Techniques for Interfacing Electromagnetic Transient Simulation Programs With General Mathematical Tools

IEEE Taskforce on Interfacing Techniques for Simulation Tools

S. Filizadeh, *Member, IEEE*, M. Heidari, *Student Member, IEEE*, A. Mehrizi-Sani, *Student Member, IEEE*, J. Jatskevich, *Senior Member, IEEE*, and Juan A. Martinez, *Member, IEEE*

***Abstract*—This paper describes methods and issues related to interfacing electromagnetic transient simulation programs with general mathematical algorithms, which are either custom-developed by the user or are available through other mathematical analysis platforms. Various interfacing types and techniques are described along with potential areas of application. Implementation methods are detailed for each type of interface as well. This paper presents several interfacing examples from a wide variety of applications, including advanced switching schemes for power converters and controller implementation.**

***Index Terms*—Electromagnetic transient simulation, interfacing, mathematical algorithms.**

I. Introduction

INTERCONNECTED electric power networks are large dynamical systems with complex interactions. The analysis and design of these systems often requires use of an array of simulation tools that represent the system behavior in a manner suitable for the intended studies. Simulation tools enable the user (e.g., system operator or designer) to assess various aspects of the operation of the network with ease and without recourse to repetitive prototyping, or costly and potentially harmful tests on the real system.

The level of detail used in the modeling of individual elements and the choice of the solution method used in a simulation tool depend on the nature of the phenomena and requirements of the studies to be carried out. While simplified models suffice in studies such as load flow, detailed, frequency-dependent models of individual elements, including nonlinear and switching components, need to be used when short-term, high-frequency events are concerned [1].

As the complexity of power systems grows, the need for more sophisticated simulation tools escalates. Modern simulation programs have been adapted to this increasing need by such measures as: 1) incorporation of improved user interfaces [2]; 2) enhanced analysis tools [3]; and 3) pre and postprocessing mechanisms [4], [5]. There has, however, been a strong tendency to interface existing simulation tools to obtain further-enhanced simulation capabilities. This approach can ideally allow for a synergetic combination of simulation tools.

Interfacing allows the constituent components (i.e., the simulation tool and the external algorithms/tools) to communicate in a specified manner in order to carry out the overall simulation cooperatively and efficiently. The level of cooperation between the interfaced tools can range from mere postprocessing and visualization of simulation results [4] to relegating an integral part of the simulation to an external tool where specific analyses can be done more effectively than in the original simulation tool [6]–[8].

This paper addresses aspects related to interfacing electromagnetic transient simulation tools [Electromagnetic Transients Program (EMTP)-type programs] with other mathematical tools, including both external programs and stand-alone algorithms. Interfacing EMTP-type tools with external programs/algorithms extends their applicability in areas where more streamlined techniques (e.g., for sophisticated control system design, are available through the external agent). Shorter development time and increased credibility of results are other examples of the benefits of interfacing an EMTP-type tool with a tested and verified external algorithm.

Various kinds of interfaces and methods for their implementation are presented in the paper. For illustration purposes, the PSCAD/EMTDC electromagnetic transient simulation program is used; however, the methods are treated in a generic manner to ensure they remain applicable to arbitrary EMTP-type tools.

II. Electromagnetic Transient Simulation Tools

Electromagnetic transient simulation tools are used to study short-term behavior of complex electrical systems. Due to the

Manuscript received December 28, 2007. First published May 7, 2008; current version published September 24, 2008. Paper no. TPWRD-00811-2007

S. Filizadeh and M. Heidari are with the Department of Electrical and Computer Engineering, University of Manitoba, Winnipeg, MB R3T 5V6, Canada (e-mail: sfilizad@ee.umanitoba.ca; mheidari@ee.umanitoba.ca).

A. Mehrizi-Sani is with the Department of Electrical and Computer Engineering, University of Toronto, Toronto, ON M5S 3G4, Canada (e-mail: ali.mehrizi.sani@utoronto.ca).

J. Jatskevich is with the Department of Electrical and Computer Engineering, University of British Columbia, Vancouver, BC V6T 1Z4, Canada (e-mail: juri@ece.ubc.ca).

J. A. Martinez is with the Department d'Enginyeria Elèctrica of the Universitat Politècnica de Catalunya, Catalunya 080208, Spain (e-mail: martinez@ee.upc.edu).

Task Force on Interfacing Techniques for Simulation Tools is with the Working Group on Modeling and Analysis of System Transients Using Digital Programs, IEEE Power Engineering Society T&D Committee.

Task Force members: U. Annakkage, V. Dinavahi, S. Filizadeh, A. M. Gole, R. Iravani, J. Jatskevich, A. J. Keri, P. Lehn, J. A. Martinez, B. A. Mork, A. Monti, L. Naredo, T. Noda, A. Ramirez, M. Rioual, M. Steurer, K. Strunz.

Digital Object Identifier 10.1109/TPWRD.2008.923533

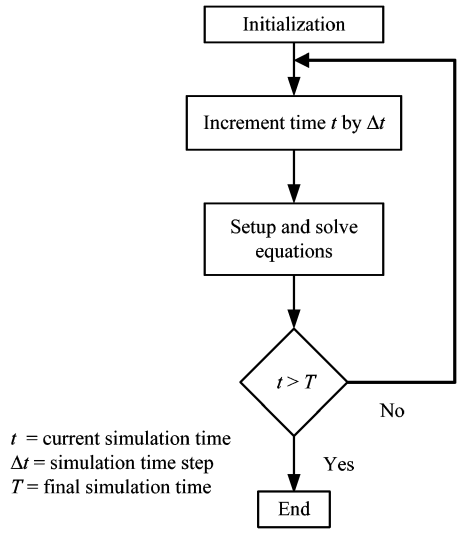

Fig. 1. Schematic diagram of a generic EMTP-type tool.

high level of detail used in representation of system elements, computational intensity of these programs (e.g., CPU requirements), is typically high and their use is mostly limited to networks smaller than what is possible in stability analysis programs [1], [9].

These tools have been used extensively for the analysis and design of various elements of modern systems, where transients (often fast transients) are involved. Typical applications include design of controllers for power apparatus [10], overvoltage and insulation coordination [11], [12], protection [13], flexible ac transmission systems (FACTS) [14], HVDC [15], [16], and other power-electronic applications [17].

The majority of commercially available EMTP-type programs use nodal analysis (admittance matrix formulation) and numerical integration to setup and solve system equations in a progressive manner and in time domain [18], [19]. This formulation is based on the pioneering work by H. W. Dommel in 1969 [1]. Other formulation methods such as state space method have also been used [20], but their mechanics are not discussed here.

Fig. 1 shows a schematic diagram of a generic electromagnetic transient simulation program. Aside from minor modifications introduced by add-on features such as interpolation [21], the overall method for solution of system equations is based on a discretized time axis, where the entire simulation time T is divided into small intervals, each with a constant width of Δt, which is referred to as the simulation time step. Use of a fixed time step allows for simplified modeling and expedites the simulation by eliminating the need for inverting the network admittance matrix repetitively.

The solver of the system equations in each time step uses characteristics of the electrical elements and also those of dynamical devices, such as machines and controls, to obtain updated node voltages. Solution in each time step in general requires some information from previous time step(s). Such history terms are encountered both in electrical network elements (e.g., capacitors and inductors), and in control functions such as integrators [1], [22], [23].

III. Need for Interfacing

Commercially available transient simulation tools include a fairly comprehensive library of components that allow the user to conveniently assemble a circuit for the purpose of simulation. These so-called library components may represent sources, electric machines, cables and transmission lines, semiconductor and power electronic devices, and control and processing functions (e.g., gain blocks, PI controllers, and integrators).

Despite the availability of standard library components, users often find that components for performing specialized computations are not readily available. Therefore, there is a need to extend the capabilities of these programs by incorporating facilities for performing such computations. In EMTP-type programs, these facilities are provided through enabling user-defined components and/or interfacing to other simulation software or programming languages [19], [24].

Interfacing has been used for simulation of complex protection systems [25], development of advanced digital control systems [26], and simulation of power electronic converters using EMTP-TACS [3]. An interface that uses synchronizing clocks for connecting a simulation program with a real controller hardware is proposed in [27]. Attempts have also been made towards development of simulation platforms in which multiple tools interact. Examples of interfaces between PSCAD/EMTDC and MATLAB/SIMULINK are presented in [4], [28], and [29]. A different interfacing method in which an entire simulation is broken into smaller pieces is reported in [30] and is demonstrated using the CIGRE HVDC benchmark model [31]. In [32], a transient simulator is interfaced with a nonlinear simplex optimization algorithm written in FORTRAN to add optimization features to the simulator. A comprehensive example of interfacing between a transient simulator and MATLAB/SIMULINK and a comparison between EMTDC and MATLAB/SIMULINK-PSB is presented in [33], in which CIGRE HVDC benchmark model [31] is considered as the base network.

Besides EMTP-type tools, interfacing has also been used for interconnecting electronic circuit simulators as well. Reference [34] presents DELIGHT.SPICE, which is an integration of DELIGHT interactive optimization-based CAD system and SPICE for circuit optimization. Reference [35] interfaces optimization routines written in MATLAB with SABER circuit simulator. References [36] and [37] show other examples of interfacing circuit simulation programs with optimization algorithms.

IV. Methods for Interfacing

The method used for interfacing EMTP-type tools with other algorithms and the level of complexity involved in doing so depend on the problem it targets to solve. In the following subsections, a number of interfacing methods will be addressed.

A. Static Interfacing

Consider for example interfacing a transient simulator with an external agent in order to plot traces of simulated variables. Ordinarily, one needs to establish a channel between the simulator and the plotting agent to send (in a unidirectional manner) data for the intended variables as they are obtained at each time step. Note that no buffer is necessary to store past values, as data is sent to the plotting agent as it becomes available. Moreover,

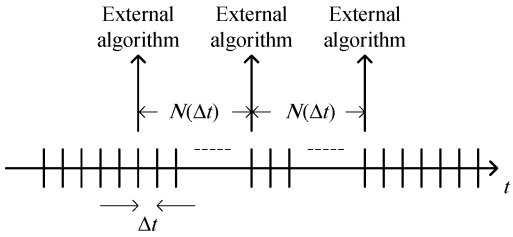

Fig. 2. Static interface to an external algorithm.

one may note that plotting every point on the simulated trace is not necessary and plotting the waveform sampled at regular intervals (other than every time step) still produces graphs of acceptable accuracy. Therefore, it is possible to lower the communication burden by calling the plotting algorithm at regular intervals with a width of N samples. The time line shown in Fig. 2 depicts the procedure graphically for intermittent communication with an external algorithm. Static interfacing may be used for such purposes as plotting or computation of complex functions.

When the simulation case is assembled, the code sections pertaining to the components used are gathered along with their control law. Thus, at every time step, the code for each component is run only if its control law so allows.

Static interfaces have been further categorized as online or offline [4]. In an online interface, the transient simulation tool communicates with the external algorithm throughout the simulation (e.g., in the visualization example later in this paper). In an offline interface, an external tool is called following the completion of the simulation, which does further processing on the simulated data.

B. Dynamic Interfacing and Memory Management

Dynamic interfaces are more involved than static ones, as memory management becomes an integral part of the interface. Processes that simulate dynamic elements, such as controllers or filters, fall in this category. Unlike static interfaces, in which only data from the current time step are communicated between the transient simulator and the interfaced algorithm, dynamic interfaces exchange current as well as past data.

As an example of a dynamic interface, consider a peak-detection component. The component is supposed to track a given signal $x(t)$ and find its peak value and update its output whenever a higher peak is detected. The peak-detector algorithm will be shown.

The algorithm uses a storage variable (`PeakValue`) to store the last peak detected. Although the algorithm is quite simple, its implementation requires access to memory location(s) that are kept intact from one time step to the next. These memory segments are used for storing variables by components that need such storage, and transient simulation tools often provide access to this kind of storage.

An important issue when dealing with memory segments is to note that the pointers to memory locations should be updated and maintained in a unified manner. Since in each time step, the simulation executes the code of individual elements in the order they are placed within the code, it is important to ensure that the stored variables can be retrieved properly. By properly incrementing the pointers, it is guaranteed that they always point to the correct memory locations for all components.

ALGORITHM 1—PEAK DETECTION

```
1 if current simulation time (t) = 0
2  PeakValue = x(0)
3  Output = PeakValue
4 Else
5  if x(t) > PeakValue
6    PeakValue = x(t)
7  End
8  Output = PeakValue
9 End
```

C. Wrapper Interfaces

A wrapper interface is one that does not communicate with the transient simulator on a regular basis throughout the simulation as static and dynamic interfaces do. Instead, it has limited communication at specific points in time, normally at the beginning and end of a simulation. This kind of interface is made when external code controls the simulations in a certain way. An example of this interface is given in Section VII, where optimization and run-control algorithms are discussed.

Note that a wrapper interface is different from an off-line one in that the wrapper interface is often a supervisory algorithm that controls the simulation program and normally performs multiple simulations, whereas an offline interface is usually meant to perform postprocessing of simulation results.

V. WHERE TO IMPLEMENT: EXTERNAL VERSUS INTERNAL INTERFACES

Once an algorithm is developed or selected for interfacing with a simulation tool, one needs to decide where to implement the algorithm. Options for implementation will be discussed.

A. External Interfaces

For algorithms that are available externally through standalone software, external interfacing is normally the most logical option. Depending on the input/output configuration of the algorithm, external interfacing can potentially eliminate the need for rewriting the external code in the indigenous language of the simulation tool. Physical establishment of the interface requires access to the memory management routines of the transient simulator. An example of an external interface is interfacing an EMTP-type program to MATLAB [24], which allows the user to store required variables in predefined locations, called MATLAB, to execute a standard or user-developed code, and retrieve data back to the simulation program for further processing. The interface may allow execution of the external algorithm in each time step or intermittently; therefore, both dynamic and static interfaces described earlier can be implemented externally.

An important observation about external interfacing is the speed implications involved. Transient simulation tools often use optimized methods for enhancing simulation speed. External programs, however, are not necessarily designed with

such provisions. Therefore, a simulation tool that uses an interface with an external program can be drastically slower than the same procedure implemented entirely internally in the EMTP-type tool. Apart from the intrinsic speed differences between the two agents, the overhead of communication between the programs can also significantly affect the overall simulation performance. Unless they are high speed, efficient communication methods are deployed, exchange of large amounts of data between the interfaced tools normally results in a marked reduction in the speed. The problem will be exacerbated if the interface is used as part of a multirun simulation.

Depending on the facilities present in the externally interfaced tool, this type of interfacing can serve as a powerful means for rapid algorithm development, verification, and debugging. It is sometimes easier to make changes to an algorithm developed in a dedicated external agent such as MATLAB than one implemented in the rigidity of an EMTP-type program. Modifications can be easily done and tested through the combined interface. If the speed reduction due to external interfacing is severe, one can consider converting the external interface to an internal one.

Another important aspect of external interfacing is the ability of interfacing to multiple platforms. For example, when an EMTP-type tool is interfaced with MATLAB, other simulation tools (e.g., SIMULINK) or mathematical and programming tools (e.g., coding in multiple languages) may become available as well.

B. Internal Interfaces

A method to alleviate drawbacks associated with external interfaces is to use internal interfacing, where a user-supplied algorithm (unavailable in the transient simulation tool) is implemented within the transient simulator. Internal interfacing is possible when the user has access to the code of the algorithm and is knowledgeable about its inner workings.

Internal interfaces have been used for interfacing nonlinear optimization algorithms with transient simulators (described later). Dynamical models of electric vehicles [38], advanced switching schemes for power converters [39], and specialized motor drives and mechanical models for vehicular power systems [38], [40] have also been interfaced using internal interfacing mechanisms.

Note that internal interfaces are faster than external ones due to the elimination of the communication overhead. However, their implementation is normally more involved than external ones.

VI. Multiple Interfacing

In addition to the main high-power electric circuitry, modern power equipment often contains advanced control blocks, digital processors, nonlinear elements, etc. Proper simulation of these systems should allow uncompromised analysis, and as such, it is sometimes necessary to interface more than two simulation programs, each with special features for detailed modeling of certain aspects of a complex circuit. In this section, some of the schemes for multiple interfacing of a transient simulation programs with other simulation programs or mathematical tools are explained. Variations of these schemes are obviously possible, but are not discussed here.

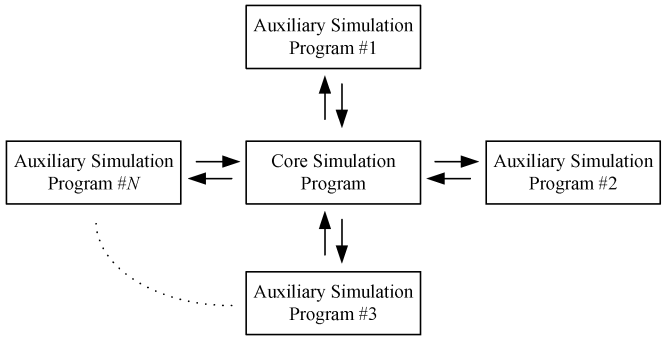

Fig. 3. Core-type interfacing.

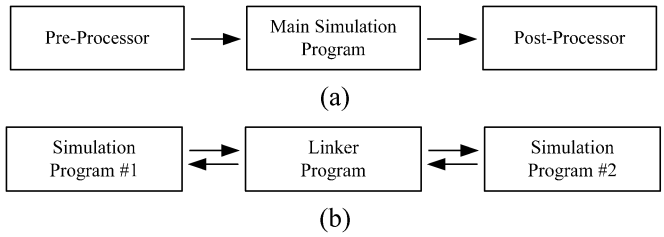

Fig. 4. Variations of chain-type interfacing. (a) Chain-type interfacing for pre and postprocessing. (b) Chain-type interfacing for linking noncompatible simulators.

A. Core-Type Interfacing

In core-type interfacing of simulation programs, one program serves as the core and all of the other (auxiliary) programs are connected to the core. Fig. 3 shows a block diagram of such a structure. The auxiliary programs in this structure can be implemented externally or internally (refer to Section V), and may manifest static, dynamic, or wrapper properties as discussed earlier.

The core-type interfacing structure usually occurs when a major portion of the system under study can be modeled in a single simulation program (the core), and the auxiliary programs are assigned minor tasks, such as data visualization or other calculations. The firing pulse generation and visualization example shown later in this paper is a core-type interface, in which generation of firing pulses and visualization tasks are assigned to auxiliary algorithms that communicate with the core simulator in which the main power circuit is simulated.

B. Chain-Type Interfacing

Unlike a core-type interface where the core program is used as a common node for all other programs, in chain-type interfacing, the simulation programs are connected to each other in a row. There are two common templates for chain-type interfacing, as shown in Fig. 4.

In the first scheme [see Fig. 4(a)], chain-type interfacing is used for preprocessing and postprocessing of the data. As an example, consider simulation of a network with transmission lines. Prior to simulating the network, an algorithm is often used to calculate the line constants to be used in the actual simulator (preprocessing); visualization of the simulated data using a graphing program constitutes postprocessing and the entire scheme takes on the form of a chain-type interface.

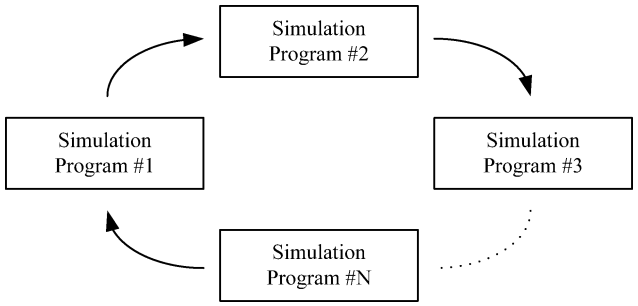

Fig. 5. Loop interfacing scheme.

A second variation of a chain-type interface [see Fig. 4(b)] may be used when simulation programs cannot be interfaced directly and readily. An intermediate agent, such as MATLAB, can be used to bridge the interfacing gap between originally incompatible simulators.

C. Loop Interfacing

If in a chain of simulation programs or external hardware, the last program is also connected to the first one, the result will be a loop interfacing scheme, as shown in Fig. 5. Such combinations occur frequently when real-time simulators are connected to several interacting external pieces of equipment (e.g., relays, controllers, amplifiers, and digital signal processors [41]). Since interfacing of real-time simulators is not the focus of this paper, loop interfacing is not discussed in any further detail.

VII. Examples of Interfacing

A. Interfacing to MATLAB/SIMULINK

This section explains an interface made between a transient simulator (PSCAD/EMTDC) and MATLAB/SIMULINK. Similar interfaces have also been made between EMTP and MATLAB [24]. These interfaces can be used in a variety of ways, allowing full exploitation of the computational facilities in MATLAB and modeling capabilities of SIMULNK.

The interface between the EMTP-type simulation engine and MATLAB is essentially an external interface. The transient simulation engine can communicate with MATLAB either in each time step or intermittently, depending on the nature and requirements of the externally sourced task.

To interface with MATLAB, the user needs to perform the tasks of: 1) declaring memory requirements; 2) storing input variables to MATLAB (transient simulator outputs); 3) calling MATLAB; and 4) receiving MATLAB outputs and feeding them back to the simulator. In EMTDC, the subroutine `MLAB_INT`, which is accessible by user-defined components, establishes the connection between the two agents. The exchange of data between the simulator and MATLAB is administered through the use of data storage queues for storing floating-point, integer, and other data types. Fig. 6 shows a schematic diagram of the sequence of events that occur within the EMTDC/MATLAB interface.

As shown, a number of memory storage locations that are equal to the total number of inputs and outputs communicated to and from MATLAB is first declared. Inputs to the MATLAB environment are placed in the respective storage locations and then the MATLAB interface subroutine is called, which reads the data from the memory locations and communicates them with the respective MATLAB function.

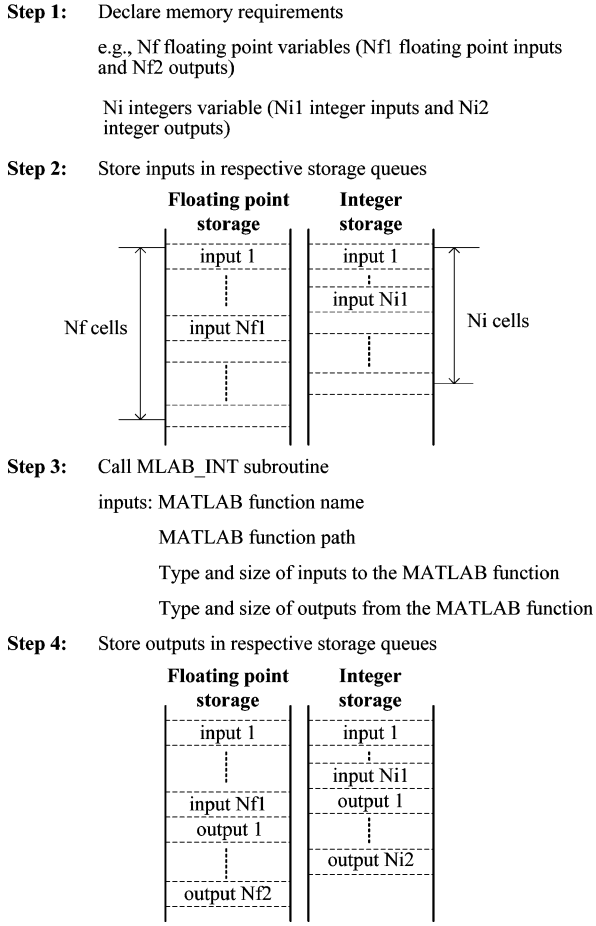

Fig. 6. Sequence of events in the EMTDC/MATLAB interface.

Upon completion of the tasks in MATLAB (this can include some SIMULINK models as well), the `MLAB_INT` returns the outputs to the remaining memory locations assigned for output storage. At this point, the transient simulator is able to access and read the outputs.

Note that the respective MATLAB function may: 1) contain user-developed algorithms, 2) call built-in MATLAB functions, or 3) setup and call SIMULINK. Examples of such possibilities are presented in Section VIII where some interfacing cases are discussed.

B. Wrapper Interfacing: Run Controllers and Optimization

Transient simulation tools are sometimes used in studies where multiple simulations are conducted. A number of parameters in the simulated network are varied sequentially or randomly (with a given distribution) and simulations are done in order to assess the impact of such parameter variations on the simulation results. Simulation results for a given set of parameter values are often distilled into a small number of indices that represent a figure of merit for the parameters used. For example, severity of a lightning-strike fault as a function of its location can be examined by conducting multiple simulations in which the fault location is varied along a given transmission line

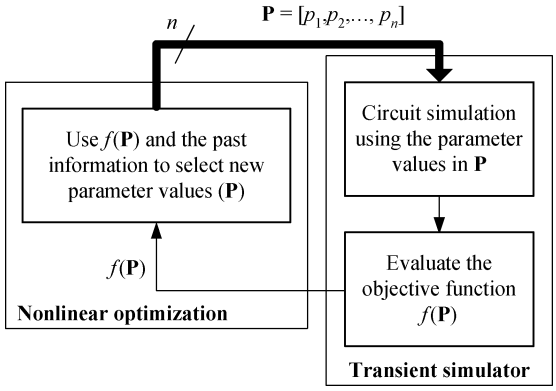

Fig. 7. Optimization interfacing.

and the magnitude of the resulting voltage surge is recorded. EMTP-type tools often provide built-in engines for conducting multiple simulations using specified parameter variations [19].

The so-called multiple-run simulation can be described as in the following algorithm (Algorithm 2). As shown, the multiple-run algorithm is responsible for: 1) selecting suitable parameter values according to the specified parameter variation rule, 2) feeding the simulation with the parameters, and 3) recording the respective figure of merit for further processing.

ALGORITHM 2—MULTIPLE-RUN SIMULATIONS

```
1 create a set of parameters Pi (a total of N
points representing parameter combinations)
2 i = 1
3 if i <= N
4  run the simulator with parameter set P(i)
5  record the figure-of-merit for P(i)
6  i = i + 1
7 End
```

Note that the preselection of parameter combinations in a sequential multiple-run, or random selection with a given distribution resembles a passive approach to the true potential of the multiple-run algorithm. Through proper interfacing, however, one can design more advanced run-control algorithms to conduct multiple simulations in ways other than the conventional approach, where it is possible to steer future simulations based on the outcome of current and past simulations.

1) Interfacing to an Optimization Algorithm: An example of an enhanced multiple-run algorithm without a predetermined set of search parameters is the optimization facility in a transient simulation program. Some EMTP-type programs have incorporated nonlinear optimization as an integral part of the simulation suite [19], [42]. Here, a nonlinear optimization algorithm can be called to replace the conventional multiple-run algorithm to conduct several simulations, each with a new set of parameters determined by the optimization algorithm according to the collective history of past simulations [32]. Fig. 7 shows a schematic diagram of the simulator–optimization interface.

Note that unlike a conventional multiple-run simulation, the nonlinear optimization algorithm decides, based on the objective function value it receives from the simulation program and the history of past simulations, what parameter values need to be generated and submitted to the simulation program for the next simulation.

```
...
1       initialize P1 and P2
2       if f(P1)>f(P2)
3               % use the following code (A)
4               % for generation of new points
                ...
5       else
6               % use the following code (B)
7               % for generation of new points
                ...
        ...
```

Fig. 8. Excerpt of a generic optimization algorithm.

An optimization engine implemented in ETMDC is interfaced internally and is used to link a number of optimization algorithms to the simulation engine, including nonlinear simplex of Nelder and Mead [43], genetic algorithms, and a number of gradient-based algorithms [44]. The choice of an internal interface has been made to remove potential speed reductions and to expedite the design cycle that normally involves several simulations. There are, however, advanced optimization algorithms that make use of complex functions that are available in mathematical software such as MATLAB, and their implementation is either excessively complicated or not possible due to the confidentiality of the code. For such cases, external interfacing between the simulation program and MATLAB has been used [45]. In this case, the interface resembles a master/slave relationship, where the host of the optimization algorithm (MATLAB) also serves as the master and calls upon the slave (transient simulation tool) for objective function evaluation.

Internal interfacing of optimization algorithms, however, embeds some challenges that need detailed knowledge of the mechanics of the simulation tool. Consider optimization of the function $f(\mathbf{P})$, whose evaluation is done through simulation (refer to Fig. 7 for details). A heuristic optimization algorithm often compares objective function evaluations at a number of points within the algorithm in order to generate new parameter values. Therefore, a portion of such an optimization algorithm may be as shown in Fig. 8.

After initializing two sets of parameter values ($\mathbf{P}1$ and $\mathbf{P}2$ in line 1), the algorithm needs to compare their corresponding objective functions (line 2). However, each of the two values $f(\mathbf{P}1)$ and $f(\mathbf{P}2)$ are obtained by transient simulation of the network using the simulation tool itself, which requires two complete simulations. In other words, the optimization, embedded within and called by the simulation tool, will call the simulation tool at several locations, thus creating a nested loop. Appropriate use of flags and control statements ensures that sequence of the execution of statements in the simulation program is transferred to the exact location where it was interrupted.

2) Advanced Run Controllers: Aside from optimization, there are other types of multiple-run controllers that use the concept of adaptive steering of simulations. An example of this is reported in [46] where an adaptive, internally interfaced search algorithm is used for finding the largest value of an inductor and its critical point-on-wave switching instant that result in no commutation failure at the terminals of an HVDC converter.

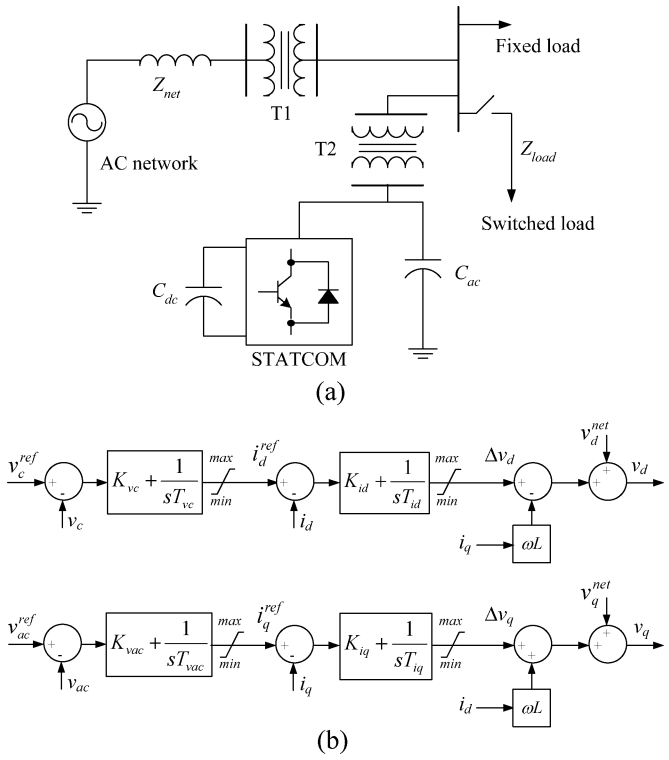

Fig. 9. Network schematic diagram and the STATCOM controls. (a) Schematic diagram of the network. (b) STATCOM control system.

Similar interfaces have been made for calculation of sensitivity indices of a given performance index [44]. In all cases indicated before, the choice of internal interfacing has been due to the simulation time restrictions.

VIII. Interfacing Case Studies

This section presents examples of interfacing methods discussed in this paper. An ac system equipped with a static compensator (STATCOM) is selected to serve as the basis for all of the interfacing instances shown. Several interfacing possibilities are explored between the transient simulator and other mathematical tools for purposes, such as data visualization and controller implementation. A model of the entire network is developed in the transient simulation program (EMTDC) and is used as a reference for validation of the results obtained from other models made through interfacing EMTDC with other tools/algorithms.

A. Description of the Network and Associated Controls

Fig. 9(a) shows the schematic diagram of the network under consideration. The STATCOM is connected at the terminals of the load supplied by the network, and is used for regulating the load terminal voltage during load variations. The control system structure for the STATCOM is shown in Fig. 9(b), in which outer ac voltage (v_{ac}) and dc capacitor-voltage (v_c) loops drive inner-loop, decoupled controllers for the current components (i_d and i_q) in a synchronously rotating dq-frame. The resulting voltage components (v_d and v_q) are then used by a pulse-width modulator (PWM) to synthesize STATCOM terminal voltages [47].

TABLE I
STATCOM and AC System Specifications

AC system		
Rated voltage: 115 kV	Frequency: 60 Hz	Transformer (T1): 115 kV/20 kV, 30 MVA, 8%
STATCOM		
Switching frequency: 1980 Hz		Rated MVA: 10
i_d current limits: [-0.1,0.1] kA		Transformer (T2):
i_q current limits: [-0.45,0.45] kA		3.3 kV/20 kV, 10 MVA, 14%
DC capacitor (C_{dc}): 1950 μF		
AC capacitor bank (C_{ac}): 49.7 μF (7.5 MVAR)		
Loads		
Fixed load: 0.1 MW, (unity power factor)		Switched load: 20 MVA, 0.9 PF

Space-vector modulation (SVM) [48] is used for the generation of the firing pulses given to the STATCOM voltage-source converter (VSC). Other system specifications are given in Table I.

B. Simulation Data Visualization Using MATLAB

Transient simulation tools provide the user with some level of ability for data visualization (e.g., generation of graphs of the simulated data). Advanced data visualization, available in mathematical tools, such as MATLAB, however, is often far more diverse and flexible. The generation of 3-D graphs and phasor diagrams overlaid with other graphical elements are examples of tasks that can be completed with ease in an advanced graphing tool.

In the following section, an interface between the transient simulator and MATLAB is described, which is used for the generation of dynamically changing phasor representations of the voltages generated by the STATCOM under space-vector modulation (SVM) [48].

1) SVM Implementation and Visualization in MATLAB: In a two-level converter, there are eight converter states (six active and two that are zero) based on the ON/OFF status of its controlled switches. Space-vector modulation places the converter in a combination of states so that the synthesized voltage approximates the desired output voltage waveform. Dynamic visualization of the SVM is a useful method for inspecting its inner workings and can also serve as an educational tool. It is desired that the active space vectors, the current sampled reference vector, and the two nonzero space vectors used for its composition be shown on a graph that is updated as the simulation progresses and the reference vector rotates.

Internal interfacing has been used for creation of an SVM-based firing pulse generator. Subsequently, interfacing to MATLAB has been used for the purpose of visualization of the SVM-generated converter states. The information about the current state of the converter, the sampled reference vector (its phase and magnitude), and the two nonzero vectors plus the modulation index and the available dc voltage are passed to MATLAB. The interfaced MATLAB script draws a hexagon with six vectors pointing toward its vertices. It also draws the employed vectors in a distinct color, whose magnitudes are proportional to their time shares. The actual sampled reference vector and its locus are also plotted.

Fig. 10 shows two snapshots of a graph generated by MATLAB using the information provided by the transient

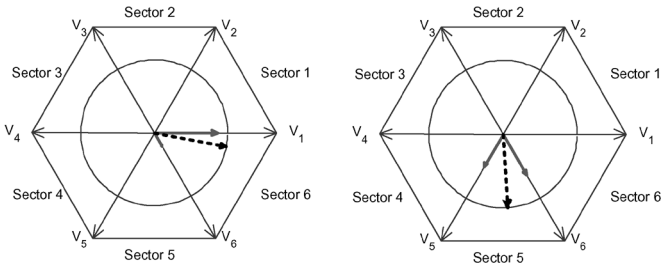

Fig. 10. Snapshots of the SVM visualization output in MATLAB.

simulator. At the given instants of time, the reference vectors and their projections onto respective space vectors are shown.

2) Interfacing Aspects: Since this type of interfacing only transmits data from the main simulator (EMTP-type program) to the visualization platform (MATLAB) and does not involve any storage or manipulation of the data, it is categorized as a static, external, core-type interface. Moreover, the data are transmitted (intermittently) throughout the simulation, making it an online interface.

C. Modeling of Control Systems in MATLAB/SIMULINK

In this section, the use of (external) interfacing for modeling of control systems in MATLAB/SIMULINK is demonstrated. The intention is to highlight the interfacing aspects and as such, two simple scenarios are considered for implementation of the STATCOM control system shown in Fig. 9(b): 1) modeling entirely in MATLAB, and 2) modeling in MATLAB and SIMULINK. The control system is then interfaced with the power circuits that are modeled in the main transient simulator. A separate model developed entirely in the transient simulator is used as the basis for validation.

1) PI Controller Implementation in MATLAB: Control algorithms can be directly coded in MATLAB and called for generation of controller output(s). Consider, for example, the implementation of the ac voltage controller (proportional gain K_{vac}, integral time constant T_{vac}) in Fig. 9(b). Its implementation in MATLAB involves the development of the integrator and proportional actions and imposing the limits (as given in Table I). Limits similar to those used for the entire PI block are also imposed on the integrator. This ensures that the integrator does not plunge deeply into saturation, and expedites the response time of the PI controller. The integrator action is implemented using trapezoidal integration method. A component is developed that accepts the ac voltage magnitude error (as its input) and communicates it along with the current simulation time and time step with a MATLAB function that implements the PI action.

Similar to the procedure outlined in Fig. 6, the interface component stores the input variables in the respective queue, called the `MLAB_INT` subroutine, which, in turn, runs the MATLAB function for the PI action, and retrieves the PI controller output from the respective memory location. Fig. 13 shows traces of the ac voltage variations when the network is subjected to a load disconnection at 0.4 s followed by its reconnection at 0.6 s. The traces are shown for both the standard PI controller block of the EMTDC as well as the PI controller implemented in MATLAB and interfaced with the transient simulator. As expected, the waveforms are identical and overlap.

Step 1: If $t = 0.0$ sec:
Set all the initial states to zero

Step 2: 1) Set the simulation start time to the current time (t)
2) Set the simulation end time to $t+\Delta t$
3) Execute the SIMULINK model for the given states, simulation start time and simulation end time

Step 3: Save the final state values to the initial state values for the next time step

Fig. 11. Sequence of events in the SIMULINK interface.

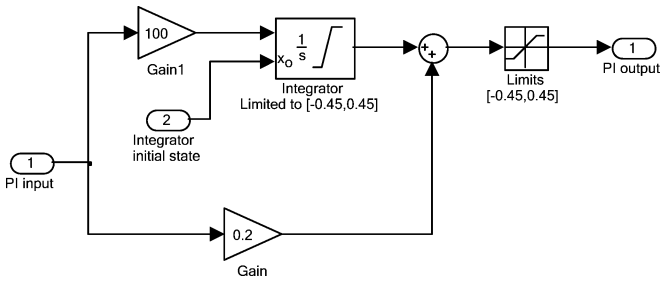

Fig. 12. SIMULINK model of the ac voltage PI controller.

2) Control System Implementation in MATLAB/SIMULINK: As the complexity of systems to be modeled in MATLAB (or other external agents) increases, it becomes increasingly difficult to write detailed codes for calculations or controller actions as was done in the previous example. Use of SIMULINK, which offers a graphical interface for development of complex interconnected systems, can highly simplify this task. An implementation of the PI controller in the previous section is thus undertaken using the SIMULINK.

For this purpose, an interface to MATLAB is developed, which in turn sets the simulation and workspace parameters for a SIMULINK model of the PI controller and invokes SIMULINK for the actual simulation. Communication of the inputs and outputs to and from MATLAB follows the sequence depicted in Fig. 6. The sequence of events in the SIMULINK interface is shown in Fig. 11. Note that the SIMULINK model is executed only for the interval $[t, t + \Delta t]$ each time it is invoked. Fig. 12 shows the developed SIMULINK model of the ac voltage PI controller.

The latest state of the integrator needs to be saved at the end of the $[t, t + \Delta t]$ interval, so that it can be used as the initial state for the next time step; therefore, an external input is added to the integrator block (input port 2 in Fig. 12) so that the correct initial state can be fed into the integrator. Similar to the original PI controller in EMTDC and the one coded in MATLAB in the preceding section, identical lower and upper limits (-0.45 and 0.45, respectively) are applied to both the integrator and the entire PI block outputs. The SIMULINK model shown in Fig. 12 has been interfaced with the transient simulation model of the STATCOM system and has been tested for the same set of disturbances described in the previous section. Fig. 13 shows the simulation results obtained, which are identical to the ones depicted for the original model and the MATLAB-interfaced model (both models start from zero initial conditions).

3) Computer Time and Interfacing: As indicated earlier, externally interfaced tools tend to be slower than their internal

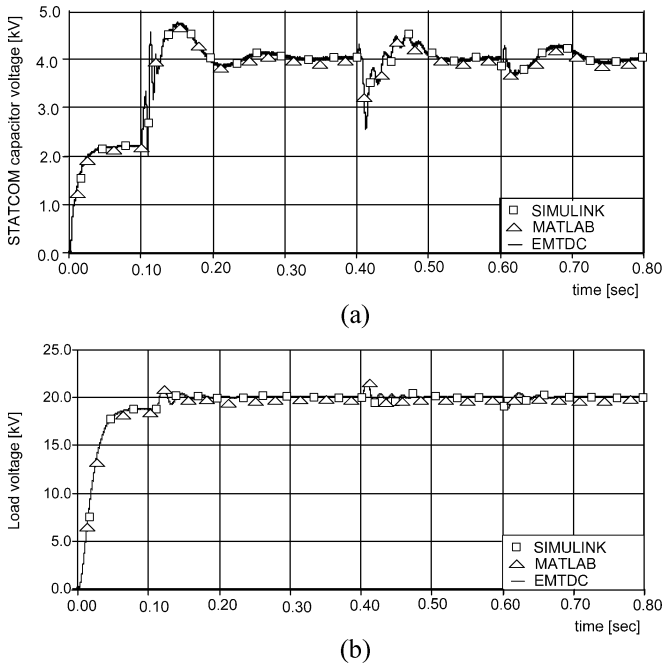

Fig. 13. Simulated waveforms for various network modeling approaches. (a) Capacitor voltage waveform. (b) Load terminal voltage waveform.

TABLE II
COMPARISON OF SIMULATION TIME FOR THE STATCOM CASE

EMTDC Model	EMTDC interfaced with MATLAB	EMTDC interfaced with MATLAB/SIMULINK
32.3 sec	1258.3 sec	3531.5 sec

counterparts. An audit of the simulation time often reveals important information about the intensity of interfacing. Table II shows computer times used for the simulation of the STATCOM system in the EMTDC, EMTDC interfaced with MATLAB and EMTDC interfaced with SIMULINK. The simulation time interval is 0.8 s and is traversed using a 5 μs time step and 100 μs plot step on a Pentium 4, 3.0 GHz machine with 1.0 GB of RAM.

The interfaced models in MATLAB and MATLAB/SIMULINK are slower than the original transient simulation model, as evidenced by their respective computer times of 1258.3 s and 3531.5 s as opposed to 32.3 s in the EMTDC. This shows simulation times in excess of 1.5 and 2.0 orders of magnitude slower than the original model, respectively. It is therefore advisable that interfacing be used only when such prolonged simulations times are offset by the benefits of the tool to which the interface is established.

D. Optimization and Sensitivity Analysis

1) Wrapper Interfacing for Gradient-Based Optimization: An optimization algorithm adopts an iterative approach to find the optimum point of a function. By defining a suitable objective function and use of an optimization algorithm to control the simulation runs, it is possible to adjust the design parameters of a circuit so that a given set of design objectives is met as closely as possible.

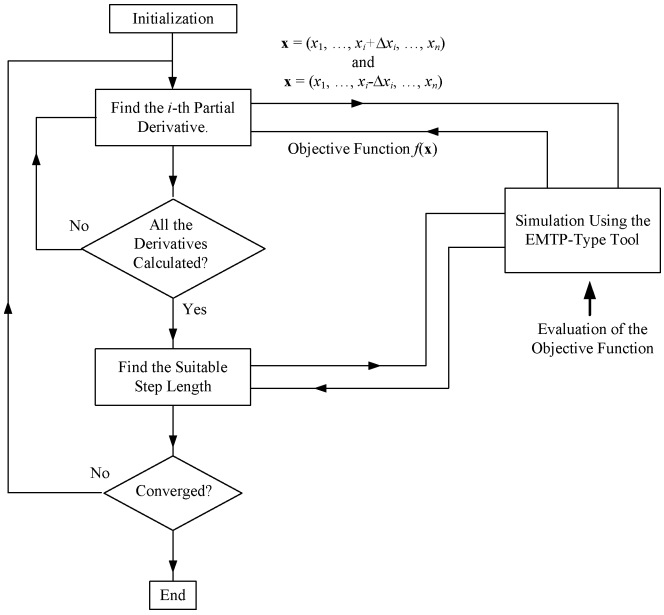

Fig. 14. Simulation-based Fletcher–Reeves optimization tool.

This section considers a gradient-based technique known as the Fletcher–Reeves method [49], and uses it for optimization of the control system parameters of the STATCOM in Fig. 9(b).

The block diagram shown in Fig. 14 summarizes the steps involved in the simulation-based optimization using the Fletcher–Reeves method. As shown, the optimization algorithm interacts with the simulator in two stages. The first stage is devoted to the calculation of partial derivatives, during which incremental changes are made successively to the parameter values, and the corresponding objective function values are calculated by running the simulation program. For an N-variable optimization problem, this stage requires $2N$ simulations. During the second stage, the simulation program is run a few times until a suitable step length (one which results in a reduction in the value of the objective function) is obtained. The algorithm has been implemented in FORTRAN, and is used as a wrapper interface (internally developed) to the transient simulation program.

The optimization tool is used to obtain optimal values for the parameters of the control system shown in Fig. 9(b). The goal of the optimization is to obtain smooth transitions in the network voltage and the dc capacitor voltage when the network is subjected to a load rejection/reconnection disturbance. The objective function used is as follows:

$$f(\mathbf{x}) = K_1 \left\| v_{ac} - v_{ac}^{ref} \right\| + K_2 \left\| v_c - v_c^{ref} \right\|. \qquad (1)$$

The objective function in (1) approaches zero as the network voltage and the dc capacitor voltage stay tightly close to their reference values during transients and in steady state. Fig. 15 shows the variations of the dc-capacitor voltage before and after optimization. The disturbances applied are a load rejection at 1.0 s followed by its reconnection at 1.5 s. AC voltage variations are shown in Fig. 16. Both figures show significant improvement in the performance of the system as a result of optimization.

The aforementioned optimization process takes about 300 simulation runs to converge to an optimum point in about 5 h on

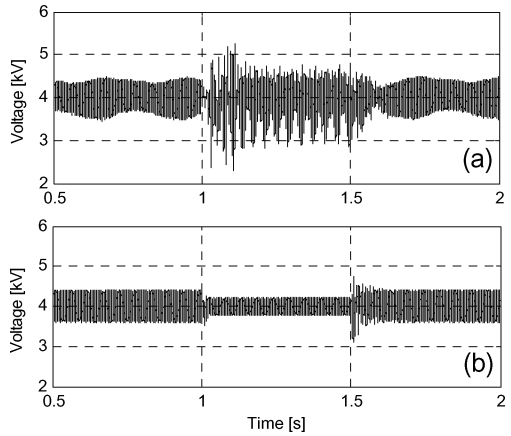

Fig. 15. DC capacitor voltage. (a) Before and (b) after optimization.

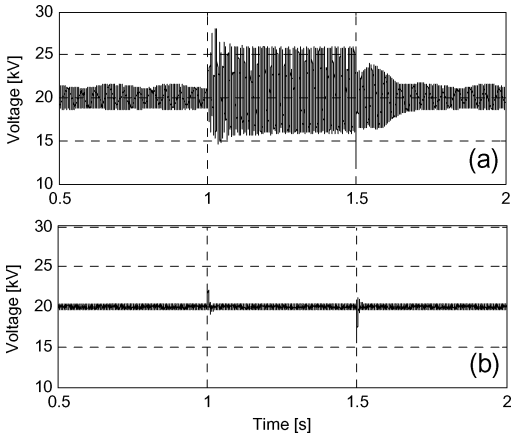

Fig. 16. Network voltage. (a) Before and (b) after optimization.

a 1.7 GHz Intel Celeron processor. Note that the number of simulation runs required also depends on the optimization starting point (i.e., depending on how far the actual (local) optimum is from the starting point), the number of simulations required will change accordingly.

2) Internal Wrapper Interface for Sensitivity Analysis: Due to such factors as operating conditions and aging, circuit parameters may deviate from their original values over time. It is therefore important to evaluate the performance of a design when parameter values are subject to variations from their nominal values (i.e., to perform sensitivity analysis).

Interfacing an electromagnetic transient simulator with a mathematical algorithm for sensitivity analysis can facilitate the task by allowing (automatic) calculation of the partial derivatives of the desired performance index (f) with respect to the design parameters (x_i). The partial derivatives can then be used to estimate the variations in the performance function as follows:

$$\Delta f = \frac{\partial f}{\partial x_1}\Delta x_1 + \cdots + \frac{\partial f}{\partial x_n}\Delta x_n. \tag{2}$$

Note that the calculation of the derivatives follows the multiple–simulation approach outlined in Section VII; therefore, a wrapper interface that feeds the simulator with proper parameter values is developed that performs multiple simulations and calculates the derivatives.

TABLE III
SENSITIVITY INDICES OF THE NETWORK USING SIMULATION

DC capacitor	AC capacitor bank	Load Impedance	AC network impedance	Converter transformer
$\frac{\partial T_s}{\partial C_{dc}}.C_{dc}$	$\frac{\partial T_s}{\partial C_{ac}}.C_{ac}$	$\frac{\partial T_s}{\partial Z_{load}}.Z_{load}$	$\frac{\partial T_s}{\partial Z_{net}}.Z_{net}$	$\frac{\partial T_s}{\partial Z_{tran}}.Z_{tran}$
0.124	0.537	0 .102	4.440	0.298

In the following, sensitivity analysis as described before is used to assess the impact of component aging and load variations on the performance of the STATCOM control system. The settling time of the control system, defined as the time required for reaching and stabilizing in a band within 0.25% around the final steady-state operating point, is selected as the performance index. Sensitivity of the settling time (T_s) to variations in the ac and dc capacitor sizes (C_{ac} and C_{dc}), STATCOM transformer (T2) impedance (Z_{tran}), ac network impedance (Z_{net}), and load impedance (Z_{load}) is estimated using the developed wrapper interface.

Since five design parameters are considered, estimation of partial derivatives using (2) requires ten simulations, each with a given parameter deviation (positive and negative increments are considered for calculation of derivatives). Table III shows the estimated partial derivatives obtained using transient simulation of the network. As suggested by its large partial derivative, variations in the ac capacitor bank have a significant impact on the settling time of the control system, in contrast to the lower impact expected from variations in the dc capacitor and load impedance.

One important study that can be performed using sensitivity indices is the worst-case scenario (i.e., to determine the largest possible deviation in the performance index for given tolerances in the design parameters). The worst-case (longest) settling time of the STATCOM control system when the five selected system parameters are allowed to vary within a given range (denoted by x in percent) can be obtained by using (3)

$$T_{sw} = \left|\frac{\partial T_s}{\partial C_{dc}}.C_{dc}x\right| + \left|\frac{\partial T_s}{\partial C_{ac}}.C_{ac}x\right| + \left|\frac{\partial T_s}{\partial Z_{load}}.Z_{load}x\right| + \left|\frac{\partial T_s}{\partial Z_{net}}.Z_{net}x\right| + \left|\frac{\partial T_s}{\partial Z_{tran}}.Z_{tran}x\right| + T_{s0} \tag{3}$$

where T_{sw} is the worst-case settling time and T_{s0} is the original settling time for the original parameter settings. The settling time of the STATCOM control system for the original parameters values given in Table I is equal to 0.125 s, as evidenced by response of the load ac voltage shown in Fig. 17(a). For a small tolerance of 2.5% in all five selected design parameters, the worst-case analysis using (5) with the numerically estimated partial derivatives in Table III, yields a settling time of 0.262 s. The actual simulation result obtained for the worst-case combination of the design parameters is shown in Fig. 17(b), and shows a settling time of 0.263 s, which agrees well with the estimated worst-case value.

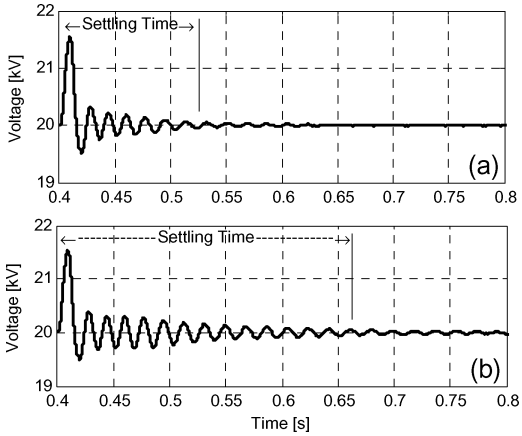

Fig. 17. AC voltage waveform for (a) the original system parameters, and (b) the worst-case combination of design parameters with 2.5% tolerance.

IX. Other Interfacing Opportunities

The majority of interfacing instances most likely fall within one of the categories described previously in this paper. There are, however, other instances where interfacing becomes important. Two such cases are described below.

A. Interfacing With External Hardware

Transient simulation tools provide valuable information about short-term dynamics of a power system. For example, such detailed information becomes particularly useful when tuning and designing protection systems, which are required to respond in a timely fashion to faults and other maloperations in a network.

Although conventional transient simulation tools do not have synchronized and real-time capabilities (except dedicated real-time simulators [50], [51]), their simulated waveforms can be recorded, and then played back in real time into external protection hardware using amplifiers of appropriate bandwidth, and voltage and current ratings in order to test the performance of the protective relaying equipment in response to simulated faults.

Appropriate interfaces exist to enable recording and playing back of the simulated waveforms. Standard formats (e.g., COMTRADE [52]), exist to facilitate communication between the simulation tool and the external hardware. Note that playing back the simulated waveforms onto a protective relay can be used to investigate the behavior of the relay; however, since no direct connection exists between the relay under test and the simulator, the scheme does not form a closed-loop simulator in which simulation continues following the relay operation. Interfacing techniques for hardware applications become more relevant with regards to real-time simulators; however, this falls beyond the scope of this paper.

B. Distributed Simulation and Co-Simulation

Transient simulation of large power networks requires massive computer facilities. It is therefore beneficial if simulations could be partitioned or performed on platforms comprising several processing units.

To take advantage of the distributed simulation, it is first necessary to partition the original network into two or more subsystems. The partitioning can happen, for example, at convenient boundaries involving natural delays such as transmission lines; more complicated systems with electrical networks, power electronic converters, electrical machines, controllers, mechanical drive trains, prime movers, etc., may also be partitioned at some convenient structural (or component-based) boundaries. In each case, the smaller subsystems will have to interface with each other and communicate respective coupling variables (uni or bidirectionally) as necessary.

In general, the subsystems may be tackled using different EMTP-type programs, which then have to be interfaced with each other in order to pass the respective coupling variables. Many simulation programs allow user-defined code (e.g., C and/or FORTRAN) and permit calling other programs that execute on the same computer [53], [54].

As the systems to be simulated grow in size, so does the time required for computing the corresponding time-domain transient response. Although the speed of modern computers constantly increases, the computing time continues to be a critical issue in the simulation of many practical systems when only one computer is utilized. When the number of subsystems increases, it may be further advantageous to execute the respective simulations on different computers and interface them across a network (e.g., TCP/IP). The approach of distributed heterogeneous simulation (DHS) [55], [56] views the overall system model as a collection of interconnected subsystems. Each subsystem may be solved using an independent EMTP-type program (nodal analysis- or state variable-based) using its own time step. The interactions between subsystems are represented through the exchange of interface variables. This approach has been used to simulate naval integrated power system [56] and electrical power system of an aircraft [57].

X. Conclusion

This paper described methods for interfacing EMTP-type programs with general mathematical tools. Static, dynamic, and wrapper interfaces were introduced as the three significant categories into which the majority of interfacing instances fall. External and internal interfacings were introduced as two major options for the implementation of an interface. It was shown that internal interfacing provides fast simulation; whereas external interfacing allows for incorporation of a large collection of pre-made algorithms, albeit at the expense of prolonged simulation time.

The paper also described two advanced instances of interfacing, namely the MATLAB interface and optimization and run-control interfaces. In both cases details about memory management, data exchange, and interactions involved were provided. The examples included in the paper demonstrated the use of interfacing for a variety of applications. Data visualization using MATLAB, controller implementation in MATLAB/SIMULINK, and optimization and sensitivity analysis were described along with discussion about the interfacing requirements, simulation time constraints, and potential benefits and drawbacks.

REFERENCES

[1] H. W. Dommel, "Digital computer solution of electromagnetic transients in single- and multiphase networks," *IEEE Trans. Power App. Syst.*, vol. PAS-88, no. 4, pp. 388–399, Apr. 1969.

[2] O. Nayak, G. Irwin, and A. Neufeld, "GUI enhances electromagnetic transients simulation tools," *IEEE Trans. Comput. Appl. Power*, vol. 8, no. 1, pp. 17–22, Jan. 1995.

[3] J. Mahseredjian, S. Lefebvre, and D. Mukhedkar, "Power converter simulation module connected to the EMTP," *IEEE Trans. Power Syst.*, vol. 6, no. 2, pp. 501–510, May 1991.

[4] A. M. Gole, P. Demchenko, D. Kell, and G. D. Irwin, "Integrating electromagnetic transient simulation with other design tools," presented at the Int. Conf. Power System Transients, 1999.

[5] T. Grebe and S. Smith, "Visualize system simulation and measurement data," *IEEE Comput. Appl. Power*, vol. 12, no. 3, pp. 46–51, Jul. 1999.

[6] J. Mahseredjian, "Merging, prototyping and hybrid tools for power system transient simulation," in *Proc. Power Eng. Soc. Summer Meeting*, Jul. 2000, vol. 2, pp. 768–769.

[7] X. Wang, P. Wilson, and D. Woodford, "Interfacing transient stability program to EMTDC program," in *Proc. Int. Conf. Power System Technology*, Oct. 2002, vol. 2, pp. 1264–1269.

[8] J. M. Zavahir, J. Arrillaga, and N. R. Watson, "Hybrid electromagnetic transient simulation with the state variable representation of HVDC converter plant," *IEEE Trans. Power Del.*, vol. 8, no. 3, pp. 1591–1598, Jul. 1993.

[9] D. A. Woodford, A. M. Gole, and R. W. Menzies, "Digital simulation of dc links and synchronous machines," *IEEE Trans. Power App. Syst.*, vol. PAS-102, no. 6, pp. 1616–1623, Jun. 1983.

[10] D. A. Woodford, "Electromagnetic design considerations for fast acting controllers," *IEEE Trans. Power Del.*, vol. 11, no. 3, pp. 1515–1521, Jul. 1996.

[11] N. Kolcio, J. A. Halladay, G. D. Allen, and E. N. Fromholtz, "Transient overvoltages and overcurrents on 12.47 kV distribution lines: Computer modeling results," *IEEE Trans. Power Del.*, vol. 8, no. 1, pp. 359–366, Jan. 1993.

[12] Q. Bui-Van, G. Beaulieu, H. Huynh, and R. Rosenqvist, "Overvoltage studies for the St. Lawrence River 500-kV DC cable crossing," *IEEE Trans. Power Del.*, vol. 6, no. 3, pp. 1205–1215, Jul. 1991.

[13] A. K. S. Chaudhary, K.-S. Tam, and A. G. Phadke, "Protection system representation in the electromagnetic transients program," *IEEE Trans. Power Del.*, vol. 9, no. 2, pp. 700–711, Apr. 1994.

[14] S. Jiang, U. D. Annakkage, and A. M. Gole, "Platform for validation of FACTS models," *IEEE Trans. Power Del.*, vol. 21, no. 1, pp. 484–491, Jan. 2006.

[15] M. Saeedifard, H. Nikkhajoei, and R. Iravani, "A space vector modulation approach for a multimodule HVDC converter system," *IEEE Trans. Power Del.*, vol. 22, no. 3, pp. 1643–1654, Jul. 2007.

[16] A. M. Gole and M. Meisingset, "An AC active filter for use at capacitor commutated HVDC converters," *IEEE Trans. Power Del.*, vol. 16, no. 2, pp. 335–341, Apr. 2001.

[17] C. K. Sao, P. W. Lehn, and M. R. Iravani, "A benchmark system for digital time-domain simulation of a pulse-width-modulated D-STATCOM," *IEEE Trans. Power Del.*, vol. 17, no. 4, pp. 1113–1120, Oct. 2002.

[18] W. Long, D. Cotcher, D. Ruiu, P. Adam, S. Lee, and R. Adapa, "EMTP—A powerful tool for analyzing power system transients," *IEEE Trans. Comput. Appl. Power*, vol. 3, no. 3, pp. 36–41, Jul. 1990.

[19] *"EMTDC Manual,"* Manitoba HVDC Research Centre, Apr. 2004.

[20] L. A. Dessaint, K. Al-Haddad, H. Le-Huy, G. Sybille, and P. Brunelle, "A power system simulation tool based on Simulink," *IEEE Trans. Ind. Electron.*, vol. 46, no. 9, pp. 1252–1254, Dec. 1999.

[21] A. M. Gole, I. T. Fernando, G. D. Irwin, and O. B. Nayak, "Modeling of power electronic apparatus: Additional interpolation issues," in *Proc. Int. Conf. Power System Transients*, Seattle, WA, Jun. 1997, pp. 23–28.

[22] N. Watson and J. Arrillaga, *Power Systems Electromagnetic Transients Simulation*, ser. Power Energy. London, U.K.: Inst. Elect. Eng., 2002.

[23] L. Dube and H. W. Dommel, "Simulation of control systems in an electromagnetic transients program with TACS," in *Proc. IEEE PICA*, 1977, pp. 266–271.

[24] J. Mahseredjian, G. Benmouyal, X. Lombard, M. Zouiti, B. Bressac, and L. Gerin-Lajoie, "A link between EMTP and MATLAB for user-defined modeling," *IEEE Trans. Power Del.*, vol. 13, no. 2, pp. 667–674, Apr. 1998.

[25] M. Kezunovic and Q. Chen, "A novel approach for interactive protection system simulation," *IEEE Trans. Power Del.*, vol. 12, no. 2, pp. 668–674, Apr. 1997.

[26] L. X. Bui, S. Casoria, G. Moin, and J. Reeve, "EMTP TACS-FORTRAN interface development for digital controls modeling," *IEEE Trans. Power Syst.*, vol. 7, no. 1, pp. 314–319, Feb. 1992.

[27] J. Reeve and S. P. Lane-Smith, "Integration of real-time controls and computer programs for simulation of direct current transmission," *IEEE Trans. Power Del.*, vol. 5, no. 4, pp. 2047–2053, Nov. 1990.

[28] G. Wild, H. Messner, A. Moosburger, M. H. Xie, A. M. Gole, and D. P. Brandt, "An integrated simulation and control implementation environment," presented at the Int. Conf. Power System Transients, Seattle, WA, Jun. 1997.

[29] A. M. Gole and A. Daneshpooy, "Toward open systems: A PSCAD/EMTDC to MATLAB interface," presented at the Int. Conf. Power System Transients, Seattle, WA, Jun. 1997.

[30] K. Strunz and E. Carlso, "Nested fast and simultaneous solution for time-domain simulation of integrative power-electric and electronic systems," *IEEE Trans. Power Del.*, vol. 22, no. 1, pp. 277–287, Jan. 2007.

[31] M. Sczechtman, T. Wess, and C. V. Thio, "First benchmark model for HVDC control studies," *Electra*, no. 135, pp. 54–73, Apr. 1991.

[32] A. M. Gole, S. Filizadeh, R. W. Menzies, and P. L. Wilson, "Optimization-enabled electromagnetic transient simulation," *IEEE Trans. Power Del.*, vol. 20, no. 1, pp. 512–518, Jan. 2005.

[33] M. O. Faruque, Y. Zhang, and V. Dinavahi, "Detailed modeling of CIGRE HVDC benchmark system using PSCAD/EMTDC and PSB/SIMULINK," *IEEE Trans. Power Del.*, vol. 21, no. 1, pp. 378–387, Jan. 2006.

[34] W. Nye, D. C. Riley, A. Sangiovanni-Vincentelli, and A. L. Tits, "DELIGHT.SPICE: An optimization-based system for the design of integrated circuits," *IEEE Trans. Computer-Aided Design Integr. Circuits Syst.*, vol. 7, no. 4, pp. 501–519, Apr. 1988.

[35] H. Kragh, F. Blaabjerg, and J. K. Pedersen, "An advanced tool for optimized design of power electronic circuits," in *Proc. IEEE Industry Applications Conf.*, 1998, pp. 991–998.

[36] K. Rigbers, S. Schroder, T. Durbaum, M. Wendt, and R. W. De Doncker, "Integrated method for optimization of power electronic circuits," in *Proc. 35th Annu. IEEE Power Electronics Specialists Conf.*, 2004, pp. 4473–4478.

[37] A. R. Conn, P. K. Coulman, R. A. Haring, G. L. Morrill, C. Visweswariah, and C. W. Wu, "JiffyTune: Circuit optimization using time-domain sensitivities," *IEEE Trans. Computer-Aided Design Integr. Circuits Syst.*, vol. 17, no. 12, pp. 1292–1309, Dec. 1998.

[38] D. R. Northcott and S. Filizadeh, "Electromagnetic transient simulation of hybrid electric vehicles," presented at the IEEE Int. Symp. Industrial Electronics, Vigo, Spain, 2007.

[39] A. Mehrizi-Sani, S. Filizadeh, and P. L. Wilson, "Harmonic and loss analysis of space-vector modulated converters," presented at the Int. Conf. Power Systems Transients , Lyon, France, Jun. 2007.

[40] A. Chevrefils and S. Filizadeh, "Transient simulation of an ac synchronous permanent magnet motor drive for an all-electric all-terrain vehicle," presented at the Vehicle Power and Propulsion Conf., Arlington, TX, 2007.

[41] M. Kezunovic, J. Domaszewicz, V. Skendzic, M. Aganagic, J. K. Bladow, S. M. McKenna, and D. M. Hamai, "Design, implementation and validation of a real-time digital simulator for protection relay testing," *IEEE Trans. Power Del.*, vol. 11, no. 1, pp. 158–164, Jan. 1996.

[42] L. Xianzhang, E. Lerch, D. Povh, and B. Kulicke, "Optimization-a new tool in a simulation program system," *IEEE Trans. Power Del.*, vol. 12, no. 2, pp. 598–604, May 1997.

[43] J. A. Nelder and R. Mead, "A simplex method for function optimization," *Comput. J.*, vol. 7, no. 4, pp. 308–313, 1965.

[44] M. Heidari, S. Filizadeh, and A. M. Gole, "Support tools for simulation-based optimal design of power networks with embedded power electronics," *IEEE Trans. Power Del.*, accepted for publication.

[45] K. Kobravi, "Optimization-enabled transient simulation for design of power circuits with multi modal objective functions," M.Sc. dissertation, Dept. Elect. Comput. Eng., Univ. Manitoba, Winnipeg, MB, Canada, 2007.

[46] E. Rahimi, S. Filizadeh, and A. M. Gole, "Commutation failure analysis in HVDC systems using advanced multiple-run methods," presented at the Int. Conf. Power System Transients, Montreal, QC, Canada, 2005.

[47] C. Scahuder and H. Mehta, "Vector analysis and control of advanced static VAR compensators," *Proc. Inst. Elect. Eng., Gen. Transm. Distrib.*, vol. 140, no. 4, pp. 299–306, Jul. 1993.

[48] H. W. van der Broeck, H. C. Skudelny, and G. V. Stanke, "Analysis and realization of a pulsewidth modulator based on voltage space vectors," *IEEE Trans. Ind. Appl.*, vol. 24, no. 1, pp. 142–150, Jan./Feb. 1988.

[49] G. V. Reklaitis, A. Ravindran, and K. M. Ragsdell, *Engineering Optimization, Methods and Applications*. New York: Wiley, 1983.

[50] R. Kuffel, J. Giesbrecht, T. Maguire, R. P. Wierckx, and P. McLaren, "RTDS-a fully digital power system simulator operating in real time," in *Proc. Int. Conf. Energy Management and Power Delivery*, Nov. 1995, vol. 2, pp. 498–503.

[51] C. A. Rabbath, M. Abdoune, and J. Belanger, "Effective real-time simulations of event-based systems," in *Proc. Winter Simulation Conf.*, Dec. 2000, vol. 1, pp. 232–238.

[52] A. G. Phadke *et al.*, "COMTRADE: A new standard for common format for transient data exchange," *IEEE Trans. Power Del.*, vol. 7, no. 4, pp. 1920–1926, Oct. 1992, Working Group H-5 of the Relaying Channels Subcommittee of the IEEE Power Syst. Relaying Committee.

[53] R. Dougal, T. Lovett, A. Monti, and E. Santi, "A multilanguage environment for interactive simulation and development of controls for power electronics," in *Proc. IEEE Power Eng. Soc. Conf.*, Vancouver, BC, Canada, 2001, pp. 1725–1729.

[54] O. Haedrich and U. Knorr, "Electric circuit and control system simulation by linking Simplorer and Matlab/Simulink analysis of interactions of subsystems of modern electric drives," in *Proc. IEEE Workshop on Computers in Power Electronics*, Jul. 2000, pp. 192–196.

[55] J. Jatskevich, O. Wasynczuk, E. A. Walters, E. C. Lucas, and E. Zivi, "Real-time distributed simulation of a dc zonal electrical distribution system," in *Proc. SAE Power Systems Conf.*, Coral Springs, FL, Oct. 2002, pp. 3–10.

[56] C. E. Lucas, E. A. Walters, and J. Jatskevich, "Distributed heterogeneous simulation of naval integrated power system," presented at the Amer. Soc. Naval Engineers (ASNE), Electric Machine Technology Symp., Philadelphia, PA, Jan. 2004.

[57] C. E. Lucas, E. A. Walters, O. Wasynczuk, and P. T. Lamm, "Cross-platform distributed heterogeneous simulation of a more-electric aircraft power system," in *Proc. SPIE*, 2005, vol. 5805, pp. 328–336.