Abstract—A step-based tutoring system for linear circuit analysis is being developed with the capabilities to automatically generate circuit problems with specified characteristics, including randomly generated topologies and element values. The system further generates fully-worked, error-free solutions using the methods typically taught in such classes, and accepts a rich variety of student input such as equations, matrix equations, numerical and multiple-choice answers, re-drawn circuit diagrams, and sketches of waveforms. A randomized, controlled study was conducted using paid student volunteers to compare the effectiveness of two of our tutorials in comparison to working conventional textbook-based problems. The average learning gain was only 3/100 points for the textbook users, but 29/100 points, about 10 times higher, for the tutorial users. The effect size on the post-test scores was 1.21 pooled standard deviations (Cohen d-value) and was statistically significant. A motivational survey administered to these students yielded a 0.53 point higher rating for the software than for the textbook (on a 1-5 scale). The system is being used in Spring 2013 by over 340 students in EEE 202 at Arizona State and two community colleges. About 99% of these students rated the system as “very helpful” or “somewhat helpful.”

Keywords—linear circuit analysis; computer-aided instruction; step-based tutoring

I. INTRODUCTION

Linear circuit analysis is a foundational topic for electrical engineers, but is also widely studied by other engineering majors. For example, this course is taught to over 600 students a year at ASU in 11 sections, about 90% of whom are non-majors. Students frequently struggle with this material, due to factors as delayed or inaccurate feedback on their homework, inadequate use of active and cooperative learning strategies, an insufficient (from the student’s perspective) supply of worked examples of carefully graded difficulty, and failure on the part of instructors to recognize persistent misconceptions about basic electricity that students frequently bring into these classes [1-3]. Traditional lecture-based instruction offers only a single pace to all students, regardless of prior knowledge and ability, and inevitably leaves some students bored and others unable to keep up. One method to address these issues is to supplement conventional approaches with computer-aided instruction, which has been employed in a number of prior studies [4-35]. While useful, many of these studies have developed only incomplete or partial prototypes, or have not carried out rigorous evaluations to determine their effectiveness in increasing student learning. There has been little sustained, widespread usage of previously developed systems, and most have not had the ability to generate new circuit problems, which can be a tedious and error-prone task when done by hand.

Publisher-based web sites as supplements to textbooks have been more widely used in recent years. Whereas these sites provide additional resources to students and can be used to automate grading, they mainly provide algorithmic problems, in which some of the element values are varied in a given problem. They are typically answer-based tutors, which provide rapid feedback on the correctness of an answer, but do not accept a sufficient amount or variety of student input to diagnose the reason for a wrong answer (they lack sufficient “bandwidth” in the language of intelligent tutoring systems). A recent meta-analysis by VanLehn concludes that the typical effect size (Cohen’s d-value) of answer-based tutoring systems is around 0.31, compared to 0.76 for step-based tutors and 0.79 for expert (expensive) human tutors [36]. Thus, we have undertaken to develop a potentially more effective step-based system that can do a better job of analyzing a wide variety of student inputs [37, 38]. Further, we automate the process of generating problems and solutions, eliminating both human error that can result in great frustration to students and the temptation to misuse solution manuals that are widely available on the Internet. This type of system can provide rapid, accurate feedback and an unlimited supply of problems, examples, and solutions of any desired difficulty and complexity.

II. SOFTWARE DESIGN AND FEATURES

A. Circuit Generation

The algorithms we use to generate circuit problems and solutions have been described elsewhere in detail [37, 38]. We generate circuit layouts rather than netlists to ensure that our circuits are planar so that mesh analysis can be applied. Briefly,
the circuit generation process involves a series of three steps, to avoid the prohibitive combinatorial issues involved in trying to place circuit elements in a purely random way. We first generate a “topology” with the desired number of meshes, consisting of only shorts or opens placed on a square grid with the desired number of rows and columns. The generation algorithm ensures that it is fully connected and has no “dangling” shorts. It is checked to ensure that it is not “hinged” (i.e., it cannot be drawn so that two parts of the circuit are connected by only a single wire, and therefore constitutes effectively separate circuit problems). In the second step, the desired number of shorts are replaced by generic circuit elements, leaving the others as shorts and leaving all opens permanently as opens. In doing so we place at least two elements on every mesh including the “outer” mesh around the circuit periphery, to ensure that no element is shorted and that there are no meshes of shorts (which would reduce the true number of meshes below the desired value). We then check that this “populated topology” has not become hinged as a result of element placement (which includes shorted elements). If the result is not acceptable, the process is restarted.

In the third step, the generic elements are replaced with actual circuit elements of the desired types. To avoid insoluble (inconsistent) problems, we first find all or many trees of the populated topology. Voltage sources and inductors are placed exclusively on the twigs of a randomly selected tree, and current sources and capacitors are placed exclusively on the links of that tree, thereby avoiding insoluble problems involving loops of only voltage sources or stars consisting only of current sources (resistors can be placed anywhere). The placement algorithm also allows us to limit the number of voltage sources in series and the number of current sources in parallel, and can optionally prohibit passive elements of the same type in series or parallel with each other. Further, it optionally avoids creating problems where a given circuit element is “redundant” and has no significant impact on the rest of the circuit (such as any element in series with a current source or in parallel with a voltage source). The algorithm also has the ability to create a desired (feasible) number of floating supernodes (i.e., supernodes that do not include the reference node), through a combination of source repositioning and/or choice of specific reference nodes. Circuits can also be automatically selected having a desired number of supermeshes. The user also has the option to reject circuits whose node voltages are all controlled directly by voltage sources, or whose mesh currents are controlled directly by current sources, since such problems are relatively trivial to solve. Control variables for dependent sources are randomly selected in accordance with rules described elsewhere [37]. Element values including dependent source gains are then randomly generated within specified ranges, and the circuit is checked and modified as necessary to ensure that it is soluble.

To specify the type of circuit to be generated, the user specifies the number of squares in the grid in both $x$ and $y$-directions, the number of nodes, the number of meshes, and the number of each type of circuit element to be used. However, any one of the last three quantities is determined by the other two, so that the user has the choice of which two to specify [37].

Once a circuit has been generated, the program randomly selects a user-specified number and type of “sought quantities,” or values of unknowns for which the student is asked to solve. These may be branch voltages, branch currents, non-branch voltages (i.e., voltages that do not appear directly across any one circuit element), or branch powers. We avoid specifying quantities that are trivial to determine (such as a branch current for an element in series with an independent current source). Alternatively, the student can be required to solve for all node voltages or all mesh currents.

B. Solution Generation

The system currently generates and displays complete sets of node or mesh equations and solves them using matrix methods. Equations are automatically adjusted based on a user-selected choice of reference node. There is an optional “pre-simplification” step prior to this solution, in which independent voltage sources in series and independent current sources in parallel are combined into single sources of each type, and in which any passive elements of the same type that are in series or parallel with each other are combined. The circuit diagram can be automatically re-drawn after each such simplification step to illustrate the process to the student. Work is currently in progress to implement a number of other common solution methods, such as use of voltage and current division, superposition, source transformation, and use of Thévenin or Norton equivalent circuits. We also plan to extend the system from its current coverage of DC resistive circuits to cover steady-state AC phasor analysis and transients using both differential equation and Laplace transform approaches. A sample automatically-generated circuit and its solution by mesh analysis are shown in Fig. 1.

C. User Input Modules

Students input node or mesh equations using a specially designed template interface as shown in Fig. 2, in which they are offered a palette of properly formed terms from which to choose [depending on the type of equation, such as a voltage constraint equation, Kirchoff’s current law (KCL) equation, dependent source control variable equation, etc.]. They can drag these terms into the equation area as needed, and easily reposition or delete them. In Fig. 2, for example, the user has dragged the various terms needed for a KVL equation for mesh 3 from the upper palette into the equation entry area below, and is about to check the equation for correctness. Once the user selects the appropriate templates, they complete each term by filling in the numerical values and subscripts. Students appear to find this interface to be highly intuitive and easily adapt to using it with little or no instruction. The program then indicates immediately if the equation is correct, and optionally displays the correct solution if it was not. In the latter case, the student then has to work an additional problem of the same type to proceed.

Simplified forms of the algebraic equations are entered on a special form designed for that purpose, as is the matrix form of the equations. They are provided with feedback at each stage, so they do not waste time and become frustrated by proceeding to the next stage when they have already made a mistake (as
Some problems of a more qualitative nature are also being developed, to help dispel typical student misconceptions. An example is one in which a circuit diagram is displayed and students are asked to input lists of which elements are in series or in parallel in the circuit (or which form wye or delta connections). Students typically have difficulty in doing this at first, particularly for the parallel case, which can lead to many other errors in quantitative analyses.

D. Pedagogical Features

We have included a number of items to help students understand and visualize the methods by which circuits can be analyzed. For example, different nodes can optionally be color coded, to help visualize the nodes. The circuit elements can even be temporarily blanked out, leaving only the wires, to help students understand the definition of nodes. We also illustrate sets of elements in series or parallel (or forming wye or delta connections) by selectively highlighting each set in red in turn. Nodes and mesh currents can be automatically labeled and numbered. There are also features to help students understand the origin of node or mesh equations. For example, the currents leaving a selected node or supernode can be automatically labeled with color-coded arrows, in which case the terms of the KCL equation are correspondingly color-coded to match. Similarly, the path around which a KVL equation is written for a supermesh can be labeled automatically, and voltage drops around any selected mesh or supermesh can be labeled with color-coded +/- signs, with corresponding color-coding of the terms in the KVL equation. In Fig. 1, for example, the first KVL equation is the one that has been selected for highlighting, so the supermesh path and +/- signs appear around meshes 1 and 3, which form a supermesh. (In this view, the unknown voltage $V_o$ is not shown.)

E. Tutorial Modules

To date we have implemented three software tutorial sequences on the topics of identifying elements in series and parallel.
parallel and on writing node and mesh equations. The series/parallel tutorial includes a sequences of illustrations and examples with associated interactive questions, followed by the option to view examples and complete exercises at four different levels of difficulty. The node and mesh equation tutorials present a brief set of instructions, followed by the opportunity to view worked examples and do exercises at five different levels. Student activity in each tutorial is logged to a server to record their progress, allowing them to stop or re-start at any time. The results can be viewed and downloaded by the instructor. Students are required to complete the highest level of difficulty in each tutorial before receiving full credit. At the completion of each tutorial, a one-question survey is administered and the student can enter comments to provide feedback.

We are also developing scripts for future tutorial sequences, which will be implemented later in software using a planned tutorial scripting and execution interface. A total of 14 scripts have been generated to date, including 5 on node analysis, 5 on mesh analysis, and 4 on basic electrical concepts such as current, voltage, power, KCL, KVL, etc. The scripts outline sequences of instructional material together with interactive questions and exercises that will be posed to the students.

### III. LABORATORY-BASED STUDY

To test the effectiveness of our software in improving student learning, a laboratory-based study was conducted in December 2012 using 33 paid student volunteers, all of whom were either enrolled in the relevant course (EEE 202) or had completed it within the last year. Students were given a pre-test and a post-test using two different test forms that were randomly assigned and similar in difficulty. The two topics were identification of series and parallel circuit elements and writing node equations for DC resistive circuits. The students were randomly assigned either to use two of the corresponding software tutorials for a total of one hour (for 25 min. and 35 min., respectively), or to work on similar textbook problems, as they would normally do in the course. All students were given a copy of the course textbook [39] and advised that they could consult any relevant material in it as needed. After completing the post-test, both groups of students were asked to complete the Instructional Materials Motivation Survey (IMMS) developed by Keller to assess the effects of the different instructional approaches on student motivation [40].

The pre-test and post-test data for both groups are summarized in Table I. Both groups had very similar pre-test scores (averages of 59% and 58%), as would be expected from the randomized experimental design. The textbook users had only a small average increase in average score up to 62%, about 3 points higher. The software users saw an approximately 10× larger improvement, to an average of 86% on the post-test. The effect size (Cohen $d$-value) of the experimental condition on the post-test scores was found to be 1.21 in units of the pooled standard deviation of those scores. The difference was statistically significant assuming independent samples (but not equal variances, as they are substantially different) at the 95% confidence interval, $t(19.7) = 3.303, p < 0.05$. We therefore conclude that the tutorials are much more effective than conventional homework assignments in terms of immediate learning gains. We attribute the difference to the step-based nature of the tutoring process, the rapid feedback provided to the student, their ability to practice or view examples as many times as needed (within the available study time), and the pedagogical features that help them learn to understand the formation and structure of the equations.

The results were better for both the qualitative (series-parallel identification) and quantitative (node equation) topic, suggesting that this approach may be effective for a wide variety of course topics when fully developed. The mean post-test scores for the series-parallel identification were 68% and 91% for textbook and computer users, respectively, and were 57% and 83% for the node equation topic for the two corresponding groups. In particular, the easier node analysis problem (having only independent current sources and resistors, with four nodes) on the post-test yielded a remarkable average score of 98% for the computer users (and only 70% for textbook users), suggesting near-perfect mastery of that level of the topic after only 35 minutes of automated instruction. It is unlikely that most students were able to complete the tutorials during the allotted time, so results on more difficult problems could be even better given more time.

The results of the IMMS are shown in Table II. Overall scores were determined (where 1=least favorable and 5=most favorable response) as well as scores on four subscales corresponding to the attention-relevance-confidence-satisfaction (ARCS) factors proposed by Keller to model the effects of instructional materials on student motivation [40]. The mean score was higher for the software on every scale, though the difference was statistically significant only for the total scores and attention and satisfaction subscales. The relevance factor showed the least difference, which makes sense as both the textbook and tutorials addressed the same topics and the students seem to be making appropriate connections between the practice problems and real-world engineering applications in both cases. The degree to which the software engaged students’ attention and especially their overall satisfaction were the biggest differences (the latter being nearly a full point higher for the software users, with a large effect size of 1.27 pooled standard deviations). We believe that it was the ability to repeat exercises and examples without limitation for practice, and the rapid feedback provided on student errors, that led to the higher satisfaction rating. The Cronbach alpha coefficients we measured for the total score,

<table>
<thead>
<tr>
<th>Exptl. Group</th>
<th>Pre-Test Score</th>
<th>Post-Test Score</th>
<th>Gain</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average</td>
<td>Textbook</td>
<td>58.6</td>
<td>61.6</td>
</tr>
<tr>
<td>Median</td>
<td>Textbook</td>
<td>60.5</td>
<td>67.0</td>
</tr>
<tr>
<td>Std. Dev.</td>
<td>Textbook</td>
<td>25.3</td>
<td>28.0</td>
</tr>
<tr>
<td>Average</td>
<td>Software**</td>
<td>57.8</td>
<td>86.4</td>
</tr>
<tr>
<td>Median</td>
<td>Software</td>
<td>57.0</td>
<td>85.0</td>
</tr>
<tr>
<td>Std. Dev.</td>
<td>Software</td>
<td>21.1</td>
<td>11.5</td>
</tr>
<tr>
<td>Std. Dev.</td>
<td>Pooled</td>
<td>23.0</td>
<td>20.5</td>
</tr>
</tbody>
</table>

*16 users. **17 users.
TABLE III. SAMPLE STUDENT COMMENTS ON TUTORIALS

| Good job on the game! It was actually fun going through it and trying to do a good job! Thanks for making this. |
| Worked as intended, didn't take too long, kind of fun, and I feel like it helped! |
| I HAVEN'T EVEN LEARNED IT YET BUT IT WAS REALLY EASY TO GRASP USING THIS! YAY |
| I really thought it was awesome; it was very helpful. I understood the concepts, but this helped me develop a thought process on it. |
| I like how you are not marked off for getting on wrong, you just get to try again. You only really fail if you give up, and that is reassuring. |
| These modules honestly do help me learn circuit analysis. I feel that it is extremely helpful to have a good amount of practice problems, and a system that provides instant feedback. This helps me learn the correct techniques and master |
| I AM A PRO AT THIS. Major self-confidence booster. Really though, I feel like I'm talented at this node analysis! |
| It definitely helped me understand supernodes, I think this was more useful than book work |
| This exercise helped me understand loop analysis very well. The assignment was great. |
| I would prefer to have a statistics page showing # of correct and incorrect attempts and possibly even a ladder [leader?] board showing how well different students did as opposed to everyone getting a congratulatory gold medal for doing their hw |
| Wow is all i can say... This is the best, better than any hw I have done so far |

TABLE II. RESULTS OF INSTRUCTIONAL MATERIALS MOTIVATION SURVEY (SCALE = 1-5, 5=BEST)

<table>
<thead>
<tr>
<th>Group</th>
<th>Statistic</th>
<th>Total</th>
<th>Attention</th>
<th>Relevance</th>
<th>Confidence</th>
<th>Satisfaction</th>
</tr>
</thead>
<tbody>
<tr>
<td>Software Users</td>
<td>Means</td>
<td>3.54</td>
<td>3.44</td>
<td>3.22</td>
<td>3.94</td>
<td>3.62</td>
</tr>
<tr>
<td></td>
<td>Std. Dev.</td>
<td>0.40</td>
<td>0.49</td>
<td>0.60</td>
<td>0.52</td>
<td>0.66</td>
</tr>
<tr>
<td></td>
<td>Medians</td>
<td>3.57</td>
<td>3.54</td>
<td>3.11</td>
<td>3.83</td>
<td>3.75</td>
</tr>
<tr>
<td>Textbook Users</td>
<td>Means</td>
<td>3.01</td>
<td>2.84</td>
<td>2.99</td>
<td>3.51</td>
<td>2.65</td>
</tr>
<tr>
<td></td>
<td>Std. Dev.</td>
<td>0.77</td>
<td>0.80</td>
<td>0.83</td>
<td>0.99</td>
<td>0.91</td>
</tr>
<tr>
<td></td>
<td>Medians</td>
<td>3.01</td>
<td>2.88</td>
<td>3.00</td>
<td>3.72</td>
<td>2.33</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Comparisons</th>
<th>Diff. of Means</th>
<th>0.53*</th>
<th>0.60*</th>
<th>0.23</th>
<th>0.44</th>
<th>0.97*</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Pooled Std. Dev.</td>
<td>0.58</td>
<td>0.64</td>
<td>0.70</td>
<td>0.75</td>
<td>0.76</td>
</tr>
<tr>
<td></td>
<td>Cohen d-value</td>
<td>0.91*</td>
<td>0.94*</td>
<td>0.33</td>
<td>0.58</td>
<td>1.27*</td>
</tr>
</tbody>
</table>

*Statistically significant difference with $p < 0.05$.

IV. CLASSROOM TRIALS

Preliminary trials were conducted on a voluntary basis in Summer 2012 and Fall 2012 in our course EEE 202, but sample sizes (in either the control group or the experimental group, depending on the class section) were not large enough to make statistically significant evaluations of the effects of the software (though favorable effects were suggested by the data).

A much larger trial with strongly encouraged or mandatory participation is being conducted in Spring 2013 with over 340 students in five sections with five different instructors. Assessment of the impact of the software on student learning is in progress, but of course it is challenging to isolate the effects from variations in instruction and students from semester to semester. However, student satisfaction with the tutorials appears to be quite high. At the completion of each tutorial, students were asked to rate the tutorial as “very helpful,” “somewhat helpful,” “not very helpful,” or “a waste of time,” and were given the opportunity to enter comments. About 98.5% of students gave favorable ratings of “very helpful” or “somewhat helpful,” and 74% rated them as “very helpful.” Some sample student comments (verbatim) are shown in Table III. Most negative comments (relatively uncommon) asked for better instructions on the user interface (which we will add), or complained about platform compatibility, as the program currently runs only on Windows with Microsoft PowerPoint installed. A future web-based version is planned to address that issue. Some students requested more detailed feedback on their errors, which could be added in a later version.

Some limitations are that the topical coverage is not yet large enough to have a major impact on the overall class (though additional development is in progress), and retention of material from the time of the tutorial completion until the time of exams may be an issue. Refinement of the tutorial to require regular “refresher” exercises is planned to try to address retention. The DIRECT concept inventory [1] is being used as a pre- and post-test, but post-test results for Spring 2013 are not yet available. Additional analysis of the effect on student grades is planned, as is additional usage in future semesters. Ultimately we hope to assess if the students can transfer the knowledge gained using this software to work in subsequent courses and to real-world engineering applications.

V. CONCLUSIONS

Our progress in developing a system to automate the generation of problems and solutions for linear circuit analysis has been described, as well as initial incorporation of these modules into a tutorial system that can be used for homework assignments in this type of course. The rapid feedback characteristic of a step-based tutor, pedagogical features, and unlimited practice opportunities appear to be very popular with students and increase learning in laboratory-based trials compared to conventional textbook exercises by a factor of about 10. Additional work is needed to expand the scope of material covered by the system, and to refine the user interface and platform compatibility. The modular system is designed to be very flexible, so that it could be used, for example, to generate homework problems and solution manuals for a conventional textbook, to automate the generation and grading of individually customized homework assignments, to generate problems for examinations and quizzes, and to create problems.
for interactive in-class exercises. The overall approach of this system may be extendable to a number of other domains in engineering education as well where students must solve problems of well-defined types, such as logic circuit design, statics, and so forth.

ACKNOWLEDGMENTS

We thank Daniel Sayre of John Wiley & Sons, Inc. for providing the textbook copies used in our laboratory study and for his support. We thank Professors R. Gorur, G. Karady, B. Matar, C. Tepedelenliglu, T. Thornton, Chao Wang, Hongbin Yu, and Hongyu Yu for using our software in their EEE 202 sections and for their interest in this project. We also thank P. Selveraj and S. Ramalingam for their assistance.

REFERENCES


