grid equations with an input term, i.e. :

$$\dot{X}(t) = AX(t) + BU(t) \tag{3.12}$$

This equation can be rewritten as below, by using the equation (3.5) and (3.11) :

$$\begin{bmatrix} \dot{\bar{X}}_R(t) \\ \dot{\bar{X}}_I(t) \end{bmatrix} = \begin{bmatrix} A & \omega_s I_d \\ -\omega_s I_d & A \end{bmatrix} \begin{bmatrix} \bar{X}_R(t) \\ \bar{X}_I(t) \end{bmatrix} + \begin{bmatrix} B & 0 \\ 0 & B \end{bmatrix} \begin{bmatrix} \bar{U}_R(t) \\ \bar{U}_I(t) \end{bmatrix} \tag{3.13}$$

Hence, for linear systems, deducing the phasor equations is trivial and can be automatically performed. However, when the system contains non-linear equations and harmonics, the new equations system has to be derived for each considered unit-model. Plenty of publications correspond to the derivation of electrical components models for taking into account most of the studied dynamics. In [59][60][30], these models were mainly written using a symmetrical components approach for dealing with asymmetrical faults. In [14] [16][15], models are expressed either in ABC or in DQ0 representation depending on

the considered electrical component. Hence, in spite of their efficiency, dynamic phasors possibly require fastidious and very particular developments. In addition, as the phasor approach requires several high assumptions on the solution and modify the original equation system in a non-equivalent way, it is not suited to our framework whose main objective is to control the computational error.

3.1.2 Related methods

In [61] [62] Strunz & Al proposed a simulation approach called Frequency Adaptive Simulation of Transient (FAST). It combines the previously presented dynamical phasor modeling with EMTP's model assembling process [17][41]. In addition, in their methodology, a frequency shift is introduced in order to have a degree of freedom on the system frequency. Its numerical properties are investigated in [76]. Then, the idea is to write unit-models, called companion models, corresponding to the discretized unit-components dynamical phasor model. For instance, in [75][38], synchronous machine models are presented using FAST. The global model then combine them using the modified-augmented nodal analysis [41]. This approach has been tested within EMTP tools, whose results are presented in [22][42]. However, in spite of its performances, FAST does not seem suited to our final objective as it is based on the same hypotheses as dynamical phasors. Another limitation is that testing different solvers requires to rewrite the associated new models, as the components models are written in discretized form.

The dynamic phasors approach has also been extended within the so-called Dynamic Harmonic Domain [55]. Indeed, when signal harmonics occur (multiple of the basis signal frequency), for example in transmission lines modelling [9] and its interfacing with synchronous machine [8], the dynamical system state space representation is transformed in the frequency domain. The DHD accuracy depends on the time step size and the harmonics number. Nevertheless, Chavez and Ramirez mention that there is no criterion for determining the number of harmonics given a predefined error [9]. Additionally, the highest harmonic is closely related to the time step used for the simulation. A steady state solution using Harmonic Domain can be used to figure out the approximate number of harmonic coefficients required in a dynamic study. Compared with the corresponding standard time-domain equation, it is clear that phasor dynamic models (or DHD) result in an increase of the equations number that is proportional to the number of retained harmonics.

3.1.3 Time Domain Transformation

The Time Domain Transformation (TDT), proposed in [18] by Fan and Ding, also consists in transforming the original equations system on the oscillating variables waveform into an equations system on their envelope. However, this approach can be seen as a "quasi-solver approach" since the new equations system is automatically derived from the original system. In order to do this, the TDT only requires the system Jacobian and time-derivative. The former is generally provided by the modeler but the latter is less common as it is not used in classical numerical integrators. Let us assume that we have access to these two partial-derivatives. In this section, we first present this approach for ODE containing only oscillating variables, as initially introduced in Fan and Ding's paper, and, in a second time, the extension that we developed for DAE systems containing both oscillating and non-oscillating solutions.

3.1.3.1 Presentation on a scalar ODE

Fan and Ding proposed a novel frequency-adaptive methodology based on the transformation of the time domain original problem in order to take into account a slow time varying waveform superimposed on the fundamental frequency ω_0 waveform.

Hence, they considered the following model for representing real-valued oscillating variable x_s, whose pulsation is ω_0, in the time-domain:

$$x_s(t) = A_s(t)\cos(\omega_0 t + \phi_s(t)) \tag{3.14}$$

Let us consider a scalar ODE

$$\dot{x}_s(t) = f_s(t, x_s) \tag{3.15}$$

As we have one equation to define two unknowns corresponding to the parameters $A_s(t)$ and $\phi_s(t)$, the problem written in this form is not well defined. A second equation is necessary. This second equation is artificially deduced by computing the solution second-order time-derivative from the ODE (3.15). The TDT then consists in 2 steps:

1. Creating the second ODE by computing the ODE function f_s total time derivative. By this way, a coupled system of ODE on the actual solution x_s and its scaled time-derivative $\delta_{x_s} \overset{def}{=} \frac{\dot{x}_s}{\omega_s}$ is obtained. ω_s is generally set to the system nominal frequency, i.e. $\omega_s = \omega_0$.
2. Substituting the variables in order to "project" the resulting system into a rotating plan, as in the phasor modeling.

Hence, the first step consists in computing the total time derivative of f from the classical formula:

$$\ddot{x}_s(t) = \frac{Df_s(t,x_s)}{Dt} = \frac{\partial f_s(t,x_s)}{\partial t} + \frac{\partial f_s(t,x_s)}{\partial x_s}\dot{x}_s \tag{3.16}$$

By combining the equations (3.15) and (3.16), the coupled ODE system on x_s and δ_{x_s} is directly obtained as below:

$$\begin{cases} \dot{x}_s &= f_s(t,x_s) \\ \dot{\delta}_{x_s} &= \dfrac{1}{\omega_s}\dfrac{\partial f_s(t,x_s)}{\partial t} + \dfrac{\partial f_s(t,x_s)}{\partial x_s}\delta_{x_s} \stackrel{def}{=} \Phi_s(t,x_s,\delta_{x_s}) \end{cases} \tag{3.17}$$

In a second time, the idea is to introduce the same frequency reduction approach as in phasors. In other words, the two unknowns x_s and δ_{x_s} are projected into a rotating plan whose rotation speed is associated to the sinusoidal variables frequency. So, let u and v be the projections of x_s and δ_{x_s} into this rotating plan :

$$\begin{bmatrix} u \\ v \end{bmatrix} = \underbrace{\begin{bmatrix} \cos(\omega_s t) & -\sin(\omega_s t) \\ -\sin(\omega_s t) & -\cos(\omega_s t) \end{bmatrix}}_{=R(t)} \begin{bmatrix} x_s \\ \delta_{x_s} \end{bmatrix} \tag{3.18}$$

where, the matrix R is symmetric and orthogonal ($R(t)^T = R(t)$ and $R(t)^{-1} = R(t)$) and its time-derivative is given by $\dot{R}(t) = \omega_s R(t + \frac{\pi}{2\omega_s})$. Finally, the expected ODE on u and v is obtained by computing the time derivative of (3.18):

$$\begin{bmatrix} \dot{u} \\ \dot{v} \end{bmatrix} = \omega_s R(t + \frac{\pi}{2\omega_s}) \begin{bmatrix} x_s \\ \delta_{x_s} \end{bmatrix} + R(t) \begin{bmatrix} f_s(t,x_s) \\ \Phi_s(t,x_s,\delta_{x_s}) \end{bmatrix} \tag{3.19}$$

Then, u and v give the envelope of x_s. Since they are computed in the rotating plan, as in phasors, their variations are much slower than those of x_s and δ_{x_s}. Thus this reference frame change enables to use much larger time steps to solve (3.19) than (3.15). Indeed, by assuming that $A_s(t)$ and $\phi_s(t)$ variations are sufficiently small, we can prove that :

$$\begin{cases} u(t) &\approx A_s(t)\cos(\phi_s(t)) \\ v(t) &\approx A_s(t)\sin(\phi_s(t)) \end{cases} \tag{3.20}$$

To illustrate this point, let us consider u from the definition (3.18):

$$\begin{aligned} u(t) &= x_s(t)\cos(\omega_s t) - \delta_{x_s}(t)\sin(\omega_s t) \\ &= A_s(t)\cos(\omega_0 t + \phi_s(t))\cos(\omega_s t) - \frac{\dot{A}_s(t)}{\omega_s}\cos(\omega_0 t + \phi_s(t))\sin(\omega_s t) \\ &- \frac{A_s(t)}{\omega_s}(\omega_0 + \dot{\phi}_s(t))\sin(\omega_0 t + \phi_s(t))\sin(\omega_s t) \end{aligned}$$

We assume that $\omega_s = \omega_0$:

$$u(t) = A_s(t) \underbrace{(\cos(\omega_0 t + \phi_s(t))\cos(\omega_0 t) + \sin(\omega_0 t + \phi_s(t))\sin(\omega_0 t))}_{=\cos(\phi_s(t))} - \frac{\dot{A}_s(t)}{\omega_0}\cos(\omega_0 t + \phi_s(t))\sin(\omega_0 t) - \frac{\dot{\phi}_s(t)}{\omega_0} A_s(t)\sin(\omega_0 t + \phi_s(t))\sin(\omega_0 t)$$

So, if $|\dot{A}_s(t)| \ll \omega_0$ and $|\dot{\phi}_s(t)| \ll \omega_0$, corresponding to classical TS studies assumptions, we finally get $u(t) \approx A_s(t)\cos(\phi_s(t))$. For instance, if we consider the steady-state differential equation $\dot{x}(t) = -\omega_0 A_0 \sin(\omega_0 t + \phi_0)$, whose solution is $x_s(t) = A_0 \cos(\omega_0 t + \phi_0)$, the computation of x_s with a classical approach is constrained by the pulsation of x_s while the TDT solves $\begin{bmatrix} \dot{u} \\ \dot{v} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$, for which very large time steps can be used.

3.1.3.2 Extension to DAE systems

After having tested the TDT on simple differential equations, we have extended this method to DAE systems in semi-explicit form with both oscillating and non-oscillating components, i.e. :

$$\begin{cases} \dot{x}(t) & = f(t, x(t), y(t)) \quad \text{with} \quad x \in \mathbb{R}^{d_x} \\ 0 & = g(t, x(t), y(t)) \quad \text{with} \quad y \in \mathbb{R}^{d_y} \end{cases} \tag{3.21}$$

We recall that, in this system, $x : \mathbb{R} \to \mathbb{R}^{d_x}$ are the differential state variables of the solution and $y : \mathbb{R} \to \mathbb{R}^{d_y}$ represents the algebraic state variables of the solution. Furthermore, as mentioned in the introduction, some unknowns are oscillating : let us define x_s and x_{ns}, respectively the differential oscillating and non-oscillating variables, and y_s and y_{ns}, the algebraic oscillating and non-oscillating variables.

First of all, the DAE system oscillating and non-oscillating equations are separated:

$$\begin{cases} \dot{x}_s(t) & = f_s(t, x(t), y(t)) \\ 0 & = g_s(t, x(t), y(t)) \\ \dot{x}_{ns} & = f_{ns}(t, x(t), y(t)) \\ 0 & = g_{ns}(t, x(t), y(t)) \end{cases} \tag{3.22}$$

Once this DAE Jacobian and its partial-derivative with respect to the time are known, we derived a systematic methodology for using the Time Domain Transformation for this DAE system from these data. As seen before, we compute the total time derivative of the equations associated to the oscillating variables in order to complete this system and

have the right equations number:

$$\begin{cases} \dot{x}_s &= f_s(t,x,y) \\ \ddot{x}_s &= \partial_t f_s(t,x,y) + \partial_x f_s(t,x,y)\dot{x} + \partial_y f_s(t,x,y)\dot{y} \\ 0 &= g_s(t,x,y) \\ 0 &= \partial_t g_s(t,x,y) + \partial_x g_s(t,x,y)\dot{x} + \partial_y g_s(t,x,y)\dot{y} \\ \dot{x}_{ns} &= f_{ns}(t,x,y) \\ 0 &= g_{ns}(t,x,y) \end{cases} \tag{3.23}$$

In the above equations, $\dot{x}_s$ and $\dot{y}_s$ are given by the TDT method, $\dot{x}_{ns}$ is directly obtained from the associated ODE and $\dot{y}_{ns}$ is computed from the g_{ns} total time derivative such as

$$\dot{y}_{ns} = -\frac{\partial g_{ns}}{\partial y_{ns}}^{-1}\left(\frac{\partial g_{ns}}{\partial t} + \frac{\partial g_{ns}}{\partial x_s}\dot{x}_s + \frac{\partial g_{ns}}{\partial x_{ns}}\dot{x}_{ns} + \frac{\partial g_{ns}}{\partial y_s}\dot{y}_s\right) \tag{3.24}$$

which means that deducing the algebraic non-oscillating solutions requires to perform a linear system resolution, whose computational cost may be important. Let us introduce $\delta_x = \frac{\dot{x}}{\omega_s}$ and $\delta_y = \frac{\dot{y}}{\omega_s}$ to get the coupled DAE system:

$$\begin{cases} \dot{x}_s &= f_s(t,x,y) \\ \dot{\delta}_{x_s} &= \dfrac{1}{\omega_s}\partial_t f_s(t,x,y) + \partial_x f_s(t,x,y)\delta_x + \partial_y f_s(t,x,y)\delta_y \overset{def}{=} \Phi_s(t,x,\delta_x,y,\delta_y) \\ 0 &= g_s(t,x,y) \\ 0 &= \dfrac{1}{\omega_s}\partial_t g_s(t,x,y) + \partial_x g_s(t,x,y)\delta_x + \partial_y g_s(t,x,y)\delta_y \overset{def}{=} \Gamma_s(t,x,\delta_x,y,\delta_y) \\ \dot{x}_{ns} &= f_{ns}(t,x,y) \\ 0 &= g_{ns}(t,x,y) \end{cases} \tag{3.25}$$

Finally, by introducing u and v as defined in (3.18), the final lower-frequency DAE system to solve is obtained:

$$\begin{cases} \begin{bmatrix} \dot{u}_{x,1} \\ \dot{v}_{x,1} \\ \vdots \\ \dot{u}_{x,d_{x_s}} \\ \dot{v}_{x,d_{x_s}} \end{bmatrix} &= \omega_s(I_{d_{x_s}} \otimes R(t + \frac{\pi}{2\omega_s})) \begin{bmatrix} x_{s,1} \\ \delta_{x_s,1} \\ \vdots \\ x_{s,d_{x_s}} \\ \delta_{x_s,d_{x_s}} \end{bmatrix} + (I_{d_{x_s}} \otimes R(t)) \begin{bmatrix} f_{s,1}(t,x,y) \\ \Phi_{s,1}(t,x,\delta_x,y,\delta_y) \\ \vdots \\ f_{s,d_{x_s}}(t,x,y) \\ \Phi_{s,d_{x_s}}(t,x,\delta_x,y,\delta_y) \end{bmatrix} \\ 0 &= g_s(t,x,y) \\ 0 &= \Gamma_s(t,x,\delta_x,y,\delta_y) \\ \dot{x}_{ns} &= f_{ns}(t,x,y) \\ 0 &= g_{ns}(t,x,y) \end{cases} \tag{3.26}$$

The numerical results, shown in the Appendix A, prove that this method is efficient in terms of iterations number but not in terms of CPU time. As the CPU time per iteration is much greater than those of classical methods, the final gains of performance are low. In addition, such an approach is based on assumptions on the solution. Indeed, even if a systematic approach has been derived for applying this method on DAE systems containing both oscillating and non-oscillating components, it assumes that the oscillating components are centered sinusoids without harmonics. Therefore, even if the three-phase voltages and currents generally respect this hypothesis, strong transient phenomena such as asymmetrical faults may lead to non-centered signals for instance. There is no proper way to detect numerical errors due to this assumption if it is not actually verified. In other words, such as the phasor modeling, the Time Domain Transformation also modifies the original equations system in a non-equivalent way in order to simplify its resolution a posteriori. In conclusion, this approach does not seem suited to our problem as it presents the same limitation as the previously presented frequency-domain modeling while its potential gains of performances are limited due to its computational cost by iteration.

3.2 Hybrid simulations

The previously presented methods have the common flaw of changing the entire system to solve. In particular, as this substitution is done in a non-equivalent way, controlling the simulation error associated to the original EMT system is not possible. Then, as our objective is to finely control the simulation error for design studies wherein the solution may be radically different from its expected a priori steady-state shape, such approaches seem inappropriate.

In order to accelerate EMT simulations, another proposed strategy, initially proposed by Heffernan & Al in [32][66][67], consists in performing hybrid simulations coupling an EMT simulator with a TS simulator. As EMT models lead to very costly simulations, such an approach aims at reducing the area represented with EMT models. In particular, it tries to get a compromise between accuracy and performances by using the advantages of the two simulations kinds:

- As TS models are very suited to investigate large-scale phenomena, the idea is to simulate most of the system with a TS simulator.
- As EMT models are much more accurate, an EMT simulator is used for simulating the area of interest.

Then, hybrid simulations combine a TS program, which is classicaly based on fundamental frequency, positive sequence, phasor-type data (or dynamic-phasor data as implemented

more recently in [45]), and an EMT program, which is based on the 3-phase instantaneous waveform data and especially includes several frequency components [39].

3.2.1 Common principles (space/time-slicing)

The idea of hybrid simulations is to exploit the limited EMT propagation in the space and time domain. Simulating the whole system with EMT models, which are much more accurate than TS models and thus require much higher computing resource, can seem inappropriate and inefficient. This is why hybrid simulations consist in combining these two types of modeling and especially aims at reducing as much as possible the EMT domain size. The arising technical and scientific challenges have been introduced in [54][2].

Two types of slicing can be distinguished :

1. The geographical slicing, which consists in dividing the whole power system into two areas: the domain of interest is described by the accurate EMT model while the rest of the system is described by a TS model.
2. The time-domain slicing, which consists in alternating between a pseudo-phasor model, based on the Time Domain Transformation, and an accurate EMT model for simulating transients occurring after a discrete event.

Then, the common problem to tackle for these two types of domain decomposition is the definition of the interface between the EMT and the TS modeling. Indeed, the goal is to find an optimal cutting which respects the system dynamics, i.e. the least proportion of the geographical or time domain beyond which the electromagnetic transient is negligible.

The interaction between the EMT and TS simulation is maintained via data interface buses. Parameters that are generally available for measurement include active and reactive power, voltage, current through interface bus, and also phase angle information in the case of different reference frames. The information transferred from one program to the other must determine the power flow in or out of the interface bus. Another major concern is how to properly pass the interface variables between the TS and the EMT programs. Thus, to connect these two programs types, two data converter functionalities are needed: phasor-to-waveform and waveform-to-phasor[39]. The phasor-to-waveform converting block is a signal generator controlled by amplitude, phase, and frequency [40]. The waveform-to-phasor converter uses digital signal processing techniques as curve fitting [3], Fast Fourier Transform (FFT) [40] [71], or digital filtering [73] while the FFT method requires samples of exactly one cycle to correctly produce the phasor quantities. Thus, after the detailed system has recovered from a fault or a large disturbance it takes one complete cycle for

the FFT method to compute the right value of phasors. This delay in data preparation causes problems for the TS program [39]

3.2.2 Geographical slicing

Most of the time, the geographical domain decomposition is defined before the simulation, as illustrated in 3.2, and is not dynamically changed during the execution time. This cutting is empirically decided from expert assessment inferring on the EMT transients propagation into the connected components. For instance, if lightning strikes a transmission line, the transient-overvoltage is propagated until a certain distance, beyond which the interface can be set. In [77], the authors also consider events in the TS part. However, the interface choice is really difficult and there is currently no proper mathematical way to infer ex ante on its location. Similar problems are encountered in multi-rate time-stepping techniques [31] which aim at separating the global system into several subsystems (associated to different dynamics) in order to perform their numerical integration using different time step sizes.

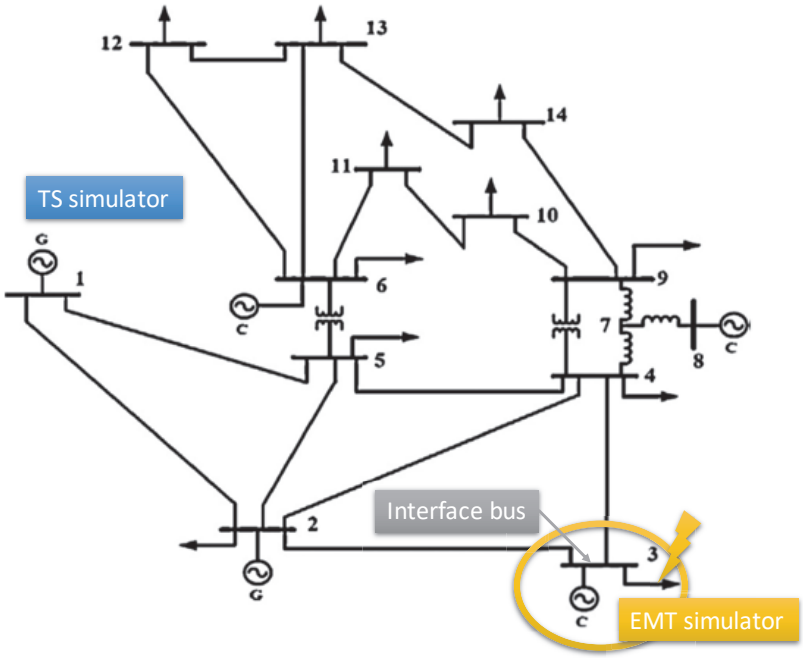

Figure 3.2: Illustration of the concept of hybrid simulation with geographical slicing.

The interface boundary conditions rigorous management is also a open question [3]. The difficulty is that these two systems have to dynamically exchange information to ensure the resolution stability while being solved in parallel. Since EMT and TS programs

have different time steps (microsecond versus millisecond), a parallel interaction protocol is especially required to coordinate the information exchange and update the external system representation. For convenience, the TS simulator step size is an integer multiple of the EMT simulator one, and the information exchange occurs at common points in time, which conventionally are the TS simulator time steps. The data dependencies are then solved by representing the connected system by its equivalent. As it is only valid for a system in steady-state, it may be seen as an explicit coupling scheme which is generally unstable. In order to stabilize the interface problem, more implicit schemes have been proposed, for instance using a relaxation method [52][51] for iteratively refining the external systems equivalents. Another enhancement consists in complexifying the external system equivalent representation, as in [37]. To finish, since a purely explicit parallel interaction protocol (as illustrated in 3.3) generally leads to an unstable parallel resolution, more elaborate protocols based on data redundancy have been proposed in order to stabilize the resolution [64][68].

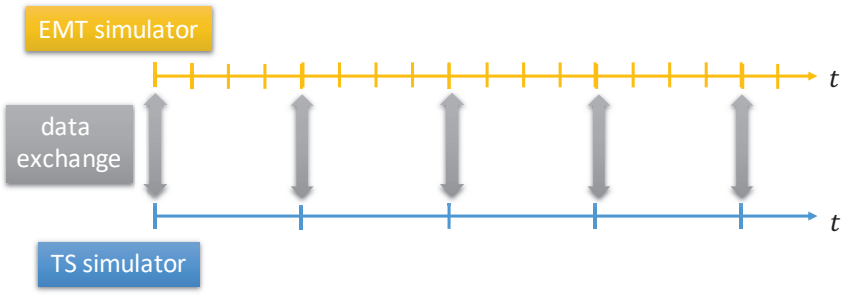

Figure 3.3: Example of explicit parallel protocol for the geographical slicing.

3.2.3 Time-domain slicing

In the time-domain slicing, presented in [26] and illustrated in 3.4, the interfacing consists in globally alternating between EMT and TS simulators. It consists in two steps:

1. When a discrete event occurs, the simulator switches from the pseudo-phasor model based on the Time Domain Transformation (as presented in section 3.1.3) to the full time domain EMT model. This transition is simply done by setting the last state of the TDT system as initial condition for the EMT solver.

2. When the solver detects that the system is in quasi-steady-state, the simulator switches back to the TDT system. As for the first transition, this one is done by setting the initial condition on x from the last state of the EMT system. Then, δ_x is simply obtained from its definition: $\delta_x(t) = \frac{\dot{x}(t)}{\omega_s} = \frac{f(t,x(t))}{\omega_s}$.

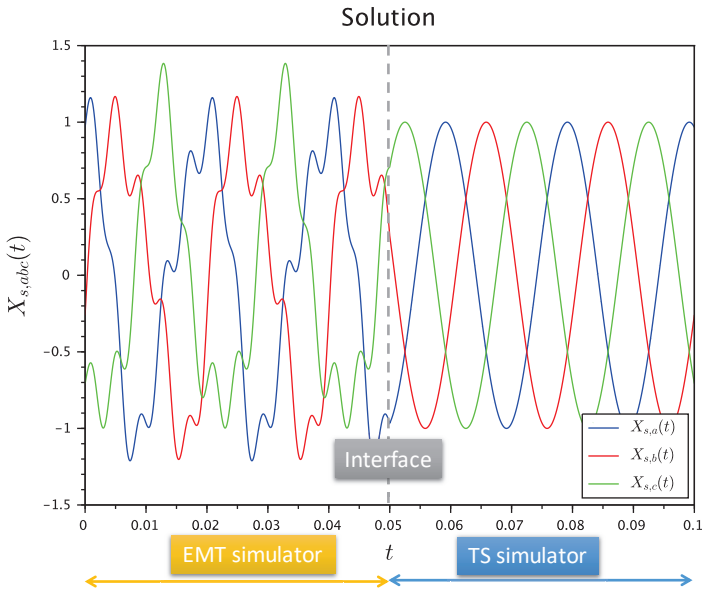

Figure 3.4: Illustration of the concept of hybrid simulation with time-domain slicing (on the abscissa: time in seconds, on the ordinate: three-phase system in arbitrary units).

As for the geographical slicing, the main difficulty is to detect if the system is in quasi-steady-state or if the EMT model is still necessary. It is also done from expert heuristics instead of more general and robust mathematical criteria. A possible enhancements might be to consider overlapping transitions between the TS and EMT models by using Schwarz based approaches, as it is done in [53][7] for time-domain decomposition.

3.3 Numerical approaches for oscillatory solutions

Contrary to the previously presented approaches, several numerical methods aim at accelerating the simulation of oscillatory solutions while solving the original equations system.

However, in most of the existing numerical methods for oscillatory solutions, these oscillations are structural, i.e. they are due to dynamical systems properties and especially their eigenvalues. A review of these methods can be found in [50]. Among these schemes, those based on the geometric numerical integration are especially suited to Hamiltonian systems [27][57] and enable to preserve dynamical invariant. For instance, they are used in astronomical applications such as stationary orbits simulation as they enable to preserve orbits and especially avoid artificial deviations. In this domain, other approaches have been developed in order to accelerate the computations by exploiting the trajectory periodicity, such as in the multirevolution algorithm [70] which consists in enhancing the

prediction step by taking into account the last orbits data for the interpolation polynomial construction. Encke's method [21], focusing the computations on the deviation to a stationary trajectory, has been also implemented for accelerating the computations in [5], which confirms the potential of such an approach. However, as for the numerical integration schemes, all these approaches are mainly designed for structural oscillations and their step sizes would always to be oscillation period multiples [50] .

On the contrary, in [48], Petzold proposed an envelope tracking method for the circuit analysis which is also suited to oscillations due to an oscillating forcing term. This method consists in taking into account the signal expected T-periodicity in order to rewrite the numerical scheme so that it focuses on the signal envelope. The idea is to exploit the fact that if the envelope is equal to the signal value at a given time t, it is also for $t + kT$ where k is a positive integer. As for harmonic domain methods, larger step sizes can used for integrating the associated equations as the high frequency signal is then substituted by a low frequency variable. In order to take into account possible system frequency variations, Petzold proposed a least-squares estimator for the period T, that has been optimized in [44]. However, in spite of the envelope-tracking approach notable efficiency, it does not seem really suited to our applications since it focuses on the envelope. In particular, an additional integration process is required to compute the true solution from the computed quasi-envelope value, which is used as initial condition. As a result, the step size adjustment used for computing the quasi-envelope seems to possibly neglect inner-cycle dynamics. On the contrary, even if our objective is to use larger step sizes in steady-state, our method aims at directly computing the original problem solution and especially not neglecting inner-cycle dynamics thanks to a very flexible step size adjustment strategy.

3.4 Conclusion

In conclusion, modeling approaches such as dynamic phasors, FAST, the dynamic harmonic domain and the time domain transformation have the main drawback to entirely change the dynamical system modeling in non-equivalent way which prevents from properly controlling the simulation error. In particular, the harmonics choice is not a parameter free of these methods. The time domain transformation has also the drawback to have an important computational cost due to the resolution of a linear system for evaluating time-derivative of the non-oscillating algebraic variables.

Hybrid simulations are also attractive notably on large networks but they are also not

parameter free, as the exchange of data needs conversions and the conversion from waveform to phasors needs sophisticated interpolation introducing some delay. In addition, properly managing the interface is very difficult, especially for inferring on its location and choosing an appropriate interaction protocol. It presents the same problem as for frequency-domain modeling since the original model is also affected by the solving process.

To finish, some numerical methods directly solving the original DAE system already exist but they do not seem suited to our applications. In particular, most of them are based on the dynamical system eigenvalues for taking advantage of structural oscillations. Other approaches such as the multirevolution algorithm used in orbits prediction and the envelope-tracking method use the solution periodicity to enhance the predictor-corrector scheme by focusing on the solution envelope or expected steady-state behavior. However, the underlying step size adjustment strategy do not seem flexible enough for our applications as the step size is strongly linked to the signal period. Moreover, an internal integration process is required for deducing the solution instantaneous value from the envelope, i.e. the inner-cycle dynamics do not seem properly considered.

Part II

Proposed solver

4

SPM presentation

Contents

As exposed in the chapter 3, the existing methods common flaw is that they require to change the modeling by rewriting the original differential algebraic equations describing the EMT system in order to simplify its resolution. On the contrary, the method that we developed and implemented focuses on solving the original system and is only implemented at the solver level. The Sinusoidal Predictor Method (SPM) consists in fitting a sinusoid corresponding to (2.4) for the oscillating components of the system and to correct it with an adaptive step size integration scheme.

4.1 Concept

We recall that the system to solve is a set of differential algebraic equations in semi-explicit form (2.2):

$$\begin{cases} \dot{x}(t) & = f(t, x(t), y(t)) \\ 0 & = g(t, x(t), y(t)) \\ (x(t_0), y(t_0)) & = (x_0, y_0) \end{cases}$$

whose solution X contains oscillating (referred as X_s, let $\mathcal{I}_s$ be the associated set of indexes) and non-oscillating (referred as X_{ns}) components. (x_0, y_0) are consistent initial conditions so that $g(t_0, x_0, y_0) = 0$.

The SPM idea is to decompose the solution as the sum of a periodic part and a correction

term. This additive decomposition is done for each time integration interval $[t_n, t_{n+1}]$:

$$X(t) = \underbrace{\bar{X}_n(t)}_{\text{periodic part}} + \underbrace{\delta(t)}_{\text{correction term}} \tag{4.1}$$

In this equation, which is illustrated in figure 4.1:

- $\bar{X}_n$ is the periodic part of the solution. It corresponds to the steady-state solution for the oscillating components, i.e. it is defined by

  $$\bar{X}_n(t) = u_n \sin(\omega t) + v_n \cos(\omega t) \tag{4.2}$$

  Then, it is a sinusoid with constant Fourier coefficients. For each time interval $[t_n, t_{n+1}]$, the set of Fourier coefficients is fixed and so $\bar{X}_n$ is a parametric function depending only on the time. The Fourier coefficients are updated by parametric estimation. Therefore, the SPM periodic function is locally defined since the Fourier coefficients are piecewise constant and so their values change for each considered time interval $[t_n, t_{n+1}]$.
- δ is the correction term on which the problem is rewritten from the original DAE system. This correction term is very flexible as it can catch both errors on the oscillating components envelope and stronger EMT transients as offsets or harmonics, etc. It is computed by integrating the differential algebraic equations corresponding to the deviation between the global solution and the periodic part. As the above mentioned sets of Fourier coefficients are fixed for each time interval $[t_n, t_{n+1}]$, δ and the associated problem are locally defined. However, the global solution and especially the associated problem do not depend on the local Fourier coefficients. So, in the numerical integration, the process consists in injecting the local Fourier coefficients, solving the local problem with its initial condition associated to the local correction term, deducing the global solution and finally estimating the next local Fourier coefficients to use. There is an alternation between the local point-of-view associated to the Fourier coefficients and the correction term, and a global point-of-view corresponding to the global solution.

When considering a system containing both oscillating and non-oscillating components (as mentioned above), the non-oscillating components of the solution are not decomposed as in (4.1). Thus, their periodic part is set to zero i.e.

$$(u_n, v_n)_i = (0, 0) \text{ if } i \in \mathcal{I}_{ns} \tag{4.3}$$

Which means that, for the non-oscillating components of the solution, the correction term is directly equal to the global solution:

$$\delta_i(t) = X_i(t) \text{ if } i \in \mathcal{I}_{ns} \tag{4.4}$$

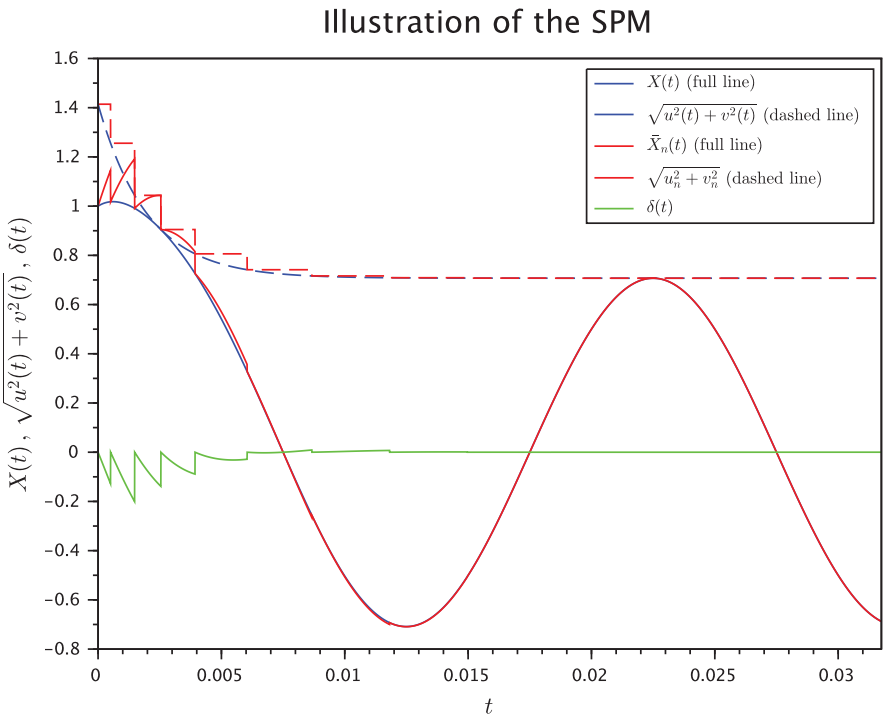

Figure 4.1: Illustration of the SPM decomposition (on the abscissa: time in second, on the ordinate: solution, Fourier coefficients and SPM components in arbitrary units). In this example, the theoretical solution (full blue line) is a centered sinusoid whose amplitude (dashed blue line) decreases. The SPM periodic function (full red line) is given by its sequence of piecewise constant Fourier coefficients (dashed red line). The correction term (full green line), which is equal to the difference between the solution and the periodic function, is then a sinusoid which catches the envelope variations. Thus, it decreases as the solution tends toward steady-state. At each interface between two time intervals, we can see a jump of the correction term. Indeed, the SPM Fourier coefficients and the resulting periodic function are locally defined, which also leads to a local definition of the correction term.

Of course, by using the same approach for all the components (i.e. $(u_n, v_n)_i = (0, 0)\ \forall i = 1, ..., d$), the SPM can be switched off for simulating the entire system with a classical integration scheme, e.g. during strong EMT transients. Indeed, in such situations, as the solution could behave in a particularly erratic way, the SPM could fail to fit a fundamental mode sinusoid. Hence, using a classical method for solving the DAE system during this period could be much more efficient since additional mathematical operations are required for estimating the SPM Fourier coefficients. Then, once the system would come

back to standard operations i.e. once the EMT phenomenon would have been cleared, the SPM could be switched on once again. During this PhD thesis, this point has not been addressed but it could be in future developments. As for the hybrid simulations with time slicing, the most difficult challenge when building such approaches is to have a sufficiently robust criterion for inferring on the system state: finding the appropriate time for reactivating the SPM requires to properly detect the standard operations.

Hence, the SPM objective is to catch as much as possible the sinusoidal behavior of the solution X into the parametric function $\bar{X}_n$, so that the correction term is damped in steady-state for the oscillating components. By this way, the correction term tends toward constant values in steady-state as the correction associated with oscillating components should be close to zero while those of non-oscillating components should be close to their asymptotic values. As a result, large step sizes could be used as the simulated variable has no dynamics.

To summarize, for each time interval $[t_n, t_{n+1}]$, the method consists in (see figure 4.2):

1. Fixing the parameters (u_n, v_n) of the oscillating part $\bar{X}$;
2. Integrating the equations on the correction δ with an adaptive step size scheme;
3. Deducting the global solution X from (4.1);
4. Updating the oscillating part $\bar{X}$ by estimating (u_{n+1}, v_{n+1}) .

4.2 Problem reformulation

At time t_n, we assume that we have an approximation of the solution $X_n = (x_n, y_n)$ and the Fourier coefficients (u_n, v_n). Our objective is to approximate the solution and the Fourier coefficients at the next time step $t_{n+1} = t_n + h_n$, i.e. $X_{n+1} = (x_{n+1}, y_{n+1})$ and (u_{n+1}, v_{n+1}). In addition, we assume that we have the modified Nordsieck vector [46] associated to the history of the solution $\vec{X}_n$. In this presentation of the method, this vector contains three components as we use the Adams-BDF2 scheme, presented in [4], which is a second-order method. It is defined by

$$\vec{X}_n = \begin{bmatrix} X_n \\ \dot{X}_n \\ \ddot{X}_n \end{bmatrix} \tag{4.5}$$

The first step of the SPM is to deduce the local problem associated to the correction term by injecting the local sinusoidal function into the original problem associated to the global solution. This corresponds to the first block in the figure 4.2. Hence, first

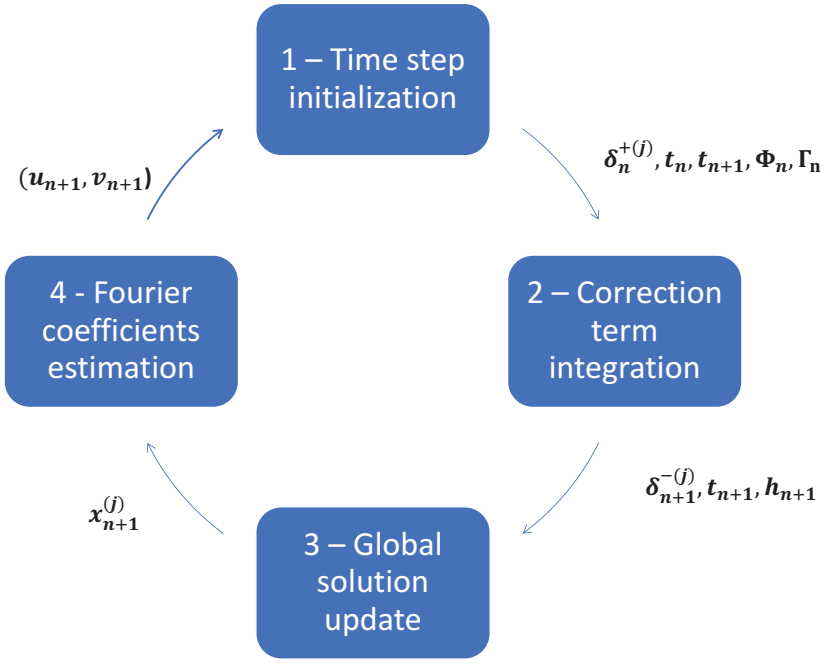

Figure 4.2: Overview of the SPM. For each time step, the SPM consists in 4 steps. Firstly, the periodic part parameters are fixed in order to rewrite the local DAE system on the correction term. This implies to compute consistent initial conditions on the correction term from the global solution and the local value of the periodic function. Then, the system is integrated with the chosen adaptive stepsize predictor-corrector scheme in order to get the correction term value at next time step. After this integration part, the global solution at next time step is deduced from the correction term value and the periodic part evaluation. To finish, the Fourier coefficients are updated by parametric estimation. Hence, as the parametric estimation is performed from data computed within the integration step, the entire method stability is to consider with the possible feedback effects. Indeed, numerical integration errors may be injected within the estimator, which may finally reintroduce estimation errors into the integrator, and so on. This point is discussed in more details in the next chapter.

of all, the Fourier coefficients (u_n, v_n) are set to get the parametric function $\bar{X}_n(t)$. By this way, for the considered time interval $[t_n, t_{n+1}]$, the periodic function $\bar{X}_n$ depends only on the time, which enables to define the local problem associated to the correction term δ.

Then, the equations (2.2) can be rewritten on the correction term δ from (4.1):

$$\dot{\delta}_x(t) = f(t, \delta_x(t) + \bar{x}_n(t), \delta_y(t) + \bar{y}_n(t)) - \dot{\bar{x}}_n(t) \tag{4.6}$$

$$0 = g(t, \delta_x(t) + \bar{x}_n(t), \delta_y(t) + \bar{y}_n(t)) \tag{4.7}$$

Let us define $\Phi_n : \mathbf{R} \times \mathbf{R}^{d_x} \times \mathbf{R}^{d_y} \to \mathbf{R}^{d_x}$, the associated right-hand-side function for the differential equation, and $\Gamma_n : \mathbf{R} \times \mathbf{R}^{d_x} \times \mathbf{R}^{d_y} \to \mathbf{R}^{d_y}$, the associated algebraic constraints, such as:

$$\dot{\delta_x}(t) = \Phi_n(t, \delta_x(t), \delta_y(t)) \tag{4.8}$$

$$0 = \Gamma_n(t, \delta_x(t), \delta_y(t)) \tag{4.9}$$

Then, a first advantage of the used additive decomposition is that the Jacobian matrix associated to the local problem on the correction term is directly equal to those of the global problem on the solution, i.e.

$$J_{\Phi_n}(t, \delta_x(t), \delta_y(t)) = J_f(t, \delta_x(t) + \bar{x}_n(t), \delta_y(t) + \bar{y}_n(t)) \tag{4.10}$$

$$J_{\Gamma_n}(t, \delta_x(t), \delta_y(t)) = J_g(t, \delta_x(t) + \bar{x}_n(t), \delta_y(t) + \bar{y}_n(t)) \tag{4.11}$$

with

$$J_f(t, x(t), y(t)) = \left[\frac{\partial f(t, x(t), y(t))}{\partial x} \quad \frac{\partial f(t, x(t), y(t))}{\partial y} \right] \tag{4.12}$$

$$J_g(t, x(t), y(t)) = \left[\frac{\partial g(t, x(t), y(t))}{\partial x} \quad \frac{\partial g(t, x(t), y(t))}{\partial y} \right] \tag{4.13}$$

Indeed

$$\frac{\partial f(t, \delta(t) + \bar{x}_n(t))}{\partial \delta} = \frac{\partial f(t, \delta(t) + \bar{x}_n(t))}{\partial x} \underbrace{\frac{\partial x}{\partial \delta}}_{=1} \tag{4.14}$$

Then, as these equations depend on (u_n, v_n) for each interval $[t_n, t_{n+1}]$, the following relations have to be verified to ensure the regularity of the solution X at times t_n and t_{n+1}:

$$\delta_n^{+(j)} = X_n^{(j)} - \bar{X}_n^{(j)}(t_n) \tag{4.15}$$

$$\delta_{n+1}^{-(j)} = X_{n+1}^{(j)} - \bar{X}_n^{(j)}(t_{n+1}) \tag{4.16}$$

where the exponent (j) refers to the j-th time derivative with $j = 0, 1, 2$, $X_n^{(j)}$ approximates $X^{(j)}(t_n)$, $\delta_n^{+(j)}$ approximates the t_n-right-value of the correction term $\delta^{+(j)}(t_n)$, and $\delta_{n+1}^{-(j)}$ approximates the t_{n+1}-left-value of the correction term $\delta^{-(j)}(t_{n+1})$. Indeed, $X_n^{+(j)} = X_n^{-(j)} (= X_n^{(j)}) \Leftrightarrow \delta_n^{+(j)} + \bar{X}_n^{(j)}(t_n) = X_n^{(j)}$. This particular treatment of the correction term discontinuity can be seen in figure 4.1 as the jumps of the Fourier coefficients are clearly visible.

4.3 Reformulated equations numerical integration

In the first step, the local DAE corresponding to the problem on the correction has been deduced from the original equations on the global solution and the local value of the

Fourier coefficients. In addition, local consistent initial values have been derived for the correction in order to prepare its numerical integration in a second time. This second step corresponds to the second block in the figure 4.2.

First of all, the DAE system (4.8)-(4.9) is completed by initial conditions on $\delta_n^{+(j)}$ at time t_n by applying (4.15) (step 1 in figure 4.2). Then, a predictor-corrector with time step h_n is applied in the interval $[t_n, t_{n+1}]$ to compute $\delta_{n+1}^{-(j)}$, $j = 0, ..., k$ (step 2 in figure 4.2). In the following description, the used method is the TR-BDF second-order scheme as introduced in [4].

For recall, a predictor-corrector scheme with adaptive step size generally consists in the three following steps [28] :

1. Prediction: computing a coarse approximation of the solution at the next time step by extrapolating a polynomial built from the solution history. It corresponds to the chosen explicit integration scheme. We will denote the associated variables by an hat: $\hat{X}_{n+1}$ and $\hat{\delta}_{n+1}^{-}$;
2. Correction: computing a more precise approximation of the solution at the next time step by solving the fixed-point problem associated to the chosen implicit integration scheme. We will note these variables X_{n+1} and δ_{n+1}^{-};
3. Step size adjustment from the fixed tolerance and the difference between the prediction and the correction.

In Nordsieck's formalism [46] , the solution history (represented by $\vec{X}$ for X, $\vec{\delta}^{+}$ for δ^{+} and $\vec{\delta}^{-}$ for δ^{-}) is stored as an approximation of its Taylor development which is actually equivalent to the chosen multistep method [28]:

$$\vec{X}_n = \begin{bmatrix} X_n \\ \dot{X}_n \\ \ddot{X}_n \end{bmatrix} \approx \begin{bmatrix} X(t_n) \\ \dot{X}(t_n) \\ \ddot{X}(t_n) \end{bmatrix} \tag{4.17}$$

Let h_n be the step size such as $t_{n+1} = t_n + h_n$. Then, the prediction is simply obtained by extrapolating the Taylor polynomial that is based on the Nordsieck vector of δ_n^{+}, i.e.

$$\hat{\delta}_n(t) = \delta_n^{+} + \dot{\delta}_n^{+}(t - t_n) + \ddot{\delta}_n^{+}\frac{(t - t_n)^2}{2} \tag{4.18}$$

at next time step t_{n+1}:

$$\hat{\delta}_{n+1}^{-} = \delta_n^{+} + \dot{\delta}_n^{+} h_n + \ddot{\delta}_n^{+}\frac{h_n^2}{2} \tag{4.19}$$

$$\dot{\hat{\delta}}_{n+1}^{-} = \dot{\delta}_n^{+} + \ddot{\delta}_n^{+} h_n \tag{4.20}$$

$$\ddot{\hat{\delta}}_{n+1}^{-} = \ddot{\delta}_n^{+} \tag{4.21}$$

Once this prediction is computed, the correction can be obtained by solving the following fixed-point problem on δ^-_{n+1} (with the Newton-Raphson algorithm [74] for instance) which corresponds to the TR-BDF scheme:

$$\delta^-_{x,n+1} = \delta^+_{x,n} + \frac{h_n}{2}(\Phi_n(t_n, \delta^+_{x,n}, \delta^+_{y,n}) + \Phi_n(t_{n+1}, \delta^-_{x,n+1}, \delta^-_{y,n+1})) \tag{4.22}$$

$$0 = \Gamma_n(t_{n+1}, \delta^-_{x,n+1}, \delta^-_{y,n+1}) \tag{4.23}$$

Whose Jacobian matrix is given by

$$J_n(\delta^-_{n+1}) = \begin{bmatrix} I_{d_x} - \frac{h_n}{2}\frac{\partial \Phi_n(t_{n+1}, \delta^-_{x,n+1}, \delta^-_{y,n+1})}{\partial \delta_x} & -\frac{h_n}{2}\frac{\partial \Phi_n(t_{n+1}, \delta^-_{x,n+1}, \delta^-_{y,n+1})}{\partial \delta_y} \\ \\ \underline{\frac{\partial \Gamma_n(t_{n+1}, \delta^-_{x,n+1}, \delta^-_{y,n+1})}{\partial \delta_x}} & \underline{\frac{\partial \Gamma_n(t_{n+1}, \delta^-_{x,n+1}, \delta^-_{y,n+1})}{\partial \delta_y}} \end{bmatrix} \tag{4.24}$$

which can be directly derived from the original system Jacobian, as seen in the equations (4.10)-(4.11).

Once the fixed point algorithm has converged, the Nordsieck vector of δ^-_{n+1} can be updated from the following formula for each component $i = 1, ..., d$:

$$\vec{\delta}^-_{i,n+1} = \vec{\hat{\delta}}^-_{i,n+1} + \vec{l}^{Nordsieck}_{i,n} \Delta^-_{i,n+1} \tag{4.25}$$

where

$$\Delta^-_{i,n+1} = \delta^-_{i,n+1} - \hat{\delta}^-_{i,n+1} \tag{4.26}$$

is the error of prediction and

$$\vec{l}^{Nordsieck}_{i,n} = \begin{cases} [1, \frac{2}{h_n}, \frac{2}{h_n^2}]^T & \text{if } i \in \mathcal{I}_x \\ [1, \frac{3}{2h_n}, \frac{1}{h_n^2}]^T & \text{if } i \in \mathcal{I}_y \end{cases} \tag{4.27}$$

Finally, the solution and its Nordsieck vector are directly obtained (step 3 in fig.4.2) from the SPM decomposition (4.1) (verifying (4.16) by construction):

$$X^{(j)}_{n+1} = \bar{X}^{(j)}_n(t_{n+1}) + \delta^{-(j)}_{n+1} \tag{4.28}$$

To finish, the step size is updated, for instance from the following formula which corresponds to the already mentioned simplest strategy [47]

$$h_{new} = \underbrace{k_s \sqrt[3]{\frac{tol}{||e_{n+1}||}}}_{\overset{def}{=} \alpha_n} h_n \tag{4.29}$$

where

- tol is the solver tolerance on the local truncation error (e.g. 10^{-6})
- $k_s \in]0,1[$ is a security coefficient (empirically fixed, for instance $k_s = 0.65$)
- e_{n+1} is an approximation of the local truncation error. It can be computed from the prediction error $\Delta^-_{n+1} = \delta^-_{n+1} - \hat{\delta}^-_{n+1}$ with the formula

$$e_{n+1} = c_n ||\Delta^-_{n+1}|| \tag{4.30}$$

  where c_n is a coefficient associated to the integration scheme.

The cubic root in this step size strategy is due to the use of a second order scheme since the local truncation error of a q-order method is given by $e_{n+1} = \mathcal{O}(h_n^{q+1})$. Then, if $||e_{n+1}|| < tol$, the step size is accepted and increased for the next time step: $h_{n+1} = h_{new} \geq h_n$. In the contrary, if $||e_{n+1}|| > tol$, the step size is rejected and decreased: $h_n \leftarrow h_{new} < h_n$. This prediction - correction - step size adjustment process is repeated until the step size h_n is validated.

4.4 Parameters update

In the previous step, the numerical integration has been performed on the correction term to compute δ^-_{n+1} and the step size has been adjusted in order to reach a tolerance target tol from the estimated error on the correction term e_{n+1}. This corresponds to the second step in figure 4.2. Once the step size is validated, this integration step is achieved. Then, from the integrated correction term and the evaluation of the local periodic function $\bar{X}_{n+1}$ at next time step t_{n+1}, the global solution X_{n+1} has been deduced during the third step in figure 4.2. Therefore, the SPM is finally completed by the parametric estimation for updating the Fourier coefficients (u_{n+1}, v_{n+1}) within the fourth step in figure 4.2. This last step enables to update the periodic function $\bar{X}_{n+1}(t)$ for the next time interval $[t_{n+1}, t_{n+2}]$.

In the initial version of the method, the Fourier coefficients were estimated (step 4 in figure 4.2) from the classical Fourier estimator:

$$u^*_{n+1,i} = \frac{2}{T} \int_{t_{n+1}-T}^{t_{n+1}} X_i(t) \sin(\omega t) dt \tag{4.31}$$

$$v^*_{n+1,i} = \frac{2}{T} \int_{t_{n+1}-T}^{t_{n+1}} X_i(t) \cos(\omega t) dt \tag{4.32}$$

with $i \in \mathcal{I}_s$. However, this formula led to numerical instabilities which were due to the coupling between the integration and estimation steps. Indeed, as the estimation is done from previously computed data, their integration errors are injected into the estimator, whose resulting error is then re-injected into the integrator and so on. As a result, the

process was unstable and a relaxation parameter $\theta_{uv} \in]0,1[$ was introduced to solve this instability:

$$(u_{n+1}, v_{n+1}) = \theta_{uv}(u^*_{n+1}, v^*_{n+1}) + (1 - \theta_{uv})(u_n, v_n) \tag{4.33}$$

Then, the method performances with this estimator were completely driven by this parameter choice, that was done empirically for each test case as a theoretical adjustment would have been very fastidious. In general, using small values for this relaxation parameter (e.g. $\theta_{uv} = 0.1$) enabled to reduce the estimator variability and so seemed to offer a greater stability. However, the resolution was still unstable in steady-state as the Fourier coefficients did not converge which led to erratic oscillations of the step size. Consequently, the SPM was not able to use large time steps in a stable way. Several unsuccessful approaches have been tested in order to enhance this outcome but the underlying heuristics were generally not suited for industrial applications. This stability issue and the tested stabilization methods are presented in more details in chapter 5.

That's why the final estimator aimed at avoiding such kind of hazardous heuristics. Finally, the idea behind the estimation process is to compute the Fourier coefficients (u_{n+1}, v_{n+1}) that minimize an energy-function measuring the stationarity of the simulated system defined by:

$$R(u, v) = \int_{t_{n+1}}^{t_{n+1}+T} ||\bar{\rho}(t)||^2 dt \tag{4.34}$$

with $||x||^2 = x^T x$ and

$$\bar{\rho}(t) = F\left(t, \begin{bmatrix} \bar{X}_s(t) \\ X_{ns,n+1} \end{bmatrix}, \begin{bmatrix} \dot{\bar{X}}_s(t) \\ 0 \end{bmatrix}\right) \tag{4.35}$$

in which $\bar{X}_s(t) = u\sin(\omega t) + v\cos(\omega t)$ and $\dot{\bar{X}}_s(t) = \omega(u\cos(\omega t) - v\sin(\omega t))$. The objective-function (5.53) is derived from the power systems steady-state properties described in chapter 2. Indeed, the motivation here is to compute the Fourier coefficients of oscillating components corresponding to the system in steady-state. Thus, it is assumed that, for $t \geq t_{n+1}$, the solution oscillating components are sinusoids with constant Fourier coefficients, i.e. $X_s(t) \approx u_{n+1}\sin(\omega t) + v_{n+1}\cos(\omega t)$ and that the non-oscillating components are constant $X_{ns}(t) \approx X_{ns,n+1}$, the approximation of X_{ns} at time t_{n+1}, and so $\dot{X}_{ns}(t) \approx 0$.

To compute the Fourier coefficients, they are expressed in an incremental way, i.e.

$$\begin{bmatrix} u_{n+1} \\ v_{n+1} \end{bmatrix} = \begin{bmatrix} u_n \\ v_n \end{bmatrix} + \begin{bmatrix} \Delta^u_{n+1} \\ \Delta^v_{n+1} \end{bmatrix} \tag{4.36}$$

Then, the problem is linearized around the guessed steady-state trajectory $\begin{bmatrix} \bar{X}_{s,n}(t) \\ X_{ns,n+1} \end{bmatrix}$ which leads to the following linear system (5.66). It corresponds to Euler's equation

applied to the linearized optimization problem:

$$\begin{bmatrix} H_n^{uu} & H_n^{uv} \\ H_n^{vu} & H_n^{vv} \end{bmatrix} \begin{bmatrix} \Delta_{n+1}^u \\ \Delta_{n+1}^v \end{bmatrix} = - \begin{bmatrix} g_n^u \\ g_n^v \end{bmatrix} \tag{4.37}$$

Since the estimation is done in the future (interval $[t_{n+1}, t_{n+1} + T]$ instead of $[t_{n+1} - T, t_{n+1}]$), the estimation is much less coupled to the integration process. Actually, the only links between the integration step and the estimation step are the non-oscillating components of the solution.

The last estimator is presented in more details in chapter 5, which further develops the crucial role of the Fourier coefficients estimator on the entire method performances. Indeed, in addition to the potential step size limitation due to an incomplete absorption of the sinusoidal behavior within the SPM periodic part, we explain its effect on the global SPM stability. By making some assumptions that limit the operational usability of such a study, we will see that the global error amplification can be modeled with a linear system, whose amplification matrix eigenvalues, and especially its spectral radius, give the method stability. We will then present the different tested heuristics to stabilize the initial estimator that finally led us to the final estimator design and development.

4.5 Algorithm

In this section, the SPM algorithm is summarized in pseudo-code. It mainly focuses on the integrator point-of-view in order to give hints for the implementation of our method into a reference solver. This point is detailed in the chapter 7 which presents the integration of the SPM into SUNDIALS IDA.

Algorithm 1: Sinusoidal Predictor Method (Nordsieck implementation)

Input data: X_0, (u_0, v_0), $\mathcal{I}_s$, TOL, h_{max} ;

while $t_n < t_{final}$ *AND* $n \leq N_{max}$ **do**

  while *acceptable = FALSE (step size not accepted)* **do**

    $t_{n+1} = t_n + h_n$;

    Continuity condition (4.15): $\delta_n^{+(j)} = X_n^{(j)} - \bar{X}_n^{(j)}(t_n)$;

    Prediction step (4.19): $\hat{\delta}_{n+1}^{-(j)} = \hat{\delta}_n^{(j)}(t_{n+1})$;

    Correction step : Solving (4.22)-(4.23)

$$\begin{cases} H_n(\delta_{n+1}^-) = \begin{bmatrix} \delta_{x,n+1}^- - \delta_{x,n}^+ - \dfrac{h_n}{2}\left[\dot{\delta}_{x,n}^+ + \Phi_n(t_{n+1}, \delta_{x,n+1}^-, \delta_{y,n+1}^-)\right] \\ \Gamma_n(t_{n+1}, \delta_{x,n+1}^-, \delta_{y,n+1}^-) \end{bmatrix} = 0 \\ \delta_{n+1(0)}^- = \hat{\delta}_{n+1}^- \end{cases} ;$$

    Prediction error computation (4.26): $\Delta_{n+1}^- = \delta_{n+1}^- - \hat{\delta}_{n+1}^-$;

    LTE estimation (4.30): $e_{n+1} = c_n ||\Delta_{n+1}^-||$;

    Step size update coefficient (4.29): $\alpha_n = k_s \sqrt[3]{\dfrac{TOL}{e_{n+1}}}$;

    if $LTE < TOL$ **then**

      $h_{n+1} = \min(\alpha_n h_n, h_{max})$;

      $acceptable = TRUE$;

      δ Nordsieck vector update (4.25):
      $\vec{\delta}_{n+1,i}^- = \vec{\hat{\delta}}_{n+1,i}^- + \vec{l}_{n,i}^{Nordsieck} \Delta_{n+1,i}^-$ for $i = 1, ..., d$;

      X Nordsieck vector update (4.16): $X_{n+1}^{(j)} = \delta_{n+1}^{-(j)} + \bar{X}_n^{(j)}(t_{n+1})$;

      Periodic part update (4.36): (u_{n+1}, v_{n+1}) parametric estimation ;

      $n = n + 1$;

    else

      $h_n = \alpha_n h_n$;

      $acceptable = FALSE$;

5

Estimator choice

Contents

In order to update the periodic part of the solution $\bar{X}_{n+1}(t)$, the SPM uses a parametric estimator for computing the Fourier coefficients (u_{n+1}, v_{n+1}). In this chapter, we further develop the importance of this estimator and especially its impact on the global method behavior. In particular, as the SPM combines a numerical integrator with a parametric estimator within a closed-loop system, their coupling can make the entire solving process unstable. The estimator choice thus drives the global method performances, especially when the simulated system is in steady-state. Indeed, as the oscillating variables Fourier coefficients should then tend toward constant values, our objective is that the SPM

Fourier coefficients also converge toward these theoretical asymptotic values. By this way, the correction term would be damped and large time steps could be used for integrating it.

As our initial estimator led to such unstable behaviors, several optimization strategies and even other estimators have been tried. They will be briefly presented in this chapter. Therefore, in a first time, we introduce different implemented strategies for stabilizing the initial estimator from a relaxation method. The main difficulty of this approach was to find optimal values for the relaxation parameter. That's why it did not enable to efficiently solve the SPM stability issue. Then, an other estimator based on a trigonometric representation of the correction term has been tried. The motivation was to ensure its convergence when introducing perfect data, which was not verified with the initial estimator. However, when introducing real data, i.e. coming from the numerical integration, it was even more unstable than the initial estimator and so the convergence results were not satisfying. A first real enhancement was obtained by developing an acceptation-rejection criterion for filtering the Fourier coefficients from a measurement of the system stationarity.

Finally, by directly computing the Fourier coefficients from this concept, excellent convergence results have been obtained. The corresponding final estimator is detailed, along with a few choice rules and the associated methodology are proposed for inferring on the a priori properties of an estimator.

5.1 Framework

5.1.1 Considered problem

In this chapter, we use a manufactured solution:

$$X_{th}(t) = u_{th}(t)\sin(\omega t) + v_{th}(t)\cos(\omega t) \tag{5.1}$$

with

$$u_{th}(t) = u_\infty + (u_0 - u_\infty)e^{\lambda t} \tag{5.2}$$

$$v_{th}(t) = v_\infty + (v_0 - v_\infty)e^{\lambda t} \tag{5.3}$$

$$\lambda < 0 \tag{5.4}$$

By computing the time-derivative of (5.1), we can derive the studied linear ODE with an

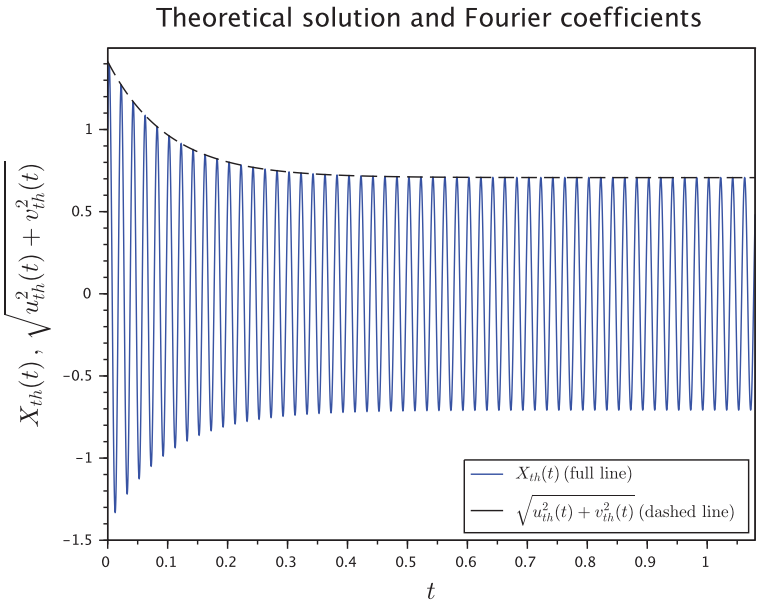

Figure 5.1: Theoretical solution and envelope (on the abscissa: time in second, on the ordinate: solution in arbitrary units).

oscillating forcing term $\bar{b}$:

$$\begin{cases} \dot{X}(t) & = \lambda X(t) + \bar{b}(t) \\ X(0) & = X_0 \\ \text{with } \lambda & < 0 \end{cases} \tag{5.5}$$

where

$$\bar{b}(t) = -(\lambda u_\infty + \omega v_{th}(t))\sin(\omega t) - (\lambda v_\infty - \omega u_{th}(t))\cos(\omega t) \tag{5.6}$$

In this case, the correction term catches the envelope time variation since the sinusoidal predictor is based on piecewise-constant Fourier coefficients. Let us define the theoretical periodic part of the solution with piecewise-constant parameters:

$$\bar{X}_{th,n}(t) = \underbrace{u_{th}(t_n)}_{u_{th,n}}\sin(\omega t) + \underbrace{v_{th}(t_n)}_{v_{th,n}}\cos(\omega t) \tag{5.7}$$

As previously mentioned, the local theoretical correction, which is based on the piecewise-constant Fourier coefficients model, catches the local envelope variation between times t_n and t, with $t \in [t_n, t_{n+1}]$, as it is defined by

$$\delta_{th,n}(t) = X_{th}(t) - \bar{X}_{th,n}(t) \tag{5.8}$$

$$= (u_{th}(t) - u_{th,n})\sin(\omega t) + (v_{th}(t) - v_{th,n})\cos(\omega t) \tag{5.9}$$

whose time-derivative is given by

$$\dot{\delta}_{th,n}(t) = \dot{u}_{th}(t)\sin(\omega t) + \dot{v}_{th}(t)\cos(\omega t) \tag{5.10}$$

$$+ \omega\left[(u_{th}(t) - u_{th,n})\cos(\omega t) - (v_{th}(t) - v_{th,n})\sin(\omega t)\right] \tag{5.11}$$

As a result, this system enables to simply evaluate the SPM performances. In particular, in steady-state, $(u_{th}(t), v_{th}(t)) \to (u_\infty, v_\infty)$ and so $\delta_{th,n}(t) \to 0$. The SPM objective is then to fit the periodic steady-state solution by properly estimating the asymptotic Fourier coefficients in order to damp the correction term. By this way, large time steps could be used.

5.1.2 Notations

In this chapter, we will focus on the scalar ordinary differential equation (5.5) whose solution is oscillating. Then, the following notations will be used:

- $X_{th}(t_n)$ is the theoretical solution at time t_n. Let X_n be its approximation by the SPM. Hence, $\epsilon_n = X_n - X_{th}(t_n)$ is the global error on the solution at time t_n;
- $(u_{th,n}, v_{th,n}) = (u_{th}(t_n), v_{th}(t_n))$ are the theoretical Fourier coefficients at time t_n. Let (u_n, v_n) be their approximation by the SPM. Hence, $(\epsilon_n^u, \epsilon_n^v) = (u_n - u_{th}(t_n), v_n - v_{th}(t_n))$ is the global error on the Fourier coefficients at time t_n;
- $\bar{X}_{th,n}(t) = u_{th,n}\sin(\omega t) + v_{th,n}\cos(\omega t)$ is the theoretical local periodic part of the solution in the time interval $[t_n, t_{n+1}]$. Let $\bar{X}_n = u_n\sin(\omega t) + v_n\cos(\omega t)$ be the approximated periodic function used by the SPM. Hence, $\bar{X}_{\epsilon_n^{uv}}(t) = \bar{X}_n(t) - \bar{X}_{th,n}(t) = \epsilon_n^u\sin(\omega t) + \epsilon_n^v\cos(\omega t)$ is the error of periodic function in the interval $[t_n, t_{n+1}]$, i.e. the periodic function whose parameters are equal to the error on the Fourier coefficients $(\epsilon_n^u, \epsilon_n^v)$;
- $\delta_{th,n}(t) = X_{th}(t) - \bar{X}_{th,n}(t)$ is the theoretical local correction term. For instance, if $X_{th}(t)$ is a phasor-like solution i.e. $X_{th}(t) = u_{th}(t)\sin(\omega t) + v_{th}(t)\cos(\omega t)$, then the theoretical local correction term catches the envelope variation $\delta_{th,n}(t) = (u_{th}(t) - u_{th,n})\sin(\omega t) + (v_{th}(t) - v_{th,n})\cos(\omega t)$. Let $\delta_n^{+(j)}$ and $\delta_{n+1}^{-(j)}$ be respectively the right value of the j-th time-derivative of the correction term computed by the SPM at time t_n and the left value of the j-th time-derivative of the correction term computed by the SPM at time t_{n+1}. Hence, $\epsilon_n^{\delta+} = \delta_n^+ - \delta_{th,n}(t_n)$ and $\epsilon_{n+1}^{\delta-} = \delta_{n+1}^- - \delta_{th,n}(t_{n+1})$ are respectively the error on the correction term at time t_n and at time t_{n+1};

5.2 Impact of the estimator on the performances

5.2.1 Impact of the Fourier coefficients error on the step size limitation

As mentioned in the previous sections, the step size is limited by the simulated variables frequency. For instance, if the trapezoidal formula is used for numerical integration, we recall that the maximum usable step size can be roughly approximated by

$$h_n = \sqrt[3]{\frac{12TOL}{||X_n^{(3)}||}} \tag{5.12}$$

For instance, if $X(t)$ is an oscillating variable in steady-state, $||X_n^{(3)}|| = \omega^3||X_n||$ where $||X_n||$ is the sinusoid amplitude. Similarly, an approximation of the maximum step size resulting from the application of the SPM is given by

$$h_n = \sqrt[3]{\frac{12TOL}{||\delta_n^{+(3)}||}} \tag{5.13}$$

So, let us consider a sinusoidal solution in steady-state i.e.

$$X(t) = u_{th,\infty}\sin(\omega t) + v_{th,\infty}\cos(\omega t) \tag{5.14}$$

If the Fourier coefficients used in the periodic part of SPM have an error $(\epsilon_n^u, \epsilon_n^v)$ i.e.

$$\bar{X}_n(t) = u_n\sin(\omega t) + v_n\cos(\omega t) \tag{5.15}$$

$$= (u_{th,\infty} + \epsilon_n^u)\sin(\omega t) + (v_{th,\infty} + \epsilon_n^v)\cos(\omega t) \tag{5.16}$$

Then the theoretical local correction term associated to the input Fourier coefficient (u_n, v_n) is given by

$$\delta_{th}|_{(u_n,v_n)}(t) = X(t) - \bar{X}_n(t) = -\bar{X}_{\epsilon_n^{uv}}(t) = -\epsilon_n^u\sin(\omega t) - \epsilon_n^v\cos(\omega t) \tag{5.17}$$

which means that the maximum usable step size can be approximated by

$$h_n = \sqrt[3]{\frac{12TOL}{\omega^3\sqrt{(\epsilon_n^u)^2 + (\epsilon_n^v)^2}}} \tag{5.18}$$

In the figure 5.2, we can see that the SPM is particularly sensitive to the error on the Fourier coefficients: even for an error being hundred times smaller than the tolerance, the step size cannot reach high values, i.e. in the range of the second. In conclusion, the SPM Fourier coefficients have to be very accurately approximated in steady-state for the method to be efficient.

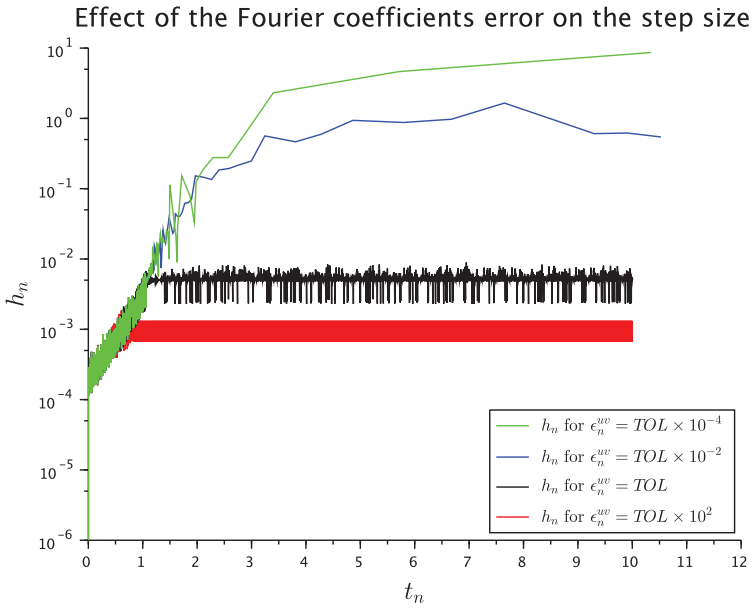

Figure 5.2: Illustration of the effect of the Fourier coefficients error on the step size used by the integration scheme with a tol=1.e-6 (on the abscissa: time in seconds, on the ordinate: step size in seconds). To produce this graph, at each time step, the exact Fourier coefficients are injected and perturbed with a fixed value ($\epsilon_n^{uv} = \{10^{-4}, 10^{-6}, 10^{-8}, 10^{-10}\}$). When the solution is in transients, we can see that this perturbation does not significantly impact the used step size. Indeed, during this phase, the Fourier coefficients time-derivatives are not negligible and so are greater than the introduced perturbation. On the contrary, when the system is in steady-state, this perturbation directly affects the used step size as it becomes prevailing.

5.2.2 Impact of the estimator on the entire method stability

In the previous section, the effect of the Fourier coefficients error on the step size has been presented. This first limitation is directly linked to the control of the integration error by the adaptive step size strategy. In addition to this direct effect, choosing an inappropriate estimator can lead to numerical instabilities. Indeed, we recall that the SPM roughly consists in two main steps:

- The numerical integration of the DAE system on the correction. In this step, the local equations on the correction term are obtained by injecting the last computed Fourier coefficients into the global equations on the whole solution. Hence, the error on the Fourier coefficients adds a contribution to the integration error on the correction and so to the integration error on the entire solution.
- The parametric estimation of the Fourier coefficients, which is based on the previ-

ously computed data. Then, the integration errors on the solution are injected into the estimator. This contributes to the Fourier coefficients bias

In figure 5.3, the resulting error re-injection cycle is presented. Indeed, as the two below-presented steps are performed within a global closed-loop solving process, the error resulting from the integration step is injected into the estimation step, and vice versa. As we want to avoid deep modifications of the underlying basic integration scheme, our objective has been to develop an estimator that does not introduce too much numerical instabilities in order to keep the whole process stable. This instability is particularly illustrated in the following section about the initial estimator. Such an instability limits the maximum step size usable by the method [58].

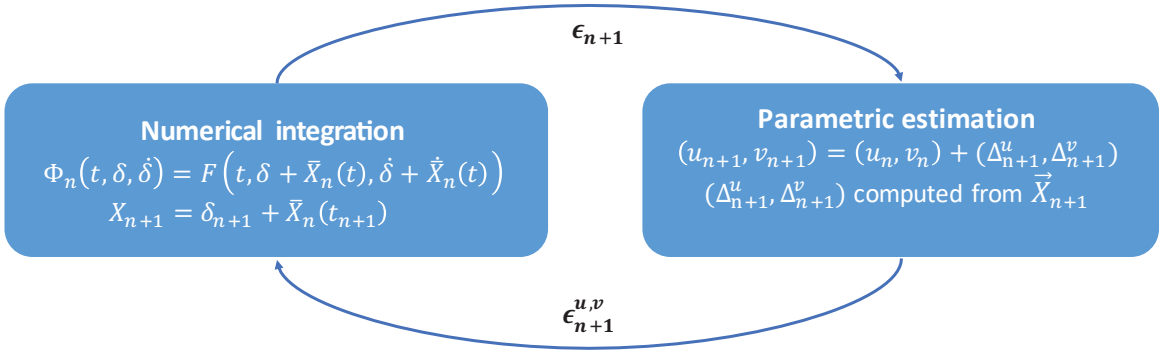

Figure 5.3: Illustration of the coupling between the integration and estimation steps. ϵ_{n+1} and $\epsilon^{u,v}_{n+1}$ respectively stand for the error on the solution and on the Fourier coefficients at time t_{n+1}. The numerical errors resulting from the numerical integration are introduced into the estimator and impact the Fourier coefficients bias which is then re-injected into the numerical integrator. Hence, the estimator choice has to be made such as the entire process is stable for the Fourier coefficients to converge in steady-state.

Then, as a first approximation, the error amplification between time steps t_n and t_{n+1} can be modeled with the following linear system:

$$\left[\begin{array}{c}\epsilon_{n+1}\\ \hline \epsilon^u_{n+1}\\ \epsilon^v_{n+1}\end{array}\right] = \underbrace{\left[\begin{array}{c|cc} A^{XX}_n & A^{Xu}_n & A^{Xv}_n \\ \hline A^{uX}_n & A^{uu}_n & A^{uv}_n \\ A^{vX}_n & A^{vu}_n & A^{vv}_n \end{array}\right]}_{\overset{def}{=} A_n} \left[\begin{array}{c}\epsilon_n\\ \hline \epsilon^u_n\\ \epsilon^v_n\end{array}\right] + \left[\begin{array}{c}e_n\\ \hline e^u_n\\ e^v_n\end{array}\right] \tag{5.19}$$

In the matrix A_n, the upper-left part containing only A^{XX}_n corresponds to the amplification coefficient of the integration scheme. For instance, for the trapezoidal formula this coefficient is equal to $A^{XX}_n = \dfrac{1+\dfrac{h_n\lambda}{2}}{1-\dfrac{h_n\lambda}{2}}$. The lower-right part constituted of the sub-matrices A^{uu}_n, A^{uv}_n, A^{vu}_n and A^{vv}_n could be seen as the amplification matrix corresponding

to the autonomous dynamics of the Fourier coefficients estimator. In section 3.1.3, the presented heuristic for stabilizing the initial Fourier coefficients estimator uses this matrix eigenvalues in order to amortize its bias. The sub-matrices A_n^{Xu}, A_n^{Xv}, A_n^{uX} and A_n^{vX} correspond to the coupling between the integration and estimation steps.

5.2.3 Estimator expected properties

The whole method performances being directly influenced by the periodic solution accuracy, this estimator should be unbiased. More precisely, it should be consistent, particularly in steady-state,. i.e. if $h_n \to 0$ then $\epsilon_{n+1}^u, \epsilon_{n+1}^v \to 0$. As a result, it should converge for perfect input data, i.e. when $\delta_{n+1}^{-(j)} = X_{th}^{(j)}(t_{n+1}) - \bar{X}_n^{(j)}(t_{n+1})$. In addition, the Fourier coefficients increment should decrease with the step size, i.e. $(\Delta_{n+1}^u, \Delta_{n+1}^v) = (u_{n+1}, v_{n+1}) - (u_n, v_n) \to (0, 0)$ for $h_n \to 0$.

By this way, the estimator would have properties close to those of the basic predictor-corrector in order to ensure the error damping and so the convergence in steady-state for being able to use high step size. Indeed, as reducing the step size used for the integration enables a better computation of $\delta_{n+1}^{-(j)}$ by consistence of the integration scheme, the Fourier coefficients estimator should imperatively have the same property in order the SPM to globally enhance the solution and so to reach optimal performances.

For instance, in figure 5.4, some results obtained with the initial estimator and the use of the relaxation strategy for smoothing the Fourier coefficients variations are presented. For producing these results, theoretical data have been injected into the estimator, i.e. $\delta_n^{-(j)} = X_{th}^{(j)}(t_n) - \bar{X}_n^{(j)}(t_n)$, but we can see that the estimator does not converge in steady-state. By reducing the proportion of Fourier coefficients update, i.e. by using smaller θ with $(u_{n+1}, v_{n+1}) = \theta(u_{n+1}^*, v_{n+1}^*) + (1 - \theta)(u_n, v_n)$, the variability is less important but the estimator still seems unstable and so non-convergent. That is why the final estimator should ideally present a much smoother and more controlled behavior, even if the convergence is a bit slower.

5.3 SPM Fourier coefficients estimator optimization

In this section, we present some tried strategies for optimizing the parametric estimator and so the SPM performances.

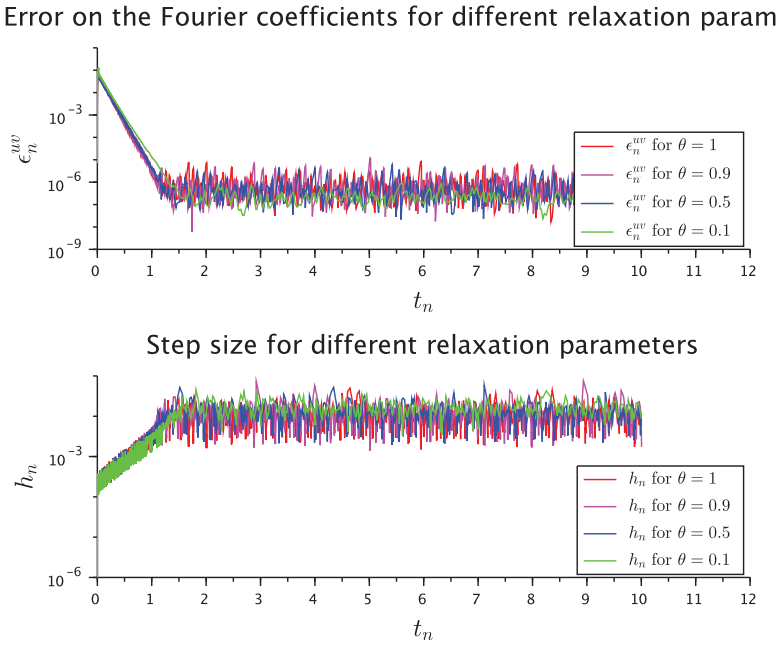

Figure 5.4: Results obtained with the initial estimator using different fixed relaxation parameters when theoretical data are introduced, i.e. when $\delta_n^{-(j)} = X_{th}^{(j)}(t_n) - \bar{X}_n^{(j)}(t_n)$. Top: Fourier coefficients error (on the abscissa: time in seconds, on the ordinate: Fourier coefficients error in arbitrary units), down: step size (on the abscissa: time in seconds, on the ordinate: step size in seconds). We can see that the estimator does not converge and seems particularly unstable.

5.3.1 Initial estimator

For enhancing the initial estimator corresponding to the classical Fourier coefficients formula, we have introduced a relaxation coefficient in order to smooth and control their variations. Hence, in this subsection, we present in more details the initial estimator, the basic idea of the relaxation coefficient and, to finish, an automatic adjustment heuristic that we have tried.

5.3.1.1 Initial estimator presentation

At first glance, the parameters have been updated with the classical Fourier coefficients formulas:

$$u_{n+1,i} = \frac{2}{T} \int_{t_{n+1}-T}^{t_{n+1}} X_i(t) \sin(\omega t) dt \tag{5.20}$$

$$v_{n+1,i} = \frac{2}{T} \int_{t_{n+1}-T}^{t_{n+1}} X_i(t) \cos(\omega t) dt \tag{5.21}$$

where $i \in \mathcal{I}_s$. Then, in order to lower the CPU time per iteration and, thus, to preserve the method efficiency, the Fourier coefficients are computed from explicit formulas which are only based on the already computed points and not on a re-sampling of the signal. Let us illustrate our strategy by detailing the computation of u_{n+1}.

First of all, we substitute X by its approximate full parametric representation :

$$X(t) \approx \bar{X}(t) + \hat{\delta}(t) \tag{5.22}$$

Where $\hat{\delta}(t)$ is the second order polynomial representing the predictor on δ, which is defined by the method and corresponds in our case to the second order Adams-Bashford scheme. Then, the integrals to compute are :

$$\sigma_{\bar{X}} = \frac{2}{T}\int_{t_{n+1}-T}^{t_{n+1}} \bar{X}(t)\sin(\omega t)dt \tag{5.23}$$

$$\sigma_{\delta} = \frac{2}{T}\int_{t_{n+1}-T}^{t_{n+1}} \hat{\delta}(t)\sin(\omega t)dt \tag{5.24}$$

$$u_{n+1} = \sigma_{\bar{X}} + \sigma_{\delta} \tag{5.25}$$

For a general interval $[a, b]$, with $a, b \in \mathbf{R}_+$, the former is given by :

$$\sigma_{\bar{X}} = \frac{2}{T}\int_{a}^{b} u\sin^2(\omega t) + v\sin(\omega t)\cos(\omega t)dt \tag{5.26}$$

This integral is calculated using common trigonometric formulae, while the latter is computed using the fact that $\hat{\delta}(t)$ is a second order polynomial. Thus, using the integration by parts formula, we have

$$\sigma_{\delta} = \frac{2}{T}\left[\hat{\delta}(t)\frac{-\cos(\omega t)}{\omega} + \dot{\hat{\delta}}(t)\frac{\sin(\omega t)}{\omega^2} + \ddot{\hat{\delta}}(t)\frac{\cos(\omega t)}{\omega^3}\right]_a^b \tag{5.27}$$

On the equations above, the n index is not specified on both $\bar{X}$ and $\hat{\delta}$ since the model is piecewise defined, and so the calculation is split between the different solver intervals. In the following equations, let q be the index such as $t_{n+1} - T \in [t_q, t_{q+1}]$. Then, the integral on the whole interval $[t_{n+1} - T, t_{n+1}]$ is decomposed as below :

$$\begin{aligned}\int_{t_{n+1}-T}^{t_{n+1}} X(t)\sin(\omega t)dt = &\int_{t_{n+1}-T}^{t_{q+1}} X(t)\sin(\omega t)dt \\ &+ \sum_{j=q+1}^{n}\int_{t_j}^{t_{j+1}} X(t)\sin(\omega t)dt\end{aligned}$$

Then, for each interval $[t_j, t_{j+1}]$, we have

$$\begin{aligned}\int_{t_j}^{t_{j+1}} X(t)\sin(\omega t)dt = &\frac{2}{T}\int_{t_j}^{t_{j+1}} u_j\sin^2(\omega t)dt \\ &+ \frac{2}{T}\int_{t_j}^{t_{j+1}} v_j\sin(\omega t)\cos(\omega t)dt \\ &+ \int_{t_j}^{t_{j+1}} \hat{\delta}_j(t)\sin(\omega t)dt\end{aligned}$$

Finally, by calculating the associated explicit formulas, updating the Fourier coefficients only requires fast analytic formulas evaluations, especially since the sum of local Fourier integrals part (i.e. the integrals whose interval is $[t_j, t_{j+1}]$) can be stored. However, even if this estimator is very efficient in terms of computational time thanks to these optimization possibilities, it leads to a global unstable solving process. As the integration errors coming from the integration of the equations on the correction term are injected into the estimator, they are then re-injected within the next time step because of their impact on the Fourier coefficients errors. Even when introducing perfect input data, this estimator has an unstable and non-convergent behavior in steady-state (see figures 5.4).

5.3.1.2 Relaxation parameter introduction

In order to stabilize the previously presented estimator, a relaxation parameter has been introduced in order to smooth the Fourier coefficients evolution. Then, two temporary Fourier coefficients estimations $u^*_{n+1,i}$ and $v^*_{n+1,i}$ are first computed from the previously exposed formulas (respectively (5.20) and (5.21)) and the associated methodology. In a second time, the relaxation is applied to average $(u^*_{n+1,i}, v^*_{n+1,i})$ and (u_n, v_n) in order to smooth the evolution between two consecutive time steps:

$$u_{n+1,i} = \theta u^*_{n+1,i} + (1-\theta)u_{n,i} \tag{5.28}$$

$$v_{n+1,i} = \theta v^*_{n+1,i} + (1-\theta)v_{n,i} \tag{5.29}$$

where $\theta \in [0, 1]$. If $\theta = 1$, the effect of the relaxation is suppressed and we get the initial estimator. The value of the relaxation parameter θ is empirically fixed as a theoretical optimal value is hard to determine. From our tests whose results are presented in figures 5.5 and 5.4, using high values for θ leads to a high sensitivity of the method to Fourier coefficients variations and so to important instabilities. Indeed, for high values (e.g. $\theta \in [0, \frac{1}{2}]$), an important variability of the Fourier coefficients can be observed, even when the system is in steady-state. On the contrary, for smaller values (e.g. $\theta \in]0, \frac{1}{2}[$), the variability is less important. Then, the Fourier coefficients could seem to converge toward their asymptotic values in steady-state, but instabilities are also noted for large time steps. In addition, the "optimal" value of θ seems really problem-dependent and, particularly, the effect of the relaxation parameter is not linear at all: one could imagine that setting θ to a very small value such as $\theta = 0.01$ leads a slower but surer convergence, but it is not verified in practical applications. Moreover, tuning this parameter is very sensitive, then even if two values give good results, using the mid-value may lead to radically different results.

In addition, in figure 5.4, we can see that this estimator is not convergent for perfect input data. On the contrary, it seems like the convergence is even worse than when using the

normal data in output of the integrator.

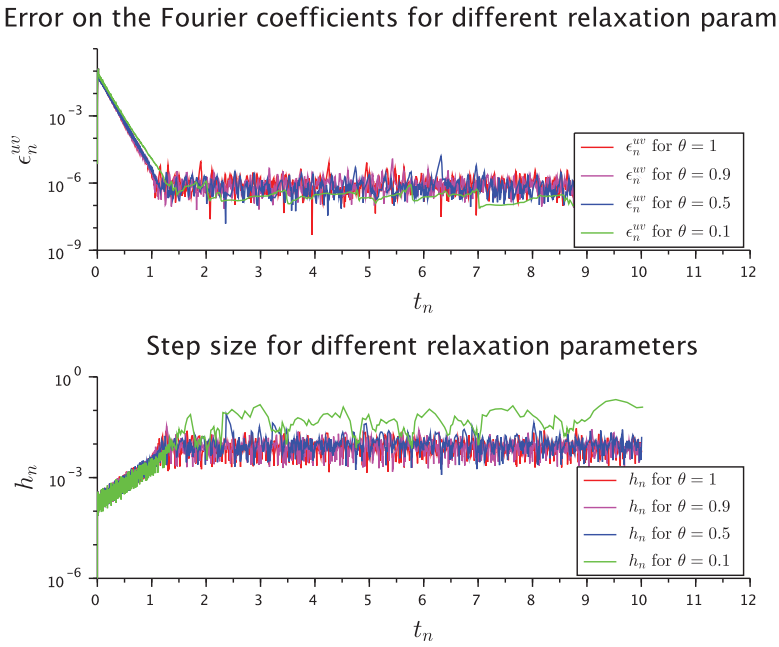

Figure 5.5: Performances comparison when using the relaxation with fixed θ values. Top: Fourier coefficients error (on the abscissa: time in seconds, on the ordinate: Fourier coefficients error in arbitrary units), down: step size (on the abscissa: time in seconds, on the ordinate: step size in seconds). The different colored lines correspond to the different tested values. The only configuration almost leading to the convergence of the Fourier coefficients and the use of large step sizes is for $\theta = 0.1$. However, we can see that the computations still seem unstable as the step size fluctuates for high values.

5.3.1.3 Relaxation parameter automatic adjustment

As fixing the relaxation parameter seemed inefficient, another tried optimization has been to automatically adjust the parameter θ used for smoothing the Fourier coefficients evolution.

In order to design such an automatic adjustment of the relaxation parameter, we make the following assumptions:

- The system is in steady-state: $X_{th}(t) = \bar{X}_{th,\infty}(t) = u_\infty \sin(\omega t) + v_\infty \cos(\omega t)$;
- The step size is greater or equal to one period: $h_n \geq T$;
- The input data are perfect: $\delta_n^{+(j)} = X_{th}^{(j)}(t_n) - \bar{X}_n^{(j)}(t_n)$;
- In order to obtain a linear amplification system, we assume that there is an error on the Fourier coefficients (u_n, v_n) at time t_n, i.e. $u_n = u_\infty + \epsilon_n^u$, $v_n = v_\infty + \epsilon_n^v$ and

$\bar{X}_n(t) = u_n \sin(\omega t) + v_n \cos(\omega t)$.

The objective is to adjust the relaxation parameter in order to stabilize the estimation. By using the second hypothesis, a simple recurrent formula can be used for describing the Fourier coefficients error propagation. The third hypothesis enables to delete the coupling phenomenon between the estimation process and the numerical integration of δ. In other words, we consider the estimator "individual" stability as previously mentioned, which leads to the study of the following linear system:

$$\begin{bmatrix} \epsilon_{n+1}^u \\ \epsilon_{n+1}^v \end{bmatrix} = \underbrace{\begin{bmatrix} A_n^{uu} & A_n^{uv} \\ A_n^{vu} & A_n^{vv} \end{bmatrix}}_{A_n(\theta)} \begin{bmatrix} \epsilon_n^u \\ \epsilon_n^v \end{bmatrix} \tag{5.30}$$

For recall, the formula of the Fourier coefficients is given by (5.20) and (5.28). As $h_n \geq T$, $t_{n+1} - T \in [t_n, t_{n+1}]$. The estimator u_{n+1}^* can then be written without summation of the local Fourier integrals (now detailing the computations for u_{n+1}):

$$u_{n+1}^* = \frac{2}{T} \int_{t_{n+1}-T}^{t_{n+1}} [\bar{X}_n(t) + \hat{\delta}_n(t)] \sin(\omega t) dt \tag{5.31}$$

where

$$\hat{\delta}_n(t) = \delta_n^+ + \dot{\delta}_n^+ (t - t_n) + \ddot{\delta}_n^+ \frac{(t - t_n)^2}{2}$$

$$\delta_n^{+(j)} = X_{th}^{(j)}(t_n) - \bar{X}_n^{(j)}(t_n) = -\bar{X}_{\epsilon_n^{uv}}^{(j)}(t_n) = -\frac{d^j}{dt^j} \left(\epsilon_n^u \sin(\omega t_n) + \epsilon_n^v \cos(\omega t_n)\right)$$

$$\Rightarrow u_{n+1}^* = u_n \underbrace{\frac{2}{T} \int_{t_{n+1}-T}^{t_{n+1}} \sin^2(\omega t) dt}_{=1} + v_n \underbrace{\frac{2}{T} \int_{t_{n+1}-T}^{t_{n+1}} \sin(\omega t) \cos(\omega t) dt}_{=0} + \underbrace{\frac{2}{T} \int_{t_{n+1}-T}^{t_{n+1}} \hat{\delta}_n(t) \sin(\omega t) dt}_{\equiv \sigma_n} \tag{5.32}$$

After the injection of the relaxation parameter θ, we have $u_{n+1} = u_n + \theta \frac{2}{T} \sigma_n$. As $\hat{\delta}_n$ is polynomial, σ_n is simply calculated from integration by parts :

$$\begin{aligned} \sigma_n = & \left(\hat{\delta}_n(t_{n+1}) - \hat{\delta}_n(t_{n+1} - T)\right) \frac{-\cos(\omega t_{n+1})}{\omega} \\ & - \left(\dot{\hat{\delta}}_n(t_{n+1}) - \dot{\hat{\delta}}_n(t_{n+1} - T)\right) \frac{-\sin(\omega t_{n+1})}{\omega^2} \\ & + \left(\ddot{\hat{\delta}}_n(t_{n+1}) - \ddot{\hat{\delta}}_n(t_{n+1} - T)\right) \frac{\cos(\omega t_{n+1})}{\omega^3} \end{aligned}$$

with

$$\hat{\delta}_n(t_{n+1}) - \hat{\delta}_n(t_{n+1} - T) = T\dot{\delta}_n^+ + (h_n T - \frac{T^2}{2})\ddot{\delta}_n^+$$

$$\dot{\hat{\delta}}_n(t_{n+1}) - \dot{\hat{\delta}}_n(t_{n+1} - T) = T\dot{\delta}_n^+$$

$$\ddot{\hat{\delta}}_n(t_{n+1}) - \ddot{\hat{\delta}}_n(t_{n+1} - T) = 0$$

Thus, after the injection of $\delta_n^{+(j)} = -\bar{X}_{\epsilon_n^{uv}}^{(j)}(t)$ and the simplification (let us introduce $s_n = \sin(\omega t_n)$ and $c_n = \cos(\omega t_n)$) :

$$u_{n+1} = u_n + \theta \underbrace{\frac{2}{T}\left[+Tc_nc_{n+1} + Ts_ns_{n+1} - \left(h_nT - \frac{T^2}{2}\right)\omega s_nc_{n+1}\right]}_{M_n^{uu}} \epsilon_n^u \tag{5.33}$$

$$+ \theta \underbrace{\frac{2}{T}\left[-Ts_nc_{n+1} + Tc_ns_{n+1} - \left(h_nT - \frac{T^2}{2}\right)\omega c_nc_{n+1}\right]}_{M_n^{uv}} \epsilon_n^v \tag{5.34}$$

$$v_{n+1} = v_n + \theta \underbrace{\frac{2}{T}\left[+Ts_ns_{n+1} + Tc_nc_{n+1} + \left(h_nT - \frac{T^2}{2}\right)\omega c_ns_{n+1}\right]}_{M_n^{vv}} \epsilon_n^v \tag{5.35}$$

$$+ \theta \underbrace{\frac{2}{T}\left[-Tc_ns_{n+1} + Ts_nc_{n+1} + \left(h_nT - \frac{T^2}{2}\right)\omega s_ns_{n+1}\right]}_{M_n^{vu}} \epsilon_n^u \tag{5.36}$$

Then, by subtracting u_∞ (and respectively v_∞) in the estimator of u_{n+1} (and respectively of v_{n+1}), the linear system (5.30) describing the error on the Fourier coefficients can be obtained in the form of:

$$\begin{bmatrix}\epsilon_{n+1}^u \\ \epsilon_{n+1}^v\end{bmatrix} = \left(I_d + \theta \begin{bmatrix}M_n^{uu} & M_n^{uv} \\ M_n^{vu} & M_n^{vv}\end{bmatrix}\right)\begin{bmatrix}\epsilon_n^u \\ \epsilon_n^v\end{bmatrix} \tag{5.37}$$

For the estimator to be convergent, the idea is to minimize the modulus of the above matrix eigenvalues. In particular, its spectral radius should be lower than 1, i.e. $\rho(A_n(\theta)) = \max(|\zeta_i|$, where $\zeta_i =$ eigenvalue of $A_n(\theta)) < 1$. The eigenvalues of $A_n(\theta)$ are given by:

$$\zeta_\pm = \frac{2 + \theta(M_n^{uu} + M_n^{vv}) \pm \theta\sqrt{(M_n^{uu} - M_n^{vv})^2 + 4M_n^{uv}M_n^{vu}}}{2} \tag{5.38}$$

By upper-bounding them such as they are lower than one, i.e. $|\zeta_\pm|^2 < 1$, we finally deduce the following objective function:

$$J(\theta) = \underbrace{\left(\frac{(M_n^{uu})^2 + (M_n^{vv})^2}{2} + |M_n^{uv}M_n^{vu}|\right)}_{>0}\theta^2 + (M_n^{uu} + M_n^{vv})\theta + 1 \tag{5.39}$$

This formula actually corresponds to the case where the eigenvalues are complex, which is sometimes not verified. However, our tests suggest that using this objective function leads to better results compared to an objective function distinguishing the two cases (real or complex eigenvalues). The second order coefficients being greater than one, the objective

function is strictly elliptic. So, its minimum is given by the Euler's equation $J'(\theta) = 0$. That is to say

$$\theta_{opt} = \frac{-(M_n^{uu} + M_n^{vv})}{(M_n^{uu})^2 + (M_n^{vv})^2 + 2|M_n^{uv} M_n^{vu}|} \tag{5.40}$$

Then, the proposed heuristic can be implemented as below

- The default relaxation parameter θ_0 is fixed to an arbitrary value (low in order to reduce the variability of u and v, e.g. $\theta_0 = 0.10$) ;
- If $h_n \geq T$, then $\theta_n = \frac{-(M_n^{uu} + M_n^{vv})}{(M_n^{uu})^2 + (M_n^{vv})^2 + 2|M_n^{uv} M_n^{vu}|}$;
- Else, $\theta_n = \theta_0$;
- $u_{n+1} = \theta_n u_{n+1}^* + (1 - \theta_n) u_n$ and $v_{n+1} = \theta_n v_{n+1}^* + (1 - \theta_n) v_n$.

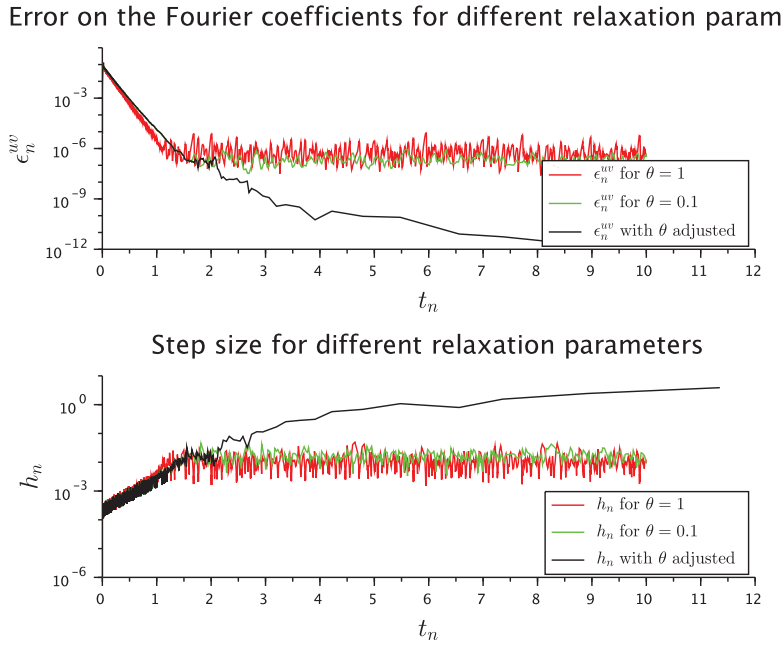

Figure 5.6: Performances comparison of the different versions of the initial Fourier coefficients estimator when perfect input data are introduced (red: non-use of the relaxation, green: use of the relaxation with a fixed parameter, use of the relaxation with an automatically adjusted parameter). Top: Fourier coefficients error (on the abscissa: time in seconds, on the ordinate: Fourier coefficients error in arbitrary units), down: step size (on the abscissa: time in seconds, on the ordinate: step size in seconds). In this case, the adjustment strategy seems really efficient as it is the configuration which leads to the convergence of the Fourier coefficients and so to the use of large step sizes.

In figure 5.6, we can see that the adjustment of the relaxation parameter θ is particularly efficient when we inject perfect input data to the estimator. Indeed, the error on the Fourier coefficients is significantly reduced and in a very regular way. So the step size

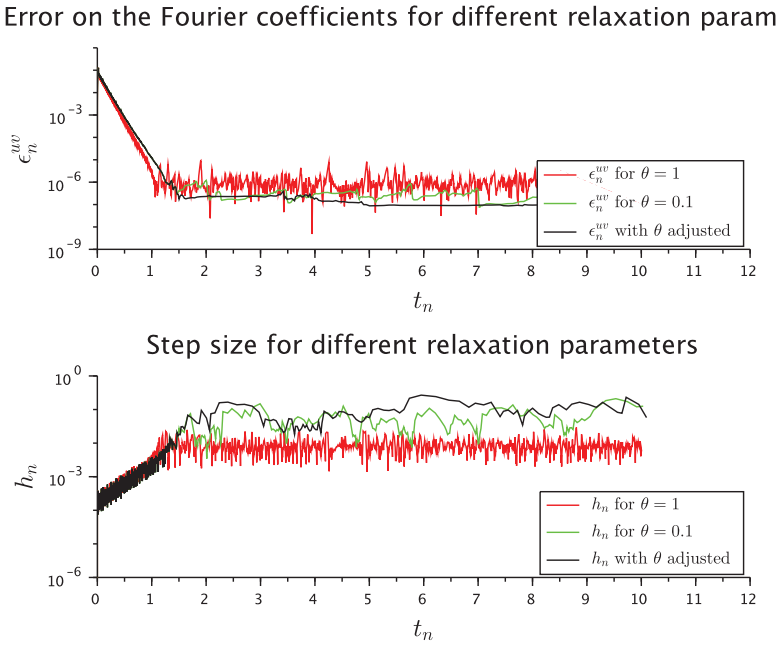

Figure 5.7: Performances comparison of the different versions of the initial Fourier coefficients estimator without introducing perfect input data (red: non-use of the relaxation, green: use of the relaxation with a fixed parameter, use of the relaxation with an automatically adjusted parameter). Top: Fourier coefficients error (on the abscissa: time in seconds, on the ordinate: Fourier coefficients error in arbitrary units), down: step size (on the abscissa: time in seconds, on the ordinate: step size in seconds). In this case, the relaxation parameter adjustment does not significantly enhance the convergence performances.

increases to reach the expected high values without the oscillations that are observed without the adjustment of θ. However, as soon as real input data are used for the integration as in figure 5.7, the performances are significantly deteriorated and using this strategy does not seem to give gains of performances. This is logical since the adjustment heuristic of θ is based on the hypothesis that the input data are perfect. In order to enhance this result, we also tried to use different relaxation parameters for u and v but the outcomes are quite similar. Furthermore, in this case, as the objective function is not elliptic anymore, the resolution is more complex and so requires more computational resources. A solution could be to optimize the spectral radius of the global linear system introduced in section 2.2, taking account of the error on the solution, i.e.

$$\begin{bmatrix} \epsilon_{n+1} \\ \boldsymbol{\epsilon}^{\mathbf{u}}_{\mathbf{n+1}} \\ \boldsymbol{\epsilon}^{\mathbf{v}}_{\mathbf{n+1}} \end{bmatrix} = \begin{bmatrix} A_n^{XX} & A_n^{Xu} & A_n^{Xv} \\ A_n^{uX} & \mathbf{A}^{\mathbf{uu}}_{\mathbf{n}} & \mathbf{A}^{\mathbf{uv}}_{\mathbf{n}} \\ A_n^{vX} & \mathbf{A}^{\mathbf{vu}}_{\mathbf{n}} & \mathbf{A}^{\mathbf{vv}}_{\mathbf{n}} \end{bmatrix} \begin{bmatrix} \epsilon_n \\ \boldsymbol{\epsilon}^{\mathbf{u}}_{\mathbf{n}} \\ \boldsymbol{\epsilon}^{\mathbf{v}}_{\mathbf{n}} \end{bmatrix} \tag{5.41}$$

In this equation, the bold elements are those taken into account in the presented method. However, such a global approach would be extremely costly as the dimension of A_n^{XX} would be equal to the global simulated system dimension. Hence, computing its eigenvalues could not be done in an analytic way, contrary to the method presented here. Finally, computing the optimal smoothing parameter would also require important computational resources.

5.3.2 Estimator based on a correction term trigonometric representation

From the previously exposed requirements and especially those on the consistency, we also tried to implement a very simple estimator based on a trigonometric representation of the correction term δ. The idea is to assume, but only for the estimation step, that the solution is a centered sinusoid and so that the correction term catches the error on the envelope. Hence, in this estimator, the correction term is seen as a sinusoid whose Fourier coefficients are piecewise constant and, in particular, equal to the difference between the instantaneous value of the theoretical Fourier coefficients at the next time step and the used Fourier coefficients, i.e.

$$\delta_{th}|_{(u_n,v_n)}(t) = (u_{th}(t) - u_n)\sin(\omega t) + (v_{th}(t) - v_n)\cos(\omega t) \tag{5.42}$$

Then, at next time step, the correction term would be equal to

$$\delta_{n+1} \approx (u_{n+1} - u_n)\sin(\omega t_{n+1}) + (v_{n+1} - v_n)\cos(\omega t_{n+1}) \tag{5.43}$$

whose time-derivative can be approximated by neglecting the Fourier coefficients time-variation as

$$\dot{\delta}_{n+1} \approx \omega\left[(u_{n+1} - u_n)\cos(\omega t_{n+1}) - (v_{n+1} - v_n)\sin(\omega t_{n+1})\right] \tag{5.44}$$

By this way, if the oscillating solution is truly a centered sinusoid, the estimator should be exact in steady-state when injecting perfect values for δ. However, it will be obviously biased when the system is in transient as the Fourier coefficients time-derivative is neglected. Then, the two equations (5.43) and (5.44) can be summarized in a linear system coupling them, as below

$$\begin{bmatrix} \sin(\omega t_{n+1}) & \cos(\omega t_{n+1}) \\ \omega\cos(\omega t_{n+1}) & -\omega\sin(\omega t_{n+1}) \end{bmatrix} \begin{bmatrix} \Delta_{n+1}^u \\ \Delta_{n+1}^v \end{bmatrix} = \begin{bmatrix} \delta_{n+1} \\ \delta'_{n+1} \end{bmatrix} \tag{5.45}$$

Δ^u_{n+1} and Δ^v_{n+1} are computed by inverting this linear system, which leads to

$$\Rightarrow \begin{bmatrix} \Delta^u_{n+1} \\ \Delta^v_{n+1} \end{bmatrix} = \begin{bmatrix} \sin(\omega t_{n+1}) & \cos(\omega t_{n+1}) \\ \omega\cos(\omega t_{n+1}) & -\omega\sin(\omega t_{n+1}) \end{bmatrix}^{-1} \begin{bmatrix} \delta_{n+1} \\ \dot{\delta}_{n+1} \end{bmatrix}$$
$$= \begin{bmatrix} \sin(\omega t_{n+1}) & \frac{1}{\omega}\cos(\omega t_{n+1}) \\ \cos(\omega t_{n+1}) & -\frac{1}{\omega}\sin(\omega t_{n+1}) \end{bmatrix} \begin{bmatrix} \delta_{n+1} \\ \dot{\delta}_{n+1} \end{bmatrix}$$

Finally, the Fourier coefficients are updated from the formulas: $u_{n+1} = u_n + \Delta^u_{n+1}$ and $v_{n+1} = v_n + \Delta^v_{n+1}$.

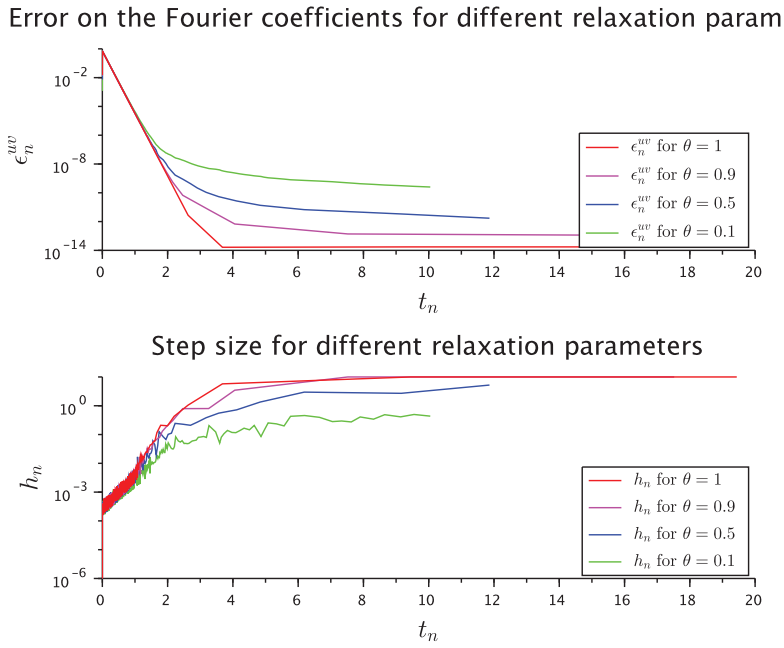

Figure 5.8: Results obtained with the estimator based on a trigonometric representation of the correction term with injection of perfect data. Top: Fourier coefficients error (on the abscissa: time in seconds, on the ordinate: Fourier coefficients error in arbitrary units), down: step size (on the abscissa: time in seconds, on the ordinate: step size in seconds).

As previously mentioned, when perfect data is injected in input, i.e. $\delta^{-(j)}_{n+1} = X^{(j)}_{th} - \bar{X}^{(j)}_n(t_{n+1})$, we can see in the upper figure of 5.8 that the Fourier coefficients converge as wanted in steady-state. In particular, the estimation error can be upper-bounded by the neglected term envelope, associated to the Fourier coefficients time-variation:

$$\epsilon^{u,v}_{n+1} = \sqrt{(\epsilon^u_{n+1})^2 + (\epsilon^v_{n+1})^2} \leq \frac{\sqrt{(\dot{u}_{th,n+1})^2 + (\dot{v}_{th,n+1})^2}}{\omega} \equiv ||\dot{u}_{n+1}, \dot{v}_{n+1}|| \tag{5.46}$$

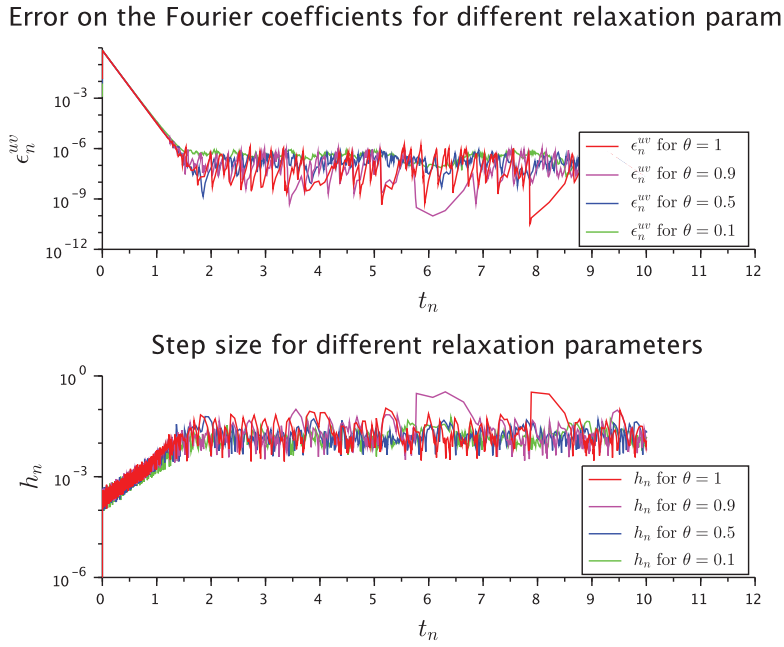

Figure 5.9: Results obtained with the estimator based on a trigonometric representation of the correction term without injection of perfect data. Top: Fourier coefficients error (on the abscissa: time in seconds, on the ordinate: Fourier coefficients error in arbitrary units), down: step size (on the abscissa: time in seconds, on the ordinate: step size in seconds).

So, the error tends toward zero when the system returns in steady-state since $\dot{u}_{th,n+1}$ and $\dot{v}_{th,n+1}$ tend toward zero. Unfortunately, when the numerical data coming from the numerical integration is used for the estimation, its performances are completely reduced: the estimator does not converge and the time step remains at a very low value (close to one period). This can be observed in the lower figure in 5.9. This estimator seems particularly unstable. Indeed, as for the initial estimator, the numerical errors due to the integration step on δ are directly re-injected into the estimator. That's why a fixed relaxation strategy has also been tried for this estimator by setting:

$$(u_{n+1}, v_{n+1}) = (u_n, v_n) + \theta(\Delta^u_{n+1}, \Delta^v_{n+1}) \tag{5.47}$$

But, as for the initial estimator, our results suggest that it does not solve the instability issue. However, this estimator gave us the idea of taking into account the sinusoidal behavior of the solution in a more analytic way during the estimation step.

5.3.3 Acceptation-rejection criterion

As the Fourier coefficients updates computed by the previously presented estimators introduced numerical instabilities, our idea has been to filter the estimated Fourier coefficients at each time step. Therefore, we implemented an acceptation-rejection criterion in order to choose if the Fourier coefficients should be updated from the last estimation or if we should keep their previous value. The underlying idea is that, if the Fourier coefficients are exact when the system is in steady-state, then $X_{th}(t) = \bar{X}_{th,\infty}(t) \equiv \bar{X}_{th}(t)$, thus $\bar{X}_{th}$ is the solution of the differential equation, i.e.

$$\dot{\bar{X}}_{th}(t) = f(t, \bar{X}_{th}(t)) \tag{5.48}$$

Which enables to define the following residual function and, as a consequence, computing more accurate Fourier coefficients should enable to reduce its magnitude:

$$\bar{\rho}(t) = \frac{\dot{\bar{X}}_{th}(t) - f(t, \bar{X}_{th}(t))}{\omega} \approx 0 \tag{5.49}$$

Indeed, as ρ is possibly a oscillating function, the objective is to reduce its magnitude and not only its instantaneous value. In the other case, if we only considered its instantaneous values, the zero-crossing due to the oscillations would completely suppress the validity of such a measure. Furthermore, the division by ω is done to have a per unit quantity.

Then, the idea is to compute the RMS measurement of $\bar{\rho}(t)$ for the different tested Fourier coefficients and to use it as a norm in our acceptation-rejection method:

$$R_{n+1}(u_{n+1}, v_{n+1}) = \sqrt{\frac{1}{T}\int_{t_{n+1}}^{t_{n+1}+T} \bar{\rho}_{n+1}^2(t)dt} \quad \text{with} \quad \bar{\rho}_{n+1}(t) = \frac{\dot{\bar{X}}_{n+1}(t) - f(t, \bar{X}_{n+1}(t))}{\omega} \tag{5.50}$$

In this equation, for some practical reasons and for ensuring the generality of our approach, the integral present in R_{n+1} is computed with a numerical quadrature rule. We chose the trapezoidal formula (where K is the sample size), so

$$R_{n+1}(u_{n+1}, v_{n+1}) \approx \sqrt{\frac{1}{T}\sum_{j=1}^{K} \frac{T}{K} \frac{\bar{\rho}_{n+1}^2\left(t_{n+1} + (j-1)\frac{T}{K}\right) + \bar{\rho}_{n+1}^2\left(t_{n+1} + j\frac{T}{K}\right)}{2}}$$

Finally, the chosen Fourier coefficients are those reducing R_{n+1}. Moreover, on a scalar linear case, we can easily prove that the minimum of R_{n+1} is reached for $(u_{n+1}, v_{n+1}) = (u_{th}, v_{th})$.

To recap, the acceptation-rejection criterion is used from the following procedure:

1. The past Fourier coefficients (u_n, v_n) are known.
2. The estimation is performed to compute (u_{n+1}, v_{n+1}).

3. R_{n+1} is computed for these two sets of Fourier coefficients, i.e.

$$R_{n+1}(u_{n+1}, v_{n+1}) = \sqrt{\frac{1}{T}\int_{t_{n+1}}^{t_{n+1}+T} \bar{\rho}_{n+1}^2(t)dt} \quad \text{with} \quad \bar{\rho}_{n+1}(t) = \frac{\dot{\bar{X}}_{n+1}(t) - f(t, \bar{X}_{n+1}(t))}{\omega} \tag{5.51}$$

$$R_{n+1}(u_n, v_n) = \sqrt{\frac{1}{T}\int_{t_{n+1}}^{t_{n+1}+T} \bar{\rho}_{n}^2(t)dt} \quad \text{with} \quad \bar{\rho}_{n}(t) = \frac{\dot{\bar{X}}_{n}(t) - f(t, \bar{X}_{n}(t))}{\omega} \tag{5.52}$$

4. Finally, $R_{n+1}(u_{n+1}, v_{n+1})$ and $R_{n+1}(u_n, v_n)$ are compared to select the Fourier coefficients:
   - If $R_{n+1}(u_{n+1}, v_{n+1}) \leq R_{n+1}(u_n, v_n)$, the lastly estimated Fourier coefficients are accepted.
   - Else, if $R_{n+1}(u_{n+1}, v_{n+1}) > R_{n+1}(u_n, v_n)$, the lastly estimated Fourier coefficients are rejected. Then, $(u_{n+1}, v_{n+1}) = (u_n, v_n)$.

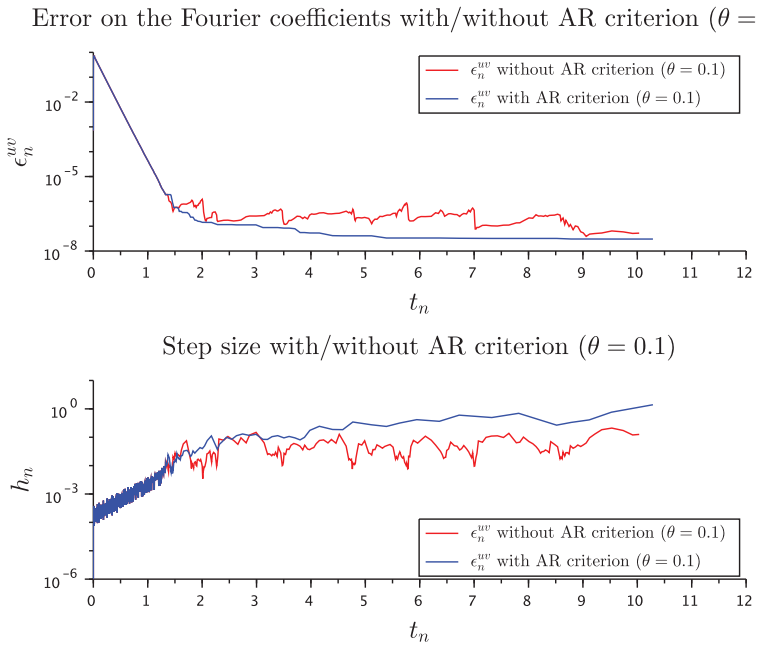

Figure 5.10: Results obtained with (blue) and without (red) the Fourier coefficients acceptation-rejection criterion when the estimation method is the initial estimator with a constant relaxation parameter $\theta = 0.1$. Top: Fourier coefficients error (on the abscissa: time in seconds, on the ordinate: Fourier coefficients error in arbitrary units), down: step size (on the abscissa: time in seconds, on the ordinate: step size in seconds).

In figures 5.10 and 5.11, we can see that the acceptation-rejection criterion drastically increases the tested estimators performances. Indeed, even for those based on the trigonometric representation of the correction which is particularly unstable, the Fourier coefficients convergence is relatively fast and so it enables to use large time step. However,

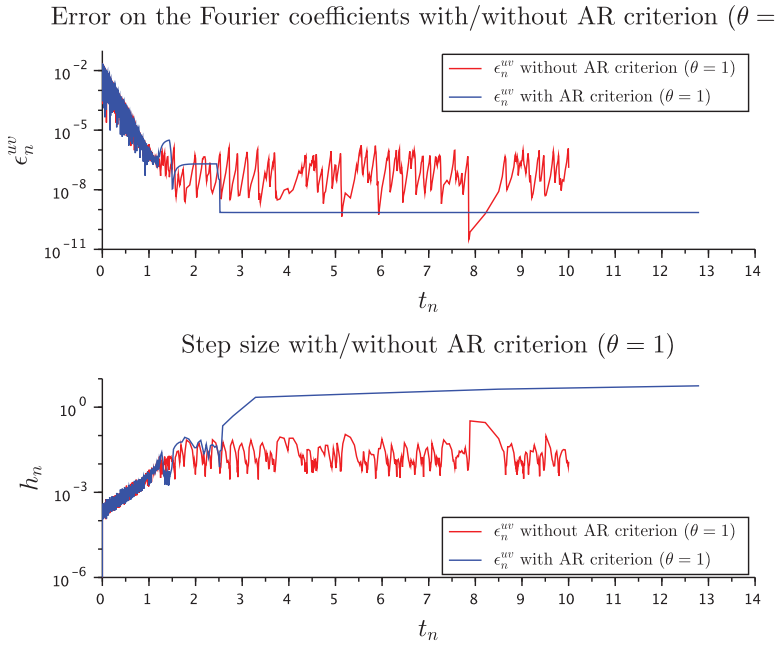

Figure 5.11: Results obtained with (blue) and without (red) the Fourier coefficients acceptation-rejection criterion when the estimation method is the estimator based on the trigonometric representation of the correction term (without relaxation). Top: Fourier coefficients error (on the abscissa: time in seconds, on the ordinate: Fourier coefficients error in arbitrary units), down: step size (on the abscissa: time in seconds, on the ordinate: step size in seconds).

even if this acceptation-rejection criterion seems attractive, using it in this way would not be suited for an industrial implementation. Indeed, its application to a vector case would require to test each combination of old and new Fourier coefficients, leading to an unsustainable computational cost because of the exponentially increasing number of combinations to test. That said, since this methodology is efficient with two radically different estimators and for different values of the relaxation parameter θ, it seems to indicate a good direction for computing the Fourier coefficients. This assessment led us to implement the final estimator.

5.4 Final estimator

In the previous section, we presented several heuristics that were implemented to stabilize the initial estimator, derived from the classical Fourier coefficients formula. On the contrary, as mentioned in the SPM presentation, the objective of the method final version was actually to avoid such kinds of hazardous heuristics. The different tested

approaches presented limitations for industrial applications as they generally required numerous assumptions especially on the input data to be efficient. However, in spite of its limitation to scalar cases, the last presented acceptation-rejection criterion offers a real improvement of our estimators convergence. In particular, this convergence seems very regular, which is the real problem to address. Then, we deduced that this criterion gave the good "direction" for the Fourier coefficients estimator to converge. Hence, instead of using it for filtering the Fourier coefficients estimated by a chosen estimator, the idea has been to directly compute the Fourier coefficients from this acceptation-rejection criterion. In other words, as the minimum of the objective-function corresponding to this criterion seems to be reached for the exact Fourier coefficients, they could be computed by finding the root of its gradient.

Therefore, the idea behind the estimation process is to compute the Fourier coefficients (u_{n+1}, v_{n+1}) that minimize an energy-function measuring the simulated system stationarity defined by:

$$R(u,v) = \int_{t_{n+1}}^{t_{n+1}+T} ||\bar{\rho}(t)||^2 dt \tag{5.53}$$

with $||X||^2 = X^T X$ and

$$\bar{\rho}(t) = F\left(t, \begin{bmatrix} \bar{X}_s(t) \\ X_{ns,n+1} \end{bmatrix}, \begin{bmatrix} \dot{\bar{X}}_s(t) \\ 0 \end{bmatrix}\right) \tag{5.54}$$

where

$$\bar{X}_s(t) = u\sin(\omega t) + v\cos(\omega t) \tag{5.55}$$

$$\dot{\bar{X}}_s(t) = \omega(u\cos(\omega t) - v\sin(\omega t)) \tag{5.56}$$

so that we can define the residual associated to the computed Fourier coefficients (u_{n+1}, v_{n+1}), i.e. to the periodic function $\bar{X}_{s,n+1}$, as below:

$$\bar{\rho}_{n+1}(t) = F\left(t, \begin{bmatrix} \bar{X}_{s,n+1}(t) \\ X_{ns,n+1} \end{bmatrix}, \begin{bmatrix} \dot{\bar{X}}_{s,n+1}(t) \\ 0 \end{bmatrix}\right) \tag{5.57}$$

The objective-function (5.53) is derived from (2.4)-(2.5)-(2.6)-(2.7). Indeed, the motivation here is to compute the Fourier coefficients of the oscillating components corresponding to the system in steady-state. Thus, it assumes that, for $t \geq t_{n+1}$

$$X_{ns}(t) \approx X_{ns,n+1} \tag{5.58}$$

where $X_{ns,n+1}$ is the approximation of the non-oscillating components X_{ns} at time t_{n+1} and so, as they are assumed to be constant

$$\dot{X}_{ns}(t) \approx 0 \tag{5.59}$$

In order to minimize the objective-function (5.53), numerous strategies could be implemented. Here, we present a first approach which consists in solving the associated linearized optimization problem. So, first of all, the new Fourier coefficients are decomposed as the sum of the previous one (u_n, v_n) and an increment $(\Delta^u_{n+1}, \Delta^v_{n+1})$, i.e

$$u_{n+1} = u_n + \Delta^u_{n+1} \tag{5.60}$$

$$v_{n+1} = v_n + \Delta^v_{n+1} \tag{5.61}$$

Let us define $\bar{X}_{s,\Delta^{uv}_{n+1}}$ the oscillating function corresponding to this increment such as $\bar{X}_{s,n+1}(t) = \bar{X}_{s,n}(t) + \bar{X}_{s,\Delta^{uv}_{n+1}}(t)$, i.e. $\bar{X}_{s,\Delta^{uv}_{n+1}} = \Delta^u_{n+1}\sin(\omega t) + \Delta^v_{n+1}\cos(\omega t)$. Then, using this decomposition, $\bar{\rho}_{n+1}(t)$ can be linearized around the predicted a priori steady-state trajectory which is associated to the past periodic function for the solution oscillating components, i.e. $\bar{X}_{s,n}(t)$, and the last computed value for the solution non-oscillating components, i.e. $X_{ns,n+1}$. This linearization leads to:

$$\bar{\rho}_{n+1}(t) \approx \left[\sin(\omega t)\partial_{X_s}\bar{\rho}_n(t) + \omega\cos(\omega t)\partial_{\dot{X}_s}\bar{\rho}_n(t)\right]\Delta^u_{n+1} \tag{5.62}$$

$$+ \left[\cos(\omega t)\partial_{X_s}\bar{\rho}_n(t) - \omega\sin(\omega t)\partial_{\dot{X}_s}\bar{\rho}_n(t)\right]\Delta^v_{n+1} \tag{5.63}$$

$$+ \bar{\rho}_n(t) \tag{5.64}$$

where

$$\bar{\rho}_n(t) = F\left(t, \begin{bmatrix}\bar{X}_{s,n}(t)\\ X_{ns,n+1}\end{bmatrix}, \begin{bmatrix}\dot{\bar{X}}_{s,n}(t)\\ 0\end{bmatrix}\right) \tag{5.65}$$

Finally, after the injection of this formula into (5.53) and the computation of the resulting objective-function gradient, Δ^u_{n+1} and Δ^v_{n+1} are simply obtained by solving the linear system (5.66) which corresponds to Euler's equation ($\nabla R(u_{n+1}, v_{n+1}) = 0$) applied to the linearized problem:

$$\begin{bmatrix}H^{uu}_n & H^{uv}_n\\ H^{vu}_n & H^{vv}_n\end{bmatrix}\begin{bmatrix}\Delta^u_{n+1}\\ \Delta^v_{n+1}\end{bmatrix} = -\begin{bmatrix}g^u_n\\ g^v_n\end{bmatrix} \tag{5.66}$$

The linear system (5.66) coefficients are then given by

$$
\begin{aligned}
H_n^{uu} &= \int_{t_{n+1}}^{t_{n+1}+T} \Big[\sin(\omega t)\partial_{X_s}\bar{\rho}_n(t) + \omega\cos(\omega t)\partial_{\dot{X}_s}\bar{\rho}_n(t)\Big]^T \\
&\qquad \Big[\sin(\omega t)\partial_{X_s}\bar{\rho}_n(t) + \omega\cos(\omega t)\partial_{\dot{X}_s}\bar{\rho}_n(t)\Big]\, dt \\
H_n^{uv} &= \int_{t_{n+1}}^{t_{n+1}+T} \Big[\sin(\omega t)\partial_{X_s}\bar{\rho}_n(t) + \omega\cos(\omega t)\partial_{\dot{X}_s}\bar{\rho}_n(t)\Big]^T \\
&\qquad \Big[\cos(\omega t)\partial_{X_s}\bar{\rho}_n(t) - \omega\sin(\omega t)\partial_{\dot{X}_s}\bar{\rho}_n(t)\Big]\, dt \\
H_n^{vu} &= \int_{t_{n+1}}^{t_{n+1}+T} \Big[\cos(\omega t)\partial_{X_s}\bar{\rho}_n(t) - \omega\sin(\omega t)\partial_{\dot{X}_s}\bar{\rho}_n(t)\Big]^T \\
&\qquad \Big[\sin(\omega t)\partial_{X_s}\bar{\rho}_n(t) + \omega\cos(\omega t)\partial_{\dot{X}_s}\bar{\rho}_n(t)\Big]\, dt \\
H_n^{vv} &= \int_{t_{n+1}}^{t_{n+1}+T} \Big[\cos(\omega t)\partial_{X_s}\bar{\rho}_n(t) - \omega\sin(\omega t)\partial_{\dot{X}_s}\bar{\rho}_n(t)\Big]^T \\
&\qquad \Big[\cos(\omega t)\partial_{X_s}\bar{\rho}_n(t) - \omega\sin(\omega t)\partial_{\dot{X}_s}\bar{\rho}_n(t)\Big]\, dt
\end{aligned}
$$

$$
\begin{aligned}
g_n^u &= \int_{t_{n+1}}^{t_{n+1}+T} \Big[\sin(\omega t)\partial_{X_s}\bar{\rho}_n(t) + \omega\cos(\omega t)\partial_{\dot{X}_s}\bar{\rho}_n(t)\Big]^T \bar{\rho}_n(t)dt \\
g_n^v &= \int_{t_{n+1}}^{t_{n+1}+T} \Big[\cos(\omega t)\partial_{X_s}\bar{\rho}_n(t) - \omega\sin(\omega t)\partial_{\dot{X}_s}\bar{\rho}_n(t)\Big]^T \bar{\rho}_n(t)dt
\end{aligned}
$$

Thus, we have to compute several integrated quantities as $I_{n+1} = \int_{t_{n+1}}^{t_{n+1}+T} \phi(t)dt$. In our current implementation, they are computed from the trapezoidal quadrature rule, i.e.

$$
I_{n+1} = \sum_{j=0}^{K-1} \frac{T}{K} \frac{\phi(t_{n+1} + j\frac{T}{K}) + \phi(t_{n+1} + (j+1)\frac{T}{K})}{2} \tag{5.67}
$$

where K is the sample size, whose choice is arbitrary. Finally, a drawback of this estimator is its possible computational cost since it requires to perform several linear operations such as:

1. Matrix restrictions for reducing the partial-derivatives of the DAE residual function to the oscillating components, which is done by selecting the corresponding columns: $\partial_X\bar{\rho}_n(t) \to \partial_{X_s}\bar{\rho}_n(t)$ and $\partial_{\dot{X}}\bar{\rho}_n(t) \to \partial_{\dot{X}_s}\bar{\rho}_n(t)$;
2. Scaling by $\sin(\omega t)$, $\cos(\omega t)$, $\omega\sin(\omega t)$ and $\omega\cos(\omega t)$.
3. Matrix sums between the scaled matrices based on $\partial_{X_s}\bar{\rho}_n(t)$ and $\partial_{\dot{X}_s}\bar{\rho}_n(t)$ in the integral whose integration interval is $[t_{n+1}, t_{n+1}+T]$.
4. Transposition of the resulting matrices for the matrix-matrix and matrix-vector products.
5. Matrix-matrix products between the transposed resulting matrices and the non-transposed resulting matrices in the integral;

6. Matrix-vector products between the transposed resulting matrices and the evaluation of the DAE residual function in the integral;
7. Matrix sums between the final matrices at the different sample times of the integration interval $[t_{n+1}, t_{n+1} + T]$ due to the trapezoidal formula estimation. However, even if K affects the quadrature rule accuracy, a few number of samples is required, e.g. $K = 5$. Finding a good compromise between the precision and computational cost is determinant for the entire method performances. In particular, using a too large number of sample points could lead to drastically increase the estimator computational cost. Our strategy is to exploit the incremental convergence of the estimator and thus our objective is to limit the individual cost of each estimation. Indeed, as the main performances limitation is due to the system deviation to steady-state conditions, transient phases (even soft transients) lead to an important number of iterations and so of estimations wherein the Fourier coefficients may be biased since the objective function is based on a measure of the system stationarity.

In order to optimize the SPM performances, a possible enhancement could be to compare the system stationarity measurement using the past Fourier coefficients, i.e. $R(u_n, v_n)$, with a threshold to determine. By this way, they could be eventually reused in order to avoid the estimator linear system construction and resolution which is computationally expensive.

In figure 5.12, the results obtained with final estimator are presented. We can see that it leads to a very fast and particularly regular convergence of the Fourier coefficients. When the system is in steady-state, the estimation error tends toward zero and so very large time steps can be used. Using this estimator, the resolution seems really stable as there is no oscillation of the step size and notably when it reaches very high values.

This approach could be generalized by not linearizing (5.57) and using a more general optimization procedure to compute (u_{n+1}, v_{n+1}). The current estimator actually corresponds to the application of a modified Newton's algorithm iteration for solving the optimization problem associated to the objective function (5.53) using the previous Fourier coefficients as initial guess. Indeed Newton's algorithm m-th iteration for solving Euler's equation is defined as:

$$\begin{bmatrix} \Delta^u_{n+1(m)} \\ \Delta^v_{n+1(m)} \end{bmatrix} = - \left[\nabla^2 R(u_{n+1(m)}, v_{n+1(m)}) \right]^{-1} \nabla R(u_{n+1(m)}, v_{n+1(m)}) \tag{5.68}$$

$$(u_{n+1(m+1)}, v_{n+1(m+1)}) = (u_{n+1(m)}, v_{n+1(m)}) + (\Delta^u_{n+1(m)}, \Delta^v_{n+1(m)}) \tag{5.69}$$

where $R(u_{n+1(m)}, v_{n+1(m)})$ is computed by injecting $(u_{n+1(m)}, v_{n+1(m)})$ into the residual,

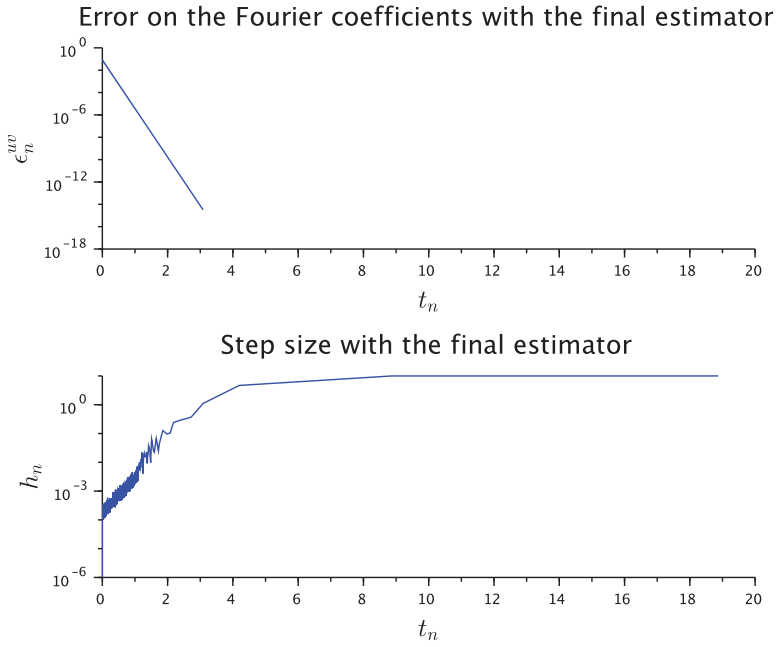

Figure 5.12: Performances obtained with the final estimator. Top: Fourier coefficients error (on the abscissa: time in seconds, on the ordinate: Fourier coefficients error in arbitrary units), down: step size (on the abscissa: time in seconds, on the ordinate: step size in seconds). When the system is in steady-state, the Fourier coefficients convergence is really fast and regular which enables to reach high step sizes.

i.e. $\bar{\rho}_{n+1(m)}$, which is then defined from (5.57) as:

$$\bar{\rho}_{n+1(m)}(t) = F\left(t, \begin{bmatrix} \bar{X}_{s,n+1(m)}(t) \\ X_{ns,n+1} \end{bmatrix}, \begin{bmatrix} \dot{\bar{X}}_{s,n+1(m)}(t) \\ 0 \end{bmatrix}\right) \tag{5.70}$$

with $\bar{X}_{s,n+1(m)}(t) = u_{n+1(m)} \sin(\omega t) + v_{n+1(m)} \cos(\omega t)$. Thus our implemented estimator finally consists in setting $(u_{n+1(0)}, v_{n+1(0)}) = (u_n, v_n)$ and applying a Newton's algorithm iteration using the exact gradient value i.e.

$$\nabla R(u_{n+1(m)}, v_{n+1(m)}) = 2 \int_{t_{n+1}}^{t_{n+1}+T} \nabla \bar{\rho}_{n+1(m)}(t)^T \bar{\rho}_{n+1(m)}(t) dt \tag{5.71}$$

and approximating the Hessian matrix by neglecting the residual second-order partial-derivatives terms, i.e.

$$\nabla^2 R(u_{n+1(m)}, v_{n+1(m)}) \approx 2 \int_{t_{n+1}}^{t_{n+1}+T} \nabla \bar{\rho}_{n+1(m)}(t)^T \nabla \bar{\rho}_{n+1(m)}(t) dt \tag{5.72}$$

where

$$\nabla \bar{\rho}_{n+1(m)}(t) = \nabla F\left(t, \begin{bmatrix} \bar{X}_{s,n+1(m)}(t) \\ X_{ns,n+1} \end{bmatrix}, \begin{bmatrix} \dot{\bar{X}}_{s,n+1(m)}(t) \\ 0 \end{bmatrix}\right) \tag{5.73}$$

with

$$\begin{aligned}
\nabla_u \bar{\rho}_{n+1(m)}(t) &= \sin(\omega t)\partial_{X_s} F\left(t, \begin{bmatrix} \bar{X}_{s,n+1(m)}(t) \\ X_{ns,n+1} \end{bmatrix}, \begin{bmatrix} \dot{\bar{X}}_{s,n+1(m)}(t) \\ 0 \end{bmatrix}\right) \\
&\quad + \omega\cos(\omega t)\partial_{\dot{X}_s} F\left(t, \begin{bmatrix} \bar{X}_{s,n+1(m)}(t) \\ X_{ns,n+1} \end{bmatrix}, \begin{bmatrix} \dot{\bar{X}}_{s,n+1(m)}(t) \\ 0 \end{bmatrix}\right) \\
\nabla_v \bar{\rho}_{n+1(m)}(t) &= \cos(\omega t)\partial_{X_s} F\left(t, \begin{bmatrix} \bar{X}_{s,n+1(m)}(t) \\ X_{ns,n+1} \end{bmatrix}, \begin{bmatrix} \dot{\bar{X}}_{s,n+1(m)}(t) \\ 0 \end{bmatrix}\right) \\
&\quad - \omega\sin(\omega t)\partial_{\dot{X}_s} F\left(t, \begin{bmatrix} \bar{X}_{s,n+1(m)}(t) \\ X_{ns,n+1} \end{bmatrix}, \begin{bmatrix} \dot{\bar{X}}_{s,n+1(m)}(t) \\ 0 \end{bmatrix}\right)
\end{aligned}$$

However, the resulting estimator computational cost might be very high because of the estimator linear system construction and resolution which are computationally expensive.

5.5 Estimators comparison

In figure 5.13, the results obtained with different estimators are presented. We compare the different implemented estimators within their best configuration:

- the initial estimator using the fixed relaxation strategy ($\theta = 0.1$) and the acceptation-rejection criterion ;
- the estimator based on the trigonometric representation of the correction term using the acceptation-rejection criterion ;
- the final estimator.

For all these estimators, the Fourier coefficients converge but the final error significantly differ depending on the used estimator. For the initial estimator, the error remains at a non-negligible value which finally limits the maximum used step size. On the contrary, the final error with the estimator based on the trigonometric representation of the correction term is sufficiently low for using high step sizes. The best results are obtained with the final estimator, which leads to a very fast and regular convergence of the Fourier coefficients and so to the use of very large time steps.

In table 5.1, the results obtained within different tested configurations are summarized. Several indicators are then presented:

- N_{ite}: number of time steps;
- t_{CPU}: computational time;
- h_{mean} and h_{max}: respectively the mean and maximum used step size;
- $\epsilon^{uv}_{N_{ite}}$: Fourier coefficients error at the simulation final time;

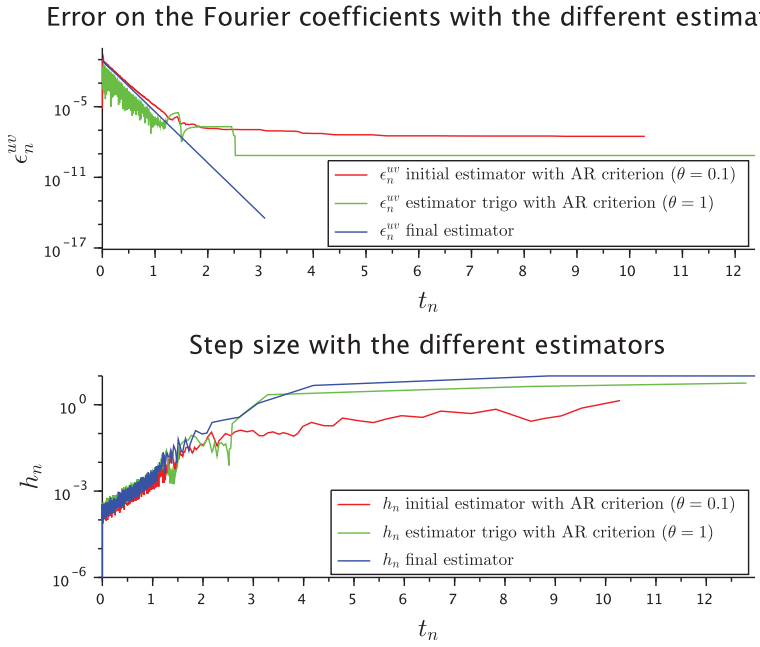

Figure 5.13: Performances comparison between the different tested estimators (red: initial estimator with the use of the acceptation-rejection criterion and the relaxation, green: estimator based on the trigonometric representation of the correction term with the use of the acceptation-rejection criterion but without the relaxation, blue: final estimator). Top: Fourier coefficients error (on the abscissa: time in seconds, on the ordinate: Fourier coefficients error in arbitrary units), down: step size (on the abscissa: time in seconds, on the ordinate: step size in seconds). When the system is in steady-state, the convergence of the final estimator is really fast and regular which enables to reach high step sizes.

- *stable*: indicates if the solver has a globally stable behavior in steady-state (no oscillation of the step size, etc).

These results prove that exploiting the system stationarity, with the acceptation-rejection criterion or more directly in the final estimator, is a very efficient strategy for getting good performances in terms of Fourier coefficients convergence and so of number of iterations. The highest step size is reached with the final estimator whose maximum used step size is equal to 10, which is the solver maximum step size set by the user. In addition, the final Fourier coefficients error with this estimator is equal to zero.

Estimator	N_{ite}	t_{CPU}	h_{mean}	h_{max}	$\epsilon^{uv}_{N_{ite}}$	stable
Initial / $\theta = 0.1$ / without AR	2908	81.84	3.49e-3	2.102e-1	5.26e-8	no
Initial / $\theta = 0.5$ / without AR	3480	93.26	2.88e-3	7.73e-2	6.38e-7	no
Initial / $\theta = 1.0$ / without AR	3681	98.88	2.72e-3	2.92e-2	8.45e-7	no
Initial / θ adj. / without AR	2811	76.13	3.61e-3	2.68e-1	9.75e-8	no
Trigo / $\theta = 0.1$ / without AR	2801	11.42	3.58e-3	5.98e-2	3.52e-7	no
Trigo / $\theta = 0.5$ / without AR	2518	10.05	3.98e-3	6.18e-2	2.73e-7855	no
Trigo / $\theta = 1.0$ / without AR	2382	9.63	4.20e-3	3.28e-1	1.26e-7	no
Initial / $\theta = 0.1$ / with AR	2757	94.92	4.24e-3	1.40	3.04e-9	yes
Initial / $\theta = 0.5$ / with AR	2584	86.30	4.49e-3	9.67e-1	2.22e-9	yes
Initial / $\theta = 1.0$ / with AR	2599	85.91	4.27e-3	8.92e-1	1.13e-8	yes
Initial / θ adj. / with AR	2769	91.84	3.67e-3	4.85e-1	4.07e-8	no
Trigo / $\theta = 0.1$ / with AR	2282	24.20	4.70e-3	7.90e-1	4.29e-9	no
Trigo / $\theta = 0.5$ / with AR	2049	21.41	7.22e-3	2.87	1.84e-9	yes
Trigo / $\theta = 1.0$ / with AR	1995	20.60	9.25e-3	5.65	7.16e-10	yes
Final	2373	26.75	1.22e-2	10	0 (machine)	yes

Table 5.1: Results obtained with the different implemented estimators and several configurations.

6

Theoretical analysis

Contents

In this chapter, the numerical properties of the Sinusoidal Predictor Method are investigated. In addition to classical numerical properties such as the consistency and the stability, a special focus is given on comparing its behavior when the system is in steady-state and during transients. Ideally, the step size adjustment strategy used in the SPM should enable to reduce it for catching fast transient phenomena and considerably increase it for optimizing the simulations computational cost in steady-state. To perform this study, we use the same linear ordinary equation with an oscillating forcing term

obtained from a manufactured solution as in the previous chapter about the estimator optimization. In this equation, the linear coefficient enables to control the problem stiffness. The underlying solution corresponds to a phasor-like transient as it consists in an envelope variation. Then, this theoretical study will focus on several important aspects: quantifying the global error of the method due to the integration part when the Fourier coefficients error is known, investigating the behavior of the Fourier coefficients estimator, with a particular attention to its convergence in steady-state and its error-bounding during transients, and, to finish, the step size adaptation during the simulation. In other words, our objective is to theoretically validate the SPM potential by verifying that it enables to use very large time steps in steady-state while being able to accurately catch transient phenomena.

6.1 Considered problem

In general, the used equation for the numerical schemes theoretical study is the Dalhquist equation $\dot{Y}(t) = \Lambda Y(t)$, where $\Lambda \in \mathbb{C}$. Then, to assess the stability properties of the considered numerical scheme, the common methodology [29] consists in injecting this equation into the numerical scheme in order to substitute the time-derivatives. By this way, the method characteristic polynomial is obtained and so the objective is to verify that the error amplification ratio modulus is lower than 1. It especially enables to determine if a method is A-stable, i.e. if the above-mentioned property is verified for all Λ whose real part is negative, and A-unstable, i.e. if it is not verified for all Λ whose real part is positive.

However, in the SPM context, this equation is not appropriate as oscillations are not structural but generally come from an oscillating forcing term. That's why, we use the same example as for the previous chapter on the estimator optimization, i.e. $\dot{X}(t) = \lambda X(t) + \bar{b}(t)$ where $\lambda \in \mathbb{R}$ and $\bar{b}(t)$ is an oscillating forcing term. In addition, this equation enables to study the SPM global behavior and especially investigating the evolution of its performances when the system comes back to steady-state after having faced a transient phase.

6.2 SPM expected properties

As mentioned in this chapter introduction, our final objective is to have a flexible method enabling to optimize the computational cost of time-domain simulations by using large time steps in steady-state while finely controlling the numerical error, especially during transients. In this section, some expected properties of the SPM are exposed. A spe-

cial focus is given on its particular behavior in steady-state and during transients. To summarize, the SPM is expected to have the properties mentioned in table 6.1 below.

	Transient	Steady-state
Estimator	$\epsilon_n^{uv} = \sqrt{(\epsilon_n^u)^2 + (\epsilon_n^v)^2}$ high but bounded	convergence: $\epsilon_n^{uv} \to 0$
Integrator	Error control: reduction of h_n for compensating ϵ_n^{uv}	$h_n \to h_{max}$
Performances	Limited	High

Table 6.1: Summary of the SPM expected properties

6.2.1 Equivalence between the problem rewritten on the correction term and the original problem

The idea of the SPM is to optimize time-domain simulations of AC power systems by enriching the solver. In contrast with most of the existing approaches, which focus on the modeler, the underlying main objective of our methodology is to finely control the computational error while making possible the use of larger integration step sizes. As computational errors are considered to only come from the solver in our approach, a special attention is given on its numerical properties. That's why the first aspect to verify is that the problem rewritten on the correction term δ is equivalent to the problem on the global solution X. Indeed, as previously mentioned, if a step size adjustment strategy is used but on a non-equivalent problem, controlling the computational error has no sense. This property is proven in the section 6.3.1.

6.2.2 Performances when the system is in steady-state

In steady-state, the solution oscillating components are expected to tend toward centered sinusoids with constant Fourier coefficients, i.e. if we consider a scalar problem

$$X_{th,s}(t) \to \bar{X}_\infty(t) = u_\infty \sin(\omega t) + v_\infty \cos(\omega t) \tag{6.1}$$

Then, the SPM should have the following behavior:

- The Fourier coefficients estimated by the SPM should tend toward their theoretical asymptotic values, i.e. $(u_n, v_n) \to (u_\infty, v_\infty)$, so that the SPM periodic part tends toward the theoretical sinusoid: $\bar{X}_n(t) \to \bar{X}_\infty(t) = X_{th,s}(t)$. This property is proven in the section 6.4.2.
- By this way, as $\bar{X}_\infty(t) = X_{th,s}(t)$, the correction term of the SPM decomposition should be damped, i.e. $\delta(t) \to 0$.
- Finally, as the dynamics of the correction term, which is the integrated variable,

should be completely damped, the step size should consequently increase, i.e. $h_n \rightarrow h_{max}$.

6.2.3 Control of the error during transients

During transients, the oscillating components of the solution may be significantly different from sinusoids with constant Fourier coefficients, i.e. $X_{th,s}(t) \neq \bar{X}_{s,\infty}(t)$. In particular, we can distinguish two cases:

1. If the system is in smooth transients, corresponding to phasor-like dynamics, the solution may be a sinusoid with time-varying Fourier coefficients. The SPM should be able to cope with this envelope variation. This property is proven in the section 6.4.3.
2. If the system is in stronger transients, corresponding to EMT-like dynamics, the solution may contain an offset, higher-order harmonics or even present a more dynamical behavior. As the solution might radically differ from a sinusoid, the SPM should probably be switched off in order to avoid unnecessary additional cost due to SPM-specific mathematical operations such as the estimation step, etc.

In addition, the step size used for integrating the correction term should ideally be decreased so that the global error on the complete solution remains lower than the chosen tolerance. By this way, the step size adjustment strategy can enable to compensate the erroneous periodic part. This point is discussed in the section 6.3.4.

6.3 Integration error analysis

The role of the SPM correction term integration can be seen as compensating potential errors coming from the periodic part. Therefore, we firstly show that the problem rewritten on the correction term is well equivalent to the original one on the global solution, contrary to most of the existing approaches. This property is the basis for theoretically validating the error control within our method. In a second time, our study focuses on the numerical error due to the integration step at the local level first and then at the global level. Finally, we propose a methodology for investigating in more details the step size adjustment during the simulation.

6.3.1 Equivalence between the problem on the correction term and those on the global solution

Proposition 1. *At each time step of the SPM, i.e. for each time interval $[t_n, t_{n+1}]$, the local problem rewritten on the correction term δ is equivalent to the original system of differential equations on the global solution X at the continuous level.*

Proof. Let us consider an ordinary differential equation whose form is $\dot{X}(t) = f(t, X(t))$. We recall that, by injecting the local values of the Fourier coefficients (u_n, v_n) for the time interval $[t_n, t_{n+1}]$, the local equation on the correction term is given by

$$\dot{\delta}(t) = \Phi_n(t, \delta(t)) = f(t, \delta(t) + \bar{X}(t)) - \dot{\bar{X}}_n(t) \tag{6.2}$$

Therefore, the local problem on the correction term δ and the global problem on the complete solution are equivalent by construction. Indeed, if we consider the Picard integral form of the above-mentioned equation, we have

$$\begin{aligned}
\delta(t_{n+1}) + \bar{X}_n(t_{n+1}) &= \delta(t_n) + \int_{t_n}^{t_{n+1}} \Phi_n(t, \delta(t))dt + \bar{X}_n(t_{n+1}) \\
&= \delta(t_n) + \int_{t_n}^{t_{n+1}} f(t, \delta(t) + \bar{X}_n(t)) - \dot{\bar{X}}_n(t)dt + \bar{X}_n(t_{n+1}) \\
&= \delta(t_n) + \int_{t_n}^{t_{n+1}} \dot{X}(t) - \dot{\bar{X}}_n(t)dt + \bar{X}_n(t_{n+1}) \\
&= \delta(t_n) + X(t_{n+1}) - X(t_n) - \bar{X}_n(t_{n+1}) + \bar{X}_n(t_n) + \bar{X}_n(t_{n+1}) \\
&= X(t_{n+1})
\end{aligned}$$

□

6.3.2 Local truncation error and consistency

In this section, we compare the numerical solutions coming from a classical solver and from the SPM. To do this, we inject the global solution theoretical values as input data in order to compute their local truncation error, i.e. $X_n^{(j)} = X_{th}^{(j)}(t_n)$. Indeed, even if the local problem rewritten on the correction term is equivalent to the original one on the global solution at the continuous level, the numerical solution computed by the SPM and so the resulting local truncation error slightly differ from those of a classical scheme. In particular, the SPM introduces a specific residual term which is directly due to the SPM decomposition and is associated to the periodic function.

Proposition 2. *The SPM with the trapezoidal formula as basic integration scheme is second-order consistent as its local truncation error is given by*

$$e_{n+1} = -\frac{h_n^3}{12}\delta^{(3)}|_{(u_n,v_n)}(t_n) + \mathcal{O}(h_n^4) \tag{6.3}$$

where $\delta^{(3)}|_{(u_n,v_n)}(t_n) = \frac{h_n^3}{12}(X_{th}^{(3)}(t_n) - \bar{X}_n^{(3)}(t_n))$ *is the third time-derivative at time* t_n *of the correction term associated to the used local values of the Fourier coefficients* (u_n, v_n).

Proof. If we apply the Trapezoidal formula to the local differential equation on the correction term (6.2), we obtain:

$$\delta_{n+1} = \delta_n + \frac{h_n}{2}(\Phi_n(t_n, \delta_n) + \Phi_n(t_{n+1}, \delta_{n+1})) \tag{6.4}$$

where

$$\delta_{n+1} = X_{n+1} - \bar{X}_n(t_{n+1}) \tag{6.5}$$

$$\delta_n = X_n - \bar{X}_n(t_n) \tag{6.6}$$

$$\Phi_n(t_n, \delta_n) = f(t_n, X_n) - \dot{\bar{X}}_n(t_n) \tag{6.7}$$

$$\Phi_n(t_{n+1}, \delta_{n+1}) = f(t_{n+1}, X_{n+1}) - \dot{\bar{X}}_n(t_{n+1}) \tag{6.8}$$

Inserting these values into the equation (6.4) leads to the following global solution formula:

$$X_{n+1} = \underbrace{X_n + \frac{h_n}{2}(f(t_n, X_n) + f(t_{n+1}, X_{n+1}))}_{=X_{n+1}^{TR}} + \underbrace{\bar{X}_n(t_{n+1}) - \bar{X}_n(t_n) - \frac{h_n}{2}(\dot{\bar{X}}_n(t_n) + \dot{\bar{X}}_n(t_{n+1}))}_{=e_{n+1}^{SPM}} \tag{6.9}$$

Where one can identify the solution obtained by applying the classical trapezoidal formula to the original differential equation to compute the global solution at the time t_{n+1} :

$$X_{n+1}^{TR} = X_n + \frac{h_n}{2}(f(t_n, X_n) + f(t_{n+1}, X_{n+1})) \tag{6.10}$$

which leads to the following local truncation error if $X_n^{(j)} = X_{th}^{(j)}(t_n)$:

$$e_{n+1}^{TR} = -\frac{h_n^3}{12}X_n^{(3)} + \mathcal{O}(h_n^4) \tag{6.11}$$

In addition, we can note the presence of a residual term e_{n+1}^{SPM} due to the numerical integration scheme (in this example, the Trapezoidal Formula) which is implicitly applied to the solution periodic part, thus

$$e_{n+1}^{SPM} = -\frac{h_n^3}{12}\bar{X}_n^{(3)}(t_n) + \mathcal{O}(h_n^4) = \frac{\omega^3 h_n^3}{12}(u_n \cos(\omega t_n) - v_n \sin(\omega t_n)) + \mathcal{O}(h_n^4) \tag{6.12}$$

In other words, by injecting identical input data, the SPM computes the same solution as the classical trapezoidal scheme with the add of an $\mathcal{O}(h_n^3)$ term. Thus, the second order consistency of the SPM is locally ensured:

$$\begin{aligned} e_{n+1} &= X_{th}(t_{n+1}) - (X_{n+1}^{TR} + e_{n+1}^{SPM}) \\ &= -\frac{h_n^3}{12} X_{th}^{(3)}(t_n) + \frac{h_n^3}{12} \bar{X}_n^{(3)}(t_n) + \mathcal{O}(h_n^4) \\ &= -\frac{h_n^3}{12} (X_{th}^{(3)}(t_n) - \bar{X}_n^{(3)}(t_n)) + \mathcal{O}(h_n^4) \\ &= -\frac{h_n^3}{12} \delta^{(3)}|_{(u_n,v_n)}(t_n) + \mathcal{O}(h_n^4) \end{aligned}$$

□

In this equation, it is important to note that $\delta^{(3)}|_{(u_n,v_n)}(t_n)$ may differ from the previously defined theoretical local correction term $\delta_{th,n}(t)$ which is associated to the theoretical Fourier coefficients $(u_{th,n}, v_{th,n})$.

6.3.3 Derivation of the global error formula

In the previous section, the impact of using the SPM has been highlighted at the local level. Indeed, we have seen that integrating the differential equation with the SPM leads to the introduction of a specific residual term that is associated to the used periodic function. Hence, in a second time, our objective is to quantify the global error of the SPM and particularly when, at each time step, an error is introduced on the Fourier coefficients (u_n, v_n) i.e.

$$(u_n, v_n) = (u_{th,n}, v_{th,n}) + (\epsilon_n^u, \epsilon_n^v) \tag{6.13}$$

By representing the Fourier coefficients in such a way, the objective is to model the injection of Fourier coefficients estimation errors into the integrator and especially to highlight its contribution.

Proposition 3. *For the equation* (5.5)*, the global error of the SPM with the trapezoidal formula as basic integration scheme can be described by the following recurrence formula:*

$$\epsilon_{n+1} = \eta_n \epsilon_n + \mu_n \left[LTE_n^{\delta_{th,n}} - LTE_n^{\bar{X}_{\epsilon_n^{uv}}} \right] \tag{6.14}$$

where

$$\eta_n = \frac{1 + \frac{h_n\lambda}{2}}{1 - \frac{h_n\lambda}{2}} \tag{6.15}$$

$$\mu_n = \frac{1}{1 - \frac{h_n\lambda}{2}} \tag{6.16}$$

$$LTE_n^{\delta_{th,n}} = \frac{h_n^3}{12}\delta_{th,n}^{(3)}(t_n) + \mathcal{O}(h_n^4) \tag{6.17}$$

$$LTE_n^{\bar{X}_{\epsilon_n^{uv}}} = \frac{h_n^3}{12}\bar{X}_{\epsilon_n^{uv}}^{(3)}(t_n) + \mathcal{O}(h_n^4) \tag{6.18}$$

Thus, in this particular case, as $\delta_{th,n}$ tends toward zero, the SPM is stable if the periodic function associated to the Fourier coefficients error $\bar{X}_{\epsilon_n^{uv}}$ is regularly damped and independent from the global error ϵ_n.

Proof. The periodic solution can be decomposed into two parts:

$$\bar{X}_n(t) = \bar{X}_{th,n}(t) + \bar{X}_{\epsilon_n^{uv}}(t) \quad \text{with} \quad \begin{cases} \bar{X}_{th,n}(t) & = u_{th,n}\sin(\omega t) + v_{th,n}\cos(\omega t) \\ \bar{X}_{\epsilon_n^{uv}}(t) & = \epsilon_n^u \sin(\omega t) + \epsilon_n^v \cos(\omega t) \end{cases} \tag{6.19}$$

First, we define the local theoretical correction which is based on the piecewise-constant Fourier coefficients model: $\delta_{th,n}(t) = X_{th}(t) - \bar{X}_{th,n}(t) = (u_{th}(t) - u_{th,n})\sin(\omega t) + (v_{th}(t) - v_{th,n})\cos(\omega t)$. Furthermore, the local ODE associated to δ is derived from the original ODE as below:

$$\Phi_n(t, \delta(t)) = f(t, \bar{X}_n(t) + \delta(t)) - \dot{\bar{X}}_n(t) = \lambda\delta(t) + \lambda\bar{X}_n(t) + \bar{b}(t) - \dot{\bar{X}}_n(t) \tag{6.20}$$

where we can define the local oscillating forcing term $\bar{b}_n$ which is directly linked to the difference between the original equation oscillating forcing term $\bar{b}(t) = -(\lambda u_\infty + \omega v_{th}(t))\sin(\omega t) - (\lambda v_\infty - \omega u_{th}(t))\cos(\omega t)$ and the periodic function used by the SPM $\bar{X}_n(t)$:

$$\bar{b}_n(t) = \bar{b}(t) + \lambda\bar{X}_n(t) - \dot{\bar{X}}_n(t) \tag{6.21}$$

In addition, let us introduce $\bar{b}_{\epsilon_n}$, the oscillating forcing term associated to the perturbation on the Fourier coefficients:

$$\bar{b}_{\epsilon_n} = \lambda\bar{X}_{\epsilon_n^{uv}}(t) - \dot{\bar{X}}_{\epsilon_n}(t) \tag{6.22}$$

Therefore, the ODE RHS function Φ_n can be rewritten as a linear function of δ with an oscillation forcing term:

$$\Phi_n(t, \delta(t)) = \lambda\delta(t) + \bar{b}_n(t) \tag{6.23}$$

First, we have to bring out the theoretical time-derivative of $\delta_{th,n}$, i.e. $\dot{\delta}_{th,n}(t)$. So, in a

first time, let us consider the oscillating forcing term associated to the correction (6.21):

$$\begin{aligned}\bar{b}_n(t) &= \bar{b}(t) + \lambda\bar{X}_n(t) - \dot{\bar{X}}_n(t)\\ &= \bar{b}(t) + \lambda\left[\bar{X}_{th,n}(t) + \bar{X}_{\epsilon_n^{uv}}(t)\right] - \left[\dot{\bar{X}}_{th,n}(t) + \dot{\bar{X}}_{\epsilon_n}(t)\right]\\ &= \dot{u}_{th,n}\sin(\omega t) + \dot{v}_{th,n}\cos(\omega t) + \omega\left[(u_{th}(t) - u_{th,n})\cos(\omega t) - (v_{th}(t) - v_{th,n})\sin(\omega t)\right] + \bar{b}_{\epsilon_n}(t)\\ &= \dot{\delta}_{th,n}(t) - [\dot{u}_{th}(t) - \dot{u}_{th,n}]\sin(\omega t) - [\dot{v}_{th}(t) - \dot{v}_{th,n}]\cos(\omega t) + \bar{b}_{\epsilon_n}(t)\end{aligned}$$

Hence, if we use the fact that $\dot{u}_{th}(t) - \dot{u}_{th,n} = \lambda(u_{th}(t) - u_\infty) - \lambda(u_{th,n} - u_\infty) = \lambda(u_{th}(t) - u_{th,n})$ and, similarly, $\dot{v}_{th}(t) - \dot{v}_{th,n} = \lambda(v_{th}(t) - v_{th,n})$, we have

$$\begin{aligned}\Phi_n(t, \delta_{th,n}(t)) &= \lambda\delta_{th,n}(t) + \bar{b}_n(t)\\ &= \lambda(u_{th}(t) - u_{th,n})\sin(\omega t) + \lambda(v_{th}(t) - v_{th,n})\sin(\omega t) + \dot{\delta}_{th,n}(t)\\ &- \lambda(u_{th}(t) - u_{th,n})\sin(\omega t) + \lambda(v_{th}(t) - v_{th,n}) + \bar{b}_{\epsilon_n}(t)\\ &= \dot{\delta}_{th,n}(t) + \bar{b}_{\epsilon_n}(t)\end{aligned}$$

In addition, we can readily prove the following formulas, which are important for the next developments:

$$\begin{aligned}\Phi_n(t_n, \delta_n^+) &= \Phi_n(t_n, \delta_{th,n}(t_n) + \epsilon_n^{\delta+}) = \Phi_n(t_n, \delta_{th,n}(t_n)) + \lambda\epsilon_n^{\delta+}\\ \Phi_n(t_{n+1}, \delta_{n+1}^-) &= \Phi_n(t_{n+1}, \delta_{th,n}(t_{n+1}) + \epsilon_{n+1}^{\delta-}) = \Phi_n(t_{n+1}, \delta_{th,n}(t_{n+1})) + \lambda\epsilon_{n+1}^{\delta-}\end{aligned}$$

Then, from the SPM decomposition, the global error can be also decomposed as

$$\begin{aligned}\epsilon_{n+1} &= X_{n+1} - X_{th}(t_{n+1})\\ &= (\bar{X}_n(t_{n+1}) + \delta_{n+1}^-) - (\bar{X}_{th,n}(t_{n+1}) + \delta_{th,n}(t_{n+1}))\\ &= \bar{X}_{\epsilon_n^{uv}}(t_{n+1}) + \epsilon_{n+1}^{\delta-}\end{aligned}$$

The objective is now to quantify $\epsilon_{n+1}^{\delta-}$:

$$\begin{aligned}\epsilon_{n+1}^{\delta-} &= \delta_{n+1}^- - \delta_{th,n}(t_{n+1})\\ &= \delta_n^+ + \frac{h_n}{2}\left[\Phi_n(t_n, \delta_n^+) + \Phi_n(t_{n+1}, \delta_{n+1}^-))\right] - \delta_{th,n}(t_{n+1})\\ &= \underbrace{\delta_{th,n}(t_n) + \frac{h_n}{2}\left[\dot{\delta}_{th,n}(t_n) + \dot{\delta}_{th,n}(t_{n+1}))\right] - \delta_{th,n}(t_{n+1})}_{\text{=Residual of the Trapezoidal Formula=}LTE_n^{\delta_{th,n}}} + \epsilon_n^{\delta+} + \frac{h_n}{2}\left[\lambda(\epsilon_n^{\delta+} + \epsilon_{n+1}^{\delta-}) + \bar{b}_{\epsilon_n}(t_n) + \bar{b}_{\epsilon_n}(t_{n+1})\right]\end{aligned}$$

where

$$LTE_n^{\delta_{th,n}} = \frac{h_n^3}{12}\delta_{th,n}^{(3)}(t_n) + \mathcal{O}(h_n^4)$$

Consequently

$$\epsilon_{n+1}^{\delta-} = \eta_n \epsilon_n^{\delta+} + \mu_n \left[LTE_n^{\delta_{th,n}} + \frac{h_n}{2}(\bar{b}_{\epsilon_n}(t_n) + \bar{b}_{\epsilon_n}(t_{n+1})) \right] \quad \text{where} \quad \begin{cases} \eta_n &= \dfrac{1 + \frac{h_n\lambda}{2}}{1 - \frac{h_n\lambda}{2}} \\ \mu_n &= \dfrac{1}{1 - \frac{h_n\lambda}{2}} \end{cases} \tag{6.24}$$

Then, by injecting $\epsilon_n^{\delta+} = \delta_n^+ - \delta_{th,n}(t_n) = \epsilon_n - \bar{X}_{\epsilon_n^{uv}}(t_n)$ into this equation, the relation (6.14) is finally obtained:

$$\epsilon_{n+1} = \underbrace{\eta_n \epsilon_n}_{\text{classical numerical damping}} + \underbrace{\mu_n \left[LTE_n^{\delta_{th,n}} - LTE_n^{\bar{X}_{\epsilon_n^{uv}}} \right]}_{\text{local error (specific to the SPM)}}$$

□

If we look more closely to this recursive sequence, we can see that the formula is divided into two parts:

1. $\eta_n \epsilon_n$ corresponding to the previous global error damping, which is similar to those of a classical integration scheme.
2. $\mu_n LTE_n^{\delta_{th,n}} - \mu_n LTE_n^{\bar{X}_{\epsilon_n^{uv}}}$ which is the local error injection. Concerning this local error several comments can be done:
   - Both $LTE_n^{\delta_{th,n}}$ and $LTE_n^{\bar{X}_{\epsilon_n^{uv}}}$ are $\mathcal{O}(h_n^3)$ terms, which means that the method is second-order consistent, as wanted.
   - $LTE_n^{\delta_{th,n}}$ corresponds to the theoretical correction which is defined in reference to the chosen periodic function. For instance, in our developments, it was a sinusoid oscillating at a fixed fundamental frequency, without harmonics, and with piecewise constant Fourier coefficients. So the theoretical periodic function was this function with the theoretical values of the Fourier coefficients evaluated at the left bound of the integration interval. Then, the theoretical correction was a sine function with time-varying Fourier coefficients, corresponding to the difference between time-varying and the constant envelopes, i.e. $(u_{th}(t) - u_{th}(t_n), v_{th}(t) - v_{th}(t_n))$. Therefore we can deduct that, if the system comes back to steady-state, this difference tends toward zero as $\dot{u}_{th}(t), \dot{v}_{th}(t) \to 0$. Finally, this means that the theoretical correction naturally tends towards zero and so the corresponding injection of error.
   - $LTE_n^{\bar{X}_{\epsilon_n^{uv}}}$ corresponds to the injection of error due to the error done on the Fourier coefficients which could be large during transients but should tend toward zero when the system comes back to steady-state. It is thus necessary to have an accurate and especially a convergent estimator of the Fourier coefficient in order to get optimal performances.

This recursive sequence can be rewritten as

$$\epsilon_{n+1} = \eta_n\eta_{n-1}...\eta_0\epsilon_0 + \eta_n\eta_{n-1}...\eta_1\mu_0 LTE_0^{SPM} + \eta_n\eta_{n-1}...\eta_2\mu_1 LTE_1^{SPM} + ... \\ + \eta_n\mu_{n-1}LTE_{n-1}^{SPM} + \mu_n LTE_n^{SPM}$$

with $LTE_n^{SPM} = LTE_n^{\delta_{th,n}} - LTE_n^{\bar{X}_{\epsilon_n^{uv}}}$. By introducing the convention $\eta_{n+1} = 1$, this sum can be rewritten in a more compact form as

$$\epsilon_{n+1} = \sum_{k=1}^{n+1}\left[\prod_{j=k}^{n+1}\eta_j\right]\mu_{k-1}LTE_{k-1}^{SPM} + \left[\prod_{j=0}^{n+1}\eta_j\right]\epsilon_0 \tag{6.25}$$

Therefore, as $\lambda < 0$, the global error can be upper-bounded by a constant which is independent from the step size since there exists $\tilde{\eta} > \max_j \eta_j$ such as $\tilde{\eta} \leq 1$ and $\tilde{\mu} > \max_j \mu_j$ such as $\tilde{\mu} \leq 1$.

Furthermore, the formula (6.14) enables to compare the SPM with the classical trapezoidal formula, whose global error is given by

$$\epsilon_{n+1}^{TR} = \eta_n\epsilon_n^{TR} + \mu_n LTE_n^{X_{th}} \tag{6.26}$$

Actually, if we use the same step sequence for the classical Trapezoidal formula and the SPM (which is never done in practice as the aim of the SPM is to use larger steps rather than to be more precise, since we use an adaptive step size strategy which adapts the time step to keep the global error close to a fixed tolerance), we only have to compare the local injection of error of each method, i.e.

$$e_{n+1}^{SPM} = LTE_n^{\delta_{th,n}} - LTE_n^{\bar{X}_{\epsilon_n^{uv}}} \\ e_{n+1}^{TR} = LTE_n^{X} = LTE_n^{\delta_{th,n}} + LTE_n^{\bar{X}_{th,n}}$$

So, if we assume that the relative error on the Fourier coefficients is small enough i.e. if $||\epsilon_n^{uv}|| \ll \sqrt{u_{th,n}^2 + v_{th,n}^2}$, which is generally true, the SPM is more accurate than the trapezoidal formula. Indeed, if the Fourier coefficients used in the SPM are too biased or noisy, it would likely mean that the system is facing a strong EMT transient, which the SPM is not designed for. However, as mentioned above, this actually means that the SPM will use larger step size than the classical trapezoidal formula since it is based on an adaptive step size strategy.

6.3.4 Step size adaptation

In this part, we present hints for investigating the step size adaptation in response to Fourier coefficients errors. We will focus on the local truncation error, so we assume

that the global error at time t_n is equal to zero, i.e. $\epsilon_n = 0$ and $X_n^{(j)} = X_{th}^{(j)}(t_n)$. Our objective is to verify if the local adjustment of the step size h_n enables the integration step on the correction term δ to compensate the Fourier coefficients error $(\epsilon_n^u, \epsilon_n^v)$ so that the local truncation error on the global solution remains under the chosen tolerance, i.e. $||\epsilon_{n+1}|| < TOL$. To do this, we could use the following methodology:

1. Considering the above-described input data: the global solution is exact at time t_n so that $X_n^{(j)} = X_{th}^{(j)}(t_n)$ and the erroneous Fourier coefficients $(u_n, v_n) = (u_{th,n}, v_{th,n}) + (\epsilon_n^u, \epsilon_n^v)$.
2. The local initial correction term have an error which is equal, in steady-state, to $\epsilon_n^{\delta+} = -X_{\epsilon_n}(t_n)$.
3. Compute the correction term at next time t_{n+1}, i.e. δ_{n+1}^+, and deducing the associated integration error $\epsilon_{n+1}^{\delta-}$.
4. By assuming that the step size adjustment enables to accurately control the global error on the correction term, evaluating the maximum step size h_{n+1} such as $|\epsilon_{n+1}^{\delta-}| \leq TOL$.
5. Deducing the global error on the complete solution ϵ_{n+1} from those on the correction term $\epsilon_{n+1}^{\delta-}$ and those on the periodic function $\bar{X}_{\epsilon_n^{uv}}(t_{n+1})$ with the estimated maximum step size h_{n+1} and verifying that $||\epsilon_{n+1}|| \leq TOL$.

6.4 Theoretical results on the final estimator

The results obtained in the previous section partially show that the adaptive step size integration performed on the correction term enables to control the global error. In particular, the methodology is done so that it can compensate the error introduced by the periodic function, for instance when the Fourier coefficients are biased, which may be the case when the system is in transients. Hence, in this section, we focus on the estimator, by deriving a global formula for the Fourier coefficients updates when it is applied to our test equation at first and then deducing its particular behavior when the system is in steady-state and in transients. Indeed, in the considered case, the global formula suggests that the Fourier coefficients updates can be decomposed into two parts: one involving the difference with their asymptotic value and the other corresponding to a transient term.

6.4.1 Derivation of the global formula

Our objective is firstly to derive a global formula for the Fourier coefficients when the SPM is applied to the studied equation. By this way, the estimator particular behavior

in steady-state and during transients could then be deduced.

Let us recall the final estimator formula

$$\begin{bmatrix} u_{n+1} \\ v_{n+1} \end{bmatrix} = \begin{bmatrix} u_n \\ v_n \end{bmatrix} - \begin{bmatrix} H_n^{uu} & H_n^{uv} \\ H_n^{vu} & H_n^{vv} \end{bmatrix}^{-1} \begin{bmatrix} g_n^u \\ g_n^v \end{bmatrix} \tag{6.27}$$

where H_n and g_n are respectively an approximation of the Hessian matrix of the objective function $R(u, v)$ defined in (5.53) and its gradient, whose formulas are introduced in the section 5.4.

Our scalar equation with an oscillating forcing term (5.5) can be rewritten in implicit form as

$$F(t, X(t), \dot{X}(t)) = \dot{X}(t) - \lambda X(t) - \bar{b}(t) \tag{6.28}$$

By injecting the current periodic function $\bar{X}_n(t)$ into this residual function, we obtain the residual function corresponding to the stationarity measurement with reference to the prior steady-state trajectory:

$$\bar{\rho}_n(t) = \dot{\bar{X}}_n(t) - \lambda \bar{X}_n(t) - \bar{b}(t) \tag{6.29}$$

And the partial-derivatives of the DAE residual function, that are used in the estimator, are given by

$$\partial_{X_s} \bar{\rho}_n(t) = -\lambda \tag{6.30}$$

$$\partial_{\dot{X}_s} \bar{\rho}_n(t) = 1 \tag{6.31}$$

Hence, when applied to the differential equation (5.5), the matrix corresponding to the Hessian matrix of the objective function within the estimator linear system is composed of

$$H_n^{uu} = \int_{t_{n+1}}^{t_{n+1}+T} \left[-\lambda \sin(\omega t) + \omega \cos(\omega t)\right]^2 = \frac{T}{2}(\lambda^2 + \omega^2) \tag{6.32}$$

$$H_n^{vv} = \int_{t_{n+1}}^{t_{n+1}+T} \left[-\lambda \cos(\omega t) - \omega \sin(\omega t)\right]^2 = \frac{T}{2}(\lambda^2 + \omega^2) \tag{6.33}$$

$$H_n^{uv} = \int_{t_{n+1}}^{t_{n+1}+T} \left[-\lambda \sin(\omega t) + \omega \cos(\omega t)\right] \left[-\lambda \cos(\omega t) - \omega \sin(\omega t)\right] = 0 \tag{6.34}$$

$$H_n^{vu} = H_n^{uv} = 0 \tag{6.35}$$

And the vector corresponding to its gradient is given by

$$g_n^u = \int_{t_{n+1}}^{t_{n+1}+T} \left[-\lambda \sin(\omega t) + \omega \cos(\omega t)\right] \left[\dot{\bar{X}}_n(t) - \lambda \bar{X}_n(t) - \bar{b}(t)\right] dt \tag{6.36}$$

$$= \frac{T}{2}(\lambda^2 + \omega^2)(u_n - u_\infty) + \tilde{g}_n^u \tag{6.37}$$

$$g_n^v = \int_{t_{n+1}}^{t_{n+1}+T} \left[-\lambda \cos(\omega t) - \omega \sin(\omega t)\right] \left[\dot{\bar{X}}_n(t) - \lambda \bar{X}_n(t) - \bar{b}(t)\right] dt \tag{6.38}$$

$$= \frac{T}{2}(\lambda^2 + \omega^2)(v_n - v_\infty) + \tilde{g}_n^v \tag{6.39}$$

where $\tilde{g}_n^u$ and $\tilde{g}_n^v$ are transient terms that tend toward zero when the system returns to steady-state. Indeed, the oscillating forcing term defined in (5.6) can be rewritten as a sinusoid with time-varying Fourier coefficients, i.e. $\bar{b}(t) = u_{\bar{b}}(t)\sin(\omega t) + v_{\bar{b}}(t)\cos(\omega t)$. More precisely, its Fourier coefficients contain a constant term, associated to the steady-state, and a time-varying term, since

$$u_{\bar{b}}(t) = -\lambda u_\infty - \omega v_\infty - \omega(v_0 - v_\infty)e^{\lambda t}$$
$$v_{\bar{b}}(t) = -\lambda v_\infty + \omega u_\infty + \omega(u_0 - u_\infty)e^{\lambda t}$$

To distinguish them, we introduce $\tilde{u}_{\bar{b}}(t) = -\omega(v_0 - v_\infty)e^{\lambda t}$ and $\tilde{v}_{\bar{b}}(t) = \omega(u_0 - u_\infty)e^{\lambda t}$. Let us detail the computation of $\tilde{g}_n^u$. So, we can show that

$$\begin{aligned}
\tilde{g}_n^u &= \lambda \int_{t_{n+1}}^{t_{n+1}+T} \tilde{u}_{\bar{b}}(t)\sin^2(\omega t)dt + \lambda \int_{t_{n+1}}^{t_{n+1}+T} \tilde{v}_{\bar{b}}(t)\sin(\omega t)\cos(\omega t)dt \\
&- \omega \int_{t_{n+1}}^{t_{n+1}+T} \tilde{u}_{\bar{b}}(t)\sin(\omega t)\cos(\omega t)dt - \omega \int_{t_{n+1}}^{t_{n+1}+T} \tilde{v}_{\bar{b}}(t)\cos^2(\omega t)dt \\
&= \lambda \left[\tilde{u}_{\bar{b}}(t)\frac{\lambda^2(1-\cos(2\omega t)) - \lambda(2\omega\sin(2\omega t)) + 4\omega^2}{2(\lambda^3 + 4\lambda\omega^2)}\right]_{t_{n+1}}^{t_{n+1}+T} \\
&+ \lambda \left[\tilde{v}_{\bar{b}}(t)\frac{\lambda\sin(2\omega t) - 2\omega\cos(2\omega t)}{2(\lambda^2 + 4\omega^2)}\right]_{t_{n+1}}^{t_{n+1}+T} \\
&- \omega \left[\tilde{u}_{\bar{b}}(t)\frac{\lambda\sin(2\omega t) - 2\omega\cos(2\omega t)}{2(\lambda^2 + 4\omega^2)}\right]_{t_{n+1}}^{t_{n+1}+T} \\
&- \omega \left[\tilde{v}_{\bar{b}}(t)\frac{\lambda^2(1+\cos(2\omega t)) + \lambda(2\omega\sin(2\omega t)) + 4\omega^2}{2(\lambda^3 + 4\lambda\omega^2)}\right]_{t_{n+1}}^{t_{n+1}+T}
\end{aligned}$$

Then, for simplifying this equation, we use the fact that the time-derivative of $\tilde{u}_{\bar{b}}$ and $\tilde{v}_{\bar{b}}$ are given by: $\dot{\tilde{u}}_{\bar{b}}(t) = \lambda\tilde{u}_{\bar{b}}(t)$ and $\dot{\tilde{v}}_{\bar{b}}(t) = \lambda\tilde{v}_{\bar{b}}(t)$. In addition, we have the following relations between these Fourier coefficients and those of the theoretical solution: $\tilde{u}_{\bar{b}}(t) = -\omega(v_{th}(t) - v_\infty)$ and $\tilde{v}_{\bar{b}}(t) = \omega(u_{th}(t) - u_\infty)$. To finish, the difference between the value of the theoretical Fourier coefficients and their asymptotic value is linked to their time-derivative by: $u_{th}(t) - u_\infty = \frac{\dot{u}_{th}}{\lambda}$ and $v_{th}(t) - v_\infty = \frac{\dot{v}_{th}}{\lambda}$. Finally, the different terms of $\tilde{g}_n^u$

can be rewritten as:

$$\begin{aligned}
\tilde{g}_n^u = &\frac{\lambda}{2(\lambda^3+4\lambda\omega^2)}\left[-\omega(1-\cos(2\omega t))\ddot{v}_{th}(t)+(2\omega^2\sin(2\omega t)-\frac{4\omega^3}{\lambda})\dot{v}_{th}(t)\right]_{t_{n+1}}^{t_{n+1}+T}\\
&+\frac{\lambda}{2(\lambda^2+4\omega^2)}\left[(\omega\sin(2\omega t)-\frac{2\omega^2\cos(2\omega t)}{\lambda})\dot{u}_{th}(t)\right]_{t_{n+1}}^{t_{n+1}+T}\\
&-\frac{\omega}{2(\lambda^2+4\omega^2)}\left[(-\omega\sin(2\omega t)+\frac{2\omega^2\cos(2\omega t)}{\lambda})\dot{v}_{th}(t)\right]_{t_{n+1}}^{t_{n+1}+T}\\
&-\frac{\omega}{2(\lambda^3+4\lambda\omega^2)}\left[\omega(1+\cos(2\omega t))\ddot{u}_{th}(t)+(2\omega^2\sin(2\omega t)+\frac{4\omega^3}{\lambda})\dot{u}_{th}(t)\right]_{t_{n+1}}^{t_{n+1}+T}
\end{aligned}$$

Identically, we can prove that

$$\begin{aligned}
\tilde{g}_n^v = &\frac{\lambda}{2(\lambda^2+4\omega^2)}\left[(-\omega\sin(2\omega t)+\frac{2\omega^2\cos(2\omega t)}{\lambda})\dot{v}_{th}(t)\right]_{t_{n+1}}^{t_{n+1}+T}\\
&+\frac{\lambda}{2(\lambda^3+4\lambda\omega^2)}\left[\omega(1+\cos(2\omega t))\ddot{u}_{th}(t)+(2\omega^2\sin(2\omega t)+\frac{4\omega^3}{\lambda})\dot{u}_{th}(t)\right]_{t_{n+1}}^{t_{n+1}+T}\\
&+\frac{\omega}{2(\lambda^3+4\lambda\omega^2)}\left[-\omega(1-\cos(2\omega t))\ddot{v}_{th}(t)+(2\omega^2\sin(2\omega t)-\frac{4\omega^3}{\lambda})\dot{v}_{th}(t)\right]_{t_{n+1}}^{t_{n+1}+T}\\
&+\frac{\omega}{2(\lambda^2+4\omega^2)}\left[(\omega\sin(2\omega t)-\frac{2\omega^2\cos(2\omega t)}{\lambda})\dot{u}_{th}(t)\right]_{t_{n+1}}^{t_{n+1}+T}
\end{aligned}$$

So, in this form, we can see that these terms directly depend on the theoretical Fourier coefficients time-derivatives $u_{th}^{(j)}(t)$ and $v_{th}^{(j)}(t)$, that tend toward zero when the system is in steady-state.

6.4.2 Convergence in steady-state

Proposition 4. *In steady-state, the final SPM estimator applied to the problem* (5.5) *is convergent which means that its bias tends toward zero, i.e.* $(\epsilon_n^u, \epsilon_n^v) \to 0$.

Proof. From the global estimator formula, we can deduce that the estimator is exact in steady-state. Indeed, as $u_{th}^{(j)}(t), v_{th}^{(j)}(t) \to 0$ for $j \geq 1$, the terms $\tilde{g}_n^u$ and $\tilde{g}_n^v$ vanish in steady-state. The Fourier coefficients updates computed by the estimator are finally given

by:

$$\begin{bmatrix}\Delta_{n+1}^u\\ \Delta_{n+1}^v\end{bmatrix} = -\begin{bmatrix}H_n^{uu} & H_n^{uv}\\ H_n^{vu} & H_n^{vv}\end{bmatrix}^{-1}\begin{bmatrix}g_n^u\\ g_n^v\end{bmatrix} \tag{6.40}$$

$$= -\frac{2}{T(\lambda^2+\omega^2)}\begin{bmatrix}\frac{T}{2}(\lambda^2+\omega^2)(u_n-u_\infty)\\ \frac{T}{2}(\lambda^2+\omega^2)(v_n-v_\infty)\end{bmatrix} \tag{6.41}$$

$$= \begin{bmatrix}u_\infty-u_n\\ v_\infty-v_n\end{bmatrix} \tag{6.42}$$

i.e. the estimated Fourier coefficients are exact since

$$(u_{n+1},v_{n+1}) = (u_n,v_n)+(u_\infty-u_n,v_\infty-v_n) = (u_\infty,v_\infty) \tag{6.43}$$

□

6.4.3 Behavior during transients

From the previously proven formula, we also can see that the estimated Fourier coefficients are biased during transients. This result is logical since the idea of this estimator is to compute Fourier coefficients minimizing an energy corresponding to a measurement of the system stationarity. However, the formula also shows that the bias with reference to the asymptotic values can be majored by the time-derivatives of the theoretical Fourier coefficients. Hence, the estimator remains stable.

Proposition 5. *During transients, the Fourier coefficients update transient term computed by the final SPM estimator applied to the problem* (5.5) *is bounded and decays so that it tends toward zero when the system tends toward steady-state.*

Proof. From the global formula derived in the first subsection, we can deduce the following bound for the transient term $\tilde{g}_n^u$:

$$\begin{aligned}|\tilde{g}_n^u| \leq\ & |\lambda|\omega|v_0-v_\infty|e^{\lambda t_{n+1}}|\frac{2\lambda^2+2\omega|\lambda|+4\omega^2}{|\lambda|^3+4|\lambda|\omega^2}\\ &+|\lambda|\omega|u_0-u_\infty|e^{\lambda t_{n+1}}|\frac{|\lambda|+2\omega}{\lambda^2+4\omega^2}\\ &+\omega\omega|v_0-v_\infty|e^{\lambda t_{n+1}}|\frac{|\lambda|+2\omega}{\lambda^2+4\omega^2}\\ &+\omega\omega|u_0-u_\infty|e^{\lambda t_{n+1}}|\frac{2\lambda^2+2\omega|\lambda|+4\omega^2}{|\lambda|^3+4|\lambda|\omega^2}\end{aligned}$$

and identically

$$
\begin{aligned}
|\tilde{g}_n^v| \leq\ & |\lambda|\omega|v_0 - v_\infty|e^{\lambda t_{n+1}}|\frac{|\lambda| + 2\omega}{\lambda^2 + 4\omega^2} \\
& + |\lambda|\omega|u_0 - u_\infty|e^{\lambda t_{n+1}}|\frac{2\lambda^2 + 2\omega|\lambda| + 4\omega^2}{|\lambda|^3 + 4|\lambda|\omega^2} \\
& + \omega\omega|v_0 - v_\infty|e^{\lambda t_{n+1}}|\frac{2\lambda^2 + 2\omega|\lambda| + 4\omega^2}{|\lambda|^3 + 4|\lambda|\omega^2} \\
& + \omega\omega|u_0 - u_\infty|e^{\lambda t_{n+1}}|\frac{|\lambda| + 2\omega}{\lambda^2 + 4\omega^2}
\end{aligned}
$$

□

As an opening, for further developments, the problem would be also to properly define the error on the Fourier coefficients in transients:

- Instantaneous value, i.e. $u_{th}(t_{n+1})$ and $v_{th}(t_{n+1})$
- Mean value, i.e. $\bar{u}_{th}(t_{n+1}) = \int_{t_{n+1}}^{t_{n+1}+T} u_{th}(t)dt$ and $\bar{v}_{th}(t_{n+1}) = \int_{t_{n+1}}^{t_{n+1}+T} v_{th}(t)dt$

To finish, by considering a test containing a perfect generator, a transmission line and a variable resistance, we can show that the computed Fourier coefficients are directly linked to the classical phasor solution. Indeed, in the objective function, the non-oscillating components are replaced by their last instantaneous value while, for the oscillating components, the Fourier coefficients are assumed to be constant and so their time-derivative are set to zero.

6.5 Numerical results

In this section, we present SPM performances results on the studied test case (5.5). The used linear coefficient value of the associated equation is set to $\lambda = \frac{-1}{5T}$. For recall, its solution is a sinusoid whose envelope exponentially decays. Therefore, it enables to illustrate the investigated properties of the SPM since the system is in transients at the beginning of the simulation and then progressively goes to steady-state. As explained in this chapter, the main objective of the SPM is to optimize the simulations performances by enabling the use of large time steps when the system is in steady-state while keeping a fine control on the numerical error, especially when the system is in transient during which the performances of the method are thus expected to be dramatically reduced.

Our results focus on three main aspects:

- The convergence of the Fourier coefficients: the bias of the estimated Fourier coefficients is expected to be potentially high during the transient phase of the simulation but should then decay as the time-derivatives of the theoretical Fourier coefficients

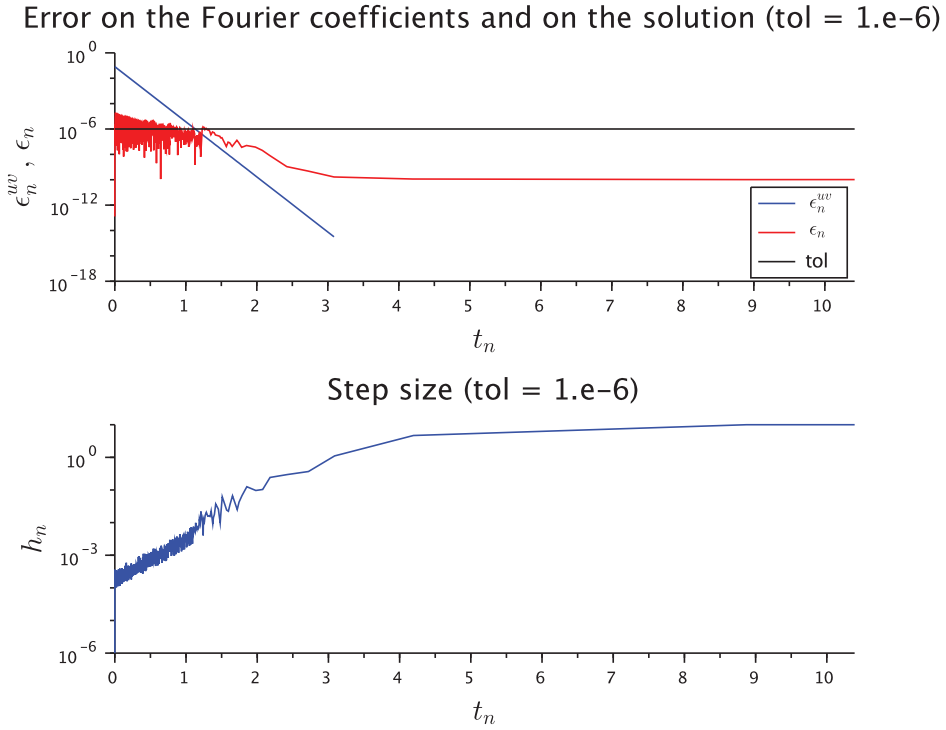

Figure 6.1: Results obtained with the SPM for TOL = 1.e-6. Top: Fourier coefficients and global solution error (on the abscissa: time in seconds, on the ordinate: Fourier coefficients and global solution error in arbitrary units), down: step size (on the abscissa: time in seconds, on the ordinate: step size in seconds).

tend toward zero. Consequently, in steady-state, the estimated Fourier coefficients should be very close to the theoretical asymptotic values.

- The global error: it should remain lower or close to the chosen tolerance. More precisely, in the light of the results presented in the section 6.3.3, the local numerical error injection should be the highest during transients because of the contributions of, on the one hand, the theoretical local correction term time-variations and, on the other hand, the Fourier coefficients error which might be important. On the contrary, when the system is in steady-state, the local error injection should be very low and so the global error should be progressively damped by the solver.
- The step size adjustment: when the system is in transients, the step size should be low in order to compensate the potentially high local error injection as mentioned above. In a second time, the step size should progressively increase to reach high values while the system tends toward its steady-state since the Fourier coefficients estimator bias and so the correction term should be damped.

Then, the figure 6.1 is divided into two parts:

- The illustration at the top shows the Fourier coefficients error (in blue) and the global error (in red) during the simulation time with reference to the chosen tolerance (in black) in order to highlight the control of the numerical error by the SPM.
- The illustration at the bottom shows the step size during the simulation which should reach high values when the Fourier coefficients error and so the global error are significantly lower than the chosen tolerance in the figure at the top.

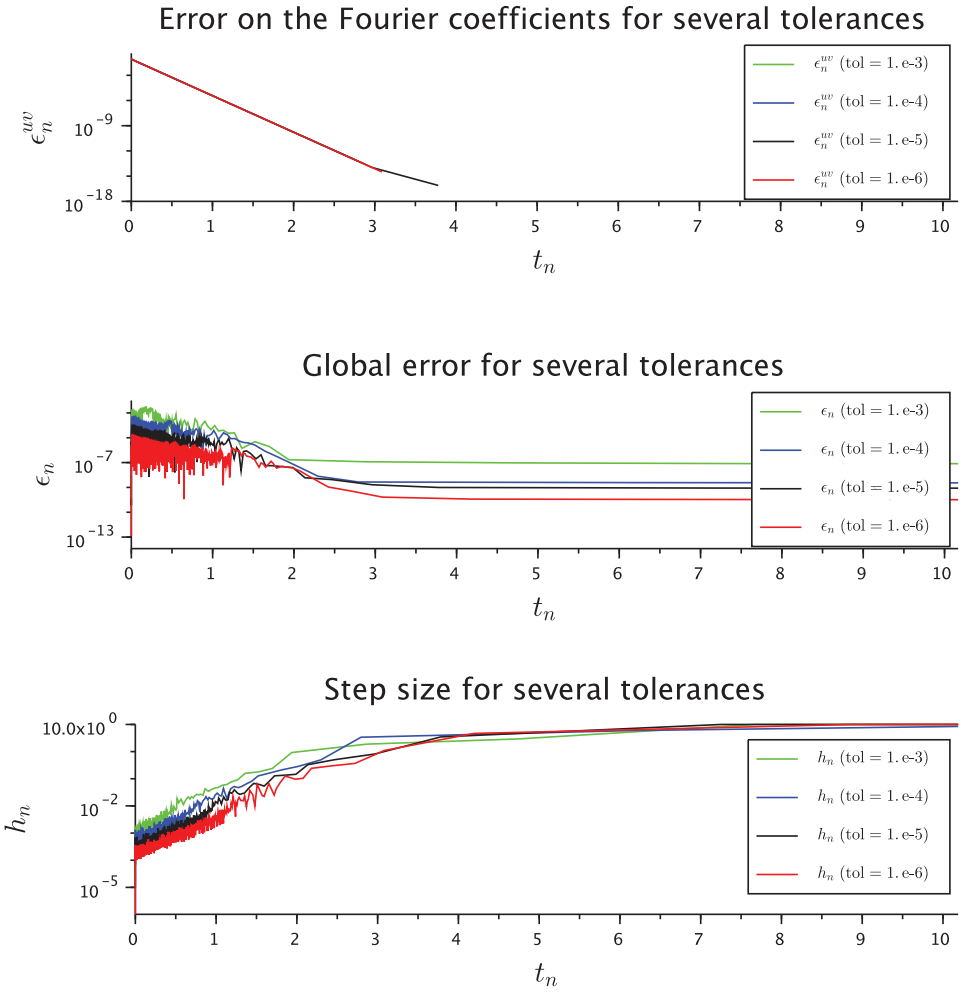

Figure 6.2: Results obtained with the SPM for several tolerances (green: 1.e-3, blue: 1.e-4, black: 1.e-5, red: 1.e-6). Top: Fourier coefficients error (on the abscissa: time in seconds, on the ordinate: Fourier coefficients error in arbitrary units), middle: global error (on the abscissa: time in seconds, on the ordinate: global error in arbitrary units), down: step size (on the abscissa: time in seconds, on the ordinate: step size in seconds).

Then, the presented results indicate that the SPM has the wanted global behavior. Indeed, we can see that the Fourier coefficients error dramatically decay when the

system reaches is in steady-state and so the step size increases to reach very high values. In particular, figure 6.2 shows that this behavior is obtained for all the tested tolerances. In this figure, an other interesting point to note is that the Fourier coefficients error does actually not seem correlated to the used tolerance. This result is logical since the considered scalar test case only contains an oscillating solution. Then, the Fourier coefficients updates only depend on their previous value and on problem-related theoretical quantities (e.g. the theoretical Fourier coefficients, their time-derivatives and their asymptotic values) but not on numerically integrated data, which would be affected by the tolerance choice. That is why further developments should consider a vector test case composed of both oscillating and non-oscillating components. Indeed, in such a system, as the non-oscillating components play the role of parameters in the optimization problem that is approximately solved by our estimator, the computed Fourier coefficients updates might be impacted by the tolerance choice. The optimization problem sensitivity to the non-oscillating components precision could then be investigated.

Concerning the control of the error, we can see that the step size varies within a wide range of values: in the beginning of the simulation, as the system is in transients, it remains very low in order the method to catch the solution envelope dynamics and it progressively increases while the system tend toward its steady-state. When the theoretical Fourier coefficients time-derivatives are negligible, the step size reaches its maximum value ($h_{max} = 10s$). However, our results (see figure 6.3) also show that the global error can be greater than the chosen tolerance during transients. This tolerance crossing is particularly visible when using low tolerances such as TOL=1.e-6. Thus, the step size adjustment strategy should probably be modified, for instance by changing the error constant used for estimating the local truncation error within the numerical integrator, in order the method to better damps the local injection of numerical error during transients. But it means that a potentially much higher number of iterations is to expect. In spite of this tolerance crossing issue, the error seems controlled in that sense that it remains stable and the convergence of the numerical solution toward the theoretical solution is regular.

To conclude, in our theoretical and numerical results light, the SPM has the expected properties on the system (5.5). In particular, even if some tuning might be necessary still, the step size adaptation globally enables to catch the investigated transients in the beginning of the simulation and to drastically optimize the solver performances once the system is in steady-state. The step size varies within a very wide range of values as it can decrease until a few tens of microseconds while its maximum value is ten seconds.

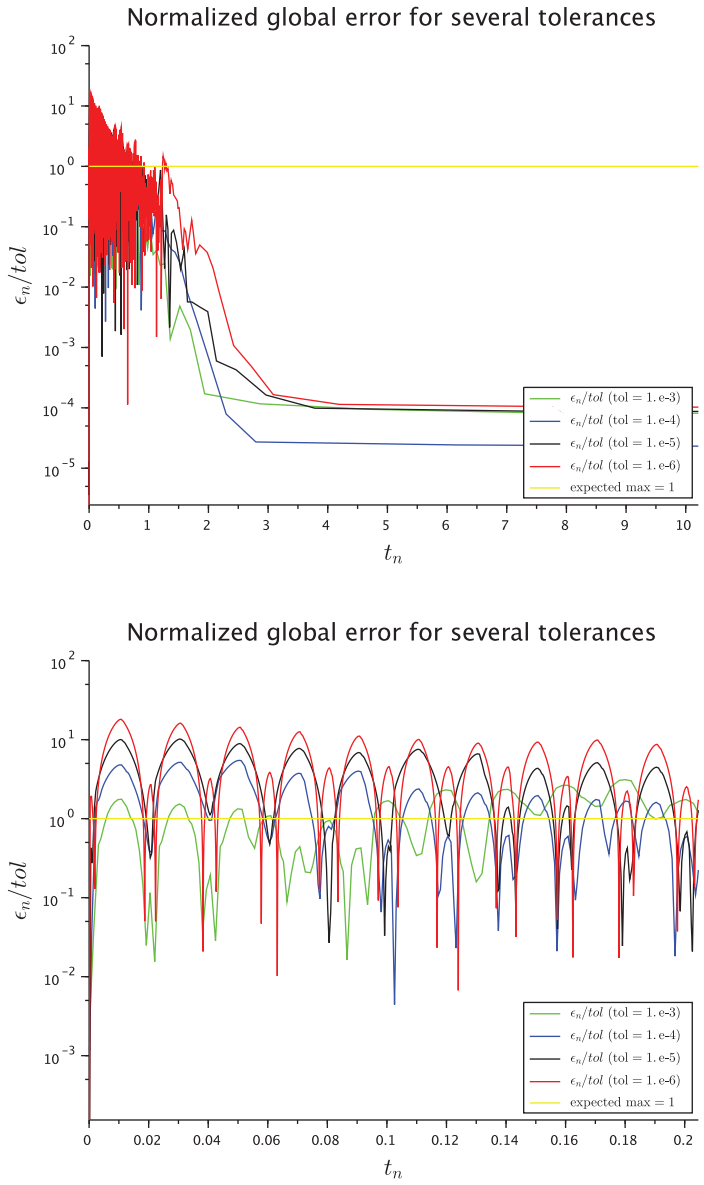

Figure 6.3: Up: Ratio between the global error and the tolerance (on the abscissa: time in seconds, on the ordinate: global error scaled by the tolerance in arbitrary units) for different tested tolerances. Down: idem with a focus on the beginning of the simulation (during transients). This quantity should ideally remain lower than one.

This results from the particularly fast Fourier coefficients estimator convergence. As an opening, this theoretical SPM validation could be pursued considering non-linear systems with both oscillating and non-oscillating solutions and also aim at developing a robust mathematical criterion for switching from a classical solver to the SPM (and vice versa) in order to efficiently deal with strong EMT phenomena.

Part III

Implementation

7

SPM integration into SUNDIALS IDA

Contents

In this chapter, IDASPM, our implementation of the SPM into IDA, is presented. IDA is the implicit differential algebraic equations solver from the SUNDIALS library [34]. It has been developed by the Lawrence Livermore National Laboratory, which still maintains its implementation. It is a modern version of the solver DASSL [49], which is an

industrially validated implementation of the adaptive step size Backward-Differentiation-Formula (BDF) [6]. That's why IDASPM also uses the BDF as basic integration scheme. As it does not require a particular type of predictor or corrector, the SPM method can actually be implemented in any adaptive time step solver. For instance, a Scilab code has been written for testing the SPM based on the mixed Trapezoidal Formula - Backward-Differentiation-Formula, as presented in the previous part. For implementing the SPM into SUNDIALS IDA, the idea has been to apply the classical integration procedure to the SPM correction term in order to solve the local DAE problem associated to the local SPM periodic function for deducing the global solution in a second time. SPM-specific functions have been implemented separately, e.g. for the Fourier coefficients estimator. At this chapter end, some optimizations made to the implementation focusing on the estimator and the Jacobian evaluation function are presented.

7.1 Basic integration scheme modification

In SUNDIALS IDA, whose algorithm is detailed in [33], the DAE system to solve is expressed in full-implicit form:

$$F(t, X(t), \dot{X}(t)) = 0 \tag{7.1}$$

The first step is to inject the SPM decomposition for the time interval $[t_n, t_{n+1}]$ in order to obtain the local DAE system on the correction term δ. So, first of all, as mentioned in chapter 3, the Fourier coefficients u_n and v_n are fixed for the time interval $[t_n, t_{n+1}]$. Then, we can define $\Phi_n : \mathbf{R} \times \mathbf{R}^d \times \mathbf{R}^d \to \mathbf{R}^d$, the local DAE function associated to the equations on δ:

$$\Phi_n(t, \delta, \dot{\delta}) = F(t, \bar{X}_n(t) + \delta, \dot{\bar{X}}_n(t) + \dot{\delta}) \tag{7.2}$$

The q-order adaptive step size BDF consists in substituting the time derivative of the solution by the following approximation:

$$\dot{X}_{n+1} \approx \frac{1}{h_n} \sum_{i=0}^{q} \alpha_{n,i} X_{n+1-i} \tag{7.3}$$

where $h_n = t_{n+1} - t_n$. As for the classical BDF, the fixed-point problem on δ_{n+1} is obtained by applying the BDF discretization formula (7.3) to approximate $\dot{\delta}_{n+1}$:

$$\dot{\delta}_{n+1} \approx \frac{1}{h_n} \sum_{i=0}^{q} \alpha_{n,i} \delta_{n+1-i} \tag{7.4}$$

In this equation, $\delta_{n+1-i} = X_{n+1-i} - \bar{X}_n(t_{n+1-i})$. Indeed, the DAE function associated to the correction term is locally defined as it depends on the Fourier coefficients u_n and v_n.

Hence, at the beginning of each time step, the history of δ is updated for taking into account the Fourier coefficients new values. By this way, the consistency of δ is ensured with Φ_n.

By injecting the discretization (7.3) of $\dot{X}_{n+1}$ into (7.1), the solution at next time step t_{n+1} can be obtained by solving the following fixed-point problem:

$$G(X_{n+1}) = F\left(t_{n+1}, X_{n+1}, \frac{\alpha_{n,0}}{h_n} X_{n+1} + \beta_n\right) \tag{7.5}$$

where $\beta_n = \frac{1}{h_n} \sum_{i=1}^{q} \alpha_{n,i} X_{n+1-i}$ corresponds to the predicted part of X_{n+1} time derivative. The Jacobian of this residual function is given by

$$J_G(X_{n+1}) = \frac{\partial F}{\partial X}\left(t_{n+1}, X_{n+1}, \frac{\alpha_{n,0}}{h_n} X_{n+1} + \beta_n\right) \tag{7.6}$$

$$+ \frac{\alpha_{n,0}}{h_n} \frac{\partial F}{\partial \dot{X}}\left(t_{n+1}, X_{n+1}, \frac{\alpha_{n,0}}{h_n} X_{n+1} + \beta_n\right) \tag{7.7}$$

Similarly by applying this integration scheme (7.4) to the differential algebraic equation (7.2), the implicit problem associated to the correction term is gotten:

$$\Gamma_n(\delta_{n+1}) = \Phi_n\left(t_{n+1}, \delta_{n+1}, \frac{\alpha_{n,0}}{h_n}\delta_{n+1} + \beta_n^{\delta}\right) \tag{7.8}$$

$$= F\left(t_{n+1}, \bar{X}_n(t_{n+1}) + \delta_{n+1}, \dot{\bar{X}}_n(t_{n+1}) + \frac{\alpha_{n,0}}{h_n}\delta_{n+1} + \beta_n^{\delta}\right) \tag{7.9}$$

where $\beta_n^{\delta} = \frac{1}{h_n} \sum_{i=1}^{q} \alpha_{n,i} \delta_{n+1-i}$ and whose Jacobian matrix can be directly evaluated from those of the original system:

$$J_{\Gamma_n}(\delta_{n+1}) = \frac{\partial F}{\partial X}\left(t_{n+1}, \bar{X}_n(t_{n+1}) + \delta_{n+1}, \dot{\bar{X}}_n(t_{n+1}) + \frac{\alpha_{n,0}}{h_n}\delta_{n+1} + \beta_n^{\delta}\right) \tag{7.10}$$

$$+ \frac{\alpha_{n,0}}{h_n} \frac{\partial F}{\partial \dot{X}}\left(t_{n+1}, \bar{X}_n(t_{n+1}) + \delta_{n+1}, \dot{\bar{X}}_n(t_{n+1}) + \frac{\alpha_{n,0}}{h_n}\delta_{n+1} + \beta_n^{\delta}\right) \tag{7.11}$$

Once δ_{n+1} is computed, the global solution X_{n+1} is directly obtained from the SPM decomposition:

$$X_{n+1} = \bar{X}_n(t_{n+1}) + \delta_{n+1} \tag{7.12}$$

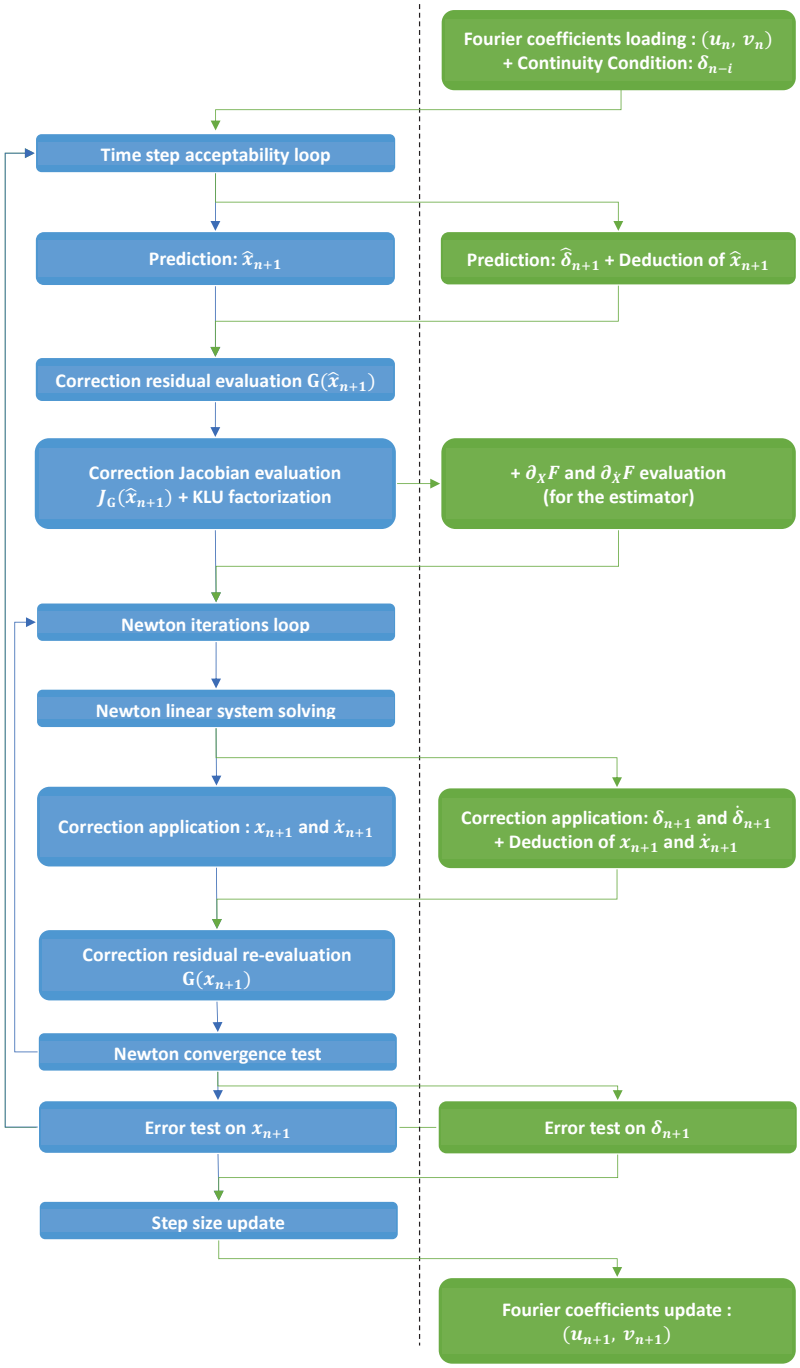

Figure 7.1: Modification of the IDA general algorithm for integrating the SPM: the left part in blue corresponds to the standard IDA algorithm and the right part in green details the SPM-specific operations. As we can see IDASPM really relies on the IDA algorithm for the integration step. The only difference with IDA is that this integration is performed on δ instead of X, in order to deduce this latter in a second time. The only really SPM-specific operations are the Fourier coefficients loading and estimation, the continuity condition and the call to the extended Jacobian evaluation function in order to compute the partial-derivatives of the DAE residual function.

7.2 IDA algorithm modification

In the previous section, we focused on the impact of the SPM on the IDA integration scheme, from a mathematical point-of-view. The resulting modifications (that are summarized in the figure 7.1) mainly affect the variable on which the integration is performed as the idea is essentially to apply the standard IDA integration process to δ instead of X. Some additional operations are required, e.g. for the continuity condition, the global solution deduction and the Fourier coefficients estimation. In this section, the SPM algorithm is presented within the history storage formalism that is used by IDA. This affects the expression of classical operations performed by the adaptive step size predictor-corrector solver. Then, we present the estimator algorithm and finish with the global IDASPM algorithm.

7.2.1 history storage specificity

In IDA, the solution history is stored from modified divided-differences [23], that are referred as ϕ_n:

$$\phi_{n,j} = \left[\prod_{l=0}^{j} \psi_{n,l}\right] [X_n, X_{n-1}, ..., X_{n-j}] \tag{7.13}$$

with

$$\psi_{n,j} = \begin{cases} t_n - t_{n-j} = \sum_{l=0}^{j-1} h_{n-l} & \text{if } \; j \geq 1 \\ 1 & \text{if } \; j = 0 \end{cases} \tag{7.14}$$

$[X_n, X_{n-1}, ..., X_{n-j}]$ is the divided-difference, whose construction rule is

$$[X_n] = X_n \tag{7.15}$$

$$[X_n, X_{n-1}] = \frac{X_n - X_{n-1}}{t_n - t_{n-1}} \tag{7.16}$$

$$[X_n, ..., X_{n-j}] = \frac{[X_n, ..., X_{n-j+1}] - [X_{n-1}, ..., X_{n-j}]}{t_n - t_{n-j}} \tag{7.17}$$

For the SPM implementation into SUNDIALS IDA, the main difficulty concerning the integration step was to properly adapt the SPM to the history storage used in IDA. Indeed, in our initial SPM implementation, the history was not stored in the modified divided-differences formalism but in Nordsieck's formalism [46], which consists in storing an approximation of the Taylor development of the solution:

$$\vec{X}_{n+1} = \left[X_{n+1} \quad h_n \dot{X}_{n+1} \quad \frac{h_n^2}{2} \ddot{X}_{n+1} \quad \dots \quad \frac{h_n^q}{q!} X_{n+1}^{(q)}\right]^T \tag{7.18}$$

In this vector, $X_{n+1}^{(j)}$ is an approximation of the j-th time-derivative of X_{n+1}, with $j = 1, ..., q$, whose expression is given by the used integration scheme. For recall, in

the SPM, the solution is decomposed as the sum of a periodic parametric function that we note $\bar{X}_n$ and a correction term δ for each time integration interval, i.e. for $t \in [t_n, t_{n+1}]$, $X(t) = \bar{X}_n(t) + \delta(t)$ with $\bar{X}_n(t) = u_n \sin(\omega_0 t) + v_n \cos(\omega_0 t)$. And so, when the Fourier coefficients (u_n, v_n) change, in order to ensure the continuity of the interpolation polynomial corresponding to the integration scheme, the Nordsieck's history vector has to be updated from the following formula that we refer as continuity condition:

$$\vec{\delta}_n = \vec{X}_n - \vec{\bar{X}}_n \tag{7.19}$$

where $\vec{X}_n$ is the global solution history which comes from the numerical integration and is globally defined, and $\vec{\bar{X}}_n$ is periodic function history which is locally defined and simply obtained by evaluating the periodic function and its time-derivatives at time t_n:

$$\vec{\bar{X}}_n = \left[\bar{X}_n(t_n) \quad h_n\dot{\bar{X}}_n(t_n) \quad \frac{h_n^2}{2}\ddot{\bar{X}}_n(t_n) \quad \dots \quad \frac{h_n^q}{q!}\bar{X}_n^{(q)}(t_n)\right]^T \tag{7.20}$$

which is very straightforward as the Fourier coefficients are locally considered as constant values, so $\bar{X}_n^{(j)}(t)$ is simply given by

$$\bar{X}_n^{(j)}(t) = \omega^j \left[u_n \sin\left(\omega t + j\frac{\pi}{2}\right) + v_n \cos\left(\omega t + j\frac{\pi}{2}\right)\right] \tag{7.21}$$

The history $\vec{X}_n$, or $\vec{\delta}_n$ in the SPM context, enables to construct the interpolation polynomial corresponding to the chosen predictor-corrector integration scheme. Once the integration step with the predictor-corrector scheme has been performed and so the correction term δ_{n+1} has been computed, the correction term Nordsieck vector $\vec{\delta}_{n+1}$ is updated from the approximation formulas (4.25) seen in chapter 4. These formulas are generally based on the prediction error $\Delta_{n+1} = \delta_{n+1} - \hat{\delta}_{n+1}$. To finish, the solution Nordsieck vector $\vec{X}_{n+1}$ can be directly deduced by applying the SPM decomposition:

$$\vec{X}_{n+1} = \vec{\delta}_{n+1} + \vec{\bar{X}}_{n+1} \tag{7.22}$$

where the Nordsieck vector corresponding to the periodic part of the solution $\vec{\bar{X}}_{n+1}$ is simply evaluated at next time step t_{n+1} as below:

$$\vec{\bar{X}}_{n+1} = \left[\bar{X}_n(t_{n+1}) \quad h_n\dot{\bar{X}}_n(t_{n+1}) \quad \frac{h_n^2}{2}\ddot{\bar{X}}_n(t_{n+1}) \quad \dots \quad \frac{h_n^q}{q!}\bar{X}_n^{(q)}(t_{n+1})\right]^T \tag{7.23}$$

These mathematical operations can be done since Nordsieck's formalism enables to perform linear operations, which is due to the linearity of the time-differentiation operator. Then, $\vec{x}_n + \vec{y}_n = (x \vec{+} y)_n$. The same linearity property is verified in the modified divided-difference formalism, i.e. $\phi_n^x + \phi_n^y = \phi_n^{x+y}$. Thus, for integrating the SPM into SUNDIALS IDA, the continuity relation and the global solution update can be rewritten using the

modified divided-difference vectors of the solution, the periodic function and the correction term:

$$\phi_n^{\delta} = \phi_n^{X} - \phi_n^{\bar{X}_n} \tag{7.24}$$

$$\phi_{n+1}^{X} = \phi_{n+1}^{\delta} + \phi_{n+1}^{\bar{X}_n} \tag{7.25}$$

In addition, the periodic function modified divided-difference vector can be decomposed as below by separating the Fourier coefficients from the time-dependent sine and cosine functions. More precisely, the idea is to store the modified-differences vector of the sine and cosine functions separately and to multiply them by their respective Fourier coefficients:

$$\phi_n^{\bar{X}_n} = u_n \phi_n^{\bar{s}_n} + v_n \phi_n^{\bar{c}_n} \quad \text{where} \quad \begin{cases} \phi_{n,j}^{\bar{s}_n} & = \prod_{l=1}^{j} \psi_{n,l}[\sin(\omega t_n), \sin(\omega t_{n-1}), ..., \sin(\omega t_{n-j})] \\ \phi_{n,j}^{\bar{c}_n} & = \prod_{l=1}^{j} \psi_{n,l}[\cos(\omega t_n), \cos(\omega t_{n-1}), ..., \cos(\omega t_{n-j})] \end{cases} \tag{7.26}$$

and

$$\phi_{n+1}^{\bar{X}_n} = u_n \phi_{n+1}^{\bar{s}_{n+1}} + v_n \phi_{n+1}^{\bar{c}_{n+1}} \quad \text{where} \quad \begin{cases} \phi_{n+1,j}^{\bar{s}_{n+1}} & = \prod_{l=1}^{j} \psi_{n+1,l}[\sin(\omega t_{n+1}), \sin(\omega t_n), ..., \sin(\omega t_{n+1-j})] \\ \phi_{n+1,j}^{\bar{c}_{n+1}} & = \prod_{l=1}^{j} \psi_{n+1,l}[\cos(\omega t_{n+1}), \cos(\omega t_n), ..., \cos(\omega t_{n+1-j})] \end{cases} \tag{7.27}$$

From this decomposition, applying the change of Fourier coefficients is straightforward. However, this approach is not valid anymore if we consider a varying angular frequency, i.e. ω_n instead of ω.

Finally, our strategy for integrating the SPM methodology into SUNDIALS IDA algorithm is simply to perform the classical operations of IDA (interpolating polynomial construction, prediction, correction, step size adjustment, ...) on the correction term δ instead of the global solution X and then to deduce the global solution X from this correction term and the current periodic function $\bar{X}_n$.

7.2.2 Prediction step

Thanks to the previously presented continuity condition (7.19), the consistency of the interpolation polynomial $\hat{\delta}_n$ with the local DAE problem Φ_n can be ensured. In Nordsieck's formalism, this interpolating polynomial just corresponds to the Taylor development of the solution:

$$\hat{\delta}_n(t) = \delta_n + (t - t_n)\dot{\delta}_n + \frac{(t - t_n)^2}{2}\ddot{\delta}_n + ... + \frac{(t - t_n)^q}{q!}\delta_n^{(q)} \tag{7.28}$$

In IDASPM, the interpolating polynomial on δ is based on the divided-difference storage and is expressed as:

$$\hat{\delta}_n(t) = \delta_n + (t - t_n)[\delta_n, \delta_{n-1}] + ... + (t - t_n)(t - t_{n-1})[\delta_n, \delta_{n-1}, \delta_{n-2}] + ... \tag{7.29}$$

$$+ (t - t_n)(t - t_{n-1})...(t - t_{n-q+1})[\delta_n, \delta_{n-1}, \delta_{n-2}, ..., \delta_{n-q}] \tag{7.30}$$

whose j-th term is equal to

$$\hat{\delta}_{n,j}(t) = \left[\prod_{l=0}^{j-1}(t - t_{n-l})\right][\delta_n, \delta_{n-1}, \delta_{n-2}, ..., \delta_{n-j}] \tag{7.31}$$

By using the previously defined quantities in the modified divided-differences formalism, this term can be rewritten as:

$$\hat{\delta}_{n,j}(t) = \left[\prod_{l=0}^{j-1}\frac{t - t_{n-l}}{\psi_{n,l+1}}\right]\phi^{\delta}_{n,j} \tag{7.32}$$

Then, the prediction consists in extrapolating the interpolation polynomial based on the data available until time t_n to the time t_{n+1}. In other words, the predictions of δ_{n+1} and $\dot{\delta}_{n+1}$ are computed as below:

$$\hat{\delta}_{n+1} = \hat{\delta}_n(t_{n+1}) = \sum_{j=0}^{q}\phi^{\delta*}_{n+1,j} \quad \text{with} \quad \phi^{\delta*}_{n+1,j} = \beta_{n+1,j+1}\phi^{\delta}_{n,j} \tag{7.33}$$

$$\dot{\hat{\delta}}_{n+1} = \dot{\hat{\delta}}_n(t_{n+1}) = \sum_{j=0}^{q}\gamma_{n+1,j+1}\phi^{\delta*}_{n+1,j} \tag{7.34}$$

where

$$\beta_{n+1,j} = \begin{cases} 1 & \text{if } \ j = 1 \\ \prod_{l=1}^{j}\dfrac{\psi_{n+1,l}}{\psi_{n,l}} & \text{else} \end{cases} \tag{7.35}$$

and

$$\gamma_{n+1,j} = \begin{cases} 0 & \text{if } \ j = 1 \\ \dfrac{1}{h_n} & \text{if } \ j = 2 \\ \prod_{l=1}^{j-1}\dfrac{1}{\psi_{n+1,l}} & \text{else} \end{cases} \tag{7.36}$$

Finally, the prediction for the global solution is deduced from the SPM decomposition:

$$\hat{X}_{n+1} = \hat{\delta}_{n+1} + \bar{X}_n(t_{n+1}) \tag{7.37}$$

$$\dot{\hat{X}}_{n+1} = \dot{\hat{\delta}}_{n+1} + \dot{\bar{X}}_n(t_{n+1}) \tag{7.38}$$

7.2.3 Correction step

The correction step consists in solving the previously exposed fixed-point problem, e.g. with a Newton type algorithm. The classical associated residual function is defined by

$$G(X_{n+1}) = F(t_{n+1}, X_{n+1}, \alpha_n X_{n+1} + \beta) \tag{7.39}$$

with

$$\alpha_n = -\frac{\alpha_s}{h_n} \quad \text{where} \quad \alpha_s = -\sum_{j=1}^{q} \frac{1}{j} \tag{7.40}$$

and

$$\beta = \dot{\hat{X}}_{n+1} - \alpha_n \hat{X}_{n+1} \tag{7.41}$$

Its Jacobian matrix is given by

$$J_G(X_{n+1}) = \partial_X F(t_{n+1}, X_{n+1}, \alpha_n X_{n+1} + \beta_n) + \alpha_n \partial_{\dot{X}} F(t_{n+1}, X_{n+1}, \alpha_n X_{n+1} + \beta_n) \tag{7.42}$$

Therefore, in IDASPM, this residual function is equal to

$$\Gamma_n(X_{n+1}) = \Phi_n(t_{n+1}, \delta_{n+1}, \alpha_n \delta_{n+1} + \beta_n^\delta) \tag{7.43}$$

$$= F(t_{n+1}, \delta_{n+1} + \bar{X}_n(t_{n+1}), \alpha_n \delta_{n+1} + \beta_n^\delta + \dot{\bar{X}}_n(t_{n+1})) \tag{7.44}$$

with

$$\beta_n^\delta = \dot{\hat{\delta}}_{n+1} - \alpha_n \hat{\delta}_{n+1} \tag{7.45}$$

And its Jacobian matrix is given by

$$J_{\Gamma_n}(X_{n+1}) = \partial_X F(t_{n+1}, \delta_{n+1} + \bar{X}_n(t_{n+1}), \alpha_n \delta_{n+1} + \beta_n^\delta + \dot{\bar{X}}_n(t_{n+1})) \tag{7.46}$$

$$+ \alpha_n \partial_{\dot{X}} F(t_{n+1}, \delta_{n+1} + \bar{X}_n(t_{n+1}), \alpha_n \delta_{n+1} + \beta_n^\delta + \dot{\bar{X}}_n(t_{n+1})) \tag{7.47}$$

The previously computed prediction is then used as starting value for the fixed-point algorithm in order to accelerate its convergence, especially as a sufficiently good approximation is necessary for the Newton algorithm to converge. In addition, an other particularity of IDA is to use a quasi-Newton algorithm in order to lower the number of Jacobian evaluations. This strategy aims at finding a compromise between the computational cost of the Jacobian evaluation and the convergence speed. Hence, the Jacobian matrix is updated if

- The ratio between the current and the previous α coefficients is out of a fixed interval. As we generally fix the maximum order to 2, this criterion is equivalent to monitoring the step size variation ratio in our case. By default, this condition is: $\frac{\alpha_{\text{current}}}{\alpha_{\text{last update}}} \notin \left[\frac{3}{5}, \frac{5}{3}\right]$.

- The convergence failed non-fatally with the old Jacobian J, for instance if the residual function norm is greater than the solver tolerance when the maximum number of iterations is reached which can be due to a small convergence rate. Indeed, the idea of such a criterion is to avoid that the used Jacobian data lead to an unreasonable number of Newton iterations.

In output of Newton algorithm, we obtain the corrected solution X_{n+1} which enables to evaluate the prediction error $\Delta_{n+1} = X_{n+1} - \hat{X}_{n+1} = \delta_{n+1} - \hat{\delta}_{n+1}$. From this quantity, the history vector can be updated from the formula:

$$\phi^{\delta}_{n+1,q} = \phi^{\delta*}_{n+1,q} + \Delta_{n+1} \tag{7.48}$$

$$\phi^{\delta}_{n+1,j} = \phi^{\delta*}_{n+1,j} + \phi^{\delta}_{n+1,j+1} \quad \text{for} \quad j = q-1, ..., 0 \tag{7.49}$$

The global solution history is then deduced from $\phi^{X}_{n+1} = \phi^{\delta}_{n+1} + \phi^{\bar{X}_n}_{n+1}$.

7.2.4 Step size adjustment

In the adaptive step size q-order BDF, the Local Truncation Error can be roughly estimated by

$$e_{n+1} = \left[\alpha_{n,q+1} + \alpha_s - \alpha^0_n\right] \phi_{n+1,q+1} + \mathcal{O}(h^{q+2}_n) \tag{7.50}$$

$$\approx \left[\alpha_{n,q+1} + \alpha_s - \alpha^0_n\right] \Delta_{n+1} \tag{7.51}$$

The step size is accepted by IDA if

$$\max\left\{\alpha_{n,q+1}, |\alpha_{n,q+1} + \alpha_s - \alpha^0_n|\right\} ||\Delta_{n+1}|| \leq 1 \tag{7.52}$$

with

$$\alpha^0_n = -\sum_{j=1}^{q} \alpha_{n,j} \tag{7.53}$$

The step size is then updated by multiplying it by the following step size ratio:

$$r = \frac{h_{new}}{h_{old}} = \frac{1}{(2E_{n,k})^{1/q+1}} \tag{7.54}$$

where $E_{n,q}$ is the error estimate and is given by

$$E_{n,q} = \sigma_{n,q+1}||\phi_{n+1,q+1}|| = \sigma_{n,q+1}||\Delta_{n+1}|| \tag{7.55}$$

The coefficient $\sigma_{n,q+1}$ is defined as

$$\sigma_{n,q+1} = \frac{h^{q+1}_n q!}{\prod_{l=1}^{q+1} \psi_{n,l}} \tag{7.56}$$

However, in some cases, the step size adjustment strategy may be more sophisticated.

7.2.5 Fourier coefficients estimator

Once the integration step has been performed, the Fourier coefficients of the SPM periodic part can be updated. In our current implementation, this update is done using the final estimator presented in the end of chapter 5 (section 5.4). Let us recall its basic principle and the resulting linear system.

The Fourier coefficients are expressed in an incremental form, i.e. $u_{n+1} = u_n + \Delta^u_{n+1}$ and $v_{n+1} = v_n + \Delta^v_{n+1}$, where Δ^u_{n+1} and Δ^v_{n+1} are the updates to compute with the estimator. As previously exposed, these updates could be computed from any optimizer but, in the current version of the method, they are computed using a linearization approach around a prior steady-state trajectory. Finally, our estimation process consists in applying an iteration of the modified Newton algorithm for minimizing a measurement of the stationarity of the system, i.e. the objective function is

$$R(u,v) = \int_{t_{n+1}}^{t_{n+1}+T} \left\| F\left(t, \begin{bmatrix} \bar{X}_s(t) \\ X_{ns,n+1} \end{bmatrix}, \begin{bmatrix} \dot{\bar{X}}_s(t) \\ 0 \end{bmatrix}\right)\right\|^2 dt$$

where $\bar{X}_s(t) = u\sin(\omega t) + v\cos(\omega t)$ so that the computed Fourier coefficients (u_{n+1}, v_{n+1}) minimize the above objective function. In this equation, $R : \mathbf{R}^{2d_s} \to \mathbf{R}$. We recall that this estimator leads to the resolution of the following linear system:

$$\begin{bmatrix} H^{uu}_n & H^{uv}_n \\ H^{vu}_n & H^{vv}_n \end{bmatrix} \begin{bmatrix} \Delta^u_{n+1} \\ \Delta^v_{n+1} \end{bmatrix} = - \begin{bmatrix} g^u_n \\ g^v_n \end{bmatrix}$$

where the sub-matrices $H^{uu}_n, H^{uv}_n, H^{vu}_n, H^{vv}_n \in \mathbf{R}^{d_s \times d_s}$ correspond to an approximation of the components of the above objective-function Hessian matrix and the right-hand-side vector composed of $g^u_n, g^v_n \in \mathbf{R}^{d_s}$ corresponds to its gradient. In consequence, these components require the evaluation of the DAE system residual function, i.e. F, and its partial-derivatives, i.e. $\partial_X F$ and $\partial_{\dot{X}} F$. However, as IDA uses the Jacobian matrix only for the correction step, the evaluation function computes the entire Jacobian $J_G = \partial_X F + c_j \partial_{\dot{X}} F$ and so these two partial-derivatives are not directly accessible. To tackle this limitation which leads to multiple computationally expensive Jacobian matrix evaluations, we implemented an extended Jacobian evaluation function which returns the full Jacobian matrix and the two partial-derivatives in a single call. This optimization is even more efficient that, at the modeler-level in RTE's simulation engine, the two partial-derivatives are computed separately and then added to compute the entire Jacobian matrix. In order to lower the number of calls to this costly function, we also chose to update the partial-derivatives used for the estimator only when the main IDA integration algorithm updates its Jacobian matrix. This point is discussed in more

details in the next section.

In the SPM specific module, the estimator is implemented as follows:

Algorithm 2: IDASPM ESTIMATOR ALGORITHM

Non-oscillating components evaluation: $X_{ns} \leftarrow X_{ns,n+1}$ and $\dot{X}_{ns} \leftarrow 0$;

for $i = 0$, $i \leq K$, $i++$ *(loop on the time-sample)* **do**

- Time-sample: $t \leftarrow t_{n+1} + i\frac{T}{K}$ where $T = \frac{2\pi}{\omega}$;
- Weight for the quadrature rule: $w \leftarrow \frac{T}{2K}\left(1 + \mathbf{1}_{i=0|i=K}\right)$;
- Oscillating components evaluation: $X_s \leftarrow u_n \sin(\omega t) + v_n \cos(\omega t)$ and $\dot{X}_s \leftarrow \omega(u_n \cos(\omega t) - v_n \sin(\omega t))$;
- Residual function evaluation: $F \leftarrow F(t, X, \dot{X})$;
- Elementary matrices computation: $M_u \leftarrow \sin(\omega t)\partial_{X_s} F + \omega \cos(\omega t)\partial_{\dot{X}_s} F$ and $M_v \leftarrow \cos(\omega t)\partial_{X_s} F - \omega \sin(\omega t)\partial_{\dot{X}_s} F$;
- Estimator linear system components update: $H_{uu} \leftarrow H_{uu} + wM_u^T M_u$; $H_{uv} \leftarrow H_{uv} + wM_u^T M_v$; $H_{vu} \leftarrow H_{vu} + wM_v^T M_u$; $H_{vv} \leftarrow H_{vv} + wM_v^T M_v$; $g_u \leftarrow g_u + wM_u^T F$; $g_v \leftarrow g_v + wM_v^T F$;

Linear system resolution (matrix factorization only if the Jacobian has been updated by IDA): $\begin{bmatrix} \Delta_{n+1}^u \\ \Delta_{n+1}^v \end{bmatrix} = - \begin{bmatrix} H_{uu} & H_{uv} \\ H_{vu} & H_{vv} \end{bmatrix}^{-1} \begin{bmatrix} g_u \\ g_v \end{bmatrix}$

Fourier coefficients update application: $\begin{bmatrix} u_{n+1} \\ v_{n+1} \end{bmatrix} = \begin{bmatrix} u_n \\ v_n \end{bmatrix} + \begin{bmatrix} \Delta_{n+1}^u \\ \Delta_{n+1}^v \end{bmatrix}$

7.2.6 IDASPM algorithm

Algorithm 3: IDASPM ALGORITHM

initialization: get the user context and the initial condition $(t_0, X_0, \dot{X}_0)$;
if *first step* **then**
    check the input data, adjust initial step size h_0 ;
    initialize the history array: $\phi_{0,1} = h_0\dot{X}_0$ and $\phi^{\delta}_{0,1} = h_0\dot{\delta}_0$ with $\dot{\delta}_0 = \dot{X}_0 - \dot{\bar{X}}_0(t_0)$;
else
    tests: zero-crossing and stop ;
    while *final time not reached* **do**
        if *too much time steps performed* **then** exit;
        if *too much accuracy requested* **then** exit;
        update Fourier coefficients (u_n, v_n) ;
        while *step size not validated* **do**
            compute the coefficients of the method ψ_{n+1}, α_n, α_s, α_0, β_{n+1}, γ_{n+1}, σ_{n+1} ;
            update the history array $\phi^{\delta *}_{n+1}$ and ϕ^{*}_{n+1} ;
            set $t_{n+1} = t_n + h_n$;
            prediction: $\hat{\delta}_{n+1}$ and $\dot{\hat{\delta}}_{n+1}$;
            global solution deduction: $\hat{X}_{n+1} = \hat{\delta}_{n+1} + \bar{X}_n(t_{n+1})$ and
            $\dot{\hat{X}}_{n+1} = \dot{\hat{\delta}}_{n+1} + \dot{\bar{X}}_n(t_{n+1})$;
            compute the initial residual $F(t_n, \hat{X}_{n+1}, \dot{\hat{X}}_{n+1})$;
            initialize the data for Newton's algorithm: $m = 0$ and $e_{n+1(0)} = 0$;
            while *Newton's algorithm not converged* **do**
                compute the correction: $\epsilon_{n+1(m)} = J_G(X_{n+1(m)})^{-1} G(X_{n+1(m)})$;
                apply the correction: $X_{n+1(m+1)} = X_{n+1(m)} - \epsilon_{n+1(m)}$ and
                $\dot{X}_{n+1(m+1)} = \dot{X}_{n+1(m)} - \alpha_n \epsilon_{n+1(m)}$;
                compute the total correction: $e_{n+1(m+1)} = e_{n+1(m)} - \epsilon_{n+1(m)}$;
                compute the residual norm and test the convergence ;
                $m = m + 1$;
            if *step size not accepted* **then**
                restore the data: t_n, ψ_n, ϕ_n and ϕ^{δ}_n ;
                reduce the step size ;
            else
                exit the step size acceptability loop ;
        update the step size, update the data ϕ^{δ}_{n+1} and deduce ϕ_{n+1};

7.3 Implementation

After having presented the modifications made into the IDA underlying integration scheme and their effect on its algorithm, this section focuses on the resulting implementation. The IDASPM architecture and the main data-structures are first introduced and, then, its impact on the code modules is more detailed. To finish, we present a typical program skeleton for explaining how to use IDASPM as a solver from the user point-of-view.

7.3.1 Global structure overview

The figure 7.2 gives an IDASPM structure overview. To summarize, the main IDA solver is implemented within the `ida` module which contains the integration algorithm (prediction, correction, step size adjustment, etc) and the solver data are contained in a central data-structure which is defined in the `ida_impl` file. At the main solver level, IDA can call different modules, e.g. for the initial conditions calculation (`ida_ic`) and the input/output interface `ida_io`. Hence, at this general algorithm level, our implementation has mainly consisted in applying the integration algorithm on the correction term δ instead of the global solution X.

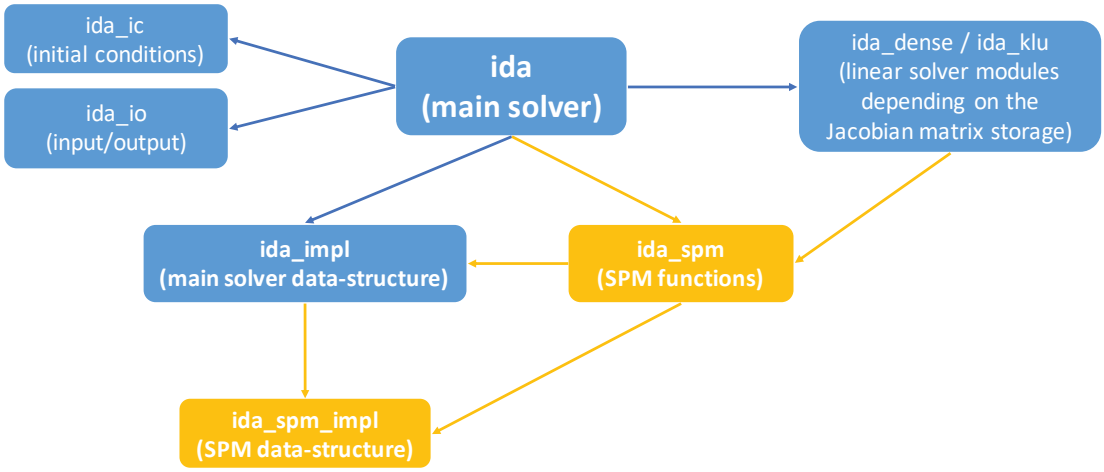

Figure 7.2: General structure of IDASPM. The blue and yellow boxes respectively correspond to the modules that are already present in the standard implementation of IDA and those that have been particularly implemented for the SPM. However, some modifications have also been made in the initially present modules for instance for applying the IDA integration scheme on the equations rewritten on the correction term.

One of IDA's greatest features is to propose a wide variety of linear solvers for solving the linear systems that arise in Newton-Raphson iterations. In particular, IDA contains

an implementation of the KLU solver for sparse matrices, which is particularly suited for power systems applications [11]. The interface between the general IDA algorithm and the linear solvers is illustrated in the figure 7.3.

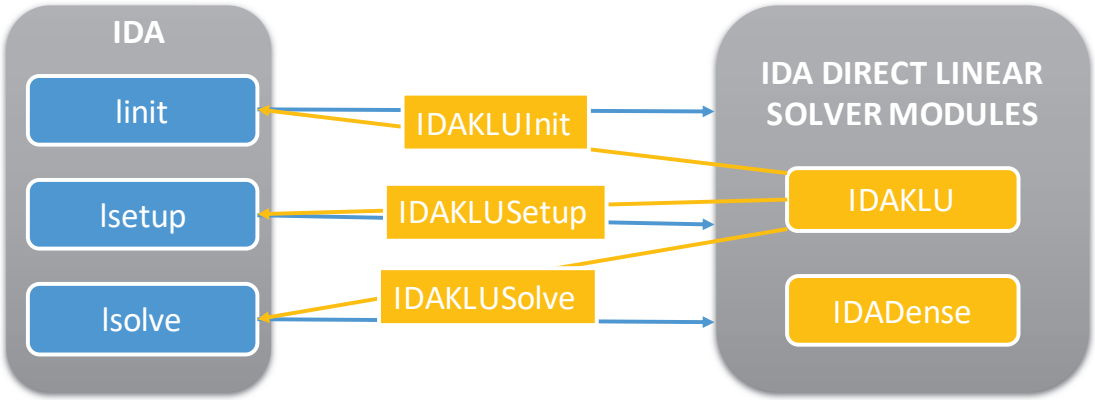

Figure 7.3: General structure of the IDA interface with the linear solvers. As IDASPM requires to access to the Jacobian matrix for the Fourier coefficients estimation step, its implementation uses either the DENSE or the KLU module. For power systems applications and especially in RTE's simulation engine, the KLU is preferred.

Hence, in the correction step, the general solver algorithm executes the Newton-Raphson iterations by calling generic functions for building and solving the linear system: initialization, setup and solve. The idea is to store a pointer to the linear solver data and to call virtual functions corresponding to the elementary operations. Then, from the appropriate linear solver module, a cast is done for converting this generic pointer to the chosen linear solver format and specifying these functions (e.g. Jacobian memory allocation, evaluation, linear system factorization for the direct linear solvers and linear system resolution). That is why the solver also communicates with specific modules corresponding to the different usable solvers, e.g. `ida_dense` and `ida_klu`. For instance, in the direct solvers, the linear system setup evaluates the Jacobian matrix and performs its factorization, and the solve function applies the matrix numerical factorization on the input right-hand-side vectors. An other important point to note is that these operations depend on the Jacobian matrix storage format: for instance, a Compressed-Sparse-Row storage is used for the KLU solver. Hence, for implementing the SPM into IDA, modifications have been made at the general algorithm level and at the matrix-specific level for taking into account these two architecture levels of IDA and the specificity of the matrix storage formats.

Moreover, two additional modules have been implemented: `ida_spm` and `ida_spm_impl`.

The former, `ida_spm`, contains the SPM-specific functions such as the estimator which can then be called by IDA at the general solver level. The latter, `ida_spm_impl`, contains the main SPM data-structure in which the SPM data are stored, e.g. the oscillating components index, the system frequency, the Fourier coefficients, etc. A pointer to this data-structure has been added to the main solver data-structure implemented in `ida_impl` for making the interface and enabling the interactions between the standard and SPM-specific functions and data. This point is presented in more details in the next subsection.

Finally, the most challenging task has been to implement the Fourier coefficients estimator, which is located in the SPM specific module `ida_spm` as mentioned above. Indeed, as the estimator uses the DAE system Jacobian matrix, its implementation depends on the matrix storage format. That's why it has been coded for the dense and KLU solvers but, as this function has to be called from the general integration algorithm, this latter actually calls a pointer to a generic function for performing the estimation which is then specified, as it is done for the linear solver operations. In particular, as it requires to solve a linear system based on the partial-derivatives of the DAE residual function, the interface with the appropriate linear algebra module is necessary. In our case, as we use a direct linear solver, we already knew that the linear solver module would automatically store the Jacobian matrix in order to perform the factorization. Then, the storage format was either dense or Compressed-Sparse-Row (CSR) leading respectively to the LU or KLU factorization algorithm.

7.3.2 Main data-structures

As mentioned above, we added a pointer to the SPM data-structure (`void *spm_mem`) to the main solver data-structure `IDAMem`, which contains:

- Problem specification data: DAE residual function, user data, ...
- Solver storage data for the current iteration and the history: divided-differences array, associated minor arrays for the previously introduced scalars and the `N_Vectors` for the solution, its time-derivative, the Newton correction, the vector distinguishing the differential and algebraic variables, ...
- Step data: order, step size, discretization coefficients, ...
- Linear solver functions prototypes (initialization, setup, resolution, performances measurement and memory deallocation) and pointer to the appropriate data-structure (which depends on the Jacobian matrix storage format and of the chosen linear solver).
- The variables for computing consistent initial conditions.
- Several counters for checking solver performances.

- Root-finding data for zero-crossing functions, etc.

Hence, `*spm_mem` is a generic pointer to a SPM-specific data structure (`SPMMem`) containing all the data needed for the SPM, i.e.

- Additional problem specifications such as the number of oscillating components and the system nominal frequency;
- The SPM solver storage for the iteration and history: divided-differences arrays for the correction term, the periodic function and the sine/cosine functions, `N_Vectors` for the correction term, the periodic part and their time-derivatives, for the Fourier coefficients and for the vector distinguishing oscillating and non-oscillating components.
- Estimator data: `N_Vectors` for the sampled state, its time-derivative, the residual function and the objective-function gradient. All these data do not depend on the Jacobian matrix storage. In addition, we introduced generic pointers for taking into account the dense or KLU cases: one to the matrix data-structure `*est_mat_mem` and another to the estimator update function `void (*idaspmupdatepriormodelparameters)(IDAMem IDA_mem)`.

To initialize this data-structure, the user only has to properly set the above-mentioned boolean vector and pulsation, which can be done from a configuration file, as it will be discussed in the next chapter. In addition, as for any IDA application, the user has to choose the linear solver to use (dense or KLU).

7.3.3 Brief description of the IDA files impacted by the implementation of the SPM

In this section, the modifications made for implementing the SPM into the IDA code are presented in more details. These modifications impacted the following modules:

- `include/ida/ida.h` and `src/ida/ida_io.c`: declaration and definition of input/output functions
- `include/ida/ida_spm.h`: introduction of initialization functions headers to be used at user-level. In particular, `int IDASPMKLUFromFile(void *ida_mem, FILE *fileptr, int nnz, int sparsetype)` is defined. This function initializes the SPM-specific data-structure of the IDA memory block (`void *ida_mem`) from the pointer to a configuration file (`FILE *fileptr`). The two other parameters (`int nnz` and `int sparsetype`) correspond to KLU-specific data.
- `src/ida/ida_impl.h`: introduction of the pointer to the SPM data-structure `void *spm_mem` into the IDA main data-structure `typedef struct IDAMemRec *IDAMem`.
- `src/ida/ida_spm_impl.h`: SPM main file which contains the definition of the SPM-

specific data-structure and functions

- `typedef struct SPMMemRec *SPMMem`: previously presented data structure containing all the SPM-specific data.
- `typedef struct EstMatKLUMemRec *EstMatKLUMem`: data structure containing the estimator data for the CSR format with KLU solver. It contains sparse matrices corresponding to the partial derivatives of the DAE system ($\partial_X F$ and $\partial_{\dot{X}} F$) and work-space for the computations (M_n^u, M_n^v for building H_n). To finish, it contains KLU data used for the solving the associated linear system.
- `void IDASPMEvaluateybarkd(IDAMem IDA_mem, realtype tt, int kd, N_Vector ybarkd)`: evaluates the periodic function `kd`-th time derivative $\bar{X}_n^{(k)}(t)$ at time `tt`.
- `void IDASPMEvaluatePriorModel(IDAMem IDA_mem, realtype tt)` and `void IDASPMEvaluatePriorModelphi(IDAMem IDA_mem)`: respectively evaluates the periodic function (and its time-derivative) and its modified-divided differences vector.
- `void IDASPMEvaluateSolution(IDAMem IDA_mem)` and `void IDASPMEvaluateSolutionphi(IDAMem IDA_mem)`: respectively deduces the whole solution (and its time-derivative from those of the periodic function and the correction term) and its modified-differences vector (from those of the periodic function and the correction term).
- `void IDASPMContinuityCondition(IDAMem IDA_mem)` and `void IDASPMContinuityConditionphi(IDAMem IDA_mem)`: respectively applies the continuity condition to the correction term and its time-derivative (from the whole solution, the correction term and their respective time-derivative) and to its modified divided-difference vector (from those of the entire solution and the periodic function).
- `int IDASPMInitSharedFromFile(IDAMem IDA_mem, SPMMem spm_mem, FILE *fileptr)`: initializes the SPM data which do not depend on the Jacobian format from the given configuration file pointer.
- `int IDASPMInitEstMemKLU(SPMMem spm_mem, int nnz, int sparsetype)`: initializes the estimator data of the SPM which is associated to the KLU solver (i.e. CSR matrix storage format).
- `void IDASPMUpdatePriorModelParameters(IDAMem IDA_mem)`: update the periodic function by estimating the Fourier coefficients. It actually uses the `void (*idaspmupdatepriormodelparameters)(IDAMem IDA_mem)` pointer to call the appropriate estimation function, depending on the Jacobian storage format and solver.

  - `void IDASPMUpdatePriorModelParametersKLU(IDAMem IDA_mem)`: specification of the above-mentioned function which estimates the Fourier coefficients by solving the associated linear system with the KLU solver.
- `src/ida/ida_spm.c`: definition of the above-mentioned functions.
- `src/ida/ida.c`: modifications of several IDA functions
  - `void *IDACreate(void)`: initialization the SPM data-structure in addition.
  - `static int IDAStep(IDAMem IDA_mem)`: introduction of the different operations corresponding to the SPM: Fourier coefficients estimation and loading, continuity condition, integration of the local DAE on the correction term and global solution and history update.
  - `static void IDASetCoeffs(IDAMem IDA_mem, realtype *ck)`: builds the interpolation polynomial corresponding to the predictor for the correction term δ instead of the global solution X.
  - `static int IDANls(IDAMem IDA_mem)`: application of the variable step size predictor-corrector BDF on the correction term instead of the entire solution.
  - `static void IDAPredict(IDAMem IDA_mem)`: prediction step on the correction term $\hat{\delta}_{n+1}^{-}$ and deduction of the prediction of the entire solution $\hat{X}_{n+1}$.
  - `static int IDANewtonIter(IDAMem IDA_mem)`: correction step on the correction term for computing δ_{n+1}^{-} and deduction of the correction of the entire solution X_{n+1}.
  - `static int IDATestError(IDAMem IDA_mem, realtype ck, realtype *err_k, realtype *err_km1)`: local truncation error test on the correction term instead of the global solution for updating the step size.
  - `static void IDARestore(IDAMem IDA_mem, realtype saved_t)`: restores the correction term modified-difference array for deleting the effect of `IDASetCoeffs` above-mentioned.
  - `static void IDACompleteStep(IDAMem IDA_mem, realtype err_k, realtype err_km1)`: updates the step size and order when the resulting local truncation error on the correction term passes the error test. In addition, it prepares the data for the next time step.
  - `int IDAGetSolution(void *ida_mem, realtype t, N_Vector yret, N_Vector ypret)`: computes the whole solution for use at the user-level from the modified divided-differences arrays of the correction term and the periodic function.
- `src/ida/ida_klu.c`: modification of `static int IDAKLUSetup(IDAMem IDA_mem, N_Vector yyp, N_Vector ypp, N_Vector rrp, N_Vector tmp1, N_Vector tmp2, N_Vector tmp3)` for computing the partial-derivatives of the

DAE system residual function in addition to the entire Jacobian. In particular, it calls an SPM-specific function (`jaceval_expanded(tn, cj, yyp, ypp, rrp, JacMat, est_matklu_mem->JFnts, est_matklu_mem->JFntsdot, jacdata, tmp1, tmp2, tmp3)`) which enables to computes these three matrices (J = `JacMat`, $\partial_X F$ = est_matklu_mem->JFnts and $\partial_{\dot{X}}$ = est_matklu_mem->JFntsdot) in a single call. Then, it required to define a special Jacobian matrix evaluation function prototype for extending the initial one (`jaceval(tn, cj, yyp, ypp, rrp, JacMat, jacdata, tmp1, tmp2, tmp3)`).

- `src/ida/CMakeLists.txt`: addition of the `ida_spm_impl.h` file in the installation directory `include/ida` for use in user-level applications.

7.3.4 Program skeleton for using IDASPM

For using IDASPM, the classical procedure of IDA [35] is extended with some specific operations:

1. Classical IDA parameters: problem dimension `d`, initial time `t0`, final time `tfinal`, DAE residual evaluation function `evalF`, DAE Jacobian evaluation function `evalJ`, vector distinguishing differential and algebraic variables `Id`, initial guess for the consistent initial conditions `X0` and `X0d`, maximum step size `hmax`, initial step size `h0`, relative tolerance relAcc and vector of tolerances `vAcc`
2. Specific IDASPM parameters (within a text file): the system pulsation `omega`, vector distinguishing the solution oscillating and non-oscillating components `Is` and DAE extended Jacobian function `evalJExtended`
3. Creation of the IDA data-structure: `IDASPMMem = IDACreate()`
4. Initialization of the IDA solver for providing the problem properties (DAE residual function, initial time and approximation of consistent initial conditions): `IDAInit(IDASPMMem, evalF, t0, X0, X0d)`
5. Set the solver tolerance: `IDASVtolerances(IDASPMMem, relAcc, vAcc)`.
6. Set the solver parameters: `IDASet*(IDASPMMem, *)` (`UserData` for the user data, `StopTime` for the final time `tfinal`, `MaxStep` for the maximum step size `hmax`, `InitStep` for the initial step size `h0`, `MaxOrd` for the maximum integration scheme order which is equal to two in our applications, `Id` for the vector distinguishing the differential and algebraic variables, ...).
7. Attach the appropriate linear solver, e.g. `IDAKLU(IDASPMMem, d, d*d, CSR_MAT)`;
8. Provide the SPM parameters: `IDASPMKLUFromFile(IDASPMMem, initSPM_fileptr, d*d, CSR_MAT)` where `initSPM_fileptr` is the pointer to the SPM configuration file in output to `fopen("initSPM.txt", "r")`.
9. Provide the Jacobian matrix evaluation function and its ex-

tended version: `IDASlsSetSparseJacFn(IDASPMMem, evalJ)` and `IDASlsSetSparseJacFnExtended(IDASPMMem, evalJExtended)` where `evalJ` computes $J = \partial_X F + c_j \partial_{\dot{X}} F$ and `evalJExtended` also returns $\partial_X F$ and $\partial_{\dot{X}} F$, which is due to the use of the SPM.

10. Refine the initial conditions: `IDACalcIC(IDASPMMem, IDA_YA_YDP_INIT, hIC)` where `hIC` is the step size used for discretizing the time-derivative $\dot{X}_0$. The output consistent initial conditions can then be extracted with `IDAGetConsistentIC(IDASPMMem, X0, X0d)`.
11. Time loop: for each time step from `t0` to `tfinal`, the function `IDASolve(IDASPMMem, tout, &tret, Xret, Xdret, IDA_ONE_STEP)` is called. `tout` is the last output point (thus beginning at `t0`) and `tret` is the target next time (which can be adjusted by IDA, that's why its pointer is given in input). `Xret` and `Xdret` are respectively the output state and its time-derivative at time `tret`.
12. Free memory: `IDAFree(&IDASPMMem)` for the solver and `N_VDestroy(...)` for the vectors.

7.4 Optimization

With the initial IDASPM implementation, gains of performances did not come up to our expectations because of its too important computational time by iteration. Finally, there was no speed up compared to a classical integration with IDA while the number of iterations was drastically reduced. For instance, on an IEEE 14-bus inspired test case, using IDASPM led to a greater computational time while the number of iterations was divided by 400. Moreover, on a Simple Electrical Grid (SEG) test case, the speedup was significant but could be enhanced. The table 7.1 presents some results for the initial performances of IDASPM after its interfacing with RTE's simulation engine. These two test cases (Simple Electrical Grid and IEEE 14-bus) are further developed in the chapter 9 which contains the results obtained with our implementation.

tol	Method	SEG	IEEE 14-bus
1.e-4	IDA	62822 it. (2.47 sec)	48504 (3.27 sec)
	IDASPM	157 it. (0.37 sec)	174 (12.03 sec)
	IDA/IDASPM	**400 (6.67)**	**279 (0.27)**
1.e-5	IDA	112026 it. (5.26 sec)	116705 (7.90 sec)
	IDASPM	290 it. (0.43 sec)	306 (13.76 sec)
	IDA/IDASPM	**386 (12.23)**	**381 (0.57)**

Table 7.1: Performances obtained with the initial version of IDASPM

If we consider the computational time by iteration, we can see that it was

- for SEG, 2.4 ms/it. with tol.=1.e-4 and 1.5 ms/it. with tol.=1.e-5;
- for IEEE, 69 ms/it. with tol.=1.e-4 and 45 ms/it. with tol.=1.e-5.

So, our implementation was very sensitive to the problem dimension. One can note that the computational time is lower when reducing the tolerance on average. This is due to the IDA management of Jacobian updates. Indeed, while the number of iterations is almost twice when passing from tol=1.e-4 to tol=1.e-5, there are only 2 additional Jacobian updates. Then, we used KCacheGrind [72] to profile our implementation performances and it revealed that two functions almost covered the totality of the solving process:

1. The linear solver setup function, which performs the Jacobian evaluation and factorization. The setup function represented 24.64% of the global computational cost. In particular, the relative cost of the Jacobian evaluation was 23.02%, i.e. about 93.43% of the setup.
2. The Fourier coefficient update function represented 75.29% of the global computational cost. In particular the relative cost of the matrix factorization, that is made for solving the estimator linear system, was 67.20%, i.e. 89.25% of the estimator cost.

Therefore these results made obvious that, for optimizing the computational cost by iteration and so the global performances of IDASPM, we had to reduce:

- the computational cost of the estimator linear system construction and resolution;
- the number of calls to the Jacobian matrix evaluation function.

By applying these different optimizations to IDASPM, the computational time by iteration has been drastically reduced, which has enabled to get much more important speed-ups with the current version of our implementation. The chapter 9 presents the results obtained with our industrial implementation prototype and compares the non-optimized and optimized versions of IDASPM.

7.4.1 Estimator linear system construction and resolution

As previously exposed, the first optimization area aimed at reducing the computational cost due to the estimator linear system construction and resolution in IDASPM. This important computational cost was actually due to an insufficient exploitation of the DAE system Jacobian matrix sparsity in our initial implementation. Let us recall the linear system appearing in our Fourier coefficients estimator:

$$\begin{bmatrix} H_n^{uu} & H_n^{uv} \\ H_n^{vu} & H_n^{vv} \end{bmatrix} \begin{bmatrix} \Delta_{n+1}^{u} \\ \Delta_{n+1}^{v} \end{bmatrix} = - \begin{bmatrix} g_n^{u} \\ g_n^{v} \end{bmatrix}$$

where, for instance

$$H_n^{uu} = \int_{t_{n+1}}^{t_{n+1}+T} \left[\sin(\omega t)\partial_{X_s}\bar{\rho}_n(t) + \omega\cos(\omega t)\partial_{\dot{X}_s}\bar{\rho}_n(t)\right]^T \left[\sin(\omega t)\partial_{X_s}\bar{\rho}_n(t) + \omega\cos(\omega t)\partial_{\dot{X}_s}\bar{\rho}_n(t)\right] dt$$

The other sub-matrices are built in a similar way, the only difference being the scaling of the partial-derivatives of ρ (evaluated from those of F). As the DAE system Jacobian matrices are commonly very sparse in power system applications, the estimator linear system matrix is also very sparse since it is based on $\partial_{X_s}F$ and $\partial_{\dot{X}_s}F$. As a result, our objective was to change the estimator construction algorithm for better exploiting the matrices sparsity and to solve the estimator linear system resolution with an efficient sparse linear solver.

In the initial implementation, a dense matrix was filled for constructing the estimator linear system from the system sparse Jacobian matrix and so, this linear system was also built without taking into account the sparsity of $\partial_X F$ and $\partial_{\dot{X}}F$. More precisely, the construction of the sub-matrices $\left[\sin(\omega t)\partial_{X_s}\bar{\rho}_n(t) + \omega\cos(\omega t)\partial_{\dot{X}_s}\bar{\rho}_n(t)\right]$ and $\left[\cos(\omega t)\partial_{X_s}\bar{\rho}_n(t) - \omega\sin(\omega t)\partial_{\dot{X}_s}\bar{\rho}_n(t)\right]$ was performed using global loops on all the rows and columns of , i.e. $row = 1, ..., d$ and $col = 1, ..., d$, which explored all the matrices components. Then, a test was done to assess if the corresponding components $\partial_X F_{row,col}$ and $\partial_{\dot{X}} F_{row,col}$ were different from zero and, if it was the case, mathematical operations were executed. Consequently, $\mathcal{O}(d^2)$ operations were performed while the proportion of non-zero components in the Jacobian matrix may be less than 1%. For instance, in one of our test cases, the system dimension was 574, leading to 568516 loop iterations, while the number of non-zero components was inferior to 2000. That's why, in the IDASPM optimized version, the corresponding loop on the Jacobian matrix components has been rewritten in order to only explore the non-zero components of the matrices.

In addition, an important computational time has been saved by solving the estimator linear system with the KLU solver [12], which is a sparse direct solver particularly suited for power system applications, as sugested in the PEGASE report [11].

7.4.2 Reduction of the number of Jacobian evaluation

In time-domain simulations, evaluating the DAE system Jacobian matrix is generally a performances bottleneck. This is particularly true in our Modelica-language-based framework as it requires to call a modeler-level function which then calls Adept for computing it from an automatic differentiation algorithm. Thus, reducing at the most the number of calls to the Jacobian evaluation function was an important lever for

enhancing our implementation performances.

In particular, on the one hand, our Fourier coefficients estimator requires the two partial derivatives of F, $\partial_X F$ and $\partial_{\dot{X}} F$ but, on the other hand, the Jacobian matrix of the DAE system is given by the formula $J_F = \partial_X F + c_j \partial_{\dot{X}} F$ in the native version of IDA and so the partial derivatives $\partial_X F$ and $\partial_{\dot{X}} F$ are not directly accessible. As a result, the initial implementation of IDASPM called the Jacobian matrix evaluation three times: once for evaluating the solver full Jacobian which is used for the integration process (with the c_j coefficient of the integrator) and twice for evaluating the two partial derivatives (with $c_j = 0$ for the X partial derivative and with $c_j = -1$ for preparing the $\dot{X}$ partial derivative extraction). In addition, once the Jacobian evaluations had been made for the estimator, some matrix operations were required for extracting the $\dot{X}$ partial derivative. Finally, the computational cost of these operations was extremely important, especially as each Jacobian evaluation calls the above-mentioned costly modeler-level function. Calling several times the entire Jacobian evaluation function was all the more paradoxical as, at the modeler level, the two sub-matrices of the partial-derivatives are independently evaluated and then assembled by exactly using the formula $J = \partial_X F + c_j \partial_{\dot{X}} F$. Thus, our first optimization idea consisted in introducing an extended Jacobian evaluation function which directly fills these three matrices (J, ∂_X and $\partial_{\dot{X}} F$) from their pointer in a single call. The main difficulty has been to consequently propagate the corresponding function prototype and call within the modeler without interfering with the modeler operations. This point is detailed in chapter 8. But, as it has been done using an extended function prototype, our strategy enables not to impact the code and so the execution of the standard modeler and solver. By this way, an important computational time has been saved as the number of calls to the Jacobian evaluation function has been directly divided by 3.

However, even after this optimization and despite the Jacobian matrix used in the estimator is updated only when IDA updates its Jacobian matrix, we noted that there were significantly more Jacobian updates when using IDASPM in comparison with IDA. Generally, the Jacobian matrix is updated by the estimator when the Newton algorithm fails to converge, which can be due to major changes in Jacobian matrix. However, there is no reason for the Jacobian matrix of IDASPM to change more than those of IDA. The only difference is that IDASPM possibly uses a much wider range of step sizes, which can affect the discretized Jacobian, but the dynamics and so the eigenvalues of the system should not be affected by the use of the SPM. In addition, updating the Jacobian requires to perform a new factorization for the Newton iterations which is very costly.

Thus, we tuned the parameters of the quasi Newton algorithm [24] for accepting wider step size variations and so avoiding unnecessary Jacobian updates. In counterpart, we also increased the maximum number of Newton algorithm iterations. Finally, the number of Jacobian evaluations has well been significantly reduced.

8

Interface between IDASPM and RTE's simulation engine

Contents

In this chapter, we present how IDASPM has been integrated into RTE's simulation engine [25]. This work has been simplified by the simulator architecture since it clearly distinguishes modeler and solver aspects. So, in the first section, we recall the RTE's simulation engine global work-flow in order to introduce the Modelica-based modeler and the solver module particularities in next sections. A focus is given on the used approach for providing it the input data. To finish, we present the additional features that were implemented for propagating the extended Jacobian function optimization exposed in the previous chapter into the simulator and especially the modeler.

8.1 Global work-flow

In RTE's simulation engine, the idea is to have independent the modeling and solving part of the simulation. A simulation is then performed several steps, which are summarized in 8.1:

1. The model writing, which is done using C++ or Modelica [20]. Modelica is generally the preferred solution as it enables to write models in equation-based style. As the simulation engine is written in C++, Modelica models are compiled using a customized OpenModelica compiler to generate a C++ code corresponding to the

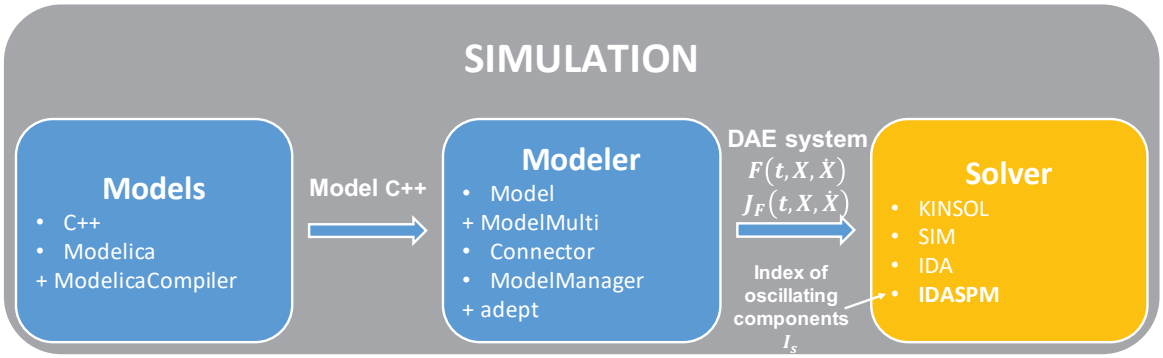

Figure 8.1: RTE's simulation engine global work-flow overview: the unit-models models are written with C++ or Modelica and given in input to the Modeler which builds the global system by assembling them using their connections. Then, the DAE system is built and solved with the chosen solver (e.g. IDASPM).

system to simulate. In addition to these unit-models, connections between them are also to provide.

2. The system assembling and building by the modeler. The objective of this module is to generate the DAE system to solve from the input models and connections. So, it contains a template class corresponding to the general definition of a model (input variables, associated DAE system and Jacobian matrix, external variables, ...), implementations of this class (vector of models, sub-model, ...) and connectors features. The Jacobian matrix of C++ models is generally provided in the associated C++ file. For Modelica models, as this Jacobian matrix is not provided, an external program called Adept [36] enables to compute it using an automatic-differentiation algorithm. In output of these different modules, the residual function of the global DAE system is generated for integration with the solver module.
3. The solver which performs the numerical integration of the DAE system in output of the modeler. It contains several solvers including IDA and IDASPM, our IDA implementation of the SPM. These modules initialize the data structure of the chosen solver from the input data provided by the user and by the modeler. In particular, it provides the DAE system written in full-implicit form, the function for evaluating the Jacobian matrix (formal or automatic-differentiation) and the vector distinguishing differential and algebraic variables. For IDASPM, additional data is provided for initializing the SPM data structure: the vector distinguishing the oscillating and non-oscillating components of the solution, and the simulated system nominal frequency. Then, consistent initial conditions for the DAE system are computed. Once the system and its initial conditions are determined, the integration is performed using the chosen solver. At each time step, RTE's simulation engine

performs tests to detect if discrete events have occurred during the last time step. Consequently, the majority of our interfacing work with RTE's simulation engine has been performed within the Solver module.

8.2 Modeler based on Modelica

OpenModelica [19] is an open-source suite based on Modelica language [20], which is used for the modeling and simulation of complex systems. Modelica is an object-oriented, formal and non-causal language for writing models in an equation-based style. Furthermore, OpenModelica contains a compiler which enables to generate simulation files, that are compatible with IDA, from a Modelica model file. In particular, Modelica enables to clearly separate the modeling and solving aspects. The European projects PEGASE [11] and then iTesla [69] have introduced and proved the industrial potential of Modelica for time-domain simulations of transmission grids. The latter resulted in the idea of developping a standard open-source Modelica library for power system analysis. Then, RTE's time-domain simulation tool team implemented a new simulation engine [25] which is based on this framework. It especially enables to easily connect models with solvers to perform simulations in a very flexible way.

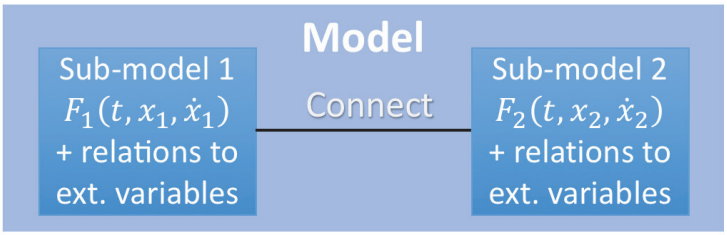

Figure 8.2: Illustration of a model written with Modelica. It consists in a global model containing two sub-models. Each model is described in a separate file and the global model specifies their connection.

In our framework, Modelica is used for the modeling part of the simulation. Our objective is to implement the simulated system model with the Modelica language and to perform the simulation with IDASPM from RTE's simulation engine. As illustrated in the figure 8.2, constructing a global model with Modelica generally consists in connecting several unit-models that are written in a equation-based style. So each sub-model contains both internal variables, equations and relations to external variables. It is thus very convenient for modeling system and performing time-domain simulations.

Once a model is written, it can be compiled for instance with OpenModelica Compiler whose compilation process is illustrated in figure 8.3. This compilation process roughly consists in constructing a global system of differential algebraic equations, referred as flat model, by injecting those of all the sub-models with their connections and then in generating an executable code in order to enable the simulation of the resulting system.

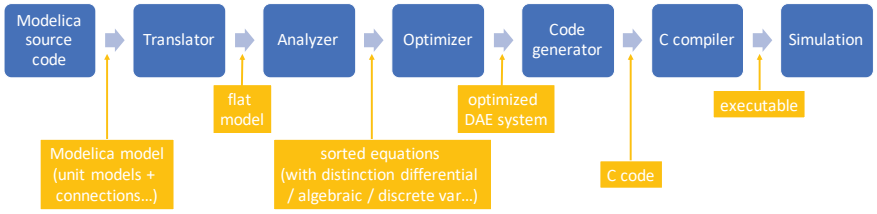

Figure 8.3: Illustration of the OpenModelica Compiler process. To sum up, the Modelica model containing the sub-models and their connections is written by the compiler which constructs a flat model in which all the equations are assembled within a large system. Then, if this option is selected, OpenModelica Compiler sorts and tries to reduce the number of equations, for instance by directly injecting the trivial relations into the appropriate equations. To finish, a C code containing the output system of equations is generated for its resolution.

8.3 Solver module

In order to access to our solver from RTE's simulation engine, we had to generate the solver library corresponding to IDASPM and then to implement a customized solver module which properly initializes the data structures of IDASPM. It is done in `SolverIDASPM::init`.

This function executes the following procedure in order to initialize IDASPM, which roughly corresponds to the program skeleton proposed in section 7.3.4:

- Creation of the solver data structure;
- Internal memory allocation for the solver data structure;
- Initialization: attaching the residual function evaluation to the data structure and giving the initial conditions;
- Solver settings: solver tolerance, stop time, minimum and maximum acceptable step size, initial step size, maximum order for the integration scheme;
- Distinction between differential and algebraic variables from a boolean vector;
- Choice of the linear solver (e.g. KLU);

- Attaching the Jacobian matrix evaluation function and, in particular for the SPM, its expanded version which also returns the partial-derivatives of the DAE system residual function;
- Providing SPM-specific input data: vector distinguishing oscillating and non-oscillating variables of the system and its nominal frequency;
- Attaching the root evaluation function;

Then, as IDASPM properly initializes the data structures associated to its standard integration algorithm and those corresponding to the SPM features, the simulation can be launched as for the classical IDA solver: computation of initial conditions and loop on the time step until the final time or the detection of a discrete-event.

So, as mentioned in the procedure description, an SPM-specific function has been implemented within the solver module for extending the Jacobian matrix evaluation function. This function enables in particular to compute the partial-derivatives of the DAE system residual function. Indeed, in IDA, the Jacobian is directly computed for taking into account the discretization of the time-derivative. Then, the Jacobian matrix used in the correction step is computed as $J_G(X) = \frac{\partial F}{\partial X} + c_j \frac{\partial F}{\partial \dot{X}}$ since the correction is defined by $G(X) = F\left(t, X, c_j X + \beta\right)$. As the estimation step requires $\frac{\partial F}{\partial X}$ and $\frac{\partial F}{\partial \dot{X}}$ in order to build the linear system associated to the correction of the Fourier coefficients, several linear algebra operations and Jacobian matrix evaluations with different parameters c_j were necessary for extracting these two matrices from the complete Jacobian matrix. This is why the `evalJtExpanded` function has been implemented at two levels: within the modeler module which performs the actual computation and at the solver model for interfacing with IDASPM.

8.4 Input data for IDASPM

As above mentioned, in order to properly initialize the periodic function of the SPM, IDASPM requires two inputs:

- The vector distinguishing the solution oscillating and non-oscillating components;
- The system nominal frequency.

These two inputs are provided to IDASPM from a configuration file which has to be inserted into the considered test case folder. Then, in addition to the `.iidm`, `.dyd`, `.crv` and `.jobs` files which are the classical input files for RTE's simulation engine, the

`initSPM.txt` is included. It simply corresponds to a vector of data defined as

$$\begin{bmatrix} \omega \\ isAC_1 \\ \vdots \\ isAC_d \end{bmatrix} \tag{8.1}$$

where $isAC \in \{0,1\}^d$ is a boolean vector indicating the oscillating and non-oscillating variables. Then, $isAC_j = 1$ if the j-th components is an oscillating component of the solution.

This configuration is currently filled by the user from its prior knowledge of the system to simulate. Generally, the assumed oscillating components of the solution are the network voltages and currents. Two enhancements of our methodology could be done:

- Automatically filling the configuration file from the modeler. For instance, the oscillating variables could be tagged in their respective Modelica model. Then, the modeler could detect this tag in order to fill either the configuration file or directly a data structure corresponding to the vector $isAC$, as it is currently done for distinguishing the differential and algebraic variables. An automatic procedure could use the same approach for distinguishing the oscillating and non-oscillating components of the solution at the modeler level.
- Deducing variables with possibly higher-order harmonics. For instance, in the model of synchronous machines, variables projected in the dq0 reference frame with the Park transformation can contain second order harmonics when the three phase abc input signals are unbalanced [14]. Then, there is both an offset component corresponding to the balanced conditions and a second-order harmonics due to this unbalance. In addition, non-linear transformations of fundamental harmonics oscillating components can lead to higher-order harmonics.

To finish, our implementation does not require the initial Fourier coefficients to be set during the initialization. They are computed directly during the simulation. This is why performances are generally lower during the first time steps, even if the system is in steady-state. Indeed, as one can see in the results presented in chapter 9, some iterations are required for the Fourier coefficients to converge and consequently for the step size to increase. On the one hand, such a result is encouraging as it shows the method robustness and the estimator accuracy. But, on the other hand, as the step size remains at low values during this initial step, this means that a possibly great computational is consumed while the system is already in steady-state. In order to compute these initial Fourier coefficients, an additional module based on KINSOL [10] could use the initial load-flow to deduce the

initial Fourier coefficients. By this way, even if this only provides a coarse estimation of the Fourier coefficients, this could enable to reduce the number of lost initial step sizes by giving a quite accurate starting point for the estimator.

8.5 Additional features due to the IDASPM `evalJtExpanded` optimization

In the previous chapter, we mentioned the optimization that we implemented consisting in computing the Jacobian matrix of the DAE system for the integration part and its partial-derivatives for the estimation part of the SPM. To do this, a non-standard evaluation function has been introduced. It contains the pointers to the sparse matrices corresponding to these three matrices instead of only those of the full Jacobian matrix. By this way, the computation is much more efficient as at low-level operations, the two partial derivatives are evaluated and then added within the full Jacobian matrix. Thus, filling the partial-derivatives theoretically does not introduce additional computational time as the data are directly accessible.

This optimization then required to propagate the addition of the `evalJExpanded` function prototype and operations at the different levels of RTE's simulation engine 8.4, in particular within the following files:

- `Solvers/SolverIDASPM/DYNSolverIDASPM.cpp`: the solver-level function `int SolverIDASPM::evalJExpanded` has been implemented. This function is the interface between the solver IDASPM and RTE's simulation engine.
- `Modeler/Common/DYNModel.h`: the template function `virtual void evalJtExpanded` has been added. This function is called within `int SolverIDASPM::evalJExpanded` with the instruction `model->evalJtExpanded`. It is the interface between the solver-level and the modeler-level of RTE's simulation engine. Then, depending on the type of model, the appropriate extended Jacobian evaluation can be called.
- `Modeler/Common/DYNModelMulti.cpp/.h`: the function `void evalJtExpanded` iterates on the different sub-models in order to call their respective extended Jacobian evaluation function.
- `Modeler/Common/DYNSubModel.cpp/.h`: the function `evalJtExpanded` computes the considered sub-model Jacobian matrix and partial-derivatives. It is a template function which enables to have different Jacobian matrix method depending on the type of sub-model. For instance, as our models are written with Modelica, this

function uses the ModelManager class.

- `Modeler/ModelManager/DYNModelManager.cpp/.h`: the function `ModelManager::evalJtExpanded` calls `ModelManager::evalJtAdeptExpanded` for computing the Jacobian matrix and the partial-derivatives with Adept [36]. Thus, `ModelManager::evalJtAdeptExpanded` is the interface between RTE's simulation engine and Adept which finally calls `adept::Stack::jacobian`.

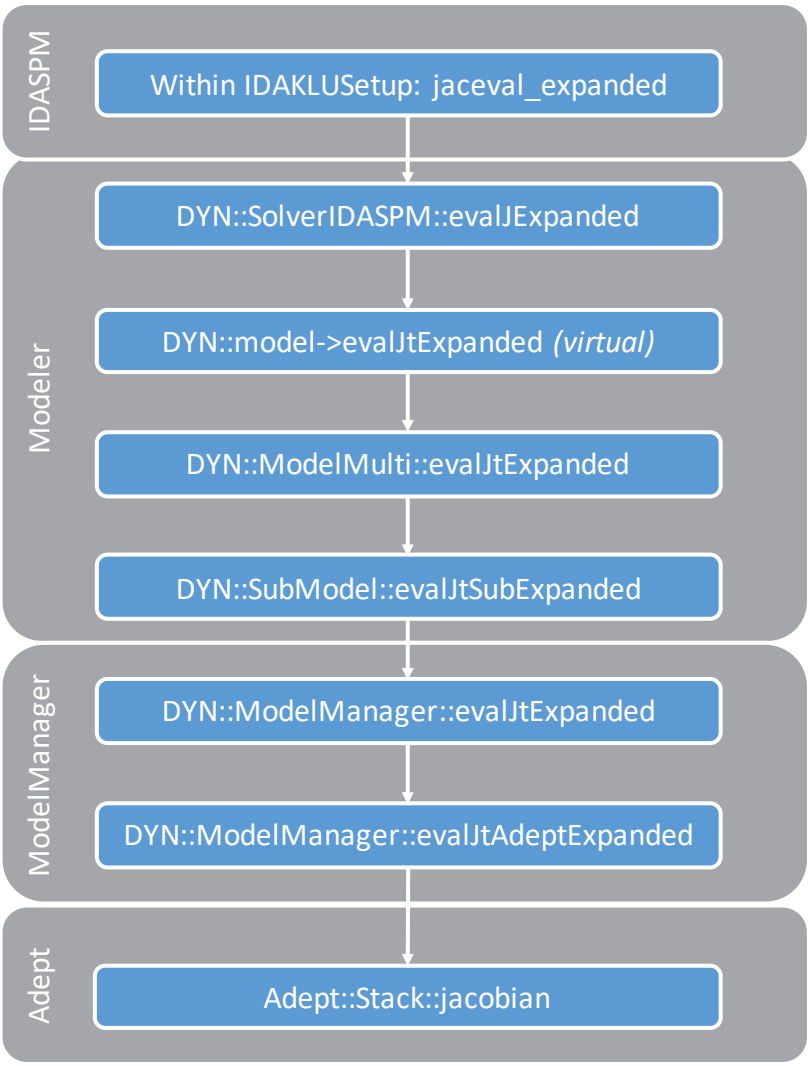

Figure 8.4: Callstack during the call to `evalJExpanded` within `IDASPM`. Modifications have been made at the `SolverIDASPM` level for calling the virtual function associated to the `model`. Then, within the `ModelMulti` class a loop is performed on the different `SubModels`. Finally, as our unit-models are fully written with Modelica, the `ModelManager` class calls `Adept` for computing the automatic-differentiation Jacobian matrix.

9

Results obtained with our implementation

Contents

In this chapter, we present the results obtained with our customized version of RTE's simulation engine using the IDASPM solver. When comparing the implementation based on IDASPM and those based on IDA, our results tend to prove that using IDASPM enables to dramatically reduce the number of iterations needed to simulate AC power systems, especially in steady-state. However, they also bring to light that a special focus is to give on the computational cost of SPM additional features and particularly those of the estimator. In our first implementation, the computational time by iteration was so important that the final speed-up did not come up to our expectations. By analyzing the code with a profiling tool, it gave us the optimization areas that have been presented within the chapter dedicated to IDASPM. These optimizations, which mainly focused on the estimator and the Jacobian matrix evaluation function, have enabled to divide the computational time by more than 10. In the end, the implementation based on IDASPM

is significantly more efficient than those based on IDA. The most notable aspect is the maximum step size reached by IDASPM in steady-state, which can be in the range of seconds while those of a classical implementation based on IDA is lower than a millisecond.

The presented results have been obtained on two simple electrical system configurations, with respectively 4 and 14 buses, mainly composed of classical linear components such as resistors, inductors and capacitors. Indeed, important issues have been encountered for implementing more realistic tests systems containing synchronous machines or grid-feedings. In particular, the initialization is really difficult within the OpenModelica environment, even more for setups containing non-linear elements. Especially, as the steady-state assumption $\dot{X}_0 = 0$ is not valid in the context of EMT simulations because of the presence of oscillating components represented in full-waveform, important pre-processing treatments should be applied in order OpenModelica for enabling to compute consistent initial conditions.

These results have been obtained from a Fedora Virtual Machine (launched from VirtualBox) running with 2 processors (CPU: Intel i5-5257U @2.7GHz, cache=3MB) and 4GB of RAM.

9.1 Results on a small power system

In this section, we present the results obtained with our implementation on a small test case consisting in a three-phase power system with 4 nodes.

9.1.1 Test case presentation

In figure 9.1, our 4-buses power system is presented.
This test case contains :

- 3 perfect generators that are modeled with a trivial algebraic equation:

$$V_\infty(t) = \begin{bmatrix} A_\infty \cos(\omega t + \phi_\infty) \\ A_\infty \cos(\omega t + \phi_\infty - \frac{2\pi}{3}) \\ A_\infty \cos(\omega t + \phi_\infty + \frac{2\pi}{3}) \end{bmatrix} \tag{9.1}$$

- 1 time-varying resistive load modeled as

$$R_c(t) = \begin{cases} R_0 & \text{if } t < t_e \\ R_\infty + (R_0 - R_\infty) e^{\lambda(t-t_e)} & \text{if } t \geq t_e \end{cases} \tag{9.2}$$

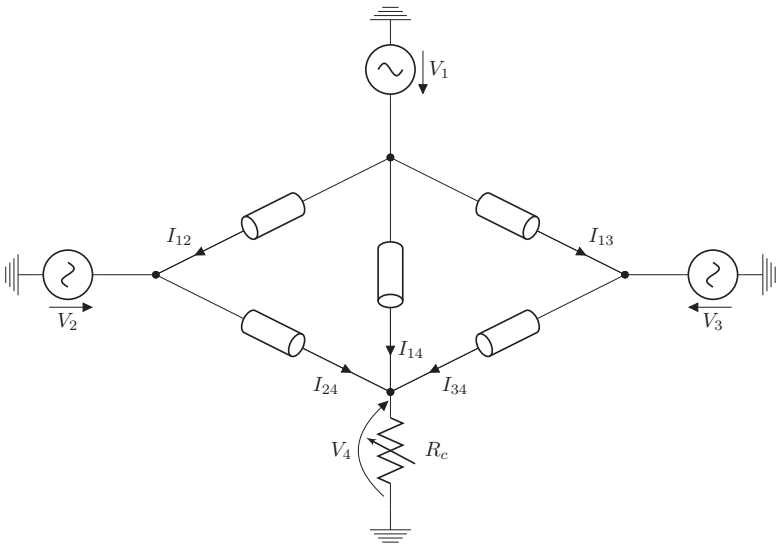

Figure 9.1: Simple Electrical Grid test case

where R_0 and R_∞ are respectively the initial and asymptotic resistance values, λ is the decay constant and t_e is the time of the discrete-event. Therefore, there are three main time intervals:

  - From the initial time t_0 to the discrete-event time t_e, the system is in steady-state as the resistance is fixed to a constant value R_0;
  - In the few times after the discrete-event t_e, the system is in transients as the resistance is subject to an exponential decay;
  - In the farther times after the discrete-event t_e, the system returns to steady-state since the resistance tends towards its asymptotic value R_∞.

- 5 transmission lines modeled as RL branches, i.e.

$$\begin{bmatrix} V_{L,a}(t) \\ V_{L,b}(t) \\ V_{L,c}(t) \end{bmatrix} = \begin{bmatrix} R_{L,a} & 0 & 0 \\ 0 & R_{L,b} & 0 \\ 0 & 0 & R_{L,c} \end{bmatrix} \begin{bmatrix} I_{L,a}(t) \\ I_{L,b}(t) \\ I_{L,c}(t) \end{bmatrix} + \begin{bmatrix} L_L & M_L & M_L \\ M_L & L_L & M_L \\ M_L & M_L & L_L \end{bmatrix} \begin{bmatrix} \dot{I}_{L,a}(t) \\ \dot{I}_{L,b}(t) \\ \dot{I}_{L,c}(t) \end{bmatrix} \tag{9.3}$$

  where $I_L = (I_{L,a}, I_{L,b}, I_{L,c})$ is the three-phase current of the transmission line, $V_L = (V_{L,a}, V_{L,b}, V_{L,c})$ is the difference between the line input and output three-phase voltages, $R_{L,a}$, $R_{L,c}$ and $R_{L,c}$ constitutes the line resistance matrix, L_L and M_L are respectively the line self- and mutual-inductance.

In our framework, this model results in a system of 188 equations including 15 differential equations. The simulation is performed from the initial time $t_0 = 0s$ to the final time $t_{final} = 10s$ with an exponential decay of the electrical load at time $t_{event} = 1s$. The solution in the neighborhood of t_e is shown in figure 9.2.

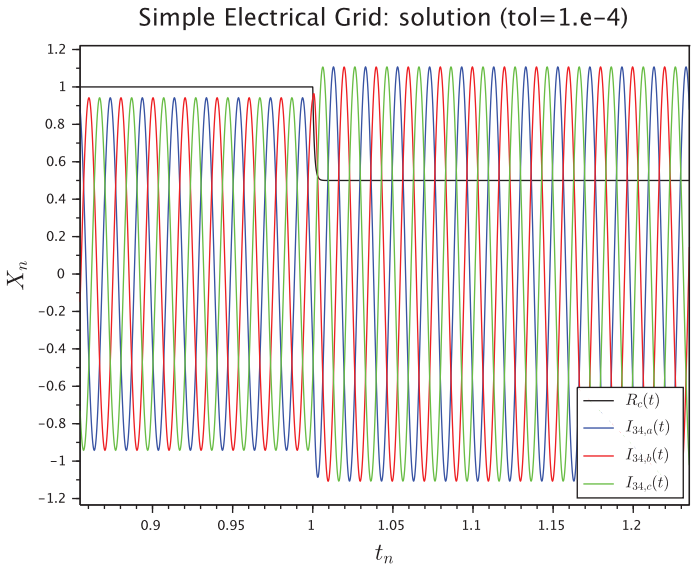

Figure 9.2: Solution computed with IDA (tolerance = 1.e-4) in the neighborhood of time $t_e = 1s$ (on the abscissa: time in seconds, on the ordinate: solution in arbitrary units). The black line corresponds to the time-varying resistance which is subject to an exponential decay. The blue, red and green lines correspond to the three-phase current through the line connecting the bus 3 to the bus 4. The transient resulting from the resistive load variation is a change of the three-phase current envelope.

9.1.2 Numerical results

In this section, we present the validation results for the optimized version of IDASPM and compare the numerical performances of the optimized and non-optimized versions of IDASPM with IDA.

9.1.2.1 Validation

The figures 9.3 and 9.4 show the solution computed with IDA and with the non-optimized and optimized versions of IDASPM. In both figures, the blue line and the red crosses respectively correspond to the solution computed with IDA and with IDASPM (optimized or non-optimized). Only one phase among the three-phase signal is shown for a better readability. Since the discrete event happens at time $t_e = 1s$, our figures focus on the neighborhood of that particular time in order to illustrate the step size adaptation. We can see that IDASPM has the expected behavior as the distance between the computed points varies as wanted: in the neighborhood of the discrete event, the time steps become closer in order to accurately catch the envelope variation and then larger when the system

returns to steady-state. For both versions of IDASPM, the solution accurately fits those computed with the IDA standard implementation.

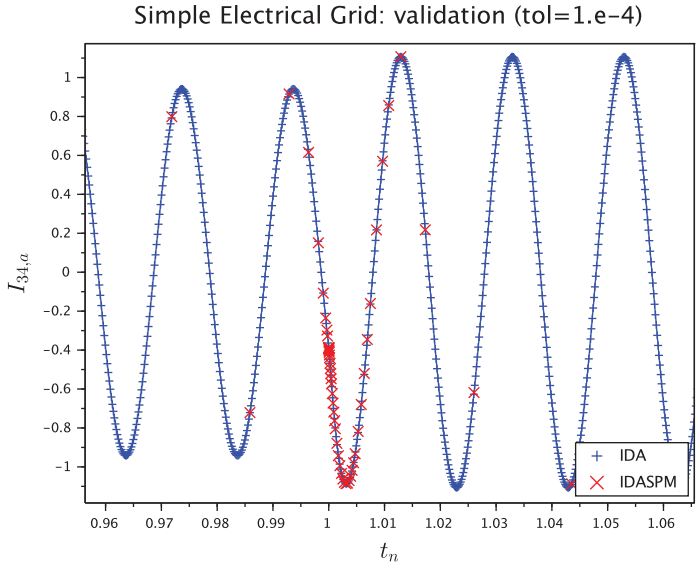

Figure 9.3: Comparison of the solutions obtained with IDA (blue linked crosses) and the non-optimized version of IDASPM (red crosses) on the Simple Electrical Grid test case with the tolerance set to tol = 1.e-4 (on the abscissa: time in seconds, on the ordinate: solution in arbitrary units).

9.1.2.2 Numerical performances

In figure 9.6, the step sizes used by IDA and IDASPM are compared.
We can see that the SPM globally enables to integrate the DAE with much larger time steps. As expected, the step size used with the SPM reaches high values when the system is in steady-state. Hence, as previously mentioned, one can identify three time intervals:

1. From the beginning of the simulation to the event, the step size increases regularly. At the very beginning, the SPM uses small time steps as the Fourier coefficients are initialized to zero. Then, some iterations are required for them to converge.
2. During the load variation, the step size remains at low values. As the estimator is designed from the assumption that the system is in steady-state, this step size limitation is logical.
3. When the system returns to steady-state, the step size increases again and reaches high values.

Table 9.1 summarizes the performances obtained with IDA and with the two versions of IDASPM (optimized and non-optimized) using tolerances from 1.e-3 to 1.e-6. We can see

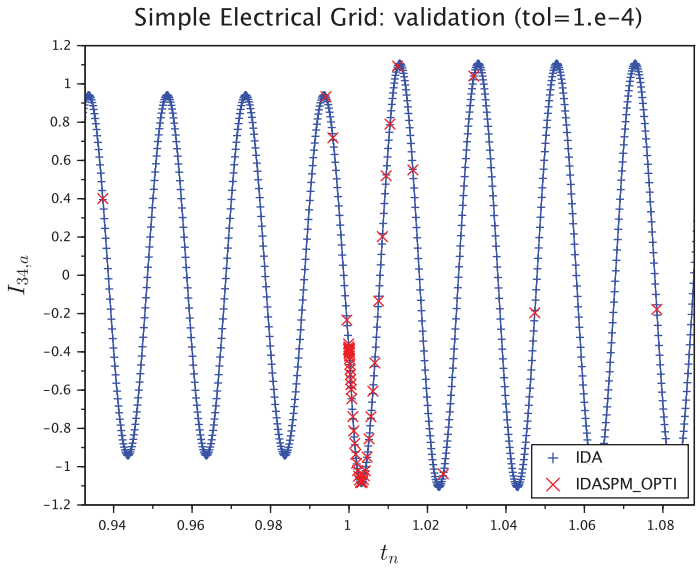

Figure 9.4: Comparison of the solutions obtained with IDA (blue linked crosses) and the optimized version of IDASPM (red crosses) on the Simple Electrical Grid test case with the tolerance set to tol = 1.e-4 (on the abscissa: time in seconds, on the ordinate: solution in arbitrary units).

that, for all the tested tolerances, significant speed-up are obtained using both versions of IDASPM. However, the optimized version of IDASPM requires almost two times more iterations for performing the simulation than the non-optimized version. This increase is logical since one of the implementation optimizations of IDASPM consists in reducing the number of Jacobian evaluations by tuning the parameters of the Jacobian update rule. In spite of this number of iterations increase, the final speed-up is three to four times higher with the optimized versions of IDASPM as the average computational time by iteration is about five times lower in the optimized version of IDASPM. In addition, we can note that the speed-up drastically increases with the reduction of the tolerance which is explained by the fact that the number of iterations using IDA is much higher while those using IDASPM remains low. Actually, it seems like the increase factors are quite similar between all the solvers but, as the ranges of the starting value are radically different (about 25000 for IDA compared to about 100-150 for IDASPM), the final results completely differ. To finish, an important point to notice is that the maximum step size used in IDASPM is not sensitive to the change of tolerances, which can also be seen in figure 9.7.

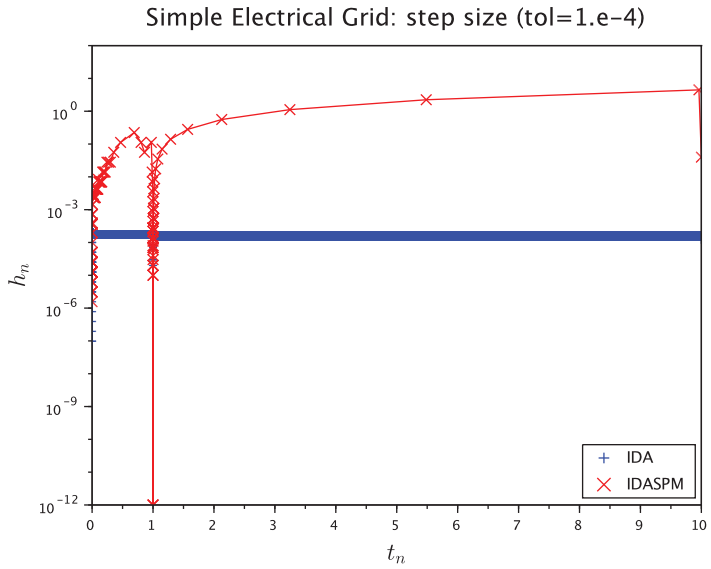

Figure 9.5: Comparison on the step size used by IDA (blue line) and the non-optimized version of IDASPM (red line) on the Simple Electrical Grid test case with with the tolerance set to $tol = 1.e-4$ (on the abscissa: time in seconds, on the ordinate: step size in seconds).

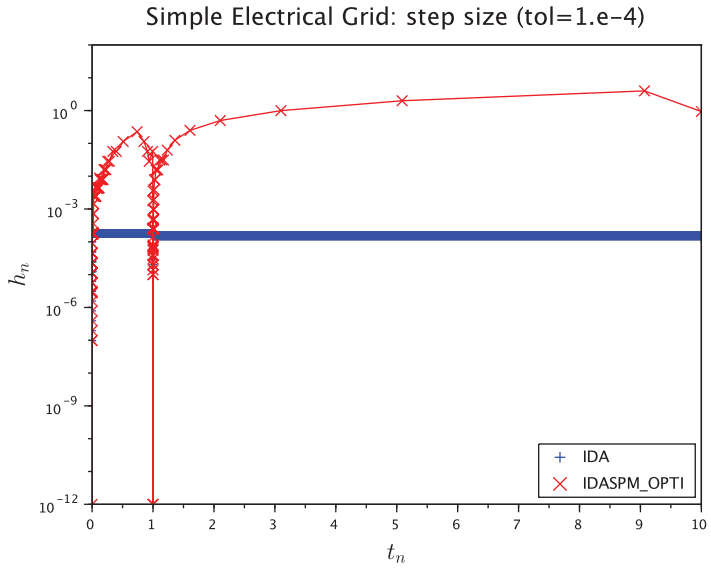

Figure 9.6: Comparison on the step size used by IDA (blue line) and the optimized version of IDASPM (red line) on the Simple Electrical Grid test case with the tolerance set to $tol = 1.e-4$ (on the abscissa: time in seconds, on the ordinate: step size in seconds).

Tolerance	Solver	$N_{iterations}$	t_{CPU}	h_{mean}	h_{max}
1.e-3	IDA	24707	1.08	0.0004047	0.0005
	IDASPM	87	0.23	0.1149425	3.39813
	IDASPM_OPTI	142	0.08	0.0704225	3.12262
1.e-4	IDA	62822	2.47	0.0001592	0.0002001
	IDASPM	157	0.37	0.0636943	4.47579
	IDASPM_OPTI	266	0.084	0.0375940	3.97862
1.e-5	IDA	112026	5.26	0.0000893	0.0001296
	IDASPM	290	0.43	0.0344828	3.22227
	IDASPM_OPTI	483	0.107	0.0207039	3.06773
1.e-6	IDA	238807	11.65	0.0000419	0.00006
	IDASPM	575	0.65	0.0173913	4.17677
	IDASPM_OPTI	998	0.16	0.0100200	3.48291

Table 9.1: Performances comparison between IDA and IDASPM (optimized and non-optimized versions) on the SEG test case

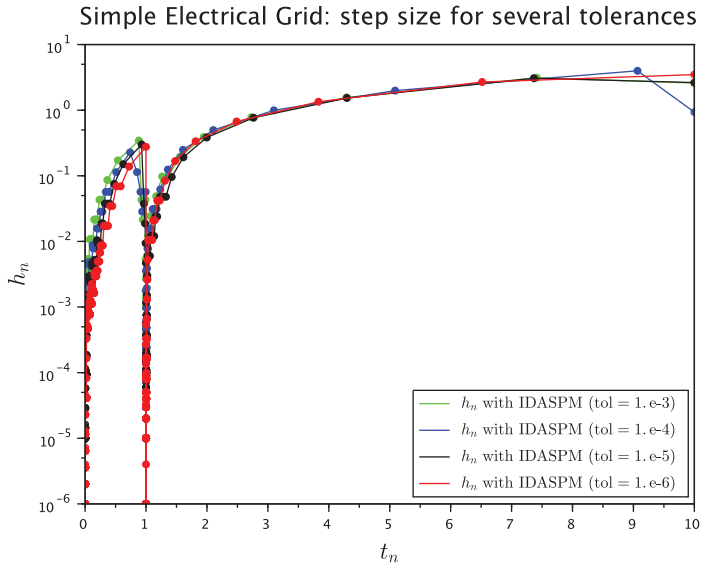

Figure 9.7: Step sizes used by the optimized version of IDASPM on the Simple Electrical Grid test case for different tolerances (red: 1.e-6, black: 1.e-5, blue: 1.e-4, green: 1.e-3). On the abscissa: time in seconds, on the ordinate: step size in seconds.

9.2 Results on a customized version of the IEEE 14-bus reference test case

After having obtained interesting results with the previous small Simple Electrical Grid example, our implementation has been tested on a larger power system with 14 nodes. The objective was then to significantly increase the considered test case dimension in order to assess the SPM scalability. As seen in chapter 7 about IDASPM, it led to the presented IDASPM optimization.

9.2.1 Test case presentation

Figure 9.8 represents the one-line diagram of our second example inspired from the IEEE-14 bus reference test case.

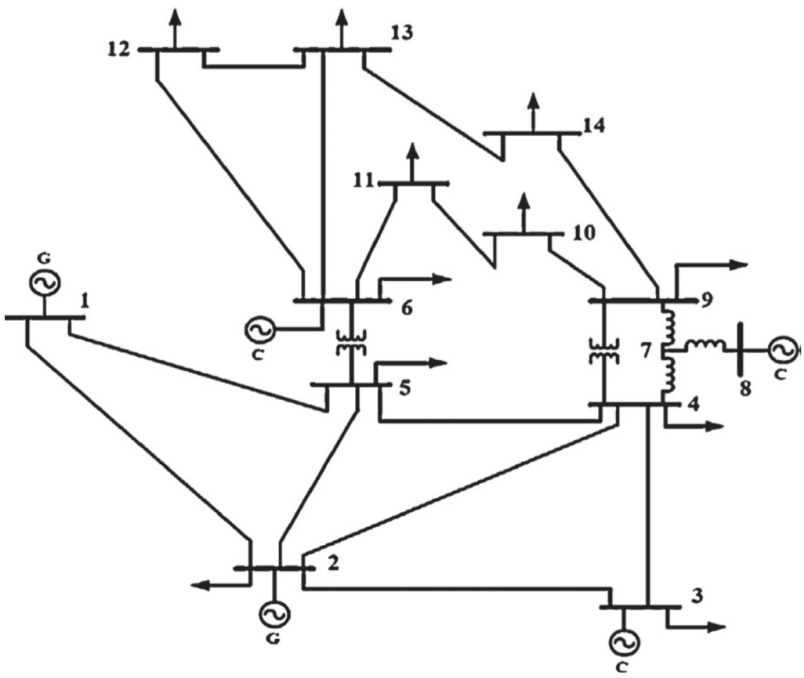

Figure 9.8: IEEE 14-bus test case (figure from [1])

However, as said in the chapter introduction, this test case has been simplified as our Modelica library did not contain models for PQ-loads, transformers and synchronous machines:

- The PQ-loads have been replaced by resistive loads;
- The transformers are considered as ideal transformers, i.e. constituted of RL-branches;

- The synchronous machines have been substituted by perfect generators. Indeed, our model of synchronous machine led to important stability issues, especially as the initialization process, which is complex to perform in a proper way in the OpenModelica environment, has not been addressed during this thesis.

Finally, it contains 5 perfect generators, 11 resistive loads, connections with transmission lines modeled with RL branches and ideal transformers. In our framework, this model results in a system of 754 equations including 51 differential equations. Therefore, in spite of the above-mentioned modeling simplifications, this test case has enabled to perform a first scalability assessment of the SPM. Then, this system is subject to an exponential increase of the electrical load attached to Bus 4 at time $t_e = 1s$. Figure 9.9 shows the three-phase current through the transmission line connecting the buses 4 and 5. As in the previous test case, the transient consists in an envelope variation.

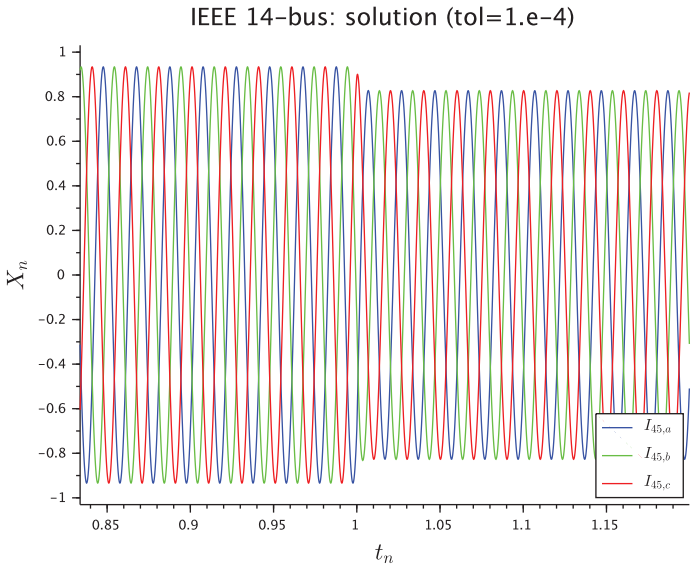

Figure 9.9: Solution computed with IDA (tolerance = 1.e-4) in the neighborhood of time $t_e = 1s$ (on the abscissa: time in seconds, on the ordinate: solution in arbitrary units). The blue, red and green lines correspond to the three phases of the current through the line connecting the bus 4 to the bus 5. As in the previous test case, the transient resulting from the variation of the resistive load is a change of the three-phase current envelope.

9.2.2 Numerical results

9.2.2.1 Validation

In figures 9.10 and 9.11, the solutions obtained with IDA and IDASPM are compared. As for the previous test case, the solution computed with IDASPM accurately fits computed with the standard implementation of IDA.

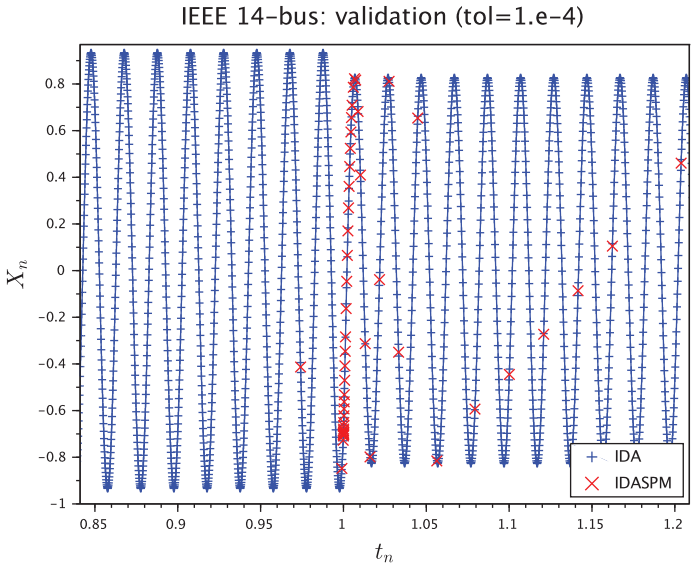

Figure 9.10: Comparison of the solutions obtained with IDA (blue linked crosses) and the non-optimized version of IDASPM (red crosses) on the IEEE 14-bus inspired test case with the tolerance set to $tol = 1.e-4$ (on the abscissa: time in seconds, on the ordinate: solution in arbitrary units).

9.2.2.2 Numerical performances

In figures 9.12 and 9.13, the step sizes used by the standard implementation of IDA and respectively the non-optimized and optimized version of our IDASPM implementation are compared. As for the previous test case, we can see that the step size varies as wanted for IDASPM since it decreases at the time of the discrete-event and increases again until reaching very high values when the system returns to steady-state.

Table 9.2 summarizes the performances obtained with the three solvers. Similar gains of performances can be noted concerning the number of iterations using both versions of IDASPM. However, with this test case, we can observe that the computational time of

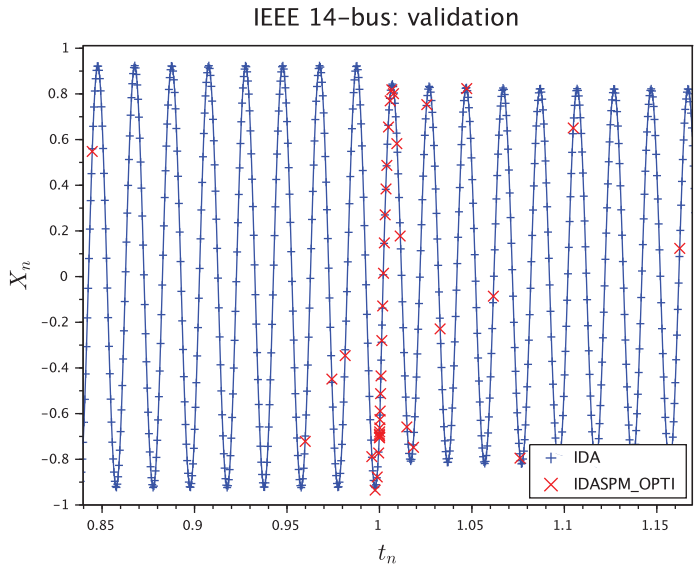

Figure 9.11: Comparison of the solutions obtained with IDA (blue linked crosses) and the optimized version of IDASPM (red crosses) on the IEEE 14-bus inspired test case with the tolerance set to $tol = 1.e-4$ (on the abscissa: time in seconds, on the ordinate: solution in arbitrary units).

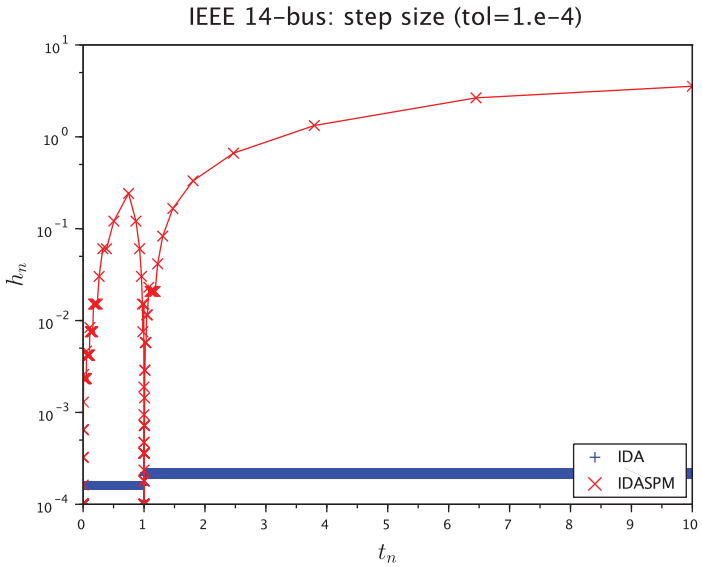

Figure 9.12: Comparison on the step size used by IDA (red line) and the non-optimized version of IDASPM (blue line) on the IEEE 14-bus inspired test case with the tolerance set to $tol = 1.e-4$ (on the abscissa: time in seconds, on the ordinate: step size in seconds).

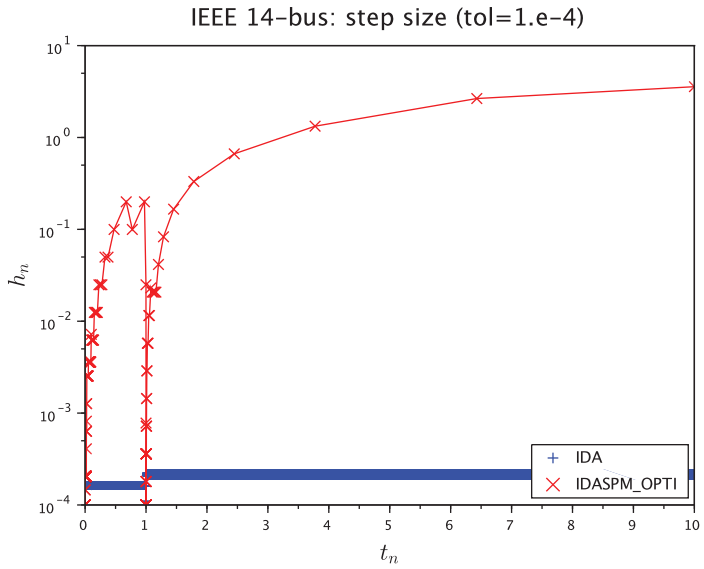

Figure 9.13: Comparison on the step size used by IDA (red line) and the optimized version of IDASPM (blue line) on the IEEE 14-bus inspired test case with the tolerance set to $tol = 1.e-4$ (on the abscissa: time in seconds, on the ordinate: step size in seconds).

the non-optimized version of IDASPM is greater than those of the standard IDA implementation. This is why the optimized version of IDASPM has been implemented. Indeed, the computational time by iteration of the non-optimized version of IDASPM was so high that it finally covers the expected gains of performances due to the reduction of the number of iterations. On the contrary, using the optimized version of IDASPM, we can see that important speed-up are obtained, especially for the least tolerance (tol=1.e-6), for which the simulation with this final implementation of our solver is about 10 times faster than with IDA. In addition, as for the previous test case, the maximum step size is not affected by the reduction of the tolerance. This can be seen in figure 9.14 which shows the evolution of the step size used by the optimized version of IDASPM with different tolerances. Therefore, low tolerances such as tol=1.e-6 could be used in order to accurately catch fast transient phenomena while being able to use large time steps when the system is in steady-state.

Tolerance	Solver	$N_{iterations}$	t_{CPU}	h_{mean}	h_{max}
1.e-3	IDA	22241	1.66	4.50e-4	5.00e-4
	IDASPM	90	8.68	1.11e-1	3.69
	IDASPM_OPTI	138	0.85	7.25e-2	3.69
1.e-4	IDA	48504	3.27	2.06e-4	2.20e-4
	IDASPM	174	12.03	5.75e-2	3.55
	IDASPM_OPTI	252	0.96	3.97e-2	3.57
1.e-5	IDA	116705	7.90	8.57e-5	9.50e-5
	IDASPM	306	13.76	3.27e-2	3.29
	IDASPM_OPTI	516	1.27	1.94e-2	4.15
1.e-6	IDA	207563	14.58	4.82e-5	5.31e-5
	IDASPM	593	22.50	1.69e-2	3.40
	IDASPM_OPTI	948	1.69	1.05e-2	3.93

Table 9.2: Performances comparison between IDA and IDASPM (optimized and non-optimized versions) on the SEG test case

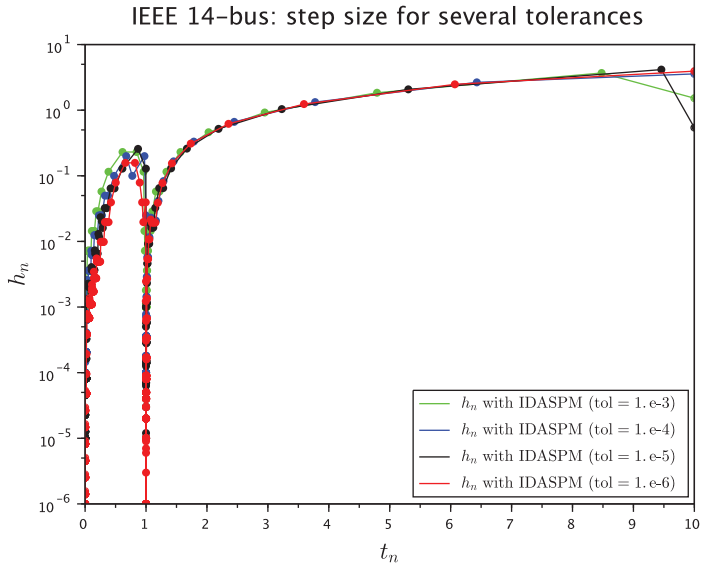

Figure 9.14: Step sizes used by the optimized version of IDASPM on the IEEE 14-bus inspired test case for different tolerances (red: 1.e-6, black: 1.e-5, blue: 1.e-4, green: 1.e-3). On the abscissa: time in seconds, on the ordinate: step size in seconds.

Conclusion

10

Conclusions and Prospects

In the next decades, mid-to-long term electromagnetic time-domain simulations of power systems will be required to ensure the transmission systems security of supply while introducing new components. The main limit to fast simulations is due to the presence of oscillating components whose behavior is paradoxically trivial in steady-state as they tend toward simple sinusoids. In the currently existing numerical methods, this property is not taken into account or in a non-flexible way. Therefore, the step size is generally fixed to very small values, significantly lower than the sinusoids period. The numerical method proposed in this PhD aims at unlocking the potential of adaptive step size numerical methods in EMT simulations context. Such a scheme enables both to reduce the time step when the system is in transients for catching fast phenomena arising in EMT simulations and to increase it when the system is in steady-state in order to reduce simulation's computational cost.

The developed Sinusoidal Predictor Method, which consists in splitting the solution into a periodic function completed by a correction term, combines a parametric estimator for the periodic function with a classical adaptive step size scheme for integrating the correction term. Thanks to the SPM theoretical analysis, we have shown the importance of the estimator choice. In addition to its bias direct effect on the maximum usable step size, its indirect impact on the entire solving process stability has been highlighted. In a first implementation, this stability issue resulted in a step size limitation, particularly in steady-state when the method should be able to use very large time steps. Finally, with our predictive Fourier coefficients estimator based on the minimization of an energy function associated to the system stationarity, Fourier coefficients are accurately estimated in steady-state. By this way, the correction term is reduced to a constant term, i.e their asymptotic value for the non-oscillating components and zero for the oscillating components of the solution. Thus, very high step sizes can be used, even for low tolerances on the local truncation error. In addition, this leads to an important integration method flexibility as a very wide range of step sizes is used. It can especially vary from the micro-second to a few tens of seconds.

Moreover, an implementation of this methodology within a framework distinguishing modeler and solver aspects has been achieved. To do so, the reference solver SUNDIALS IDA has been firstly modified to introduce features corresponding to the SPM. Then, this solver has been successfully interfaced with RTE's simulation engine, enabling to implement models with Modelica. Finally, using this implementation, tests have been conducted on examples inspired by reference IEEE test cases. In particular, RTE's simulation engine with IDASPM has been tested on a simplified version of the IEEE 14-buses reference test case. The results firstly obtained on this test case enabled to highlight several optimization areas for our implementation concerning the DAE system Jacobian matrix evaluation and our estimator. By implementing these different optimizations, important speedups have been obtained. For instance, the computational time has been divided by 10 for the IEEE 14-buses inspired test case with a tolerance of 10^{-6}.

In further developments, methodological and industrial aspects could be enhanced.

Concerning the methodology, the SPM could be extended for taking into account multiple harmonics, as it is done within dynamic phasor approaches, and variations of the system frequency. When considering test cases involving synchronous machines, the reference system frequency may be significantly different from its nominal value. However, such an extension should be studied in the angle of a compromise between the iterations number and their individual cost. Indeed, using an optimizer to solve the arising fitting problem could be costly and lead to reduce the overall performances. In addition, if the frequency is reasonably different from its fixed assumed value, the resulting residual oscillations should have a low frequency and so the step size should keep quite high values.

For the industrial aspects, several tasks should be done to manage to prove the method potential: continuing the solver implementation optimization in order to increase the speed up and testing the method on industrial test cases containing realistic power systems components such as synchronous machines, etc. Concerning the former point, the estimator implementation could still be enhanced by optimizing the linear system construction. For the latter point, implementing a reference library for EMT applications with OpenModelica would be necessary. In addition, new dedicated features should be implemented within OpenModelica in order to solve the initialization issues due to this environment current rigidness for power system applications.

Bibliography

[1] S. Abbas Taher, H. Mahmoodi, and H. Aghaamouei. Optimal pmu location in power systems using mica. *Alexandria Engineering Journal*, 2015.

[2] R. Adapa and J. Reeve. A new approach to dynamic analysis of ac networks incorporating detailed modeling of dc systems. ii. application to interaction of dc and weak ac systems. *IEEE transactions on power delivery*, 3(4):2012–2019, 1988.

[3] G. Anderson, N. Watson, N. Arnold, and J. Arrillaga. A new hybrid algorithm for analysis of hvdc and facts systems. In *Energy Management and Power Delivery, 1995. Proceedings of EMPD'95., 1995 International Conference on*, volume 2, pages 462–467. IEEE, 1995.

[4] J. Astic, A. Bihain, and M. Jerosolimski. The mixed adams-bdf variable step size algorithm to simulate transient and long term phenomena in power systems. *IEEE Transactions on Power Systems*, 9(2):929–935, 1994.

[5] R. Barrio and S. Serrano. Performance of perturbation methods on orbit prediction. *Mathematical and Computer Modelling*, 48(3-4):594–600, 2008.

[6] R. K. Brayton, F. G. Gustavson, and G. D. Hachtel. A new efficient algorithm for solving differential-algebraic systems using implicit backward differentiation formulas. *Proceedings of the IEEE*, 60(1):98–108, 1972.

[7] T. Cadeau and F. Magoulès. Coupling parareal and waveform relaxation methods for power systems. In *Electrical and Control Engineering (ICECE), 2011 International Conference on*, pages 2947–2950. IEEE, 2011.

[8] J. Chavez, A. Ramirez, and V. Dinavahi. Dynamic harmonic domain modelling of synchronous machine and transmission line interface. *IET generation, transmission & distribution*, 5(9):912–920, 2011.

[9] J. J. Chavez and A. Ramirez. Dynamic harmonic domain modeling of transients in three-phase transmission lines. *IEEE Transactions on Power Delivery*, 23(4):2294–2301, 2008.

[10] A. M. Collier, A. C. Hindmarsh, R. Serban, and C. S. Woodward. User documentation for kinsol v2. 6.0. *Center for Applied Scientific Computing, Lawrence Livermore National Laboratory*, 2009.

[11] CRSA and RTE. Pegase d4.1: Algorithmic requirements for simulation of large network extreme scenarios. Technical report, Technical Report [Online]. Available: http://www. fp7-pegase. eu/download.html.

[12] T. A. Davis and E. Palamadai Natarajan. Algorithm 907: Klu, a direct sparse solver for circuit simulation problems. *ACM Transactions on Mathematical Software (TOMS)*, 37(3):36, 2010.

[13] T. Demiray. *Simulation of power system dynamics using dynamic phasor models*. PhD thesis, ETH Zurich, 2008.

[14] T. Demiray and G. Andersson. Comparison of the efficiency of dynamic phasor models derived from abc and dqo reference frame in power system dynamic simulations. In *7th IET International Conference on Advances in Power System Control, Operation and Management (APSCOM 2006)*. IET, 2006.

[15] T. Demiray, G. Andersson, and L. Busarello. Evaluation study for the simulation of power system transients using dynamic phasor models. In *Transmission and Distribution Conference and Exposition: Latin America, 2008 IEEE/PES*, pages 1–6. IEEE, 2008.

[16] T. Demiray, F. Milano, and G. Andersson. Dynamic phasor modeling of the doubly-fed induction generator under unbalanced conditions. In *Power Tech, 2007 IEEE Lausanne*, pages 1049–1054. IEEE, 2007.

[17] H. W. Dommel. *EMTP theory book*. Microtran Power System Analysis Corporation, 1990.

[18] S. Fan and H. Ding. Time domain transformation method for accelerating emtp simulation of power system dynamics. *IEEE Transactions on Power Systems*, 27(4):1778–1787, 2012.

[19] P. Fritzson, P. Aronsson, A. Pop, H. Lundvall, K. Nystrom, L. Saldamli, D. Broman, and A. Sandholm. Openmodelica-a free open-source environment for system modeling, simulation, and teaching. In *Computer Aided Control System Design, 2006 IEEE International Conference on Control Applications, 2006 IEEE International Symposium on Intelligent Control, 2006 IEEE*, pages 1588–1595. IEEE, 2006.

[20] P. Fritzson and V. Engelson. Modelica—a unified object-oriented language for system modeling and simulation. In *European Conference on Object-Oriented Programming*, pages 67–90. Springer, 1998.

[21] T. Fukushima. Generalization of encke's method and its application to the orbital and rotational motions of celestial bodies. *The Astronomical Journal*, 112:1263, 1996.

[22] F. Gao and K. Strunz. Multi-scale simulation of multi-machine power systems. *International Journal of Electrical Power & Energy Systems*, 31(9):538–545, 2009.

[23] C. W. Gear. Unified modified divided difference implementation of adams and bdf

fomulas. Technical report, Illinois Univ., Urbana (USA). Dept. of Computer Science, 1980.

[24] M. Geradin, S. Idelsohn, and M. Hogge. Computational strategies for the solution of large nonlinear problems via quasi-newton methods. In *Computational methods in nonlinear structural and solid mechanics*, pages 73–81. Elsevier, 1981.

[25] A. Guironnet, M. Saugier, S. Petitrenaud, F. Xavier, and P. Panciatici. Towards an open-source solution using modelica for time-domain simulation of power systems. under review, 2018.

[26] A. Haddadi, J. Mahseredjian, and C. Dufour. Time warping method for accelerated emt simulations. In *2016 Power Systems Computation Conference (PSCC)*, pages 1–7, June 2016.

[27] E. Hairer, C. Lubich, and G. Wanner. *Geometric numerical integration: structure-preserving algorithms for ordinary differential equations*, volume 31. Springer Science & Business Media, 2006.

[28] E. Hairer, S. P. Nørsett, and G. Wanner. *Solving Ordinary Differential Equations I: Nonstiff Problems*. Springer-Verlag, Berlin, Heidelberg, 1993.

[29] E. Hairer and G. Wanner. *Solving Ordinary Differential Equations II: Stiff and Differential-Algebraic Problems*. Springer-Verlag, Berlin, Heidelberg, 1996.

[30] M. Hannan and K. Chan. Modern power systems transients studies using dynamic phasor models. In *Power System Technology, 2004. PowerCon 2004. 2004 International Conference on*, volume 2, pages 1469–1473. IEEE, 2004.

[31] B. Haut, V. Savcenco, and P. Panciatici. A multirate approach for time domain simulation of very large power systems. In *2012 45th Hawaii International Conference on System Sciences*, pages 2125–2132. IEEE, 2012.

[32] M. Heffernan, K. Turner, J. Arrillaga, and C. Arnold. Computation of ac-dc system disturbances-part i. interactive coordination of generator and convertor transient models. *IEEE transactions on Power Apparatus and Systems*, (11):4341–4348, 1981.

[33] A. Hindmarsh. The pvode and ida algorithms. Technical report, Technical Report UCRL-ID-141558, LLNL, 2000.

[34] A. C. Hindmarsh, P. N. Brown, K. E. Grant, S. L. Lee, R. Serban, D. E. Shumaker, and C. S. Woodward. Sundials: Suite of nonlinear and differential/algebraic equation solvers. *ACM Transactions on Mathematical Software (TOMS)*, 31(3):363–396, 2005.

[35] A. C. Hindmarsh, R. Serban, and A. Collier. User documentation for ida v2.9.0. *Center for Applied Scientific Computing Lawrence Livermore National Laboratory*, 2016.

[36] R. J. Hogan. Fast reverse-mode automatic differentiation using expression templates in c++. *ACM Transactions on Mathematical Software (TOMS)*, 40(4):26, 2014.

[37] Q. Huang and V. Vittal. Application of electromagnetic transient-transient stability hybrid simulation to fidvr study. *IEEE Transactions on Power Systems*, 31(4):2634–2646, 2016.

[38] Y. Huang, M. Chapariha, F. Therrien, J. Jatskevich, J. Martí, et al. A constant-parameter voltage-behind-reactance synchronous machine model based on shifted-frequency analysis. *IEEE Trans. Energy Convers*, 30(2):761–771, 2015.

[39] V. Jalili-Marandi, V. Dinavahi, K. Strunz, J. Martinez, and A. Ramirez. Interfacing techniques for transient stability and electromagnetic transient programs ieee task force on interfacing techniques for simulation tools. *IEEE Transactions on Power Delivery*, 24(4):2385–2395, 2009.

[40] B. Kasztenny and M. Kezunovic. A method for linking different modeling techniques for accurate and efficient simulation. *IEEE Transactions on Power Systems*, 15(1):65–72, 2000.

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

[42] J. R. Martí, H. W. Dommel, B. D. Bonatto, and A. F. Barrete. Shifted frequency analysis (sfa) concepts for emtp modelling and simulation of power system dynamics. In *Power Systems Computation Conference (PSCC), 2014*, pages 1–8. IEEE, 2014.

[43] J. R. Marti and J. Lin. Suppression of numerical oscillations in the emtp. *IEEE Power Engineering Review*, 9(5):71–72, 1989.

[44] T. Mei and J. Roychowdhury. A time-domain oscillator envelope tracking algorithm employing dual phase conditions. *IEEE transactions on computer-aided design of integrated circuits and systems*, 27(1):59–69, 2008.

[45] K. Mudunkotuwa and S. Filizadeh. Co-simulation of electrical networks by interfacing emt and dynamic-phasor simulators. *Electric Power Systems Research*, 163:423–429, 2018.

[46] A. Nordsieck. On numerical integration of ordinary differential equations. *Mathematics of Computation*, 16(77):22–49, 1962.

[47] O. Østerby. Step change strategies for multistep methods. *DAIMI Report Series*, 14(196), 1985.

[48] L. R. Petzold. An efficient numerical method for highly oscillatory ordinary differential equations. *SIAM Journal on Numerical Analysis*, 18(3):455–479, 1981.

[49] L. R. Petzold. Description of dassl: a differential/algebraic system solver. Technical report, Sandia National Labs., Livermore, CA (USA), 1982.

[50] L. R. Petzold, L. O. Jay, and J. Yen. Numerical solution of highly oscillatory ordinary differential equations. *Acta numerica*, 6:437–483, 1997.

[51] F. Plumier, P. Aristidou, C. Geuzaine, and T. Van Cutsem. Co-simulation of electromagnetic transients and phasor models: A relaxation approach. *IEEE Transactions on Power Delivery*, 31(5):2360–2369, 2016.

[52] F. J. Plumier, P. Aristidou, C. Geuzaine, and T. Van Cutsem. A relaxation scheme to combine phasor-mode and electromagnetic transients simulations. In *Power Systems Computation Conference (PSCC), 2014*, pages 1–7. IEEE, 2014.

[53] F. Pruvost, P. Laurent-Gengoux, F. Magoulés, and B. Haut. Accelerated waveform relaxation methods for power systems. In *Electrical and Control Engineering (ICECE), 2011 International Conference on*, pages 2877–2880. IEEE, 2011.

[54] J. Reeve and R. Adapa. A new approach to dynamic analysis of ac networks incorporating detailed modeling of dc systems. i. principles and implementation. *IEEE Transactions on Power Delivery*, 3(4):2005–2011, 1988.

[55] J. J. Rico, M. Madrigal, and E. Acha. Dynamic harmonic evolution using the extended harmonic domain. *IEEE Transactions on Power Delivery*, 18(2):587–594, 2003.

[56] S. R. Sanders, J. M. Noworolski, X. Z. Liu, and G. C. Verghese. Generalized averaging method for power conversion circuits. *IEEE Transactions on Power Electronics*, 6(2):251–259, 1991.

[57] T. Simos. Modified runge-kutta methods for the numerical solution of odes with oscillating solutions. *Applied mathematics and computation*, 84(2-3):131–143, 1997.

[58] G. Söderlind. Automatic control and adaptive time-stepping. *Numerical Algorithms*, 31(1-4):281–310, 2002.

[59] A. M. Stankovic and T. Aydin. Analysis of asymmetrical faults in power systems using dynamic phasors. *IEEE Transactions on Power Systems*, 15(3):1062–1068, 2000.

[60] A. M. Stankovic, S. R. Sanders, and T. Aydin. Dynamic phasors in modeling and analysis of unbalanced polyphase ac machines. *IEEE Transactions on Energy Conversion*, 17(1):107–113, 2002.

[61] K. Strunz. Flexible numerical integration for efficient representation of switching in real time electromagnetic transients simulation. *IEEE Transactions on Power Delivery*, 19(3):1276–1283, 2004.

[62] K. Strunz, R. Shintaku, and F. Gao. Frequency-adaptive network modeling for integrative simulation of natural and envelope waveforms in power systems and circuits. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 53(12):2788–2803, 2006.

[63] M. Stubbe, A. Bihain, J. Deuse, and J. Baader. Stag-a new unified software program for the study of the dynamic behaviour of electrical power systems. *IEEE*

Transactions on Power Systems, 4(1):129–138, 1989.

[64] H. Su, K. W. Chan, L. A. Snider, and J.-C. Soumagne. Advancements on the integration of electromagnetic transients simulator and transient stability simulator. In *Proceedings of International Conference on Power System Transients*, pages 1–6, 2005.

[65] TE and KTH. itesla d3.1: Part ii – limitations of current modelling approaches. Technical report, Technical Report [Online]. Available: http://www.itesla-project.eu/deliverables.html.

[66] K. Turner, M. Heffernan, C. Arnold, and J. Arrillaga. Computation of ac-dc system disturbances. pt. ii-derivation of power frequency variables from convertor transient response. *IEEE Transactions on Power Apparatus and systems*, (11):4349–4355, 1981.

[67] K. Turner, M. Heffernan, C. Arnold, and J. Arrillaga. Computation of ac-dc system disturbances. pt. iii-transient stability assessment. *IEEE Transactions on Power Apparatus and Systems*, (11):4356–4363, 1981.

[68] A. A. van der Meer, M. Gibescu, M. A. van der Meijden, W. L. Kling, and J. A. Ferreira. Advanced hybrid transient stability and emt simulation for vsc-hvdc systems. *IEEE Transactions on Power Delivery*, 30(3):1057–1066, 2015.

[69] L. Vanfretti, T. Rabuzin, M. Baudette, and M. Murad. itesla power systems library (ipsl): A modelica library for phasor time-domain simulations. *SoftwareX*, 5:84–88, 2016.

[70] C. Velez. Numerical integration of orbits in multirevolution steps. 1970.

[71] X. Wang, P. Wilson, and D. Woodford. Interfacing transient stability program to emtdc program. In *Power System Technology, 2002. Proceedings. PowerCon 2002. International Conference on*, volume 2, pages 1264–1269. IEEE, 2002.

[72] J. Weidendorfer. Sequential performance analysis with callgrind and kcachegrind. In *Tools for High Performance Computing*, pages 93–113. Springer, 2008.

[73] S. Wong, K. Sze, L. Snider, K. Chan, and C. Larose. Overcoming the difficulties associated with interfacing different simulation programs. 2003.

[74] T. J. Ypma. Historical development of the newton–raphson method. *SIAM review*, 37(4):531–551, 1995.

[75] P. Zhang, J. R. Marti, and H. W. Dommel. Synchronous machine modeling based on shifted frequency analysis. *IEEE Transactions on Power Systems*, 22(3):1139–1147, 2007.

[76] P. Zhang, J. R. Martí, and H. W. Dommel. Shifted-frequency analysis for emtp simulation of power-system dynamics. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 57(9):2564–2574, 2010.

[77] Y. Zhang, A. M. Gole, W. Wu, B. Zhang, and H. Sun. Development and analysis of applicability of a hybrid transient simulation platform combining tsa and emt elements. *IEEE Transactions on Power Systems*, 28(1):357–366, 2013.

Appendix

A Time Domain Transformation: numerical comparison with TR-BDF and the SPM

A.1 System in steady-state

Method	$N_{iterations}$	t_{CPU}	t_{CPU}/ite	h_{mean}	h_{max}
TR-BDF	2825	39.50	0.014	3.5e-4	3.8e-4
TDT	5	0.47	0.094	0.25	0.65
Ratio TR-BDF/TDT	565	84	0.15	1.4e-3	5.9e-4
SPM	4	0.94	0.23	0.33	0.65
Ratio TR-BDF/SPM	706	42	0.060	1.1e-3	5.9e-4

A.2 System in slow-evolution

Method	$N_{iterations}$	t_{CPU}	t_{CPU}/ite	h_{mean}	h_{max}
TR-BDF	14425	199.32	0.014	3.47e-4	3.8e-4
TDT	496	71.92	0.14	0.010	0.058
Ratio TR-BDF/TDT	29	3	0.095	0.034	6.6e-3
SPM	981	11.75	0.012	5.1e-3	0.092
Ratio TR-BDF/SPM	15	17	1.15	0.068	4.2e-3

A.3 System following a scenario

Method	$N_{iterations}$	t_{CPU}	t_{CPU}/ite	h_{mean}	h_{max}
TR-BDF	30446	422.66	0.014	3.28e-4	3.84e-4
TDT	1106	107.07	0.097	9.1e-3	0.65
Ratio TR-BDF/TDT	28	3.95	0.14	0.036	5.9e-4
SPM	4213	55.49	0.013	2.4e-3	0.95
Ratio TR-BDF/SPM	7	7.62	1.05	0.14	4.0e-4

Utilisation de prédicteurs sinusoïdaux pour la simulation temporelle de systèmes électriques en courant alternatif

Résumé : Simuler temporellement les réseaux électriques modernes requiert d'importants moyens de calcul de par la dimension et la raideur des systèmes différentiels algébriques résultants. De plus, la fréquence d'oscillation de certains signaux simulés contraint fortement le pas d'intégration des schémas classiques, y compris en régime établi où ils sont proches de sinusoïdes oscillant à la fréquence nominale du système. L'objectif de la méthode des prédicteurs sinusoïdaux proposée dans cette thèse est donc de tirer parti de cette propriété afin d'améliorer les performances du solveur tout en contrôlant l'erreur de calcul. Elle consiste à décomposer la solution en deux parties : une sinusoïde, dont les coefficients de Fourier sont fixés pour chaque intervalle d'intégration puis mis à jour par estimation paramétrique, et un terme de correction sur lequel le système d'EDA est reformulé et résolu à l'aide d'un schéma d'intégration à pas adaptatif. Une attention particulière a été portée au choix de l'estimateur paramétrique, ce dernier ayant un impact direct sur le pas d'intégration de par sa précision et indirect de par son effet sur la stabilité globale de la méthode. L'estimateur finalement développé consiste à calculer les coefficients de Fourier qui minimisent une mesure de la stationnarité du système. Ce dernier étant convergent en régime permanent, le terme de correction est progressivement amorti, permettant ainsi d'accroître considérablement le pas d'intégration. Cette méthode, intégrée au sein du solveur SUNDIALS IDA puis interfacée avec un moteur de calcul industriel, permet d'accélérer très nettement les simulations en comparaison avec une implémentation classique.
Mots-clés : simulation temporelle de systèmes électriques, schéma d'intégration à pas adaptatif, estimation paramétrique, stabilité numérique

---

Use of sinusoidal predictors for time domain simulation of AC power systems

Abstract: Modern power systems time-domain simulations require important computational resources due to the resulting differential algebraic systems dimension and stiffness. In addition, some simulated signals oscillation frequency dramatically limits the classical schemes step size, even in steady-state during which they are close to sinusoids oscillating at system nominal frequency. That's why the sinusoidal predictors method proposed in this thesis aims at taking this property into account in order to enhance solver performances while controlling the integration error. It consists in decomposing the solution into two parts: a sinusoid, whose Fourier coefficients are fixed for each time integration interval and then updated by parametric estimation, and a correction term on which the DAE system is rewritten and solved using an adaptive step size integration scheme. A particular focus has been given on the estimator choice, given its precision direct impact on the step size and its indirect effect on the global method stability. The finally developed estimator consists in computing Fourier coefficients minimizing a system stationarity measurement. As it converges in steady-state, the correction term is progressively damped, which enables to considerably increase the step size. This method, integrated into the reference solver SUNDIALS IDA and interfaced with an industrial simulation engine, enables to very significantly accelerate simulations in comparison with a classical implementation.
Keywords: power systems time-domain simulation, adaptive step size integration scheme, parametric estimator, numerical stability