Automated state model generator for simulation and analysis of electric power systems

J. Jatskevich¹, O. Wasynczuk², C. E. Lucas² & E. A. Walters³

¹Electrical and Computer Engineering, University of British Columbia, Canada.
²Electrical and Computer Engineering, Purdue University, USA.
³Krause & Associates, Inc., USA.

Abstract

An algorithmic method of developing the state equations of complex power circuits and systems has recently been developed. In this approach, referred to as the Automated State Model Generator (ASMG), the system is described by the pertinent branch parameters and the circuit topology; however, unlike existing circuit-based approaches, the difference equations are not implemented at the branch level. Instead, the composite system state equations are established automatically and updated dynamically for each new topology of the switching network. Since the state equations are available, linearization, eigensystem analysis, and model-order reduction can be performed quickly and conveniently at the system level. In addition, it becomes possible to identify the operational modes of converters and inverters based on the cyclically repeated sequence of topologies. The ASMG includes a built-in switching logic for common power-electronic switching devices. The proposed simulation approach is particularly effective when modeling advanced power electronic systems that include special-purpose multi-phase electrical machines, transformers, and other components that are not available in the libraries of commonly used circuit simulators. An ASMG toolbox that is compatible with MATLAB/SIMULINK has been developed and used to implement a detailed simulation of a 6-phase generator/rectifier system.
1 Introduction

There are numerous computer programs that can be used to establish time-domain simulations of electric circuits and systems. Although each program has unique features, most can be placed into one of two general categories. In the first category, which includes programs such as PSpice [1] and Saber [2], EMTP [3], etc., that are based on nodal or modified-nodal analysis techniques implying discretization of differential equations at the branch level, from which the time-domain response is then established. The model developer may describe the circuit in the form of a network list or “netlist”, or graphically for example, defining the layout of the circuit in terms of parameters and the built-in or user-defined components. A significant advantage of these programs is that the model developer is not forced to derive the state equations or block-diagram representation of the circuit being simulated. However, a significant disadvantage is that a state-space model of the overall circuit is never established or calculated by the program. Consequently, numerous analysis and control techniques developed for the state-space models cannot be applied. Additionally, the use of standard library elements sometimes may be too restrictive when “nonstandard” circuit components such as mutual/nonlinear/variable inductances, capacitances, and/or resistances need to be implemented.

In the second category, which includes programs such as ACSL [4], Simulink [5], EASY 5 [6], etc., the underlying numerical algorithms are based on a state-space model. The system may be specified for example textually - using a specific syntax of the program being used, or graphically - using boxes or icons to represent summers, multipliers, function generators, integrators, transfer functions, etc. Thereafter, a system of differential and/or differential algebraic equations (DAEs) that constitute the state-variable-based model of the overall system is assembled. Depending upon the features of a given program, the DAEs may be converted into an equivalent system of ordinary differential equations (ODEs). The time-domain response is then calculated numerically using appropriate integration algorithms and/or solvers. Modeling of systems and devices based upon corresponding DAEs is perhaps one of the best ways to gain an analytical understanding of the physical system. However, the use of these programs can be especially challenging for modeling power-electronic circuits with a large number of possible topologies.

An automated state model generation (ASMG) algorithm was originally proposed in [7] and since then significantly enhanced [8]-[10]. The ASMG enables rapid development of models of power-electronic circuits without requiring the user to derive state-space equations. No parasitic elements are introduced to facilitate the switching. Consequently, spurious states are not introduced when simulating power-electronic-based switched circuits. The resulting models can then be used to study and analyze the dynamic characteristics using established state-space methodologies for stability analysis, control optimization, model-order reduction, etc. In this approach, a state-space model is established automatically based upon a netlist that defines the circuit. For systems with switches and/
or variable parameters, the state model is automatically re-assembled or updated as necessary. The time-domain response is obtained by numerically integrating the state equations using appropriate ODE solvers whereupon the branch currents and voltages are concatenated across switching boundaries. Using this modeling technique, it becomes possible to define the operational modes of complex switched circuits based upon the cyclically repeated sequence of topologies that can be observed during normal or abnormal operation.

2 State model and operational modes

Modeling of switched electrical systems involves the formulation and subsequent solution of the corresponding DAEs. In a generalized form of the algorithm, the circuit elements are grouped into structurally compact branches as depicted in Fig. 1. The first two branches are topological duals. Simple resistors, inductors, or sources can be represented by setting the appropriate parameters to zero. A switched electrical network constructed from a finite number of elementary branches can be defined by a corresponding graph \( G \), a set of branch parameters \( P \), and a topological state vector \( s \in \mathbb{S}^m \), where \( \mathbb{S} = \{0, 1\} \) (implying that each switch can be open or closed) and \( m \) is the number of switch branches [8]. Symbolically, a network object is defined as

\[
N = (G, P, s) \tag{1}
\]

where the graph is defined in terms of node (vertex) and branch (edge) sets as \( G = (N, B) \) [11]. The parameter set \( P \) can be expressed as

\[
P = \left\{ R_{br}, L_{br}, \frac{dL_{br}}{dt}, e_{br}, G_{br}, C_{br}, \frac{dC_{br}}{dt}, j_{br} \right\} \tag{2}
\]

where, for example, the diagonal elements of \( L_{br} \in \mathbb{R}^{n_B \times n_B} \) represent the branch.

![Branch models](image)

Figure 1: Branch models: (a) inductive, (b) capacitive, (c) switch.
inductances, and the off-diagonal elements, if nonzero, represent mutual inductive coupling between associated branches. The nonzero elements of \( e_{br} \in \mathbb{R}^{n_p} \) correspond to the values of the independent source \( e_{br} \) in Fig. 1(a) for those inductive branches whose independent sources are nonzero. In general, any of the parameters can vary with time; however, if the inductances and/or capacitances vary with time, their rates of change need to be defined in order to formulate a state model. Other matrices are defined similarly. Typically, the aforementioned matrices/vectors are sparse and only non-zero elements are stored [8],[10].

When formulating a state model for a given topological state \( s^i \), a composite electrical network is divided into tree- and link-branches. Therein, minimal sets of natural state variables; inductor link-currents and capacitor tree-voltages are readily identified. The result state equation has the following general form

\[
M^i(x^i, t) \frac{dx^i}{dt} = F(x^i, t) + g^i(u, t)
\]

(3)

where \( F(x^i, t) \) represents state-dependent terms; the forcing term \( g^i(u, t) \) takes all external sources and; \( M^i(x^i, t) \) is a positive definite matrix [9].

Over interval of time, the network switches sequentially from an initial topology \( s^0 \) to a final topology \( s^n \). The change in topology can be initiated by external control action or as the result of a current zero-crossing through a diode, for example. Symbolically, the sequence of topologies can be expressed

\[
\{s\}^n_0 = s^0, s^1, ..., s^i, ..., s^n
\]

(4)

For each topological instance \( s^i \) there exists a minimal state equation (3) with state vector \( x \in \mathbb{R}^{\alpha^i} \), where \( \alpha^i \) is the number of state variables. Overall, for the sequence (4), there is a corresponding sequence

\[
\{\alpha\}^n_0 = \alpha^0, \alpha^1, ..., \alpha^i, ..., \alpha^n
\]

(5)

For any two adjacent topologies \( s^i \) and \( s^{i+1} \), the corresponding minimal state equations will have different state vectors with possibly non-equal dimensions. In order to establish the overall response, the initial conditions for the subsequent topology is established in such a way that the currents through inductors and voltages across capacitors are continuous according with circuits laws. For sufficiently long studies, the sequence (4) typically will include cyclic patterns. In this case, it is possible to identify a subsequence of length \( p \)

\[
\{s\}_{i+p-1}^{i+p} = s^i, s^{i+1}, ..., s^{i+p-1}
\]

(6)

such that the switching pattern is repetitive, \( s^{i+p} = s^i \). Similarly, a repetitive subsequence of length \( q \) in (5) can be defined as
\[
\{\alpha\}_{i}^{i+q-1} = \alpha^i, \alpha^{i+1}, ..., \alpha^{i+q-1}
\]  

(7)

where \(\alpha^{i+q} = \alpha^i\).

It is important to note that the subsequence (7) may be shorter than (6), i.e. \(q \leq p\) with \(p\) being an integer multiple of \(q\). This typically occurs in multi-phase-ac/dc systems, where the ripple period of the dc variables is related to the subsequence (7) and the period of the ac variables is related to the longer subsequence (6). These subsequences define the operational mode of the circuit; moreover, they can be identified automatically by monitoring changes in \(s\) and \(\alpha\).

3 Toolbox

In general, the ASMG can be readily integrated within any differential-equation-based simulation language that allows calls to functions from a user-defined library. For use with Simulink [5], the ASMG is build as a masked CMEX S-function block that can be dragged from the Simulink Library and dropped into a new model. Although it is possible to model a complex circuit using one large branch list, it may be convenient to view the circuit as a collection of sub-circuits that are coupled through their inputs and outputs. In this case, several instances of the ASMG block may be used in one Simulink model.

A functional block diagram of the ASMG is shown in Fig. 2. The layout of the overall circuit is defined in an initialization file. In addition, any time-varying or dependent parameters and the external inputs such as \(e_{br}\) and/or \(j_{br}\) for those branches that have source terms must be declared. Based upon the user-supplied information, the ASMG generates a state model (3) that can be expressed in explicit form.

![ASMG functional block diagram](image-url)
\[
\frac{d\mathbf{x}}{dt} = f(\mathbf{x}, e_{br}, \mathbf{j}_{br}, t)
\]
(8)

\[
[\mathbf{i}_{br}, v_{br}, z_{xnp}]^T = \mathbf{g}(\mathbf{x}, e_{br}, \mathbf{j}_{br}, \mathbf{u}_{br}, t)
\]
(9)

where \(z_{xnp}\) is a variable that is monitored to identify switching events. An ODE solver integrates (8) to establish the state vector \(\mathbf{x}\), whereupon (9) is used to establish the vectors of output variables \(\mathbf{i}_{br}\) and \(v_{br}\). If the circuit includes variable inductances, capacitances, and/or resistances, (8) and (9) are automatically updated at run time without changing the structure of the system.

For circuits with switches, it is important to establish a precise time of a switching event. To accomplish this task, the ODE solver monitors and “zooms-into” negative-to-positive zero-crossings of the variable \(z_{xnp}\). If a zero-crossing is detected, the Discrete Event Processor performs the necessary updates of (8) and (9). Different logic may be specified to determine when a given switch should be opened or closed. In general, the switching logic does not permit the opening of switches that would cause discontinuities of currents in inductors and/or current sources, as well as closing of switches that would cause discontinuities of capacitor voltages and/or voltage sources. Four built-in switch types have been implemented so as to represent the idealized characteristics of diodes, thyristors, GTO, transistors (MOSFET, BJT, IGBT, etc.), triacs, arcing switches, ac breakers, etc. and to allow the user to quickly and easily model a wide variety of power electronic systems.

**Type-1: Unlatched bidirectional switch (UBS)**

UBS conducts current in either direction and may be open or closed by setting the control variable \(u_{br}\) to a positive or negative value, respectively.

**Type-2: Unlatched unidirectional switch (UUS)**

UUS conducts current only in positive direction. UUS closes when the variable \(e_{on} = \min(u_{br}, v_{br})\) crosses zero going negative-to-positive, and opens automatically when \(e_{off} = \min(u_{br}, i_{br})\) crosses zero going positive-to-negative.

**Type-3: Latched bidirectional switch (LBS)**

LBS can conduct current in either direction and may be closed by setting the control variable \(u_{br}\) to a positive value. LBS opens automatically at the next current-zero-crossing after \(u_{br}\) has been set to zero or negative.

**Type-4: Latched unidirectional switch (LUS)**

LUS conducts current only in positive direction. LUS closes when the variable \(e_{on} = \min(u_{br}, v_{br})\) crosses zero going negative-to-positive, and opens automatically when \(i_{br}\) crosses zero going positive-to-negative.

There are no parasitic elements or parameters that are introduced to facilitate switching or state model development. The user has a complete control over the
level of detail used to model a given circuit. If the idealized characteristics of switches are sufficient, the built-in switches can be used. Alternatively, the turn-on or turn-off transients can also be represented using behavioral circuit models [12]. "Spice-level" simulations can also be achieved by implementing the corresponding circuit models. However, this will result in a much slower simulation due to the added states and stiffness of the resulting ODEs. The user is ultimately responsible for the level of detail appropriate for the study objectives.

4 Example system

In order to demonstrate the ASMG features, an example system comprised of a 6-phase synchronous generator connected to two rectifiers and an interphase transformer is considered [13]. The circuit diagram of the stator network illustrating branch and node numbering is shown in Fig. 3. The stator windings are grouped as two sets of Y-connected 3-phase windings that are displaced from each other by 60 electrical degrees. The corresponding neutral points can be isolated or connected to each other. The system parameters summarized in Table 1 correspond to a 210-kW, 355-V (line-to-neutral), 240-Hz synchronous machine designed for naval application [13]. A synchronous machine with such particular design features is not likely to be available in the libraries of commonly used circuit simulators. Moreover, the possibility of connecting/disconnecting the generator neutrals results in different operational modes that are difficult to establish analytically. Therefore, this system represents a challenging example for modeling and analysis.

The synchronous machine is modeled in a so-called voltage behind-reactance (VBR) form [14] with dynamic saliency neglected [15]. This modeling technique provides a significant computational advantage and is equivalent to the Park’s representation over a wide range of frequencies. The relevant equations can be found in [15] and, therefore, are not repeated here. The ASMG was used to imple-
Table 1: System parameters.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Value</th>
<th>Branch(s)/Parameter</th>
<th>Symbol</th>
<th>Value</th>
<th>Branch(s)/Parameter</th>
</tr>
</thead>
<tbody>
<tr>
<td>( r_s )</td>
<td>0.114 ( \Omega )</td>
<td>16-21, stator phase winding resistance</td>
<td>( L_{ikd} )</td>
<td>0.650 mH</td>
<td>d-axis damper winding leakage inductance</td>
</tr>
<tr>
<td>( L_{ls} )</td>
<td>0.135 mH</td>
<td>16-21, stator phase winding leakage inductance</td>
<td>( r_{kd} )</td>
<td>0.118 ( \Omega )</td>
<td>d-axis damper winding resistance</td>
</tr>
<tr>
<td>( L_{mq} )</td>
<td>1.60 mH</td>
<td>16-21, q-axis magnetizing inductance</td>
<td>( r_{fd} )</td>
<td>0.015 ( \Omega )</td>
<td>d-axis field winding resistance</td>
</tr>
<tr>
<td>( L_{ikq} )</td>
<td>0.06 mH</td>
<td>q-axis damper winding leakage inductance</td>
<td>( r_{tr} )</td>
<td>0.05 ( \Omega )</td>
<td>13, 14, interphase transformer resistance</td>
</tr>
<tr>
<td>( r_{kq} )</td>
<td>0.110 ( \Omega )</td>
<td>q-axis damper winding resistance</td>
<td>( L_l )</td>
<td>0.25 mH</td>
<td>13, 14, interphase transformer leakage inductance</td>
</tr>
<tr>
<td>( L_{md} )</td>
<td>1.82 mH</td>
<td>16-21, d-axis magnetizing inductance</td>
<td>( L_m )</td>
<td>1.0 mH</td>
<td>13, 14, interphase transformer mutual inductance</td>
</tr>
<tr>
<td>( L_{ifd} )</td>
<td>0.255 mH</td>
<td>d-axis field winding leakage inductance</td>
<td>( r_L )</td>
<td>10.0 ( \Omega )</td>
<td>15, load resistance</td>
</tr>
</tbody>
</table>

ment the stator network shown in Fig. 3; whereas the standard Simulink library blocks were used to implement the rotor state model. The overall Simulink model is depicted in Fig. 4, and a snapshot of the Matlab file defining circuit is shown in Fig. 5.

In the following computer study, a constant excitation and generator speed are assumed. The system starts up with disconnected generator neutrals (branch 22) and initial conditions selected close to steady-state operation with a given load. At \( t = 0.01 \text{s} \), the switch branch 22 is closed after which the model is continued to run until \( t = 0.02 \text{s} \). The computer-generated transient responses of the phase \( a1 \) generator voltage \( v_{as1} \), generator current \( i_{as1} \), the output dc voltage \( v_{dc} \) and the rectifier currents \( i_{rec1} \) and \( i_{rec2} \) are shown in Fig. 6.

When the generator neutrals are disconnected, the upper and lower rectifiers essentially operate as two independent 3-phase rectifiers shifted by 60 degrees and connected in parallel; thus producing dc voltages with the same in-phase ripple. Since the dc currents \( i_{rec1} \) and \( i_{rec2} \) have in-phase ripple as shown in Fig. 6, the interphase transformer does not produce the desired harmonic cancellation. Also, since the rectifiers operate in parallel, the currents are essentially equally divided between the two sets of stator windings. Therefore, each diode conducts approximately one-half of the load current during its conduction period, which results in symmetric phase current shown in Fig. 6.

The system performance changes significantly when the generator neutrals are connected to one another. In particular, since the negative rails of both rectifiers are common, the load current \( i_{load} \) can return through any of the six negative rail diodes. However, the positive rails are separate, resulting in the splitting of the load current between the two bridges. Also, because of the 60-degree shift of the stator windings, the current ripple produced by each bridge is out-of-phase. Therefore, after the neutrals have been connected, the dc currents \( i_{rec1} \) and \( i_{rec2} \) have out-of-phase ripple as shown in Fig. 6. Additionally, each diode connected to the negative rail now conducts the peak load current for a part of the period;
interphase transformer. As a result, the output dc voltage ripple is reduced; however, the ac currents are now asymmetrical. The operational mode for this case is readily established using the ASMG and summarized in Table 2. Based on the mode given in Table 2, it can be noted that each diode connected to the negative rail participates only in 5 out of 24 states of the mode; whereas each diode con-

Figure 4: Simulink model of the example system.

Figure 5: Snapshot of initial file defining topology of example system.
Figure 6: System response to connecting generator neutrals.

Table 2: Operational mode for the system with generator neutrals connected.

<table>
<thead>
<tr>
<th>Switching State</th>
<th>Active Valves</th>
<th>$\alpha$</th>
<th>Switching State</th>
<th>Active Valves</th>
<th>$\alpha$</th>
<th>Switching State</th>
<th>Active Valves</th>
<th>$\alpha$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$y^{0}$</td>
<td>1,6,11,12</td>
<td>3</td>
<td>$y^{18}$</td>
<td>2,3,7,8</td>
<td>3</td>
<td>$y^{16}$</td>
<td>4,5,9,10</td>
<td>3</td>
</tr>
<tr>
<td>$y^{1}$</td>
<td>1,11,12</td>
<td>2</td>
<td>$y^{9}$</td>
<td>3,7,8</td>
<td>2</td>
<td>$y^{17}$</td>
<td>5,9,10</td>
<td>2</td>
</tr>
<tr>
<td>$y^{2}$</td>
<td>1,7,11,12</td>
<td>3</td>
<td>$y^{10}$</td>
<td>3,7,8,9</td>
<td>3</td>
<td>$y^{18}$</td>
<td>5,9,10,11</td>
<td>3</td>
</tr>
<tr>
<td>$y^{3}$</td>
<td>1,7,12</td>
<td>2</td>
<td>$y^{11}$</td>
<td>3,8,9</td>
<td>2</td>
<td>$y^{19}$</td>
<td>5,10,11</td>
<td>2</td>
</tr>
<tr>
<td>$y^{4}$</td>
<td>1,2,7,12</td>
<td>3</td>
<td>$y^{12}$</td>
<td>3,4,8,9</td>
<td>3</td>
<td>$y^{20}$</td>
<td>5,6,10,11</td>
<td>3</td>
</tr>
<tr>
<td>$y^{5}$</td>
<td>1,2,7</td>
<td>2</td>
<td>$y^{13}$</td>
<td>3,4,9</td>
<td>2</td>
<td>$y^{21}$</td>
<td>5,6,11</td>
<td>2</td>
</tr>
<tr>
<td>$y^{6}$</td>
<td>1,2,3,7</td>
<td>3</td>
<td>$y^{14}$</td>
<td>3,4,5,9</td>
<td>3</td>
<td>$y^{22}$</td>
<td>1,5,6,11</td>
<td>3</td>
</tr>
<tr>
<td>$y^{7}$</td>
<td>2,3,7</td>
<td>2</td>
<td>$y^{15}$</td>
<td>4,5,9</td>
<td>2</td>
<td>$y^{23}$</td>
<td>1,6,11</td>
<td>2</td>
</tr>
</tbody>
</table>
5 Conclusions

The automated state variable approach presented here is a convenient tool for modeling complex power-electronic-based systems that include special-purpose multi-phase electrical machines, transformers, and other components that are not available in the libraries of commonly used circuit simulators. An ASMG toolbox that is compatible with MATLAB/SIMULINK has been developed and used to implement a detailed simulation of a 6-phase generator/rectifier system. The possibility of connecting/disconnecting the generator neutrals results in different operational modes that are difficult to establish analytically. The automated identification of the operational modes has been shown very useful in analysis of system performance.

References
