Int. J. Engng Ed. Vol. 18, No. 6, pp. 704–710, 2002
Printed in Great Britain.

0949-149X/91 $3.00+0.00
© 2002 TEMPUS Publications.

Graphical Simulation of an Analog Computer Using Spreadsheets*

ALI EL-HAJJ, SAMI KARAKI and KARIM KABALAN
Electrical and Computer Engineering Department, Faculty of Engineering and Architecture, American University of Beirut, P.O. Box 11–0236, Riad El Solh Beirut 1107 2020, Lebanon. E-mail: kabalan@aub.edu.lb

This paper presents a spreadsheet method for the graphical simulation of analog computers (AC) used in control systems. It is based on simulating basic AC blocks like adders, integrators, potentiometers, inverters, and nonlinear devices. To make the simulation simple and user friendly, the construction of an AC system is done graphically, whereby blocks are drawn at the simple clicks of buttons. Blocks can then be easily connected using a built-in graphical interface. The initialization and running of a given system is fully menu-driven and done using toolbars and buttons. The simulation of a system allows the calculation and plotting of its time response for any input signal. This toolbar-customized simulation is characterized by its low cost, flexibility, and simplicity. The procedure for building the graphical symbols and toolbars is briefly described, and illustrative examples are presented to show the capabilities of the developed simulation system.

INTRODUCTION

SPREADSHEETS were initially conceived to solve problems in business and accounting applications. Their flexibility and wide availability have also made them well established in mathematics, physics and engineering, primarily for educational purposes [1–3]. More recently they were also used to simulate engineering systems, such as logic networks and control systems [4]. In simulating engineering systems, basic building blocks are developed and connected together using spreadsheet formulas. In logic circuits, the basic blocks are logic gates, flip-flops, clocks, and MSI circuits, which are used to simulate combinational, sequential, synchronous and asynchronous networks. In linear control systems, the basic blocks are essentially adders, and integrators. Any transfer function can be simulated in the s domain by connecting together a number of integrators and adders with appropriate scaling coefficients. An overall control system is consequently simulated by connecting together the constituent transfer functions.

Nonlinear control systems are also simulated by using nonlinear elements in some of the basic blocks. Sampled data control systems are also simulated using the delay z^{-1} as a basic block. By connecting a number of z^{-1} blocks using appropriate scaling coefficients it is possible to simulate any transfer function in the z domain. This simulation allows the calculation of the time response of the control system for any input signal.

In this work, a specially developed toolbox is described to graphically simulate the usual components of analog computers used in control systems. This toolbox was constructed by following the simulation method introduced in [4] and summarized in the next section. The toolbox interface makes use of toolbars and click-on buttons to make the simulation simple and user friendly. The basic blocks used are adders, integrators, potentiometers, inverters, and some nonlinear devices. Each of these basic blocks can be inserted in a system design by clicking at a corresponding button of the toolbar. Additional toolbar buttons allow the connection between blocks. Other buttons are added to initialize the system or to run it for a certain number of iterations.

This method has primarily the following advantages:

1. It is of low cost since spreadsheets are basic software packages widely available in many institutions.
2. It is easy to learn by users familiar with spreadsheet programs.
3. It allows the user to build an insight into the behavior of a system, and to determine the effect of changing one or more parameters on this behavior.
4. The graphical display of the results via the chart toolbar provides the user with a quick feedback needed in the repetitive analyses of a design exercise.

The process of generating the basic blocks and connecting them using spreadsheet tools is first explained in some detail. Then, the development of the various analog computer blocks is described. Two illustrative system simulation examples are then presented and the results obtained from the developed toolbox are compared to the analytical

* Accepted 21 March 2002.

ones. The method is discussed and concluding remarks are finally given.

THE SIMULATION TOOLS

The spreadsheet tools used in the simulation are graphics, formulas, macros, and toolbars. To develop a basic block, the drawing toolbar is used to graphically represent this block using its conventional symbol usually found in textbooks. Formulas are then used to calculate the block output as a function of the inputs and other parameters. These two steps are recorded and the corresponding code is generated by Excel and stored in a macro. The macro code, which is in Visual Basic, is edited to ensure that all drawings and references are relative to the current active worksheet cell. The macro is then attached to a toolbar button, which can be clicked to execute the macro and cause the block to be generated at the location of the active cell.

For example, to create an adder, the drawing toolbar is used to draw the adder with three inputs at cells B6, C6, and D6 and one output C9, as shown in Fig. 1. The inputs are multiplied with scaling factors stored at cells B7, C7, and D7, and initialized to 1, 1, and 10. These scaling factors can be directly modified by the user on the worksheet. The adder output should be in the range of the amplifiers saturation levels, which are the same for all amplifiers and are stored in cells I2 and I3. The formula:

=MAX(MIN(–B7*B6–C7*C6–D7*D6;$I$3);$I$2)

that relates the adder's output to its inputs is written at cell C9. This process of drawing and writing formulas is recorded in a macro called 'Adder'. The Adder macro code is edited to ensure that all drawings and references are relative to the current active worksheet cell. This macro is linked to a button called 'Adder' included in a toolbar called 'Analog Computer'. This toolbar is constructed to graphically simulate the devices usually found in an analog computer. In Fig. 1, to generate the second adder, cell E11 is made the active cell and the Adder button is clicked.

In order to manage the simulation process, initialization and running mechanisms are implemented by using some variables stored in the upper rows of the worksheet. An initialization flag stored in cell B1 is set to zero when no calculation is required on the worksheet, and is set to 1 when calculation is required. Some blocks use this flag to reset their state to an initial condition. A counter stored in cell B2 indicates the number of iterations (i.e. spreadsheet calculations) that are performed. This counter is initialized to zero and incremented for each worksheet calculation. This is done by writing at cell B2 the formula:

=IF($B$1=0,0,B2+1).

The integration step is stored in cell B3, and the time is calculated and stored in cell B4 using the formula:

=$B$2*$B$3.

A spinner is used in cells F1:F2 to set the number of worksheet calculations to be performed in a single run. The initialization mechanism is done using the 'Initialize' button with an underlying macro that resets the initialization flag and the counter to zero. When this macro is invoked for the first time in a new worksheet, it also generates the spinner and writes the saturation cells in range H1:I3 . Resetting is needed before starting any new run. The running mechanism is done using the 'Go' button, which has an underlying macro to set the initialization flag to one and recalculate the worksheet a number of times indicated by the spinner. It is possible to execute the 'Go' macro several times, consecutively, using different spinner values.

The simulation of a system requires connecting the output cell of one block (source) to an input cell of another block (destination). This is done by drawing a line that connects the centers of the two cells and by writing at the destination cell a formula equal to the source cell address. This is done in two steps. In the first step the source cell is made active and the 'From' button is clicked. The underlying macro calculates and saves the address of the source cell. In the second step the destination cell is made active and the 'To' button is clicked. The underlying macro calculates the address of the destination cell, writes in this cell a formula equal to the source cell address, and draws a line that connects the centers of the source and destination cells. The 'To' button has effect when clicked directly after the 'From' button.

For example, in Fig. 1, to connect the first adder output to the second adder input cell C9 is made active, the 'From' button is clicked, then cell D11 is made active and the 'To' button is clicked.

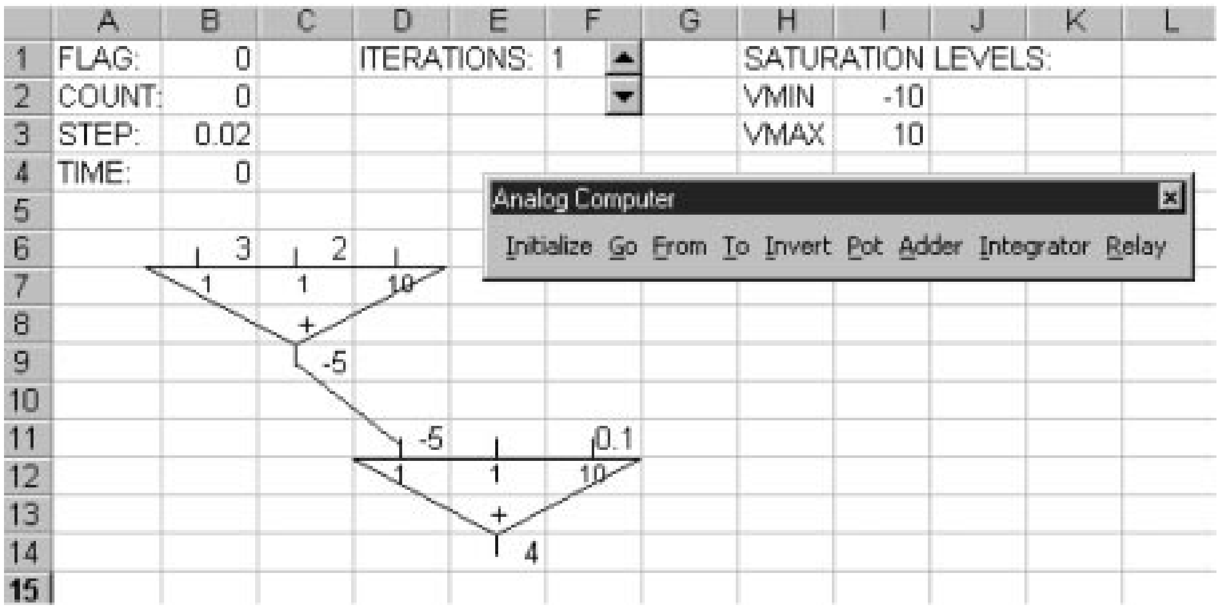
ANALOG COMPUTER BLOCKS SIMULATION

Other AC blocks are simulated following an approach similar to that of the adder described in the previous section. These blocks are the inverter, the potentiometer, the integrator, and the relay, which is a nonlinear device.

In this simulation, the inverter of Fig. 2 is obtained by making cell B7 active and then clicking the 'Invert' button. The output formula at cell B9 is '–B7'. The potentiometer in the same figure is obtained by making cell D7 active and then clicking the 'pot' button. The pot coefficient is stored at cell E8 and initialized to one. This coefficient can be directly changed by the user but cannot be set to more than one.

This feature is implemented by using at the output cell D10 the formula:

=D7*MIN(E8;1)



FLAG: 0
COUNT: 0
STEP: 0.02
TIME: 0
ITERATIONS: 1
SATURATION LEVELS:
VMIN -10
VMAX 10

Fig. 1. Simulation tools.

In the case of an integrator, the output $y(t)$ is obtained as function of the input $x(t)$ by solving the equation:

$$\frac{dy}{dt} = x(t) \qquad (1)$$

with $y(0) = y_0$. Many methods discussed in [5] can be used to calculate the output sequence y_i given an input sequence x_i. The trapezoidal rule is chosen with an integration step h as follows:

$$y_{i+1} = y_i + \frac{(x_i + x_{i+1})}{2} h \qquad (2)$$

Consider the integrator with three inputs B6, C6, and D6 and one output C10 shown in Fig. 3. The inputs are multiplied with scale factors stored at cells B7, C7, and D7, and initialized to 1, 1, and 10. These scaling factors can be directly modified on the worksheet by the user. The initial value is stored in cell B9 and assigned a default value of zero when the integrator is generated but can be directly modified to y_0 on the worksheet by the user.

The input x_{i+1} is calculated in cell D8 using the formula:

=–B7*B6–C7*C6–D7*D6

Due to the row-wise calculation of a worksheet, at every iteration the previous input value x_i is stored in cell C8, which is calculated before cell D8, using the formula:

=IF($B$1=0;0;D8).

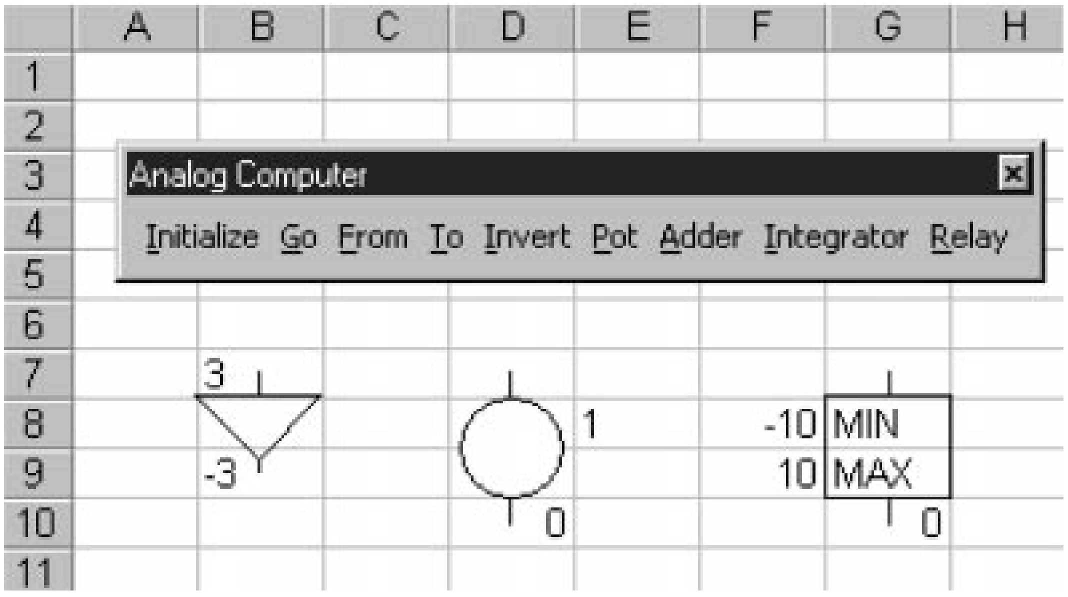
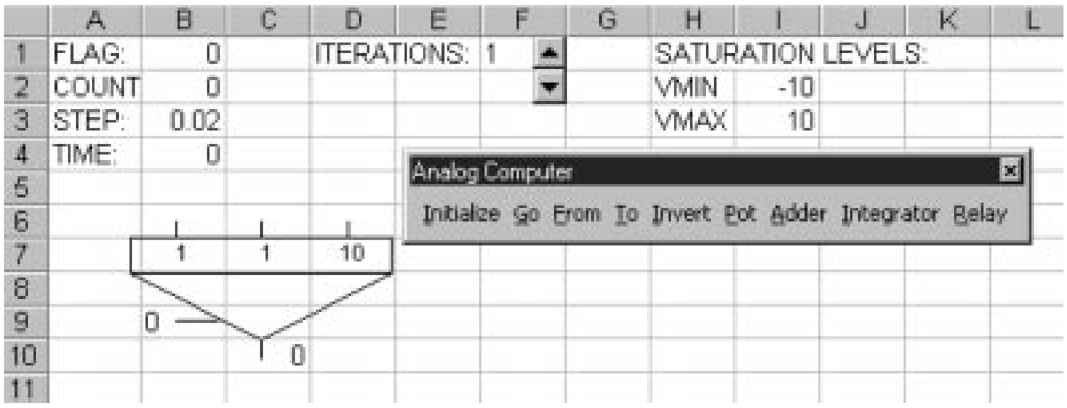


Fig. 2. Inverter, potentiometer, and relay simulation.



	A	B	C	D	E	F	G	H	I	J	K	L
1	FLAG:	0		ITERATIONS:		1		SATURATION LEVELS:				
2	COUNT	0						VMIN	-10			
3	STEP:	0.02						VMAX	10			
4	TIME:	0										
5												
6												
7		1	1	10								
8												
9		0										
10			0									
11												

Fig. 3. Integrator simulation.

Since the output y_{i+1} is stored at cell C10, its previous value y_i is stored in a cell C9, which is calculated before cell C10, using the formula:

=IF($B$1=0;B9;C10).

The output is obtained at cell C10 by implementing Equation (2) as follows:

=MAX(MIN(IF($B$1=0;B9;
C9+0.5*$B$3*(C8+D8));$I$3);$I$2)

As in the case of the adder, this formula takes into account the saturation levels. The integrator generation process is automated using the 'Integrator' toolbar button.

Nonlinear devices can be simulated in a similar way. It is possible to use formulas in order to implement an operation that is not simulated by a block. As an example, consider the simulation of the relay of Fig. 2 with input G7 and output G10. The relay levels are assigned default values of –10 and 10, which can be modified by the user directly on the worksheet. The output formula in cell G10 is given by:

=IF(G7<0;F8;IF(G7>0;F9;0)).

The relay generation process is also automated using the 'Relay' toolbar button.

ILLUSTRATIVE EXAMPLES

Two examples are used to illustrate the simulation of analog computers and compare the results with available solutions. First, consider the linear system shown in Fig. 4, with its spreadsheet simulation shown in Fig. 5. Since Excel calculates its cells row-wise from left to right, the order and the placement and connections of devices should be carefully done. The value of K is stored in cell B5 and cell D11 contains the formula '=B5'. The unit step response is calculated by applying a unit value at the system input in cell D5. The corresponding system time responses are arranged in adjacent columns. For example, in order to record the system output for counter values 0, 20, 40 . . . , 2400, these values are filled in the range A22: A142. The corresponding time values are stored in range B22:B142 by multiplying the counter values in range A21:A142 by the step length given in cell B3.

Successive outputs at cell D18 (named OUTPUT) are stored in range C22:C142 by writing in cell C22 the formula:

=IF($B$1=0;0;IF(A22=$B$2;OUTPUT;C22))

This formula is copied to the range C23:C142. The Excel chart toolbar is then used for plotting. The results are almost identical with the solution obtained and given in [6].

As a second example, consider the differential equation $y'' + 6y' + 5y = 0$ with the initial conditions $y(0) = 2$ and $y'(0) = -2$. A spreadsheet simulation of this equation obtained after scaling is shown in Fig. 6. The normalized output obtained is $-2y(t)$. The calculated solution, $y(t)$, is obtained at cell C21 using the formula: '–C20/2'. The calculated solution is plotted by following a similar approach to that of the previous example. This solution is in close agreement with the exact solution given by $y(t) = 2\exp(-t)$ and plotted on the same graph.

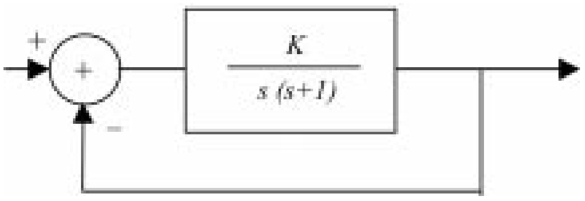
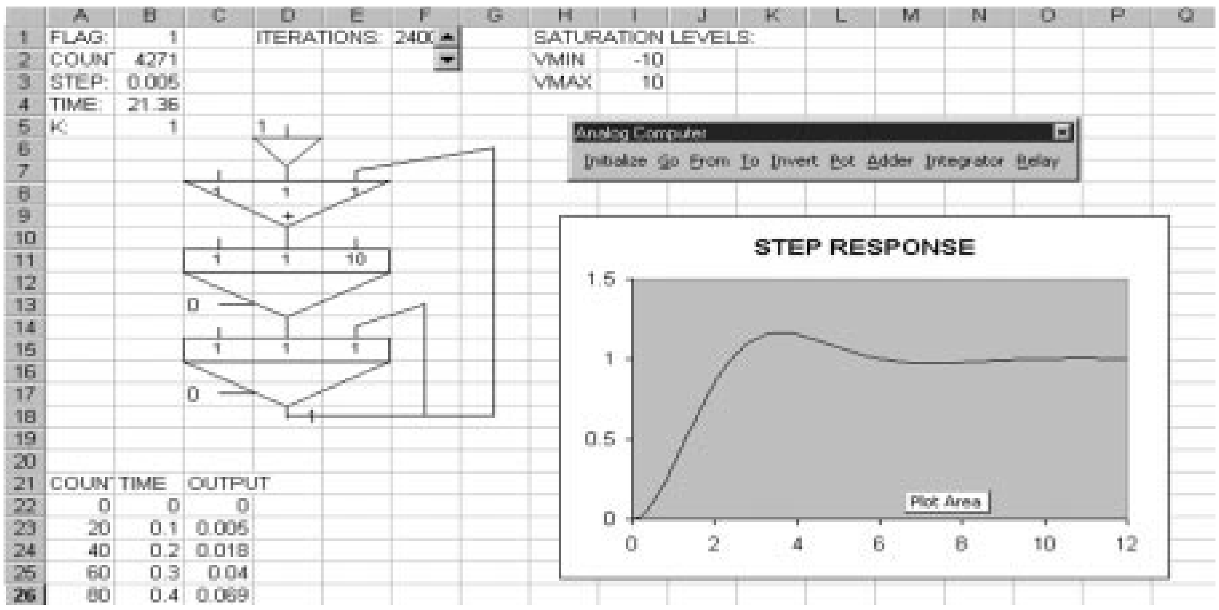


Fig. 4. A Simulation example.

DISCUSSION

The basic task facing computer simulation of a given process is that of providing a cost-effective facility for building the object from one end of the system at a rate and a level of reliability and quality that are acceptable to a user at the other end. These key parameters are integrated in this design process and can be outlined as follows:



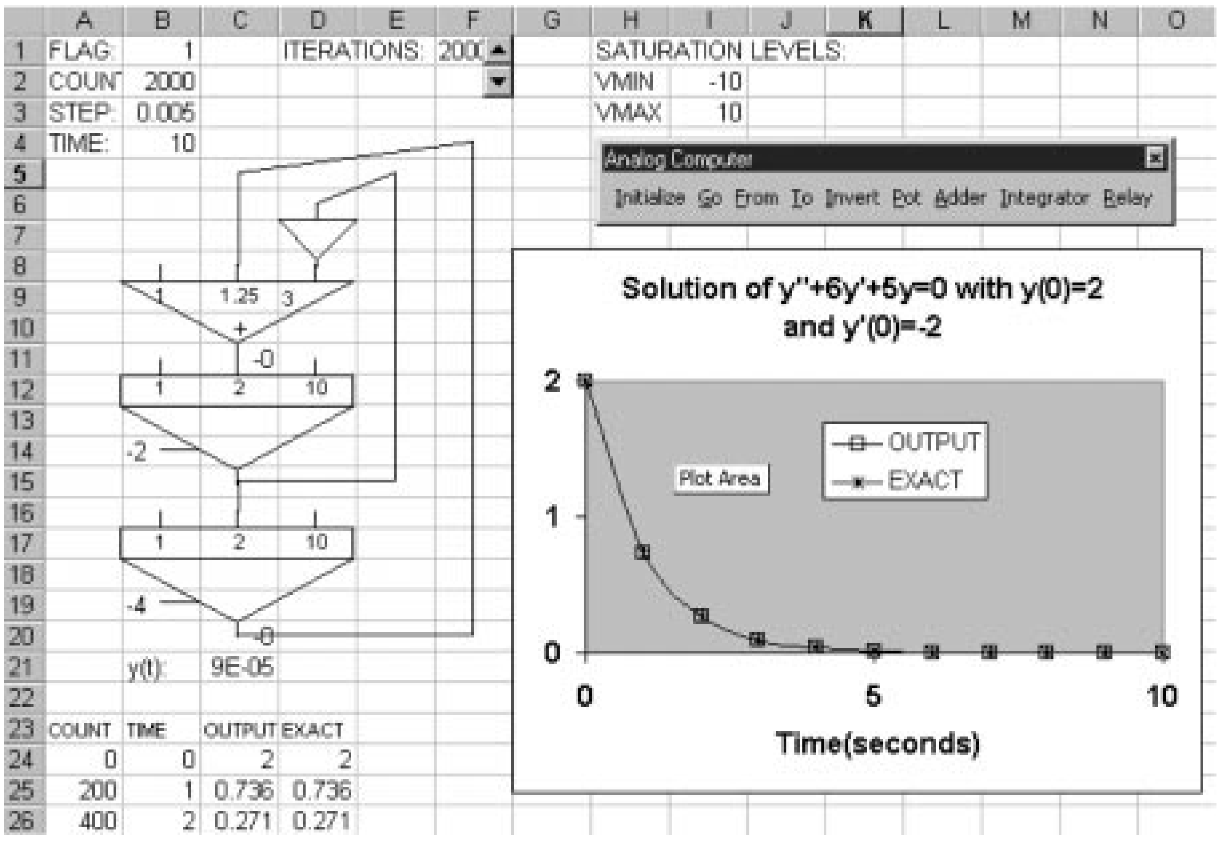
FLAG:	1		ITERATIONS:	2400
COUN	4271			
STEP:	0.005			
TIME:	21.36			
K:	1			

SATURATION LEVELS:	
VMIN	-10
VMAX	10

COUN	TIME	OUTPUT
0	0	0
20	0.1	0.005
40	0.2	0.018
60	0.3	0.04
80	0.4	0.069

Fig. 5. Spreadsheet simulation of the system shown in Fig. 4.

1. The current simulation method relies on the use of spreadsheets which are widely available on most computers, definitely more than other related simulation packages such as MATLAB-Simulink, and are familiar to a large range of users. Thus, no extra cost is required and as such it is very useful when the user has no time or means to access a sophisticated package. It is useful in particular in educational settings where budgets and resources are sometimes limited.
2. Being familiar with spreadsheets, it is possible to develop this application using macros and formulas. Excel automatically generates the code (in Visual Basic) after recording different tasks in macros. This code consists



FLAG:	1		ITERATIONS:	2000
COUN	2000			
STEP:	0.005			
TIME:	10			

SATURATION LEVELS:	
VMIN	-10
VMAX	10

y(t): 9E-05

COUNT	TIME	OUTPUT	EXACT
0	0	2	2
200	1	0.736	0.736
400	2	0.271	0.271

Analog Computer
Initialize Go From To Invert Pot Adder Integrator Relay
Solution of y''+6y'+5y=0 with y(0)=2 and y'(0)=-2
Plot Area
OUTPUT
EXACT
2
1
0
0 5 10
Time(seconds)

Fig. 6. A second simulation example.

of a few pages that can be directly accessed by users.

3. The size of the worksheet containing the code and the toolbar is in the order of 80 kilobytes. Thus only limited computer resources are needed to run the application.
4. Furthermore, this method is easy to use and simply requires learning the function of each of the nine toolbar buttons.
5. For analysis and results, the spreadsheet data processing facilities can be used, which allows an iterative step-by-step or continuous run of the simulated system. This procedure shows the values at the output of any block in the worksheet. The outputs with different parameter value(s) can be simply obtained and plotted using the what-if feature of spreadsheets.

This method is also sufficiently accurate for a large number of practical applications, since Excel stores numbers and performs calculations using 15 digits of precision. To increase the accuracy of the results, the integration step can be reduced or more sophisticated formulas can be used in simulating some blocks. However, reducing the integration step may lead to an increase in the number of iterations (worksheet calculations), which makes this method slow in simulating large control systems.

CONCLUSIONS

A method is presented that graphically simulates analog computers using modern spreadsheet programs, which are available and familiar to a wide audience of computer users. The method is very convenient for a quick simulation when one does not have the time or means to write a sophisticated program or to access an advanced simulation package. It is also very useful for educational purposes to simulate a system in a step-by-step fashion, while recording the output values of any block in the worksheet, and then to repeat the calculations with different parameter values. The easy and quick plotting of results provides the user with a good feedback when carrying out repetitive analyses studies of a given design proposal. This method is sufficiently accurate for a large number of practical applications since Excel stores numbers and performs calculations using 15 digits of precision. To increase the accuracy of the results, however, it is possible to reduce the integration step or to use more sophisticated formulas in simulating some blocks. For example, the trapezoidal rule used to simulate the integrator may be replaced by a higher order rule [5].

REFERENCES

1. M. Hagler, Spreadsheet solution of partial differential equations, *IEEE Trans. Education*, **30**(3), August 1987, pp. 130–134.
2. T. T. Crow, Solutions to Laplace's equation using spreadsheets on a personal computer, *American Journal of Physics*, **55**(9), September 1987, pp. 817–823.
3. F. R. Shapiro, The numerical solution of Poisson's Equation in a pn diode using a spreadsheet, *IEEE Trans. on Education*, **38**(4), November 1995, pp. 380–384.
4. A.El-Hajj, Functional simulation using spreadsheets, *SIMULATION*, **73**(2), August 1999, pp. 80–90.
5. S. C. Chapra and R. P. Canale, *Numerical Methods for Engineers*, New York: McGraw-Hill (1990).
6. R. C. Dorf, and R. H. Bishop, *Modern Control Systems*, Addison Wesley, 1995.

Ali El-Hajj was born in Aramta, Lebanon. He received the Lisense degree in Physics from the Lebanese University, Lebanon in 1979, the degree of 'Ingenieur' from L'Ecole Superieure d'Electricite, France in 1981, and the 'Docteur Ingenieur' degree from the University of Rennes I, France in 1983. From 1983 to 1987, he was with the Electrical Engineering Department at the Lebanese University. In 1987, he joined the American University of Beirut where he is currently Professor of Electrical and Computer Engineering. His research interests are numerical solution of electromagnetic field problems and engineering education.

Sami H. Karaki is an associate professor of electrical engineering at the American University of Beirut (AUB), Beirut, Lebanon. He joined AUB in 1991 and contributed to the development of its Electric Power Engineering program. From 1981 to 1990 he was with the Kuwait Institute for Scientific Research, Kuwait where he contributed to two regionally leading projects on the power system interconnection of Arabic countries. He obtained his BE from AUB in 1975 and his Ph.D. from the University of Manchester Institute of Science and Technology, UK, in 1980. His main fields of interest are in renewable energy systems modeling, power system planning, short-term load forecasting, and artificial intelligence applications in power systems.

Karim Y. Kabalan was born in Jbeil, Lebanon. He received the B.S. degree in Physics from the Lebanese University in 1979, and the M.S. and Ph.D. degrees in Electrical Engineering from Syracuse University, in 1983 and 1985, respectively. During the 1986 Fall semester, he was a visiting assistant professor of Electrical Engineering at Syracuse University. Currently, he is a Professor of Electrical Engineering with the Electrical and Computer Engineering Department, Faculty of Engineering and Architecture, American University of Beirut. His research interests are numerical solution of electromagnetic field problems and software development.