Teaching Linear Circuit Analysis Techniques with Computers

Dr. Brian J Skromme, Arizona State University

Dr. Brian Skromme is a professor of Electrical, Computer, and Energy Engineering and assistant dean of the Fulton Schools of Engineering at Arizona State University. He holds a Ph.D. in Electrical Engineering from the University of Illinois at Urbana-Champaign and was a Member of Technical Staff at Bellcore from 1985 to 1989.

Mr. Qiao Wang, Arizona State University
Paul Rayes, Arizona State University

Paul Rayes is an undergraduate student studying towards a B.S. in Electrical Engineering at Arizona State University. His interests include digital and solid-state circuits and computer programming. He is a member of the Society of Hispanic Professional Engineers and the Institute of Electrical and Electronics Engineers.

John M Quick, Arizona State University

John M. Quick is an Educational Technology doctoral candidate at Arizona State University who is interested in the design, research, and use of educational innovations. He has been active in the creation of both entertainment and serious games. His current research explores the intersections of individual characteristics, enjoyment, learning, and video games.

Prof. Robert Kenneth Atkinson, Arizona State University

Dr. Robert Atkinson is an associate professor with a joint appointment in the School of Computing, Informatics, and Decision Systems Engineering in the Ira A. Schools of Engineering and the Division of Educational Leadership and Innovation in the Mary Lou Fulton Teacher's College.

Dr. Tim Frank, South Mountain Community College

Teaching Linear Circuit Analysis Techniques with Computers

Abstract

We describe recent progress in the development of a step-based computer-based tutoring system to aid in the teaching of introductory linear circuit analysis courses, and preliminary assessment of its effect on student learning in a controlled trial. The system is based on a software engine that can generate problems with random circuit topologies of specified characteristics as well as full solutions performed using techniques typically taught in introductory classes. A graphical circuit editor, an equation entry system using pre-defined templates for each term, a simplified algebraic and matrix equation entry system have been implemented to accept and evaluate a wide variety of student inputs, rather than just numerical answers. Pedagogical features such as color coding and labeling of specific currents and voltages have been implemented to clarify the origin of node and mesh equations describing a circuit. Tutorials have been developed covering identification of series and parallel circuit elements, and writing of node and mesh equations. A laboratory-based evaluation of two of these tutorials using paid student volunteers showed that they are about 10X as effective as conventional textbook exercises in promoting student learning of these topics when used for the same period of time, with a statistically significant difference. The effect size of the tutorial usage is found to be 1.21 pooled standard deviations (i.e., a Cohen *d*-value of 1.21). This type of system is therefore expected to be a great improvement over conventional homework, when fully implemented.

1. Introduction

In a previous paper,[1] we described the motivation and goals of our project to develop software to aid in the teaching of introductory linear circuit analysis, and briefly reviewed the prior literature in this area. As noted there, this course is foundational in most electrical engineering programs and is also taken by many other engineering majors. Many students face difficulties in successfully completing the course and retaining this material. Conventional homework assignments provide delayed feedback, and even answer-based tutoring systems such as those available with many textbooks only inform students of right or wrong answers, with no indication of where the student may have gone wrong in a lengthy problem. Also, conventional textbooks provide a limited set of examples, which may be insufficient for many students, and a limited set of potential problems, which may encourage inappropriate collaboration or copying, even if the numbers are varied in a problem by an on-line system ("algorithmic" problems). Errors are also often present, which may further confuse students.

Our system provides an unlimited supply of problems or examples with fully-worked solutions (not just answers), of user-specified difficulty and complexity, and free of errors. By accepting a rich variety of student inputs, such as equations, re-drawn circuit diagrams, student-generated waveform sketches, matrix equations, in addition to simple numerical answers or multiple choice inputs, we provide immediate, detailed feedback that prevents students from wasting time and becoming frustrated by solving incorrect equations or deriving equations from incorrect circuit diagrams. We also aim to provide exercises that specifically target common student misconceptions, to develop both qualitative and quantitative understanding of the topics. We include a variety of pedagogical features designed to help students understand the material, such as color-coding and labeling on the circuit diagrams.

In the following, we describe the addition of new features to our core circuit generation algorithms, a new graphical circuit editor, a new structured equation entry system, facilities to input and check simplified systems of algebraic equations and the corresponding matrix equation as well as numerical answers, and the addition of pedagogical features to explain the origin of various terms in node and mesh equations. We further describe three tutorials that cover identification of series and parallel circuit elements, and writing of node and mesh equations. Finally, we discuss initial utilization of the software in a circuits class and the results of a controlled laboratory trial comparing the impact on student learning of software usage to that of conventional homework exercises.

2. New Software Features

2.1. New Features of Circuit Generation Algorithms

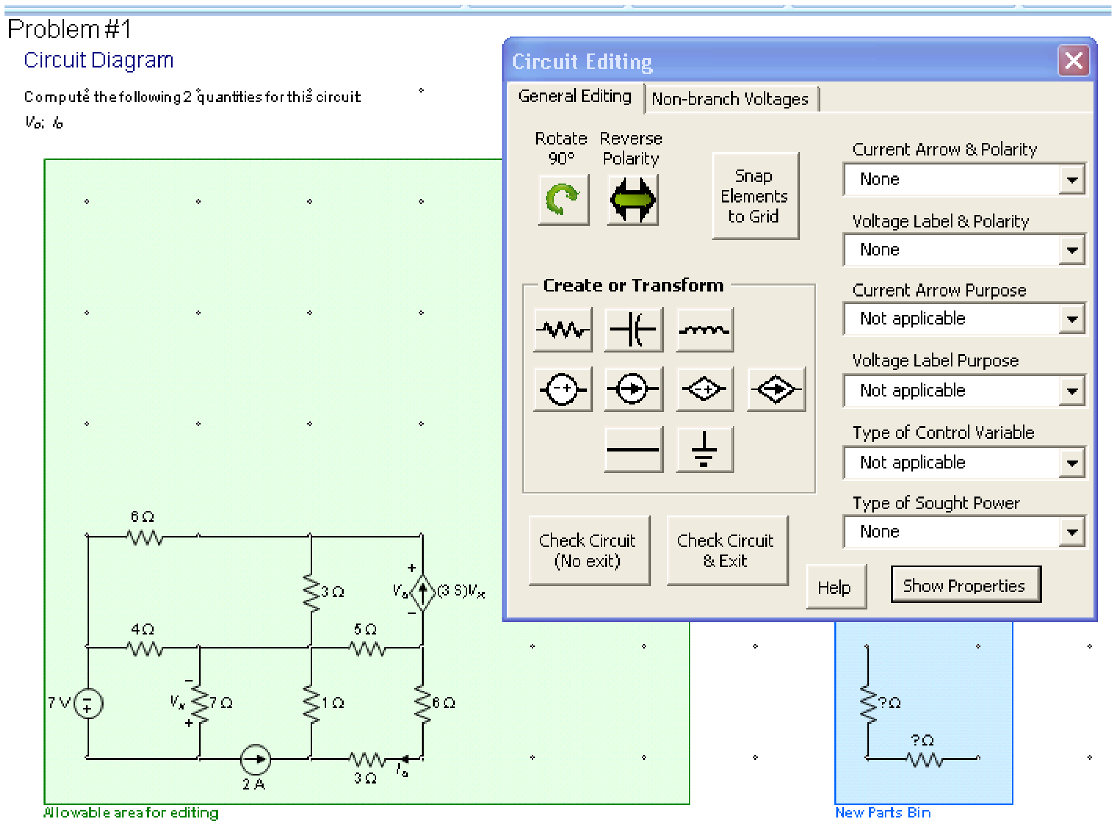
Our basic circuit generation algorithms were described previously.[1] We now include optional specification of the number of floating supernodes (i.e., supernodes not connected to the reference node), the number of supermeshes, and whether or not series and/or parallel passive elements of the same type (such as resistors) are allowed. Supernodes and supermeshes are concepts used to permit writing Kirchoff's current law equations involving nodes attached to floating voltage sources, or to permit writing Kirchoff's voltage law equations for meshes involving internal or shared current sources, respectively. Each supernode consists of a set of two or more nodes all connected by independent or dependent voltage sources.[2] Therefore, we control the number of floating supernodes by re-positioning already placed voltage sources as needed on a selected tree of the layout to increase or decrease the total number of supernodes as needed, and then choose the reference node at a location that leaves the appropriate number floating. A supermesh consists of two or more meshes with shared "internal" current sources between them.[2] Re-positioning current sources to control the number of supermeshes is extremely complicated. Therefore, we simply reject circuits with the wrong number of supermeshes until the desired number is achieved, which is generally a very effective method. These features are used to control the difficulty of node or mesh analysis problems presented to students, such as whether or not a supernode or supermesh is required to solve the problem.

Series and parallel passive elements of the same type are prevented, when desired, by selecting a different tree of the network when they occur during a random element placement process, and then re-placing the circuit elements. Many textbook problems avoid such elements in series or parallel because they can be combined prior to solution. The software already includes a provision to limit the number of independent voltage sources in series and independent current sources in parallel, since those can also be combined.

2.2. Graphical Circuit Editor

Circuit solutions often involve re-drawing a given circuit diagram, such as when combining elements in series and parallel, performing source transformations, applying superposition, or when deriving Thévenin and Norton equivalents.[2] We therefore wish to allow students to redraw circuit diagrams as needed during a solution process via a graphical interface, so that their work can be checked at each step. We have therefore implemented a circuit drawing/re-drawing interface, as illustrated in Fig. 1. Any existing element can be changed into any other simply by selecting it and clicking on the desired type of element. Element values can be edited directly on the display. New elements can be added by dragging them out of the "New Parts Bin," and the type of parts displayed there is controlled by clicking on a button for the desired type. When positioning them, they automatically snap to our pre-defined grid to facilitate alignment. Elements can also be rotated 90^o or their polarity can be flipped (for elements such as sources that have polarities). By clicking on a particular element, current arrows or voltage labels can be applied to it using the drop-down menus on the right side of the dialog box, which are used as control variables for dependent sources, and/or as the "sought quantities" such as currents, voltages, or powers for which a student is asked to solve (the latter feature would not normally be made available when a student is editing an existing circuit). A ground symbol can be placed or relocated. Once the circuit is edited (which can include creating a circuit from scratch), it is automatically checked for validity against the criteria discussed previously.[1] Additional editing must be performed if it is not valid.

The ability to store circuits to disc or load them is also included. This facility can be used by instructors or tutorial writers to create specific circuits. This initial version of the circuit editor allows editing on a PowerPoint slide. We plan however to revise this system so that editing is performed on a form instead, which will enable a greater degree of control over the user interactions. We are further in the process of developing the ability to check edited circuits as


Problem #1

Circuit Diagram

Compute the following 2 quantities for this circuit

V_o; I_o

Circuit Editing

Current Arrow & Polarity: None

Voltage Label & Polarity: None

Current Arrow Purpose: Not applicable

Voltage Label Purpose: Not applicable

Type of Control Variable: Not applicable

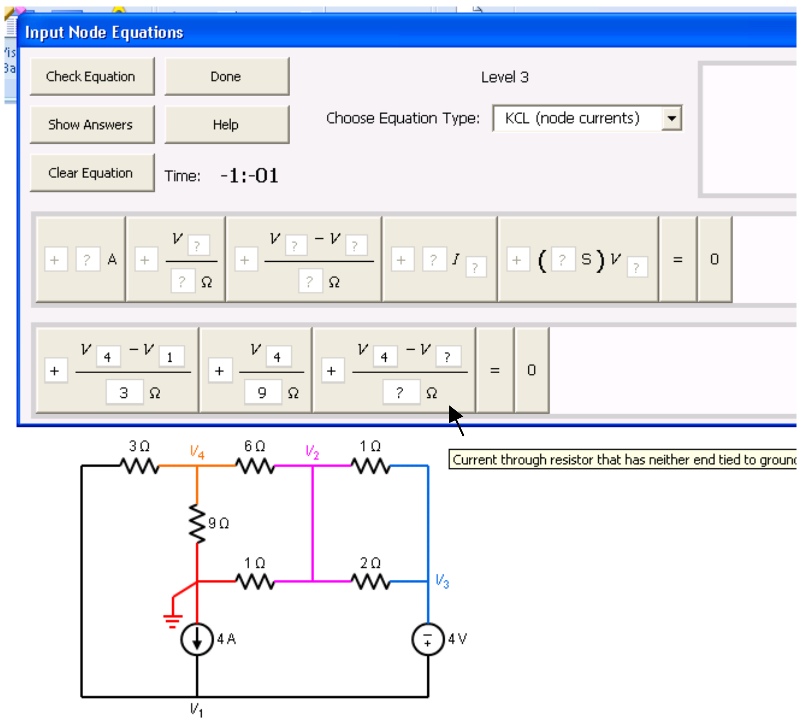
Type of Sought Power: None

Fig. 1. Interactive circuit editor implemented in PowerPoint.

part of the problem-solving process, to determine if a student has performed a valid modification during editing.

2.3. Equation Entry System

In our step-based tutorial system we need to be able to accept student input in the form of algebraic equations, rather than just final numerical answers. We have chosen to implement a system where the student is provided templates for each type of term that can properly occur in a given type of equation, to help provide guidance in learning to write them. An example of this forms-based interface is shown in Fig. 2. A palette of term types is shown across the middle row, which can be dragged down to the equation entry area in the bottom row. This palette is adjusted to display term types appropriate to the selected equation type (KCL equation in the case shown). Terms can be dragged to re-order them or delete them as necessary. After selecting the appropriate terms to form a given equation, the user fills in blanks such as element values, subscripts, and signs to complete the equation. A count-down timer can be used to offer assistance when the user takes too long to compete a given equation. When "Check Equation" is clicked, the equation is checked against all possible equations of the specified type that have not already been entered, and designated as right or wrong. Syntax errors are also noted (such as having two equals signs, or none at all.) The user can attempt to fix the equation if it is incorrect, or can ask to see the answer (in which case another problem of the same type and difficulty will be presented). In future versions, we plan to ask the user which node or source the equation is being entered for, in which case more detailed feedback could be provided. Observations of student users suggest that they understand this interface very readily and appear to begin using it effectively almost immediately. Pop-up "tool tips" appear when a user mouses over a particular term type, explaining its purpose (as illustrated).



Level 3

Choose Equation Type: KCL (node currents)

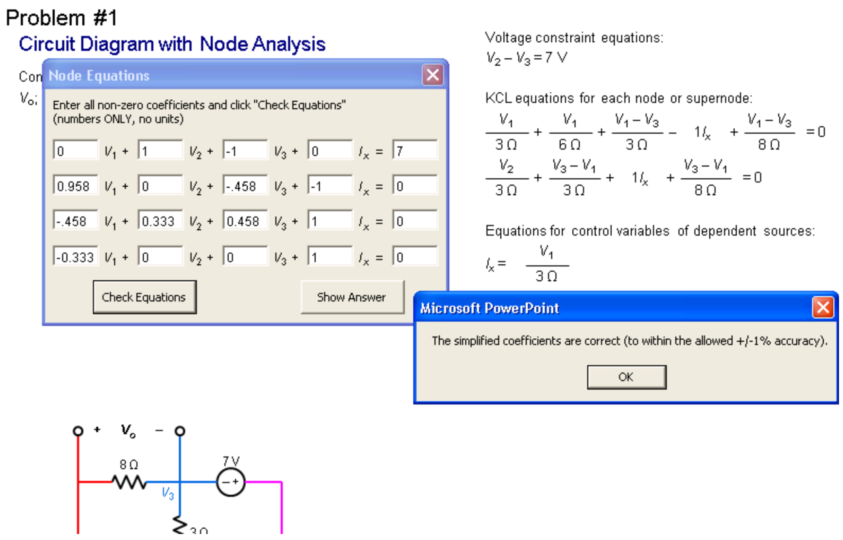
Time: -1:-01

Current through resistor that has neither end tied to groun

Fig. 2. Structured equation entry interface, illustrated for the case of node analysis. The middle row shows a palette of term types, some of which have been dragged onto the lower row where the equation is being formed and blanks are being filled in.

2.4. Simplified Equation and Matrix Equation Entry System

Once a full set of correct node or mesh equations has been constructed by the user, the next step in the solution process is to write those equations in simplified form (collecting coefficients of each node voltage or mesh current), and then to construct the corresponding matrix equation. The interface used for the entry of simplified equations is illustrated in Fig. 3. A similar interface (not shown) is then used to input the matrix equation. If correct, the entries are displayed on the slide as shown in Fig. 4. If incorrect, the user is given opportunities to revise them, or to display the correct answer if they give up.


Problem #1

Circuit Diagram with Node Analysis

Node Equations

Enter all non-zero coefficients and click "Check Equations"
(numbers ONLY, no units)

0	V_1 +	1	V_2 +	-1	V_3 +	0	I_x =	7
0.958	V_1 +	0	V_2 +	-.458	V_3 +	-1	I_x =	0
-.458	V_1 +	0.333	V_2 +	0.458	V_3 +	1	I_x =	0
-0.333	V_1 +	0	V_2 +	0	V_3 +	1	I_x =	0

Voltage constraint equations:

$V_2 - V_3 = 7\ \mathrm{V}$

KCL equations for each node or supernode:

$$\frac{V_1}{3\,\Omega} + \frac{V_1}{6\,\Omega} + \frac{V_1 - V_3}{3\,\Omega} - 1I_x + \frac{V_1 - V_3}{8\,\Omega} = 0$$

$$\frac{V_2}{3\,\Omega} + \frac{V_3 - V_1}{3\,\Omega} + 1I_x + \frac{V_3 - V_1}{8\,\Omega} = 0$$

Equations for control variables of dependent sources:

$$I_x = \frac{V_1}{3\,\Omega}$$

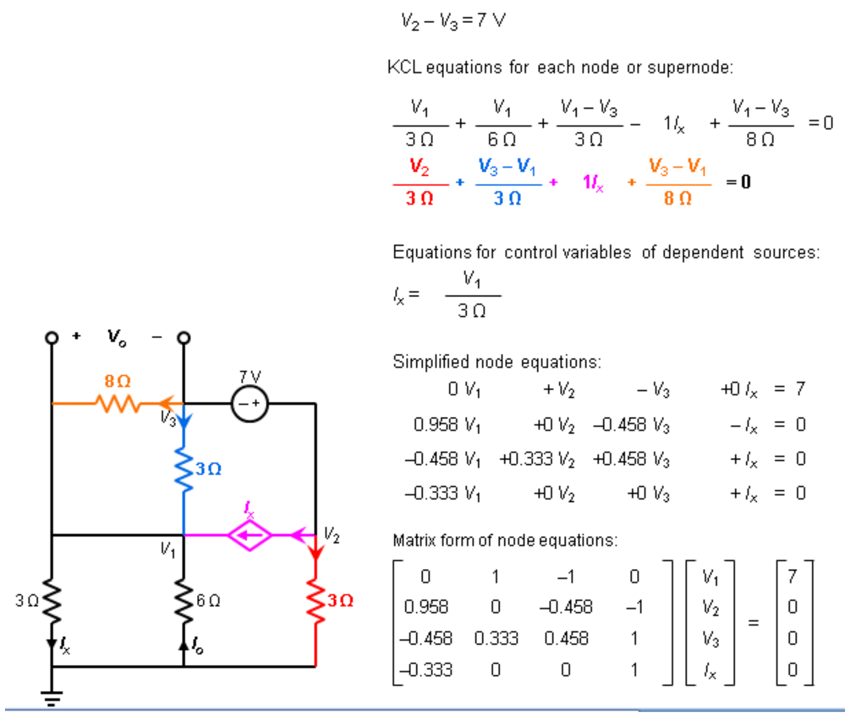
Microsoft PowerPoint

The simplified coefficients are correct (to within the allowed +/-1% accuracy).

Fig. 3. Simplified equation entry interface.

2.5. New Pedagogical Features

In addition to the pedagogical features described previously,[1] such as color coding of nodes, optionally "erasing" the circuit elements to make nodes clearer, and optional highlighting of a selected set of series or parallel elements in red, the program can label currents leaving a node or supernode with colored arrows as shown in Fig. 4. The terms corresponding to each of these currents are then color-coded in the selected equation to match, so that the origin of the equation is made clear. A similar facility is used to color code the terms in a KVL equation in mesh analysis for a selected mesh or supermesh. In this case (not shown), colored + and – signs are placed by each element in the loop for which a KVL equation is being written, and a dotted line is used to demarcate the loop itself.


$V_2 - V_3 = 7\ \mathrm{V}$

KCL equations for each node or supernode:

$$\frac{V_1}{3\,\Omega} + \frac{V_1}{6\,\Omega} + \frac{V_1 - V_3}{3\,\Omega} - 1I_x + \frac{V_1 - V_3}{8\,\Omega} = 0$$

$$\frac{V_2}{3\,\Omega} + \frac{V_3 - V_1}{3\,\Omega} + 1I_x + \frac{V_3 - V_1}{8\,\Omega} = 0$$

Equations for control variables of dependent sources:

$$I_x = \frac{V_1}{3\,\Omega}$$

Simplified node equations:

$$\begin{array}{rrrrl} 0\,V_1 & +V_2 & -V_3 & +0\,I_x & = 7 \\ 0.958\,V_1 & +0\,V_2 & -0.458\,V_3 & -I_x & = 0 \\ -0.458\,V_1 & +0.333\,V_2 & +0.458\,V_3 & +I_x & = 0 \\ -0.333\,V_1 & +0\,V_2 & +0\,V_3 & +I_x & = 0 \end{array}$$

Matrix form of node equations:

$$\begin{bmatrix} 0 & 1 & -1 & 0 \\ 0.958 & 0 & -0.458 & -1 \\ -0.458 & 0.333 & 0.458 & 1 \\ -0.333 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} V_1 \\ V_2 \\ V_3 \\ I_x \end{bmatrix} = \begin{bmatrix} 7 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

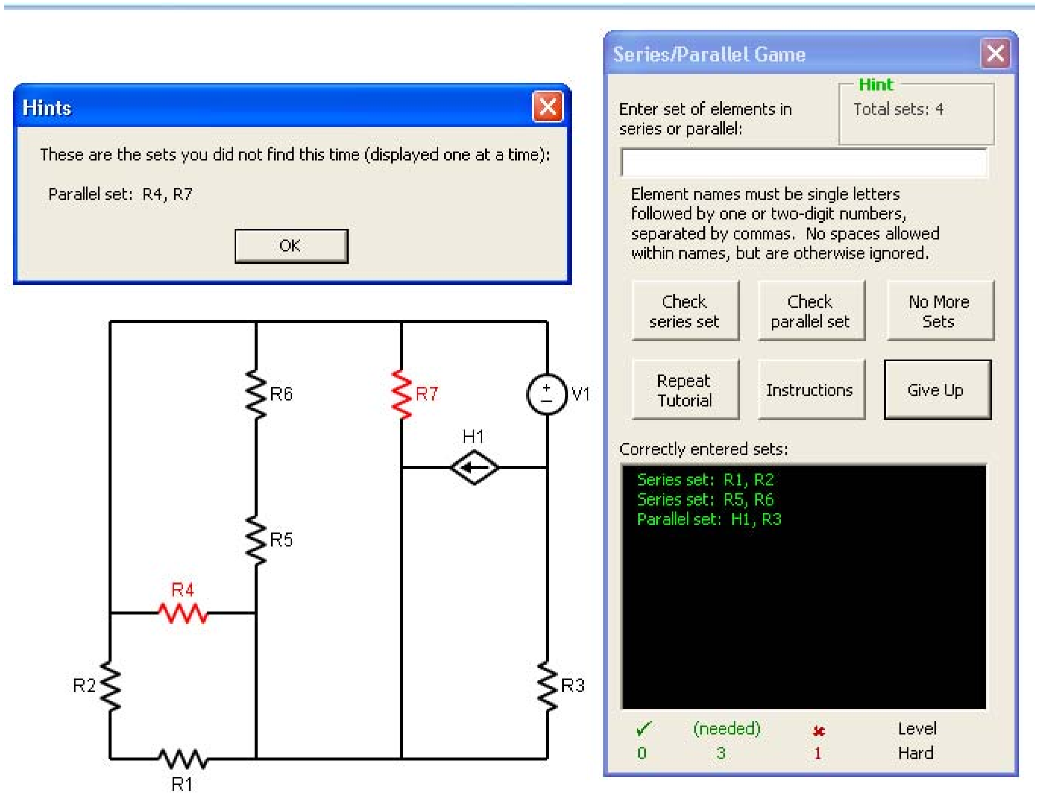
Fig. 4. Display of successfully entered simplified node equations and corresponding matrix equation. Highlighting of a selected KCL equation and labeling of the currents leaving a selected supernode with arrows that are color-coded to match the color of each term in the equation is also illustrated. The voltage V_o (for which terminals are automatically displayed) and the current I_o are the "sought quantities" the student is asked to compute in this problem.

3. Tutorial Sequences

3.1. Identification of Series and Parallel Circuit Elements

The first tutorial constructed to date teaches students to identify elements of any type in series and parallel with each other. It begins with a series of brief discussions and illustrations of the relevant concepts, asking students multiple choice questions to test their understanding interactively at each stage. It is specifically designed to confront common misconceptions, such as thinking that elements must be geometrically parallel to be electrically parallel, or that two resistors can be in series even if a voltage source (or other element) is connected at their junction. Students often mistakenly assume that the current through a voltage source is zero, because it is only a “voltage source.” We also use questioning to emphasize that a short-circuited resistor can not be in series with another element, a fact that is often overlooked by students.

After the series of examples and questions is completed, the student is allowed to select examples or exercises at each of four levels of complexity. A typical problem at level 3 is illustrated in Fig. 5, where a student is being asked to list all elements in series and parallel in a given circuit. The list of elements is typed in by the student, and added to the list of those found if correct. At easier levels of difficulty (not shown), the nodes are color-coded to assist the student, but this “assistive device” and other hints (such as the number of sets to find) are gradually turned off as the student progresses to harder levels. If a set is incomplete (e.g., only


Hints

These are the sets you did not find this time (displayed one at a time):

Parallel set: R4, R7

Series/Parallel Game

Hint: Total sets: 4

Enter set of elements in series or parallel:

Element names must be single letters followed by one or two-digit numbers, separated by commas. No spaces allowed within names, but are otherwise ignored.

Correctly entered sets:

Series set: R1, R2
Series set: R5, R6
Parallel set: H1, R3

✓	(needed)	✗	Level
0	3	1	Hard

Fig. 5. Sample exercise in the series/parallel tutorial at the “Hard” level (level 3). The student has successfully identified three sets (listed in green) and gave up, unable to find the fourth set, which is now illustrated for them.

two of three parallel elements are listed), the student is so advised and allowed to complete the set. After completing three exercises at a given level of difficulty without errors, the student is allowed to advance to the next level. If they give up twice, they are required to go back to an easier level. Three exercises must be completed at the hardest level to finish the tutorial. An unlimited number of examples may be viewed at any level of difficulty without penalty, where each series or parallel set is successively highlighted as shown in Fig. 5. Nearly all students were able to complete the tutorial successfully. Throughout all of our tutorials, the student actions and correct/incorrect answers (including the time required for each) are recorded to a log file for later analysis.

3.2. Writing Node Equations

The second tutorial focuses on writing node equations using the structured equation entry system illustrated in Fig. 2 above. An unlimited number of fully-worked examples and exercises is available at each of five levels of difficulty. In the examples, the equation terms are color-coded to match arrows representing the currents leaving each node or supernode, as shown in Fig. 4 above. The first level includes only current sources and resistors, the second level introduces voltage sources tied to ground, the third level introduces a floating voltage source (and hence requires a supernode), the fourth level requires two supernodes, and the fifth level introduces dependent sources in addition to the other features. Students must write the full set of node equations correctly for at least one circuit at each level (starting at any level up to level 3) to advance to the next level, and must complete all five levels. Again virtually all students were able to complete the tutorial.

3.3. Writing Mesh Equations

The third tutorial teaches students to write mesh equations, in a fashion very similar to the node equation tutorial above. A structured equation entry system is again used. In the unlimited number of worked examples, mesh currents are labeled, supermesh paths (if needed) are represented as dotted lines, and each equation has color-coded terms to match labeled voltage drops around one of the meshes or supermeshes (displayed successively for each mesh and supermesh). Five difficulty levels are again used. The first level involves only resistors and voltage sources, the next level adds external current sources, the third level adds shared internal current source (and therefore requires a supermesh), and so forth. The requirements are the same as for the node equation tutorial.

4. Preliminary Utilization and Assessment

4.1. Utilization in Courses

The series/parallel, node equation, and mesh equation tutorials have thus far been used on a voluntary basis (with extra credit or homework points in some cases as an incentive) by 51, 47, and 25 students (out of a total of 134 students in the relevant sections) in an introductory linear circuit analysis course at Arizona State University in summer 2012 and fall 2012. (The lower utilization for the mesh tutorial is because it was not completed in time for the summer session.) The effect of the two tutorials on student performance during the summer session could not be determined because the series-parallel tutorial was not completed by most students until after the

Table 1. Comparison of Course Scores for Students Using or Not Using Tutorials on a Voluntary Basis.

Average Scores in One Section (Fall 2012)					
# of tutorials completed	Midterm #1	Midterm #2	Final Exam	Course Score	*N*
3	71%	66%	84%	82%	21
0-2*	59%	52%	59%	65%	7
Pooled Std. Dev.	14%	15%	18%	10%	28

*Was 0 in 5/7 cases; results similar if cases of 1 or 2 completed excluded.

first mid-term examination (which is most closely related to this topic), and the node analysis tutorial was not available until after the second mid-term. The final examination had no questions on DC circuit analysis, the subject of the tutorials. In two of the three fall sections where the tutorials were offered, only a very small number of students completed them (only 2-3 students/section completed all three). In the third fall section, participation was much higher, but the number of students not using them (the control group) was relatively small, as shown in Table I. The results suggest a possible improvement in performance for those who used all three tutorials, but are not considered very significant due to the small size of the control group and possible effects of self-selection bias.

Student satisfaction with the tutorials was generally high, given that they used a preliminary version with an equation entry interface that was more difficult to use than that described above. Students were asked to rate the tutorials as “very useful,” “somewhat useful,” “not very useful,” or “a waste of time,” and 90% rated the series/parallel tutorial as either “very” or “somewhat” useful in Fall 2012. Corresponding ratings were 82% and 83% for the node and mesh analysis tutorials, and ratings in Summer 2012 were similar. A majority generally rated them as “very useful.”

Much more extensive utilization is planned in Spring 2013, when the tutorials will be used on a mandatory basis by about 360 engineering students in five sections of the introductory circuits class at Arizona State University (EEE 202) by five different instructors, by 23 students in the corresponding course at Chandler-Gilbert Community College (by Prof. Bassam Matar), by 70 students in spring and summer sessions at the University of the Pacific (by Prof. Jennifer Ross), and possibly others. Comparisons will be made to prior sections taught by the same faculty members using similar examinations.

4.2. Controlled Laboratory Trial

To obtain a better measure of the impact of our software on student learning, we conducted a randomized, controlled laboratory-based trial in December, 2012. Paid student volunteers were solicited from students who were enrolled in EEE 202 at Arizona State University in Fall 2012, or who had completed that course in the past year. The 33 students were each given a written pre-test and a post-test, each lasting 25 minutes and covering the topics of identifying series & parallel elements in a circuit, and writing node equations for DC resistive circuits. Two different tests A and B were used, designed to be very similar and of similar difficulty, and students were randomly assigned take either test A or test B as a pre-test, and the other test as a post-test. The average scores on the two tests were found to be 63.7 and 68.8% for tests A and B, respectively. The difference in difficulty was traced almost entirely to questions involving identification of

Table II. Results of Randomized Laboratory Study Comparing Textbook-Based Homework Problem to Software Usage.

	Exptl. Group	Pre-Test Score	Post-Test Score	Gain	Learning Gain*
Average	Textbook	58.6	61.6	2.9	7%
Median	Textbook	60.5	67.0	1.5	*N* = 16
Std. Dev.	Textbook	25.3	28.0	14.1	
Average	Software	57.8	86.4	28.6	68%
Median	Software	57.0	85.0	30.0	*N* = 17
Std. Dev.	Software	22.1	11.5	14.9	
Std. Dev.	Pooled	23.0	20.5	14.1	*N* = 33

*Defined as actual gain divided by maximum possible gain, based on the pre-test score. *N* is the number of students in each group.

series and parallel elements, where test A had more sets of elements in parallel and test B had more sets of elements in series, and identification of elements in parallel is apparently more difficult for students. This difficulty should be averaged out by our random assignment procedure.

Between the pre-test and the post-test, students were randomly assigned either to work individually on a list of end-of-chapter problems from the Irwin textbook we use,[2] or to use our software for the same period of time. All students were provided with a hard copy of the textbook and instructed that they were free to review any material in the relevant sections as needed, which were identified for them. They were required to work for 25 minutes on series and parallel elements, and for 35 minutes on node analysis. The assigned node analysis problems in the textbook were selected to be similar to the ones in our software, except that they requested full algebraic and numerical solutions, whereas the software only asks for the relevant equations to be written. Since the textbook does not have any exercises specifically addressing the identification of series and parallel circuit elements, we asked students to work textbook exercises where they are required to combine resistors in series or parallel, which exercises the same skill. The textbook does not provide answers to its problems.

The overall results are summarized in Table II. The average gain from pre-test to post-test is about 10× higher for the software users than for the textbook users. In terms of grades, using a typical grading scale of 90-100% = A, 80-89% = B, 70-79% = C, and 60-69% = D, the textbook users went from a high E to a low D. The software users, however, went from a high E to a solid B grade. Note that all students in this trial had already received conventional instruction in these topics via lectures and conventional homework assignments, though it is clear that most of them either did not learn or did not retain the material very well. The effect size is calculated to be a Cohen's *d*-value of 1.21 pooled standard deviations, where the pooled standard deviation is defined as

$$\sigma(pooled) = \sqrt{\frac{(n_1 - 1)\sigma_1^2 + (n_2 - 1)\sigma_2^2}{n_1 + n_2}} ,$$

where σ_1 and σ_2 are the sampled standard deviations of the post-test scores of the experimental and control groups, and n_1 and n_2 are the number of students in each group. This effect size is considered to be quite large. An independent samples *t*-test confirmed a statistically significant difference at the 95% level in the post-test scores for the students who worked on the book

Table III. Results of Laboratory-Based Comparison, Broken Down by Topic Area

Topic	Exptl. Group	Pre-Test Avg.	Post-Test Avg.	Gain
Series/ Parallel	Textbook	72%	68%	-4%
Series/ Parallel	Software	71%	91%	20%
Node Equations	Textbook	49%	57%	8%
Node Equations	Software	49%	83%	34%

problems versus the score gains for the students who used the computer software; $t(19.7)=3.303$ without assuming equal variances.

We also evaluated the effect of the tutorials on the two separate topics, as summarized in Table III. Note that identification of series and parallel elements is a typical qualitative skill required in this course, whereas writing node equations is a quantitative or mathematical skill. The results show a higher gain in the quantitative topic than for the qualitative one, but the difference may well be because of the higher pre-test scores in the latter case. Both topics saw large gains in student performance. In particular, student scores on the easier node analysis problem on the pre- and post-tests (involving only DC current sources and resistors, with four nodes and three KCL equations) increased from 59% to 98% for the software users (and from 57% to 70% for textbook users), indicating that the software leads to an almost perfect mastery of this topic. During in-class testing in Fall 2012, students needed an average (median value) of 47 (36) minutes to complete the series/parallel tutorial and 43 (39) minutes to complete the node equation tutorial, so that they were likely not all able to complete them in the laboratory study. Higher gains might therefore be expected if a longer study time were allocated.

5. Conclusion

We have continued our development of a software system designed to generate and solve linear circuit analysis problems, which accepts a rich variety of student inputs. Three tutorials using the software have been implemented. Laboratory based studies show a statistically significant and large (~1.21 standard deviation) increase in student learning as a result. Further work will focus on expansion and completion of this system.

Acknowledgment

This work was supported by the National Science Foundation through the Transforming Undergraduate Education in Science, Technology, Engineering and Mathematics Program under Grant No. DUE-1044497. We thank Daniel Sayre of John Wiley & Sons, Inc. for providing the textbook copies used in our laboratory experiment.

References

[1]C. D. Whitlatch, Q. Wang, and B. J. Skromme, "Automated problem and solution generation software for computer-aided instruction in elementary linear circuit analysis," in *Proceedings of the 2012 American Society for Engineering Education Annual Conference & Exposition* (Amer. Soc. Engrg. Educat., Washington, D.C., 2012), p. Session M356.
[2]J. D. Irwin and R. M. Nelms, *Basic Engineering Circuit Analysis* (Wiley, Hoboken, NJ, 2010).