Use of the Sinusoidal Predictor Method within a Fully Separated Modeler/Solver Framework for Fast and Flexible EMT Simulations

Pierre-Marie Gibert[1], Romain Losseau[1], Adrien Guironnet[1], Patrick Panciatici[1], Damien Tromeur-Dervout[2] and Jocelyne Erhel[3]
[1]*System Expertise Department, RTE, Versailles, France*
[2]*Institut Camille Jordan, University Claude Bernard Lyon 1, Lyon, France*
[3]*Fluminance Team, Inria Rennes, Rennes, France*

Keywords: Electromagnetic Transients, Time-domain Simulations, Adaptive Step Size Algorithm.

Abstract: In order to help the introduction of renewable energy sources into the power system, transmission system operators need extremely reliable simulation tools. In most of the existing simulation tools, heuristics used to accelerate the simulations combine the modeling and solving aspects. Such kinds of approaches lead to results whose validity can be questioned. In contrary, our framework aims at avoiding such heuristics by completely separating the modeler and the solver. However, such an approach is difficult as the computational cost may be significantly higher. In addition, as we simulate the full-waveform of AC power systems, the time step size used by classical methods for integrating the arising systems of DAE is limited by the frequency of some electrical components. This is why we developed an efficient solver based on the reference solver SUNDIALS IDA which takes into account the guessed sinusoidal behavior of the solution oscillating parts in order to lower the number of iterations for performing a simulation. Our first results seem to indicate that the proposed solver enables to use much larger time steps than a standard integration scheme, while preserving a very high flexibility of modeling.

1 INTRODUCTION

Modern power systems are characterized by the development of smart-grids and the increasing proportion of renewable energy sources in the energy mix. Therefore, more and more new smart controls and power electronics devices are introduced into the transmission grid. In order to support this technological changes while providing a constant security of supply to their customers, transmission system operators (TSOs) perform numerous time-domain simulations. Time-domain simulations generally consist in studying the dynamical behavior of the power system when it is subject to scenarios, which can be roughly seen as sequences of discrete events associated with perturbations and parades.

However, these simulations require important computational resources because of the stiffness of the differential algebraic equations (DAE) systems that arise when modeling power systems in the time-domain. Indeed, on the one hand, modern components introduce electromagnetic transients (EMT) 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, electromechanical components can be dramatically affected by electromagnetic phenomena (e.g. magnetic saturation of the coils in synchronous machines). Then, the study of these two kinds of phenomena can be less and less separated.

Historically, as the time and space propagation of EMT through the transmission grid was assumed to be very limited, simulations were done only on a small part of the system and during very short time intervals. This resulted to the use of different models and so of different simulation tools depending of the dynamics to study. The simulation of electromagnetic transient phenomena was based on so-called EMT models. And the simulation of electromechanical transient phenomena was done from so-called TS (for Transient Stability) models. On the contrary, as

Gibert, P-M., Losseau, R., Guironnet, A., Panciatici, P., Tromeur-Dervout, D. and Erhel, J.
Use of the Sinusoidal Predictor Method within a Fully Separated Modeler/Solver Framework for Fast and Flexible EMT Simulations.
DOI: 10.5220/0006860400340043
In *Proceedings of 8th International Conference on Simulation and Modeling Methodologies, Technologies and Applications (SIMULTECH 2018)*, pages 34-43
ISBN: 978-989-758-323-0

EMT models are more detailed and so contain TS models in some ways, our objective is to perform mid-to-long-term simulations of large-scale power systems with full-EMT models.

In addition, in most of the existing simulation tools, the modeling and solving aspects are generally combined. Indeed, components are directly implemented in discretized form: trapezoidal formula for smooth dynamics and backward Euler formula for discontinuities. Although it enables faster computations, such an approach leads to an increasing number of heuristics in models and consequently to simulations whose reliability can be questioned. To finish, the DAE systems are generally solved with fixed step size integration schemes. 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.

For performing reliable long-term time-domain simulations of power systems represented with EMT models, our strategy is based on two cornerstones:

1. Completely separating the modeler and the solver. From a mathematical point of view, the idea is that the modeler should only build the DAE system to solve, which is written in implicit form i.e. $F(t,X,\dot{X}) = 0$. Then, the solver should perform the numerical integration of this system.
2. Integrating the DAE with an adaptive step size solver. By this way, the time step size is adjusted in order to meet a tolerance target, which finally reinforces the reliability of the simulation results. Furthermore, the step size adjustment enables to possibly optimize the computational cost. Indeed, as the step size is directly linked to the dynamics of the solution (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.

However, in EMT simulations, using an adaptive step size strategy is generally inefficient because of the sinusoidal behavior of the simulated three-phase voltages and currents. Indeed, their frequency limits the time step size that is used for the integration. To tackle this constraint, most of the existing approaches consist in rewriting the DAE system in order to lower the frequency of the computed solution. For instance, in Dynamic Phasors (Demiray, 2008)-(Demiray et al., 2007) the oscillating components are represented by their envelope instead of their waveform. Then, as the frequency of the resulting variables is close to zero, much larger time steps can be used. However, as this approach directly affects the model by modifying the equations to solve in a non strictly equivalent way, it is not adapted to our framework.

Hence, the method that we implemented takes into account the sinusoidal behavior of the oscillating components in the solver. Basically, the idea is to decompose the solution as the sum of a periodic part and a correction term. The former is updated using parametric estimation. The latter is integrated using an adaptive step size solver. In previous publications, results were obtained with Scilab (Scilab et al., 2012), a Matlab-like language, which is not suited to large-scale tests. In this paper, we present the first results obtained with the implementation of our method into the reference solver SUNDIALS IDA (Hindmarsh et al., 2005). In order to set our simulations, we used Dynamo (Guironnet et al., 2018), which is a simulation engine based on the OpenModelica (Fritzson et al., 2006) environment and developed by RTE for time-domain simulations.

This paper is structured as follows. In the second section, the mathematical problem to solve when considering EMT simulations is introduced. In the third section, our numerical method is presented. In particular, we detail its implementation into the reference solver IDA and its interface with Dynamo. In the fourth section, some numerical results on two power systems are presented. The fifth section concludes this paper.

2 MATHEMATICAL PROBLEM TO SOLVE

In EMT simulations, the mathematical problem to solve is a set of differential-algebraic equations, which can be written in its most general form as

$$F(t,X(t),\dot{X}(t)) = 0 \tag{1}$$

Let d be the dimension of the system. In this equation $t \in \mathbf{R}$, $X : \mathbf{R} \to \mathbf{R}^d$ and $F : \mathbf{R} \times \mathbf{R}^d \times \mathbf{R}^d \to \mathbf{R}^d$. In this paper, we present our method for such implicit DAEs. Our previous paper (Gibert et al., 2018) focuses on semi-explicit DAEs, as DAE systems can generally be rewritten in such form for most of the power system applications (Astic et al., 1994).

Furthermore, the solution of this system X is assumed to contain both oscillating (for instance, the three-phase currents and voltages) and non-oscillating components, i.e.

$$X(t) = \begin{bmatrix} X_s(t) \\ X_{ns}(t) \end{bmatrix} \tag{2}$$

where X_s and X_{ns} respectively refer to the oscillating components and to the non-oscillating components.

Let d_s (respectively $d_{ns} = d - d_s$) be the number of oscillating (respectively non-oscillating) components and I_s (respectively I_{ns}) the associated set of index. These index sets are known a priori (e.g. network voltages and currents).

The idea of the Sinusoidal Predictor Method (SPM) is to take advantage of the behavior of the oscillating components in steady-state. Indeed, AC power systems are designed such as voltages and currents are as close as possible to balanced three-phase quantities oscillating at the reference frequency (for instance 50 Hz for the Continental Europe). In other words, its means that three-phase systems should verify:

$$X_{s,\infty}(t) \approx \begin{bmatrix} A_\infty \cos(\omega_0 t + \phi_\infty) \\ A_\infty \cos(\omega_0 t + \phi_\infty - \frac{2\pi}{3}) \\ A_\infty \cos(\omega_0 t + \phi_\infty + \frac{2\pi}{3}) \end{bmatrix} \quad (3)$$

Where A_∞ and ϕ_∞ are respectively constant amplitude and phase. In addition, non-oscillating components should be constant by definition, which means that $\dot{X}_{ns,\infty} \approx 0$. Then, in steady-state, the DAE system should verify

$$F\left(t, \begin{bmatrix} X_{s,\infty}(t) \\ X_{ns,\infty} \end{bmatrix}, \begin{bmatrix} \dot{X}_{s,\infty}(t) \\ 0 \end{bmatrix}\right) = 0 \quad (4)$$

Therefore, the SPM, presented in (Gibert et al., 2017) and (Gibert et al., 2018), consists in decomposing the solution for each time interval $[t_n, t_{n+1}]$ as the sum of a periodic part and a correction term, i.e.

$$X(t) = \bar{X}_n(t) + \delta(t) \quad (5)$$

In this equation, $\bar{X}_n(t)$ refers to the periodic part of the solution which corresponds to the guessed steady-state behavior of the oscillating components. Then, as mentioned in (3), these variables should be sinusoid oscillating at the reference frequency i.e.

$$\bar{X}_{s,n}(t) = u_n \sin(\omega_0 t) + v_n \cos(\omega_0 t) \quad (6)$$

where $\omega_0 = 2\pi f_0$ is the nominal pulsation (with $f_0 = 50Hz$ e.g.). Hence, u_n and v_n are the Fourier coefficients associated to the fundamental mode f_0. The objective of the SPM is to catch as much as possible the simple sinusoidal behavior of the solution into this periodic part. Indeed, as the power system is designed to generate balanced three-phase signals, the oscillating components of the solution should be sinusoids with constant amplitude and phase in steady-state. So, the f_0 oscillation finally corresponds to a trivial dynamical component of the solution which is paradoxically the main limit to solver performances since it constrains the usable step size. The Fourier coefficients are updated with a parametric estimator which takes advantage of the property (4).

Therefore the other part of the solution, that we refer as the correction term, enables to catch transient parts of the solution that are mainly introduced by discrete events. For instance, it can be envelope variations (as the Fourier coefficients are set to constant values for each time interval) or more complex dynamics such as harmonics and offsets. This correction term is computed by integrating the equations rewritten on it.

Then, an iteration of the SPM simply consists in (see figure 1):

1. fixing the parameters of the periodic part;
2. integrating the equations rewritten on the correction term;
3. deducing the global solution from the current periodic part and the integrated correction term;
4. updating the periodic part, by estimating its parameters.

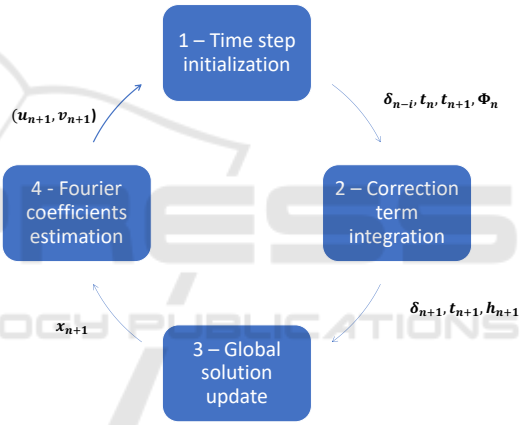

Figure 1: Overall view of an SPM iteration.

3 PROPOSED SOLVER: IDASPM

3.1 Existing Reference Solver: IDA

IDA is the implicit differential algebraic equations solver from the SUNDIALS library (Hindmarsh et al., 2005), developed and still maintained by Lawrence Livermore National Laboratory. It is a modern version of the solver DASSL (Petzold et al., 1982), which is an industrially validated implementation of the adaptive step size BDF (Brayton et al., 1972). The adaptive step size BDF of order q consists in substituting the time derivative of the solution by the following approximation:

$$\dot{X}_{n+1} \approx \frac{1}{h_n} \sum_{i=0}^{q} \alpha_{n,i} X_{n+1-i} \quad (7)$$

where $h_n = t_{n+1} - t_n$. Then, by injecting this discretization of $\dot{X}_{n+1}$ into (1), the solution at next time step t_{n+1} can be obtained by solving the following fixed-point problem:

$$G(X_{n+1}) = F\left(t_{n+1}, X_{n+1}, \frac{\alpha_{n,0}}{h_n} X_{n+1} + \beta_n\right) \quad (8)$$

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

$$J_G(X_{n+1}) = \frac{\partial F}{\partial X} + \frac{\alpha_{n,0}}{h_n}\frac{\partial F}{\partial \dot{X}} \quad (9)$$

More details on the theoretical aspects of IDA can be found in (Hindmarsh et al., 2005). From an implementation point of view, one of the greatest features of IDA is its wide variety of linear solvers for solving the linear systems that arise in Newton-Raphson iterations. In particular, it contains an implementation of the KLU solver for sparse matrices, which is particularly suited for power systems applications (CRSA et al., 2011).

Then, the architecture of IDA roughly consists in:

- a generic differential algebraic equations solver, which corresponds to the implementation of the BDF with variable step size and order. So, this general module performs the prediction, the correction and the step size and order adjustment. In our case, the maximum order is fixed to 2. More precisely, in the correction step, it executes the Newton-Raphson iterations by calling virtual functions for building and solving the linear system: initialization, setup and solve.
- Specific modules for the different linear solvers. For instance, in direct solvers, the linear system setup evaluates the Jacobian matrix and performs the factorization. Then, the solve function applies the numerical factorization of the matrix on the input right-hand-side vectors. In particular, the Jacobian matrix storage is different depending on the linear solver. For instance, a Compressed-Sparse-Row storage is used for the KLU solver.

3.2 Theoretical Aspects of IDASPM

In this section, the SPM is presented for its implementation into IDA, i.e. when using the adaptive step size Backward-Differentiation-Formula as basic integration scheme. However, the SPM can be implemented in any numerical solver. For instance, in (Gibert et al., 2018), the integration of the SPM into the mixed Trapezoidal Formula - Backward-Differentiation-Formula (Astic et al., 1994) is presented.

So, as mentioned in section 2, the first step of the SPM consists in fixing the Fourier coefficients u_n and v_n 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:

$$\Phi_n(t, \delta, \dot{\delta}) = F(t, \bar{X}_n(t) + \delta, \dot{\bar{X}}_n(t) + \dot{\delta}) \quad (10)$$

As for the classical BDF, the fixed-point problem on δ_{n+1} is obtained by applying the BDF discretization formula (7) to approximate $\dot{\delta}_{n+1}$:

$$\dot{\delta}_{n+1} \approx \frac{1}{h_n}\sum_{i=0}^{q} \alpha_{n,i}\delta_{n+1-i} \quad (11)$$

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 new values of the Fourier coefficients. By this way, the consistency of δ is ensured with Φ_n.

By applying this integration scheme on the differential algebraic equation (10), we get an implicit problem similar to (8) that can be solved with a fixed-point algorithm:

$$\begin{aligned}
&\Gamma_n(\delta_{n+1}) \\
&= \Phi_n\left(t_{n+1}, \delta_{n+1}, \frac{\alpha_{n,0}}{h_n}\delta_{n+1} + \beta_n^{\delta}\right) \\
&= F\left(t_{n+1}, \bar{X}_n(t_{n+1}) + \delta_{n+1}, \dot{\bar{X}}_n(t_{n+1}) + \frac{\alpha_{n,0}}{h_n}\delta_{n+1} + \beta_n^{\delta}\right)
\end{aligned} \quad (12)$$

Whose Jacobian matrix is actually equal to the Jacobian matrix of the original system:

$$J_{\Gamma_n}(\delta_{n+1}) = \frac{\partial F}{\partial X} + \frac{\alpha_{n,0}}{h_n}\frac{\partial F}{\partial \dot{X}} \quad (13)$$

Once δ_{n+1} is computed, the global solution X_{n+1} is directly obtained from (5):

$$X_{n+1} = \bar{X}_n(t_{n+1}) + \delta_{n+1} \quad (14)$$

Then the parameters of the periodic part of the solution can be updated. In our current implementation, this update is done using a predictive parametric estimator. Hence, our estimation process consists in applying an iteration of the modified Newton algorithm for minimizing a measurement of the stationarity of the system. Indeed, as shown in (4), in steady-state:

- The oscillating components of the solution should be centered sinusoids with constant Fourier coefficients oscillating at the reference frequency of the system.
- The non-oscillating components of the solution should be constant and so their time-derivative should be equal to zero.

So, our objective is to minimize the energy function associated to (4):

$$R(u_{n+1}, v_{n+1}) = \int_{t_{n+1}}^{t_{n+1}+T} \left\| F\left(t, \begin{bmatrix} \bar{X}_{s,n+1}(t) \\ X_{ns,n+1} \end{bmatrix}, \begin{bmatrix} \dot{\bar{X}}_{s,n+1}(t) \\ 0 \end{bmatrix}\right) \right\|^2 dt \quad (15)$$

In this equation, $R : \mathbf{R}^{2d_s} \rightarrow \mathbf{R}$. This estimator is presented in more details in (Gibert et al., 2018). However, it is important to recall that it leads to the resolution of the following linear system:

$$\begin{bmatrix} H^{uu}_{n+1} & H^{uv}_{n+1} \\ H^{vu}_{n+1} & H^{vv}_{n+1} \end{bmatrix} \begin{bmatrix} \Delta^{u}_{n+1} \\ \Delta^{v}_{n+1} \end{bmatrix} = - \begin{bmatrix} g^{u}_{n+1} \\ g^{v}_{n+1} \end{bmatrix} \quad (16)$$

where the sub-matrices $H^{uu}_{n+1}, H^{uv}_{n+1}, H^{vu}_{n+1}, H^{vv}_{n+1} \in \mathbf{R}^{d_s \times d_s}$ correspond to an approximation of the components of the Hessian matrix of (15) and the right-hand-side vector composed of $g^{u}_{n+1}, g^{v}_{n+1} \in \mathbf{R}^{d_s}$ corresponds to its gradient. Hence, these components require the evaluation of the DAE system residual function and Jacobian matrix, i.e. F and J_F. Once this linear system is solved, the new Fourier coefficients are simply given by $u_{n+1} = u_n + \Delta^{u}_{n+1}$ and $v_{n+1} = v_n + \Delta^{v}_{n+1}$.

3.3 Practical Considerations of IDASPM

Then, in order to implement the SPM into IDA, modifications have been made at the two architecture levels of IDA: at the general algorithm level and at the matrix-specific level.

To begin, we added a member to the main data structure IDAMem, which contains the solver data. This member is a generic pointer to a SPM-specific data structure containing all the data needed for the SPM. More precisely, it contains the boolean vector distinguishing the oscillating and non-oscillating components of the solution, the vector of Fourier coefficients and the reference pulsation of the system. Furthermore, it contains integration and estimation work-space data. To initialize this data-structure, the user only has to properly set the above-mentioned boolean vector and pulsation. Then, the method automatically performs the DAE system reformulation, the decomposition of the solution, the correction term integration and the Fourier coefficients update.

Hence, in the general IDA algorithm, our modifications enable to integrate the DAE system rewritten on the correction term. So, first of all, the Fourier coefficients are fixed and the continuity condition is applied for making the history of δ consistent with Φ_n. Then, the predictor on δ is built from these history data and applied at time t_{n+1} for computing the prediction $\hat{\delta}_{n+1}$. The full solution prediction $\hat{X}_{n+1}$ is deduced by adding $\bar{X}_n(t_{n+1})$. Once this prediction is known, the DAE residual function is evaluated for the Newton iterations of the correction step. Finally, when the correction step is complete, the error test on δ is performed and the step size is updated consequently.

In a SPM specific module, the estimator is implemented. As the estimator uses the Jacobian matrix of the DAE system, its implementation depends on the matrix storage format. Since the KLU is mainly used for power systems applications, our current implementation is done for CSR matrix storage. In a first implementation, the Jacobian matrix was updated within the estimator function but it resulted in a very important computational time by iteration. So, in our current implementation, the Jacobian matrix used in the estimator is updated only when IDA updates its Jacobian matrix. By this way, the number of Jacobian matrix evaluations is dramatically reduced while preserving a good level of approximation. Indeed IDA performs several robust numerical tests to infer on the validity of the Jacobian matrix.

The figure 2 summarizes the modifications made to the IDA algorithm for using the SPM.

3.4 Interface with Our Customized Framework based on Modelica

OpenModelica (Fritzson et al., 2006) is an open-source suite based on Modelica language (Fritzson and Engelson, 1998), which is used for the modeling and simulation of complex systems. It 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 (Chieh et al., 2011) and then iTesla (Vanfretti et al., 2016) have introduced and proved the industrial potential of Modelica for time-domain simulations of transmission grids. Furthermore, the latter resulted in the development of the iPSL library, which is open-source and still maintained. Then, the dynamical-study tools team of RTE implemented Dynamo, a new simulation engine (Guironnet et al., 2018) which is based on this framework. It especially enables to easily connect models with solvers to perform simulations in a very flexible way.

Hence, in our framework, Modelica is used for the modeling part of the simulation. Then, our objective is to implement the model of the simulated system

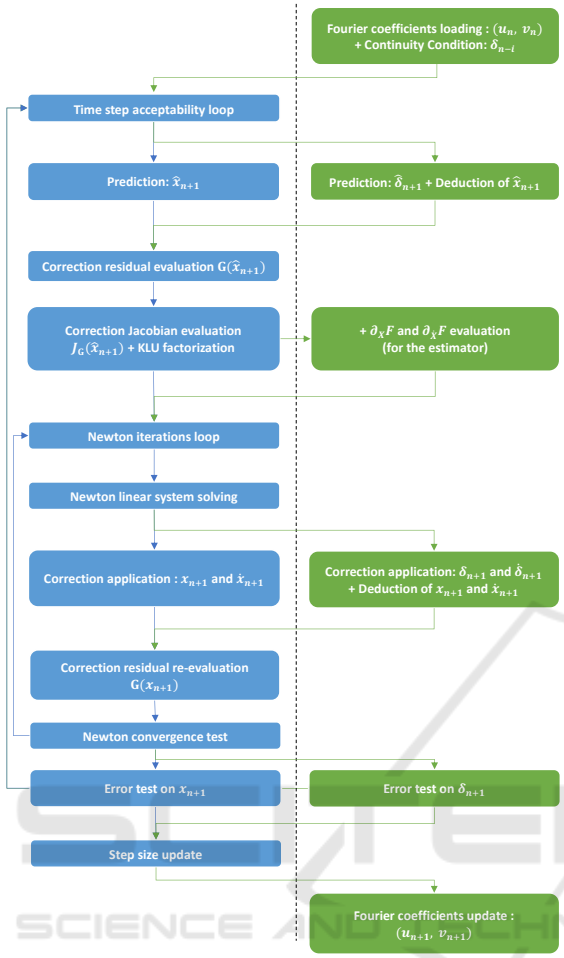

Figure 2: Modification of the general algorithm of IDA for integrating the SPM: the left part corresponds to the standard IDA algorithm and the right part details the SPM-specific operations.

with the Modelica language and, then, to perform the simulation with IDASPM from Dynamo. To summarize, in OpenModelica-type frameworks the simulation process consists in three main steps:

1. The user implements a model from existing or user-defined libraries. A model generally consists in assembling unit models within a global model from their physical connections.
2. The model compiler, e.g. OpenModelica Compiler, interprets this model to construct the associated DAE system to be solved. So, it constructs a "flat model" by injecting all the mathematical equations associated to the different unit components and connections. In output of this process, OpenModelica Compiler generates several source files corresponding to the simulation: residual function of the DAE system, parameters, initial values, etc.
3. To finish, as we have the files containing the DAE system at our disposal, it can be integrated using the chosen solver, e.g. SUNDIALS IDA. To do this, our simulation engine contains generic modules which automatically initialize the solver data structures and data, and perform the simulation from the chosen solver library.

Therefore, in order to interface our solver with Dynamo, we have to generate the solver library corresponding to IDASPM and then to implement a customized solver module which properly initializes the data structures of IDASPM. In particular, it supplies to IDASPM the pulsation of the system and the boolean vector distinguishing the oscillating and non-oscillating variables from a configuration file. In the current version of our implementation, this file is manually completed by the user and provided at execution time, which could be automatically done from the modeler in a future version. Then, from these data, IDASPM can be used for the simulation. 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 some iterations are required for the Fourier coefficients to converge and consequently for the step size to increase.

4 RESULTS AND DISCUSSION

4.1 Results on a Small Power System

First of all, our implementation has been tested on a small three-phase power system with 4 nodes (see figure 3). It contains 3 perfect generators, 1 resistive load and connections with transmission lines modeled with RL branches. In our framework, this model results in a system of 188 equations including 15 differential equations. Then, this system is subject to an exponential increase of the electrical load at time $t_{event} = 1s$. Our 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.

The figure 4 shows that the solution obtained with IDASPM perfectly fits with those obtained with IDA. Then, in the figure 5, the step sizes used by IDA and IDASPM are compared. Then, 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, 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 very 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.

Hence, IDASPM requires 157 time steps and 0.64 seconds to perform the simulation. IDA requires 52115 time steps and 2.69 seconds to perform the simulation. Their performances are summarized in table 1. So, using the SPM enables to dramatically reduce the number of iterations for performing this simulation (divided by more than 300). The speedup is also significant (divided by 4). However, we can see that it is not as important as for the number of iterations. Indeed, the SPM requires to perform more mathematical operations whose computational cost is not negligible than a classical integration method. In particular, as our current implementation is not completely optimized, the computational cost of the estimator remains important. This cost is mainly due to the use of the Jacobian matrix of the DAE system and the different linear algebra operations that are done during the estimation process. This is why the final computational time by iteration is currently high.

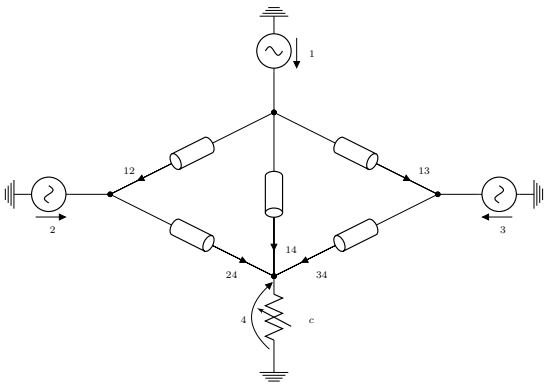

Figure 3: Simple Electrical Grid test case (figure from (Gibert et al., 2018)).

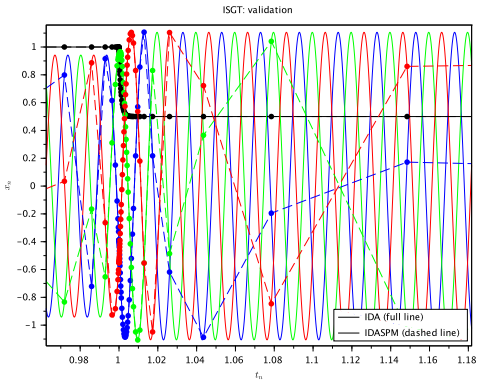

Figure 4: Comparison of the solutions obtained with IDA (full line) and IDASPM (dashed line) on the Simple Electrical Grid test case (tolerance = 1.e-4).

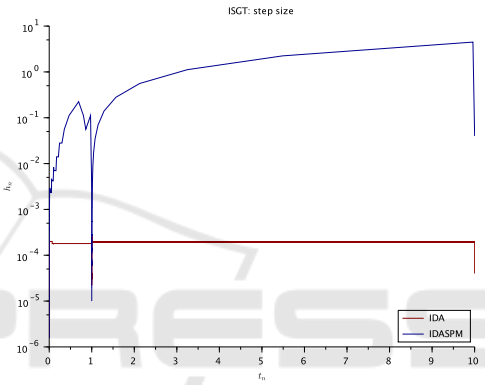

Figure 5: Comparison on the step size used by IDA (red line) and IDASPM (blue line) on the Simple Electrical Grid test case with tol=1.e-4.

Table 1: Performances comparison between IDA and IDASPM on the SEG test case.

tol	Method	SEG
1.e-4	IDA	52115 it. (2.69 s)
	IDASPM	157 it. (0.64 s)
	IDA/IDASPM	**332 (4.20)**
1.e-5	IDA	112035 it. (6.65 s)
	IDASPM	290 it. (0.76 s)
	IDA/IDASPM	**386 (8.75)**

4.2 Results on a Customized Version of the IEEE 14-bus Reference Test Case

Then, our implementation has been tested on a larger power system with 14 nodes (see figure 6), inspired on the reference test case IEEE-14 bus. Its 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

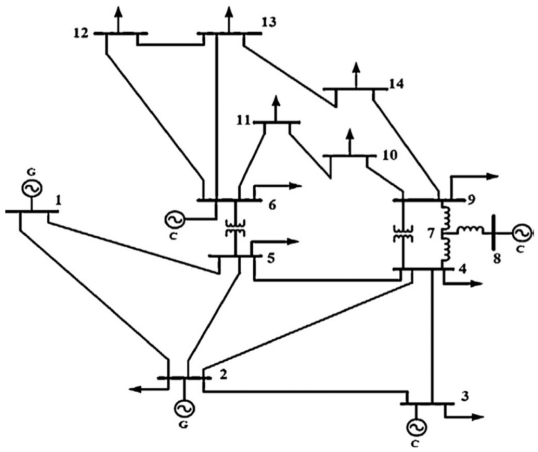

Figure 6: IEEE 14-bus test case (figure from (Abbas Taher et al., 2015)).

differential equations. Then, this system is subject to an exponential increase of the electrical load attached to Bus 4 at time $t_{event} = 1s$.

The figure 7 shows that the solution obtained with IDASPM perfectly fits with those obtained with IDA. In figure 8, the step sizes used by IDA and IDASPM are compared. Results are very similar to those obtained on the Simple Electrical Grid test case. Indeed, IDASPM requires 174 time steps and 11.66 seconds to perform the simulation. IDA requires 48504 time steps and 3.21 seconds to perform the simulation. Their performances are summarized in table 2. Then, as for the Simple Electrical Grid test case, the number of time steps is very significantly reduced by using the SPM. However, as the computational cost by iteration is still too important in our current version of IDASPM, there is finally no speedup. Further developments will focus on optimizing the Fourier coefficients estimator, which represents the large majority of the computational time.

Table 2: Performances comparison between IDA and IDASPM on the IEEE 14-bus inspired test case.

tol	Method	IEEE-14
1.e-4	IDA	48504 it. (3.21 s)
	IDASPM	174 it. (11.66 s)
	IDA/IDASPM	**279 (0.28)**
1.e-5	IDA	116705 it. (7.90 s)
	IDASPM	306 it. (13.24 s)
	IDA/IDASPM	**381 (0.60)**

4.3 Discussion and Area for Improvement of IDASPM

In conclusion, in spite of the large reduction of number of iterations for performing time-domain simulations, the final speedup with our current implementation of IDASPM is low for some test cases. For instance, on the IEEE 14-bus inspired test case, using IDASPM leads to a greater computational time while the number of iterations is divided by a factor close to 300. On the contrary, on the SEG test case, the speedup is significant but still could be enhanced.

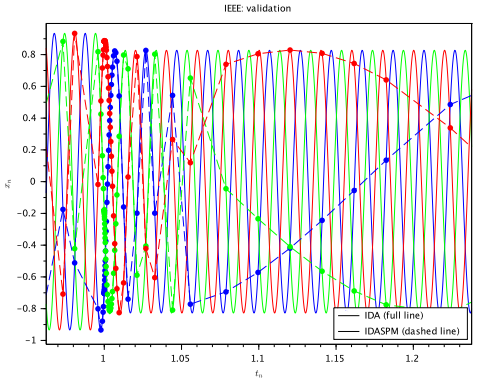

Figure 7: Comparison of the solutions obtained with IDA (full line) and IDASPM (dashed line) on the IEEE 14-bus inspired test case (tolerance = 1.e-4).

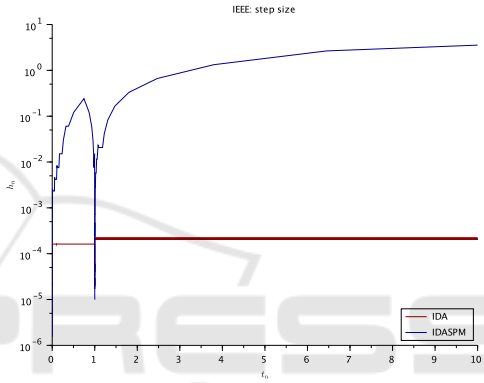

Figure 8: Comparison on the step size used by IDA (red line) and IDASPM (blue line) on the IEEE 14-bus inspired test case (tolerance = 1.e-4).

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

- for SEG, 4.1ms/it. with tol.=1.e-4 and 2.6ms/it. with tol.=1.e-5;
- for IEEE, 67ms/it. with tol.=1.e-4 and 43ms/it. with tol.=1.e-5.

In other words, the computational time by iteration is about 16 times greater between SEG and IEEE, i.e. the dimension increase factor squared. So, we can see that our implementation is currently 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 (Weidendorfer, 2008) to profile the performances of our implementation. Two functions cover the large majority of the solving process:

1. The linear solver setup function, which performs the Jacobian evaluation and factorization. The setup function represents 24.64% of the global computational cost. In particular, the relative cost of the Jacobian evaluation is 23.02%, i.e. about 93.43% of the setup.
2. The Fourier coefficient update function represents 75.29% of the global computational cost. In particular the relative cost of the matrix factorization, that is done for solving the linear system of the estimator (16), is 67.20%, i.e. 89.25% of the estimator cost.

Therefore, these first investigations make several optimization ideas evident:

- In IDA, the Jacobian matrix of the DAE system is given by the formula (9) i.e. $J_F = \partial_X F + c_j \partial_{\dot{X}} F$. Then, as our Fourier coefficients estimator requires the two partial derivatives of F, $\partial_X F$ and $\partial_{\dot{X}} F$, the current implementation of IDASPM calls the Jacobian matrix evaluation function 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, each Jacobian matrix evaluation calls a modeler-level function which computes the Jacobian matrix by performing an automatic differentiation of the residual function. To finish, once the Jacobian matrix evaluations have been done for the estimator, some matrix operations are required for extracting the $\dot{X}$ partial derivative. So, our first optimization idea consists in modifying the IDASPM Jacobian evaluation function for filling the three different Jacobian matrix in a single call. Indeed, in the modeler-level code, the two partial derivatives are evaluated and then added. By this way, an important computational time could be saved.
- Furthermore, the current implementation fills a dense matrix for the estimator linear system from the sparse Jacobian matrix of the system. However, as the estimator matrix should also be very sparse, an important computational time could be saved by solving this linear system with the KLU, which is a sparse solver.
- In addition, we noted that there are significantly more Jacobian updates when using IDASPM compared with IDA. Generally, the Jacobian matrix is updated by the integrator when the Newton algorithm fails to converge. These convergence fails may be due to major changes in Jacobian matrix. However, there is no apparent reason for the Jacobian matrix of IDASPM to change more than those of IDA. This point is to investigate, especially as updating the Jacobian matrix requires to perform a new factorization for the Newton iterations.
- To finish, in our current implementation of IDASPM within Dynamo, the Fourier coefficients are set to zero. Then, for each tested case, we can see that the step size is limited at the very beginning of the simulation. So, by properly initializing the Fourier coefficients, some iterations could be saved at that very time. However, the presented results prove the robustness of the estimator as Fourier coefficients converge in a few number of iterations.

5 CONCLUSIONS

In conclusion, the implementation presented in this paper paves the way to long-term EMT simulations of power systems within a very flexible framework. Using Modelica and especially our customized environment based on OpenModelica enables to clearly distinguish the modeling and solving aspects of the simulation. Furthermore, using the proposed IDASPM solver leads to a very important reduction of the number of iterations for performing a simulation, compared to the original IDA solver. However, as the current implementation of our method is not completely optimized, there is no speedup on larger test cases. Therefore, our further developments will focus on optimizing the estimator, in which the main part of the computational time is currently spent. Indeed, as mentioned in the results discussion, some accessible enhancements are envisioned, especially for reducing the computational cost of linear algebra operations. In addition, we could investigate strategies for activating and deactivating the Fourier coefficients update during transient phases. Indeed, in such phases, as the Fourier estimation is perturbed, the step size is finally limited.

ACKNOWLEDGMENT

The authors would like to thank the French National Research Agency in Technology who founded this project under the contract CIFRE nº 2015/0885.

REFERENCES

Abbas Taher, S., Mahmoodi, H., and Aghaamouei, H. (2015). Optimal pmu location in power systems using mica. *Alexandria Engineering Journal*.

Astic, J., Bihain, A., and Jerosolimski, M. (1994). 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.

Brayton, R. K., Gustavson, F. G., and Hachtel, G. D. (1972). A new efficient algorithm for solving differential-algebraic systems using implicit backward differentiation formulas. *Proceedings of the IEEE*, 60(1):98–108.

Chieh, A. S., Panciatici, P., and Picard, J. (2011). Power system modeling in modelica for time-domain simulation. In *PowerTech, 2011 IEEE Trondheim*, pages 1–8. IEEE.

CRSA, RTE, TE, and TU/e (2011). D4.1: Algorithmic requirements for simulation of large network extreme scenarios. Technical report, PEGASE Consortium.

Demiray, T. (2008). *Simulation of power system dynamics using dynamic phasor models*. PhD thesis, ETH Zurich.

Demiray, T., Milano, F., and Andersson, G. (2007). Dynamic phasor modeling of the doubly-fed induction generator under unbalanced conditions. In *Power Tech, 2007 IEEE Lausanne*, pages 1049–1054. IEEE.

Fritzson, P., Aronsson, P., Pop, A., Lundvall, H., Nystrom, K., Saldamli, L., Broman, D., and Sandholm, A. (2006). 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.

Fritzson, P. and Engelson, V. (1998). Modelica: A unified object-oriented language for system modeling and simulation. In *European Conference on Object-Oriented Programming*, pages 67–90. Springer.

Gibert, P.-M., Losseau, R., Guironnet, A., Panciatici, P., Tromeur-Dervout, D., and Erhel, J. (2018). Speedup of emt simulations by using an integration scheme enriched with a predictive fourier coefficients estimator. under review.

Gibert, P.-M., Tromeur-Dervout, D., Panciatici, P., Beaude, F., Wang, P., and Erhel, J. (2017). A generic customized predictor corrector approach for accelerating emtp-like simulations. In *PowerTech, 2017 IEEE Manchester*, pages 1–6. IEEE.

Guironnet, A., Saugier, M., Petitrenaud, S., Xavier, F., and Panciatici, P. (2018). Towards an open-source solution using modelica for time-domain simulation of power systems. under review.

Hindmarsh, A. C., Brown, P. N., Grant, K. E., Lee, S. L., Serban, R., Shumaker, D. E., and Woodward, C. S. (2005). Sundials: Suite of nonlinear and differential/algebraic equation solvers. *ACM Transactions on Mathematical Software (TOMS)*, 31(3):363–396.

Petzold, L. R. et al. (1982). A description of dassl: A differential/algebraic system solver. *Scientific computing*, 1:65–68.

Scilab, E. et al. (2012). Scilab: Free and open source software for numerical computation. *Scilab Enterprises, Orsay, France*, page 3.

Vanfretti, L., Rabuzin, T., Baudette, M., and Murad, M. (2016). itesla power systems library (ipsl): A modelica library for phasor time-domain simulations. *SoftwareX*, 5:84–88.

Weidendorfer, J. (2008). Sequential performance analysis with callgrind and kcachegrind. In *Tools for High Performance Computing*, pages 93–113. Springer.