Multi-format Graphical User Interface for EMTP-based Programs

Jesús Calviño-Fraga
Department of Electrical and Computer Engineering
The University of British Columbia
2356, Main Mall, Vancouver, B.C., V6T 1Z4, Canada
Tel: +1(604) 228-2710
jesusc@ece.ubc.ca

Benedito Donizeti Bonatto *
Department of Electrical and Computer Engineering
The University of British Columbia
2356, Main Mall, Vancouver, B.C., V6T 1Z4, Canada
benedito.bonatto@ieee.org

Abstract - **The use of EMTP-based programs in academia and industry has usually been regarded as difficult, in particular by students and new users, mainly due to non-friendly user interfaces. Furthermore, whether there is a graphical user interface (GUI) available, it is not usually compatible with some other EMTP-based programs or electronics more oriented programs such as SPICE. This paper presents the fundamental concepts for the development of a multi-format graphical user interface for EMTP-based programs, which can be designed to allow the automatic generation of input data files for any type of EMTP-based program, as well as for electronics simulation programs, such as SPICE. This property is especially useful if one needs to perform comparison simulations among different programs or tools. Besides that advantage, the new EEVS – Electrical Engineering Vector Schematics – have been designed to have flexible libraries of components and help wizards, which can make it very user-friendly. Examples are presented in this paper to illustrate the main properties of this multi-format computer tool for EMTP-based and electronics simulation programs.**

Keywords: EMTP, electrical engineering education, transients, electronics, graphical user interface, SPICE, vector schematics, computer simulation tools.

I. INTRODUCTION

Since the publication of [1], many developments have contributed to the widespread acceptance of EMTP-based programs around the world. For many EMTP users, however, specially for students and new users, the use of EMTP-based programs has been regarded as difficult, possibly not only due to the complexity of electromagnetic transient phenomena and eventually a lack of background knowledge in electrical engineering, but probably also due to non-friendly user interfaces.

Furthermore, whether there is a graphical user interface (GUI) available (as in [2]), it is not usually compatible with some other EMTP-based programs or electronics more oriented programs such as SPICE [3]. This paper presents the fundamental concepts for the development of a multi-format graphical user interface for EMTP-based programs, which has been designed to allow the automatic generation of input data files for any type of EMTP-based program, as well as for electronics simulation programs. This property is especially useful if one needs to perform comparison simulations among different programs or tools. Besides that, with a multi-format graphical user interface the user can easily change input parameters and receive a fast feedback from the simulations.

* Receiving a scholarship from CAPES/Brasília – Brazil.

II. IDEAL CHARACTERISTICS OF A MULTI-FORMAT GRAPHICAL USER INTERFACE

The evolution of computer software has resulted in various possible alternatives for the development of graphical user interface programs. However, the need for special tools, capable of capturing all the details of complex circuit modelling and at the same time able to provide a flexible and easy-to-use environment for a variety of input data formats for EMTP-based programs, has motivated the development of this work. Indeed, there might be a *synergetic economy* if a standard multi-format graphical user interface for EMTP-based programs become widely accepted and used.

Ideally, a multi-format graphical user interface must have, at least, the following main characteristics:

1. Intuitively easy to use (should have commands and behaviour similar to any Windows standard application so the user does not necessarily have to learn everything new);
2. Allows conversion of schematic circuit components data to any type of input data format required by any supported simulator program (i.e., creation of data files for different types of simulators);
3. Provides sharp and scalable drawings and uses standard symbols [4] for library components and devices;
4. Allows user to create custom libraries and data modules;
5. Allows the drawing of a single schematic circuit, or multiple schematics composing a project;
6. Provides project templates and help wizards for very complex circuit design;

To accomplish these requirements, the software EEVS – Electrical Engineering Vector Schematics has been developed. The main features of this program are presented in the following sections of this paper.

III. GENERATION OF INPUT DATA FOR CIRCUIT SIMULATOR

The ideal aim of EEVS is to allow the user to create a input data file for a particular circuit simulator as quicker and simpler as possible. The user starts the process by creating one or more schematic files, then EEVS generates a net-list file, from which the final input data file is created using a tailored net-list converter program. The resulting final input data file can be fed into the chosen circuit simulator, such as one of the many available versions of EMTP or SPICE. Fig. 1 presents the flowchart of this simulation process.

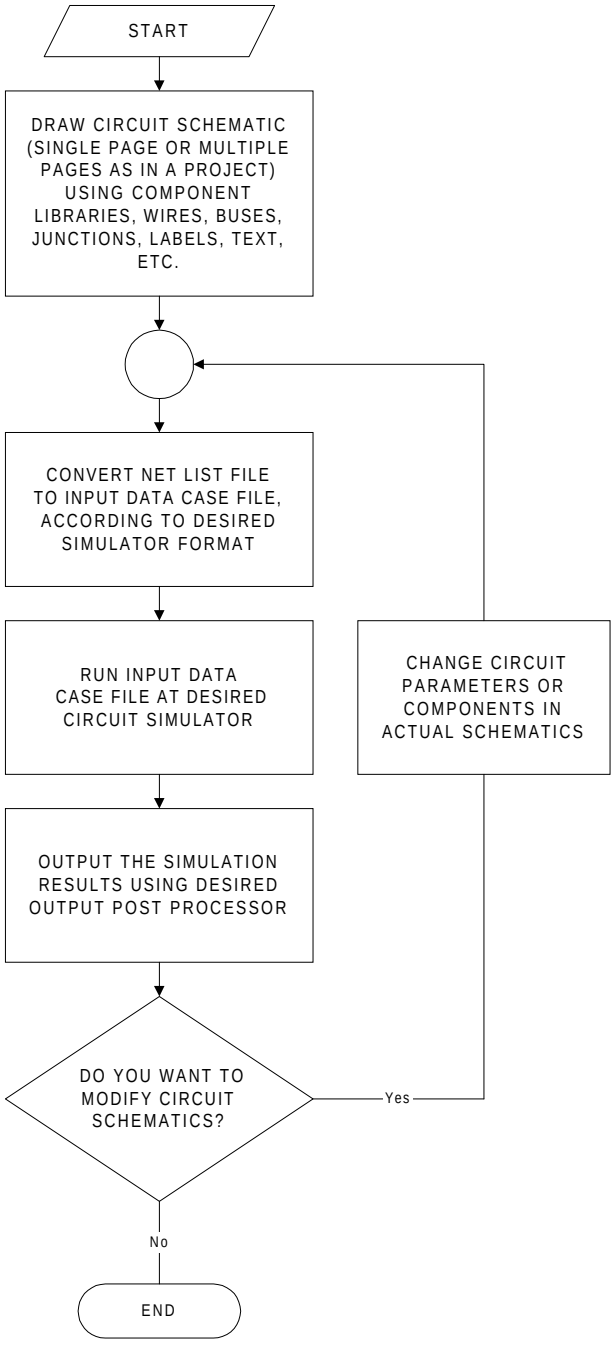

Fig. 1 – Flowchart of the transient simulation process with multi-format graphical user interface.

The process of creating a schematic with EEVS is designed to be simple. It starts by the user selecting the components from the available libraries, or creating its own library, if the desired components are not available. Assuming, for example, that the user wants to study transients in a typical RC circuit. The components are then arranged in such a way that they are easy to connect to each other and also the aesthetics of the schematic drawing is considered satisfactory by the user. Then, the wiring connections between the components are placed as in Fig. 2. Finally, the parameters of the components are set to the correct values, as illustrated in Fig. 3. When the schematic entry is completed, the net-list file can be generated. In the case of EEVS, a native EEVS net-list file is automatically produced.

The native net-list file produced by EEVS consists of a number of statements delimited by brackets (Fig. 3). Each statement includes the information required to describe either a circuit component or a simulator command. The words at the beginning of each statement are used to describe (characterise) the statement so that the net-list converter can easily process and transform it into the desired format. After that, a series of keywords are listed. The equal sign and a list of one or more parameters, terminated by a semicolon follow each keyword. For example, the statement {Res R1 nodes=N1 Vc; value=1000; Iout=0;} in Fig. 4 represents a resistor whose reference name is R1, connected between nodes “N1” and “Vc” with a value of 1000 ohms, and “Iout”, a parameter required by some EMTP-based programs, set to zero. There is no established order at which the statements should be listed. That is, a statement can be at any line of the net-list, and can use more than one line.

Also notice from Fig. 4 that there are statements that do not describe the circuit but set some parameters for a particular simulator. The last statement in Fig. 4, for example, describes the case identification card required by some EMTP-based program. Observe that some parameters are not set to any particular value, such as “COpt” in the last statement. It is the responsibility of the net-list converter program to deal with this situation, i.e. when a particular parameter in a given statement is omitted. Default values could then be used.

For each supported circuit simulator there should be a net-list converter program to translate the input data into the format required for such simulator. Since most of the components present in one particular EMTP-based simulator eventually are similar to the components of other EMTP-based simulators, the information provided by the native net-list format of EEVS should be sufficient to create such components in the target data format. Otherwise, more data parameters could be easily added. Fig. 5 illustrates the input data file for the EMTP-based program MicroTran [5], created from the net-list file of Fig. 4, by using the appropriate format through a net-list converter program.

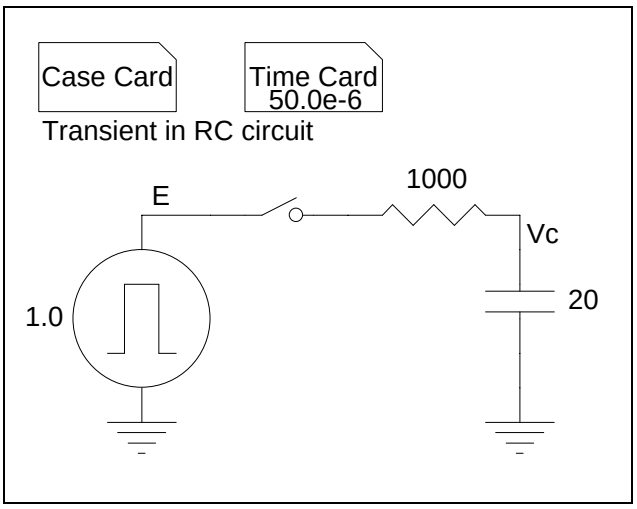

Fig. 2 – Schematic of a RC circuit.

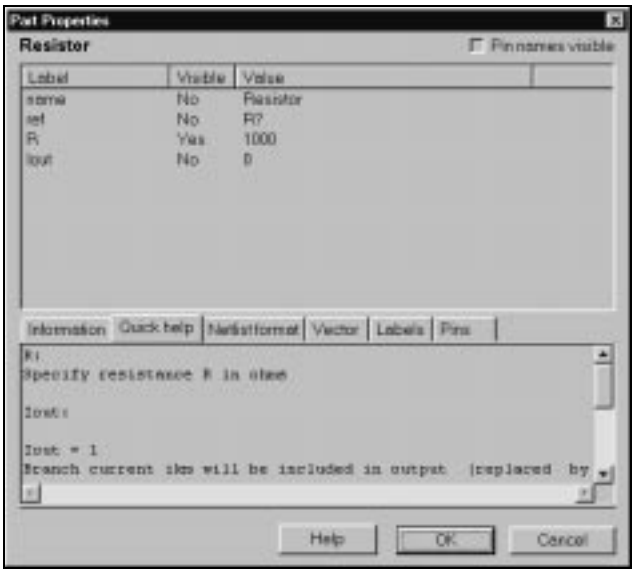

Fig. 3 – Window to input parameters for the resistor in the circuit schematic of Fig. 2.

```
{ Source volt step V1 nodes=E GROUND; Vmax=1.0;
Tstart=; Tstop=; }
{ Res R1 nodes=N1 Vc; value=1000; Iout=0; }
{ Cap C1 nodes=Vc GROUND; value=20; Iout=1; }
{ Switch Time Sw1 nodes=E N1; R=; TClose=0;
Topen=1e-3; CurrentMargin=;  Iout=; }
{ Time DeltaT=50.0e-6; Tmax=100e-3;  Iout=; IPunch=;
Iskip=; Epsilon=1e-12; Isteady=; IPower=; f_min=;
delta_f=; f_max=;  IFlux=; Ismooth=; Ishort=;
Iswitch=; IOscl=; Pulse=; }
{ CaseId Title=Transient in RC circuit; IFile=; XOpt=;
  COpt=; }
```

Fig. 4 – Net-list file generated from the schematic of the RC circuit in Fig. 2.

```
*  .  .  .  .  .  .  .  . Case identification card
Transient in RC circuit
*
*  .  .  .  .  .  .  .  . Time card
0.000050000.10000000        1.00e-12
*
*  .  .  .  .  .  .  .  . Lumped RLC branch
  N1    Vc              1000.0                                  0
  Vc                          20.000                            1
$  = =  End of level 1: Linear and nonlinear elements = = = = = = = = = = = = =
*
*  .  .  .  .  .  .  .  . Switches and piecewise linear elements
  E     N1            0.00100000
$ = = =  End of level 2: Switches and piecewise linear elements = = = = = = = =
*
*  .  .  .  .  .  .  .  . Voltage or current sources
11E      1.00000000
$ = = =  End of level 3: Sources  = = = = = = = = = = = = = = = = = = = = = = =
*
*  .  .  .  .  .  .  .  . Voltage-output nodes
 1
$ = = =  Level 5: End of data case  = = = = = = = = = = = = = = = = = = = = =
```

Fig. 5 – Input data file generated from the net-list file (Fig. 4), for a particular input format of an EMTP-based program.

IV. MAIN FEATURES OF EEVS

EEVS is an object-oriented application programmed in C++ [6], and it was build around the document/view architecture provided by the Microsoft Foundation Classes (MFC) [7]. The use of MFC provides EEVS with many of the features of commercially available professional Microsoft Windows applications. Therefore, printing, file management, help, toolbars, etc. behave in EEVS exactly as in many other Microsoft Windows popular applications. This is important, since first time users of EEVS most likely will know most of the commands of EEVS by just being familiar with other Microsoft Windows applications.

It is also possible, in principle, to port a Windows MFC-based application into other platforms, such as UNIX, Linux, etc. (see http://www.bristol.com/windu/), thus allowing the development and maintenance of a single high quality source code. Fig. 6 illustrates the actual EEVS software environment, with a schematic representation of a three-phase network, submitted to a single-line to-ground fault in phase “a”.

Under the document/view architecture of MFC, the EEVS schematic or library file becomes the document and the schematic or library display becomes the view. Any given EEVS schematic file is actually composed of five basic types of elements: parts, wires, buses, junctions, and texts. Library files (Fig. 7 and Fig. 8, for example) contain only part and text elements. For portability and accessibility reasons, in EEVS both schematic and library files are plain ASCII text files.

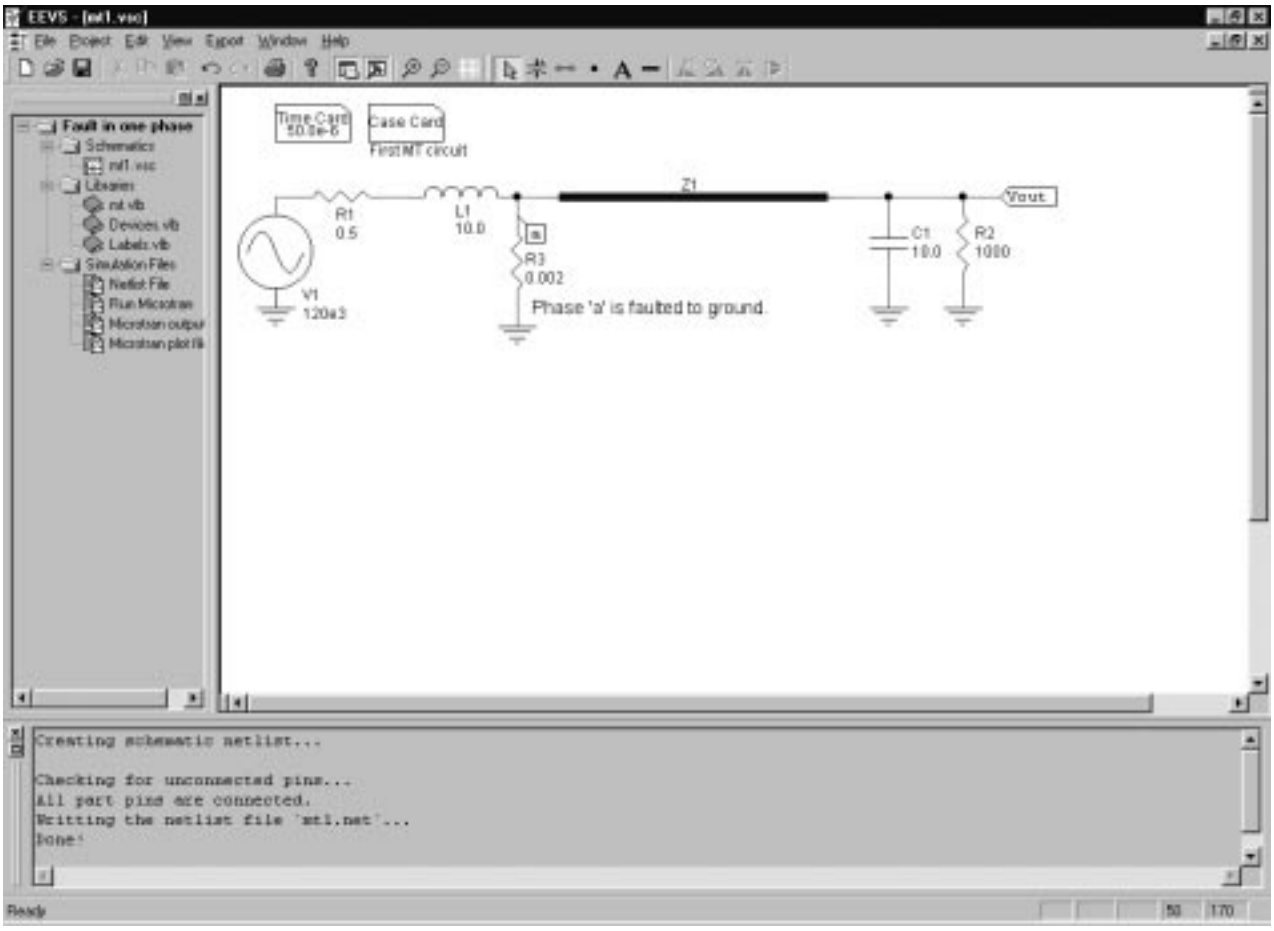

Fig. 6 – Actual EEVS software environment.

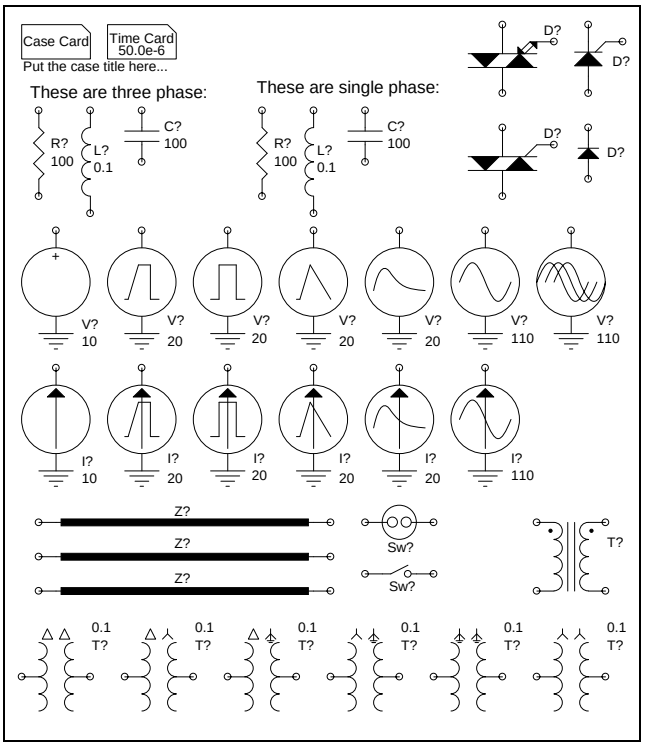

Fig. 7 – Example of a circuit components library file.

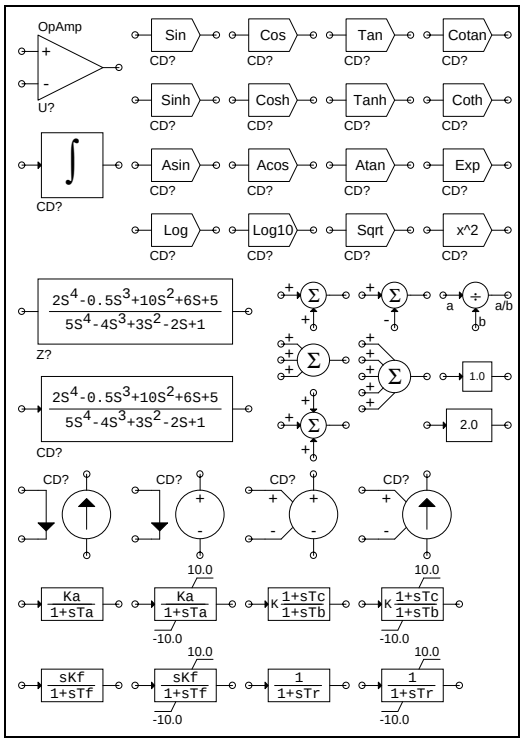

Fig. 8 – Example of a control and devices library file.

Part elements are by far the most complex elements in EEVS. Part elements can represent circuit components, such as resistors, sources, etc., but can also represent simulator commands or settings such as the "time card" in EMTP-based programs, or labels and other wiring elements such as the standard "ground" symbol. A label part element forces the name of the node to which it is connected to adopt the name of the label. This feature is very important for complex multiple schematic pages projects, since it allows connections between different schematic pages. A part element includes the format to generate its net-list, a list of vector graphics commands to draw it, the parameters required at simulation time, the points of connection with other parts of wires, and any required labels and help information associated with the part, as illustrated in Fig. 5.

Wire and bus elements are used in EEVS to interconnect part elements. Junction elements are used to join two or more wire elements when they cross each other. Bus or wire elements can be named. If two bus or wire elements have the same name, then they are virtually connected. This feature is very important since it allows EEVS to interconnect nodes that are far apart from each other in the same schematic or nodes in different schematics without actually putting a wire element connecting them. Wire elements can also adopt the name of the label part element to which they are connected, as mentioned before. For example, all wires connected to the "ground" symbol become the "ground" node. Text elements are just texts used to document the schematic.

V. EXAMPLES

Fig. 10 presents the schematic circuit of an operational amplifier [8],.and Fig. 11 illustrates an electronic circuit (single-phase diode bridge rectifier).

Fig. 12 presents the block diagram of a single-machine infinite bus system [9] for small signal stability studies, which simulation, eventually, can be performed with EMTP-based programs. Linear and non-linear control libraries can also be used by EEVS to represent detailed control block diagrams. Fig. 13 illustrates a circuit schematic to evaluate the step response of a control transfer function.

VI. CONCLUSIONS

This paper has presented the fundamental concepts for the development of a multi-format vector graphical user interface for EMTP-based programs, which is designed to allow the automatic generation of input data files for any type of EMTP-based program, as well as for electronics simulation programs, such as SPICE. The software EEVS – Electrical Engineering Vector Schematics has been developed to accomplish these requirements, so far for particular EMTP-based program, and its main features have been presented in this paper. Further developments will possibly consider the support for input formats of other EMTP-based programs.

```
<PART>
<HEADER> 0 `Resistor` 1 0 20 30 160 60 0 1 </HEADER>
<VARIABLES>
`name` `Resistor` 10 -10 1 0
`ref` `R?` 7 -15 0 0
`R` `1000` 7 -5 0 1
`Iout` `0` -10 10 1 0
</VARIABLES>
<VECTOR> line 10 00 13 03 line 13 03 07 09 line 07 09 13
15 line 13 15 07 21 line 07 21 13 27 line 13 27 10 30
</VECTOR>
<PINS> T10 `k` B10 `m` </PINS>
<NETLIST> {Res %ref% nodes=%k% %m%; value=%R%;
 Iout=%Iout%; } </NETLIST>
<HELP>
R:
  Specify resistance R in ohms

Iout = 1
Branch current ikm will be included in output (replaced by
instantaneous power if IPOWER = 1 on time card).

Iout = 2
Branch voltage vk - vm will be included in output.

Iout = 3
Branch current (or instantaneous power) and branch voltage
will be included in output.
</HELP>
<INFO> Write comments here... </INFO>
<WINHELP> `mt.hlp` 1001 </WINHELP>
</PART>
```

Fig. 9 – Definition of part components in EEVS.

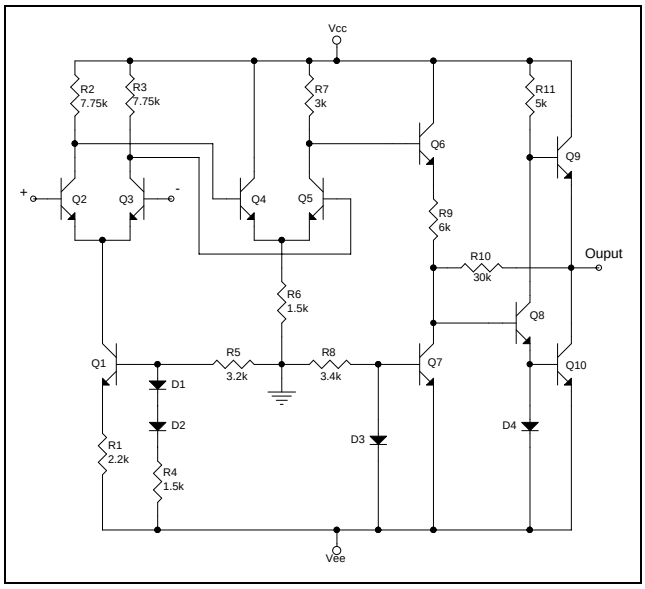

Fig. 10 – Circuit schematic of an operational amplifier.

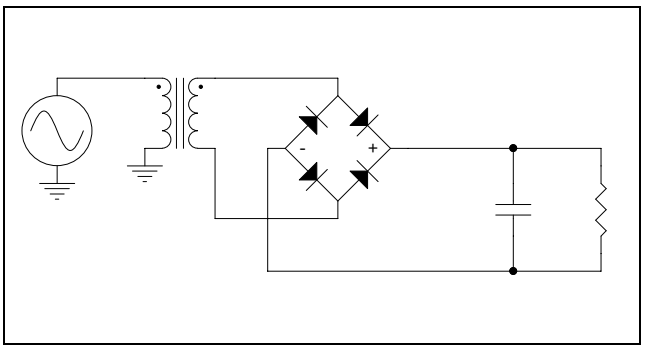

Fig. 11 – Single-phase diode bridge rectifier

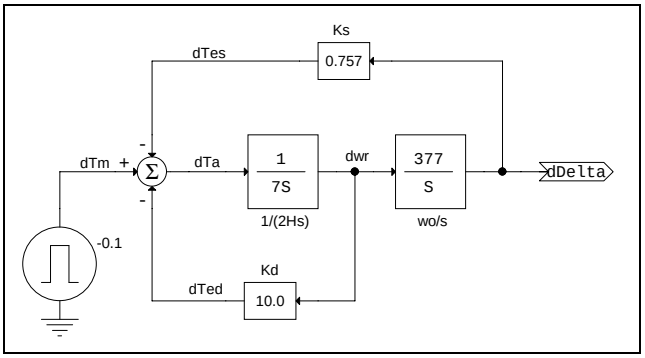

Fig. 12 – Block diagram of a single-machine infinite bus system.

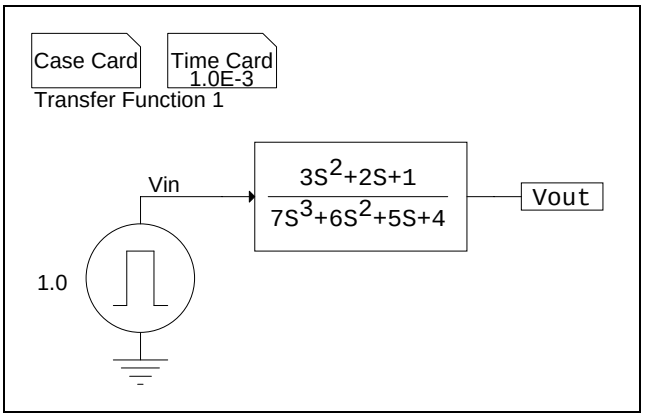

Fig. 13 – Circuit schematics to evaluate the step response of a control transfer function.

VII. ACKNOWLEDGEMENTS

The authors would like to thank the stimulating discussions and encouragement of graduate students and professors of the UBC Power Group during a presentation of EEVS – Electrical Engineering Vector Schematics at the Department of Electrical and Computer Engineering of the University of Bristish Columbia, Vancouver, B.C., Canada.

VIII. REFERENCES

[1] H.W. Dommel, "Digital Computer Solution of Electro–magnetic Transients in Single- and Multiphase Networks", *IEEE Transactions on Power Apparatus and Systems*, vol. PAS-88, April 1969, pp. 388-399.

[2] PSCAD-EMTDC web site, Manitoba HVDC Research Center, http://www.hvdc.ca/main/pscad/index.html.

[3] T. Quarles, A. R. Newton, D. O. Pederson, A. Sangioovanni-Vicentelli, *SPICE3 Version 3f3 User's Manual*, May 1993, (available for downloading at: http://www-cad.eecs.berkeley.edu/Software/software.html).

[4] Institute of Electrical and Electronics Engineers, *Complete in One Volume all the IEEE Standards and American National Standards on Electrical and Electronics Graphic Symbols and Reference Designations*, 2nd edition, New York: Institute of Electrical and Electronics Engineers, 1987.

[5] Microtran Power System Analysis Corporation, *MicroTran® Reference Manual*, Vancouver, B.C., Canada, Aug. 1997, (available for downloading at: http://www.microtran.com/).

[6] Stanley B. Lippman, *C++ Primer*, 2nd Edition, Addison-Wesley, 1997.

[7] Bryan Waters, *Writing Windows Applications with Microsoft Foundation Classes*, M&T Books, New York, 1994.

[8] Jacob Millman and Christos C. Halkias, *Integrated Electronics Analog and Digital Circuits and Systems*, McGraw-Hill Inc., New York, 1972. pp. 515

[9] Prabha Kundur, *Power System Stability and Control*, EPRI Series, McGraw-Hill Inc., New York, 1993.