Elimination of numerical delays in the solution of control systems in EMTP

J. Mahseredjian[1] L. Dubé[2] M. Zou[3] S. Dennetière[4] G. Joos[3]

Abstract-- The classical approach for solving control systems in EMTP is to insert one-time-step delays for breaking feedback loops when one or more nonlinear functions are encountered. Although this approach may remain acceptable for several well-behaved cases, it provides a non-simultaneous solution for a nonlinear system. Fast varying components may create instabilities in such a solution and/or simply end up into a wrong operating region. In some cases, it imposes severe limitations on the integration time-step for minimizing delays. This paper presents a new approach based on a Jacobian matrix for eliminating numerical delays in the solution of control systems.

Keywords: EMTP, Control systems, nonlinear problems

I. Introduction

In EMTP type computer programs, control systems are represented using high-level blocks. These are also control blocks. An example taken from EMTP-RV [1][2] is shown in Fig. 1. Similar approaches are used in other applications.

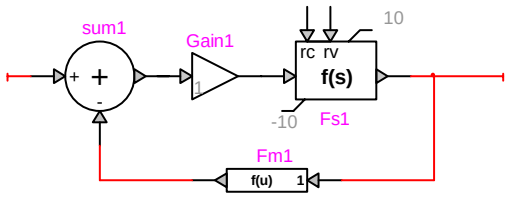


Fig. 1 Block-diagram example for control system representation

A typical primitive library of control blocks includes devices such as limiters, integrators, transfer functions and generic equation blocks. In addition to modeling power system control functions, these blocks can be used to solve algebraic-differential equations for user-defined models. Contrary to standard control system models, user-defined modeling is mathematically more demanding since it may include several nonlinear loops and complex interactions between several sets of equations.

Contrary to power system devices, control system devices have fixed orientation due to given inputs and outputs.

At each simulation time-point *t*, the generic system of equations for an arbitrary control system can be written as:

$$\mathbf{Ax} = \mathbf{b} \quad (1)$$

Bold characters are used to denote vectors and matrices. All controls signals are listed in the unknowns vector **x**, the matrix **A** can be seen as the matrix of constraint equations on control signals and **b** is the vector of known fixed quantities and history variables. History results from the discretization process required in some control blocks (devices). The trapezoidal integration method can be used for discretization.

The solution of (1) is straightforward if all devices in a control system or in a group are linear. This is however seldom the case. In the example of Fig. 1, the feedback loop device Fm1 may be a nonlinear function and Fs1 may also become nonlinear if anyone of its limits is exceeded or if its reset signal is applied.

A simple and widely used approach [3] is to insert one-time-step delays for breaking feedback loops when one or more nonlinear functions are encountered. Although this approach may remain acceptable for several well-behaved cases, it provides a non-simultaneous solution for a nonlinear system. Fast varying components may create instabilities in such a solution and/or simply end up into a wrong operating region. In some cases it imposes severe limitations on the integration time-step for minimizing delays. User-defined modeling using control blocks as a general purpose solver environment is particularly more sensitive to delays since in the majority of cases it requires a simultaneous solution for model equations.

This paper presents a new approach for eliminating delays in the solution of control systems. The complete system of control equations is formulated using a Jacobian matrix and solved through an iterative process. All types of nonlinear functions can be represented. Re-solve problems for reset functions and hard limiters and transposition into a linear solver are also analyzed.

The new approach provides two solution methods, one fully iterative and one non-iterative. Both methods are demonstrated using simple cases and the user-defined modeling of a complete asynchronous machine model.

II. Ordering of Equations

Since the initial introduction of TACS [3], the number of significant contributions in this field has been limited. In addition to simply adding delays to break nonlinear feedback conditions, it has been suggested to enforce ordering strategies [4]. This scheme can be explained with the simple example of Fig. 2 taken from a turbine-governor system.

(1) École Polytechnique de Montréal, Canada
jean.mahseredjian@polymtl.ca
(2) DEI Technology, Montréal, Canada
(3) McGill University, Montréal, Canada
(4) Électricité de France, Clamart, France

Presented at the International Conference on Power Systems Transients (IPST'05) in Montreal, Canada on June 19-23, 2005
Paper No. IPST05 - 144

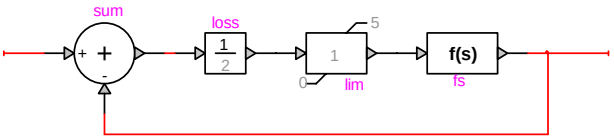


Fig. 2 A limiter loop example

If the transfer function fs is replaced only by its gain of 2 for simplifying the presentation, then the equivalent of equation (1) for this diagram is given by:

$$\begin{bmatrix} 1 & -0.5 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & -1 \\ 0 & 0 & 1 & -2 & 0 \\ 0 & 0 & 0 & 2 & -0.5 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_{loss} \\ x_{sum} \\ x_{fs} \\ x_{lim} \\ x_{step} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 10 \end{bmatrix} \quad (2)$$

It is assumed that the input to the sum is 10 at the particular solution time-point. The signals named in the **x** vector are the output signals of the corresponding blocks with x_{step} being the input signal to the sum block. Since x_{step} is known it is ordered first. The output signal of the limiter x_{lim} is forced to appear next so that it is known before solving for the remaining blocks. It is therefore related only to the forcing input. This technique works with only one limiter in the loop, if an extra limiter appears on the output of fs, its limiting action will not be detected by the block lim and the answer will be incorrect.

If the lim block is replaced by a nonlinear function, then the **b** part of equation (2) will contain a nonlinear function of the form $f(x_{loss})$. Since this function cannot be directly included in the **A** matrix, it must evaluated separately from the previous time-point solution for x_{loss} and creates the time-step delay.

As explained earlier, the time-step delay approach (delay-based solver) can be acceptable in well behaved cases. Although it will not give the exact solution, the error can be minimized by selecting smaller time-steps. The actual control system diagram may become elaborate and use several feedback loops and overlapping between feedback groups. In cases where the automatically inserted delays are insufficient, a common practice is to insert delays manually to force correct simulations. A typical consequence is a delayed solution accompanied in some cases by initial oscillations [4].

There are cases where the time-step delay approach will not work and a simultaneous solution is compulsory. In the example of Fig. 3, Fu is a nonlinear function given by G^2 (G is the input signal), the Gain value is set to 1 and the input to sum is assumed to be a constant value of 10. It is the block-diagram representation of:

$$f(x) = x^2 + x - 10 = 0 \quad (3)$$

Even if the nonlinear block is solved first and a delay is added to its output, this system will encounter numerical overflow and the value of G will go to infinity.

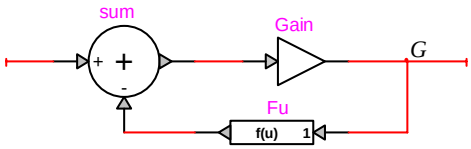


Fig. 3 Nonlinear test case

III. New Methods

A. Iterative method

Based on the fact that equations such as (3) have a solution when a Newton method is used, the contribution of this paper is to implement such a method for solving generic control systems in EMTP. It is briefly recalled that in the most general context the solution of a nonlinear function $\mathbf{f}(\mathbf{x}) = 0$ can be found using an iterative procedure with a Jacobian matrix:

$$\mathbf{f}(\mathbf{x}^{(k)}) + \mathbf{J}^{(k)}\Delta\mathbf{x}^{(k)} = 0 \quad (4)$$

where **J** is the Jacobian matrix and k is the iteration count. In the case of a control system the solved function is given by equation (1):

$$\mathbf{f}(\mathbf{x}) = \mathbf{A}\mathbf{x} - \mathbf{b} = \mathbf{0} \quad (5)$$

It can be demonstrated that the calculation of the differential of (5) is achieved by finding the linearization of each nonlinear function for the last iterative solution vector $\mathbf{x}^{(k)}$. Since $\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)} = \Delta\mathbf{x}^{(k)}$, equations (4) and (5) can be combined to give:

$$\mathbf{J}^{(k)}\mathbf{x}^{(k+1)} = \mathbf{J}^{(k)}\mathbf{x}^{(k)} - \mathbf{A}\mathbf{x}^{(k)} + \mathbf{b}^{(k)} \quad (6)$$

The **b** part includes the actual evaluation of nonlinear functions. By using the linearization principle, equation (6) is equivalent to:

$$\hat{\mathbf{A}}^{(k)}\mathbf{x}^{(k+1)} = \mathbf{B}^{(k)} \quad (7)$$

Matrix $\hat{\mathbf{A}}$ rows are different from **A** for nonlinear device equations and identical to **A** for linear devices. Due to the resulting cancellation of terms in the right hand side of (6), vector **B** holds the y-axis intercepts of line equations at linearization points. This is in addition to existing determined constants and history terms.

It is noticed that equation (7) is applicable also to multivariable nonlinearities. This is the case when a nonlinear device, such as f(u) has several inputs as shown in Fig. 4

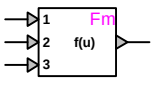


Fig. 4 Multivariable nonlinearity

When an f(u) device contains generic mathematical expressions, then its partial derivatives are obtained from input variable perturbation.

Equation (7) provides a simple transition from the original linear formulation (1). If 'hard' nonlinearities such as limiters and reset conditions are used, then the matrix $\hat{\mathbf{A}}$ is simply

modified to account for changes of condition. The hard limiter changes from a simple gain to a constant when a limit is reached. In view of the fact that an iterative process is used and all nonlinear functions are given an iteratively updated linearized equation, all delays are eliminated.

B. Non-iterative variant

A variant to the above proposed method is to use linearization, but cancel the iterative loop. Device linearization is updated only at each new solution time-point. This new non-iterative approach is particularly more justifiable when the integration time-step is sufficiently small. An extra step is needed to account for hard nonlinearities. When a hard nonlinearity is encountered the system is simply re-solved until no more events are detected. Correct initialization is achieved by automatically turning on the full iterative method at the first time-point t=0 solution. The main advantage over the previous method is computational speed when comparable precision can be achieved for the same integration time-step.

C. Ordering

With the proposed approach, forced ordering is no more an issue. In a generic control system, however, various topologies can still benefit from ordering to increase computational speed and improve convergence properties.

The following grouping and sequencing approach has been adopted in the practical implementation of this new method. A list of predecessors of each signal is first obtained by accumulating the explicit and implicit signal dependencies of a block-diagram. This information is used for separating the equations into an oriented set of groups, each group containing only coupled equations. The groups are not looped with each other and can be solved independently provided that all groups are solved in the correct sequence with respect to each other. The algorithm relies on graph theory theorems that guarantee correctness and optimality for grouping and sequencing.

Some groups may be only sequential (no looping), in which case the iterative solver is not needed since the solution is straightforward. If a group is completely linear, but contains loops, it can be solved directly without iterations.

IV. Examples

In the following test cases, Method 1 is the new non-iterative method with nonlinear device linearization and re-solve procedure for events. It is a variant of Method 2. Method 2 is the new fully iterative method. The proposed methods have been fully implemented and tested in EMTP (EMTP-RV version).

A. Test case 1: nonlinear function

The first test case is shown in Fig. 3. This test is designed to stress the numerical qualities of the proposed methods. When varying the input signal within the existing solution range, both methods 1 and 2 give the correct answer even when larger time-steps are used. A time-step of 10µs is used in Fig. 5. A step function is applied to the input signal at 2ms to test the behavior of Method 1 for non-smooth signals. When the step value is sufficiently small, Method 1 is still able to follow, but shows a slight overshoot. For a larger step value, Method 1 goes into numerical overflow.

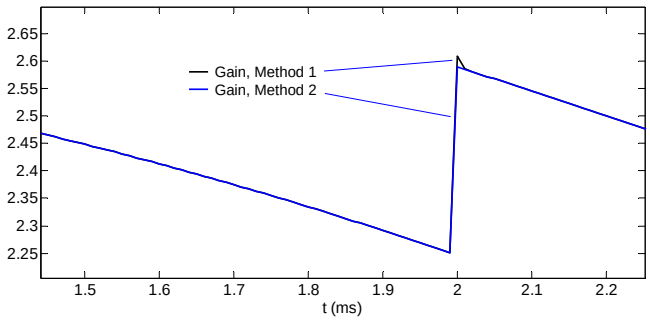


Fig. 5 Solution for the output of Gain block shown in Fig. 3

B. Test case 2: PLL problem

The PLL circuit shown in Fig. 6 is taken from [5]. It has nonlinear devices and feedback loops. The input u is stepped up at 125ms. If this system is simulated using a delay-based solver, the output will not be able to follow and results into a numerical noise as shown in Fig. 7 (y is the PLL output and F is the input function). As expected, when the time-step is reduced from 10µs to 1µs, the solver is able to recover from the step change.

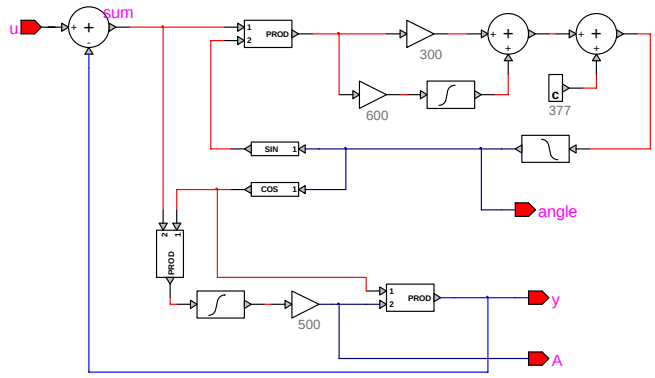


Fig. 6 PLL diagram

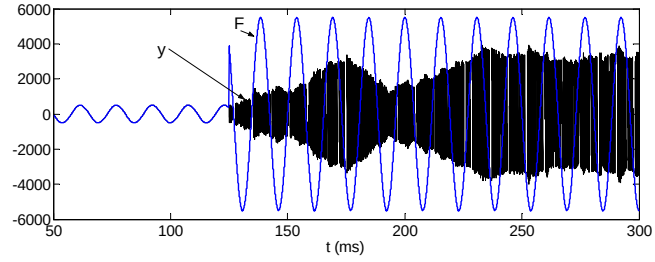


Fig. 7 PLL simulation results with a delay-based solver

Contrary to the above, the fully iterative Method 2 is perfectly capable of following the input waveform F with a 10µs time-step. Furthermore, Method 2 is capable of

maintaining stability and precision with time-steps of 10µs and even 20µs . Results with 20µs are shown in Fig. 8.

Method 1 behaves correctly up to 10µs . Fig. 9 compares the PLL amplitude from Method 1 against Method 2.

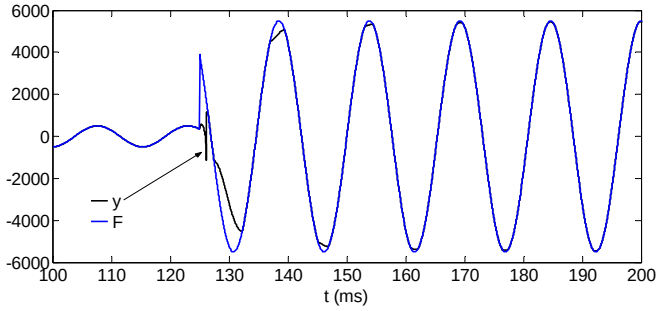


Fig. 8 PLL simulation with Method 2

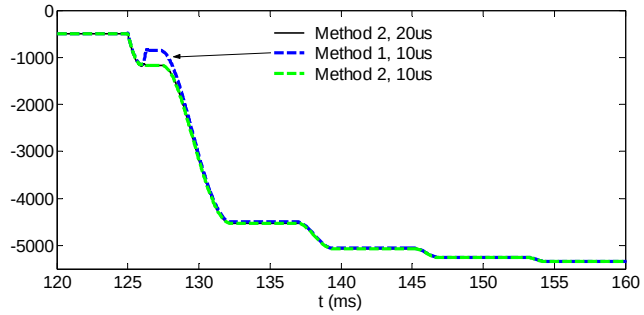


Fig. 9 PLL output amplitudes with Methods 1 and 2

C. *Asynchronous machine model*

The objective of this test case is to create a user-defined asynchronous machine model using control diagram blocks. There are several advantages from the user point of view for creating such models, since it provides an open architecture design and allows implementing various specific functionalities. Validation is provided by comparison with an existing and validated EMTP (EMTP-RV version [1][2]) hard-coded asynchronous machine model. The hard-coded model benefits from a tailored programming from reduced symbolic formulations. It also possesses internal iterative loops for simultaneous solutions in all machine variables.

The design with the hard-coded model is shown in Fig. 10.

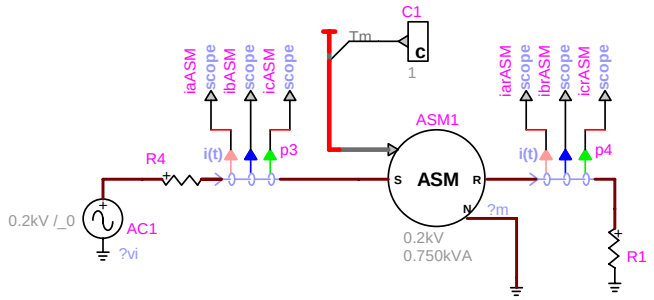
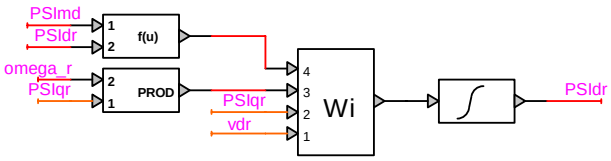


Fig. 10 Asynchronous machine model, hard-coded version

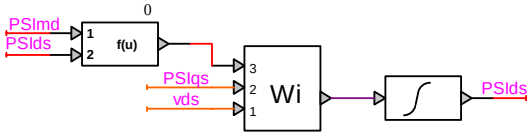
The user-defined model simply replaces the wound-rotor ASM device shown above, by a subcircuit containing machine equations assembled using control devices (function blocks).

The entire design is too large to be shown in this paper. Only some parts are presented below. It is a classical model with the equivalent circuit representation of the induction machine in the synchronous reference frame. The d-axis equations taken directly from the graphical user interface are shown in Fig. 11. The subscript *s* is for stator quantities, it also means quantities seen on the stator side; the subscript *r* is for rotor quantities. Some signals are connected by name for better legibility. It is noticed that generic f(u) function blocks are used for a direct mapping of model equations. A similar diagram is assembled for the q-axis.

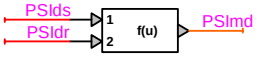
$$\Psi_{dr} = \int_0^t \left[\omega_b v_{dr} + (\omega - \omega_r)\Psi_{qr} + \frac{\omega_b R_r}{x_{lrs}}(\Psi_{md} - \Psi_{dr}) \right] dt$$



$$\Psi_{ds} = \int_0^t \left[\omega_b v_{ds} + \omega\Psi_{qs} + \omega_b \frac{R_s}{x_{lss}}(\Psi_{md} - \Psi_{ds}) \right] dt$$



$$\Psi_{md} = x_M \left(\frac{\Psi_{ds}}{x_{lss}} + \frac{\Psi_{dr}}{x_{lrs}} \right)$$



$$i_{ds} = \frac{\Psi_{ds} - \Psi_{md}}{x_{lss}} \qquad i_{dr} = \frac{\Psi_{dr} - \Psi_{md}}{x_{lrs}}$$

Fig. 11 Block diagram machine model, d-axis equations

The nonlinear function shown in Fig. 12 is used for including the saturation of leakage reactances (stator and rotor). It is given by [5]:

$$x_{lrs} = \frac{x'_{lrs}}{2}(1 + F_{sat}) \tag{8}$$

with the factor F_{sat} given by:

$$F_{sat} = \frac{2}{\pi}\left[\sin^{-1}\left(\frac{I_{sat}}{I}\right) + \left(\frac{I_{sat}}{I}\right)\sqrt{1 - \left(\frac{I_{sat}}{I}\right)^2} \right] \tag{9}$$

I_{sat} (6A in this example) is a threshold value for applying this factor and $I = \max(|i_{ar}|, |i_{br}|, |i_{cr}|)$. Similar equations are used for the stator.

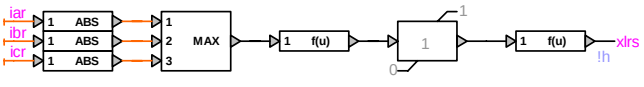


Fig. 12 Modeling of leakage reactance saturation (rotor side)

Since the underlying solver eliminates numerical delays, the setup of the entire design is straightforward and simplifies the mapping of model equations into actual control blocks. This is a major advantage. A delay-based algorithm [3] cannot be used to simulate this model directly.

Simulation results with a $5\mu s$ time-step are almost identical when Methods 1 and 2 are compared with each other and verified with the hard-coded model (reference) of Fig. 10. Even when the time-step is increased to $50\mu s$, both Methods 1 and 2 continue to maintain precision. The mean number of iterations in Method 2 is below 2 after an initialization count of 10. Since Method 1 does not use iterations after the initial startup, it remains more efficient. Experiments with a large set of cases indicate that Method 1 can be given the default status and remains acceptable in the majority of simulations.

Simulation waveforms of the machine are shown in Fig. 13 for a time-step of $50\mu s$. Methods 1 and 2 are almost identical to the reference (hard-coded) model. This observation is also confirmed for the electromagnetic torque waveforms shown in Fig. 14.

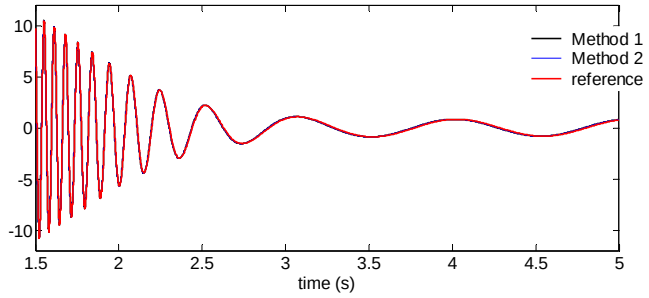


Fig. 13 Rotor current (A)

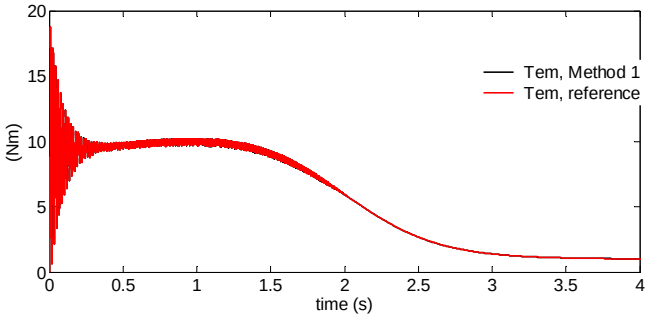


Fig. 14 Electromagnetic torque

If the control system equations are solved with a one-time-step delay with network equations, then some phase error will result in the user-defined equations. This phase error has been completely eliminated in the above presentation by providing the control equations with network Thevenin equivalents at interfacing points. These equivalents are automatically calculated and updated at each solution time-point.

V. Conclusions

This paper contributed a new approach for the computation of control system equations in the simulation of electromagnetic transients. The new approach allows programming two new methods. A fully iterative method and its non-iterative variant have been demonstrated of being capable to eliminate numerical delays in generic bloc-diagrams with multiple feedback loops and nonlinearities. Although the fully iterative method remains superior in all cases, the non-iterative variant is faster and can be confidently used in a large variety cases.

VI. References

[1] www.emtp.com

[2] J. Mahseredjian, L. Dubé, L. Gérin-Lajoie: "New Advances in the Simulation of Transients with EMTP: Computation and visualization Techniques". Proceedings of 7th International conference on Modeling and Simulation of Electric Machines, Converters and Systems. August 18-21, 2002, Montreal

[3] L. Dubé and H. W. Dommel: "Simulation of Control Systems in an Electromagnetic Transients Program with TACS", IEEE Power Industry Computer Applications Conference, 1977, pp 266-271

[4] R. H. Lasseter, J. Zhou: "TACS enhancements for the Electromagnetic Transients Program", IEEE Transactions on Power Systems, Vol. 9, Issue 2, May 1994, pp. 736-742

[5] M. Karimi-Ghartemani, M.R. Iravani: "A new phase-locked loop (PLL) system" *Circuits and Systems, 2001*. MWSCAS 2001. Proceedings of the 44th IEEE 2001 Midwest Symposium, Volume 1, 14-17, Aug. 2001 pp 421 -424

[6] G. J. Rogers, D. Shirmohammadi, Induction machine modeling for electromagnetic transient program, IEEE Trans. On energy Conversion, Vol. EC-2, No. 4, December 1987, pp. 622-628

VII. Biographies

Jean Mahseredjian (M'87) graduated from École Polytechnique de Montréal with M.A.Sc. (1985) and Ph.D. (1991). From 1987 to 2004 he worked at IREQ (Hydro-Québec) on research and development activities related to the simulation and analysis of electromagnetic transients. In December 2004 he joined the faculty of electrical engineering at École Polytechnique de Montréal.

Laurent Dubé (M'77) obtained the B.A. (1967) and B.Eng. (1972) from Sherbrooke University and M.Eng. (1973) from École Polytechnique de Montréal. He developed the TACS program of EMTP and the MODELS simulation language and solver. He recently worked with IREQ (Hydro-Québec) on the development of user-defined modeling for EMTP-RV. His consulting activities are in the field of user-defined modeling methods and languages.

Ming Zou graduated form Concordia University (Montréal, Canada) with M.A.Sc. Since 2002 he is a Ph.D. student at McGill University, Montréal. His research topic is on advanced methods for the simulation of control systems in EMTP. In May 2004, he joined BCTC (British Columbia Transmission Corporation, Vancouver, Canada) as a regional system planner

Sébastien Dennetière (M'04) graduated from École Supérieure d'Electricité (Supélec) in France in 2002. He received the M.A.Sc. degree from École Polytechnique de Montréal in 2003. From 2002 to 2004 he worked at IREQ (Hydro-Québec) on research and development activities related to the simulation and analysis of electromagnetic transients. In October 2004 he joined the research center of EDF where his interests are currently in the field of insulation coordination and power system simulations.

Geza Joos (M'82, SM'89) graduated from McGill University, Montreal, Canada, with an M.Eng. (1974) and Ph.D. (1987).

His employment experience includes ABB, the Ecole de technologie supérieure, Concordia University, and McGill University, Montreal, Canada. He is involved in fundamental and applied research related to the application of high-power electronics to power conversion, including distributed generation, and power systems, and in consulting activities.