1992 ACC/ TA10

Object Oriented Approach to the Simulation of Shipboard Electric Power Systems

Norbert H. Doerry J.V. Amy Jr. Mary Tolikas J.L. Kirtley Jr. Marija D. Ilic

Naval Sea Systems Command (NAVSEA 05Z) Washington, D.C.

Department of Electrical Engineering and Computer Science
Department of Ocean Engineering
Massachusetts Institute of Technology
Cambridge, MA 02139

Abstract

An object oriented approach for time domain simulations of shipboard electric power systems is proposed. It offers a flexible, user friendly alternative to conventional simulators, allowing design and operation analysis under a variety of operating conditions and topologies. The most general model of an electric power system consists of a set of nonlinear differential equations subject to algebraic constraints, which describes the evolution of the system, following certain contingencies. The algorithm presented adopts an object oriented approach by decomposing the system into its underlying components/devices, developed and maintained independent of one another. The differential equations are solved on the device level using a waveform representation of variables, which exploit both partitioning and parallel processing features. The ideas presented are tested on different test cases and conclusions are drawn.

1. Introduction

Naval electric power systems pose a significant simulation challenge due to the tight coupling of machine dynamics with control dynamics, lack of rotational inertia, extensive use of power electronics, and lack of time scale separation of electrical and mechanical dynamics. A typical propulsion electrical distribution system consists of one to three gas turbine generators rated between 19 and 22MW, two to four Load-Commutated-Inverter fed 20,000HP synchronous motors, one to three propulsion derived ship service (PDSS) power converters, and potentially pulsed power loads such as electro-thermal guns and directed-energy weapons.

To develop numerical methods suitable for simulating naval electric power systems, the simulation development tool WAVESIM was created. Key features of WAVESIM include the terminal description method for describing device interfaces, treating variables as waveforms over a specific time interval, expressing waveforms as a data type, and partitioning the system through the creation of a System Structural Jacobian Matrix.

2. Description of WAVES IM

2.1 Device and Variable Definition

A system is composed of one or more devices which are connected by attaching device terminals to system nodes. Each device forms a separate module which is built and maintained separately from other device models. Each model is characterized by a set of constitutive equations, describing its dynamic behavior, and by a set of parameters, which may be changed at will at the beginning of each simulation. Each device is attached to the rest of the system according to its particular terminal configuration. Every model is stored in an expandable library of devices. The device configuration may be changed without affecting the rest of the code.

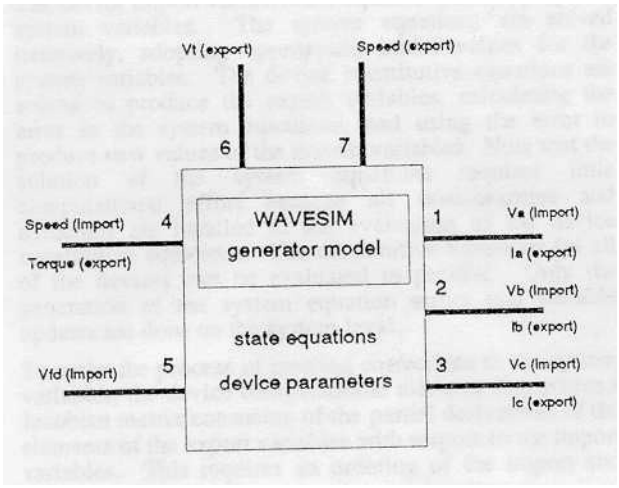

Figure 1 - WAVESIM Model of Synchronous Generator

Device interaction with the rest of the system is achieved by signals (interface variables) exchanged between device terminals and system nodes. Consider the WAVESIM model of a synchronous generator, illustrated in Figure 1. The device consists of seven terminals through which the machine interacts with the system nodes. The interface variables are the phase voltages and currents, the field voltage, V,, the terminal voltage, V,, and the mechanical speed and torque. These may be further classified as either "potential" or "flow" variables. Potential variables

adopt the value associated with the respective node and are referenced to zero. In the generator model, speed and voltages form potential variables, but this category may also include pressures, deflections etc. Currents, torques, forces, and power flows represent flow variables. They obey the law that the sum of all flow variables attached to a node is zero. This is a generalized form of Kirchoff's Current Law (KCL).

Interface variables may be either inputs to or outputs of the device. In the example of Fig. 1, speed and voltage are inputs to the generator, referred to as imports. Torque and current are then calculated according to the device constitutive equations. They are explicitly defined in terms of the device import variables and are referred to as exports.

Another type of variable is the state variable. State variables are variables that are continuous over successive time increments. Their values are updated at the end of each time interval and remain unchanged until the next update. For generators, the 7th-order model of reference [2] has been adopted to represent the device dynamics.

Two types of terminals exist: "normal" and "information". Normal terminals, on one hand, have associated with them a flow and a potential variable. Their salient characteristic is that there is transfer of energy or mass. In the case of Fig. 1, terminals 1, 2, 3 and 4 are associated with electrical and mechanical power transfer. Information terminals on the other hand, are associated with only one potential variable, which conveys information between devices with no energy or mass transfer. For the generator model, terminals 6 and 7 will only sustain information on torque and terminal voltage. They may be connected to a prime mover model and a voltage regulator model respectively, if a more complicated system incorporating control effects needs to be considered.

To ensure a consistent set of system equations, the number of export variables associated with normal terminals must equal the number of normal terminals (likewise with the import variables). There is no restriction on the number of import or export variables associated with information terminals.

Nodes are the means by which systems are created from one or more devices. A normal node has at least one normal terminal attached to it. Information terminals can be associated with a normal node as long as none of the information terminal potentials are defined as export variables. A normal node can have two types of system variables associated with it, one node potential and zero or more attached import flow variables. Two types of system equations can also be associated with a normal node, one KCL and zero or more potential difference equations associated with each attached export variable. An information node on the other hand, can have only information terminals attached to it, and only one of them may be an export variable. Since there are no flow variables associated with an information node, no KCL equation can be written.

In creating the constitutive equations for a device, there is no requirement for KCL to be satisfied within the device. This flexibility allows flow losses within the device. In power systems analysis, real and reactive power constitute the flow variables. The power entering a transmission line model, for example, does not equal the power leaving the device. Hence, KCL within such a model does not hold.

A possible reference frame problem can exist if one or more devices have groups of normal terminals which do satisfy KCL. This situation is typical in circuit simulators and is solved by defining a reference node where the voltage is specified and KCL is not written. Within the terminal description method, a reference is established by attaching a single normal terminal reference device to a node.

The reference device's potential is defined as an export variable and is usually equated with 0, i.e. datum. The flow variable is defined as an import variable. Reference frame problems can be identified by using KCL Group numbers. The normal terminals of a device are broken into sets called KCL Groups which satisfy KCL by definition. If a terminal can not be assigned to a KCL Group, it is given KCL Group Number 0. All other terminals are given a KCL Group Number according to the KCL Group they belong to. In other words, a common KCL Group number indicates which terminals are connected. By examining the system and identifying which nodes are connected through common KCL Group numbers, a subsystem of nodes within the system can be identified. For a reference frame problem to not occur, at least one of the nodes of a subsystem must have a terminal with a KCL Group Number of 0. Otherwise the set of KCL equations for tare nodes and the KCL Groups will be linearly dependent.

The device import variables are equated to the appropriate system variables. The system equations are solved iteratively, adopting appropriate initial values for the system variables. The device constitutive equations are solved to produce the export variables, calculating the error in the system equations, and using the error to produce new values of the system variables. Note that the solution of the system equations requires little computational effort because all nonlinearities and dynamics are handled in the evaluation of the device constitutive equations. The constitutive equations for all of the devices can be evaluated in parallel. Only the generation of the system equation errors and variable updates are done on the system level.

To assist the process of creating corrections to the system variables, the device computational modules also return a Jacobian matrix consisting of the partial derivatives of the elements of the export variables with respect to the import variables. This requires an ordering of the import and export variables.

2.2 Waveforms

Another feature of WAVESIM is the use of waveforms to describe variables over a time interval. Waveforms are represented as a series of Legendre coefficients, Chebyshev coefficients, Fourier coefficients, polynomial coefficients, or a series of data points. As a result, each waveform consists of a vector of numbers, a type indicator, and the time interval. The type indicator identifies the type of waveform representation, i.e., it specifies how the vector of numbers are interpreted for the analysis. A continuous function representing the variable over the particular time interval is generated accordingly.

Since each waveform type has its own particular advantages and disadvantages, device constitutive equations have the option of arbitrarily converting waveforms between the different types. Appropriate conversion routines have been developed and incorporated within the program structure. The Legendre Series and Chebyshev Series representations are particularly good for differential equations since integration and differentiation become simple linear matrix operations which are exact. Furthermore, the use of orthogonal series provides a means for controlling truncation error. Integration for example, increases the order of a polynomial. Since the value of a Legendre Series coefficient is independent of the number of terms in the series, truncating higher order terms makes more sense than simply eliminating the highest order polynomial coefficient. Note that since the integration process is exact, numerical stability of the integration method is not a concern. The problem is reduced completely to truncation error control.

The import and export variables for a device can be of any arbitrary waveform type. Waveform operators exist to perform mathematical functions on the waveforms and to convert waveforms from one type to another. Devices may perform the calculations independent of the waveform type, or can convert the waveform to a particular waveform type, perform the calculations necessary to generate export variables, then convert export variable waveforms to the proper type.

2.3 Assembling the System Equations

Once models for all devices in the system have been generated, an input file is created by the user. The input file defines the device parameter values and initial states for the particular simulation and the system topology. The terminal description provided creates a system of N equations in N unknown system variables. The system unknowns are of two types: node potentials and all import flow variables. The system equations are also of two types: Normal node KCL equations and Potential Difference equations (difference between node potential and export potential variable) for every export potential variable. The number of system equations equals the number of normal node KCL equations (nn) plus the number of export potential variables associated with normal terminals (nf) plus the number of export potential variables associated with information terminals (ni). Similarly, the number of variables equals the number of normal node potentials (nn) plus the number of information node potentials (ni) plus the number of import flow variables (nf). Hence the number of system equations and variables are equal to N.

Mathematically the problem is formulated as follows:

Denote F as a function space.

$$F = R^N \times R^N$$

$$0 = \mathbf{f}(\mathbf{x}(y), \mathbf{h}(\mathbf{x}(y)))$$

$$0 = \mathbf{g}(\mathbf{h}(\mathbf{x}(y))) \quad (1)$$

$y \in F$ are the system variables,

$\mathbf{x} \in F$ are the device import variables such that $\mathbf{x} = \mathbf{x}(y)$,

$\mathbf{h}: F \to F$ are the device export variables such that $h = h(x)$.

$\mathbf{h}$ are the device constitutive equations, where all the device dynamics are located.

$\mathbf{f}: F \to F$ are the KCL equations at the nodes.

$\mathbf{g}: F \to F$ are the KVL equations at the nodes.

The system equations form a set of nonlinear algebraic equations which are to be solved simultaneously.

$$P(Y) = 0 \quad (2)$$

The terminal description method creates a rather large set of system equations and variables. These equations however, may not need to be solved all at once. Instead, WAVESIM identifies a sequence of blocks of equations and variables which can be solved independently of the blocks which follow it. The blocks are identified by constructing and analyzing a System Structural Jacobian Matrix. Each element of the System Structural Jacobian Matrix consists of a character indicating the nature of the relationship between a system equation and a system variable. The character can be one of the following:

Table 1 - Structural Jacobian Element Descriptions

0	Zero	Structural zero (always zero)
I	Identity	Always the identity marix
D	Diagonal	Always a linear diagonal matrix
L	Linear	Always a linear time invariant matrix
N	Nonlinear	Nonlinear matrix
U	Unknown	Treated as a nonlinear element

The System Structural Jacobian matrix is constructed by combining appropriate elements of Device Structural Jacobian matrices which indicate the nature of the dependence of export variables with respect to import variables. The Structural Jacobian elements follow a very simple algebra.

$$\begin{aligned} x + y &= x \quad \text{for} \quad x > y \\ x + x &= x \quad \text{for} \quad x \neq I \\ I + I &= D \\ x - y &= x \quad \text{for} \quad x > y \\ x - x &= x \quad \text{for} \quad x \neq I \\ I - I &= 0 \end{aligned} \quad (3)$$

Here, the > operator assumes an ordering of {0 I D L N U}.

Once the System Structural Jacobian Matrix is constructed, "presolve" and "postsolve" blocks are alternately identified. A presolve block of size n is found by identifying n rows which only have nonzero elements in the same n columns. It solves variables which are independent from the remaining system variables. A postsolve block of size n is found by identifying n columns which only have nonzero elements in the same n rows. This block solves variables which are wholly

dependent upon the previously calculated system variables. Once a block has been identified, its corresponding rows and columns in the System Structural Jacobian Matrix are ignored in further block identifications. Maintaining the order of block identification is important since presolve blocks must be solved in the order of their detection followed by postsolve blocks solved in reverse order.

The algorithm for identifying blocks in WAVESIM currently starts by identifying as many presolve blocks of size 1 as possible. Once these have been found, as many postsolve blocks of size 1 as possible are identified. Presolve blocks of size 2 are then identified. If a presolve block of size 2 is found, then presolve blocks of size 1 are searched for before looking for another presolve block of size 2. This process is repeated until no more presolve blocks of sizes 1 or 2 can be found. Similarly presolve blocks of size 1 and 2 are then alternately searched for. The algorithm continues for block sizes of 3 or more until twice the block size is greater than the number of the remaining rows in the system.

2.4 Solving the System

Once the sequence of blocks has been identified, the Newton-Raphson method is employed to solve one block at a time for its corresponding variables. Initial guesses for the waveforms may be provided by the user, but must fall within the region of attraction of the solution. For guesses too far from the solution, the Newton-Raphson method fails and a continuation scheme deforms a simple problem to the one to be solved is employed as an alternative.

2.5 Algorithm Structure and Object Oriented Concepts

The algorithm outlined above exploits object oriented characteristics, which further promote efficiency, portability and ease of debugging. The first concept of the object oriented paradigm is that of modularity. The code is a structured collection of objects. Every object (module) consists of a set of operations that define its interaction with the rest of the modules and a set of hidden methods that implement its specific function and change its associated variables accordingly.

In the algorithm presented, different system devices form distinct objects. They interact with the rest of the system according to a set of rules (KCL and KVL equations) and encapsulate their own dynamic behavior.

Objects can communicate via message (signal) passing. Since each object is an independent entity, each object can interact with the rest of the system only by these messages. Variable definition in WAVESIM further promotes modularity, since signals are described by waveform objects.

Waveform and device objects implement another aspect of object oriented programming, class inheritance. New waveform types may be defined by the user at any time. These types inherit the same properties of the waveform structure (type indicator, vector of coefficients, time interval) but allow for different waveform interpretations and device functions.

The routines supporting any new waveform type for example, need only establish the way mathematical functions are performed on the particular waveform and the way conversion takes place between the new type and already existing types. Hence, the rest of the code may be left untouched, enabling quick compilation and easy debugging. Note that the modularity of the program is still sustained at this level. New waveform types may be defined by the user at any time without affecting already existing waveform types.

Polymorphism is also exploited through the use of generalized functions that may be applied uniformly to a number of different objects. Waveform addition for example, is not limited to a certain type of waveform. Different waveform types may be added by the use of the same function just like in any commercial programming language. The '+' sign may be used to add two integers or two reals or even a real and an integer. In WAVESIM, the waveform addition function may be used to add two Chebyshev polynomials, two data waveforms or even a Fourier with a Legendre series.

3. Simulations

Three simulations were performed using the synchronous generator model illustrated in Fig. 1. The generator parameters employed for these simulations appear in Table 2.

Table 2 – Synchronous Generator Parameters

Parameter	Value	Description
x_d	2.0 pu	"d" axis synchronous reactance
x_q	1.8 pu	"q" axis synchronous reactance
x_d'	0.4 pu	"d" axis transient reactance
x_d''	0.2 pu	"d" axis subtransient reactance
x_q''	0.2 pu	"q" axis subtransient reactance
T_{do}'	5.0 sec	"d" axis open circuit transient time constant
T_{do}''	0.2 sec	"d" axis open circuit subtransient time constant
$T_{qo},''$	0.2 sec	"q" axis open circuit subtransient time constant
T_a	0.03 sec	armature time constant
H	3.0 sec	inertia constant

Figure 2 shows the a-phase current transient behavior during a symmetrical short circuit. The generator is initially open-circuited. At time zero, the fault is applied at its terminals. Constant rotor speed is maintained throughout the simulation. Figure 3 shows the mechanical torque characteristic during the short circuit.

The second simulation is an open circuit test. The rotor is turning at rated speed with all electrical states and inputs equal to zero. At time zero, constant field excitation is applied. The build-up of a-phase's voltage at the open circuited armature terminal is shown in Figure 4.

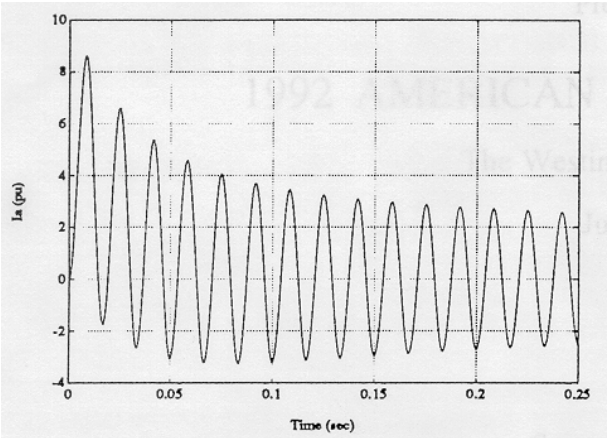

Figure 2 - Synchronous Generator Short Circuit a-Phase Current

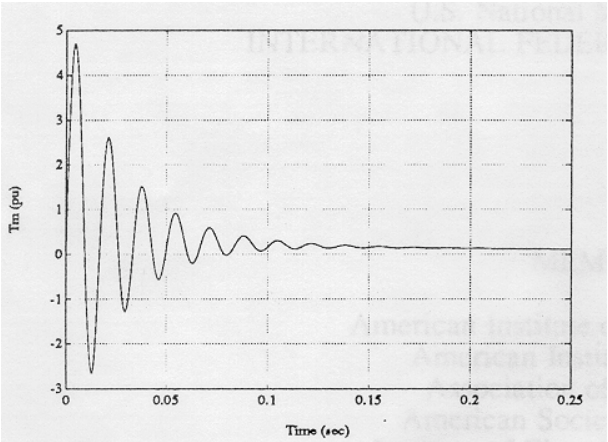

Figure 3 - Synchronous Generator Short Circuit Torque

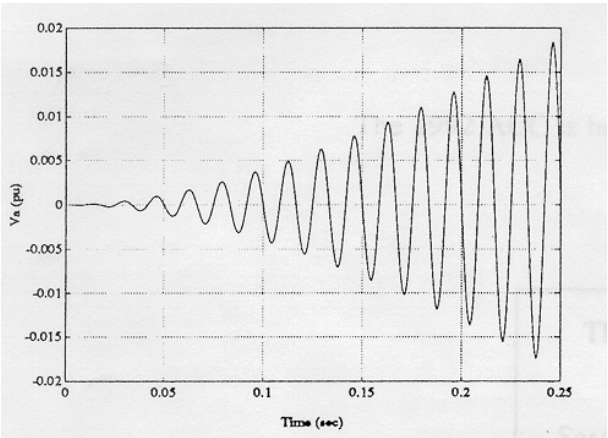

Figure 4 - Synchronous Generator Open Circuit a-Phase Voltage

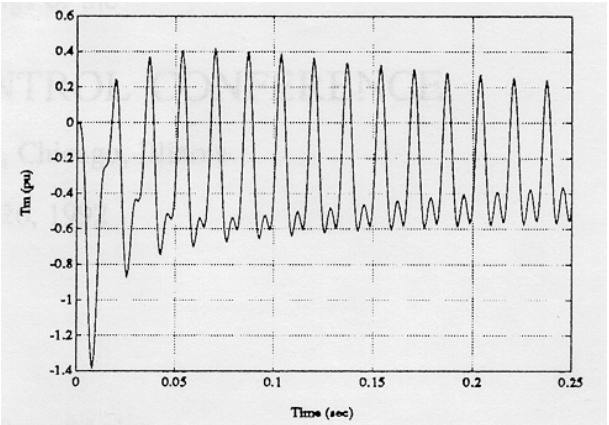

Figure 5 - Synchronous Generator Blocked Rotor Torque

A blocked rotor test is performed by holding the rotor fixed and applying sinusoidal voltages at the armature terminals. The field coil is not excited. Figure 5 shows the torque response of the blocked rotor.

4. Conclusions

A new object oriented algorithm is presented for the dynamic analysis of shipboard electric power systems. The system is broken into its component devices. Their dynamics are treated on the device level. Variables are described by waveforms which are abstract data types. Appropriate functions for numerical integration are developed which exploit numerical properties more stable than numerical integration. Choice of an integration step is not an issue, whereas truncation error control is an issue. The algorithm developed results initially in a larger number of equations than produced by methods arising from branch description of devices. However, the System Structural Jacobian notion is introduced, allowing for significant reduction in the order of the systems of equations to be solved simultaneously. WAVESIM is used successfully in the simulation of systems consisting of synchronous generators, switches, prime movers, and various types of controllers.

Acknowledgements

This work has been supported by the Defense Advanced Research Projects Agency, Naval Sea Systems Command (Code 05Z), David Taylor Research Center and the Office of Naval Research.

5. References

[1] Norbert H. Doerry, "Advanced Numerical Methods for Simulating Nonlinear Lumped Parameter Models," Ph.D. Thesis, Massachusetts Institute of Technology, May 1991.

[2] James L. Kirtley Jr., "Synchronous Machine Dynamic Models," LEES Technical Report TR-87-008, Laboratory for Electromagnetic & Electronic Systems, Massachusetts Institute of Technology, Cambridge, MA, June 5, 1987.